Alignment of Organizational Security Policies

Theory and Practice

Trajče Dimkov

Alignment of Organizational Security Policies

Theory and Practice

Trajce Dimkov

Composition of the Graduation Committee:

Prof. Dr. Ir.	A.J. Mouthaan	Universiteit Twente
Prof. Dr.	P.H. Hartel	Universiteit Twente
Prof. Dr.	R.J. Wieringa	Universiteit Twente
Prof. Dr.	M. Junger	Universiteit Twente
Prof. Dr.	D. Gollmann	Hamburg University of Technology
Dr.	C.W. Probst	Technical University of Denmark
Prof. Dr.	E.R. Verheul	Radboud Universiteit Nijmegen

This research is supported by the Sentinels program of the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs under projects number TIT.7628.

CTIT Ph.D. Thesis Series No. 12-218
Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE
Enschede, The Netherlands.

IPA Dissertation Series No. 2012-04
The research reported in this thesis has been carried out under the auspices of IPA, the Dutch Research School for Programming research and Algorithmics.

ISBN : 978-90-365-3331-7
ISSN : 1381-3617 (CTIT Ph.D.-thesis series No. 12-218)
DOI number : 10.3990/1.9789036533317
Official URL: http://dx.doi.org/10.3990/1.9789036533317

Typeset with LaTeX. Cover photo: Dragan Siskov.

ALIGNMENT OF ORGANIZATIONAL SECURITY POLICIES
THEORY AND PRACTICE

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on Thursday, 23rd of February 2012 at 14:45

by

Trajce Dimkov

born on 20th of June 1983,
in Kavadarci, Macedonia

The dissertation is approved by:

Prof. Dr. P.H. Hartel (promotor)

Abstract

To address information security threats, an organization defines security policies that state how to deal with sensitive information. These policies are *high-level policies* that apply for the whole organization and span the three security domains: physical, digital and social. One example of a high-level policy is: ”The sales data should never leave the organization.” The high-level policies are refined by the Human Resources (HR), Physical Security and IT departments into implementable, *low-level policies*, which are enforced via physical and digital security mechanisms and training of the employees. One example of low-level policy is: ”There should be a firewall on every external-facing system”.

The erroneous refinement of a high-level policy into a low-level policy can introduce design weaknesses in the security posture of the organization. For example, although there is a low-level policy that places firewalls on every external-facing system, an adversary may still obtain the sales data through copying it on a USB stick. In addition, the erroneous enforcement of a low-level policy using a specific security mechanisms may introduce implementation flaws. For example, although there might be a firewall on every external-facing system, the firewall might not be configured correctly. The organization needs assurance that these errors are discovered and mitigated.

In this thesis we provide methods for testing whether (a) the high-level policies are correctly refined into low-level policies that span the physical, digital and social domain, and (b) whether low-level policies are correctly enforced is specific mechanisms. Our contributions can be summarized as follows:

1. We propose a formal framework, Portunes, which addresses the correct refinement of high level policies by generating attack scenarios that violate a high-level policy without violating any low-level policies. Portunes binds the three security domains in a single formalism and enables the analysis of policies that span the three domains. We provide a proof of concept implementation of Portunes in a tool and polynomial time algorithms to generate the attack scenarios.

2. We propose a modal logic for defining more expressive high-level policies. We use the logic to express properties of Portunes models and model evolutions formally. We provide a proof of concept implementation of the logic in the Portunes tool.

3. We propose two methodologies for physical penetration testing using social engineering to address the correct enforcement of low-level policies. Both methodologies are designed to reduce the impact of the test on the employees and on the personal relations between the employees. The methodologies result in a more ethical assessment of the implementation of security mechanisms in the physical and social domain.

4. We provide an assessment of the commonly used security mechanisms in reducing laptop theft. We evaluate the effectiveness of existing physical and social security mechanisms for protecting laptops based on (1) logs from security guards regarding laptop thefts that occurred in a period of two years in two universities in the Netherlands, and (2) the results from more than 30 simulated thefts using the methodologies in contribution 3. The results of the assessment can aid in reducing laptop theft in organizations.

5. We propose a practical assignment of an information security master course where students get practical insight into attacks that use physical, digital and social means. The assignment is based on the penetration testing methodologies from contribution 3. The goal of the assignment is to give a broad overview of security to the students and to increase their interest in the field. Besides for educational purposes, the assignment can be used to increase the security awareness of the employees and provide material for future security awareness trainings.

Using these contributions, security professionals can better assess and improve the security landscape of an organization.

Samenvatting

Om informatiebeveiligingsrisico's het hoofd te bieden, stellen organisaties een beveiligingsbeleid op hoofdlijnen op, dat bepaalt hoe omgegaan dient te worden met gevoelige informatie. Dit beleid is geldig voor de gehele organisatie en heeft betrekking op drie beveiligingsdomeinen: fysiek, digitaal en sociaal. Een voorbeeld van dergelijk beleid is "Verkoopgegevens mogen nooit buiten de organisatie komen." Het beleid wordt door de afdelingen van Personeel en Organisatie (P&O), IT en fysieke beveiliging verder uitgewerkt in gedetailleerde beveiligingsregels, die worden afgedwongen door fysieke en digitale beveiligingsmechanismen, en door training van medewerkers. Een voorbeeld van zo'n regel is "Elk van buiten toegankelijk systeem moet een firewall hebben."

Fouten die optreden bij de vertaling van het beleid naar concrete regels of van regels naar specifieke beveiligingsmechanismen, kunnen het beveiligingsniveau van de organisatie aantasten. Alhoewel er een regel is die firewalls verplicht stelt, kan een aanvaller bijvoorbeeld toch de verkoopdata verkrijgen door deze op een USB stick te kopiren. Bovendien kunnen er in de handhaving van de gedetailleerde regels implementatiefouten zitten. Zo kan de firewall wellicht onjuist geconfigureerd zijn. Organisaties moeten daarom de zekerheid hebben dat deze fouten ontdekt en gerepareerd worden.

In dit proefschrift ontwikkelen we methoden om te testen of (a) het beveiligingsbeleid op correcte wijze is uitgewerkt in beveiligingsregels (fysiek, digitaal en sociaal) correct is, en (b) deze regels op correcte wijze gehandhaafd worden door beveiligingsmechanismen. Onze bijdragen zijn als volgt samen te vatten:

1. We introduceren een formeel raamwerk, Portunes, dat onderdeel (a) uitwerkt door aanvalsscenario's te genereren die het beveiligingsbeleid overtreden, zonder daarbij de gedetailleerde beveiligingsregels te doorbreken. Portunes kan de drie beveiligingsdomeinen in n model representeren, en de bijbehorende beveiligingsregels analyseren. We beschrijven een proof-of-concept implementatie van Portunes in een tool en algoritmen die in polynomische tijd aanvalsscenario's genereren.

2. We presenteren een modale logica voor het definiren van geavanceerder beveiligingsbeleid op hoofdlijnen. We gebruiken deze logica om eigenschappen van Portunes modellen en hun evoluties formeel uit te drukken. We presenteren tevens een proof-of-concept implementatie van deze logica in de Portunes tool.

3. We stellen twee methoden voor om on-site penetratietesten uit te voeren gebruikmakend van social engineering, als uitwerking van onderdeel (b). Beide methodologien zijn ontwikkeld om de impact van de testen op de medewerkers en hun onderlinge relaties zo veel mogelijk te beperken, en daarmee een meer verantwoorde beoordeling van de implementatie van beveiliging in het fysieke en sociale domein mogelijk te maken.

4. We presenteren een evaluatie van de meestgebruikte beveiligingsmechanismen om laptopdiefstal te reduceren. We evalueren de effectiviteit middels de analyse van (1) rapporten van beveiligingsmedewekers met betrekking tot laptopdiefstallen die hebben plaatsgevonden in een periode van twee jaar bij twee Nederlandse universiteiten, en (2) de resultaten van meer dan 30 gesimuleerde laptopdiefstallen op basis van de methoden van bijdrage 3. De resultaten kunnen helpen om laptopdiefstal in de betreffende organisaties te beperken.

5. We presenteren een opdracht in de context van een mastervak informatiebeveiliging, waarin studenten praktische inzichten verkrijgen in aanvallen die fysieke, digitale, en sociale technieken gebruiken. De opdracht is gebaseerd op de technieken voor penetratietesten uit bijdrage 3.Het doel van de opdracht is het geven van een breed perspectief op informatiebeveiliging en het vergroten van de interesse van de studenten in het vakgebied. Naast onderwijsdoeleinden kan de opdracht ook gebruikt worden om het beveiligingsbewustzijn van medewerkers te vergroten. Ook levert de opdracht materiaal voor toekomstige security awareness trainingen.

Met behulp van deze bijdragen kunnen professionals op het gebied van informatiebeveiliging het beveiligingslandschap van een organisatie doeltreffender beoordelen en verbeteren.

Acknowledgements

It is morning. I have a coffee and a chocolate muffin next to me. The university laptop is in front of me, ready for its last task. While sipping from the warm coffee, I am thinking of the last four years of my life. Four years full of nice memories, years in which I traveled, I tried things, I laughed, I got (at least I think) smarter and I met my love, Paula.

During these years I was surrounded with great colleagues and friends. Without their support, my PhD would have been a very lonely and boring journey. First of all I would like to thank to my promoter Pieter Hartel and my daily supervisors, Qiang Tang in my early days, and Wolter Pieters in my more ”mature” days. Pieter, it was a pleasure working with you. You guided me from day one of my research, when you introduced me to Latex, up to last night, when we were discussing the abstract of the thesis. I particularly enjoyed the ”knifes on the table” meetings, which sometimes lasted for hours. You were always strong in defending your views and noble in admitting when you were wrong. I guess if all your students were like me, by now you would have had white hair or no hair at all. Nevertheless, you were patient and determined until the very end. Thank you for not giving up on me.

And there is Qiang. We started our lives in the university the same day, Qiang as a post-doc, I as a PhD. Qiang came full with ideas and I came full with enthusiasm. I vividly remember, during my first week in Enschede, I lost my passport. Qiang and I were walking twice the whole distance from Macandra to the University (10km), in the dark, searching through the autumn leaves for a Macedonian passport. It is one of those things you can never forget. Qiang, I will also remember you by the trip we took in Chicago and all the parties we went....especially the ones before Shenglan joined you. Although you were my daily supervisor only for the first year of my research, you were always a friend happy to provide insights in the security field and the PhD life.

After the first year, Wolter become my daily supervisor. From the beginning he was the person I could share my ideas with, discuss how to proceed forward, and

was the perfect mediator between Pieter and me. Wolter, thank you for the guidance in the last three years and for the fresh ideas you brought in my research. I am looking forward to work together with you on topics beyond my PhD research.

Besides my supervisors I had my colleagues next to me. Andre, Shashank and Ove, we were the three musketeers and d'Artagnan. We started our research life at the same time and we are finishing it at the same time (more or less). André, you were my best friend and my colleague in research. Thank you for being always happy to help me with an advice or a drink. Shashank and Ove, this PhD would have been less colorful without you two. When I will look backward at this period of my life, I will remember with a smile the weekends we had at the summer house of Andre and the great trip to Ibiza.

I also want to thank all the current and former colleagues from the DIES group: Stefan, Christoph, Arjan, Dina, Michael, Frank, Jonathan, Saeed, Luan, Mohsen, Frank, Begül, Dusko and the others that I am now forgetting. I will miss our coffee breaks at 3pm, where we discussed about all the topics in life. I will remember the spontaneous trip we took to Barcelona as a result of one of those discussions (Christoph, it is not a nine hours drive to Barcelona). Always present were my friends from SecurityMatters: Damiano, Emmanuele, Michele and Spase. We had great discussions together (when Michele would let us talk) and were supporting each other throughout the years. Guys, thank you for being part of my research life. I am looking forward to work with you in the future. I am also grateful to my favorite secretaries, Bertine and Nienke. Nienke, I was always happy to start the day with you and a hot coffee. Thank you for helping me through the mazes of the Dutch laws and the university procedures. Bertine, thank you for helping me in preparing this thesis and for the useful advices on holiday destinations.

The last four years I spent living in Macandra, the ugliest building in Enschede with the biggest hart in the Netherlands. I met a lot of lifelong friends there. At the beginning it were students from the Erasmus network, with which I spent lots of sleepless nights and great adventures. In the later years, I met there Juan Carlos, Mirjam, Yuri, Edit, Mehmet and Damla. Friends, thank you for the Munchkin games, the skiing visits at Bottrop and the many events we shared together.

In Macandra I met not only my friends, but also my love. Paula, I love you with all my heart and I am grateful for the support and the patience you had during my PhD. Together we spent the best moments of my PhD life. I am looking forward for many more

Finally, the biggest gratitude goes to my family. Mama, tato, bato i Maja. Vie ste mi familija od soništata. Vo poslednive četiri godini, tokmu zaradi vas nikogaš ne se počustvuvav sam. Cel život bezuslovno me podržuvate vo moite nameri i

sekogaš ste bile pokraj mene. Se nadevam vo idnina će se druzime počesto i će imamme ušte mnogu srećni momenti zaedno.

Well, my coffee is over, and my muffin is long gone. Time to start the day and get to work. A new chapter in my life is awaiting to be written.

Den Haag, *Trajce Dimkov*
February 2012

To Paula...

Contents

Chapter 1

Introduction

"Confidential information on almost 130,000 prisoners and dangerous criminals, which was stored on an unencrypted computer memory stick, has been lost by the Home Office, sparking yet another Government data crisis."

The Telegraph, 22.08.2008

"Soldier smuggled highly classified data out of his intelligence unit on a disc disguised as a music CD [...] He is suspected of disclosing more than 150,000 diplomatic cables, more than 90,000 intelligence reports on the war in Afghanistan and one video of a military helicopter attack - all of it classified. Most of the information was given to WikiLeaks."

The New York Times 08.07.2010, 07.04.2011

"The Stuxnet worm, designed to be delivered through a removable drive like a USB stick [...] was designed specifically to attack the Siemens-designed working system of the Bushehr plant and appears to have infected the system via the laptops and USB drives of Russian technicians who had been working there."

Guardian 26.09.2010 02.10.2010

.

1.1 Introduction

The threat of a security breach and loss of sensitive information forces organizations to provide secure and safe environments where the information is stored and

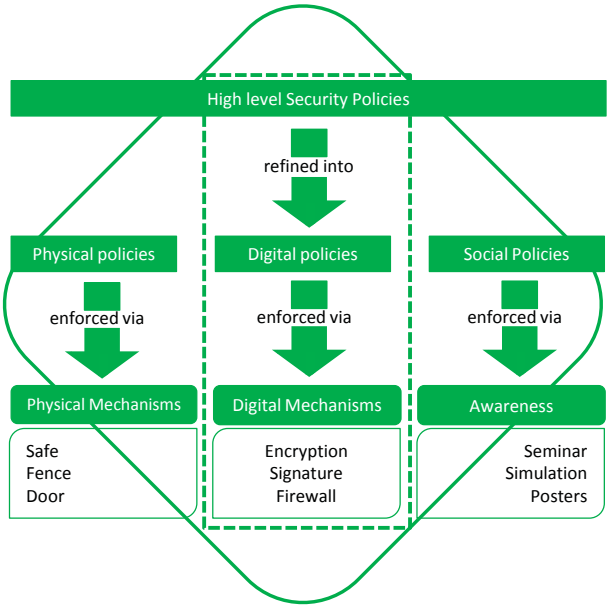


Figure 1.1: High-level policies are refined into low-level, implementable policies. The majority of the current IT research (dashed line) focuses on modeling and analysis of the digital aspect of security, limiting the expressiveness of the models to attacks where the adversary uses only digital means to achieve her goal. The focus of this thesis is the modeling and analysis of attacks where the adversary uses physical, digital and social means (solid line).

processed. An organization protects sensitive information by developing a security program. The security program starts with the management defining all security requirements through high-level security policies. These policies describe the desired behavior of the employees (social domain), the physical security of the premises where the employees work (physical domain) and the IT security of the stored and processed information (digital domain) [88]. After the high-level policies have been designed, the Human Resources (HR), Physical Security and IT departments refine these policies into implementable, low-level policies [17], which are enforced via physical and digital security mechanisms and training of the employees.

During the refinement and enforcement of the policies mistakes may occur. These mistakes could be exploited by both external parties as well as disgruntled employees, insiders, to achieve a malicious goal. Therefore, the management needs assurance that both refinement and enforcement are done correctly. This assurance is achieved in two steps: auditing and penetration testing. During the auditing process, auditors assess whether the security policies produced by the departments

are correct with respect to the policies defined by the management. After the policies from the departments have been audited, penetration testers test the security mechanisms correctly enforce the policies from the departments.

Both auditing and penetration testing are mature fields in information security and follow methodologies that aim for reliable, repeatable and reportable results. However, the attention to the physical and social domain in these methodologies is limited (Figure 1.1). Unfortunately, the adversaries do not limit their actions only to the digital domain but they use any weak link they can find, regardless of the domain. The lack of methodologies for auditing and testing the alignment of security policies across all three domains makes an organization vulnerable to an attack where the adversary combines physical, digital and social actions to achieve her goal.

This thesis focuses on assessing the security of an organization by methodological and experimental tool support for the specification and analysis of security policies that span the three domains, as well as enforcement of these policies via security mechanisms. We show how the contributions in the thesis can help in mitigating the threat from insider attacks, where employees with intimate knowledge of the limitations and the gaps in the existing security policies and security mechanisms obtain access to sensitive information.

1.2 Motivating example

The management of a fictitious organization ACME has defined a set of high-level policies that allow the organization to mitigate security threats and support business processes. For example, to comply with legislation the management has defined the high-level policy HLP_1*: Aggregate sales data should be given to all shareholders.* In the past few years ACME has grown rapidly, causing a shortage of working places for the employees in its facility. As a response, the management produced the policy HLP_2*: One quarter of the employees should work from home*.

Recently, the management identified a new threat. A new competitor is entering the market, offering the same services as ACME. The management wishes to protect its client information from the threat of industrial theft and introduces a new high-level policy HLP_3*: Sales data should not leave the financial department.* This policy is implemented by the departments for physical security, IT security and HR (human resources). In turn, each of the departments refines the policy from management into a set of more specific threats with concomitant security policies in their domain.

High-level threat: The competitors get the list of clients.		
High-level policy from management: **Sales data should not leave the financial department.**		
Domain	**Example low-level threat**	**Example low-level policies**
Physical	Hard drives get stolen from the office	All windows should be locked. Enforce two-factor authentication on all entrance doors of the department. Kensington locks on all computers.
Digital	Malware infection from the Internet	Monitor all network traffic. Forbid remote connections on the computers. Forbid software download.
Social	Employee discloses information	Forbid bringing non-employees at work. Forbid sharing any sales information with non-employees. Forbid employees sharing security policies with competitor employees.

Figure 1.2: A high-level policy and the response from each of the three departments

Table 1.2 provides one representative sample threat identified by each department and three sample policies introduced to mitigate the sample threats. In reality, the number of identified threats and the number of low-level policies that mitigate these treats is much larger and depends on the size and the security requirements of the organization.

Each of the three departments focuses on security policies that mitigate threats from their domain, and relies on policies from the other departments for the other domains. For example, the IT department focuses only on threats from malicious outsiders using remote access. The IT department relies on the physical security department to provide physical isolation between the data and non-employees and on the HR department to educate the employees against being tricked into giving the data away.

However, a number of actions allowed in one domain, when combined with allowed actions from the other two domains, may lead to an undesired behavior. Consider the road apple attack:

> *The competitor leaves a number of dongles with malicious software in front of the premises of the organization. An employee takes one of the dongles and plugs it in his computer in the financial department. When plugged in,*

the malicious software uses the employee credentials to get the sales data, encrypts the data and sends it to a remote server.

In this example the competitor obtains the sales data by intelligently combining the unawareness of the employee, the inability of the doors to stop the dongle and the inability of the firewalls to inspect encrypted traffic. However, none of the departments can individually produce all policies that will stop this attack, because for some policies there are no mechanisms that can enforce them, or the departments cannot identify a threat in their domain that requires such a policy.

The management must be assured that the low-level policies stop all forbidden behaviors and allow all allowed behaviors. Thus, the policies should not only mitigate attacks that use purely digital, physical or social actions, but also any combination of them.

Problem 1: *How can the management be sure that the total set of low-level policies produced by the three departments matches their high-level policy?*

After the low-level policies have been defined, technicians and trainers implement security mechanisms to enforce them. Even if the policies address all allowed and forbidden behaviors, there might still be mistakes in their enforcement. Technicians might put the wrong lock on a door, an employee might ignore or forget some of the policies or some computers might be misconfigured and still accept remote connections. Therefore the departments need to be able to test whether the security policies are properly enforced. These tests should include attempts of gaining physical access to the restricted areas, as well as attempts in tricking the employees to violate a policy. However, organizations are reluctant to execute these tests, because they fear that the tests may stress the employees when asked to violate a policy or disrupt the working process because of accidental damage during the physical access, which results in financial loss.

Problem 2: *How can the three departments be sure that the security mechanisms in place are following the design specifications of the low-level policies?*

1.3 Policy alignment

Policies can be defined at different level of abstraction. In this thesis we use a view of the world as presented by Abrams, Olson and Bailey [73, 10].

Definition 1. *Policy alignment is the process of adjusting security policies among different levels of abstraction to support the business goals of the organization.*

Policy alignment consists of horizontal alignment of high-level policies, vertical alignment of high-level policies into low-level policies and enforcement of low-level policies via security mechanisms.

Definition 2. *Policy refinement is the process of defining multiple policies with a greater level of detail for a given general policy.*

The refinement step should be repeated for each level of abstraction, starting from the policies defined on the highest level of abstraction, toward policies to a lower level of abstraction [73]. To simplify the presentation, we use just two levels of abstraction for the policies.

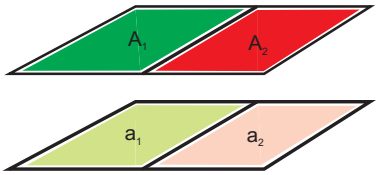
Definition 3. *High-level policies are statements that allow or forbid a set of behaviors.*

A behavior is a sequence of actions, where an action is a discrete event that cannot be broken up further. For example, the road apple attack is a behavior which consists of the actions: competitor leaves the dongle, an employee takes the dongle, an employee plugs the dongle in her computer, the malicious software gets the data, the software encrypts the data and the software sends the encrypted data to a remote server.

The high-level policies divide the space of possible behaviors into behaviors that are allowed, behaviors that are forbidden and behaviors that are neither forbidden nor allowed. In the motivating example HLP_1 and HLP_2 define two sets of behaviors that are allowed, while HLP_3 defines a set of behaviors that is forbidden. All other behaviors are neither allowed nor disallowed.

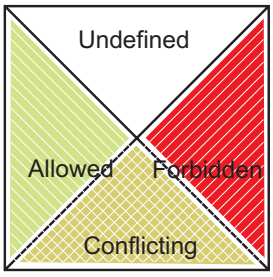
Definition 4. *Low-level policies are implementable rules close to the abstraction level of security mechanisms.*

The low-level policies focus on events rather than on behaviors. Since an event can either occur or not but not both, the low-level policies either allow or forbid an action, dividing the space of possible actions into two disjunct sets. A behavior is allowed by the low-level policies if all the actions it consists of are allowed by the low-level policies. A behavior is forbidden by the low-level policies if at least one of its actions is forbidden by the low-level policies.



A_1: Sales data should never leave the organization.
A_2: Some employees should work from home.
a_1: Low-level policies that forbid the sales data leaving the organization.
a_2: Low-level policies that enable employees to work from home.

Figure 1.3: Ideally, there is no gap nor conflict between high-level policies, and all high-level policies are completely refined into low-level policies.



Allowed: Aggregate sales data should be given to all shareholders.
Forbidden: Sales data should not leave the financial department.
Undefined: Any data other than the sales data.

Figure 1.4: High-level policies may conflict with each other or might be not defined.

1.3.1 Horizontal alignment of policies

Definition 5. *A set of high-level policies is mutually consistent if there is no behavior that is both allowed and forbidden by the policies.*

Definition 6. *A set of high-level policies is jointly exhaustive if every behavior is either allowed or forbidden by the policies.*

Definition 7. *Horizontal policy alignment is the process of positioning high-level policies that are at the same level of abstraction so that they are mutually consistent and jointly exhaustive.*

Consistency between policies means that the policies should not conflict with each other and exhaustiveness means that the policies address all possible behaviors that might occur.

In the motivating example, the organization has a high-level policy that enforces a behavior: *Aggregate sales data should be given to all shareholders*. With the introduction of the policy that forbids a behavior: *Sales data should not leave the financial department* the set of high-level policies is not consistent anymore. There is a conflict between the two policies, because the first policy forbids the sales data leaving the financial department, while the second policy requires some of the sales data to leave the organization.

On the other hand, the absence of high-level policies allowing or forbidding a behavior may introduce a gap in security. In the motivating example, there will be no mechanism that stops an employee giving data other than the sales data to the competitors, because what happens with the rest of the data is not addressed by any of the high-level policies. Since the management has no clear policy on this behavior, security professionals would not know whether to allow or forbid it.

1.3.2 Vertical alignment of policies

Definition 8. *A set of low-level policies is complete with respect to a set of high-level policies if every behavior allowed by the high-level policies is allowed by the low-level policies and every behavior forbidden by the high-level policies is forbidden by the low-level policies [10].*

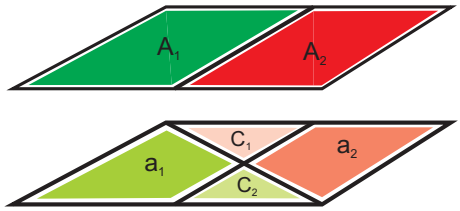
Definition 9. *Vertical policy alignment is the process of refining the high-level policies into low-level policies so that the low-level policies are complete with respect to the high-level policies.*

Even when a set of high-level policies is exhaustive and consistent, the refinement of high-level, organizational policies to low-level, implementable policies may still be incomplete. A high-level policy might be refined into overly permissive or overly restrictive low-level policies, which introduces an opportunity for an adversary to violate the high-level policy (Figure 1.3).

In the motivating example, overly permissive low-level policies such as allowing employees to bring storage devices to work and allowing dongles to be plugged in the computer allow the violation of the high-level policy HLP_3.

There might be two cases when a set of low-level policies is not complete:

- A behavior that is allowed by a high-level policy is forbidden by the low-level policies (area C_1 from Figure 1.5). Such conflicts occur because the high-level policy is refined in overly restrictive low-level policies. In the motivating example, if an employee tries to work from home, she will be stopped by the low-level security policy: *"Forbid remote connections on the computers"*.
- A behavior that is forbidden by a high-level policy is allowed by the low-level policies (area C_2). Such conflicts occur because the high-level policy is refined into low-level policies that are too permissive. In the motivating example, the road apple attack occurs because the low-level policies are too permissive. The policies allow the employees to bring storage devices at work and allow dongles to be plugged in the computers.



A_1: Sales data should never leave the organization.
A_2: Some employees should work from home.

***C_1*:** An employee cannot log-in from home.
***C_2*:** The data is moved to a remote server.

Figure 1.5: In a realistic case, there are behaviors that are allowed by the high-level policies but are forbidden by the low-level policies (C_1), and behaviors that are forbidden by the high-level policies but yet the low-level policies allow them (C_2).

One possible approach in addressing Problem 1 from Section 1.2 is providing a formal assessment whether the low-level policies are complete with respect to the high-level policies. The first part of the thesis uses this approach to address the problem.

1.3.3 Policy enforcement

Definition 10. *Policy enforcement is a process where low-level policies are enforced via security mechanisms.*

During policy enforcement, the security and IT departments place security mechanisms that enforce the low-level policies from the physical and digital domain, and the HR department educates the employees on which actions are forbidden. To test whether the set of security mechanisms is complete, testers check whether these mechanisms are sufficient to enforce the policies. Such tests are done using social engineering in the social domain, physical access in the physical domain and hacking in the digital domain.

Definition 11. *A set of security mechanisms is complete with respect to a set of low-level policies, if every action that is allowed by the low-level policies is allowed by the mechanisms, and every action that is forbidden by the low-level policies is forbidden by the mechanisms.*

In the motivating example, the penetration testers would test whether the employees when politely asked would let a foreign person inside the financial department, or test whether the computers have remote access disabled.

One possible approach in addressing Problem 2 from Section 1.2 is orchestrating

ethical penetration tests that include obtaining physical access and usage of social engineering. The second part of the thesis uses this approach to address the problem.

1.4 Research question

This thesis tackles the problem of policy alignment across the three security domains. The focus of the thesis is assessing the vertical policy alignment from high-level to low-level security policies and testing the enforcement of low-level security policies via security mechanisms.

The main research question we seek to answer in the thesis is:

> **Main research question:** How can we align and enforce security policies spanning the physical, digital and social domain?

Aligning security policies across domains requires three preliminaries. First, the departments should not work in isolation but cooperate in aligning the policies. To work together, the departments need a common language for representing the policies and specify a behavior. Second, obtaining a complete set of behaviors that violate a policy requires exhaustive search on all possible behaviors that can occur for the given low-level policies. Finally, policy testing requires the usage of social engineering and attempts in obtaining physical access.

To address these issues, we refine the main research question in the following refined research questions:

> **Research question 1:** How can we represent the policies from the three domains in one formal framework?

Representing all three security domains in a single formalism is challenging. Firstly, the appropriate abstraction level needs to be found. A too low-level of abstraction for each domain (down to the individual atoms, bits or conversation dynamics) makes the representation complicated and unusable. However, abstracting away from physical spaces, data and relations between people might ignore details that contribute to an attack. Secondly, the domains have different properties making them hard to integrate. For example, mobility of digital data is

less restricted than mobility of objects in the physical domain. Likewise, physical objects cannot be reproduced as easily as digital data.

> **Research question 2:** How can we efficiently discover all cross-domain threats caused by policy misalignment?

Having a formal definition of the environment allows formal methods and tools to exhaustively search all possible behaviors that can occur in the organization. This list of allowed behaviors can then be compared to the behaviors that are allowed by the high-level policies to assess whether any of the produced behaviors is forbidden by the high-level policies. The challenge of this approach is to make it scalable and to ease the assessment of the large amount of behaviors it produces.

> **Research question 3:** How can we test and improve the enforcement of the low-level policies?

Addressing the third refined research question rises three challenges. First, during a penetration test the testers use social engineering on the employees and try to obtain physical access to a specific resource or location. Social engineering always includes some form of deception of the employee, which in turn may cause stress, discomfort or even disgruntlement among employees.

Second, the deployment of security mechanisms and training of the employees is limited by a fixed budget. Currently, the organizations have no clear overview of the effect of security mechanisms from one domain on the security in the other domains. Without a clear overview on how security mechanisms from the three domains supplement each other, it is challenging to prioritize security mechanisms deployment.

Finally, to perform good quality tests, the testers should have training in exploiting vulnerabilities in each of the domains and how have in-depth knowledge on how the vulnerabilities relate between each other. Universities are an excellent location to provide this education, because they can provide environment where the testers can test vulnerabilities and expose them to the ethical implications of penetration testing. However, teaching penetration testing at university level raises the issue whether the students will abuse the obtained skills and knowledge.

1.5 Contribution

This thesis provides methodological and experimental tool support to assess completeness of the policy refinement and techniques for testing the policy enforcement. The results from this thesis can be used as a mitigation of the threat from insider attacks. In detail, the contributions of this thesis can be summarized as follows:

- **A FRAMEWORK** that binds the three domains in a single formalism. We present Portunes, a formal framework which integrates all three security domains in a single environment, thereby enabling analysis of policies that span the three domains. Portunes consists of a graph and a language, that describe a model of the environment of interest at a different level of abstraction. The graph is a visual representation of the environment focusing on the relations between the three security domains. It provides a conceptual overview of the environment that is easy to understand by the user. The language is at a relatively low level of abstraction, close to the enforcement mechanisms. The language is able to describe low-level security policies as predicates and behaviors as process definitions. We provide a proof of concept implementation of Portunes and polynomial time algorithms that produce possible behaviors for a given Portunes model.

- **A LOGIC** for defining high-level policies. We propose a modal logic to describe high-level policies and to express properties of Portunes models and model evolutions formally. The logic is used to find subsets of actions that lead to violation of a high-level policy. The logic enables security professionals to focus only on subsets of attack scenarios that share a common property. We provide a proof of concept implementation of the logic in the Portunes tool.

- **TWO METHODOLOGIES** for physical penetration testing using social engineering. The goal of the penetration tests is to gain possession of an asset from the premises of the organization by using a combination of hacking, physical access and social engineering. Both methodologies are designed to reduce the impact of the test on the employees and the relationship between the employees.

- **AN ASSESSMENT** of the commonly used security mechanisms in reducing laptop theft. We evaluated the effectiveness of existing physical and social security mechanisms for protecting laptops based on (1) logs of laptop thefts which occurred in a period of two years in two universities in Netherlands, and (2) the results from more than 30 penetration tests we orchestrated over the last three years, where students tried to gain possession of marked laptops in the same universities. The results from the log analysis and the penetration tests show that the security of an asset depends mainly on the level of security awareness of the employees, and to a lesser extent on the technical or physical security mechanisms.

- **AN ASSIGNMENT** for increasing the security awareness for employees and future security professionals. We designed the practical assignment of an information security master course where students get practical insight on attacks that use physical, digital and social means. The goal of the security course is to give a broad overview of security to the students and to increase their interest in the field.

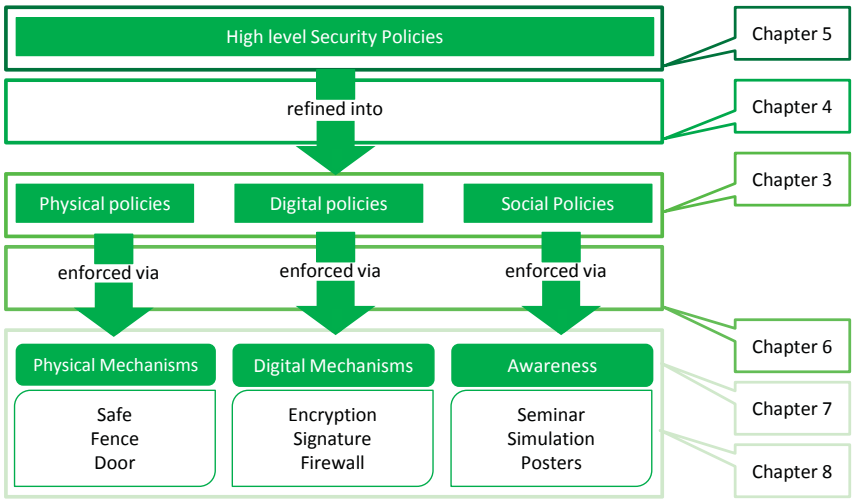


Figure 1.6: Contributions of the thesis

	Research questions		
	RQ1	RQ2	RQ3
Chapter 1			
Chapter 2	✓		
Chapter 3	✓		
Chapter 4		✓	
Chapter 5	✓	✓	
Chapter 6			✓
Chapter 7			✓
Chapter 8			✓
Chapter 9			

Figure 1.7: Research questions addressed in the chapters

1.6 Outline of the thesis

The remainder of this thesis is divided in two parts: *Vertical Policy Alignment* and *Policy enforcement*. The outline of the thesis is depicted in Figure 1.6.

Part I provides a novel approach for representing high-level and low-level policies and techniques for assessing the refinement of the high-level into low-level policies. Chapter 2 first introduces a set of requirements that a model representing all three domains should satisfy. The chapter describes the current state of the art models and analyzes their compliance with the distilled requirements. Chapter 3 introduces Portunes. Portunes is a formal framework which integrates all three security domains in a single environment, thereby enabling the analysis of policies that span the three domains. Chapter 4 describes algorithms that generate all possible behaviors for a given Portunes model and a proof of concept implementation. Chapter 5 provides a modal logic that enables description of high-level policies. We apply the presented framework and logic to describe malicious behavior of an insider, who uses actions that span the three domains to achieve her goal. As a running example through out the first part of the thesis, we use the road apple attack, where the insider uses the trust from a colleague to obtain the financial data.

Part II expands the field of testing policy enforcement. Chapter 6 proposes two methodologies for performing physical penetration tests using social engineering. Chapter 7 assesses the effectiveness of security mechanisms in the physical and

social domain. Chapter 8 proposes a practical assignment for teaching students penetration testing skills. As a running example of the second part of the thesis, we explore the problem of protecting laptops from theft. The last chapter of the thesis, Chapter 9, summarizes the main contributions and provides an outlook on future research directions.

Figure 1.7 illustrates which research questions are addressed in the chapters.

Part I

Vertical policy alignment

The first part of the thesis focuses on vertical policy alignment. We show how low-level policies and high-level policies can be modeled in a single formal framework, and how to analyze the completeness of low-level policies with respect to high-level policies. We use the vertical policy alignment to help in describing, generating and analyzing malicious insider behaviors. As a running example throughout the first part of the thesis, we use the road apple attack, where an insider uses the trust from a colleague to obtain secured data.
First, in Chapter 3 we show how to model low-level policies, behaviors and aspects from the three security domains. In Chapter 4 we show how for a given model, we can *automatically generate* a possible malicious behavior. In Chapter 5 we present a logic that can be used to represent high-level policies. There can be many behaviors that lead to the violation of a single high-level policy. Therefore the logic can be used to select a subset of behaviors that satisfy a high-level policy. The results from the first part of the thesis can be used in two domains, physical penetration testing and auditing. In penetration testing, the testers are interested in a set of attack scenarios that do not violate any low-level policies but still allow them to achieve their goal. After scouting the premises of an organization, the testers can use Portunes to generate a model of the implemented low-level policies and produce attack scenarios automatically. In auditing, the auditors want to assess whether the low-level policies are complete with respect to the high-level policies. Auditors can use Portunes to check whether there exists *any* behavior that can violate a high-level policy.

Chapter 2

Modeling the physical, digital and social domain*

Models play an important role in securing IT systems. They are used to identify possible threats and represent attack propagation throughout the network. We show that current models are not powerful enough to identify the emerging threats from miss-aligned policies due to the inability to represent physical and social aspects from security, such as physical mobility, physical access and social interaction between people. Researchers have proposed security models that particularly focus on representing physical access and social interaction. We show that none of the current security models simultaneously considers the physical and social aspect of security to a satisfactory extent. As a result, none of the current security models effectively represents the security policies from the physical, digital and social domain. Therefore these models cannot identify potential security threats where an adversary uses physical access and social interaction to achieve a malicious goal.

*This chapter is a minor revision of the paper "On the inability of existing security models to cope with information mobility in dynamic organizations" [4] published in the Proceedings of the Workshop on Modeling Security (MODSEC'08), CEUR Workshop Proceedings, 2008

2.1 Introduction

To secure their sensitive information, organizations define policies that restrict physical mobility of people and assets, digital mobility of information and social interactions between employees. In the last decade three main trends have emerged in information systems, that increase the need for a formal approach in studying such policies. The first is information omnipresence raised by the increasing usage of mobile devices. The second trend is the increasing usage of outsourcing. Organizations gain access to a highly trained workforce by becoming decentralized and by outsourcing whole business processes and departments. The last trend is the increasing cooperation between organizations. To increase market share, organizations carry out joint projects with other organizations and extensively hire part-time consultants. These trends lead to increased risk from social engineering attacks [69] and attacks where the adversary uses physical access [11]. Attacks that use physical access and social engineering emphasize the need for closer analysis of the policies that define the access to information and interaction between employees and their alignment to the high-level security policies of the organization.

Researchers from the industry are aware of the increase of mobility of people and assets [63, 75, 100] as well as the impact of social interactions on security [15, 107, 62]. A number of mechanisms, such as best practices of protecting against laptop theft and increasing the security awareness of the employees are proposed to help the organization mitigate the threats due to mobility and social interaction [61, 118, 119, 116, 117]. All of the solutions partially restrict the mobility of data and laptops and are based on best practice criteria.

Problem Information omnipresence, outsourcing and cooperation between organizations increase information mobility and social interactions more than ever, making it increasingly difficult to align the low-level security policies with the high-level security policies in the organization.

Contribution A step toward understanding the security implications of the mobility of information and the social interactions in an organization is to create a model that includes the digital, physical and social aspect of security. We show that threats that arise from mobility of information and social interaction cannot be presented with the existing security modeling techniques. We define the requirements for an integrated security model and look in the literature at alternative models of security that can represent the mobility of information and social interaction. We analyze state of the art security models using attack scenarios presented in a case study, show that none of the new security models consider both of information mobility and social interaction to a satisfactory extent, and present

requirements for an integrated model that addresses this deficiency.

The remainder of the chapter is organized as follows. Section 2.2 provides a case study of current threats that include mobility of objects, interaction between a person with a machine and interaction between people. Section 2.3 introduces the requirements for an integrated security model that is able to present the attacks presented in the case study. Section 2.4 presents the analysis of current models and shows to which extent the security models satisfy the requirements of the integrated security model. Section 2.5 briefly touches on a few informal models that describe physical access and social interaction and Section 2.6 concludes the chapter.

2.2 Case study

To provide a focus for the analysis, we present two attacks on a laptop. The first type of attack is based on permanent physical possession of the laptop and focuses on the confidentiality of the information stored inside. The second type of attack introduces social engineering as a way to provide access to the laptop and focuses on the integrity of the data in the laptop.

We chose these attacks because they include a combination of social engineering with physical and digital access, making them a representative set of the type of attacks we are interested in and a suitable set for analyzing the expressiveness of presented models.

2.2.1 Confidentiality of the data in a laptop

If the adversary is in possession of the laptop, the adversary is also in possession of the encryption keys, making the storage of encryption keys in tamper resistant hardware crucial. The threat model of a storage device [55, 27] provides a variety of options for the adversary to consider, such as removal or tampering with parts of the device. The need for a good protection of the encryption keys has become widely acknowledged after the coldboot attack [53], which is therefore worthy of further study.

To present the coldboot attack, we first introduce a simplified example of presenting encrypted data to a user as shown in Figure 2.1. The snapshot is taken from the Microsoft Threat and Analysis Modeling tool (TAM) and modified (e.g. numbers are added to present the sequence of the calls), to give a better overview of

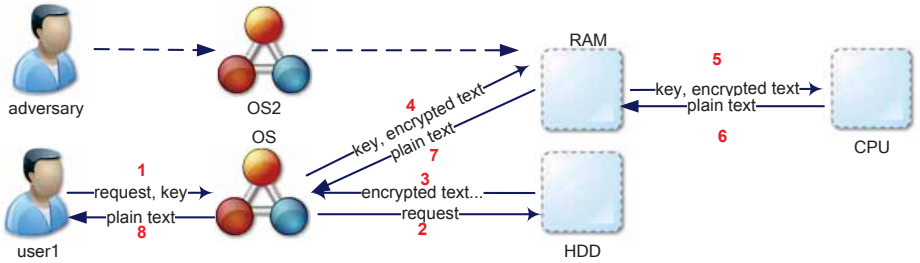


Figure 2.1: Coldboot attack

the example.

The user presents to the operating system a key coupled with a request that defines the data the user wants to read (1). The operating system forwards the request to the hard drive (2) and recovers the encrypted data (3). Then, the operating system loads the encrypted data together with the key into the RAM (4). From the RAM, the operating system feds the data into the processor (5), which as a result returns the plain text (6). Th operating system then sends the plain text to the user (7,8). In the coldboot attack, the adversary does not target the hard drive with the sensitive information, nor the operating system, but the RAM where the encryption keys are stored. When it is not possible to boot the computer from another media, the adversary physically transfers the RAM to another computer, and dumps the memory on a hard drive. Later, the adversary has all the time needed to use search algorithms on the dumped memory to get the encryption keys.

2.2.2 Rootkit attacks on a laptop using social engineering

Stealing a laptop provides an instantaneous benefit to the adversary. However, installing malware that sends data periodically from the internal network of the organization to the adversary is more dangerous. To infect the network, the adversary needs to combine social engineering with malicious software such as rootkits [93], making the mobile device an excellent carrier of the malicious software. A rootkit [93] is software that hides itself and other files from diagnostic and security software and is used in a bundle with viruses, Trojans and other malicious software. A rootkit can be installed on the ROM of any peripheral device [111], in the ACPI tables in the BIOS [112] or in the RAM of the laptop [109]. There are several ways an adversary can use to install a rootkit [93] on a laptop.

The term road apple refers to an apple that is found on a road, tempting the finder

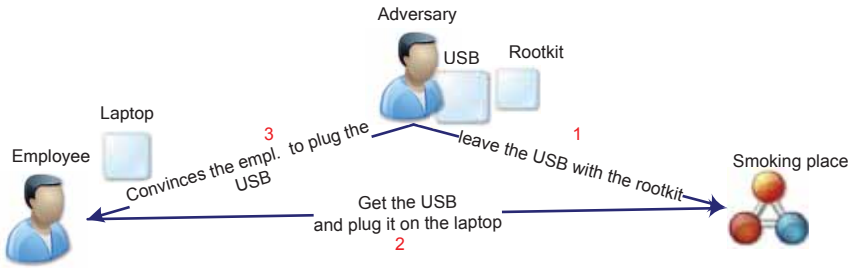


Figure 2.2: Road apple attack

to take it. In the IT world, the apple is usually an infected generic dongle (ex. USB stick) with the logo of the organization left by the adversary in a social place of the organization, such as a cafeteria. When an employee finds the dongle he may be tempted to plug the dongle into his laptop [122]. In the rest of the chapter we call this case road apple 1.

Another approach by the adversary to realize the road apple attack is through direct interaction with the employee. For example, the adversary impersonates higher level management and builds a trust relationship with the employee. The adversary provides a fake identity and simulates an emergency, asking to send a file he has on a dongle through the laptop of the employee. If the employee plugs the dongle on the laptop, the dongle will install the rootkit without the employee's knowledge [12, 30, 31]. In the rest of the chapter we call this case road apple 2.

2.3 Integrated security model of the world

When an adversary tries to compromise a system, the adversary uses all available resources, which besides digital penetration include physical possession of a device and usage of social means to acquire sensitive information. To model the coldboot attack and physical tampering with devices, we need to be able to model the tamper resistance of components in a laptop. We also need to present the removal/addition of components in the laptop. The road apple attack, as many other social engineering attacks [69] relies on activities occurring in the digital, physical and social world. Thus, we need a model which presents movement and roles, as well as physical and digital objects.

The digital, social and physical aspects are defined by Wieringa [104] and we quote his definitions below:

> The physical world is the world of time, space, energy and mass mea-

> sured by kilograms, meters, second, Amperes, etc. The social world is the world of conventions, money, commercial transactions, business processes, job roles, responsibility, accountability, etc. structured in terms of conceptual models shared by people. At the interface between the social and physical worlds we have the digital world which consists of symbols that have a meaning for people.

A step towards understanding the security implications in an organization caused by the mobility of assets and information as well as the social interactions between people, is to create a model that includes the digital, physical and social aspect of the world. Implicitly, this topic is touched upon in the system requirements domain [57], where the user describes the environment in which the system operates.

Here we provide requirements of an integrated security model of the world from the digital, social and physical aspect, together with the basic building blocks the model needs to include.

The requirements we want an integrated security model to achieve are:

1. *The model should be capable of representing the data of interest.*
2. *The model should be capable of representing the physical objects in which the data resides and the locations where the physical objects are stored.*
3. *The model should be capable of representing the roles a user can have.*
4. *The model should define the interactions between the data, physical objects and the roles.*

The first three requirements present the digital, physical and social aspect of the world, while the last binds them together. Following the requirements and the definitions of the physical, digital and social aspect, elements of interest in the integrated security model are: *data*, *physical objects*, *roles* and *interaction relations*.

We use the attacks from the case study to provide focus of the analysis and show how the above requirements present properties of real-life attacks. In Section 2.4 we use the same attacks to show how the inability of a model to satisfy a requirement leads to inability to present a specific attack from the case study.

From the digital aspect represented by the data, we believe that the integrated model needs to present the data at rest as well as data in movement. The spatial/temporal characteristic provides information about the movement of the objects which is needed to model the attacks presented in Section 2.2. To represent

Aspect	Element	Property
Digital	Data	Static, Dynamic
Physical	Object	Resistance, Spatial
Social	Role	Interaction, Transition

Table 2.1: Properties of interest for an integrated model

tampering with a device, the model should be capable of representing the physical properties of an object including the boundary of the object. From the social aspect we are interested in the transition of one role to another, as well as the interaction between roles. Through role interaction and role transition we can represent the impersonation of an adversary and adversary's direct interaction with an employee as presented in Section 2.2.2.

A model that will enable a security expert to represent the physical and social security aspects in organizations will give the security expert better insight in the threats and attack vectors, leading to an understanding of which low-level policies are not aligned with the high-level policies.

To predict the behavior of a system over time we need a state based model. Schneider [90] argues that a static model cannot enforce security policies because the capability of a user can change over time. Goguen [48] presents a capability state model to present dynamic changes in the system, and based on the changes of the capability of a user, defines dynamic security policies. Goguen uses predicates defined over the sequences of operations used to reach the current state, instead of using a predicate on a single state.

In an integrated security model of the world, states or a sequence of states, should be classified based on the properties we want to model. One example is distinguishing the difference between states that are possible in the real world and states that are not. Another example is classification between states that cause violation of a high-level policy and states that do not violate a high-level policy.

2.4 Security models

Motivated by the examples of attacks described in section 2.2 we did an exhaustive literature search for models that are capable of presenting the attacks from the case study. The most promising line of work comes from Probst et al. [84], and uses

a modification of the Klaim language [70]. In Chapter 3 we present the Klaim language in greater detail and show how we improve upon it. In this section we present a list of relevant formal models that use other formalism and present their weaknesses. During the literature search we also found models that represent informal models that use conceptual approach in describing of the three security domains. They are shortly addressed in Section 2.5.

Most of the formal models we found focus on modeling the data from the digital aspect (e.g. data flow) and only a limited number of models consider the location of the data. To the best of our knowledge there is no integrated security model which includes all three aspects (digital, physical, social), and thus there is no model that can truthfully represent the security implications on data mobility in dynamic organizations.

We focus on models from the computer science domain modeling a security property of the system, such as privacy or confidentiality. TAM and Secure Tropos (ST) (Subsection 2.4.1) are static and used in the software industry for generation of threats for a specific software application. Then we move into dynamic, state based security models (Subsections 2.4.3 and 2.4.4) that include mobility of the components in the system. These dynamic models are all inspired by the ambient calculus [23], for which we provide the basic structure. Later we explore how the ambient calculus is extended to focus on different properties of the world in two other security models. We analyze the characteristics of these models with respect to the requirements presented in Section 2.4.5. A more detailed and technical elaboration of the conclusions is presented in Appendix A.

2.4.1 TAM and Secure Tropos

One of the first steps when looking at a security issue is to create a threat model [96]. To generate the threats, the threat model needs to provide a security model of the system on which it runs the threat generation algorithm. Usually, the input of a threat model is the security model of the system, and the output is a set of threats. The model does not specify how these threats could happen (which makes the model attack independent) but recognizes the existence of such threats. This set is later used as an input for risk assessment and report generation. In the literature, threat modeling focuses on applications and networks. The scientific community has worked on a formalization of threat modeling [108, 29] and produced algorithms for threat generation [72, 77] and sorting [28]. This led to a number of tools which partially automate the threat modeling and generation process space [115, 120]. Here we consider TAM [115] which is a state of the art tool

used for internal threat generation and analysis in software development organizations, as well as Secure Tropos, a formal model used for high-level presentation of software requirements.

In Section 2.2.1 we used TAM to model the coldboot attack (Figure 2.1). Besides being able to model data structures (OS and $OS2$) and data flow (the solid lines between the objects) the tool also presents physical objects (RAM, CPU and HDD) as well as roles ($user1$ and $adversary$). The addition in the model, where the adversary takes the key from the RAM using the second operating system is presented with the dashed line.

TAM considers the physical component and the role as static and the data as dynamic, allowing the TAM threat generation algorithm to focus on the flow of data. Although this reasoning is understandable and valid in software modeling, in the presented attacks TAM proves to be restrictive. TAM does not take into consideration the possibility that a component can be removed, such as the RAM in the coldboot attack nor that a component is mobile, such as the dongle in the road apple attack.

TAM presents neither role interaction nor role transition. Because of the lack of states, even with manipulation of the relationships and entities in the model, TAM cannot present interaction between roles and role transition. The role in TAM is used to describe the privileges over a component in an access control table, but does not define transition between roles such as escalation of privileges between a normal and an administrator role nor any interaction between roles, such as delegation or separation of duty. As a result, TAM cannot present the road apple attack where the adversary has direct interaction with the employee.

TAM cannot present physical properties of a component. A component is defined through the service type the component provides and the data and roles the component interacts with. Since TAM does not consider the component as a physical object, the component's resistance to physical attacks cannot be expressed in the model.

We can change the meaning of the components to present the attacks from the case study, but not without changing or blurring the relationship between the components. We can "attach" a new operating system to the RAM. As the number of mobile components increases the number of such "attachments" also increases, degrading the model usability as well as blurring the meaning of the relationship between components. Still TAM model "attachments" are used in modeling the coldboot attack as presented in Figure 2.1.

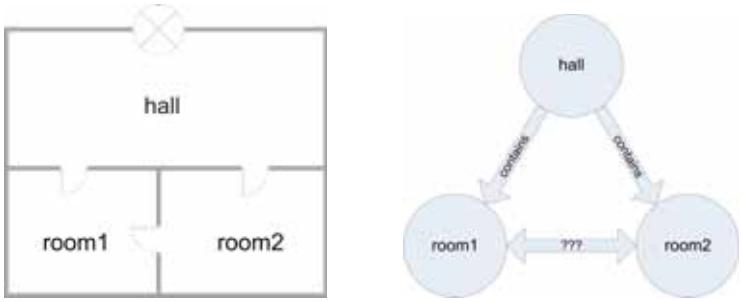


Figure 2.3: A floor plan and its tree representation [80]

2.4.2 Ambient calculus

Ambient calculus [23] provides an excellent apparatus for modeling a world with mobile components. The calculus is capable of presenting spatial and temporal properties of a component(with running processes inside) in the model. Ambient calculus serves as an inspiration for the state of the art security models that consider mobility of components. The ambient calculus has been expanded into typed ambient calculus [22], boxed ambient calculus [21] etc. All of these calculi focus on a specific security property such as boundary interference [20].

Ambient calculus does not define the properties of an entity nor the relationship between entities, making the calculus generic enough to present any model of interest. The calculus presents a comprehensive theoretical framework for reasoning about mobility. But, without additional formal naming convention and definition of the properties of interest in the component, cannot be directly implemented in any model on which mechanisms such as policies or threat generation algorithms need to be applied.

Ambient calculus cannot present tampering with a device. In ambient calculus data decides to leave the device or not based on the capability of the data, which is not the case when an adversary tampers with a device. Although tamper resistance can be presented through a stack of ambients, the manipulation of the stack cannot be done at run time, because any rearrangement or removal of a layer requires a dynamic change of the capabilities of the data inside.

Finally, ambient calculus is based solely on a containment relation. As pointed out in the work of Pieters [80], containment based models cannot present neighboring relationship between objects. For example, we cannot model a floor that has neighboring rooms (Figure 2.3), or networks separated by firewalls.

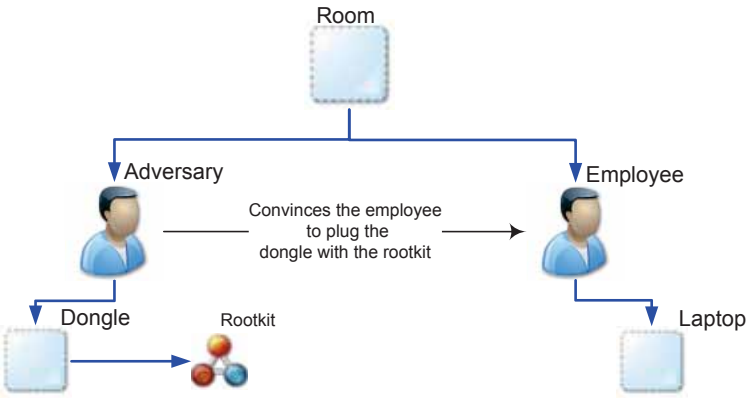


Figure 2.4: Road apple using entities. The model of Scott can present the spatial location of entities (blue arrows), but not the social interaction between them (black arrow).

2.4.3 Model of Scott

Scott [91] builds a security model of the world by adding a spatial relationship between the elements in the ambient calculus. Scott's model is based on a building block called an *entity*. An entity is a spatial location. Every entity belongs to only one of six defined *sorts*. To distinguish physical entities from digital entities, Scott defines a *context*, a physical/virtual machine capable of running code. Scott's model uses capabilities from ambient calculus (*in/out*) and renames the capabilities depending on which entity uses the capability. If the entity is a person moving between rooms, the capabilities are *walk in/walk out*. If the entity is a person interacting with a laptop, the capability is *pick up/put down*. If the entity is an agent moving between contexts, the capabilities are *emit/receive*.

To present tamper resistance of an entity, we can add multiple layers of protection to the data by inserting additional entities. But the definition of the *emit/receive* command teleports an entity from source address to destination address without taking in account the layers in between, making the model oblivious to the tamper resistance imposed by the device.

There is no social factor in the model of Scott. There is a sort `person`, but the meaning is spatial. The only capability this entity has is to pick up or put down a mobile entity. Through this we could present the coldboot attack, where the person physically changes the location of the RAM as well as the first version of the road apple attack. But the model cannot represent the direct interaction between the adversary and the employee in the second version of the road apple

attack, where the adversary directly interacts with the employee and convinces the employee to insert the dongle (Figure 2.4). Thus, the model cannot fully present the road apple attack.

2.4.4 Model of Dragovic

Dragovic [41] presents a security model of the world by expanding Scott's model and focusing on exposure treats. The main building blocks are *data object*, which presents a collection of data with equal sensitivity as determined by a security policy and *container*, which is an ambient (digital or physical) containing a data object or a lower level container. In a Dragovic model, the container has as a boundary that protects the container or data object inside from the outside influences with variable degree of success. Every container propagates downwards its own influences in addition to the influences the container inherits from the parent container. Boundary transparency is defined based on the degree of protection the parent container offers to the child container. Dragovic uses *class* (similar to Scott's sort) to group elements. Another distinction is made by adding a *type* to the container, which presents the behavior of the container when exposed to an influence from the environment. Mobility of the data is presented by four operations: *enter*, *leave*, *migrate*, which atomically binds the previous two operators and *state_update*, which is used to update the status of the attributes of a container. The model presented by Dragovic [40, 41] besides considering the spatial/temporal characteristics of the object, considers the object's physical properties, such as the object's capability to resist influences from the surrounding environment, making the model suitable for presenting the tamper resistance of a device.

The model of Dragovic includes Scott's model with the addition of the physical property of the objects, as well as the definition of sensitivity of data, allowing us to model tampering with a device and the coldboot attack to a level where all elements are realistically presented. Figure 2.5 presents the spatial relationship of the containers in the cooldboot attack. To model the coldboot attack, we define the RAM as a container and the encryption key as a data object. The accessibility of the RAM is defined by the RAM's transparency in addition of the laptop's transparency. Before the coldboot attack, we consider the RAM as a container with limited tamper resistance. After the RAM is removed from the laptop, the tamper resistance of the RAM increases due to the degradation of the data. Thus, we can successfully present the coldboot attack.

Dragovic does not define an object *person*, therefore there is no defined interaction between a person and a container. By presenting the employee and the adversary

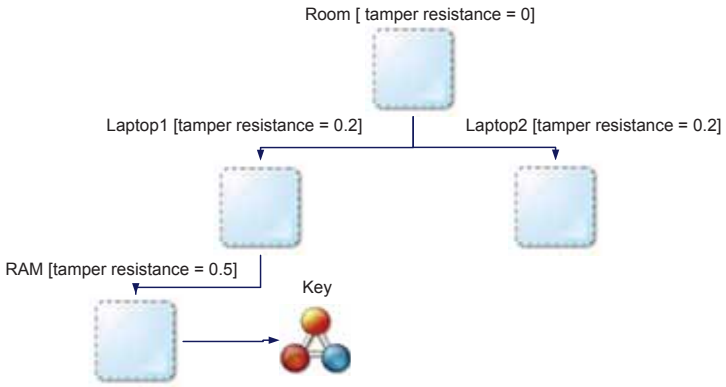


Figure 2.5: The coldboot attack using containers. Each container has a determined transparency, allowing the modeling of tampering attacks. The lines between the containers represent their spatial location.

as containers, we are able to present the movement of the dongle with the rootkit from the adversary to the employee's laptop. Yet, we are not able to present the interaction between the adversary and the employee, where the employee is convinced to insert the dongle. Thus, we cannot model the road apple attack with direct interaction.

2.4.5 Comparison of the models

This section compares the analyzed modeling approaches. Table 2.6 presents the objects and properties of the objects we are interested in the analyzed models.

From the presented results, we make the following observations. The ambient calculus is formal and capable of presenting most of the properties of interest. Other models impose restrictions on the model enabling them to focus on a specific area of interest, making the models less general than ambient calculus. This prevents the models to represent some of the properties of interest. TAM is incapable of presenting physical or social properties, because the model focuses on software representation and does not contain states. Scott and Dragovic cannot present role transition and role interaction because they do not include any social element in the model.

Table 2.7 provides an overview of the model's ability to present tampering with a physical device, the coldboot attack, as well as the road apple attack with indirect

Aspect	Element	Property	TAM & ST	Ambient calculus	Scott	Dragovic
Digital	Data	static	yes	yes	yes	yes
		dynamic	yes	yes	yes	yes
Physical	Object	spatial/temp.	no	yes	yes	yes
		resistance	no	no	no	yes
Social	Role	transitions	no	no	no	no
		interactions	no	yes	no	no

Figure 2.6: Ability of the models to present digital/physical/social elements

Name of attack	TAM & ST	Ambient calculus	Scott	Dragovic
Tampering	no	no	no	yes
Coldboot	partially	yes	yes	yes
Road apple 1	no	yes	yes	yes
Road apple 2	no	yes	no	no

Figure 2.7: Ability of the models to present the case study attacks

(road apple 1) and direct (road apple 2) interaction between the adversary and the user.

Tampering with a device can be presented with the model of Dragovic because the model can contain information about the property of a device. TAM does not have this capability, and thus is not able to present the tampering. The model of Scott can use multiple layers to represent resistance, but the teleporting ability of data makes any attempt to represent resistance obsolete. The operators in ambient calculus do not support teleporting, enabling the presentation of the tamper resistance through multiple layers. Yet, the capabilities of the ambient cannot change dynamically based on the change of the layer structure, preventing the complete presentation of tampering with data.

We are able to present the spatial movement of the dongle from the adversary to the employees laptop, but are not able to present the social interaction between the adversary and the user, where the adversary convinces the user to plug the dongle. This is the reason why Scott and Dragovic can only partially model the road apple with direct interaction.

2.5 Conceptual models

Jiang et al. [60, 59] present a data structure for the privacy issues in the ubiquitous computing through data structures called *information spaces*. The model of Jiang et al. focuses on presenting social groups and activities, which is a major improvement with respect to the previously introduced security models, but the definition of the model is informal, making the model open for interpretation. Prayogi et al. [83] provide an access control framework for selective role transition based on the change of the context in which the system resides. However, the role transitions are not formally defined. In the social and business fields, Hartmann et al. [54] provide informal model of user interaction. The model is used for optimizing profit rather than investigating security implications.

2.6 Conclusion

We analyze the capability of state of the art security models to present the treats arising from physical and digital mobility as well as social interaction in organizations. We show that none of the state of the art security models simultaneously consider the data mobility and social interaction to a satisfactory extent. Software modeling tools, like Microsoft's TAM, consider the physical infrastructure and roles to be static and this makes it hard to present dynamic changes in the system. Security models for ubiquitous computing are state based, but focus on spatial/temporal characteristics and fail to recognize social interactions, which are vital for social engineering threats. As a result, we conclude that none of the presented state of the art security models effectively describes the physical and social aspect of security. Thus, these models cannot identify the potential security threats caused by misalignment of low-level policies with high-level policies.

The information omnipresence and social interactions in organizations shift the stress from mainly digital attacks to a combination of digital, physical and social attacks. To cope with the threats, the chapter presents the requirements for an integrated state based model. The goal of the proposed requirements is to aid in defining a model of the world from all three aspects, digital, physical and social and realistically present the possible attacks. The chapter identifies the objects of interest from all three aspects and presents an initial classification of the properties affecting the security of the identified objects.

In the following chapter we define a formal security model that satisfies the requirements provided here and defines the interactions between the identified objects, based on the properties of the objects.

Chapter 3

Portunes*
Representing multi-domain behavior

In this chapter, we present Portunes, a framework that incorporates three security domains: (1) the security of the computer system itself (the digital domain), (2) the security of the location where the system is deployed (the physical domain) and (3) the security awareness of the employees who use the system (the social domain). The framework is able to present low-level policies as well as behaviors that span the three domains.

The Portunes framework can be used by auditors to assure the management that the low-level policies are complete with respect to the high-level policies. The framework can be also used to assist penetration testers by automatically generating "what if" scenarios, that can be used as parts of the tests. We explore these usages of the framework in chapter 4 and chapter 5.

We show how the framework can be used to describe malicious behavior of an insider, who uses actions that span the three domains to achieve her goal. We formalize a variation of the road apple attack as a running example, where the insider uses the trust from a colleague to obtain the secure data.

*This chapter is a minor revision of the paper "Portunes: representing attack scenarios spanning through the physical, digital and social domain" [2] published in the Proceedings of the Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'10), pages 112-129, Springer Verlag, 2010

3.1 Introduction

Malicious insiders are a serious threat to organizations. Motivated by greed or malice, insiders can disrupt services, modify or steal data, or cause physical damage to the organization. Protecting assets from an insider is challenging [110] since insiders have knowledge of the security policies in place, have certain privileges on the systems and are trusted by colleagues. An insider may use the knowledge of the security policies to avoid detection and use personal credentials or social engineer colleagues to carry out an attack. Securing the environment from a malicious insider requires (1) aligning the low-level policies with the high-level policies and (2) checking if an insider can violate a specific high-level policy.

Most current formal models for modeling insider threats [87, 99, 52] assume that the insider uses only digital means to achieve an attack. Therefore, these models do not look into mobility of people and devices nor social interactions between people, and focus only on modeling the security of a network or a host. For example, there is a lot of research that focuses on modeling and analyzing network and host configurations to generate, analyze and rank attack scenarios using attack graphs [102, 67, 103, 101, 66].

Assuming that the insider uses only digital means to achieve an attack leaves an essential part of the environment of interest not captured in the security models. Indeed, a study performed by the National Threat Assessment Center in the US (NTAC) [86] shows that 87% of the attacks performed by insiders require no technical knowledge and 26% use physical means or the account of another employee as part of the attack. Thus, a whole family of attacks, digitally-enabled physical attacks and physically-enabled digital attacks [38], in which the insider uses physical, digital and social means to compromise the asset cannot be presented nor analyzed formally.

The contribution of this chapter is Portunes[1], a framework which integrates all three security domains in a single environment, thereby enabling the analysis of multi-domain behavior. The Portunes framework consists of a graph and a language, which can describe an environment at a different level of abstraction. The graph is a visual representation of the environment focusing on the relations between the three security domains. It provides a conceptual overview of the environment that is easier to understand by the user. The language is at a relatively low

[1] After Portunes, the Roman god of keys

level of abstraction, close to the enforcement mechanisms. The language is able to describe the low-level policies defined by the security departments as predicates and the behaviors that span across the three domains as process definitions.

The rest of the chapter is structured as follows. Section 3.2 gives an overview of related work contributing to the design of Portunes. Section 3.3 formalizes the Portunes graph and Section 3.5 formalizes the Portunes language. Section 3.6 concludes the chapter.

3.2 Related work

The design of the Portunes framework is influenced by several research directions, such as insider threat modeling, physical modeling and process calculi. This section lists several papers which influenced the design of Portunes and describes how Portunes extends or deviates from them.

Dragovic et al. [41] are concerned with modeling the physical and digital domain to determine data exposure. Their model defines a containment relation between layers of protection. Data security is determined not by access control policies, but by the number of layers of protection above the data and the confidentiality provided by each layer. The Portunes framework uses a similar relation to present the location of elements, but uses explicit access control policies to describe security mechanisms.

Scott [91] focuses on the mobility of software-agents in a spatial area and usage policies that define the behavior of the agents depending on the locality of the hosting device. The mobility of the agents is restricted through edges on a graph. The Portunes framework adds semantics to the graph structure by giving meaning to the nodes and edges and defines invariants enforced directly into the semantics of the language.

Mathew et al. [66] use capability acquisition graphs to describe the physical structure of a building. The nodes in the graphs are static, and the graph can present the progress of the insider in the graph. In our solution the structure of the graph evolves as the attack progresses, and the insider can interact with other employees to obtain additional capabilities.

Klaim [70] is a process calculus for agent interaction and mobility, consisting of three layers: nodes, processes and actions. There are several Klaim dialects, including μKlaim [50], OpenKlaim [19] and acKlaim [84]. The goal of the acKlaim language, which is closest to our work, is to present insider threats by combining

the physical and digital security domain. Mobility is presented by remote evaluation of processes. The Portunes language builds upon these Klaim dialects. Firstly, the actions for mobility and embedding of objects (login, logout) are similar to OpenKlaim. Secondly, the policies expressed in the Portunes language are similar to acKlaim and μKlaim. However, in the Portunes language mobility is represented by moving nodes rather than evaluating processes. Finally, the Portunes language lacks tuple spaces which are present in all other Klaim variants. The tuples in the physical and digital world are completely replaced by the containment sets. The social world is presented through the low-level policies, thus no tuples are needed. The absence of tuple spaces reduces the number of possible process definitions, allowing their automatic generation.

3.3 Portunes

This section presents the Portunes framework. We first present the requirements which Portunes needs to satisfy and the motivation behind some of the design decisions. Based on the requirements, we formally define the Portunes graph and the Portunes language. To show the expressiveness of the framework, we use a variant of the road apple attack as an example of a malicious behavior.

3.3.1 Requirements and motivation

The three security domains focus on different aspects of security. Physical security restricts access to buildings, rooms and objects. Digital security is concerned with access control on information systems. Finally, security awareness of employees focuses on resistance to social engineering, and is achieved through education of the employees.

Representing all three security domains in a single formalism is challenging. Firstly, the appropriate abstraction level needs to be found. A too low level of abstraction for each domain (down to the individual atoms, bits or conversation dynamics) makes the representation complicated and unusable. However, abstracting away from physical spaces, data and relations between people might omit details that contribute to an attack. Thus, a model integrating multiple security domains needs to be expressive enough to present the relevant details of an attack in each security domain. In chapter 2, we provided the basic requirements for an integrated security model to be expressive enough to present detailed attacks. Briefly, an integrated security model should be able to present the data of

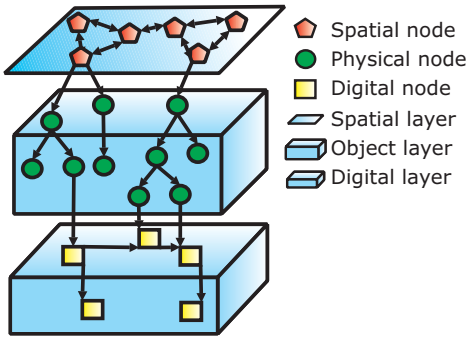


Figure 3.1: Graphic presentation of elements in Portunes

interest, the physical objects in which the data resides, the people that manipulate the objects and the interaction between data, physical objects and people.

Secondly, the domains have different properties making them hard to integrate. For example, mobility of digital data is not restricted by its locality as is the case with objects in the physical domain. Likewise, physical objects cannot be reproduced as easily as digital data. An additional requirement for Portunes is to restrict interactions and states which are not possible in reality. For example, it is possible to put a laptop in a room, however, putting a room in a laptop is impossible; a person can move only to a neighboring location, while data can move to any location; data can be easily copied, while the reproduction of a computer requires assembling of other objects or materials.

3.4 The Portunes graph

To present the different properties and behavior of elements from physical and digital security, the Portunes graph stratifies the environment of interest in three layers: spatial, object and digital. The spatial layer presents the facility of the organization, including rooms, halls and elevators. The object layer consists of objects located in the facility of the organization, such as people, computers and keys. The digital layer presents the data of interest. Stratification of the environment in three distinct layers allows specification of actions that are possible only in a single layer (copying can only happen for digital entities) or between specific layers (a person can move data, but data cannot move a person).

A Portunes graph abstracts the environment of an organization in a stratified graph and restricts the edges between layers to reflect the neighbor and containment

relations that occur in reality. A node abstracting a location, such as an elevator or a room, belongs to the spatial layer L and it is termed a spatial node. A node abstracting a physical object, such as a laptop or a person, belongs to the object layer O and it is termed an object node. A node abstracting data, such as an operating system or a file, belongs to the digital layer D and it is termed a digital node. The edges between spatial nodes denote a neighbor relation and all other edges in the graph denote a containment relation. The ontology used in Portunes is given in Figure 3.2. An edge (n, m) between two spatial nodes means n *is a neighbor of* m. This is a symmetric relation where the direction of the edge is not important. For all other nodes, an edge (n, m) means that node n *contains* node m; this is an asymmetric relation.

layer	node	edge
spatial	location	neighbors
		contains
object	physical object	contains
		contains
digital	data	contains

Figure 3.2: The ontology of a Portunes graph

The above statements are illustrated in Figure 3.1 and formalized in the following definition.

Definition 12. *Let* $G = (Node, Edge)$ *be a directed graph and* $\mathcal{D} : Node \rightarrow Layer$ *a function mapping a node to* $Layer = \{L, O, D\}$*. A tuple* $(G, \mathcal{D})$ *is a Portunes graph if it satisfies the following invariants* $I(G, \mathcal{D})$*:*

1. *Every object node can have only one parent.*
   $\forall n \in Node : \mathcal{D}(n) = O \rightarrow indegree(n) = 1$
2. *One of the predecessors of an object node must be a spatial node.*
   $\forall n \in Node : \mathcal{D}(n) = O \rightarrow \exists m \in Node : \mathcal{D}(m) = L \wedge \exists \langle m, ..., n \rangle$*; where* $\langle m, ..., n \rangle \in Edge^+$ *denotes a finite path from* m *to* n*, and* $Edge^+$ *is a finite set of finite paths.*
3. *There is no edge from an object to a spatial node.*
   $\nexists (n, m) \in Edge : \mathcal{D}(n) = O \wedge \mathcal{D}(m) = L$
4. *There is no edge from a digital to an object node.*

$\nexists(n,m) \in Edge : \mathcal{D}(n) = D \wedge \mathcal{D}(m) = O$

5. *A spatial and a digital node cannot be connected.*
   $\nexists(n,m) \in Edge : (\mathcal{D}(n) = D \wedge \mathcal{D}(m) = L) \vee (\mathcal{D}(n) = L \wedge \mathcal{D}(m) = D)$

6. *The edges between digital nodes do not generate cycles.*
   $\nexists\langle n, ..., m\rangle \in Edge^{+} : \mathcal{D}(n) = ... = \mathcal{D}(m) = D \wedge n = m$

The intuition behind the invariants is as follows. An object node cannot be at more than one place, thus an object node can have only one parent (1). An object node is contained in a known location (2). An object node cannot contain any spatial objects (3) (for example, a laptop cannot contain a room) nor can a digital node contain an object node (4) (for example, a file cannot contain a laptop). A spatial node cannot contain a digital node and vice versa (5), and a digital and object nodes cannot contain itself (6) and Theorem 1. Edges between spatial nodes represent a neighborhood relation which is a reflexive property. However, the edges between object and between digital nodes represent a contain relation, which is not reflexive. For example, it should not be possible for a person to contain a bag, which in turn contains the same person.

Theorem 1. *A Portunes graph* $(G, \mathcal{D})$*, where* $G = (Node, Edge)$*, can have cycles only in the spatial layer:*
$\exists\langle n, ..., m\rangle \in Edge^{+} : n = m \rightarrow \mathcal{D}(n) = ... = \mathcal{D}(m) = L$

Proof. The theorem follows from three properties, which we prove in turn:

1. There are no cycles between layers.
2. There are no cycles in the object layer.
3. There are no cycles in the digital layer.

1. There are no cycles between layers
   $\nexists\langle n_0 ... n_i ... n_k\rangle : n_0 = n_k \wedge \mathcal{D}(n_0) \neq \mathcal{D}(n_i)$
   Lets assume that such a cycle exists:
   $\exists\langle n_0 ... n_i ... n_k\rangle : n_0 = n_k \wedge \mathcal{D}(n_0) \neq \mathcal{D}(n_i)$
   Thus, there are at least two edges in the graph which connect nodes from different layers:
   $\exists(n_{j-1}, n_j), (n_l, n_{l+1}) \in Edge : \mathcal{D}(n_{j-1}) \neq \mathcal{D}(n_j) \wedge \mathcal{D}(n_l) \neq \mathcal{D}(n_{l+1}) \wedge \mathcal{D}(n_{j-1}) = \mathcal{D}(n_{l+1}) \wedge \mathcal{D}(n_j) = \mathcal{D}(n_l)$
   From the invariants 3, 4, 5 (tabulated in Table 3.1) follows that such a pair of edges does not exist.

Layer 1 (L_1)	Layer 2 (L_2)	Edge from L_1 to L_2	Edge from L_2 to L_1
L	O	+	- (invariant 3)
L	D	- (invariant 5)	- (invariant 5)
O	D	+	- (invariant 4)

Table 3.1: Invariants 3,4,5 forbid any cycles between layers.

2. There are no cycles in the object layer.
   $\nexists\langle n, ..., m\rangle : \mathcal{D}(n) = ... = \mathcal{D}(m) = O \wedge n = m$
   Lets assume such a cycle exists:
   $\exists\langle n, ...n_{i-1}, n_i..., m\rangle : \mathcal{D}(n) = ... \mathcal{D}(n_i) ... = \mathcal{D}(m) = O \wedge n = m.$
   From invariant 2, for the node n_{i-1} exists a spatial node m, such that there is a path between m and n_{i-1}, $\exists k \in Node : \mathcal{D}(k) = L \wedge \exists\langle k, ...n_{i-1}^{'}, n_i\rangle$.
   It follows there are two edges to n_i, $\exists(n_{i-1}^{'}, n_i), (n_{i-1}, n_i)$.
   If $n_{i-1}^{'} \neq n_{i-1}$ there is a contradiction with invariant 1. Otherwise $\mathcal{D}(n_{i-1}^{'}) = O$, and the analysis is repeated for the path $\langle k, ...n_{i-1}^{'}\rangle$. Because $\langle k, ...n_{i-1}^{'}\rangle$ is finite, at one point the path reaches a spatial node, and $n_{i-1}^{'} \neq n_{i-1}$. This again contradicts with invariant 1. Thus, such cycle does not exist.

3. There are no cycles in the digital layer.
   $\nexists\langle n, ..., m\rangle : \mathcal{D}(n) = ... = \mathcal{D}(m) = D \wedge n = m$
   This follows directly from invariant 6.

□

Example: Road apple attack To show how Portunes can be used to represent the three domains and represent behaviors across the domains, we use the example of the road apple attack [122, 30, 31] which we introduced in Chapter 1. In an insider version of the road apple attack, the insider may abuse the trust of a colleague and convince the colleague to take the dongle. Instead of the competitor spreading multiple infected dongles around the vicinity of the employee's working place, in the insider version of the road apple attack, the insider social engineers the employee. In this chapter we will formalize the attack in the following steps. First, the insider convinces the employee to take the dongle by abusing her trust (social domain). Then, the employees goes to a server in a restricted area and plugs in the dongle (physical domain). Finally, the malicious software from the dongle transfers the sensitive data to a remote server (digital domain).

To describe the attack, the environment in which the behavior takes place needs to include information from all three security domains. Concerning physical security, the organization has a restricted area where a server with sensitive data resides. Additionally there is a public area where employees can socialize. Re-

$$\mathcal{D}(hall) = \mathcal{D}(secureRoom) = \mathcal{D}(world) = L$$
$$\mathcal{D}(remoteServer) = \mathcal{D}(insider) = \mathcal{D}(employee) =$$
$$\mathcal{D}(secureServer) = \mathcal{D}(dongle) = O$$
$$\mathcal{D}(serverData) = \mathcal{D}(rootkit) = D$$

Figure 3.3: The function $\mathcal{D}$ for the road apple attack environment

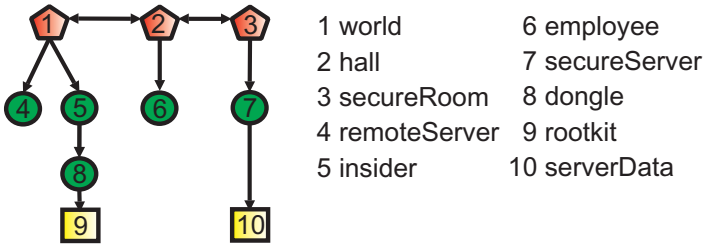


Figure 3.4: Graph of the road apple attack environment

garding the digital domain, the data on the server can be accessed only internally. The security awareness of the employees is such that they trust each other enough to share office material (for example: CDs and dongles).

The segregation of the nodes among the layers is presented in Figure 3.3. The nodes *hall*, *secureRoom* and *world* are spatial nodes, *serverData* and *rootkit* are digital nodes. All other nodes are object nodes. The Portunes graph is visually presented in Figure 3.4. The spatial nodes are presented as pentagons, the object nodes as circles and the digital nodes as squares. The edges in the graph present the relationship between the nodes. For example, the hall is neighboring the secure room and the secure room contains a secure server which in turn contains server data.

In Section 3.5 we define the language that formally specifies the environment of interest and in Section 3.5.2 we will revisit the example and show *how* the road apple attack takes place using the formal specification. In Chapter 5 we will show how to present properties in the road apple example formally.

3.5 The Portunes language

In the previous section, we defined a graph-based approach to present the facilities of an organization, the objects in a facility and the data of interest. The Portunes graph represents the environment on a conceptual level, and compared to the language, it provides simplified presentation of the environment to the user. In this

section we introduce the Portunes language. The language formally describes the environment and makes it more suitable to describe and analyze the enforcement mechanisms as well as to formally specify the interaction between the nodes. The language consists of nodes, processes and actions, where a node in the Portunes language represents a node in the Portunes graph.

The language captures two interactions, *mobility* and *delegation*. By making all nodes first class citizens, every node can move. For example, an object node representing an insider can move through the organization and collect keys, which increase the initial privileges of the insider. Similarly, a spatial node representing an elevator can move between floors in a building. In the Portunes language, a delegator node can delegate a task to a delegatee node. By delegation here we refer to the act of granting the delegatee additional privileges to carry out a task on behalf of the delegator.

The above two interactions, mobility and delegation, are restricted by the invariants from Definition 12 and by the low-level security policies associated with each node. Policies on nodes from the spatial and object layer represent the physical security. These policies restrict the physical access to spatial areas in the facility and the objects inside the spatial areas. Policies on nodes from the digital layer represent the digital security of the organization and focus on access control on the data of interest. In the Portunes language people can interact with other people. Policies on people give the social aspect of the environment, or more precisely, they define under which circumstances a person *trusts* another person.

3.5.1 Overview of Klaim

The Portunes language is inspired by the Klaim family of languages. Klaim (Kernel Language for Agent Interaction and Mobility) is an experimental kernel programming language designed to model and program distributed concurrent applications with code mobility [70]. The syntax of Klaim is presented in Figure 3.5.

Klaim relies on the concept of a distributed tuple space. A tuple space is a multiset of tuples. A tuple t is a container of information which can be either an actual value such as an expression e, process P, or a locality ℓ, or a formal field such as a value variable $!x$, process variable $!X$ or locality variables $!u$. An example of a tuple is: $(5, "person", !var)$, where 5 and $"person"$ are expressions and $!var$ is a value variable. Tuples are anonymous and Klaim uses pattern matching to select tuples from a tuple space.

A node contains one tuple space and processes. Nodes can be identified through

N	$::=$		Node
	$\mid$	0	Empty net
	$\mid$	$s ::_\rho P$	Single node
	$\mid$	$N_1 \parallel N_2$	Net composition
P	$::=$		Process
	$\mid$	*nil*	Null process
	$\mid$	$act.P$	Action prefixing
	$\mid$	$P_1 + P_2$	Choice
	$\mid$	$P_1 \mid P_2$	Parallel composition
	$\mid$	X	Process variable
	$\mid$	$A\langle \tilde{P}, \tilde{\ell}, \tilde{e} \rangle$	Process invocation

$act ::=$ out(t)@ℓ | in(t)@ℓ | read(t)@ℓ | eval(P)@ℓ | newloc(u)

$t ::= e \mid P \mid \ell \mid !x \mid !X \mid !u \mid t_1, t_2$

Figure 3.5: Syntax of the Klaim language [70]

two types of addresses: sites s and localities ℓ. Sites are absolute identifiers through which nodes can be uniquely identified within a net and localities are symbolic names for nodes and have a relative meaning depending on the node where they are interpreted. Localities are associated with sites through allocation environments ρ, represented as partial functions on each node.

Klaim processes may run concurrently and can perform five basic operations over nodes. Three of them, $in(t)@\ell$, $read(t)@\ell$, $out(t)@\ell$ are used to manipulate the tuples, $newloc(u)$ creates a new node and $eval(P)@\ell$ spawns a process P for execution at node ℓ.

3.5.2 Syntax of the Portunes language

As with other members of the Klaim family, the syntax of the Portunes language consists of nodes, processes and actions. The Portunes language lacks the tuple spaces and the actions associated with tuple spaces, which are present in the Klaim family of languages, and focuses on the connections between nodes. This is because connectivity is the main interest from the perspective of security modeling. The Portunes language is also simplified by removing variables and localities, because our goal is to automatically generate programs rather than program them.

N	::=		Node
	|	$l ::_s^\delta P$	Single node
	|	$N_1 \parallel N_2$	Net composition
P	::=		Process
	|	*nil*	Null process
	|	$P_1 \mid P_2$	Process composition
	|	$a^l.P_1$	Action prefixing
a	::=		Action
	|	*login(l)*	Login
	|	*logout(l)*	Logout
	|	*logout'(l)*	Logout'
	|	*eval(P)@l*	Spawning

Figure 3.6: Syntax of the Portunes language

The syntax of the Portunes language is shown in Figure 3.6. A single *node* $l ::_s^\delta P$ consists of a name $l \in \mathcal{L}$, where $\mathcal{L}$ is a universe of node names, a set of node names $s \in 2^{\mathcal{L}}$, representing nodes that the node l contains , a low-level security policy δ and a process P. The relation between the Portunes graph and the expressions in the Portunes language is intuitive: a node l in the graph represents a node with name l in the language, an edge (l, l') in the graph connects l to a node name $l' \in s$ of the node $l ::_s^\delta P$. Thus, the node name uniquely identifies the node in the graph, while the set s defines which other nodes the node *contains* or *is a neighbor of*. These two relations identify the relative location of each element in the environment. A net is a composition of nodes.

A *process* P is a composition of actions. Namely, nil stands for a process that cannot execute any action and $a^l.P_1$ for the process that executes action a using privileges from node $l \in \mathcal{L}$ and then behaves as P_1. The label l identifies a node from where the privileges originate, and it is termed the origin node. The structure $P_1|P_2$ is for parallel composition of processes P_1 and P_2. A process P represents a task. A node can perform a task by itself or delegate the task to another node. Recursive and mutually recursive process definitions are not allowed in the Portunes language. Thus, every behavior described using the language has to be finite.

An *action* a is a primitive which manipulates the nodes in the language. There are four primitives, $login(l)$, $logout(l)$, $logout'(l)$ and $eval(P)@l$. The actions $login(l)$ and $logout(l)$ provide the mobility of a node, by manipulating the set s. The action $logout'$ restricts in the mobility of a node by checking whether the

node is allowed to move, but does not manipulate the set s. The action $eval(P)@l$ delegates a task P to a node l by spawning a process in node l.

Notation To simplify the representation of processes, the nil at the end of non-empty processes is omitted. For example, instead of writing the process as: $logout(hall)^{employee}.login(secureRoom)^{employee}.nil$, we write the process as: $logout(hall)^{employee}.login(secureRoom)^{employee}$. If the process has multiple actions from the same origin, then we put the origin node at the end of the process rather than after each action. The process from the above example will be represented as: $[logout(hall).login(secureRoom)]^{employee}$.

Example: Road apple attack (continued)

For a node representing a room, $secureRoom ::_s^\delta nil$, the low-level policy δ defines the conditions under which other entities can enter or leave the secure room. The set s contains the names of all objects that are located in the room and the names of the locations neighboring the room. Let an insider and an employee be in a hall $hall ::_{\{insider,\, employee,\, secureRoom\}}^\delta nil$ which is neighboring the secure room. An insider delegating a task to the employee is: $insider ::_s^\delta eval(P)@employee^{insider}$ where P is a process denoting the task, $employee$ is the node to which the task is delegated and the label $insider$ is the origin node. An employee entering the secure room as part of the task delegated from an insider is presented through $employee ::_s^\delta login(secureRoom)^{insider}.P'$, while an employee leaving the room $employee ::_s^\delta logout(secureRoom)^{insider}.P''$. This example shows that the actions $login$ and $logout$ are abstractions of objects leaving or entering locations. The same actions can be used to specify objects being put into or removed from other objects. To keep the level of abstraction sufficiently high and consistent with the constructs presented by Bettini et al. [19], the action names are generic rather than named specifically, such as "put/take" or "enter/leave".

An origin node can grant a set of capabilities $C = \{ln, lt, e\}$ to another node, where ln is a capability to execute the action $login$, lt to execute the action $logout$ or $logout'$ and e to execute the action $eval$. Which capabilities the origin node can grant depends on its identity, location and credentials. The low-level security policy δ is a function $\delta : (\mathcal{L} \cup \{\bot\}) \times (\mathcal{L} \cup \{\bot\}) \times 2^{\mathcal{L}} \rightarrow 2^C$. The first and the second parameter denote identity based access control and location based access control respectively. If the identity or the location does not influence the policy, it is replaced by $\bot$. The third parameter denotes credential based access control, which requires a set of credentials to allow an action. If a policy is not

affected by credentials, the third parameter is an empty set. A policy can present a situation where: 1) only credentials are needed, such as a door that requires a key $(\bot, \bot, \{key\}) \mapsto \{ln\}$, 2) only the identity is required, such as a door that requires biometrics information $(John, \bot, \emptyset) \mapsto \{ln\}$, or 3) only the location is required, such as data that can be reached only locally $(\bot, \mathit{office}, \emptyset) \mapsto \{ln\}$. The low-level policy supports combinations of these attributes, such as a door requiring biometrics and a key $(John, \bot, \{key\}) \mapsto \{ln\}$. The policies focus on the allowed action, not of the content of the action. For example, the policy $(insider, \bot, \emptyset) \rightarrow \{ln\}$, at a node $employee$, states the employee trusts the insider sufficiently to accept any object from her. The least restrictive policy that can be used is: $(\bot, \bot, \emptyset) \mapsto \{ln, lt, e\}$.

We introduce types on nodes to define the spatial, object and digital layer on the nodes in the language. Typing also allows us to avoid impossible containment relationships between nodes from the same layer, such as a node containing itself. Each node has a type $t \in T$, where T is a finite partially ordered set defined by the relation $\succ\!\!\succ_{ln}$. The function $\mathcal{T}$ maps a node to its type $\mathcal{T} : N \rightarrow T$. The relation $\succ\!\!\succ_{ln}$ provides ordering between nodes based on their type. As a convention, we write types with a capital first letter. For the road apple example, T is defined as $T = \{Room, Person\}$, and the ordering relation as $\succ\!\!\succ_{ln} = \{(Room, Person)\}$. The mapping between the nodes and their types is: $\mathcal{T}(secureRoom) = \mathcal{T}(hall) = Room, \mathcal{T}(employee) = \mathcal{T}(insider) = Person$. The ordering is not transitive: For example, a room can contain a dongle and a dongle can contain digital data. But, the room cannot contain the digital data. Also, the ordering is not reflexive: a dongle might not be able to contain a dongle, nor an insider can contain an employee. The only assumption on $\succ\!\!\succ_{ln}$ is that it does not invalidate invariant 7 in Definition 12, or put differently, the relation does not allow cycles between nodes in the digital layer.

Example: Road apple attack (continued)

In section 3.4 we introduced the Portunes graph of the environment where the road apple attack takes place. We defined the relation between the elements through a graph and their stratification in the graph through the function $\mathcal{D}$. Now, we additionally define the $\succ\!\!\succ_{ln}$ relation and the low-level policies on each of the nodes.

Figure 3.7 presents the environment as a net composition. The representation contains detailed information about the low-level policies in place, making them suitable for analysis. For example, the node $world$ has no processes (nil), contains the remote server, the insider and is neighboring the hall. The node also has one low-level policy $(\bot, \bot, \emptyset) \mapsto \{ln, lt\}$, which means every node is allowed

$$world ::^{(\bot,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{remoteServer,\, insider,\, hall\}} nil$$

$$\|\ \ hall ::^{(\bot,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{employee,\, secureRoom\}} nil$$

$$\|\ \ secureRoom ::^{(employee,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{secureServer\}} nil$$

$$\|\ \ remoteServer ::^{(\bot,\bot,\emptyset)\mapsto\{ln\}}_{\{\}} nil$$

$$\|\ \ insider ::^{(\bot,\bot,\emptyset)\mapsto\{ln,lt,e\}}_{\{dongle\}} P_1$$

$$\|\ \ employee ::^{(insider,\bot,\emptyset)\mapsto\{ln\}\,;\,(employee,\bot,\emptyset)\mapsto\{ln,lt,e\}}_{\{\}} P_2$$

$$\|\ \ secureServer ::^{(\bot,secureRoom,\emptyset)\mapsto\{ln,lt\}\,;\,(\bot,secureServer,\emptyset)\mapsto\{ln,lt\}}_{\{serverData\}} nil$$

$$\|\ \ dongle ::^{(\bot,\bot,\emptyset)\mapsto\{e\}\,;\,(dongle,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{rootkit\}} P_3$$

$$\|\ \ rootkit ::^{(dongle,\bot,\emptyset)\mapsto\{ln,lt,e\}}_{\{\}} P_4$$

$$\|\ \ serverData ::^{(\bot,secureServer,\emptyset)\mapsto\{e\}}_{\{\}} nil$$

Figure 3.7: The road apple attack environment in the Portunes language

to enter and exit the area that is out of the company premises. The policy at employee $(employee, \bot, \emptyset) \rightarrow \{ln, lt, e\}$ states the employee will accept all actions originating from herself. Removing this policy would prevent the node executing any action using its own privileges.

To reduce the number of nodes, the majority of the policies presented here are identity based. For example, the policy $(employee, \bot, \emptyset) \rightarrow \{ln, lt\}$ on $secureRoom$ requires the biometrics of the employee to identify itself when entering and leaving the room. In a less secure environment, the policy can be replaced with $(\bot, \bot, \emptyset) \rightarrow \{lt\}, (\bot, \bot, \{key\}) \rightarrow \{ln\}$ meaning that everyone can leave the room, but a person containing a key can enter.

The processes P_1, P_2, P_3 and P_4 describe intended behavior of the nodes. In Section 3.5.5 we show how these processes can describe a behavior that represents the road apple attack and in Chapter 4 generate them automatically.

The available types are $T = \{Space, Person, Server, ItObject, Data\}$. The mapping and the Hasse diagram are given in Figure 3.8 and Figure 3.9. The

$$\mathcal{T}(world) = \mathcal{T}(hall) = \mathcal{T}(secureRoom) = Space,$$
$$\mathcal{T}(employee) = \mathcal{T}(insider) = Person,$$
$$\mathcal{T}(remoteServer) = \mathcal{T}(secureServer) = Server,$$
$$\mathcal{T}(rootkit) = \mathcal{T}(serverData) = Data,$$
$$\mathcal{T}(dongle) = ItObject.$$

Figure 3.8: Type definition of the nodes

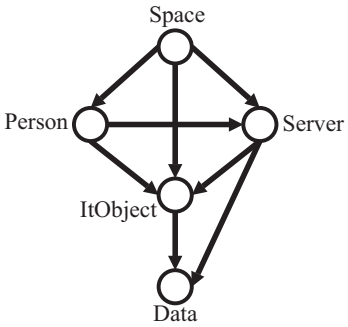


Figure 3.9: The Hasse diagram of the types of the nodes

ordering relation is:

$$\succ\!\succ_{ln} = \{(Space, Person), (Space, Server), (Space, ItObject), (Person, ItObject), (Server, ItObject), (Person, Server), (ItObject, Data), (Server, Data)\}$$

A net N is a formal representation of the environment. A net N, together with the mapping functions $\mathcal{D}$, $\succ\!\succ_{ln}$ on its nodes presents a single state of the environment and the processes P represent intentions of the nodes.

3.5.3 Auxiliary functions

Having defined the behavior of nodes using the three primitive actions, $login$, $logout$ and $eval$, we now look at the context where these actions can be executed. A node $l ::_s^\delta a^{l'}.P$ can be restricted in executing an action a from an origin node l' to a target node for four reasons: (1) the origin node might not have sufficient privileges, (2) execution of an action invalidates the invariants in Definition 12, (3) the target node might not be in the vicinity of the node l or (4) the target node is not physically able to contain the node. This section defines the auxiliary functions for a given net N, which take care of these restrictions. The auxiliary functions are defined in Figure 3.10 and are used in the operational semantics of the language.

$$grant(l_o, \delta_t, a) = \exists k_1, k_2 \in \mathcal{L} \cup \{\bot\}, \exists K \in \mathcal{P}(\mathcal{L}) : a \in \delta_t(k_1, k_2, K) \wedge$$
$$\underbrace{(k_1 = l_o \vee k_1 = \bot)}_{(1)} \wedge \underbrace{(k_2 \in parents_N(l_o) \vee k_2 = \bot)}_{(2)} \wedge \underbrace{(K \subseteq children_N(l_o))}_{(3)},$$
where $parents_N(l_o) = \{\, l_{po} |\, l_{po} ::^{\delta_{po}}_{s_{po}} R \in N \wedge\ l_o \in s_{po}\}$
and $children_N(l_o) = s_o$ such that $l_o ::^{\delta_o}_{s_o} R \in N$

$$l_t \succ_{ln} l = \begin{cases} false & \text{iff } (\mathcal{D}(l_t) = D \wedge (\mathcal{D}(l) = O \vee \mathcal{D}(l) = S) \vee \\ & \quad (\mathcal{D}(l_t) = O \wedge \mathcal{D}(l) = S) \vee (\mathcal{D}(l_t) = S \wedge \mathcal{D}(l) = D) \\ \mathcal{T}(l_t) \succ\!\succ_{ln} \mathcal{T}(l) & otherwise \end{cases}$$

$$l \succ_e l_t = \underbrace{(\mathcal{D}(l) \neq L \wedge \mathcal{D}(l_t) \neq L)}_{(4)} \wedge \underbrace{\neg(\mathcal{D}(l) = D \wedge \mathcal{D}(l_t) = O)}_{(5)}$$
$$\wedge\, (\underbrace{l_t \in children_N(l)}_{(6)} \vee \underbrace{(\exists l_p ::^{\delta_p}_{s_p} R \in N : l \in s_p \wedge l_t \in s_p)}_{(7)} \vee \underbrace{\mathcal{D}(l_t) = D)}_{(8)}$$

Figure 3.10: Auxiliary function $grant$ and $\succ$ relations

The $grant$ function checks if an origin node l_o has sufficient privileges to execute an action a on a target node with low-level policy δ_t. The first parameter is the name of the origin node l_o, the second parameter is the low-level policies on the target node δ_t and the third parameter is a label of an action a. A node can execute an action depending on the identity l_o of the origin node (1), its location $parents(l_o)$ (2) or the keys $children(l_o)$ it contains (3). Note that the value of $grant$ depends solely of the origin node, not the node executing the process.

The relation $l_t \succ_{ln} l$ states that a node l_t can contain a node l. The goal of this relation is to ensure the invariants 3-6 in Definition 12 are satisfied during the net evolution. From the relation we see that a digital node cannot contain a spatial or a physical node, an object node cannot contain a spatial node and a spatial node cannot contain a digital node.

The ordering relation $l \succ_e l_t$ states that node l can delegate a task to node l_t by means of spawning a process. The relation restricts delegation of tasks between nodes depending on the layer a node belongs to and the proximity between nodes. An object node can delegate a task to a digital node or another object node, while a digital node can delegate a task only to another digital node. Thus, spatial nodes cannot delegate tasks, nor can a task be delegated to spatial nodes (4), and digital nodes cannot delegate tasks to object nodes (5). Furthermore, a non-digital node can delegate a task only to nodes it contains (6) or nodes that are in the same

location (7). In digital nodes the proximity does not play any role in restricting the delegation of a task (8). The decision (8) assumes the world is pervasive and digital nodes can delegate tasks from any location as long as they have the appropriate privileges.

The expressions from Figure 3.10 focus on the relation between nodes. The $grant$ function provides the security constraints in the language based on the location and identity nodes, while the $\succ_{ln}$, $\succ\succ_{ln}$ and $\succ_{e}$ relations provide non-security constraints derived from the layer the nodes belong to and their location. In addition, we put a restriction on the processes inside a node, to distinguish tasks originating from a single node. We call such processes simple processes, and define an additional auxiliary function $origin$, which helps to determine if a process is a simple process.

Definition 13. *Let* $origin : Proc \rightarrow 2^{\mathcal{L}}$ *be a function which returns all the action labels of a given process.*

$$origin(nil) = \{\}$$

$$origin(a^l.P) = \{l\} \cup origin(P)$$

$$origin(P_1|P_2) = origin(P_1) \cup origin(P_2)$$

A process P*, which is either* nil *or which contains actions only from one origin node is a simple process:* $origin(P) \subseteq \{l_0\}$

In the semantics of the Portunes language this function forbids processes from one origin to spawn processes from other origins. For example, the process definition

$$insider ::_s^{\delta} eval(logout(hall)^{employee}.login(secureRoom)^{employee})@insider^{insider}$$

is not allowed, because both nodes $employee$ and $insider$ are origins of actions in the process. This process definition can be interpreted as: the insider delegates herself a task to enter the secure room using the privileges from the employee. The execution of this process does not require any interaction with the employee and does not represent a realistic scenario. We also found that the processes can be better mapped in real life behaviors if they execute actions only from a single origin. Naturally, a node can still execute other simple processes from other origins in parallel.

3.5.4 Operational semantics

Following Bettini et al. [19], the semantics of the Portunes language is divided into process semantics and net semantics. The process semantics is given in terms of a labeled transition relation $\xrightarrow{a}$ and describes both the intention of a process to perform an action and the availability of resources in the net. The label a contains the name of the node executing the action, the target node, the origin node and a set of node names which identify which nodes the target node contains. The net semantics is given in terms of a transition relation $\Rightarrow$ which describes possible net evolutions and relies on the labeled transition relation $\xrightarrow{a}$ from the process semantics.

$$\frac{origin(P) \subseteq \{l_o\} \quad l_t \succ_{ln} l \quad grant(l_o, \delta_t, ln)}{l ::_s^\delta login(l_t)^{l_o}.P \| \; l_t ::_{s_t}^{\delta_t} Q \xrightarrow{login(l,l_t,l_o,s_t)} l ::_s^\delta P \| \; l_t ::_{s_t \cup \{l\}}^{\delta_t} Q} \quad \textbf{[login]}$$

$$\frac{origin(P) \subseteq \{l_o\} \quad grant(l_o, \delta_t, lt) \quad l \in s_t}{l ::_s^\delta logout(l_t)^{l_o}.P \| \; l_t ::_{s_t}^{\delta_t} Q \xrightarrow{logout(l,l_t,l_o,s_t)} l ::_s^\delta P \| \; l_t ::_{s_t \setminus \{l\}}^{\delta_t} Q} \quad \textbf{[logout]}$$

$$\frac{origin(P) \subseteq \{l_o\} \quad grant(l_o, \delta_t, lt) \quad l \in s_t}{l ::_s^\delta logout(l_t)^{l_o}.P \| \; l_t ::_{s_t}^{\delta_t} Q \xrightarrow{logout'(l,l_t,l_o,s_t)} l ::_s^\delta P \| \; l_t ::_{s_t}^{\delta_t} Q} \quad \textbf{[logout']}$$

$$\frac{origin(P) \subseteq \{l_o\} \quad origin(Q) \subseteq \{l_o\} \quad l \succ_e l_t \quad grant(l_o, \delta_t, e)}{l ::_s^\delta eval(Q)@l_t^{l_o}.P \| \; l_t ::_{s_t}^{\delta_t} R \xrightarrow{eval(l,l_t,l_o,Q)} l ::_s^\delta P \| \; l_t ::_{s_t}^{\delta_t} R|Q} \quad \textbf{[eval]}$$

$$\frac{l ::_s^\delta P \xrightarrow{a} l ::_s^\delta P'}{l ::_s^\delta P|Q \xrightarrow{a} l ::_s^\delta P'|Q} \quad \textbf{[pComp]}$$

Figure 3.11: Process semantics

The process semantics of the language is defined in Figure 3.11. A node l can login to node l_t **[login]** if it has sufficient privileges to perform the action ($grant$), if the node can be contained in the target node ($\succ_{ln}$) and if the process is a simple process with origin node l_o ($origin$). As a result of executing the action, node l enters node l_t, or put differently, the target node l_t now contains node l.

For a node to logout from a target node **[logout]**, the target node must contain the node ($l \in s_t$), the origin node must have proper privileges ($grant$) and the process must be a simple process with origin node l_o ($origin$). The action results in l leaving l_t, specified through removing its node name from s_t. The rule **[logout']**

has the same premises but does not remove the node l from l_t. The **[logout']** is needed in the net semantics of the language in Section 3.5.5 where the data nodes can be copied rather than moved from one node to another.

Spawning a process **[eval]** requires both the node executing the action and the target node to be close to each other or the target node to be digital ($l \succ_e l_t$), the origin node should have the proper privileges ($grant$) and both processes P and Q need to be simple processes with origin node l_o ($origin$). The action results in delegating a new task Q to the target node, which contains actions originating from the same origin node as the task P. Note that for delegation to occur, in the Portunes language it is sufficient for the employee (delegatee) to trust the insider (delegator), rather than requiring mutual trust between them. The reason behind this design decision is that we are interested in whether the insider can convince the employee to execute a task, rather than whether the insider trusts the employee.

$$\frac{N \xrightarrow{eval(l,l_t,l_o,P)} N_1}{N \xRightarrow{neteval(l,P,l_t)} N_1} \textbf{ [neteval]} \qquad \frac{N_1 \xRightarrow{a} N_1'}{N_1 \parallel N_2 \xRightarrow{a} N_1' \parallel N_2} \textbf{ [nComp]}$$

$$\frac{N \xrightarrow{logout'(l,l_{t_1},l_o,s_{t_1})} N_1 \quad N_1 \xrightarrow{login(l,l_{t_2},l_o,s_{t_2})} N_2 \quad \mathcal{D}(l) = D}{N \xRightarrow{netcopy(l,l_{t_1},l_{t_2})} N_2} \textbf{ [netcopy]}$$

$$\frac{N \xrightarrow{logout(l,l_{t_1},l_o,s_{t_1})} N_1 \; N_1 \xrightarrow{login(l,l_{t_2},l_o,s_{t_2})} N_2 \; (l_{t_1} \in s_{t_2} \vee l_{t_2} \in s_{t_1} \vee \mathcal{D}(l) = D)}{N \xRightarrow{netmove(l,l_{t_1},l_{t_2})} N_2} \textbf{ [netmove]}$$

Figure 3.12: Net semantics

3.5.5 Net semantics

The net semantics in Figure 3.12 uses the process semantics to define the possible actions in the Portunes language. Spawning a process is limited solely by the process semantics **[neteval]**.

To move, a node executes the logout and login actions in sequence **[netmove]**. Both actions should have the same origin node and should be executed by the same node. Furthermore, an object node can move only to a node in its vicinity, while digital nodes do not have this restriction ($l_{t_1} \in s_{t_2} \vee l_{t_2} \in s_{t_1} \vee \mathcal{D}(l) = D$).

Data can be copied, which is presented by data entering a new node without leaving the previous **[netcopy]**. Although the data can be copied, it still needs permis-

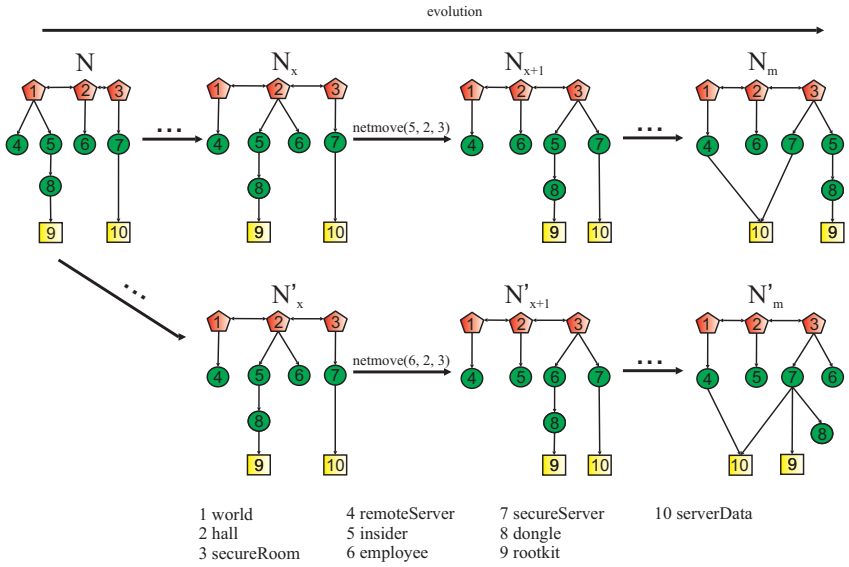


Figure 3.13: Example of a net evolution

sion from both the node it resides at l_{t_1} and from the node it is copied to l_{t_2}.

A net and two possible behaviors are presented in Figure 3.13. Both of these evolutions lead to the insider obtaining the server data. The nets and the net evolutions together present a Portunes model of the environment.

The standard rules for structural congruence apply and are presented in Figure 3.14.

$$\begin{array}{ll} \text{(ProcCom)} & P_1|P_2 \equiv P_2|P_1 \\ \text{(NetCom)} & N_1\|N_2 \equiv N_2\|N_1 \\ \text{(Abs)} & P_1|nil \equiv P_1 \end{array}$$

Figure 3.14: Structural congruence of processes and nets

Definition 14. *Vicinity of a node l with a parent node l_{t_1} is defined by all nodes l_{t_2} that share the same parent node ($l_{t_2} \in s_{t_1}$) or the child of l_{t_2} is a parent of l: ($l_{t_1} \in s_{t_2}$).*

Proposition 1. *A node from the object and spatial layer $l_{t_1}::^{\delta_1}_{s_{t_1}} P_1$ can move only to a node $l_{t_2}::^{\delta_2}_{s_{t_2}} P_2$ in its vicinity.*

Proof. The proposition follows directly from the $netmove$ premise: $l_{t_1} \in s_{t_2} \vee$

$l_{t_2} \in s_{t_1}$. □

Proposition 2. *Nodes from the object and spatial layer can evaluate processes only to child and sibling nodes.*

Proof. The property follows directly from the premise of the $eval$ action: $\succ_e$. □

Theorem 2. *Let $(G, \mathcal{D})$ be a Portunes graph and N be a net the represents the same environment. The function Map maps a net in a Portunes graph, such that $I(Map(N), \mathcal{D})$ holds. The evolutions of the net N do not invalidate the invariants I.*

Proof. Suppose there is a net N_1 which satisfies the invariants $I(Map(N_1), \mathcal{D})$. Suppose exists a net N_2 which is a product of a net transformation on N_1. $\exists N_2 : N_1 \Rightarrow N_2$. We need to prove that $I(Map(N_2), \mathcal{D})$ also holds.

The relation $\Rightarrow$ is used in the net actions $neteval$, $netcopy$ and $netmove$.

1. $neteval$ does not cause any changes of the structure of the net. Thus any execution of $neteval$ cannot invalidate an invariant.
2. $netmove$ removes an edge (l_{t_1}, l) and generates a new one (l_{t_2}, l). We need to show that the $\xrightarrow{login(l,l_{t_2},l_o,s_{t_2})}$ action does not invalidate any invariant.
   (a) Let $\mathcal{D}(l) = O$. After $\xrightarrow{logout(l,l_{t_1},l_o,s_{t_1})}$, $indegree(l) = 0$. Every $logout$ action is accompanied by a $login$ action. When $\xrightarrow{login(l,l_{t_2},l_o,s_{t_2})}$ is applied, $indegree(l) = 1$. Thus, invariant 1 is not invalidated.
   (b) Let $\mathcal{D}(l) = O$. After $\xrightarrow{login(l,l_{t_2},l_o,s_{t_2})}$ is applied, from $\succ_{ln}$, $\mathcal{D}(l_{t_2}) = L$ or $\mathcal{D}(l_{t_2}) = O$. The former case does not invalidate the second invariant by definition. Since $I(Map(N_1), \mathcal{D})$, $\exists m \in Node : \exists \langle m...l_{t_2} \rangle \wedge \mathcal{D}(m) = L$, the latter case also does not invalidate the second invariant.
   (c) The invariants 3, 4, 5 are not invalidated by the definition of $\succ_{ln}$.
   (d) The last invariant is not invalidated because of the assumption in $\succ\succ$.
3. The effect of $netcopy$ is an additional edge in the graph edge (l_t, l) generated by the relation $\xrightarrow{login(l,l_t,l_o,s_t)}$. The premise of $netcopy$ enforces a restriction $\mathcal{D}(l_t) = D$. Additional restriction comes from the relation $\succ_{ln}$, which allows an edge to be generated only between a node from the object and digital layer $\mathcal{D}(l) = D \wedge \mathcal{D}(l_t) = O$ or between two nodes from the

P_1=$logout(world).login(hall).$ (a)
$eval(logout(insider).login(hall).logout(hall).$
$login(employee))@dongle$ (b)
P_2=$logout(hall).login(secureRoom).$
$eval(logout(employee).login(secureRoom).$
$logout(secureRoom).login(secureServer))@dongle.$ (c)
P_3=$eval(logout(dongle).login(secureServer))@rootkit$
P_4=$eval(logout'(server).login(remoteServer))@serverData$

Figure 3.15: Process definitions enabling the road apple attack

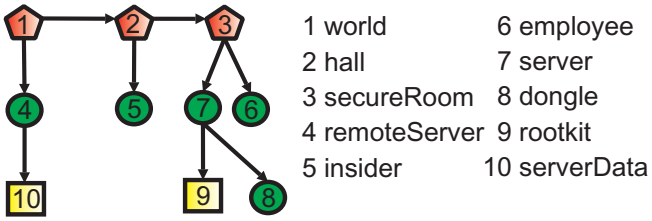


Figure 3.16: Portunes graph of the road apple attack environment after the execution of the attack

digital layer $\mathcal{D}(l) = D \wedge \mathcal{D}(l_t) = D$. The former does not invalidate any of the invariants, while the latter is restricted by the assumption on $\succ\succ$.

□

Example: Road apple attack (continued) In Section 3.5.2 we formally specified the environment where the road apple attack occurs. By using the language semantics it is now possible to reason about possible behaviors. A behavior is presented through defining the processes in the nodes, that lead to violating a high-level policy.

Figure 3.15 shows an example of the actual road apple attack as four processes, P_1, P_2, P_3 and P_4. All actions in the process P_1 have an origin node $insider$, in P_2 an origin node $employee$, in P_3 an origin node $dongle$ and in P_4 an origin node $rootkit$. For clarity, the labels on the actions representing the origin node are omitted from the process definitions.

The insider (P_1) goes in the hall and waits for the employee (process P_1 until reaches point a). Then, the insider gives the employee the dongle containing the rootkit, which the employee accepts (P_1 reaches b). Later, the employee plugs the dongle in the secure server (P_2 reaches c) using its own credentials and the server

gives the dongle (P_3) access to the local data. When the rootkit (P_4) reaches the server, it copies all the data to the remote server. The above actions represent the road apple attack with a dongle automatically running when attached to a computer [12]. After executing the processes from Figure 3.15, the data will reside in the remote server, presented through an edge $(remoteServer, data)$ in the Portunes graph in Figure 3.16. The next step is to generate these processes automatically, which is the focus of Chapter 4.

3.6 Conclusion

The main contribution of this chapter is the mapping of security aspects of the physical and social domain together with the digital domain into a single framework named Portunes. This formalization allows generating and analyzing attack scenarios which span all tree domains, and thus helps in the protection against insider threat. The framework consists of a high-level graph and a language inspired by the Klaim family of languages. To capture the three domains, Portunes is able to represent 1) physical properties of elements, 2) mobility of objects and data, 3) identity, credential and location based access control and 4) trust and delegation between people.

The applicability of Portunes is demonstrated using the example of the road apple attack, showing how an insider can attack without violating existing security policies by combining actions from all three domains.

Chapter 4

Analyzing Portunes models

In Chapter 3 we introduced a framework for representing behaviors that span the physical, digital and social domain. We used this framework to present malicious behaviors of insiders, which use the trust of their colleagues and gaps in the low-level polices to achieve their goal. In this chapter we generate such behaviors, allowing the security professional a clear overview of the possible ways in which a policy can be violated. We present an algorithm that finds in polynomial time all possible actions allowed by the low-level policies. We also provide two algorithms that generate from the actions a behavior that leads to achieving a specific goal. In this chapter a specific goal is defined informally by selecting an attribute. In Chapter 4 we show how to specify arbitrary goals using modal logic formulae.

4.1 Introduction

In the previous chapter we provided an abstraction of the physical, digital and social security domains in a single formal framework, Portunes. The framework is able to describe specific aspects, such as mobility and delegation, from the three security domains and allows their formal analysis. The analysis of these aspects provides information on how the security domains interact with each other and allows drawing conclusions on the overall security of the organization.

The goal of the analysis presented in this chapter is to check an environment formally and to reveal possible malicious behaviors. When a behavior leads to

achieving a malicious goal, then we call it an attack. The analysis achieves this goal by finding a sequence of actions that are allowed by the low-level policies, but still result in the violation of a high-level policy. Finding and analyzing a behavior is challenging because (1) it is computationally expensive to find all actions that can occur in the model and (2) the number of possible sequences of actions grows exponentially with each new action. For unsecured environments there can be an overwhelming number of behaviors that lead to an attack, making the analysis of each of them unfeasible. The analysis presented in this chapter is aimed at models of environments that are already considered as secure, such that a manageable number of attacks can be expected.

The analysis consists of three algorithms executed in sequence. The first algorithm finds the actions that can be executed by all nodes in the model. The second generates a partial behavior by combining actions into process definitions in such a way, that they lead to violating a policy. Both algorithms use the monotonicity assumption which states that an action cannot be invalidated by another action [14] (described in greater detail in Section 4.4). Because of the monotonicity assumption, a behavior generated by the second algorithm might miss a number of actions. The third algorithm shows how the monotonicity assumption can be lifted by adding the missing actions and thus generating a realistic behavior.

The worst-case computational complexity of the analysis is $O(N^4)$ where N is the number of nodes in the model. We implemented the semantics of the Portunes language and the analysis in an open source tool. We compare the performance of the first algorithm with a general purpose model checker, Groove. The first algorithm is the most computationally intensive part of the analysis, because it needs to find all possible actions that can be executed in the model under the monotonicity assumption. The second and the third algorithm select a part of the found actions that lead to the satisfaction of a goal. In the benchmarks we measure the time to find all possible actions on a number of Portunes models.

The rest of the chapter is structured as follows. In Section 4.2 we provide related work in generating attacks and in Section 4.3 we introduce the terms we use throughout the chapter. In section 4.4 we provide the algorithms for finding all possible actions, and generate a specific behavior that invalidates a goal. Section 4.6 describes the implementation of the algorithms in a tool and section 4.7 provides experimental data on the complexity of the first algorithm because this algorithm consumes the most time during the analysis. Section 4.8 concludes this chapter.

4.2 Related work

Using graphs to produce and describe multi-step attacks in computer networks is a well researched area. Previous research shows that such analysis can be done efficiently and effectively in the digital domain (for example: [14, 106, 74]). We contribute to this research area by describing and implementing an analysis capable of finding a malicious behavior in multi-domain models. The Portunes framework allows modeling social and physical aspects of security, enabling the generation of behaviors for malicious insiders in more realistic than purely digital environments, where the insider can use also physical and social means to achieve her goal.

Producing multi-step attacks in multi-domain models is a less researched area. Probst et al. [84, 85] propose a formal model for describing scenarios that span the physical and digital domain. This model allows an analysis to find which users are able to reach a certain location, based on their identity and knowledge. Due to the usage of process and object variables in the model, and not using the monotonicity assumption, the worst-case computational complexity of this analysis is exponential.

Kotenko et al. [62] also propose a model for describing attacks that use social engineering and physical access using preconditions and postconditions of atomic actions. However, the performance analysis of this approach indicates an exponential complexity based on the number of nodes in the graph, making it unfeasible for graphs bigger than about a hundred nodes.

4.3 Preliminaries

In this chapter we use the term model as it is used in the model checking community. A model consists of a set of states, which in our case are nets (for example Figure 3.7) and a set of state transitions. Each state transition is caused by the execution of an action from a process definition within a specific node and modifies the state as defined by the net semantics of the Portunes language presented in Section 3.5.5. In the analysis of this chapter, the states are not complete because the process definitions in the nodes are not known in advance but they are generated. A state that lacks process definitions is called a configuration, and the transition between two configurations is called a configuration transition. A configuration transition is an execution of one of the $netmove$, $netcopy$ or $neteval$ rules from the net semantics of the Portunes language.

An *action template* is a data structure that with preconditions that must be satisfied for a configuration transition to occur, and postconditions that specify the effects of the transition to the configuration [1]. The preconditions of a configuration transition are derived from the premises of the net rules and the postconditions of a configuration transition are derived from the results of the net rules. Each action template has a name, which states the process definition a node needs to execute to generate the configuration transition.

Both the preconditions and the postconditions in an action template consist of attributes. An $attribute$ represents a relationship between two nodes. In the Portunes language, attributes are a) containment, which states whether a node l_2 is logged into node l_1 (i.e. $l_1 ::_s^\delta P$ where $l_2 \in s$) and b) delegation, which states whether a node l_1 has a process definition originating from a node l_2 (i.e. $l_1 ::_s^\delta P$ where $l_2 \in origin(P)$).

For a single configuration, there can be a number of action templates with all their preconditions satisfied. If the precondition of a configuration transition is never invalidated by the successful execution of another configuration transition, the order of the execution of the configuration transitions does not influence the final configuration. During the analysis we use iteration numbers to determine which configuration transition can occur earlier than another configuration transition.

The initial configuration of the model has iteration 0. The configuration with iteration 1 is obtained when all possible configuration transitions are applied to the configuration at iteration 0. In general terms, the configuration N' resulting from the application of all possible configuration transitions at a configuration N is one iteration higher than the configuration N. In Section 4.4.1 we provide more intuition and an example of iterations and why are they needed in the analysis.

Example 1: An example of an action template derived from the $netmove$ rule is presented in Figure 4.1. To move the $serverData$ from $server$ to $remoteServer$, two attributes need to be satisfied: the $server$ needs to contain the $serverData$ and the $serverData$ needs to contain a process originating from $dongleData$. These two attributes are the preconditions of the action template. As a result of the execution of the action template, two attributes change: $server$ does not contain the $serverData$ anymore and the $remoteServer$ now contains the $serverData$. These two attributes are the postconditions of the action template. The iteration number shows that this action template was found during the eighth iteration of the model. The ninth iteration of the model will contain the postconditions from the action template as well as the postconditions of all the action templates that

[1] In the analysis of computer networks, action templates are called *exploits* [101, 102, 25, 26, 62].

action name:
$serverData :: [logout(server).login(remoteServer)]^{dongleData}$
precondition:
$server$ contains $serverData$
$dongleData$ delegates to $serverData$
postcondition:
$remoteServer$ contains $serverData$
$server$ does not contain $serverData$
iteration: 8

Figure 4.1: Example of an action template where the $serverData$ moves from the server to the $remoteServer$

had their preconditions satisfied in iteration eight.

4.4 Algorithms

The analysis in this chapter consists of three algorithms. The output of the analysis is a behavior. The behavior is made up of a sequence of actions that are allowed by the low-level security policies and satisfies a given goal. To determine which action is possible, the algorithms implement the rules from the operational and net semantics of the Portunes language.

The input-output relations of the three algorithms are presented in Figure 4.2. The first algorithm takes as input a Portunes model and returns a set of action templates that can be executed in the model. The second algorithm combines these action templates and using the monotonicity assumption to generate a partial behavior (partial attack) that invalidates a given goal. Finally, the third algorithm lifts the monotonicity assumption and adds missing action templates in the partial behavior.

The first two algorithms, inspired by Ammann et al. [14], consist of a forward marking stage and a backward attack finding stage. In the forward marking stage, the first algorithm starts from the initial configuration of the model and marks all action templates that can be executed. During the backward attack finding stage, the second algorithm begins from a goal, which is an attribute, and starts generating a behavior by linking action templates based on their preconditions and postconditions, until it reaches the initial configuration of the model. These two algorithms use the *monotonicity* assumption, which states that the precondition of

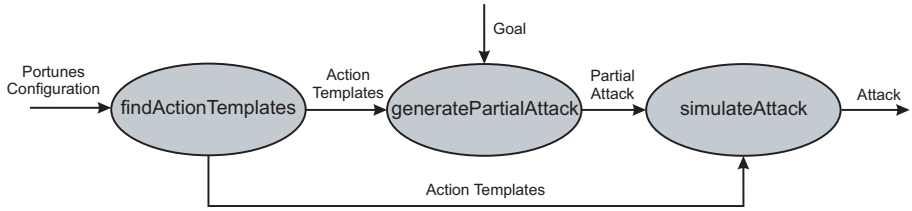


Figure 4.2: The input-output relations between the algorithms

a given configuration transition is never invalidated by the successful execution of another configuration transition. In the physical world, this assumption means that a person able to enter a room can never lose this ability, presenting the most pessimistic scenario where the adversary never loses a credential or the ability to reach a location. The *netmove* rule in the Portunes language via the rule *logout* invalidates an attribute and thus the monotonicity rule. Therefore, the *logout* rule is replaced with the *logout'* rule from Section 3.5.4, making the postconditions of *netmove* not invalidate any attribute. In Figure 3.12, the rule

$$\frac{N \xrightarrow{logout(l,l_{t_1},l_o,s_{t_1})} N_1 \; N_1 \xrightarrow{login(l,l_{t_2},l_o,s_{t_2})} N_2 \; (l_{t_1} \in s_{t_2} \vee l_{t_2} \in s_{t_1} \vee \mathcal{D}(l) = D)}{N \xRightarrow{netmove(l,l_{t_1},l_{t_2})} N_2} \textbf{ [netmove]}$$

becomes

$$\frac{N \xrightarrow{logout'(l,l_{t_1},l_o,s_{t_1})} N_1 \; N \xrightarrow{login(l,l_{t_2},l_o,s_{t_2})} N_2 \; (l_{t_1} \in s_{t_2} \vee l_{t_2} \in s_{t_1} \vee \mathcal{D}(l) = D)}{N \xRightarrow{netmove'(l,l_{t_1},l_{t_2})} N_2} \textbf{ [netmove']}$$

The monotonicity assumption leads to an over-approximation of possible behaviors, because it leads to finding action templates that might not be possible in real life.

Example 2: In realistic scenarios, physical objects can not be at two locations in the same time. Consider the following example: A restricted area and a control room are connected with a hall. In the hall there is a guard. The restricted area can be accessed only if a guard is in the control room. The guard cannot be simultaneously at both places, in front of the restricted area and inside the control room, thus it is not possible for him to enter the restricted area. The environment is presented with the following configuration:

$$restrictedArea ::^{(\bot,controlRoom,\emptyset)\to\{ln\}}_{\{hall\}} nil \;||\; controlRoom ::^{(guard,\bot,\emptyset)\to\{ln\}}_{\{hall\}} nil \;||\; hall ::^{(\bot,\bot,\emptyset)\to\{*\}}_{\{guard\}} nil \;||\; guard ::^{(\bot,\bot,\emptyset)\to\{*\}}_{\{\}} P$$

Because of the modification of the semantics of the $netmove$ rule, the following process definition P is now possible:

$P = [logout'(hall).login(controlRoom).logout'(hall).login(restrictedArea)]^{guard}$
The guard enters the control room because of the policy $(guard, \perp, \emptyset) \rightarrow \{ln\}$. Because of the monotonicity assumption, the guard will be both in the hall and the control room. The policy applied on the restricted area $(\perp, controlRoom, \emptyset) \rightarrow \{ln\}$ is satisfied because the guard is inside the control room. Simultaneously, the guard is also located in the hall allowing him to enter in the restricted area. Thus, because of the monotonicity assumption, there will be an action template requiring a physical node to be at two different locations ("$controlRoom$ contains $guard$" and "$hall$ contains $guard$") as part of its preconditions, which in reality is not possible.

Example 3: Another example when additional action templates are generated because of the monotonicity assumption is when a node gets locked in a certain location. In Example 2, let us assume the guard takes a credential from the control room. After entering the control room, the guard can never leave the room anymore, because there is no logout policy. In other words, the guard gets trapped in this location. Because of the monotonicity assumption, however, the guard is simultaneously in the hall too, so he can continue with other activities using the credential he obtained in the control room. These activities will generate additional action templates that are not possible in reality, because the guard is not able to exit the control room with the credential.

The monotonicity assumption, however, does not lead to missing any action templates. The only time the monotonicity assumption could cause missing of an action template is when the action template has a precondition that requires an attribute *not* to be satisfied [14]. In none of the actions templates we have a negative attribute as a precondition. This is because the policies do not contain negation (we cannot present policies where an absence of a credential allows an action) and none of the premises in the Portunes language requires an absence of a delegation or containment between two notes.

In Figure 4.1 the first and second algorithm will use only the first postcondition *remoteServer contains serverData* in the further iterations. The second postcondition, *server does not contain serverData* invalidates an attribute and is ignored.

Because of the monotonicity assumption, the set of action templates generated by the second algorithm does not include cyclic movements. A cyclic movement occurs when a node returns to a location it has previously been located into. In terms of the Portunes language, a cyclic movement occurs when a node N_1 moves away from a node N_2, and after a number of actions returns back to N_2.

Example 4: In the initial state of the road apple attack environment, the node $world$ contains the node $insider$. A simple example of a cyclic movement is:

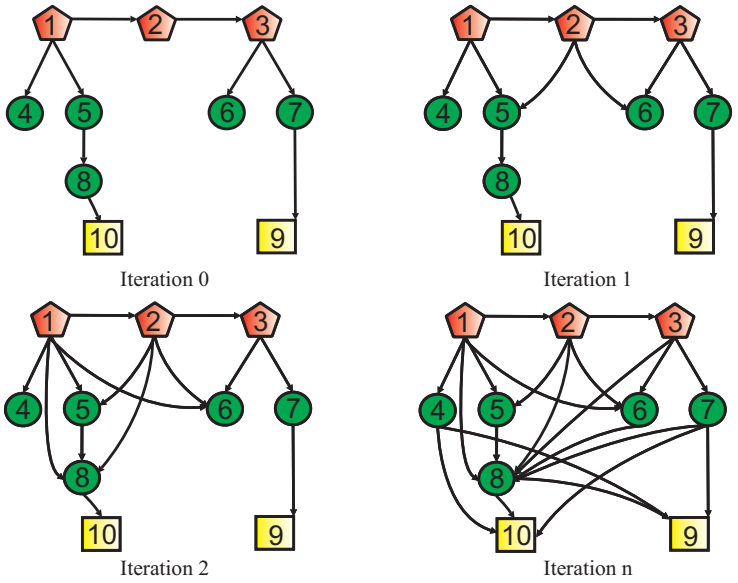


Figure 4.3: The containment relationships at the initial configuration, after the first iteration, the second iteration and after the last iteration.

$$insider ::_s^\delta [logout(world).login(hall).logout(hall).login(world)]^{insider}$$

After the movement of the insider to the hall, because of the monotonicity assumption, the second algorithm considers that both the hall and the world contain the insider.

The algorithm does not add additional action templates that return the insider in the world and thus cannot generate such a behavior. However, such behaviors are needed in situations where, for example, a person needs to go to a location to obtain a credential, and then return to a previously visited location to continue with the attack. The third algorithm shows how the effects of the monotonicity rule can be lifted and generates a realistic behavior that may include cyclic movement.

4.4.1 Intuition for the algorithms

This section continues the example from Chapter 3 and provides intuition how the analysis can be used to discover the road apple attack.

The first algorithm, *findActionTemplates*, finds all possible action templates that can be executed by the nodes $insider$, $employee$, $dongle$ and $dongleData$ in the road apple attack environment. Figure 4.3 shows the Portunes graph at the

P_1=[$logout'(world).login(hall).eval(logout'(insider).login(hall).$ (a)
$logout'(hall).login(employee))@dongle]^{insider}$ (b)
P_2=[$logout'(secureRoom).login(hall).$ (c)
$eval(logout'(employee).login(secureRoom).$ (d)
$logout'(secureRoom).login(server))@dongle]^{employee}$ (e)
P_3=[$eval(logout'(dongle).login(server))@dongleData]^{dongle}$ (f)
P_4=[$eval(logout'(server).login(remoteServer))@serverData]^{dongledata}$ (g)

Figure 4.4: Process definitions generated by *generatePartialAttack* enabling the road apple attack. The definitions are partial because they do not contain cyclic movement.

initial configuration, after the first iteration, after the second iteration and after the last iteration, when all found action templates are executed. The edges in the graph represent the initial contain relationships together with the effects from all found action templates. For example, the edge from $remoteServer$ (4) to $serverData$ (9) at iteration n means that $remoteServer$ at one point can contain $serverData$. In the first iteration four configuration transitions are possible, from which two are visible in the figure: the insider (5) can move to the hall (2), the insider can delegate a task to the dongle (8), the employee (6) can move to the hall, and the dongle can delegate a task to the rootkit (10). After the first iteration, because of the monotonicity assumption, the employee is both in the hall and the secure room (3), and the insider in the world (1) and the hall. In the second iteration another three configuration transitions are possible. The employee can move from the hall to the world and the dongle can move both to the world and the hall, because the insider is in both of the locations.

The *generatePartialAttack* algorithm uses the set of action templates generated by *findActionTemplates*, the initial configuration of the Portunes model and the goal: "$remoteServer$ contains $data$" to generate a partial attack scenario. To present them in the Portunes language as process definitions distributed among the net we need to perform two additional steps: 1) all action templates need to be sorted by the origin node of the actions they contain and 2) the action templates in every node in the net need to be ordered by iteration number. The first step defines in which node in the net the action from a template will be positioned and the second step orders the templates by the order of execution.

Figure 4.4 presents the distilled process definitions after merging action templates having actions with the same origin and ordering them by iteration number. All actions in process P_1 have the node $insider$ as an origin and after the first step will be located in the node $insider$. After the second step, these actions are ordered

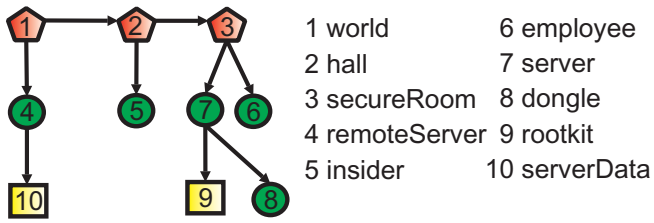


Figure 4.5: The Portunes graph after running the processes P_1-P_4.

by iteration number. In generating the templates, the templates that have lower iteration number are executed before the templates with a higher iteration number. Thus, the actions $logout'(world).login(hall)$ will be positioned in the process before or after the action $eval$ in the same process.

One interpretation of the actions is the following. The insider (P_1) goes in the hall and waits for the employee (process P_1 is then at point a). When the employee (P_2) arrives in the hall (P_2 at c), the insider gives him the dongle containing malicious software, which the employee accepts (P_1 at b). Later, the employee plugs the dongle in the secure server (P_2 at e) using its own credentials and the server gives the dongle (P_3) access to the local data. When the malicious software (P_4) reaches the server, it sends all the data to the remote server. The above actions closely resemble the road apple attack [122] with a dongle automatically running when attached to a computer [12] and covertly sending sensitive information [30, 31].

The resulting scenario defines only a partial attack. When the dongle reaches the employee, the dongle cannot move to the secure room, because the employee is located in the hall at that moment. Thus, the part of the attack scenario where the employee returns back to the secure room is missing. After running the *simulateAttack* algorithm, this cyclic movement is also included (bold font). In addition, since the *simulateAttack* uses the semantics of the $netmove$ rule, the actions using the $logout'$ rule are replaced with actions using the $logout$ rule:

$$P_2 = [logout(secureRoom).login(hall).eval(logout(employee).login(secureRoom).\\ logout(secureRoom).login(server))@dongle.\mathbf{logout(hall).login(secureRoom)}]^{employee}$$

After running the Portunes program, the final configuration of the Portunes model is given in Figure 4.5.

In the above example, the analysis combines physical, digital and social aspects of security. From the example, one can observe that enforcing a policy which forbids a server to accept remote connections is useless if there is no physical security

policy regulating which people can physically reach the server. An additional organizational policy should address dongle use among employees.

Having described the monotonicity assumption and the intuition of the algorithms in detail, now we present the three algorithms.

4.4.2 Algorithm I: Finding all action templates

name : *findActionTemplates*
type : actionTemplate:
$\langle actionname, postcondition, preconditions, iteration \rangle$
input : Queue of initially satisfied attributes $sAttributes$
output: Set of action templates $allTemplates$

```
1  begin
2      iteration = 0
3      while sAttributes ≠ ∅ do
4          a = pop an attribute from sAttributes
5          templates = all possible action templates t where
           a ∈ t.preconditions
6          foreach template t in templates do
7              if all t.preconditions are satisfied and consistent then
8                  if t.postcondition not satisfied then
9                      push t.postcondition onto sAttributes
10                 t.iteration = iteration
11                 allTemplates = allTemplates ∪ t
12         iteration++
```

Algorithm 1: Find all action templates

The first algorithm, *findActionTemplates* searches for action templates that exist in the model. This search answers the question: which attributes can be satisfied? The pseudo-code of the *findActionTemplates* algorithm is shown in Algorithm 1.

The *findActionTemplates* algorithm uses a bottom-up approach. As an input the algorithm has a queue of satisfied attributes. For each satisfied attribute, the algorithm finds the templates that have the attribute as a precondition (line 5). When all preconditions of an action template are satisfied and consistent (line 7), the (monotonic) postcondition of this template is added to the set of satisfied attributes

(line 9). The algorithm also keeps track of the iteration at which the template was found (line 10), which is used by the second algorithm.

The monotonicity assumption allows a physical node to be at multiple locations simultaneously. The preconditions of an action template are consistent if they do not require the same physical or spatial node to be simultaneously at two different locations. If an action template is inconsistent, the configuration transition can never be executed and the action template is ignored. An example of a situation where an action template requires the same physical node to be at two different locations is presented in Section 4.4, Example 2.

4.4.2.1 Termination

The termination of the algorithm depends on the $while$ loop in line 3. The algorithm terminates when all attributes that can contribute to the generation of an action template have been considered ($sAttributes = \emptyset$). An attribute is added to $sAttributes$ only if it is a result of a newly found action template and has not been satisfied before. The maximum number of attributes in the model is limited and in the worst case scenario can be N^2 delegate attributes plus $N \times (N-1)$ contains attributes. Thus, the *findActionTemplates* algorithm will terminate in maximum $2N^2 - N$ steps.

4.4.2.2 Complexity analysis

First, we present three assumptions regarding the policies in a model. We believe these assumptions are reasonable and realistic for secure environments, which are the target of the analysis. Then, we analyze every element in the algorithm that might contribute to its computational complexity.

Assumption 1: There is a constant number of nodes with a policy requiring a certain location or credential. We consider this assumption reasonable, because policies are defined based on requirements, not over a percentage of nodes. Thus, a policy can be set on a specific number of nodes which is not dependent from the total number of nodes. An example is a requirement where all rooms should be unlocked with a master key. In this case, the number of nodes with the same policy is equal to the number of rooms. However, the number of rooms does not depend on the number of total nodes in the model, but depend on the design of the building. During our modeling experience, we never encountered a model where the deployment of a specific policy depends on the number of nodes in the model.

Assumption 2: Policies require a constant number of credentials. It is unpractical to ask a user to present more than a few credentials when being granted a privilege. Formally, we assume for each policy $\delta_t(k_1, k_2, K)$, that the set K has a constant upper bound of elements. If this assumption does not hold, a policy might require up to N credentials before allowing an action, where N is the number of nodes in the model.

Assumption 3: There is an upper bound on the number of policies per node. We consider this assumption reasonable because in reality the number of policies on a security mechanism does not depend from the size of the environment. If this assumption would not hold, each node in the model can have up to $3 \times (N+1)^2 \times 2^{N+1}$ policies. There can be one policy for each of the capabilities, ln, lt and e, $(N+1)^2$ policies from all combinations of location and identity (including $\perp$), and 2^{N+1} policies from all combinations of credentials.

There are four points in the algorithm that are of interest for the complexity analysis: the while loop in line 3, the search for possible action templates in line 5, the foreach cycle in line 6 and finding all satisfied preconditions in line 7. We look closely at each of these points.

Attributes are added only once to the satisfied attributes queue. The complexity of the while loop in the algorithm is equal to the maximum number of attributes that can be satisfied for a given model. Thus, the complexity of the while loop in line 3 is $O(N^2)$.

In line 5, the algorithm searches for all action templates that have the newly satisfied attribute a as a precondition. There are three types of action template that can have the attribute a as one of their preconditions, each corresponding to one of the rules from the semantics of the Portunes language (Figure 3.12): $netmove'$, $netcopy$ and $neteval$.

Case I: $l :: [logout'(l_{t_1}).login(l_{t_2})]^{l_o}$ (netmove')
From the semantics of the $netmove'$ rule, the preconditions of the rule consist of the preconditions of the $logout'$ rule, the preconditions of the login rule, and the attributes l_{t_1} contains l_{t_2} (1) and l_{t_2} contains l_{t_1} (2). The $logout'$ rule has the same preconditions as the $logout$ rule: l_{t_1} contains l (3), l_o delegates to l (4) and the preconditions of the $grant$ function. The $grant$ function has the preconditions: l_o contains l_{co} (5) and l_{po} contains l_o (6). The $login$ rule has the precondition l_o delegates to l which is identical to the attribute (4) and the preconditions of the $grant$ function (5) and (6).

The attribute a contains information of two node names. If the attribute a is considered as one of the attributes (1), (2), (3) or (4), two of the nodes l, l_{t_1}, l_{t_2} or l_o are defined. There can be at most N^2 such action templates, one for every combi-

nation of the two unidentified nodes. If the attribute a is considered as one of the attributes (5) or (6), then the node l_o is known. Because of the first assumption, there are only a constant number of nodes l_{t_1} and l_{t_2} with policies that require attribute a. Knowing l_o and one of the nodes l_{t_1} or l_{t_2}, there are N^2 possible action templates with the attribute a as a precondition. The attribute a can be considered as any of the attributes mentioned above, thus all of them need to be checked. The attribute a can be considered as (1), (2), (3), (4), and (5), (6) for the policies at node l_{t_1} and attributes (5), (6) for the policies at node l_{t_2}. In total, there are $8 \times N^2$ $netmove'$ action templates that could have a as a precondition. As a result, the computational complexity of finding these templates is $O(N^2)$

Case II: $l :: [logout'(l_{t_1}).login(l_{t_2})]^{l_o}\ (netcopy)$
The $netcopy$ rule consists of the preconditions of the $logout'$ rule ((3), (4), (5), (6)) and the preconditions of the login rule ((5), (6)). These attributes are a subset of the attributes presented in Case I and the complexity analysis is identical. Thus, the complexity to find the $netcopy$ action templates that have the attribute a as a precondition is $O(N^2)$.

Case III: $l :: [eval(P)@l_t)]^{l_o}\ (neteval)$
From the semantics of the $neteval$ rule, the preconditions of the rule consist of the preconditions of the $eval$ rule. The $eval$ rule consists of the preconditions of the $grant$ function ((5),(6)), preconditions of the $l \succ_e l_t$ relation and l_o delegates to l (4). The relation $l \succ_e l_t$ consists of the attributes l contains l_t (7), l'_{po} contains l (8) and l'_{po} contains l_t (9).

If the attribute a is considered as one of the attributes (4), (5) or (6), the complexity analysis is similar as in Case I. If the attribute is considered as the attribute l contains l_t (7), there can be N possible $neteval$ action templates, one for every possible l_o. If the attribute a is considered as the attribute l'_{po} contains l (8), l is known, and there are N^2 possible $neteval$ action templates, for every combination of l_o and l_t. Similarly, if the attribute a is considered as attribute l'_{po} contains l_t (9), then l_t is known, and there are N^2 possible $neteval$ action templates, one for every l and l_o. Again, the worst case complexity for finding all possible $neteval$ action templates is $O(N^2)$.

From the three cases, the overall complexity of finding the action templates that have a specific attribute as an attribute is the complexity to find all $netmove'$ action templates ($O(N^2)$), all $netcopy$ action templates ($O(N^2)$) and all $neteval$ action templates ($O(N^2)$). Thus the computational complexity of line 5 in the algorithm is $O(N^2)$.

The *foreach* loop in line 6 traverses all found templates, which we showed can be at most $O(N^2)$. From the net semantics of the Portunes language and assumption

l_{t_1}	contains	l_{t_2}	(1)	l_{po}	contains	l_o	(6)
l_{t_2}	contains	l_{t_1}	(2)	l	contains	l_t	(7)
l_{t_1}	contains	l	(3)	l'_{po}	contains	l	(8)
l_o	delegates to	l	(4)	l'_{po}	contains	l_t	(9)
l_o	contains	l_{co}	(5)				

Figure 4.6: Attributes used in the semantics of the Portunes language

2, it follows that each action template has a constant number of preconditions. Thus, the number of preconditions that needs to be checked at line 7 is constant. If the satisfied attributes are implemented using hash tables, which have a constant lookup time the execution of the whole line 7 is constant. Therefore, the complexity of the whole algorithm, using the three assumption is $O(N^4)$.

The difference with the *markAttributes* algorithm proposed by Ammann et al. [14] is how the action templates are found. At every iteration, the *markAttributes* algorithm exhaustively searches for all combinations of attributes and action templates and for each action template checks whether its preconditions are satisfied. As a result, the worst case complexity of the algorithm is $O(A^2E)$, where A is number of attributes and E the number of action templates. For a given Portunes model, under the above assumptions, there can be a maximum of N^2 attributes and N^4 action templates (for each combination of node, parent, target, origin), which would make the complexity of this algorithm $O(N^8)$.

There are two main differences between the $markAttributes$ and *findActionTemplates*. First, in each iteration the *findActionTemplates* algorithm searches only for the attributes that were generated as a postcondition in the previous iteration, rather than all attributes. Second, The *findActionTemplates* algorithm searches only for action templates that have the newly satisfied attribute as a precondition, rather than all possible action templates. These two improvements decrease the computational complexity of the algorithm down to $O(N^4)$.

4.4.3 Algorithm II: Generating partial attacks

The first algorithm answers the question: *which* attribute can be satisfied? The second algorithm shows *how* an attribute can be satisfied. The pseudocode of the *generatePartialAttack* algorithm is shown in Algorithm 2.

Using the $generatePartialAttack$ it is possible to generate a sequence of actions that lead to a particular goal by backtracking from the goal to the initial situation, following the postconditions and preconditions of the action templates. The re-

sulting behavior is partial since it does not contain any cyclic movement of the nodes. An example of a cyclic movement is an insider going from a hall to a room to obtain a key, and returning to the hall to continue with the attack.

name : generatePartialAttack
input : Set of action templates $actionT$
input : Set of attributes $goals$
input : Set of satisfied attributes $sAttributes$
output: List of action templates representing a partial attack

1 **begin**
2 list of action templates $pResult = \emptyset$
3 int $maxItt$ = the maximum iteration found in $actionT$
4 return $find$ ($actionT$, $goals$, $sAttributes$, $maxItt$, $pResult$)

name : find
input : Set of action templates $actionT$
input : Set of attributes $goals$
input : Set of satisfied attributes $sAttributes$
input : Iteration at which the action template was found itt
input : List of action templates leading to the goal $pResult$
output: List of action templates representing a partial attack
5 **begin**
6 **foreach** *attribute goal in goals* **do**
7 find a template $s \in actionT$ with the smallest iteration number such that $s.postcondition = goal$ and $s \notin pResult$
8 **if** *there is no such template* **then** return error
9 $pResult.append(s)$
10 let p be a set of preconditions of s not in $sAttributes$.
11 **if** $p \neq \emptyset$ **then**
12 $sAttributes$ = $sAttributes \cup p$
13 $pResult$ = $find$ ($actionT$, p, $sAttributes$, $s.itt$, $pResult$)
14 return $pResult$

Algorithm 2: Generate a monotonic attack scenario

As output, the algorithm produces a set of action templates, contributing to a partial behavior. The behavior is partial because it does not contain action templates where the node needs to return to a previous location. For example, assume an insider located in a hall, which connects a room and a restricted area. The insider goes from the hall to the room to obtain a credential allowing her access to the re-

stricted area. Because of the monotonicity assumption, the insider can then move from the room directly in the restricted area because she is also still in the hall. If the monotonicity assumption is lifted, additional action templates are required where the insider returns from the room to the hall.

The algorithm starts from the goal set and finds an action template of which the postcondition satisfies a goal. For each unsatisfied precondition of the action template, the algorithm recursively searches for an action template whose postcondition satisfies the attribute of interest.

To generate a behavior from a given list of action templates we adapt the algorithm *findMinimal* presented by Ammann et al. [14]. The algorithm takes as an input the action templates, *actionT*, generated by the *findActionTemplates* algorithm, a set of attributes, $goals$, that need to be satisfied, a set of initially satisfied attributes $sAttributes$ and an iteration number. The resulting list of action templates ordered by the iteration present partial attack scenario.

4.4.3.1 Termination

The $generatePartialAttack$ algorithm terminates when either no action template that leads to the satisfaction of a specific attribute can be used (line 8) or when all goals are satisfied (line 14). The termination of the algorithm depends on the depth of the recursion at line 13. At each recursion, the number of satisfied attributes and used action templates increases, reducing the number of possible action templates to choose from. Thus, the algorithm at one point will use all possible attributes and terminate. In the worst case scenario, the algorithm may generate an attack scenario where all attributes need to be satisfied which translates into $2N^2 - N$ recursion calls.

4.4.3.2 Complexity analysis

From the net semantics of the Portunes language it follows that each action template has a constant number of preconditions. Thus, the loop at line 6 is constant, because the goals are derived from the preconditions of an action template. Similarly, in line 10 there is a constant number of lookups to check whether a precondition of the action template belongs to the set of satisfied attributes. The satisfied attributes are implemented using hash tables, which have a constant lookup time. Finding a template at line 7 is also constant when the action templates are implemented using hash tables. Thus, the computational worst-case complexity of the algorithm is determined by the number of recursive calls. In the worst-case

scenario, the algorithm can be executed for every possible attribute, making the computational complexity of the algorithm $O(N^2)$.

4.4.4 Algorithm III: Simulating the attacks

Because of the monotonicity assumption, the set of action templates generated by *generatePartialAttack* does not include cyclic movements. The *simulateAttack* algorithm, adds additional action templates in the partial scenarios that generate cyclic movement of the nodes.

name : *simulateAttack*
input : Set of initially satisfied attributes $base$
input : List of action templates $pAttack$
input : Set of action templates $actionT$
output: List of action templates representing an attack scenario $results$

```
1  begin
2      list result = ∅
3      while pAttack ≠ ∅ do
4          a = the last template in pAttack
5          S = set of preconditions of a not part of base
6          if S = ∅ then
7              result.append(a)
8              change conditions in base based on nonmonotonic
               a.postcondition
9              pAttack = pAttack\a
10         else
11             set of action templates partial = {∅}
12             partial = find (actionT, S, base, a.iteration, partial)
13             pAttack.append(partial)
```

Algorithm 3: Simulate the attacks

The $simulateAttack$ algorithm uses a list of action templates $pAttack$, which are generated by the *generatePartialAttack* algorithm and a set of attributes satisfied from the initial configuration of the Portunes model, and returns a list of action templates which represent an attack scenario. The algorithm takes the last action template from the list and checks if its precondition is met. If all attributes in the precondition are satisfied, the algorithm executes the action from the template and

updates the attributes with the postcondition from the action template. If a precondition is not satisfied because it is invalidated by the execution of another action template, the algorithm uses the $find$ function of the $generatePartialAttack$ algorithm to generate a new partial scenario which tries to satisfy the precondition and continues the simulation. The variable $result$ is a list containing an attack scenario which might include cyclic movement and is semantically valid when translated into the Portunes language. The attack can then be translated to the Portunes language by grouping the actions by origin node.

4.4.4.1 Termination

The output of $generatePartialAttack$ algorithm is finite, thus $pAttack$ is a finite list. The $simulateAttack$ algorithm will terminate when the number of action templates in $pAttack$ gets exhausted (line 3). The only time we add action templates in $pAttack$ is at line 13, when we need to satisfy an invalidated precondition. We show in two steps that the algorithm terminates. First, we show that for an algorithm not to terminate, we need an action template with mutually exclusive preconditions. Second, we show that such action template does not exist.

Step I: We add an action template to the list $pAttacks$ at line 13 only when a precondition of the action template is invalidated. The satisfaction of the precondition can invalidate another already satisfied precondition. We distinguish three cases based on the mutual dependence of preconditions in an action template.
Case I: An action template has an independent precondition invalidated. In this case, the algorithm will generate a finite list of action templates that will satisfy the precondition.
Case II: An action template has the precondition A invalidated, and its satisfaction invalidates an already satisfied precondition B. In this case, the algorithm will first generate one trace that satisfies the invalidated precondition A and then an additional trace to satisfy the newly invalidated precondition B.
Case III: An action template that has a precondition A invalidated, and its satisfaction invalidates an already satisfied precondition B. The satisfaction of the newly invalidated precondition B invalidates the newly satisfied precondition A. When the algorithm reaches such action template, one of the two attributes will never be satisfied. The algorithm will find new action templates (line 12) and apply them (line 8) which will lead to the satisfaction of the invalidated attribute. When the invalidated attribute gets satisfied, the other attribute will get invalidated, causing the algorithm to search for additional new action templates and leading to an infinite loop.

Thus, the algorithm will not terminate only if there is an action template with two

mutually exclusive attributes A and B as a precondition to achieve a postcondition C:

1. Satisfaction of the attribute A requires the invalidation of the attribute B.
2. Satisfaction of the attribute B requires the invalidation of the attribute A.
3. Satisfaction of the attribute C requires the satisfaction of both A and B.

Step II: Only the $netmove$ template invalidates an attribute as a postcondition. Thus, the action template must have a precondition that requires a node to be at a specific location (requiring the node to move to the location), and another precondition that requires the same node to be at another location (requiring the node to move to the other location). The first algorithm eliminates the action templates that require the same physical or spatial node to be simultaneously at two different locations as a precondition, because these actions cannot be performed in realistic scenario. Thus, there are no such action templates that can lead to non-termination of the algorithm.

4.4.4.2 Complexity analysis

The maximum number of action templates in $pAttack$ is $2N^2$ because there can be maximum of N^2 containment relationships and N^2 delegations. Thus, the complexity caused by the while loop in line 3 is $O(N^2)$. For each action template, the algorithm might need to generate a cyclic movement and call the $find$ function in line 12. From the previous algorithm, the complexity of generating a partial scenario is $O(N^2)$. The rest of the actions in the algorithm are constant. Assuming the existence of only simple cycles, meaning there are no cyclic movements within the cyclic movements, the worst case complexity of the $simulateAttack$ algorithm is $O(N^4)$.

4.5 Correctness of the analysis

The monotonicity assumption reduces the computational complexity of the analysis from exponential to polynomial. In Section 4.4 we described both the requirements and the effect of using the monotonicity assumption. The monotonicity assumption requires that the invalidation of an attribute does not influence the execution of any other action template and as a result (1) the $findActionTemplates$ algorithm produces an over-approximation of possible action templates (Example

3 in Section 4.4 and (2) the $generatePartialAttack$ algorithm produces attack scenarios without cyclic behavior (Section 4.4.1).

To satisfy the requirement of the monotonicity assumption, we changed the semantics of the Portunes language, by replacing the $netmove$ rule with the $netmove'$ rule. In this section we show that the attacks generated by the analysis are correct with respect of the semantics presented in Chapter 3: (1) the over-approximation of action templates does not lead to attack scenarios and (2) the attack scenarios generated by the analysis may contain cyclic behavior.

The *simulateAttack* algorithm follows the net semantics of the Portunes language introduced in Chapter 3. Thus, an attack scenario produced by this algorithm can be reconstructed using the Portunes language. If the algorithm *generatePartialAttack* generates an attack scenario that cannot be executed using the Portunes semantics, algorithm *simulateAttack* will either add any missing action templates to make the attack semantically correct (lines 10-13) or, if this is not possible, the *generatePartialAttack* algorithm will return an error (line 8). Thus, the attack generated by the analysis follows the semantics of the Portuens language.

4.6 Implementation

We implemented Portunes in Java. Figure 4.7 illustrates a screenshot of the Portunes interface. The interface is able to draw a new graph from scratch, including the different kind of nodes, policies and relations. The steps for a successful run of the tool are given in the activity diagram in Figure 4.8. Below we give a description of the various aspects of the Portunes interface.

1. **Toolbar** The toolbar is used for loading and saving a graph. It also provides buttons to add the different kinds of nodes to the graph (object, spatial and digital). At the end are the buttons for running the algorithms. The first algorithm will give all possible actions using the monotonicity assumption. By first selecting an edge that represents an undesirable goal and then pressing the second algorithm button, Portunes will show a number of possible behaviors with the selected edge as a final configuration.

2. **Attack scenarios** This panel shows possible behaviors after a successful run of the second algorithm. The list is ordered by the number of actions required for the behavior to complete. By selecting a behavior one can see the behavior in a step-by-step execution using the panel at the bottom (5).

3. **Graph view** With the graph view one can edit a graph by dragging nodes

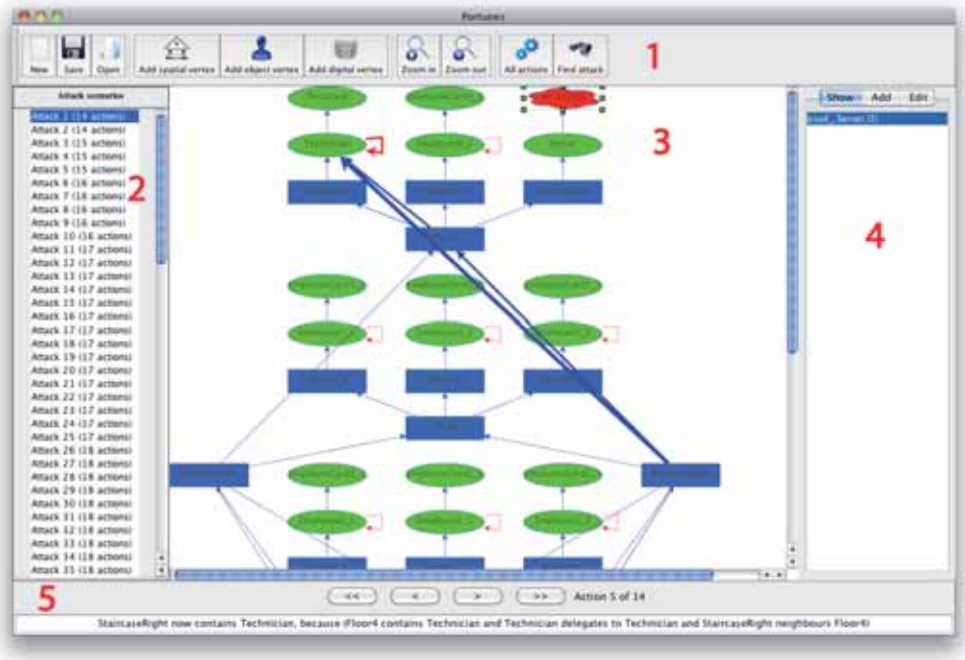


Figure 4.7: Screenshot of the implementation of Portunes

and drawing edges between them. The type of relation can be specified by right clicking a node and selecting the appropriate type. The graph view uses different colors and shapes for the different types of nodes and edges. The graph view is also used by the step-by-step execution to show each action in a graphical way by adding arrows to nodes that are involved in the current step.

4. **Policy panel** With the graph view it is possible to draw a complete configuration of the model, except for the policies. The Policy panel allows a user to define the different types of policies for a node (login, logout and eval). The panel is divided into three tabs used for giving an overview of the current policies, defining a new policy and editing an existing policy. A policy consists of an identity, location and zero or more credentials.

5. **Attack description** This panel shows a step-by-step textual description of a selected behavior. Each step is explained and the user can step back and forward through the behavior. With each step the graph view (3) changes to match the current step of the behavior.

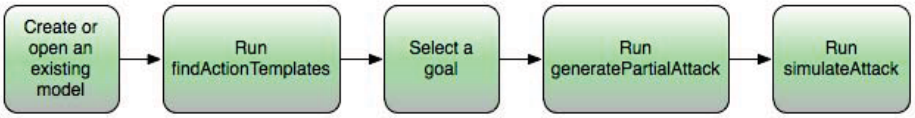


Figure 4.8: Activity diagram of the sequence of actions in the Portunes tool

4.7 Benchmark

The performance analysis is done to study the empirical complexity of the first algorithm and compare it to Groove, a general purpose model checker. The analysis was done on a Intel Core 2 Quad computer with CPU at 2.4 Ghz and 8 GB of RAM. Windows 7 64-bit was used as operating system with version 1.6 of the Java runtime environment installed. As a comparison, Groove version 4.0.2 was used.

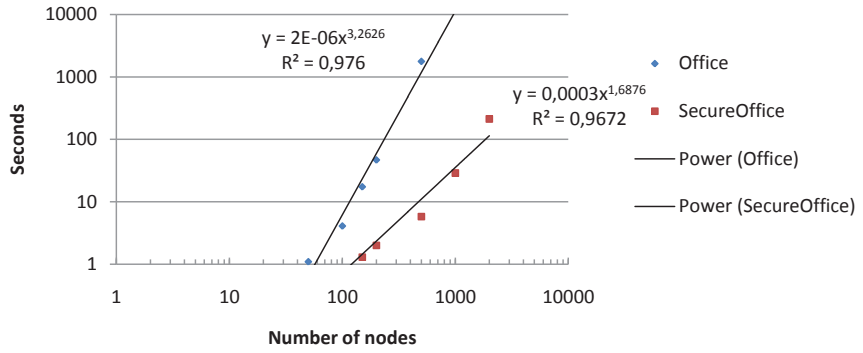
4.7.1 Groove

Groove[2] is a model checker that uses graphs for modeling the design-time, compile-time, and run-time structure of object-oriented systems, and graph transformations as a basis for model transformation and operational semantics. The tool allows encoding the Portunes semantics as a set of graph transformation rules. The graph models of Groove have the same benefit as Portunes, namely that the office model can be visualized as a graph.

4.7.2 Models

To get experimental data of the performance of the first algorithm and benchmark compared to Groove, we used a variation of the road apple attack inside an office model[3]. The model is scalable, as more floors, rooms and people can be added. We made a secure and insecure variant of the office model, by making more and less restrictive policies respectively. For example, in the insecure variant, the rooms and floors have no login restrictions, whereas the secure variant only allows card holders to enter the floors and each card holder can only enter one specific room, namely the one they start in. The insecure office model is the worst-case scenario where the organization has no explicit security mechanisms

[2] groove.cs.utwente.nl
[3] Available for download at portunes.sourceforge.net



	50	100	150	200	500	1000	2000
Office	1.1	4.1	17.4	46.8	1770.8		
SecureOffice	0.3	0.9	1.3	2.0	5.8	28.9	212.4

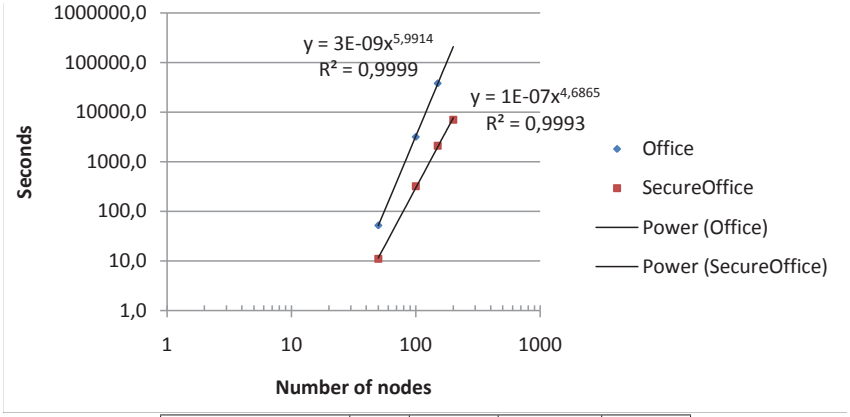
Figure 4.9: Time analysis for the findActionTemplates algorithm

that restrict user and data mobility, and the secure office model is an average-case scenario where the organization has taken actions to increase the security of the environment. The average-case scenario does not give information on the worst-case complexity, but it does show the speed for an architecture where proper security measures exist. We generated 7 models for the road apple example, containing 50, 100, 150, 200, 500, 1000 and 2000 nodes.

4.7.3 Results from the benchmark

The results of the time performance analysis are presented in Figure 4.9. To find the coefficients a and b from the complexity formula aN^b, we used power regression. The formula for the power regression curve of the Office data is $2 \times 10^{-6} N^{3.26}$ with correlation coefficient R^2=0.98[4]. The formula for the power regression curve of the SecureOffice data is $3 \times 10^{-4} N^{1.69}$ with correlation coefficient R^2=0.97. From the data, we can see that on average, the worst case complexity model grows $N^{3.26}$ with the increase of number of nodes in the model. For the SecureOffice model which represents a well secured organization this coefficient is much less, 1.69. As expected, the measured time complexity of Portunes for

[4]The correlation coefficient R^2 is the proportion of variance in a data set that is accounted for by the formula. The coefficient represents a measure of how well the formula represents the measured data. If the formula passes exactly through every point on the scatter plot then the coefficient would be 1. The further the line is away from the points, the closer the coefficient is to 0.



	50	**100**	**150**	**200**
Office	52	3160	37197	
SecureOffice	11	319	2100	7034

Figure 4.10: Time analysis for Groove

the worst-case model is below $O(N^4)$. The measured time complexity of Groove for the worst-case model is $O(N^{5.99})$.

The results show that generating a behavior quickly becomes infeasible using a general purpose model checker. Groove was unable to compute the office model with 200 nodes where Portunes could find all possible action templates in the office model 500 within an acceptable time frame ($\sim$30 minutes). Looking at the secure office model the difference becomes even more clear. Groove requires almost 2 hours for the 200 nodes model, where Portunes is able to find the same action templates for the 2000 nodes model in less than 4 minutes.

4.8 Conclusion

In this chapter we presented the algorithms to analyze attacks that span the physical, digital and social domain. We show that computing insider attacks can be done efficiently by splitting the work in two stages. In the first stage we compute all possible actions by using the monotonicity assumption. In the second stage we can recreate a specific attack by retracing the required steps. The performance analysis shows that for reasonably secure environments (i.e. *SecureOffice*) the algorithms can generate behaviors in a reasonable time (i.e. less than 4 minutes).

In this chapter we also described the Portunes tool, designed around the presented

algorithms. The tool shows that the model and the algorithms can indeed generate useful attack traces for a real life scenario. We tested multiple models with the tool, all describing different situations in order to test the algorithms.

The complexity of the second and third algorithm increases polynomially with the number of nodes. However, the number of possible behaviors grows exponentially with the action templates found by the first algorithm. Therefore, it is important to generate/select only those behaviors that are of interest to the security professionals. In the following chapter, we present a logic that specifies high-level policies formally. The logic can be used as a heuristic in the second algorithm in generating behaviors that satisfy defined properties, as well as to select specific behaviors from a set of generated behaviors.

Chapter 5

Expressing high-level policies in Portunes

In this chapter we present a temporal logic for describing high-level policies in the Portunes framework. The logic is inspired by Hennessy-Milner Logic (HML) and the modal logic for mobile agents. First, we provide requirements and motivating examples where we informally describe the properties of interest in a Portunes model. Second, we present the logic and show that it is sufficiently expressive to present these properties formally. The logic presented in this chapter serves three purposes: (1) from a modeling perspective, the logic enables definition of high-level policies, that should hold for the system as a whole, rather than on a specific object as was the case with the low-level policies in Chapter 3; (2) from an analysis perspective, the logic enables the description of a goal containing a conjunction or disjunction of multiple subgoals rather than a simple goal as was the case in Chapter 4; and (3) from a functionality perspective, the logic enables specification of subset of behaviors from a given set of behaviors, allowing the user to focus only on the set of behaviors that are of interest.

5.1 Introduction

In Chapter 3 we defined a language to describe a Portunes model and used the road apple attack as an example. However, we defined the goal of the road apple attack informally, saying that *"the data ends up at the remote server"*. In Chapter 4 we presented this property more formally, as a satisfaction of the attribute *"the node remoteServer contains the node serverData"*. In this chapter we present the goal as a formal property in the modal logic that should eventually hold in the Portunes model, $\langle\circ\rangle c(remoteServer, serverData)$. The formal presentation of properties in the model is suitable for defining complex goals because of the unambiguity of formal statements.

Modal logic is the most convenient formal tool to express properties of a Portunes model. Modal logic is usually used to specify and verify properties of concurrent models. Properties in these models are specified by means of temporal and spatial modalities. In Chapter 4 we presented algorithms to generate all possible behaviors that lead to the satisfaction of a single spatial property. In this chapter, we will present a new modal logic which can be used to express spatial and temporal properties of Portunes models.

We define the logic for Portunes in order to (1) describe adversarial goals and (2) describe high-level policies which should hold for all evolutions of a Portunes model. The logic for Portunes is primarily aimed to aid security auditors to assess whether high-level policies always hold in the organization. Portunes can search for a behavior where a person invalidates an high-level policy without invalidating any low-level policies specified on the nodes. The logic can also help penetration testers describe specific adversarial goal and isolate specific subsets of attack scenarios.

The rest of the chapter is structured as follows. In Section 5.2 we provide motivating examples to describe the properties of interest in a Portunes model. In Section 5.3 we provide an overview of related work and how the presented logic differs. Section 5.4 specifies the predicates for net and net evolutions and in section 5.5 we present the logic. Section 5.6 shows how the examples presented in Section 5.2 can be specified using the logic and in Section 5.7 we conclude the chapter.

5.2 Motivating examples

In this section we present the requirements for the logic. The requirements are distilled from observing a number of Portunes models obtained through a use case and a series of penetration tests. In the use case, we modeled a five story building and observed the low-level policies on individual objects and the general high-level policies. We also performed a series of physical penetration tests using social engineering (Chapter 6). From the attack traces obtained from the tests, we looked at which properties a penetration tester might be interested in. We present our findings in three general requirements.

For each requirement, we provide four motivating examples. The majority of the examples present properties from the road apple example and are linked to the nodes in Figure 3.4. The examples are numbered in the form x.y, where x specifies the requirement the example is aiming to clarify, and y is the number of the example. The first two examples from each requirement specify properties that are useful for penetration testers, while the second two examples specify properties that are of interest to security auditors.

Requirement 1: *The logic should be able to specify knowledge, location and possession.* We consider that an attack has occurred when an unauthorized person eventually (a) learns confidential information, (b) reaches a restricted location or (c) gains possession of an object.

Example 1.1 The server data reaches a remote server.
Example 1.2 The insider learns the employee's password.

Management may also use these properties to describe high-level policies.

Example 1.3 The server data should never leave the secure server.
Example 1.4 Only an employee can enter the secure room.

Requirement 2: *The logic should be able to distinguish among different evolutions leading to the same goal.* In a penetration test, where the quality of the security is measured by how close the tester gets to the target (the number of circumvented layers of protection), the tester is interested in specific class of attack scenarios.

Example 2.1 The insider enters the secure room and steals the data.
Example 2.2 The insider gives a dongle to the employee and steals the data.

P_1=$(logout(world).login(hall).eval(P')@secureServer)^{insider}$
P'=$(eval(login(remoteServer))@serverData)^{insider}$
P_2=$(eval(logout(hall).login(secureRoom))@insider)^{employee}$
P_3=nil
P_4=nil

Figure 5.1: A scenario where the employee lets the insider inside the secure room

The scenario defined by the process definitions in Figure 5.1 satisfies the property of example 2.1, and example 2.2 is satisfied by the scenario defined by the process definitions in Figure 3.15 that describe the road apple attack. Both scenarios eventually achieve the same result, namely the $serverData$ ends up in the $remoteServer$. A penetration tester might be more interested in scenarios satisfying the first example where the insider as part of the data theft manages to enter the secure room because in these scenarios she circumvents more protection layers. Therefore, the logic should be able to distinguish such different evolutions.

From a defensive point of view, a security auditor might be interested in specifying the proper execution order of procedures for accomplishing a task.

Example 2.3 A person can enter the secure room only through the hall.
Example 2.4 Whenever the employee receives money, the money is deposited in the secure room.

Requirement 3: *The logic should enable segregation of scenarios based on the social interaction between people, namely trust and delegation.* In Portunes trust is represented through security policies on people, while delegation is described through remote evaluation of processes on people. For example, P_2 in Figure 5.1, shows that the employee asks the insider to enter the secure room, or in other words *delegates* a task to the insider, which the insider gladly accepts. However, in the road apple attack (Figure 3.15 from Chapter 3), the insider gives the dongle to the employee, and the employee *trusts* the insider sufficiently to accept the dongle.

In some penetration tests the interaction between the tester and an employee is forbidden by the rules of engagement, or it is considered as a risky action because the outcome of the interaction is unpredictable. In other tests the main goal of the tester is to investigate the reaction of the employees in specific situations. For the first or for the second reason, penetration testers need to isolate attack scenarios that include social interaction.

Example 3.1 The insider steals the data by tricking the employee.

Examples

Requirement		1.1	1.2	1.3	1.4	2.1	2.2	2.3	2.4	3.1	3.2	3.3	3.4
1	a	y	n	y	y	y	y	y	y	y	y	n	n
	b	n	y	n	n	n	n	n	n	n	n	n	n
	c	n	n	n	n	n	n	n	y	n	n	n	n
2		n	n	y	y	y	y	y	y	y	y	y	y
3		n	n	n	n	n	y	n	n	y	y	y	y

Figure 5.2: The requirements and the examples that motivate the requirements.

Example 3.2 The insider steals the data without interacting with people.

From defensive point of view, the security auditor might want to check policies on the hierarchy of the organization:

Example 3.3 No person should delegate tasks to the boss.
Example 3.4 Only the boss should delegate tasks to other employees.

In Figure 5.2 we provide an overview of the requirements and show which property the logic should be capable to express to specify each motivating example. For example, expressing the property in Example 3.2 requires the logic to be able to express location, to show that the data is in a server controlled by the insider (requirement 1.a), to segregate among subsets of net evolutions, to select only evolutions where the insider tricks the employee (requirement 2) and to express interactions between people, to show the interaction between the insider and the employee (requirement 3).

5.3 Related work

The modal logic to reason about properties of a Portunes model and formally present goals of attack scenarios is inspired by Hennesy-Milner logic (HML) [56] and the modal logic for mobile agents [36].

In HML the temporal properties of the processes are expressed by the diamond operator $\langle a \rangle \phi$ indexed with a transition label. A process P satisfies $\langle a \rangle \phi$ if there exists a label a and a process P' such that $P \xrightarrow{a} P'$ and P' satisfies ϕ. The transition labels in HML are considered as basic entities and are syntactically characterized by the label of the modal operator.

The modal logic of De Nicola enriches HML logic with more refined action predicates and state formulae. The diamond operator, instead of being indexed with basic labels a, is indexed with abstract actions $\mathcal{A}$, which denote a set of basic labels, the localities involved in the transition and the information transmitted. Thus, the abstract operators denote properties of the transition labels. A net N satisfies a formula $\langle\mathcal{A}\rangle\phi$ if there exists a label a and a net N' such that a satisfies $\mathcal{A}$, $N \stackrel{a}{\rightarrowtail} N'$ and N' satisfies ϕ.

The logic for mobile agents allows the specification of mobile system properties specified in Klaim. Our approach uses similar notation and constructs, but adapts the semantics of the logic to the constructs of the Portunes language.

First, the Portunes language does not have variables nor logical localities, but does have node types. Thus, the predicates for variables are absent, we do not distinguish between physical and logical localities, and we introduce the type predicate u.

Second, the Klaim language has a different set of actions, thus the predicates for Klaim actions are replaced by predicates that reflect Portunes actions. Thirdly, the Portunes language does not have tuples, thus all tuple predicates from the transition labels and transition label predicates are absent. Because Portunes does not use variables nor logical spatialities all binding constructs and mapping are also absent from the logic.

Third, the modal logic for mobile agents contains state formulae to specify the distribution of the tuples in the system. Since the Portunes language does not have tuples, they are also missing from the logic presented in this chapter.

Finally, the diamond operator has a different meaning compared to the one used in HML and the logic for mobile agents. A net N satisfies a formula $\langle\mathcal{A}\rangle\phi$ if there exists a label a that satisfies $\mathcal{A}$, the net can eventually transition using the action a into a net N' ($N \stackrel{a}{\Rightarrow}^{+} N'$) and N' satisfies ϕ. Thus, a net will satisfy the formula not only if the action can occur in the next transition, but also if the action can occur in one of the further transitions.

5.4 Net and net evolution predicates

Motivated by the examples above, we now present the logic for expressing properties of a Portunes model. First we introduce the syntax of predicates for locations, actions and processes and provide their semantics. Using these predicates we can specify properties on process definitions for a given net. Next, we present the

predicates on the transition labels that describe the net evolutions. Finally, we present the semantics of the modal logic used for describing the properties of a given net.

5.4.1 Net predicates

Syntax of location predicates:

$$lp ::= 1_L \mid u \mid l$$

Location predicates can be a generic location (1_L), a type (u) or a location (l). $\mathcal{L}$ is the universe of node names, $\mathcal{L}_N \subseteq \mathcal{L}$ is a finite set of names for a given net N and Loc_N is a finite set of location predicate atoms for a given net N.

Examples of location predicates are: the predicate $insider$ is satisfied by all nodes named $insider$, the predicate $Space$ is satisfied by all nodes of type $Space$ and 1_L is satisfied by all nodes in the net.

Syntax of action predicates:

$$ap ::= lt(lp) \mid lt'(lp) \mid ln(lp) \mid e(pp)@lp$$

Every action from the Portunes language is represented by a predicate. The predicate $lt(lp)$ is satisfied by all $logout(l)$ where l satisfies the location predicate lp and $lt'(lp)$ is satisfied by all $logout'(l)$ where l satisfies the location predicate lp. Similarly, the action predicate $ln(lp)$ is satisfied by all $login(l)$ actions where l satisfies the location predicate lp and $e(pp)@lp$ is satisfied by all $eval(P)@l$, where the process P satisfies the predicate pp and the node l satisfies the predicate lp. $\mathcal{A}$ is a universe of actions, $\mathcal{A}_N \subseteq \mathcal{A}$ is a finite set of actions in a given net N and Act_N is a finite set of action predicates for a given net N.

A few examples of action predicates are: the predicate $lt(insider)$ is satisfied by all the actions $logout(insider)$, $ln(Space)$ is satisfied by all $login$ actions that perform login to a node of type $Space$, and $e(1_p)@employee$ is satisfied by all $eval$ actions that delegate a process to a node that satisfy the predicate $employee$.

Syntax of process predicates:

$$pp ::= 1_P \mid pp \wedge pp \mid ap^{lp} \rightarrow pp$$

The process predicate 1_P is satisfied by all processes and a conjunction of two process predicates $pp \wedge pp$ is satisfied by processes that satisfy both predicates. The predicate $ap^{lp} \rightarrow pp$ is satisfied by processes that contain an action satisfying the action predicate ap with an origin node satisfying the predicate lp followed by a process that satisfies the pp predicate. We define $\mathcal{P}$ as a universe of processes, $\mathcal{P}_N \subseteq \mathcal{P}$ as a finite set of processes in a given net N and $Proc_N$ as a finite set of process predicates for a given net N.

For example, the net:

$$N ::= insider ::^{\delta}_{\{money\}} P \,||\, employee ::^{\delta}_{\{secret\}} Q \,||\, hall ::^{\delta}_{\{insider,employee\}}$$

defines an environment where an employee and an insider are in the same hall. The intention of the insider to give money to the employee can be presented through the process predicate:

$$(e(ln(employee)^{insider} \rightarrow 1_P)@money^{insider}) \rightarrow 1_P.$$

The process predicate has the form $ap^{lp} \rightarrow pp$, where the action predicate ap is $e(ln(employee)^{insider} \rightarrow 1_P)@money$, the origin predicate of ap is $insider$ and pp is the predicate 1_P. The action predicate ap is of the form $e(pp)@lp$ where the process predicate is again of the from $ap^{lp} \rightarrow pp$, or $e(ln(employee)^{insider} \rightarrow 1_P$ and the locality predicate of the form $money$. This process predicate will be satisfied by all processes originating from $insider$ that contain an action $eval(P)@money$. Moreover the process P must contain an action $login(employee)$ originating from $insider$. Similarly the intention of the employee to give a secret to the insider is presented through the predicate:

$$(e(ln(insider)^{employee} \rightarrow 1_P)@secret^{employee}) \rightarrow 1_P.$$

Two sets of processes P and Q that satisfy these predicates:

$$P = eval(logout(insider)...login(employee))@money^{insider}$$
$$Q = eval(logout(employee)...login(insider))@secret^{employee}$$

5.4.2 Semantics of state predicates

The syntax of the predicates, helps in specification of nets based on the intentions of the nodes within. In other words, we can specify properties of processes within the nodes of the net.

The semantics of the predicates are presented in the form of the functions: $\mathbb{L}$:

$$\mathbb{L} : Loc_N \rightarrow 2^{\mathcal{L}_N} \qquad \mathbb{AC} : Act_N \rightarrow 2^{\mathcal{A}_N}$$

$$\mathbb{AC}[\![lt(lp)\,]\!] = \{logout(l) \mid l \in \mathbb{L}[\![lp]\!]\}$$

$$\mathbb{L}\,[\![1_L]\!] = \mathcal{L}_N \qquad \mathbb{AC}[\![lt'(lp)]\!] = \{logout'(l) \mid l \in \mathbb{L}[\![lp]\!]\}$$

$$\mathbb{L}\,[\![\,u\,]\!] = \{l \mid \mathcal{T}(l) = u\} \qquad \mathbb{AC}[\![ln(lp)]\!] = \{login(l) \mid l \in \mathbb{L}[\![lp]\!]\}$$

$$\mathbb{L}\,[\![\,l\,]\!] = \{l\} \qquad \mathbb{AC}[\![e(pp)@lp]\!] = \{eval(P)@l \mid l \in \mathbb{L}[\![lp]\!], P \in \mathbb{P}[\![pp]\!]\}$$

$$\mathbb{P} : Proc_N \rightarrow 2^{\mathcal{P}_N}$$

$$\mathbb{P}\,[\![1_P]\!] = \mathcal{P}_N$$

$$\mathbb{P}\,[\![pp_1 \wedge pp_2]\!] = \mathbb{P}[\![pp_1]\!] \cap \mathbb{P}[\![pp_2]\!]$$

$$\mathbb{P}\,[\![ap^{lp} \rightarrow pp]\!] = \{P \mid \exists a,l,Q : a \in \mathbb{AC}[\![ap]\!], l \in \mathbb{L}[\![lp]\!], origin(a) = l, P \xrightarrow{a}{}^{+} Q, Q \in \mathbb{P}[\![pp]\!]\}$$

Figure 5.3: Interpretation of location, action and process predicates

$Loc_N \rightarrow 2^{\mathcal{L}_N}, \mathbb{AC}$: $Act_N \rightarrow 2^{\mathcal{A}_N}, \mathbb{P} : Proc_N \rightarrow 2^{\mathcal{P}_N}$, which take a predicate lp, pp or ap and return a set of locations, processes and actions that satisfy the predicates respectively. The sets $\mathcal{L}_N$, $\mathcal{A}_N$ and $\mathcal{P}_N$ are derived from a named Portunes model presented by a specific net N. The semantics are defined in Figure 5.3.

The relation $P \xrightarrow{a}{}^{+} Q$ is satisfied when: $\exists P' : P \rightarrow^{*} P', P' \xrightarrow{a} Q$, where $\rightarrow^{*}$ is the reflexive, transitive closure of $\rightarrow$, defined in Section 3.5.4.

$\mathbb{L}\,[\![1_L]\!]$ returns the set of locations $\mathcal{L}_N$, $\mathbb{L}\,[\![u]\!]$ returns a set of locations that belong to a specific type and $\mathbb{L}\,[\![l]\!]$ returns a specific location $l \in \mathcal{L}_N$. $\mathbb{P}[\![1_P]\!]$ returns all processes in the net and $\mathbb{P}[\![pp_1 \wedge pp_2]\!]$ returns the processes that satisfy both predicates pp_1 and pp_2. $\mathbb{P}[\![ap^{lp} \rightarrow pp]\!]$ returns the processes that can execute an action satisfying the predicate ap, using an origin satisfying the predicate lp, and then evolve in a process that satisfies the predicate pp.

A process P from a net N satisfies the predicate pp, iff $P \in \mathbb{P}[\![pp]\!]$. Analogously, action a from a net N satisfies the predicate ap iff $a \in \mathbb{AC}[\![ap]\!]$ and a location l from a net N satisfies the predicate lp iff $l \in \mathbb{L}[\![lp]\!]$.

5.4.3 Transition label predicates

The process predicates present a set of actions that a single process might perform, and not actual net evolutions. In other words, a process predicate specifies an intention not an execution.

The transition labels, which present evolutions of a net are defined in Figure 5.4. We use Lab_N to denote a finite set of label predicates defined over a given net N

$$
\begin{array}{lll}
A ::= & \circ & lab ::= \ netcopy\ (l,l,l) \\
& \mid A_1 \cup A_2 & \quad \mid neteval\ (l,P,l) \\
& \mid A_1 \cap A_2 & \quad \mid netmove(l,l,l) \\
& \mid A_1 - A_2 & \\
& \mid src\ (lp) & \\
& \mid trg\ (lp) & \\
& \mid prt\ (lp) & \\
& \mid nc\ \ (lp_1, lp_2, lp_3) & \\
& \mid ne\ \ (lp_1, pp, lp_2\ ) & \\
& \mid nm(lp_1, lp_2, lp_3) &
\end{array}
$$

Figure 5.4: Syntax of transition labels and transition label predicates

and $\mathcal{LP}_N$ to denote a finite set of transition labels. The function that defines the meaning of the label predicates $\mathbb{A} : Lab_N \rightarrow 2^{\mathcal{LP}_N}$ is given in Figure 5.5.

We define the syntax and semantics of label predicates, where the locations and processes are replaced by location and process predicates. We use $\circ$ to denote all transition labels and $\cup$, $\cap$ and $-$ to denote union, intersection and exclusion of two sets of transition labels. The predicates src, prt and trg denote transition labels which have a specific source, parent or target node. The predicate $nc(lp_1, lp_2, lp_3)$ denotes transitions labeled $netcopy$ where the first parameter of the transition label satisfies the location predicate lp_1, the second parameter lp_2 and the third parameter lp_3. Similarly, ne and nc denote the $neteval$ and $netmove$ transition labels respectively.

Using the transition label predicates, we can specify sets of transitions labels based on a property they posses. For example, the predicate $prt(insider)$ is satisfied by all transition labels which add or remove an object or data from the node satisfying the location predicate $insider$, $trg(employee)$ is satisfied by all transition labels in which an object or data is given, or a task is delegated to a node satisfying the location predicate $employee$ and $nm(Person, 1_L, secureRoom) - nm(employee, 1_L, secureRoom)$ is satisfied by all transition labels in which node of type $Person$ other than the node satisfying the predicate $employee$ move to a node that satisfies the predicate $secureRoom$.

$$
\begin{aligned}
&\mathbb{A} : Lab_N \rightarrow 2^{\mathcal{LP}_N} \\
&\mathbb{A}\,[\![\ \circ\ ]\!] &&= \mathcal{LP}_{\mathcal{N}} \\
&\mathbb{A}\,[\![A_1 \cup A_2]\!] &&= \mathbb{A}[\![A_1]\!] \cup \mathbb{A}[\![A_2]\!] \\
&\mathbb{A}\,[\![A_1 \cap A_2]\!] &&= \mathbb{A}[\![A_1]\!] \cap \mathbb{A}[\![A_2]\!] \\
&\mathbb{A}\,[\![A_1 - A_2]\!] &&= \{a | a \in \mathbb{A}[\![A_1]\!], a \notin \mathbb{A}[\![A_2]\!]\} \\
&\mathbb{A}\,[\![\ src(lp)\ ]\!] &&= \{a | a \in \mathbb{A}[\![nm(l,1_L,1_L)]\!] \cup \mathbb{A}[\![nc(l,1_L,1_L)]\!] \cup \mathbb{A}[\![ne(l,1_P,1_L)]\!], l \in \mathbb{L}[\![lp]\!]\} \\
&\mathbb{A}\,[\![\ trg(lp)\ ]\!] &&= \{a | a \in \mathbb{A}[\![nm(1_L,1_L,l)]\!] \cup \mathbb{A}[\![nc(1_L,1_L,l)]\!] \cup \mathbb{A}[\![ne(1_L,1_P,l)]\!], l \in \mathbb{L}[\![lp]\!]\} \\
&\mathbb{A}\,[\![\ prt(lp)\ ]\!] &&= \{a | a \in \mathbb{A}[\![nm(1_L,l,1_L)]\!] \cup \mathbb{A}[\![nc(1_L,l,1_L)]\!], l \in \mathbb{L}\,[\![lp]\!]\} \\
&\mathbb{A}\,[\![nm(lp_1,lp_2,lp_3)]\!] &&= \{netmove(l_1,l_2,l_3) |\ l_1 \in \mathbb{L}\,[\![lp_1]\!], l_2 \in \mathbb{L}\,[\![lp_2]\!], l_3 \in \mathbb{L}\,[\![lp_3]\!]\} \\
&\mathbb{A}\,[\![ne(lp_1,pp,lp_2)\ ]\!] &&= \{neteval(l_1,P,l_3)\ |\ l_1 \in \mathbb{L}\,[\![lp_1]\!], l_2 \in \mathbb{L}\,[\![lp_2]\!], P \in \mathbb{P}\,[\![pp\ ]\!]\} \\
&\mathbb{A}\,[\![nc(lp_1,lp_2,lp_3)\ ]\!] &&= \{netcopy(l_1,l_2,l_3)\ |\ l_1 \in \mathbb{L}\,[\![lp_1]\!], l_2 \in \mathbb{L}\,[\![lp_2]\!], l_3 \in \mathbb{L}\,[\![lp_3]\!]\}
\end{aligned}
$$

Figure 5.5: Semantics of transition label predicates

5.5 Logic for Portunes models

Definition 15. *(Hennessy-Milner Logic) The set of HML formulas [56] for Portunes is given by the BNF grammar:*

$$\phi ::= tt \mid \neg\phi \mid \phi \wedge \phi \mid c(lp, lp) \mid \langle A \rangle \phi$$

The formula tt is always satisfied. The formula $\neg\phi$ is satisfied by a net that does not satisfy ϕ, while $\phi_1 \wedge \phi_2$ is satisfied by a net that satisfies both ϕ_1 and ϕ_2. The formula $c(lp_1, lp_2)$ is satisfied by a net in which a node l_2 with a name satisfying the predicate lp_2 belongs to the set s of a node l_1 satisfying the predicate lp_1, meaning node l_1 contains node l_2. Finally, $\langle A \rangle \phi$ is satisfied by a net N that has a transition label that satisfies A and after the transition satisfies the formula ϕ. Formulas like $[A]\phi$ and $\phi_1 \vee \phi_2$ can be derived from the logic: $[A]\phi = \neg\langle A \rangle \neg\phi$ and $\phi_1 \vee \phi_2 = \neg(\neg\phi_1 \wedge \neg\phi_2)$.

For example, $c(remoteServer, serverData)$ is satisfied by all nets where the server data is located in the remote server and $\langle nm(insider, hall, secureRoom) \rangle tt$ is satisfied by all nets where the insider moves from the hall to the secure room. We provide more examples of the formulas in the following section.

Let Net be a set of all nodes for a given network N, and Φ the set of all the formulas. The semantics is defined using the function $\mathbb{M} : \Phi \rightarrow 2^{Net}$ (Figure 5.6). A net N satisfies a formula ϕ if and only if $N \in \mathbb{M}\,[\![\phi]\!]$, and we write $N \models \phi$. We write $N \stackrel{a}{\Longrightarrow}^{+} N_1$ iff $\exists N' : N \Rightarrow^{*} N', N' \stackrel{a}{\Longrightarrow} N_1$. where $\Rightarrow^{*}$ is the reflexive, transitive closure of $\Rightarrow$ defined in Section 3.5.5.

The above formulas allow us to specify properties of the net for a single state, for all states in which the net evaluates and properties on the net evolutions.

$$\mathbb{M} : \Phi \rightarrow 2^{Net}$$
$$\mathbb{M}[\![tt]\!] = Net$$
$$\mathbb{M}[\![\neg\phi]\!] = Net - \mathbb{M}[\![\phi]\!]$$
$$\mathbb{M}[\![\phi_1 \wedge \phi_2]\!] = \mathbb{M}[\![\phi_1]\!] \cap \mathbb{M}[\![\phi_2]\!]$$
$$\mathbb{M}[\![\langle A\rangle\phi]\!] = \{N \mid \exists a, N_1 : N \stackrel{a}{\Longrightarrow}^{+} N_1, a \in \mathbb{A}[\![A]\!], N_1 \in \mathbb{M}[\![\phi]\!]\}$$
$$\mathbb{M}[\![c(lp_1, lp_2)]\!] = \{N \mid \exists l_1, l_2 : l_2 \in children_N(l_1), l_1 \in \mathbb{L}[\![lp_1]\!], l_2 \in \mathbb{L}[\![lp_2]\!]\}$$

Figure 5.6: Semantics of the logic

5.6 Using the logic to specify security policies

The logic looks straightforward but it can be difficult to develop an intuition for the meaning of the operators. In this section we show that sometimes it is difficult to find the correct formalization of a high-level policy into a formula. For example, consider the net N_0 from Figure 5.7 and the high-level policy "Sensitive information should not leave the hardened servers of the organization", which is a generalization of Example 1.3, where the $serverData$ is of type $SensitiveInfo$ (sensitive information) and $secureServer$ of type $HardServer$ (hardened server).

The net N_0 consists of a room containing two servers, one of which is hardened ($secserver$) and one is not ($normalserver$). The hardened server contains two nodes ($secdata1$ and $secdata2$) both of which are of type $SensitiveInfo$. From N_0 there are 3 transitions possible, leading to the nets N_1, N_2 and N_3. The nets N_0, N_1 and N_2 do not satisfy the policy, because from these nets some of the sensitive information can leave the hardened server. The net N_3 satisfies the policy because the hardened server has no sensitive information that can leave. The Portunes nets, together with the node types are presented in Figure 5.7.

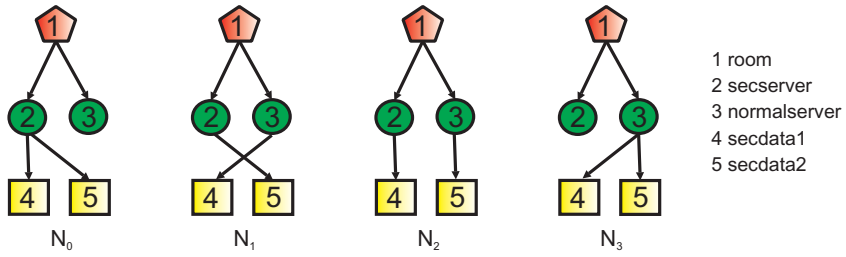
A first attempt to formalize the high-level policy into our logic would be:

$$\phi_1 = \neg c(1_L, SensitiveInfo)$$

The predicate $c(1_L, SensitiveInfo)$ in the formula is satisfied by every net where a node contains another node of type $SensitiveInfo$. The negation in ϕ_1 states that the nets that satisfy the predicate do not satisfy the formula and thus violate the policy. Every net that contains sensitive information will violate the policy, no matter where the sensitive information is located. In the example net N_0, the hardened server contains sensitive information, thus, the net would not satisfy the policy, hence $N_0 \not\models \phi_1$. This can be proved as fallows:

$$\mathbb{M}[\![\phi_1]\!] = Net - \mathbb{M}[\![c(1_L, SensitiveInfo)]\!] = \{N_0, N_1, N_2, N_3\} - \{N_0, N_1, N_2, N_3\} = \emptyset$$

However, this is not the intention of the high-level policy. The second attempt is



$$room ::^{(\bot,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{secserver,\, normalserver\}} nil$$

$$\| \quad secserver ::^{(\bot,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{secdata1,\, secdata2\}} nil$$

$$\| \quad normalserver ::^{(\bot,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{\}} nil$$

$$\| \quad secdata1 ::^{(\bot,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{\}} [logout(secserver).login(normalserver)]^{secdata1}$$

$$\| \quad secdata2 ::^{(\bot,\bot,\emptyset)\mapsto\{ln,lt\}}_{\{\}} [logout(secserver).login(normalserver)]^{secdata2}$$

$T = \{Space, HardServer, InsecureServer, SensitiveInfo\}$

$\succ\!\succ_{ln} = \{(Space, HardServer), (Space, InsecureServer),$
$(HardServer, SensitiveInfo), (InsecureServer, SensitiveInfo)\}$

$$\mathcal{T}(room) = Space,$$
$$\mathcal{T}(secserver) = HardServer,$$
$$\mathcal{T}(normalserver) = InsecureServer,$$
$$\mathcal{T}(secdata1) = \mathcal{T}(secdata2) = SensitiveInfo,$$

Figure 5.7: A net with two servers and two files.

to make an exception for the hardened servers, as follows:

$$\phi_2 = \neg(c(1_L, SensitiveInfo) \wedge \neg c(HardServer, SensitiveInfo))$$

This formula holds only if the current net has a node containing the sensitive information and if that node is not a hardened server. However, this formula checks for the satisfaction at the current state of the net, rather than all future net evolutions. Although the net from the example satisfies this formula ($N_0 \models \phi_1$), during the next net transitions, the sensitive data will leave the hardened server, which is against the high-level policy ($N_3 \not\models \phi_2$). We can prove this as follows:

$$\begin{aligned}\mathbb{M}\,[\![\phi_2]\!] &= Net - \mathbb{M}\,[\![c(1_L, SensitiveInfo) \wedge \neg c(HardServer, SensitiveInfo)]\!] \\ &= Net - (\mathbb{M}\,[\![c(1_L, SensitiveInfo)]\!] \cap \mathbb{M}\,[\![\neg c(HardServer, SensitiveInfo)]\!]) \\ &= Net - (\mathbb{M}\,[\![c(1_L, SensitiveInfo)]\!] \cap (Net - \mathbb{M}\,[\![c(HardServer, SensitiveInfo)]\!]))\end{aligned}$$

$= \{N_0, N_1, N_2, N_3\} - (\{N_0, N_1, N_2, N_3\} \cap (\{N_0, N_1, N_2, N_3\} - \{N_0, N_1, N_2\}))$
$= \{N_0, N_1, N_2\}$

The third attempt uses a temporal operator that will check whether the formula holds for all future net transitions would not solve the problem.

$$\phi_3 = \neg\langle\circ\rangle(c(1_L, SensitiveInfo) \wedge \neg c(HardServer, SensitiveInfo))$$

From the semantics in Figure 5.6, $c(HardServer, SensitiveInfo)$ will hold if there is at least one relation in the net that satisfies the predicate. Thus, the formula will be satisfied even by nets where all but one piece of sensitive information is in a hardened server, which is a violation of the high-level policy.

$$\begin{aligned}\mathbb{M}\,[\![\phi_3]\!] &= Net - \mathbb{M}\,[\![\langle\circ\rangle(c(1_L, SensitiveInfo) \wedge \neg c(HardServer, SensitiveInfo))]\!] \\ &= Net - \{N \mid \exists a, N_1 : N \stackrel{a}{\Longrightarrow}^{+} N_1, a \in \mathbb{A}\,[\![\circ]\!], \\ &\qquad N_1 \in \mathbb{M}\,[\![c(1_L, SensitiveInfo) \wedge \neg c(HardServer, SensitiveInfo)]\!]\} \\ &= \{N_0, N_1, N_2, N_3\} - \{N_0, N_1, N_2\} = \{N_3\}\end{aligned}$$

In the example, the formula seems to deliver the required result, $N_0 \not\models \phi_3$. However, under closer inspection, one can notice that for the current model the nets N_1 and N_2 do not satisfy the policy only because they eventually transition into N_3 where all data has left the hardened server. There can be models where only some of the sensitive data leaves the hardened servers (for example, the current net without one of the process definitions). If only one sensitive file leaves the hardened server, the resulting net will still satisfy the formula ϕ_3, but will be against the high-level policy. To represent the high-level policy into our logic we need to look at a behavior that causes the sensitive information to leave the hardened servers.

$$\phi_4 = \neg\langle nm(SensitiveInfo, HardServer, 1_L)\rangle tt$$

This formula would solve the issue both because it identifies the transitions where sensitive information is moved from a hardened server and because it is applied to all net transitions.

$$\begin{aligned}\mathbb{M}\,[\![\phi_4]\!] &= Net - \mathbb{M}\,[\![nm(SensitiveInfo, HardServer, 1_L)\rangle tt]\!] \\ &= \{N_0, N_1, N_2, N_3\} - \{N_0, N_1, N_2\} = \{N_3\}\end{aligned}$$

Although this policy will identify all behaviors where the information is moved from a hardened server, it still does not include the behaviors where the sensitive information is copied. Thus, the final formula that reflects the high-level policy is:

$$\begin{aligned}\phi_5 = \neg\langle nm(SensitiveInfo, HardServer, 1_L)\rangle tt \wedge \\ \neg\langle nc(SensitiveInfo, HardServer, 1_L)\rangle tt\end{aligned}$$

which is quite different from the initial formula $\neg c(1_L, SensitiveInfo)$.

5.6.1 Examples revisited

In section 3.5.2 we used the Portunes language to describe the road apple attack formally, an attack where the adversary uses physical, social and digital means to gain possession of sensitive data. In this section we use the road apple and the examples from the previous section to (1) describe adversarial goals and (2) formally define high level policies which should hold for all evolutions of the net.

Example 1.1 The server data reaches a remote server.

$$\langle\circ\rangle c(remoteServer, serverData)$$

Example 1.2 The insider learns the employee's password.

$$\langle\circ\rangle c(insider, employeePassword)$$

In Example 1.1 and 1.2 the goal is defined by a node being at a specific location. Using similar logic constructs, we can express goals including knowledge of information (person contains data) and possession (person contains object). The usage of the $\langle\circ\rangle$ means we that are not interested in the initial state of the net, but in an eventual state in the future.

Example 1.3 The server data should never leave the secure server.

$$\neg\langle nm(serverData, secureServer, 1_L)\rangle tt \wedge \\ \neg\langle nc(serverData, secureServer, 1_L)\rangle tt$$

Example 1.4 Only an employee can enter the secure room.

$$\neg\langle nm(Person, 1_L, secureRoom) - nm(Employee, 1_L, secureRoom)\rangle tt$$

Examples 1.3 and 1.4 describe high-level policies which should never be invalidated. Here we also see how location (similarly knowledge and possession) can be used to define a high-level policy.

In Example 1.3, the transition label predicate $nm(serverData, secureServer, 1_L)$ holds for all behaviors that contain a netmove action $netmove$ where the server data from the secure server moves to any other location in the net, regardless of the state the net will evolve into(tt). The negation of this predicate results in all allowed behaviors in the net that do not violate the policy. Because the data can be copied instead of moved, the formula contains a conjunct transition label predicate for behaviors that contain the netcopy action nc. In Example 1.4, the first

part of the transition label predicate $nm(Person, 1_L, secureRoom)$ holds for every behavior where node of type $Person$ moves into the secure room. The second part of the predicate, $nm(Employee, 1_L, secureRoom)$, holds for every behavior where a node of type $Employee$ moves into the secure room. The subtraction of these two predicates results into formula that holds true for all behaviors where a person moves in the secure office and is not an employee. The negation of this formula is satisfied by the all behaviors in the net except when a non-employee enters the secure room.

Example 2.1 The insider steals the data by entering the secure room.

$$\langle nm(insider, 1_L, secureRoom)\rangle(\langle\circ\rangle c(remoteServer, serverData))$$

Example 2.2 The insider steals the data by giving the employee a dongle.

$$\langle nm(dongle, 1_L, employee)\rangle(\langle\circ\rangle c(remoteServer, serverData))$$

In the above two examples, we define two strategies how the insider might get access to the data. Both strategies might be satisfied by a single net evolution. For example, the insider enters the office and then gives the dongle to the employee, or vice versa. Adding additional desired or non-desired conditions further segregates the possible evolutions of the net, allowing the penetration tester to focus only on those evolutions she is interested in.

In Example 2.1, the first transition label predicate $nm(insider, 1_L, secureRoom)$ holds only for behaviors that contain an action where an insider enters the secure room. The second part of the formula $\langle\circ\rangle c(remoteServer, serverData)$ holds for all behaviors in which eventually the remote server contains the server data. Thus, the whole formula holds for the behaviors where the insider first moves to the secure room, and eventually reaches a state where the remote server contains the server data. In Example 2.2 the structure of the formula is similar. The behaviors for which this formula holds have an action where the dongle is given to the employee, and then eventually the remote server contains the server data. Note that in both examples there might be no causality between the actions in the behavior. For example, there might be a behavior where the employee receives a dongle, but this action does not contribute to the server data ending up in the remote server.

Example 2.3 A person can enter the secure room only through the hall.

$$\neg\langle nm(Person, Space, secureRoom) - nm(Person, hall, secureRoom)\rangle tt$$

Example 2.4 Whenever the employee receives money, the money is deposited in the secure room.

$$\neg(\langle nm(money, 1_L, Employee)\rangle\neg(\langle nm(money, Employee, secureRoom)\rangle tt))$$

In Examples 2.3 and 2.4, the transition label predicates are satisfied only by a specific subset of the transition labels. Namely, all locations from where an employee can move inside the room, except the hall are forbidden. Or, as is the case of example 2.4, the property specifies only net evolutions where an employee receives money and then the money is eventually sent to the secure room.

In Example 2.3, the transition label predicate consists of two parts. The first part, $nm(Person, Space, secureRoom)$, is satisfied by all behaviors where a node of type $Person$ enters the secure room, while the second part of the predicate, $nm(Person, hall, secureRoom)$, is satisfied by all behaviors where a node of type $Person$ enters the secure room through the hall. The whole transition label predicate is satisfied by all behaviors where a node of type $Person$ enters the secure room except if he enters through the hall. The negation of the whole formula is satisfied by all behaviors, except the ones where a node of type $Person$ enters the secure room from any other place than the hall. In Example 2.4, the first transition label predicate $nm(money, 1_L, Employee)$ is satisfied by all behaviors where the employee receives money. The second transition label predicate $nm(money, Employee, secureRoom)$ is satisfied by all behaviors where the money is deposited in the secure room. When both predicates are connected with a negation between them, the resulting formula is satisfied by all behaviors where the employee receives money, but the money are not eventually deposited in the secure room. Finally, the negation in front of the first transition label predicate makes the whole formula hold for all behaviors except the ones where the employee receives money, but the money are not eventually deposited in the secure room.

Example 3.1 The insider steals the data by tricking the employee.

$$\langle ne(insider, 1_P, Employee)\rangle(\langle\circ\rangle c(remoteServer, serverData))$$

Example 3.2 The insider steals the data without interacting with people.

$$(\neg\langle ne(insider, 1_P, Person)\rangle)(\langle\circ\rangle c(remoteServer, serverData))$$

In some penetration tests, the rules of engagement forbid any interaction with the employees. In other tests, the main goal is to see the resilience of the employees against social engineering. Examples 3.1 and 3.2 show how we can segregate

attack scenarios that include contact with a specific person, or contain no contact with people.

The structure of the formulas in Example 3.1 and 3.2 are identical with the ones in Example 2.1 and 2.2. The first formula is satisfied by all behaviors where the insider delegates a process to an employee, and eventually, the remote server contains the server data. Similarly, the second formula is satisfied by all behaviors where the insider does not delegate any process to a person, but still eventually the remote server contains the server data.

Example 3.3 No person should delegate tasks to the boss.

$$\neg\langle ne(Person, 1_P, boss)\rangle tt$$

Example 3.4 Only the boss should delegate tasks to other employees.

$$\neg\langle ne(Employee, 1_P, Employee) - ne(boss, 1_P, Employee)\rangle tt$$

In Example 3.3 and 3.4 we show how the social aspects of the Portunes model can be used as high-level policies. Finding a delegation from an employee to a boss, or from an employee to another employee would mean that there is inconsistency in the policies imposed on the employees with the high-level policies.

The formula in Example 3.3 holds for all behaviors except the ones where a person delegates a task to the boss. In Example 3.4 the transition label predicate $ne(Employee, 1_P, Employee)$ is satisfied by all behaviors where there is an delegation of a task between two employees. The predicate $ne(boss, 1_P, Emploee)$ holds for all behaviors where the boss delegates a task to an employee. The formula that is a result of the subtraction between the two predicates holds for all behaviors where a task is delegated to a employee, and the person that delegates the task is not the boss. Finally, the whole formula state is satisfied by all behaviors in the net, except the behaviors in which an employee rather than the boss delegates a task.

The examples 1.1, 1.2, 2.1, 2.2, 3.1 and 3.2 present undesirable properties of behaviors, by defining a set of a) transitions $\langle A\rangle tt$, b) states $\langle\circ\rangle c(lp, lp)$ or c) transitions and states $\langle A\rangle(\langle\circ\rangle c(lp, lp)$. The logic can help penetration testers to select malicious goals and select from a set of possible attack the ones that yield the greatest chance of success. Similarly, by using a negation in the formulas, we can specify all behaviors in the net that do not violate a high-level security policy (examples: 1.3, 1.4, 2.3, 2.4, 3.3, 3.4). Such behaviors can help auditors to define multiple policies, and get a set of all allowed behaviors in the organization.

5.6.2 Other uses of the logic

In this chapter we showed how the logic can be used to describe behaviors using temporal operators, specify complex goals using spatial operators and a combination of both behaviors and goals. The logic formulas can be used to describe high-level policies which should hold for every state of the model.

Another usage of the formulas is to aid the analysis from Chapter 4. The *generatePartialAttack* algorithm may use the formulas as a heuristic in searching only for the behaviors that have the property specified by the formulas, effectively cutting the search space.

Finally, the formulas from the logic can be used as a fine grained search for a specific behavior from a set of behaviors. For example, if a set of behaviors satisfy a goal, the user can search for a subset of behaviors within the found behaviors, by providing additional formulas stating additional properties of interest.

We partially implemented the logic in the Portunes tool, by allowing multiple goals composed by combining the $\vee$ an $\wedge$ state operators in the specification of a goal. We consider the full implementation of the logic into the Portunes tool as future work.

5.7 Conclusion

In this chapter we presented a variant of Hennesy-Milner logic and the modal logic for mobile agents and used the logic to express state and transition properties of the Portunes language. The logic lacks variable, tuple and logical locality predicates, predicates for Klaim actions are replaced by predicates that reflect Portunes actions and the diamond operator has a different meaning compared to the other variations of the HML logic.

The logic is designed to specify a set of desired and undesired behaviors and states of Portunes models. These behaviors and states are represented as properties of Portunes models that (1) describe adversarial goals and (2) formally define high-level policies which should hold for all evolutions of the net. The logic lacks recursion and thus cannot express arbitrary repetition of actions because we are interested in a single achievement of an adversarial goal or invalidation of a policy. For example, if the analysis finds an attack scenario, we assume the scenario can be repeated multiple times.

In the first part of the thesis we showed how to describe and analyze behaviors

that span the three security domains. We presented low-level policies as predicates, high-level policies as modal logic formulae and describe behaviors that can occur as process definitions. We also provide an analysis that assesses the completeness of the policy refinement by taking both defensive (auditing) and offensive (penetration testing) standpoint.

The main contribution of the first part of the thesis is the mapping of security aspects of the physical and social domain together with the digital domain into a single framework named Portunes. The framework consists of a graph and a language inspired by the Klaim family of languages. To capture the three domains efficiently, Portunes is able to represent 1) physical properties of elements, 2) mobility of objects and data, 3) identity, credential and location based access control and 4) trust and delegation between people.

We chose the abstraction level of the three domains to be sufficiently high to be easy to use, but still sufficiently detailed to provide useful results. One can envision extending the framework with constructs such as negotiation between people, behavioral patterns or detection mechanisms, to increase the detail of the produced behaviors.

To bring the framework closer to practitioners, we provided a logic to help users specify high-level policies and adversarial goals and a graphical implementation of the language in a tool. This approach allows generating and analyzing attack scenarios which span all tree domains.

The applicability of the Portunes framework was demonstrated using the example of the road apple attack, showing how an insider can attack without violating existing security policies by combining actions from all three domains. We also showed how Portunes can be used to generate attack scenarios automatically for penetration testing teams that use physical access and social engineering to gain possession of a digital asset. So far, we found out that the Portunes tool can produce sufficiently detailed realistic attack scenarios for testers to execute. We describe our findings on this matter in greater detail in Chapter 8.

The behaviors generated by the analysis can be used in several areas. In quantitative risk assessment, risk estimates are based on a number of attacks on the asset obtained through brainstorming. The Portunes tool can automatically generate a complete list of attacks, greatly improving the security risk estimate for an asset. In designing new socio-technical systems it is crucial (and sometimes mandatory) to show that the system is behaving in accordance with a set of high-level policies. Portunes can aid in this analysis by generating possible scenarios that can violate these policies.

In outsourcing scenarios, the vendor to whom a task is outsourced needs to demon-

strate that the data of the user is stored in a secure location and show that all possible scenarios are considered to protect the confidentiality of the data from adversaries and internal staff members. Portunes can aid in formally providing this proof. In security awareness training programs, it is advisable to use facility specific simulations to teach the employees on possible threats to the organization. Portunes can generate these simulations automatically and greatly speed up the process of preparing the program.

Part II

Policy enforcement

The second part of this thesis focuses on testing policy enforcement. In the first part of the thesis, we used the road apple attack, where an insider gives a dongle to an employee, as a running example. This example is interesting because it uses policies from all three domains and is challenging to model. When testing the enforcement of the policies, it is harder for the tester to obtain an asset from an employee rather than give one. Therefore, as a running example of the second part we explore the problem of protecting laptops from theft. This example is more suitable than the road apple attack because it is harder for a tester to take a laptop from an employee, than to give a dongle to an employee.

Chapter 6 proposes two methodologies for performing physical penetration tests using social engineering where the tester needs to obtain an asset. In this chapter we provide a set of requirements a penetration testing methodology should satisfy, and evaluate the two proposed methodologies based on these requirements. The methodologies focus on obtaining a marked asset from the premises of a targeted organization.

Chapter 7 assesses the effectiveness of security mechanisms in the physical and social domain. Using the methodologies provided in Chapter 6, we orchestrated more than 30 penetration tests. We also analyzed the logs from stolen laptops in 2 universities over a period of two years. The results from the penetration tests and the analysis of the logs provided us with an insight of the effectiveness of CCTV, access control and security awareness of the employees as mechanisms for protecting assets in an organization.

Chapter 8 proposes a practical assignment for teaching students penetration testing skills. During a period of three years we provided a practical assignments

for first year master students in computes security. As part of this assignment, students were provided with the opportunity to perform physical penetration tests using social engineering, offline attacks on laptops and online attacks on vulnerable servers. We analyze the implications of the assignment to the students and to the employees and argue that such assignments are useful for both the students and organization if executed with diligence.

Chapter 6

Methodologies for Physical Penetration Testing using Social Engineering *

Penetration tests on IT systems are sometimes coupled with physical penetration tests and social engineering. In physical penetration tests where social engineering is allowed, the penetration tester directly interacts with the employees. These interactions are usually based on deception and if not done properly can upset the employees, violate their privacy or damage their trust toward the organization and might lead to law suits and loss of productivity.

In this chapter, we propose two methodologies for performing a physical penetration test where the goal is to gain an asset using social engineering. These methodologies aim to reduce the impact of the penetration test on the employees. We used these methodologies to orchestrate 32 penetration tests over a period of three years.

*This chapter is a minor revision of the paper "Two Methodologies for Physical Penetration Testing using Social Engineering" [5] published in the Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC'10), pages 399-408, ACM, 2010

6.1 Introduction

A penetration test can assess both the IT security and the security of the facility where the IT systems are located. If the penetration tester assesses the IT security, the goal is to obtain or modify marked data located deep in the organizations network. Similarly, in testing the physical security of the location where the IT system is located, the goal of the penetration test is to obtain a specific asset, such as a laptop or a document. Physical and digital penetration tests can be complemented with social engineering techniques, where the tester is allowed to use knowledge and help from the employees to mount the attack.

In digital penetration tests the resilience of an employee is measured indirectly, by making phone queries or sending fake mail that lure the employee to disclose secret information. These tests can be designed in an ethical manner [45] and within the legal boundaries [94]. However, measuring the resilience of an employee against social engineering in a physical penetration test is *direct* and *personal*. When the tester enters the facility of the organization and directly interacts with the employees, she either deceives the employee, trying to obtain more information about the goal, or urges the employee to help her, by letting the tester inside a secure area or giving the tester a credential. The absence of any digital medium in the communication with the employees makes the interaction between the penetration tester and the employee intense, especially if the employee is asked to break company policies.

There are three main consequences from personal interaction between the tester and the employee. First, the employee might be stressed by having to choose between helping a colleague and breaking the company policies. Second, the tester might not treat the employee respectfully. Finally, when helping the penetration tester to enter a secure location, the employee loses the trust from the people who reside in the secure location. For example, employees might stop trusting the secretary when they find out she let an intruder into their office. To avoid ethical and legal implications, organizations may avoid physical penetration testing with social engineering, leaving themselves unaware of attacks where the attacker uses non-digital means to attack the system.

This chapter tackles the problem how to perform a physical penetration test using social engineering in the most respectful manner, while still getting results that lead to improving the security of the organization. The contribution of this chapter is two methodologies for physical penetration tests using social engineering where the goal is to gain possession of a physical asset from the premises of the organization. Both methodologies are designed to reduce the impact of the test on the employees. We used these methodologies to perform 32 penetration tests over

a period of three years, where students tried to gain possession of marked laptops placed in buildings of two universities in The Netherlands. Detailed information on the execution of these tests is presented in Chapter 7.

The rest of the chapter is structured as follows. In section 2 we present related work and in section 3 we set the requirements for the methodologies. Sections 4 and 5 outline the methodologies, section 6 provides an evaluation of the structure of the methodologies and section 7 concludes the chapter.

6.2 Related work

In the computer science literature, there are isolated reports of physical penetration tests using social engineering [51, 11]. However, these approaches focus completely on the actions of the penetration tester and do not consider the impact of the test on the employees.

There are a few methodologies for penetration testing. The Open-Source Security Testing Methodology Manual (OSSTMM) [113] provides an extensive list of *what* needs to be checked during a physical penetration test. However, the methodology does not state *how* the testing should be carried out. OSSTMM also does not consider direct interaction between the penetration tester and the employees. Barret [16] provides an audit-based methodology for social engineering using direct interaction between the penetration tester and an employee. Since this is an audit-based methodology, the goal is to test *all* employees. Our methodologies are goal-based and focus on the security of a specific physical asset. Employees are considered as an additional mechanism which can be circumvented to achieve a goal, instead of being the goal. Türpe and Eichler [97] focus on safety precautions while testing production systems. Since a test can harm the production system, it can cause unforseeable damages to the organization. In our work the penetration test of the premises of an organization can be seen as a test of a production system.

In the crime science community, Cornish [34] provides mechanisms how to structure the prosecution of a crime into universal crime scripts and reasons about mechanisms how to prevent the crime. We adopt a similar reporting format to present the results from a penetration test.

The Bellman report [47] defines the ethical guidelines for the protection of humans in testing. The first guideline in the report states that all participants should be treated with respect during the test. Finn [46] provides four justifications that

need to be satisfied to use deception in research. We use the same justifications to show that our methodology is ethically sound.

6.3 Requirements

A penetration test should satisfy five requirements to be useful for the organization. First, the penetration test needs to be realistic, since it simulates an attack performed by a real adversary. Second, during the test all employees need to be treated with respect [47]. The employees should not be stressed, feel uncomfortable nor be at risk during the penetration test, because they might get disappointed with the organization, become disgruntled or even start legal action. Finally, the penetration test should be repeatable, reliable and reportable [16]. We call these the R* requirements:

Realistic - employees should act normally, as they would in everyday life.

Respectful - the test is done ethically, by respecting the employees and the mutual trust between employees.

Reliable - the penetration test does not cause productivity loss of employees.

Repeatable - the same test can be performed several times and if the environment does not change, the results should be the same.

Reportable - all actions during the test should be logged and the outcome of the test should be in a form that permits a meaningful and actionable documentation of findings and recommendations.

These are conflicting requirements. For example:

1. In a realistic penetration test, it might be necessary to deceive an employee, which is not respectful.
2. In a realistic test, arbitrary employees might be social engineered to achieve the goal, which is unreliable.
3. In a reportable test, all actions of the penetration tester need to be logged, which is unrealistic.

Orchestrating a penetration test is striking the best balance between the conflicting requirements. If the balance is not achieved, the test might either not fully assess the security of the organization or might harm the employees.

We propose two methodologies, Environment-Focused Methodology and Custodian-Focused Methodology for conducting a penetration test using social engineering. Both methodologies strike a different balance between the R* requirements, and their usage is for different scenarios. Both methodologies assess the security of an organization by testing how difficult it is to gain possession of a pre-defined asset.

The methodologies can be used to assess the security of the organization, by revealing two types of security weaknesses: errors in enforcement of social, digital and physical policies and an absence of a policy. In the first case, the tests should focus on how well the employees follow the security policies of the organization and how effective the existing physical and digital security mechanisms are. In the second case, the primary goal of the tests is to find and exploit gaps in the existing policies rather than in their implementation. For example, a test can focus on how well the credential sharing policy is enforced by employees or can focus on exploiting the absence of a credential sharing policy to obtain the target asset.

In this chapter we present the two methodologies which reduce the impact of these tests. The Environment-Focused (EF) Methodology, measures the security of the environment where the asset is located. The methodology is suitable for tests where the custodian (person who controls the asset) is not subject of social engineering and is aware of the execution of the test. One example of such test is evaluating the security of the assets residing in the office of the CEO, but not the awareness of the CEO herself. The Custodian-Focused (CF) Methodology is more general, and includes the asset owner in the scope of the test. In this methodology, the owner is not aware of the test. The CF methodology is more realistic, but it is less reliable and respectful to the employees.

6.4 Environment-Focused Methodology

First, we define the actors in the Environment-Focused methodology. Then, we introduce all events that take place during the setup, execution and aftermath of the penetration test. Finally, we validate the methodology by conducting three penetration tests and present some insights from the experience.

6.4.1 Actors

The penetration test involves four different actors.

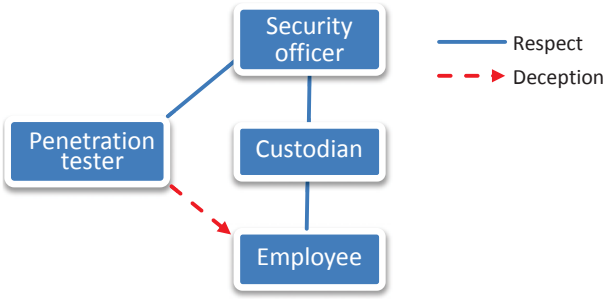


Figure 6.1: Actors in the EF methodology

Security officer - an employee responsible for the security of the organization. The security officer orchestrates the penetration test.

Custodian - an employee in possession of the assets, sets up and monitors the penetration test.

Penetration tester - an employee or a contractor trying to gain possession of the asset without being caught.

Employee - person in the organization who has none of the roles above.

The actors and the relations between them are shown in Figure 6.1. The majority of actors treat each other with respect. No respect relation between two actors means either the actors do not interact during the penetration test (for example between the tester and the custodian) or do not have a working relationship (between the penetration tester and the employee). In this methodology, the tester deceives the employee during the penetration test, presented in the figure with a dashed line.

6.4.2 Setup

Figure 6.2 provides the sequence of events that take place during the setup, execution and closure of the penetration test. During all three stages of the penetration test, employees should behave normally (1 in Figure 6.2).

As in other penetration testing methodologies, before the start of the test, the security officer sets the scope, the rules of engagement and the goal (2 in Figure 6.2). The *goal* is gaining physical possession of a marked asset. The *scope* of the testing provides the penetration tester with a set of locations she is allowed to enter, as well as business processes in the organization she can abuse, such as processes for issuing a new password, or processes for adding/removing an

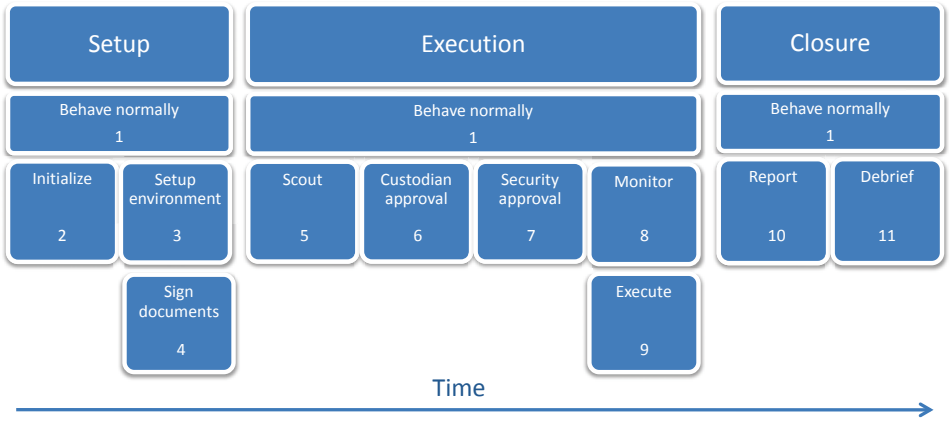


Figure 6.2: The sequence of events in the Environment-Focused Methodology. Each box represents an event which happens in sequence or parallel with other events. For example, event 3 happens after event 2 and in parallel with events 1 and 4.

employee. The *rules of engagement* restrict the penetration tester to the tools and means she is allowed to use to reach the target. These rules, for example, define if the tester is allowed to force doors, to break windows or to use social engineering.

The custodian first signs an informed consent form and then sets up the environment, by marking an asset in her possession and installing monitoring equipment. The asset should not be critical for the daily tasks of the custodian or anyone else, including the organization. Thus, when the penetration tester gains possession of the asset, the productivity of the custodian using the asset and the process flow of the company will not be affected. The custodian leaves the asset in her office or an area without people (storage area, closet). If the custodian shares an office with other employees, the monitoring equipment should be positioned in such a way that it records only the asset and not the nearby employees. The custodian knows when the test takes place, and has sufficient time to remove/obscure all sensitive and private assets in her room and around the marked asset (3 in Figure 6.2).

Meanwhile, the penetration tester needs to sign the rules of engagement (4 in Figure 6.2). The OSSTMM methodology [113] provides a comprehensive list of rules of engagement.

6.4.3 Execution

The security officer should choose a trustworthy penetration tester and monitor her actions during the execution stage.

When the penetration test starts, the tester first scouts the area and proposes a set of attack scenarios (5 in Figure 6.2). The proposed attack scenarios need to be approved first by the custodian (6 in Figure 6.2) and then by the security officer (7 in Figure 6.2). The custodian is directly involved in the test and can correctly judge the effect of the scenario on her daily tasks and the tasks of her colleagues. The security officer needs to approve the scenarios because she is aware of the general security of the organization and can better predict the far-reaching consequences of the actions of the tester.

If the custodian or the security officer disapprove an attack scenario, they need to evaluate the scenario and estimate the success. The tester puts in the report that the scenario was proposed, the reasons why the scenario was turned down and the opinion of all three roles on the success of the scenario. In this way, although the scenario is not executed, it is documented including the judgment on the effectiveness of the attack by the security officer, the custodian and the tester.

After approval from the custodian and the security officer, the tester starts with the execution of the attack scenarios (8 in Figure 6.2). The custodian and the security officer remotely monitor the execution (9 in Figure 6.2) through CCTV and the monitoring equipment installed by the custodian.

The penetration tester needs to install wearable monitoring equipment to log her actions. The logs serve three purposes. First, they ensure that if an employee is treated with disrespect there is objective evidence. Second, the logs prove that the penetration tester has followed the attack scenarios, and finally, the logs provide information how the mechanisms were circumvented, helping the organization repeat the scenario if needed.

6.4.4 Closure

After the end of the test, the penetration tester prepares a report containing a list of attack traces. Each attack trace contains information of successful or unsuccessful attacks (10 in Figure 6.2). Based on the report, the security officer debriefs both the custodians and any deceived employees during the test (11 in Figure 6.2).

Reporting. The attack traces are structured in a report that emphasizes the weak and the strong security mechanisms encountered during the penetration test, struc-

Generic Script	**Attack trace**	**Circumvented mechanisms**	**Recommendations**
Prepare for the attack	Buy a bolt cutter and hide it in a bag. Scout the building and the office during working hours. Obtain an after working hours access card.	Access control of the building entrances during working hours. Credential sharing policy.	Keep entrance doors to the building locked at all time. Provide an awareness training concerning credential sharing.
Enter the building	Enter the building at 7:30 AM, before working hours. Hide the face from CCTV at the entrance using a hat.	CCTV pre-theft surveillance.	Increase the awareness of the security guards during non-working hours.
Enter the office	Wait for the cleaning lady. Pretend you are an employee who forgot the office key and ask the cleaning lady to open the office for you.	Challenge unknown people to provide ID. Credential sharing policy.	Reward employees for discovering intruders.
Identify and get the asset	Search for the specific laptop. Get the bolt cutter from the bag and cut the Kensington lock. Put the laptop and the bolt cutter in the bag.	Kensington lock.	Get stronger Kensington locks. Use alternative mechanism for protecting the laptop.
Leave the building with the laptop	Leave the building at 8:00, when external doors automatically unlock for employees.	CCTV surveillance. Access control of the building entrances during working hours.	The motion detection of the CCTV cameras needs to be more sensitive .

Figure 6.3: Reporting a successful attempt. The figure shows an example of a generic script instantiated with an attack trace. First we define the generic script, which encompasses the stages of all attacks. In the example, they are: enter the building, enter the office, identify and get the asset, and exit the building. For each step in a trace, we identify both the mechanisms (if any) that were circumvented and mechanisms that stopped an attack. For failed attacks, the table shows which mechanisms were circumvented up to the failed action, and the mechanism that successfully stopped the attempt.

tured following 25 techniques for situational crime prevention [35]. For different domains there are extensive lists of security mechanisms to enforce the 25 techniques (for example, [61]). The combination of the attack traces together with the situational crime prevention techniques gives an overview of the circumvented mechanisms [105] (Figure 6.3)

Debriefing the employees and the custodian. After finding they were deceived by the same organization they work for, the employees might get disappointed or disgruntled. At the end of the test the security officer fully debriefs the custodian and the employees. The debriefing should be done carefully, to maintain or restore the trust between custodian and the employees who helped the tester to gain the asset.

Figure 6.4: Recording from the orchestrated tests using the EF methodology. The student provided to the janitor a fake email stating he needs to collect a laptop from the custodian office. The janitor let the student into the office and helped him find the key from the Kensington lock.

6.4.5 Case study

To test the usability of the physical penetration tests using social engineering on the employees, we executed a series of penetration tests following the EF methodology. These pilots allowed us to gain a clear, first-hand picture of each execution stage of the methodology, and draw observations from the experience. The tests are presented in detail in Chapter 7.

6.4.6 Lessons learned from the penetration tests

The observations are result of our experience with the penetration tests using qualitative social research and might not generalize to other social environments. However, the observations provide an insight of the issues that arose while using the methodology in practice.

The attack scenarios should be flexible. Although the testers provided scenarios prior to all attacks, in all cases they were forced to deviate from them, because the target employee was either not present or was not behaving as expected. Attack scenarios assure the custodian and the security officer that the actions of the

penetration tester are in the scope of the test, but at the same time there should be some freedom in adapting the script to the circumstances.

The methodology does not respect the trust relationship between the custodian and the employees. After the penetration test, the custodian knows which employees were deceived, and the trust relationship between them is disturbed. For example, if the secretary lets the penetration tester into the office of the custodian, the custodian might not be able to trust her again. Therefore, after some penetration tests the security officer might decide not do debrief the custodian completely.

During the penetration test, separating the custodian from the employees is hard. Whenever the testers approached a colleague from the office, the first reaction of the colleague was to call the custodian and ask for guidance. This led to uncomfortable situations where the custodians were forced to shut down their telephones and ignore e-mails while outside the office. These situations can be reduced if the testers inform the custodians the exact time they will carry out an attempt. Thus, the custodians can be unavailable only for short periods.

Debriefing proved to be difficult. After the test, we fully disclosed the test to all involved employees. During the debriefing we focused on the benefits of the penetration test to the university and their help setting up the test. In three of the executed penetration tests a security guard opened the office door for the penetration testers. After the debriefing, we concluded that we caused more stress to the guard during the debriefing than the testers had caused during the penetration test. We addressed this issue in the second methodology, by selectively informing the involved employees.

6.5 Custodian-Focused Methodology

In the EF methodology, the custodian is aware of the penetration test. The knowledge of the penetration test may change her normal behavior and thus influences the results of the test. Since the asset belongs to the custodian, and the asset is in the office of the custodian, in many environments it is desirable to include the custodian's resistance to social engineering as part of the test.

After performing the first series of penetration tests, we revisited and expanded the Environment-Focused Methodology. The CF methodology can be seen as a refinement of the EF methodology, based on the experience from the first set of penetration tests. In the CF methodology the custodian is not aware of the test, making the methodology suitable for penetration tests where the goal is to check

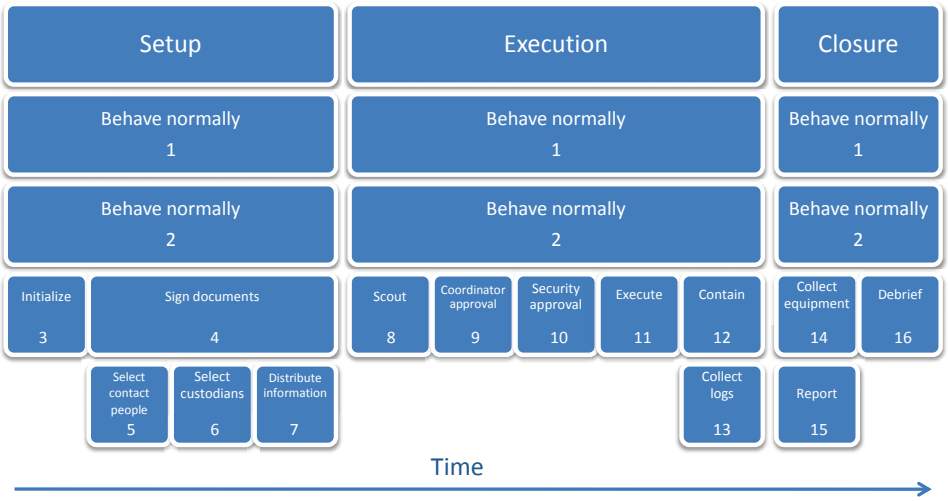


Figure 6.5: The sequence of events in the Custodian-Focused methodology

the overall security of an area *including* the level of security awareness of the custodian.

6.5.1 Actors

There are six actors in the CF methodology.

Security officer - an employee responsible for the security of the organization.

Coordinator - an employee or contractor responsible for the experiment and the behavior of the penetration tester. The coordinator orchestrates the whole penetration test.

Penetration tester - an employee or contractor who attempts to gain possession of the asset without being caught.

Contact person - an employee who provides logistic support in the organization and a person to be contacted in case of an emergency.

Custodian - an employee at whose office the asset resides. The custodian should not be aware of the penetration test (1 in Figure 6.5).

Employee - person in the organization who has none of the roles above. The employee should not be aware of the penetration test (2 in Figure 6.5).

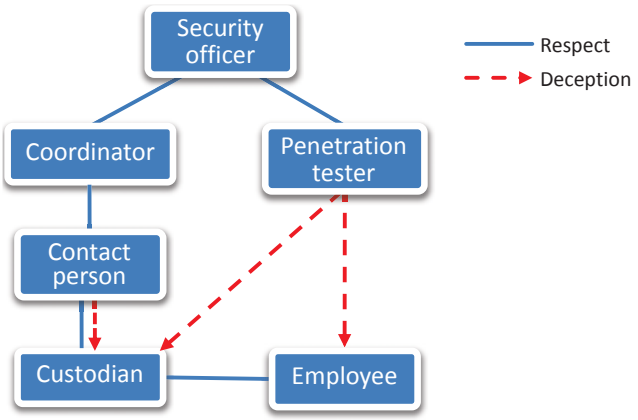


Figure 6.6: Actors in the CF methodology

Figure 6.6 shows the actors and the relations between them. In this methodology, the penetration tester deceives the employees as well as the custodian. Moreover, the contact person also needs to deceive the custodian. These relations are discussed in greater depth in section 6.6.

6.5.2 Setup

At the beginning, similar to the EF methodology, the security officer initializes the test by defining the target, scope and the rules of engagement. The security officer at this point assigns a coordinator for the penetration test and provides the coordinator with marked assets and equipment for monitoring the assets (3 in Figure 6.5). The marked assets should be similar to the asset of interest for which the security is measured. The monitoring equipment should be non-intrusive and its purpose is to have additional information on the activities of the penetration tester.

The penetration tester should sign the rules of engagement (Appendix A) before the start of the execution stage (4 in Figure 6.5). The coordinator selects a number of contact people and provides them with the marked assets and the monitoring equipment (5 in Figure 6.5). Furthermore, the coordinator provides a cover story which explains why the custodian is given the asset. The contact person selects a number of custodians based on the requirements from the security officer (random, specific roles, specific characteristics) and distributes the marked assets and the monitoring equipment to the custodians. After giving the monitoring equipment, the contact person should get a signed informed consent (Appendix B) from the

custodians (6 in Figure 6.5). If the asset can store data, the document must clearly state that the custodian should not store any sensitive nor private data in the asset. Before the penetration test starts, the coordinator distributes a list of penetration testers to the security officer, and a list of asset locations to the penetration tester (7 in Figure 6.5).

6.5.3 Execution

The first steps of the execution stage are similar to the previous methodology. The penetration tester scouts the area and proposes attack scenarios (8 in Figure 6.5). The coordinator and later the security officer should agree with these scenarios before the tester starts executing them (9 and 10 in Figure 6.5). After approval from both actors, the tester starts executing the attack scenarios. If a penetration tester is caught or a termination condition is reached, the penetration tester immediately informs the contact person. Thus, if the custodian stored sensitive data in the asset, the data is not exposed.

When the tester gains possession of the target asset, she informs the contact person and the coordinator and returns the asset to the contact person (11 in Figure 6.5). The contact person collects the monitoring equipment and informs the security officer (12 in Figure 6.5). If the tester gains possession of the asset without the knowledge of the custodian, the contact person needs to reach the custodian before the custodian reaches the office and explain to the custodian that the test is terminated. The tester should also leave a note stating that the asset has been taken as part of a test together with contact details from the coordinator and the security officer (for example, Appendix F). The security officer obtains surveillance videos from the CCTV and access logs and gives them to the coordinator (13 in Figure 6.5).

6.5.4 Closure

After the execution stage, the penetration tester writes a report of all attempts, both failed and successful, in the form of attack traces and gives them to the coordinator (14 in Figure 6.5). The coordinator has two tasks. First, she collects the marked assets and monitoring equipment from the contact person (15 in Figure 6.5) and returns them to the security officer. Second, the coordinator debriefs the security officer and the custodians and provides the custodian a form of reward for helping in the assessment (16 in Figure 6.5).

Not all employees that were social engineered should be debriefed. Employees who were treated with respect and to whom the penetration tester did not cause discomfort during the interaction should not be debriefed, because the debriefing can cause more stress than the interaction with the penetration tester. The decision which employees need to be debriefed lies with the security officer, and is based on the logs from the penetration tester and the monitoring equipment. The criteria on which employees need to be debriefed are presented in greater detail in Section 6.6.

All custodians should be debriefed, because they sign an informed consent at the beginning of the test. However, to preserve the trust between the custodian and the employees, the custodian should not know which employee contributed to the attack.

Three elements should be considered before the debriefing. First, the custodians were deceived by the organization they work for (more specifically, by the contact person). Second, in case of direct interaction, their privacy might be violated by the logging equipment from the tester. Third, they might be stressed from the penetration test either directly, through interaction with the penetration tester, or indirectly, by finding their asset is gone before the contact person reaches them.

The debriefing should focus on the contribution of the custodian in finding the security vulnerabilities in the organization, and the custodian should be rewarded for the participation.

6.5.5 Case study

Similarly as with the EF Methodology, we orchestrated a number of penetration tests to obtain first hand insight of the possible consequences of using this methodology as well as mechanisms to improve the methodology. We orchestrated 29 penetration tests with the Custodian-Focused Methodology in a period of 2 years. Part of the tests were performed in University of Twente and part in the Technical University of Eindhoven. The penetration tests are described in greater detail in Chapter 7.

6.5.6 Lessons learned from the penetration tests

It should be specified in advance which information the penetration tester is allowed to use. For example, the penetration tester should not use knowledge about the cover story used by the contact person. During the case study, six penetration

Figure 6.7: Recording from a penetration test orchestrated using the CF methodology. The student went to the office early in the morning, disguised as an employee who forgot his key. The cleaning lady let the student in. The student used a bolt cutter to remove the Kensington lock.

testers used knowledge of the cover story to convince the custodian to hand in the laptop. Thus, these tests were less realistic.

Panic situations need to be taken into consideration in the termination conditions. Several times the custodian or an employee got suspicious and raised an alarm. Since only the security officer knew about the experiment, and the other security personnel was excluded, news of people stealing laptops spread in a matter of hours. In these situations the coordinator should react quickly and explain to the employees that the suspicious activity is a test.

The penetration test cannot be repeated many times with the same persons. If a custodian participated in the penetration test once, she knows what will happen. The same holds for the employees she told about the experiments and the employees that were socially engineered.

Asking the custodians to install the monitoring equipment proved to be hard The custodians either did not have the technical skills to install the software on their PC, had no administrator rights on the PC, or had a laptop instead of PC, which they took home after working hours.

6.6 Evaluation

In this section we compare both methodologies against the R* requirements. The satisfaction of the requirements is defined by the rules of engagement, which attack scenarios are approved for execution, and the structure of the methodologies. Less restrictive rules of engagement and approving more invasive attack scenarios make the penetration test more realistic, but make the test less reliable and respectful to the employees. The evaluation below assumes these two elements are tuned to the risk appetite of the organization and focuses only on the structure of the methodologies.

Reliable: In the EF methodology, the penetration tester gains possession of a non-critical asset which the custodian is prepared to lose. Thus, the result of the penetration test will not affect the productivity of the custodian. In the CF methodology, the productivity of the custodian may be affected, since the custodian does not know the asset will be stolen. The informed consent is a mechanism to avoid productivity loss, since it explicitly states not to use the marked asset for daily tasks nor store sensitive information on the asset. In both methodologies, the productivity of other employees is not affected, since the penetration tester does not gain possession of any of their belongings without their approval.

Repeatable: The repeatability of any penetration test using social engineering is questionable, since human behavior is unpredictable. Checking if a penetration test is repeatable would require a larger set of tests on a single participant, and a larger number of participants in the test.

An additional issue of repeating a penetration tests by social engineering the same employees is the experience the employees obtain from the tests. An employee that has been social engineered may be much harder to social engineer twice. However, this is a desired outcome of the tests.

Reportable: The approach used in reporting the results of the penetration test completely covers all information needed to perform the attack in a real-life situation and provides an overview of what should be improved to thwart such attempts. The logs from the tester and the monitoring equipment installed by the custodians provide detailed information on all actions taken by the penetration tester, giving a clear overview of how the mechanisms are circumvented.

Respectful: Both methodologies should respect all the employees and the trust relationships between them.

In physical penetration testing, the social engineering element is more intense than in digital penetration testing because the interaction between the penetra-

	EF methodology	CF methodology
Reliable	+++	++
Repeatable	-	-
Reportable	+++	+++
Respectful: actors	++	+
Respectful: trust relations	-	++
Realistic	+	+++

Figure 6.8: Evaluation of both methodologies

tion tester and the employee is direct, without using any digital medium. Baumrind [18] considers deception of subjects in testing as unethical. The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research, also clearly states this in their first rule of ethical principles: ”Respect for persons” [47].

However, some tests cannot be executed without deception. Finn [46] defines four justifications that need to be met do make deception acceptable: (1) The assessment cannot be performed without the use of deception. (2) The knowledge obtained from the assessment has important value. (3) The test involves no more than minimal risk and does not violate the rights and the welfare of the individual. Minimal risk is defined as: ”the probability and magnitude of physical or psychological harm that is normally encountered in the daily lives” [71]. (4) Where appropriate, the subjects are provided with relevant information about the assessment after participating in the test. Physical penetration testing using social engineering can never be completely respectful because it is based on deception. However, the deception in both methodologies presented in this chapter is justifiable.

The first two justifications are general for penetration testing and its benefits, and have been discussed earlier in the literature (for example, Barrett [16]). The third justification states that the risk induced by the test should be no greater than the risks we face in daily lives. In the EF methodology, the only actor at risk is the employee. The penetration tester cannot physically harm the employee because of the rules of engagement, thus only psychological harm is possible. If the employees help the penetration tester voluntarily, the risk of psychological harm is minimal. The logging equipment assures the interaction can be audited in a case of dispute. In the CF methodology, an additional actor at risk is the custodian. The only case when the risk is above minimal for the custodian is if the tester gains possession of the asset without custodian’s knowledge. When the custodian finds the asset missing and does not read the letter left by the testers, her stress

level might increase. Therefore it is crucial for the contact person to reach the custodian before custodian learns about the theft.

The fourth justification states that all actors should be debriefed after the exercise. In both methodologies, all actors except the employees are either fully aware of the exercise, or have signed an informed consent and are debriefed after the exercise. Similarly to Finn and Jakobsson [45], we argue that there should be selective debriefing of the employees. Debriefing can make the employee upset and disgruntled and is the only event where the risk is higher then minimal. Thus, an employee should be debriefed only if the security officer constitutes the tester did more than minimal harm.

Besides being respectful toward all the participants, the methodology needs to maintain the trust relations between the employees. The EF methodology affects the trust between the custodian and the employees and the employees and the organization. This is a consequence of the decision to fully debrief all participants in the test. The CF methodology looks at reducing these impacts. First, the custodians are not told who contributed to the attack. Only the coordinator and the security officer have this information, and they are not related to the custodian. Second, the employees are not informed about the penetration test unless it deemed necessary. However, the trust between the custodian and the contact person is shaken. Therefore, the contact person and the custodian should not know each other prior to the test.

In conclusion, the CF methodology is less respectful to the custodian than the EF methodology, because the custodian is deceived and might get stressed when she finds out the asset is gone. The EF methodology does not preserve any trust between the employees, the organization and the custodian. The CF methodology preserves the trust bond between the custodian and the employees and between the employees and the organization. However, the trust bond between the custodian and the contact person may be affected.

Realistic: The EF methodology allows testing the resilience to social engineering of employees in the organization. Since the custodian knows about the penetration test, she is not directly involved during the execution of the test, making this methodology implementable in limited number of situations. In the CF methodology, neither the custodian nor any of the other employees know about the penetration test, making the test realistic.

One might argue that if the asset is not critical for the employee, the tests are not realistic. On the other hand, taking away ”real” assets in the penetration tests will clearly cause loss of production. In the EF methodology, this issue does not exist, as the employees who may be social-engineered are not aware of the importance

of the target asset. Therefore, they have no reason to behave differently toward the experimental asset than to a ”real” asset. However, in the CF methodology, the value of the asset as perceived by the custodian might influence the result of the tests, as the employee may be more likely to give the asset away if she knows it is not critical. As future work, we plan to investigate the effect of the perceived importance of the asset on the results of such tests.

6.7 Conclusion

Securing an organization requires penetration testing on the IT security, the physical security of the location where the IT systems are situated, as well as evaluating the security awareness of the employees who work with these systems. We presented two methodologies for penetration testing using social engineering. The Custodian-Focused methodology improves on the Environment-Focused Methodology in many aspects. However, the Environment-Focused Methodology is more reliable, does not deceive the custodian and fully debriefs all actors in the test. We provide criteria to help organizations decide which methodology is more appropriate for their environment. We evaluated both methodologies through analysis of their structure against a set of requirements and through qualitative research methods by performing a number of penetration tests ourselves. This chapter shows that physical penetration tests using social engineering *can* reduce the impact on employees in the organization, and provide meaningful and useful information on the security posture of the organization.

Chapter 7

Laptop Theft: *
An Assessment of the Effectiveness of Security Mechanisms in Open Organizations

Organizations rely on physical, digital and social mechanisms to protect their IT systems. Of all IT systems, laptops are probably the most troublesome to protect, since they are easy to remove and conceal. When the thief has physical possession of the laptop, it is also difficult to protect the data inside. In this study, we look at the effectiveness of the security mechanisms against laptop theft in two universities. The study considers the physical and social protection of the laptops. We analyze the logs from laptop thefts and complement them with qualitative and quantitative analysis on the results of the orchestrated penetration tests performed using the methodologies described in Chapter 6. The results from the study show that the effectiveness of security mechanisms from the physical domain is limited, and it depends mostly from the social domain. The study serves as a motivation to investigate further the analysis of the alignment of the mechanisms across all three security domains to protect the IT assets in an organization.

*This chapter is a major revision of the paper ”Effectiveness of Physical, Social and Digital Mechanisms against Laptop Theft in Open Organizations” [1] published in Proceedings of the 2010 IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing (GREENCOM-CPSCOM ’10), pages 727-732, IEEE, 2010

7.1 Introduction

Of all IT systems, laptops are particularly hard to protect. Laptops are mobile, easily concealable, there is a big market to sell the hardware and there can be many of them in a single building. With the increased data storage capabilities of laptops, the loss of even a single laptop can induce dramatical costs to the organization [82]. Thus, although there can be a large number of laptops in an organization, losing even a single laptop may not be acceptable.

Organizations open to the public are particularly at risk from laptop theft. Hospitals and universities, for example, accept hundreds of people that can wander in the premises every day. Marshall et al. [65] points out that 46% of data breaches occur in institutions open to the public: education, health care and the government. Laptops containing sensitive medical or academic data become highly vulnerable in these environments.

The problem security professionals face is how to protect the laptops in such open organizations. There are three types of security mechanisms to secure laptops in a building: digital, physical and social mechanisms. Digital mechanisms such as laptop tracking and remote data deletion protect the laptop and the data in the laptop by using software. Physical mechanisms, such as doors and cameras, physically isolate the thief from the laptop and/or identify her in case of a theft. Social mechanisms such as organizational policies and rules decrease the number of mistakes by employees and increase the resilience of employees toward social engineering. Using digital mechanisms to protect laptops is elaborately researched by the computer science community [123, 121, 58, 89]. However, linking these mechanisms with physical and social mechanisms in protecting laptops is still not explored.

This chapter evaluates the existing physical and social security mechanisms for protecting laptops based on (1) logs of laptop thefts which occurred in a period of two years in two universities in Netherlands, and (2) 32 penetration tests (within which we had 31 successful attacks and 31 attacks that failed) in the same universities using the methodologies described in the previous chapter. The goal of the penetration tests was to gain possession of a marked laptop from an employee unaware of the penetration test. The results from the log analysis and the penetration tests show that the security of an asset in an open organization depends mainly on the level of security awareness of the employees, and to a lesser extent on the technical or physical security mechanisms. The physical and technical mechanisms have a passive, deterrent role on reducing theft, while the employees have an active, preventive role.

The outline of the rest of the chapter is as follows. In Section 7.2 we provide a literature overview on laptop theft. In Section 7.3 we describe the methodology we used in obtaining and analyzing the logs of the laptop thefts and describe how the penetration tests were prepared and executed. In Section 7.4 we analyze the logs and the results from the penetration tests qualitatively and provide general observations that can be used as a guideline for which mechanisms should be considered first in adding security mechanisms. In Section 7.5 we provide a quantitative analysis of the results from the penetration tests, and produce a logistic regression model that provides the likelihood of a success of a theft. Section 7.6 summarizes the chapter and reiterates the main conclusions.

7.2 Literature overview

There are two areas of research that focus on protecting laptops: computer science and crime science.

In the computer science community, there has been a considerable effort to model the complex security relations between the digital, physical and social domain. Scott et al. [91, 92] provides a holistic security model of the world by using spatial relationship between the elements in the ambient calculus [23]. Dragovic et al. [40, 41] presents a model which uses the physical property of objects and the sensitivity of the data inside the objects to identify possible threats. In Chapter 3, we also provided a formal model for representing and analysis of policy misalignment between the three domains. These models provide sound policy design but do not ensure effectiveness of security mechanisms that enforce these policies.

There are multiple mechanisms in computer science that work either in the physical or digital domain. In the digital domain, several security products, such as TrueCrypt[1] and BitLocker[2] provide encryption for the whole hard drive. These solutions assume the adversary does not have physical control of the laptop, because if the adversary has physical possession of the laptop, she can always successfully execute a number of attacks [114, 24, 98]. These approaches also seem to ignore the human element, or more precisely, induce performance overhead and decrease the usability of the laptop. A recent study by Panemon [81] shows that the majority of non-IT individuals, even when provided with an encrypted laptop, turn off the encryption software.

A number of tracking applications, such as Adeona [89] and LoJack [121], can

[1] www.truecrypt.org
[2] www.tinyurl.com/microsoft-bitlocker

track the location of the laptop they are installed on. In case of theft, these solutions use the Internet to provide the owner with the current location of the laptop. These solutions suffer from two problems: (1) if the goal of the theft is obtaining data from the laptop, the thief might never connect the laptop to Internet and (2) if the goal is to obtain the hardware, the thief can easily remove the tracking application by flashing the BIOS and/or formatting the hard drive.

In the computer science community, the accent is on protecting the data residing in the laptop and finding the location of the stolen laptop. The approach from the crime science community is more general, and considers the laptop *and* its environment. The goal in this field is to prevent a thief from stealing the laptop in the first place, by either changing the environment surrounding the laptop or by creating situations that will deter a thief [35]. Kitteringham [61] provides a list of 117 strategies how to prevent laptop theft. The strategies include the implementation of physical, digital and social mechanisms. Although the list is elaborate, all suggested mechanisms focus only on a single domain and do not consider any interaction between the mechanisms.

There are several other studies that analyze laptop theft. These reports focus on the money loss from a stolen laptop [82] and the frequency of laptop theft and the most affected sectors [65]. Our results are complementary, and look at the effectiveness of conventional physical and social security mechanisms in stopping laptop theft.

7.3 Methodology

Assessing the effectiveness of a security mechanism can be achieved by auditing and penetration testing. We apply both methodologies to investigate the most commonly used security systems in the physical and social domain.

First, we look at logs of recent laptop thefts in two universities in Netherlands. From the logs we obtain information about: the last control that failed before the laptop theft, alarms raised by the theft and the role of physical mechanisms in securing the laptop and finding the thief, such as access control and surveillance cameras. The logs provide valuable information on the approaches thieves use to steal a laptop. However, the logs provide limited information about the level of security awareness of the employees. In particular, the logs do not provide any information on the possible violation of social security mechanisms, such as letting strangers inside an office and sharing credentials between employees. Even in case of a burglary, the logs did not provide any information how the thief reached

	Locked office (burglary)	Open office	Restricted location	Public location	No details	Total
Stolen laptops	18	11	2	27	1	59
Cut locks	1	5	0	1	0	7
Other damage	16	0	0	0	0	16

Figure 7.1: Information from the logs. The logs from both universities are merged to anonymize the data.

the burgled office. Therefore, to better understand the effect of the security mechanisms, we orchestrated 32 penetration tests where we used social engineering to steal a laptop. The penetration tests were executed using the methodologies presented in Chapter 6. Through the tests, we observed the security awareness of the employees as well as the efficiency of the physical security mechanisms in both universities.

7.3.1 Log analysis

In a period of two years, the two universities suffered from 59 laptop thefts (Figure 7.1). The logs from the thefts provide (1) the location from where the laptop was stolen, (2) protection mechanisms on the laptop, and (3) how the theft was discovered.

7.3.1.1 Results from the log analysis

Location of the theft: In 30% of the thefts, the thief broke into a locked office either by forcing the door or breaking a window. This number indicates failure of the physical security mechanisms in both campuses. In 46% of the thefts, the laptop was stolen when the employee left it unattended in a public location, such as a cafeteria or meeting room. These thefts indicate the level of security awareness of the employees. In 19% of the cases, the theft occurred when the employee left the office for a short period of time without locking the door. These results show a combined failure of the physical and social mechanisms. The low security awareness let the users leave the laptops unattended in a restricted area, and the physical security mechanisms did not protect the laptops from being stolen.

Protection mechanisms on the laptop: In five of the thefts that occurred in an unlocked office, the laptop was locked with Kensington lock. Only one of the

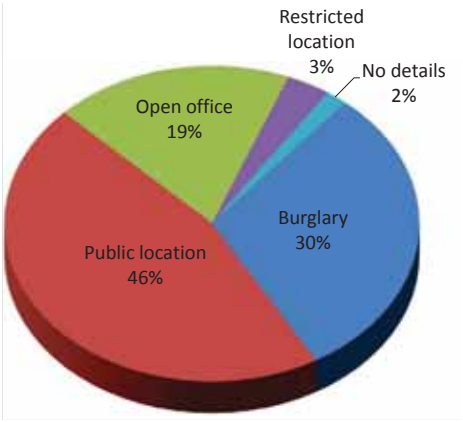


Figure 7.2: In majority of the cases, the theft occurred because the employee either left the laptop in a public location or forgot to lock the office door.

laptops stolen in a public location was locked with a Kensington lock.

Theft discovery: The majority of the thefts (93%) were reported by the laptop owner. In a few cases the report came from an employee who observed a broken door or window (5%). Only one of the thefts triggered an alarm. In this case, the thief grabbed the laptop while the employee went to collect print outs and left through the fire door, triggering the fire alarm. In all buildings, in both universities, there are surveillance cameras (CCTV) and either partially or fully centralized access control systems able to log access requests. Surprisingly, the systems provided no useful information in any of the thefts. These mechanisms are further analyzed in section 7.4.

Limitation of the logs: The logs provide information obtained after the theft took place, based on evidence found by the police and the security guards. The logs provide information only of successful theft attacks, but do not show when the security controls have succeeded in stopping the thief. Moreover, the logs do not provide information on how the thief reached the location nor on whether the security awareness of the employees contributed to the theft.

Researchers and organizations recognize that the employees are the weakest link in the organization [16, 15, 107]. Since the logs from the laptop thefts were insufficient to provide us with this information, we orchestrated a set of penetration tests where we used social engineering as a means to obtain a laptop. These tests provided us with an opportunity to validate the methodologies we introduced in the previous Chapter 6.

7.3.2 The penetration tests

To obtain more detailed information on the effectiveness of the commonly used security mechanisms in laptop theft protection, we orchestrated 3 penetration tests using the Environment-Focused Methodology and 29 penetration tests using the Custodian-Focused Methodology.

To avoid bias in the execution of the tests, we did not perform the tests ourselves, but enlisted the help from 72 master students in computer security who took the role of penetration testers. Before performing the tests we informed and received permission for the penetration tests from the chief security officers in both universities. We informed the officers exactly which locations we were going to test and the names of the staff and students involved. No other security person in the universities knew of the tests. The tests were approved by the legal department from the universities.

The students were divided in teams of three. The goal of each team was to steal a clearly marked laptop from an employee who was unaware of the penetration test. First, we did a pilot study with only three teams and three laptops. Based on the results and insights of the pilot study, we performed an additional 29 penetration tests the next two year.

As a methodology for the penetration tests, we used the Environment-Focused Methodology for the pilot study and for the rest of the penetration tests we used the Custodian Focused-Methodology from Chapter 6. The rest of the section defines (1) the roles in a penetration test, (2) the setup, (3) the execution and (4) the closure phase in the test, and discusses (5) the results and (6) the limitations of the tests.

7.3.2.1 Roles in the penetration test

We defined five roles in the penetration tests.

1 *Coordinator* - researchers from the Distributed and Embedded Security Group (DIES). The coordinators orchestrated the penetration tests.

2 *Penetration tester* - a student who attempts to gain possession of the asset without being caught.

3 *Contact person* - an employee who volunteers to distribute the asset to the custodians. In part of the penetration tests, this role was taken by colleagues of the researchers, and by the DIES researchers themselves.

4 *Custodian* - an employee at whose office the laptop was placed.

5 *Employee* - all other employees in the university who had none of the roles above.

7.3.2.2 Setup of the environment

The setup between the penetration tests following the EF and the CF Methodology varied.

In the penetration tests that followed the EF methodology, the coordinator was also the custodian, because the tests were not focused on the behavior of the custodian. We locked the laptops with Kensington locks and hid the keys in the office desk of the custodian. To monitor the laptops, we installed motion detection web cameras which streamed live feeds to an Internet server. Since the custodian shared the office with four other colleagues, the cameras were positioned in such a way to preserve the privacy of the colleagues. We told the colleagues we are doing an experiment, but we did not reveal the nature nor the goal of the experiment.

Since we knew about the penetration test, we did not allow the students to gain possession of the laptops in our presence. During the experiment, we carried on the normal work, thus the students were forced to carry on the attacks after working hours or during the lunch break.

In the penetration tests that followed the CF methodology there was a distinction between the coordinators and the custodians. In 11 of the tests we selected 4 PhD students as contact persons, who chose 11 friends as custodians (snowball sampling [49]) from University of Twente and Technical University of Eindhoven. In the other 18 tests, we selected a single contact person who randomly selected the custodians from a list of all researchers in University of Twente.

After selecting the contact people and the custodians, we marked 29 laptops as assets to be stolen. The contact persons asked the custodians to sign an informed consent form, and then distributed the clearly marked laptops, each with a web-camera and a Kensington lock. The custodians resided in two different universities in nine different buildings. To steal any of the laptops, the penetration testers needed to circumvent at least three layers of access control: the entrance of the building, the entrance of the office where the custodian works and the Kensington lock.

We gave the laptops to the custodians for two weeks usage. To avoid bias in the study, none of the custodians was aware of the real purpose of the study. Instead, we informed the custodians that the universities were conducting a usability study

on the new laptops, and thus they needed to measure the satisfaction level of the laptop users. We informed the custodians that the level of satisfaction would be measured using motion detection web-cameras that would record the usage of the laptops. The contact persons explained that they cannot tell the custodians exactly which behavior we measure, since it might change the results of the experiment.

For security reasons, the contact people instructed the custodians to lock the laptops with a Kensington lock and to keep them in the office. To reduce the risk of data leakage and loss of productivity, the contact people asked the custodians not to store any private or work data on the laptops. With these measures, we tried to reduce the risk of data leakage and loss of productivity caused by any theft.

7.3.2.3 Execution of the penetration tests

After setting up the environment, we gave each of the penetration teams the location of a single laptop they should obtain. In the first part, each team scouted their location and collected as much information as possible about the custodian and the security mechanisms at the location. Then, each team proposed a list of attack scenarios they wanted to conduct. A sample attack scenario is presented in Figure 7.4. During the second part of the test, after getting approval for executing the scenarios by the coordinator, the teams started testing.

The actions of the teams were logged using the web-cameras we positioned in the offices of the custodians and through recording devices carried by the teams during the attacks. We used such comprehensive recordings (1) to have a better overview of why the attacks succeeded/failed and (2) to be sure the employees were treated with respect by the penetration testers. The students were asked to try to avoid the CCTV cameras, to reflect the behavior of a real thief.

After each successful or failed attack, the teams provided an attack trace listing which mechanisms they circumvented and, in case of failed attacks, which mechanism caused the attack to fail. Figure 7.6 provides a summary of the successful approaches of teams and the disguises they used to obtain the laptop.

7.3.2.4 Closure

After all penetration tests were over, we debriefed the custodians and the contact people through a group presentation, where we explained the penetration test and its goal. All custodians and contact people were thanked and rewarded for helping in the assessment of the security in their university.

Figure 7.3: In twenty of the tests the custodians willingly gave the laptop or assisted in giving the laptop, either believing that the teams were from the help desk or that they were sent by the coordinator.

1. Social engineer night pass from an employee.
2. Enter the building early in the morning.
3. Social engineer the cleaning lady to access the office.
4. Cut any protection on the laptop using a bolt cutter.
5. Leave the building during office hours.

Figure 7.4: Example of an attack scenario

7.3.2.5 Results from the penetration tests

Surprisingly, all teams but one were eventually successful in stealing their laptop. Besides the 31 successful attacks, there were an additional 31 unsuccessful attacks.

The students took roles as service desk employees, students that urgently needed a laptop for a few hours, coordinator representatives or relied only on physical means to obtain the laptop. The students used mobile phones and pocket video cameras to record the conversation with the employees. In one case they took a professional camera and a cameraman, and told the custodian the recording is part of a study to measure the service quality of the service desk.

The favorite approach of the teams was to confront the custodian directly and ask for the laptop. The resistance of the custodians varied. In most cases, the custodians gave the laptop easily after being showed a fake email and being promised

Figure 7.5: In nine tests the teams social engineered a person other than the custodian. In nine of the tests the students used a bolt cutter to cut the Kensington lock or used other means to circumvent the lock.

Approach	Disguise	
Social engineered the custodian	as coordinator helpers	5
	as ICT desk	14
	as students	2
Social engineered another employee	as ICT desk	3
	as delivery person	1
Social engineered the janitor	as students	5
Social engineered the cleaning lady	as PhD student	1
Used no social engineering		2

Figure 7.6: Successful approaches and disguises of the penetration testing teams

they will get the laptop back in a few hours. In some cases the custodian wanted a confirmation from a supervisor or the coordinator. The teams succeeded in these attempts because the custodian called a number provided by the penetration testers. Needless to say, the number was of another team member pretending to be the coordinator. In one case a colleague of the custodian got suspicious and sent an email to the campus security. Since only the main security officer knew about the penetration test, in few hours the security guards were all alerted and started searching for suspicious students.

Some groups were not able to social engineer the custodian directly and were forced to look for alternative approaches. For example, in one of the cases the students entered the building before working hours. At this time the cleaning lady cleans the offices, and under the assumption it is their office let the students inside. After entering the office, the students cut the Kensington lock and left the building before the custodian arrived. A summary of all successful and unsuccessful attacks is presented in Appendix G.

7.3.2.6 Limitations of the tests

A limitation of the tests might be the high self-confidence of the testers. The security guards were not aware of the penetration test. If caught, the identification process would be unpleasant experience for the testers. Nevertheless, they knew they will not go to jail for their actions. A thief might rather wait for the laptop to be left unattended than approaching an employee directly and asking for their laptop.

7.4 Qualitative analysis

We observed three main security mechanisms in the universities: surveillance cameras, access control and the level of security awareness of the employees.

7.4.1 Surveillance cameras

Security officers do not use cameras as alarming mechanisms, but use recorded footages a posteriori, to identify an offender after an accident has taken place. The security officers cannot afford to monitor all surveillance cameras. The cameras work only when a motion is detected, and automatically store the recording in a back end server. The delay between the occurrence and report of the theft gives the thief sufficient time to leave the building.

Even when used to identify the thief a posteriori, the cameras provide limited information about the thief. In none of the logs nor during any of the penetration tests the cameras provided enough information to reveal the identity of the thief. The CCTV system is providing limited help because (1) the cameras are not mounted in offices, (2) the thief can easily conceal the laptop and (3) thieves usually know the position of the cameras and obscure their face.

The cameras are not mounted in offices. All penetration tests and 49% of the thefts took place in an office. Cameras are not mounted in offices to preserve the privacy of the employees and because mounting cameras in every office is not cost effective. Without surveillance in these offices, it is impossible to identify a thief during the act.

Instead of offices, the cameras are usually mounted on the entrance of buildings. Many people pass through the entrances with bags, and each of the bags might

conceal a stolen laptop. Even if there are only two persons observed by the camera, if the persons are not caught on the spot and challenged by the security guards, the evidence from the surveillance camera can not be used against them.

Cameras positioned to monitor public locations, such as cafeterias, halls and reception desks can record the thief during the theft. The logs show that 46% of the laptop thefts happened in public locations. During the penetration tests we noticed that these cameras are usually triggered by motion detection, and are not actively monitored by the security guards. A careful thief would obscure her face from the cameras using a hat, a hood or just covering her face with her hands before she steals the laptop. In one of the penetration tests, three penetration testers wandered with newspapers on top of their faces through the building without being challenged by anybody.

In conclusion, the surveillance system provides no help in stopping the theft and has limited usage in identifying the thief a posteriori.

7.4.2 Access control

We spotted two weaknesses of the access control in the universities. Locks are usually bypassed because (1) they are disabled during working hours and (2) the doors and windows where the locks reside are easy to force.

The access controls on the entrances of the building are easily bypassed because they are disabled during working hours and because there are too many people with credentials that can open the door. From the 32 penetration teams, the majority bypassed the entrance locks by attacking during working hours and only a few teams social engineered credentials from an employee to enter the building out of working hours.

Another attack vector for stealing a laptop is to force a door or a window. The penetration teams were not allowed to damage any property of the universities except cutting the Kensington locks. However, the logs from actual laptop thefts show that in 30% of the thefts, the thief broke a door or a window to get access to the office.

Similarly to recordings from surveillance cameras, logs from the access control systems provide limited help in identifying the thief. The logs show whose credential was used to enter a restricted area at a specific time period. Since the credentials are easy to steal or social engineer and because there are many people entering and leaving the area where the theft occurs, it is hard to deduce which person is the thief.

In conclusion, the typical access control mechanisms deployed in the universities are mainly used to deter opportunistic thieves, but provide no help against a determined thief.

7.4.3 Security awareness of the employees

The level of security awareness of the employees plays a crucial role in success or failure of a theft. The human element is the main reason behind the success of the laptop thefts. In 69% of the laptop thefts and 100% of the penetration tests, the theft occurred either because the employee left the laptop unattended in a public location or did not lock the door when leaving the office. Similarly, during the penetration tests, employees opened door from offices of their colleagues, shared credentials or handed in laptops without any identification. Therefore, even with strong access control in place, if the security awareness of the employees is low, the access control can easily be circumvented.

On the other hand, the human element is the main reason behind the failure of 67% of all failed penetration tests. In these cases, an employee informed the security guards for suspicious activities, rejected to open a door for the tester, rejected to unlock a laptop without permission from the custodian or interrupted the tester during the theft. In these cases, the employees besides enforcing the access control mechanisms, also played a role as an additional surveillance layer around the laptop.

Employees are usually considered as the weakest link in the security of an organization. We observe that employees can also be the strongest link in the security of open organization. A proper security education of employees increases the employee's resistance to social engineering, and increases effectiveness of the other security mechanisms.

7.4.4 Limitations of the observations

The observations from the test and log analysis is based on the security mechanisms in two open institutions. The observations may apply to other mobile assets, such as medical equipment, beamers and mobile phones in institutions open to the public. However, other types of organizations might have different spectrum of mechanisms for protecting their laptops.

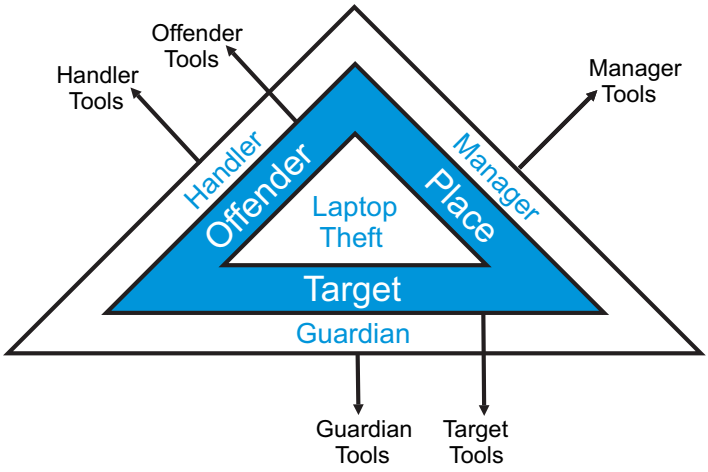


Figure 7.7: Representation of the concepts from routine activity theory

7.5 Quantitative analysis

In this section we develop a statistical model that predicts the *probability* that a theft will succeed based on the presence of guardians, managers and attack tools.

To achieve the goal, first we use Routine Activity Theory to define a number of variables that describe the attack traces from the penetration tests. Then, we calculate the correlation of the variables with the success of an attack scenario. Finally, we use the variables that are statistically significant in a multivariate logistic regression, to construct a model that will predict the probability of an attack succeeding based on the values of the variables.

The results from the analysis (1) can aid to better allocate protection mechanisms in organizations and (2) can help generate/differentiate attack scenarios likely to succeed during a penetration test.

We use Routine Activity Theory (RAT) [44] as a framework to analyze the results from the penetration tests. For an attack to succeed, routine activity theory states that three elements need to converge in space and time: a motivated offender, a suitable target and the absence of a capable guardian. Capable guardians are people that supervise people or property. When an offender and target exist, the absence of a capable guardians allows the crime to occur. Besides the guardians, there are two other groups of people that may prevent a crime, handlers and managers [43]. A handler, such as a police officer, monitors likely offenders, while a manager, such as a security guard, monitors amenable places. The concepts of routine activity theory are presented in Figure 7.7.

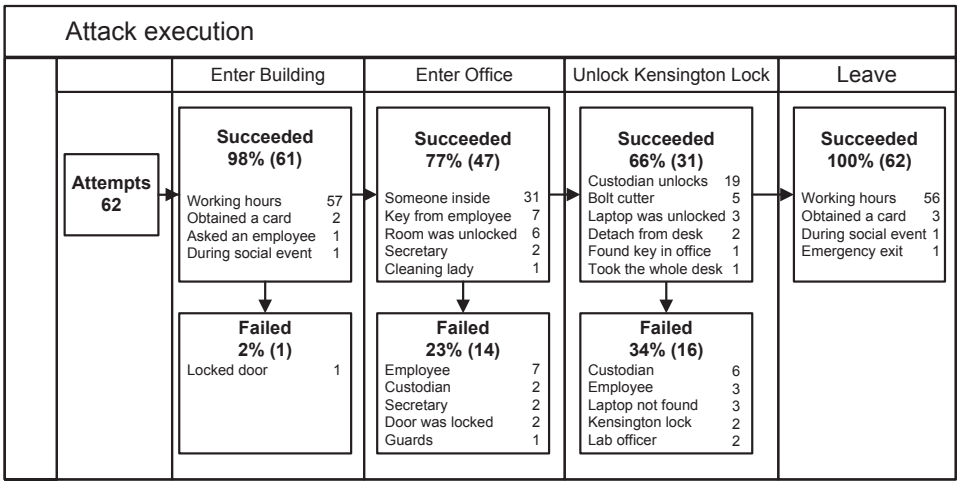


Figure 7.8: Successful and unsuccessful attacks. The figure includes the variables that contribute to the success of the attack, as well as reasons why an attack failed. The majority of the attacks failed while the testers where trying to circumvent the entrance to the office of the custodian (14) and the Kensington lock (16). In all but one attack the testers managed to enter the building and in all attacks the testers managed to leave the building.

During the penetration tests, we chose the offender (the penetration testers) and the target (the laptop), and focused on the capable guardian (the custodian) and the managers. Managers are the people whose presence in the location might prevent the theft, such as the security guards, cleaning ladies, janitors, secretaries, and employees in the office where the laptop is situated. The managers and guardians have tools that help them thwart the likely offenders, such as (1) the entrance to the building, (2) the entrance to the office, (3) the Kensington lock and (4) the exit from the building with the target. Similarly, the offenders have tools to help them steal the laptop, including physical theft and social engineering

7.5.1 Selection of the variables

Our dataset used in the analysis consists of 62 attacks distilled from the attack traces generated by the 32 penetration tests, from which 31 are successful and 31 not successful. Figure 7.8 provides a summary of how the testers managed to circumvent each layer of defense, as well as why some attacks failed.

We defined 30 variables that belong to one of the elements of interest in the routine

Variable	Description	Encoding	
		Yes	No
SocialEng	An individual is social engineered	1	0
PhysTheft	A physical theft took place	1	0
SomeoneInside	The testers entered an unlocked, non-empty office	1	0
CustodianUnl	The custodian unlocks the Kensington lock	1	0
ICTEmployee	The testers took the role of an ICT Help Desk employee	1	0
Custodian	The testers approached the custodian during the attack	1	0
Employee	The testers approached another employee during the attack	1	0

Figure 7.9: Independent variables

activity theory. We divided the variables in three groups: (1) how the four layers of defense were circumvented, (2) which managers were circumvented and (3) which tools the testers used during the attack.

From the attack traces we defined 19 variables that belong to the first group. These variables together with the frequency of occurrence are presented in the successful attacks in Figure 7.8. For the second group, the managers, we used 4 variables that specify the roles of the people that were approached during the tests: custodian, employee, janitor, cleaning lady. Finally, for the third group we used 2 variables for the technique the testers used, social engineering and physical theft, as well as 5 variables based on the roles the testers took during the attack: ICT Help Desk, PhD student, student, assistant to the coordinator or no role. The complete lest of variables is presented in Appendix H.

7.5.2 Correlation between the variables

We used the SPSS statistic package to correlate the values of the variables with the results of the attacks. Of the defined variables, 8 had a sufficient statistical significance of the correlation with the success of the attack. Of these variables 7 were present at a sufficient number of successful attacks (10 or more) and were included in the further analysis. The dependent variable in the analysis is whether the attack succeeded or failed, while the independent variables which had sufficient statistical significance are presented in Figure 7.9. A summarized result of the analysis for these independent variables is presented in Figure 7.10.

The points of interest in Figure 7.10 are the ratio between successful and total number of attack for a given value from one of the independent variables, and the correlation between the independent and dependent variable. For example, there are 31 attacks where the office was empty. From these attacks, only 11 succeeded (36%). There were another 31 attacks when someone was in the office,

Variable	SomeoneInside		CustodianUnl		Custodian		Employee	
Value	0	1	0	1	0	1	0	1
Succeeded / Number of attacks	11/31	20/31	12/43	19/19	10/34	21/28	27/47	4/15
In %	36%	65%	28%	100%	29%	75%	57%	27%
Poarson's R	0.29		0.67		0.45		-0.26	
Significance test	0.02		0.00		0.00		0.04	
Variable	ICTEmployee		SocialEng		PhysicalTheft			
Value	0	1	0	1	0	1		
Succeeded/ Number of attacks	16/41	15/21	1/15	30/47	23/52	8/10		
In %	39%	71%	7%	64%	44%	80%		
Poarson's R	0.31		0.49		0.26			
Significance test	0.02		0.00		0.04			

Figure 7.10: Relation between the distilled variables and the success of an attack

from which 20 succeeded (65%). Thus the information for the variable *SomeoneInside* can be interpreted as: *When someone was inside the office (custodian or employee), allowing the testers to get inside the office, 65% of the attacks succeeded. Only 36% of the attacks succeeded when the testers needed to find a different way to circumvent the office door.*

The next two row in Figure 7.10 present the strength and direction of the correlation between the variable in question and the outcome of the result and its significance. The significance provides the probability of obtaining the current distribution of frequencies, or one that is more uneven. The direction of the correlation between the variable *SomeoneInside* and the outcome of the attack is positive, and the strength of the correlation is 0.29. The significance of this result is 0.02 ($<$0.05), which leads to the conclusion that there is a chance of 2% to get such or more uneven distribution of the frequencies randomly.

Interpretation of the results

The first two variables, *SomeoneInside* and *CustodianUnl*, show which ways of circumventing the layers of defense (the managers) correlate the most with the results of the attacks. The circumvention of the entrance to the office is more likely when there is someone inside, because then the entrance is unlocked. Similarly, when the custodian unlocks the Kensington lock for the testers, the attack always leads to a success. From these two instances we can conclude that the defined layers of defense are not very useful because they can be disabled by the custodians and other employees that have access to the office where the target resides.

The second pair of variables, *Custodian* and *Employee*, belong to the group of variables describing the guardian and the managers that were approached during the penetration tests. Approaching the custodian as part of the attack positively correlates with the success of the attack. This is an expected result, because if the custodian agrees (even indirectly, by phone or mail) to give the laptop to the tester, the managers usually do not interfere with the execution of the attack. However, there is a negative correlation with the *Employee* variable. When employees other than the custodian are approached during an attack, they often refuse to give access to the laptop, and in most times refer the tester to come back in the office when the custodian is present. From these two instances we can conclude that the managers are more likely to interfere with an attack, while the guardians are more likely to give the device away.

The correlation of the last three variables in the analysis, *ICTEmployee*, *SocialEng* and *PhysicalTheft*, describes the dependency between the roles and techniques used by the testers, and the outcome of the attacks. When the testers took the role of an employee from the ICT Help Desk, the managers and guardians were more likely to give away the laptop rather than when the testers assumed a different role or no role at all. From the techniques used, both when the testers used social engineering as part of the test and when they used physical theft, the theft was more likely to succeed.

7.5.3 The success likelihood of an attack

We use the selected 7 variables to build a regression model. The regression model provides information on how the modification of the variables in an attack scenario (adding or removing any of them) can change the probability of a success of the attack. The probability of a success of an attack scenario depends on the influence of each of the identified properties in the scenario and their cumulative influence:

$$P_{success} = \frac{1}{1+e^{-z}}$$

$$z = \alpha + \beta_0 X_0 + ... + \beta_n X_n$$

where P represents the probability of success, and depends on the constant α, the regression coefficients β_1, , β_n that are linked to the independent variables X_1,, X_n and the values of these variables.

To build the model we use the stepwise backward multivariate logistic regression from the SPSS package. From the selected 7 variables, we discounted *CustodianUnl* because it gave uninterpretable results. The analysis finished in 4 steps,

	Variable	B	S.E.	Wald	df	Sig.	Exp(B)
Step 4	X_0: SocialEng	5.0	1.5	11.8	1	.001	151.48
	X_1: PhysicalThefts	3.1	1.3	6.0	1	.014	23.14
	X_2: Employee	-3.1	1.0	11.6	1	.001	0.05
	C : Constant	-3.8	1.4	7.2	1	.007	0.02

Figure 7.11: Variables in the equation. Variables entered on step 1: SocialEng, PhysicalTheft, SomeoneInside, ICTEmployee, Custodian, Employee.

discounting one of the variables in each step. The summarized result is presented in Figure 7.11.

The values for B represent the coefficients in the logistic regression equations. $Exp(B)$ is the exponentiated coefficient, yielding the odds ratio. $S.E.$ represents the standard error associated with each of the coefficients. As only a single independent variable is involved in each test the degrees of freedom (df) equals 1. We used the $Wald$-test to confirm the validity of our model. In order for an independent variable to be statistically relevant, the Wald value needs to be compared to a Chi-squared distribution. This gives the significance, which is also presented in the table. A variable is said to be significant when the significance is 0.05 or less [9]. In our case, as can be seen in Figure 7.11, we found that the three independent variables *SocialEng*, *PhysicalTheft*, *Employee* and the constant are significant. Thus, we can predict the chance of success for a given attack scenario based on knowledge of whether the social engineering and theft is used and whether employees are involved using the function $P_{success}$:

$$P_{success}(X_0, X_1, X_2) = \frac{1}{1+e^{-(-3.8+5.0X_0+3.1X_1-3.1X_2)}}$$

Interpretation of the results

From the results we can deduce that the use of social engineering appears to have the biggest influence on the model's outcome (increasing the predicted probability). Involvement of physical theft increases this probability even more. However, when managers are involved, the probability of the success of the attack decreases. In case none of the independent variables are true, the predicted probability will be only based on the constant coefficient.

We can evaluate this model by comparing the observed outcome with the result predicted by the model as depicted in Figure 7.12. The result shows that overall, 83.9% of the cases would be correctly predicted by the model. The probability of obtaining a false positive (22.6%) from the model seems to be slightly higher than the probability of a false negative (9.7%).

Step 4	Observed	Predicted		
		Result		Percentage correct
		0	1	
The outcome of the attack	0	24	7	77.4%
	1	3	28	90.3%
Overall percentage				83.9%

Figure 7.12: Classification table

Limitation of the results

Our dataset has 62 attacks. When using logistic regression, problematic results are known to occur if there are not enough events compared to the number of independent variables [32]. There are three different types of problems or errors: underfitting, overfitting and paradoxical fitting. These are respectively when important variables are omitted, when too many variables are used or when a variable actually suggests a factor to have a certain impact when in fact the opposite is true. To keep the model valid, these errors must at least be kept to a minimum, in order to do this it is suggested to use at least ten events per independent variable [33, 78, 79]. An event is the smaller number of the binary outcome of the logistic model. In our case an event is the successful theft of a target laptop, and the number of successful and unsuccessful attacks is the same, 31. In our model, we have 3 independent variables, which meets the minimum of 10 events per variable. To make stronger claims on the results we need to increase the dataset, which can be achieved with the execution of additional penetration tests.

7.6 Conclusion

In this chapter we evaluated the security mechanisms from the physical and social domain that influence laptop theft in organizations open to the public. We analyzed the logs of laptop thefts which occurred in a period of two years in two universities in Netherlands. We complemented the findings from these logs with 32 penetration tests using the methodologies in Chapter 6, in which we used social engineering to gain possession of marked laptops.

We observed that (1) mechanisms from a single domain can provide only limited protection against laptop and (2) the effectiveness of a physical security mechanism depends mainly on its alignment with security mechanisms from the social domain. From the quantitative analysis of the results from the penetration tests, we obtain two conclusions. First, in the crime prevention domain, we conclude

that people managers have the biggest contribution in stopping a theft. Second, in the penetration testing domain, using social engineering as a tool provides the biggest probability of successful acquisition of the laptop.

Chapter 8

Training Students to Steal*
A Practical Assignment in Computer Security Education

Practical courses in information security provide students with first-hand knowledge of technical security mechanisms and their weaknesses. However, teaching students only the technical side of information security leads to a generation of students that emphasize digital solutions, but ignore the physical and the social aspects of security. In the last two years we devised a course where students were given a practical assignment which includes a combination of physical security, social engineering and digital penetration testing. As part of the course, the students took the role of penetration testes, and using the methodologies presented in Chapter 6 obtained laptops from unaware employees throughout the university campus. The assignment provided the students with a practical overview of security and increased their awareness of the strengths and weaknesses of security mechanisms. The penetration tests executed by the students helped us draw conclusions on the effectiveness of security mechanisms presented in Chapter 7. In this chapter we present the design of the practical assignment and the observations from the execution.

*This chapter is a minor revision of the paper "Training Students to Steal: A Practical Assignment in Computer Security Education" [3] published In Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11), pages 21-26, ACM, 2011

8.1 Introduction

Educational institutes are starting to offer specialized CSIA (Computer Security and Information Assurance) courses, designed to train students in assessing and improving the security of digital systems. Computer security focuses on the protection of data from theft and corruption by using a combination of physical, digital and social mechanisms. Physical mechanisms focus on restricting and detecting physical access to the data, such as locks, CCTV, infrared sensors and heat sensors. Digital mechanisms focus on digital detection and protection of the data. Common digital mechanisms are firewalls, intrusion detection systems and encryption. Finally, social mechanisms focus on increasing the security awareness of the employees and reducing mistakes from human factors. Examples of social mechanisms that improve security are lectures on social engineering and clearly defined policies.

Graduate courses in computer security often provide a narrow view on security and focus mostly on the digital aspects (Figure 8.1, dashed line). Such a focus provides an unrealistic view of the security requirements of an organization and leads to students assuming that digital means ensure secured data.

However, most of the attacks performed by insiders require no technical knowledge [86]. In the literature there are numerous examples where an adversary uses social engineering and physical access to obtain data [16, 69, 11]. Thus, it is important to get the students acquainted with attacks in which the hacker uses also physical and social means to compromise the data (Figure 8.1, solid line).

Practical assignments clarify and support the theory students learn. In practical assignments students use the same methods and tools that hackers with malicious intent use to gain access to information. The usage of practical assignments in computer security is performed as part of many computer security courses, such as in forensics [37], in education on spam [95] and in social engineering [42]. We believe that students need to understand the hacking mentality and see how an adversary would attack also in the physical and the social domain of information security. That can be done by giving students first hand experience of the effectiveness of the physical and social security mechanisms, exploring which attack vectors are more likely to succeed than others.

In this chapter we present the practical assignment of an introductory graduate course in computer security. The goal of the course is to give a broad overview of security to the students and to increase their interest in the field. As part of the course, the students steal laptops from unaware employees, mount offline attacks on the laptop and attack a vulnerable server using the data from the laptop.

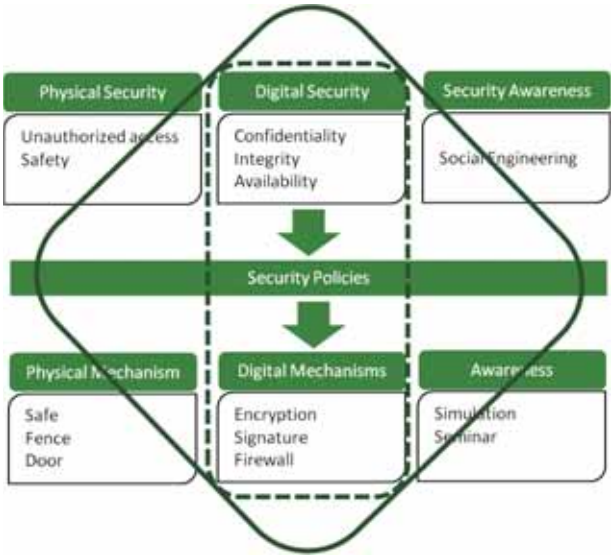


Figure 8.1: Computer security in context

In section 8.2 we present the course and give a description of the practical assignment as well as observations from the execution of the assignment. In section 8.3 we present in greater detail the practical and ethical implications of the physical penetration test from the assignment. In section 8.5 we summarize our experience.

8.2 Course description

Since 2008 we have taught a class on introduction to computer security to graduate students. The duration of the course is eight weeks, in which the students need to write a scientific paper and take part in a practical assignment which can be either suggested by them or by the lecturer. The course is part of a master track in computer security, and introduces the students to all concepts in security. The rest of the courses in the security track provides in-depth knowledge in different aspects of information security. The goal of the introductory course is threefold:

1. Describe important concepts in computer security from the perspective of physical, digital and social security.
2. Prepare the students to place security mechanisms in an overall security context; for example, design a system or analyze a situation and determine what the different physical, digital and social mechanisms could achieve in

Figure 8.2: The steps in the practical assignment

a given scenario or what techniques could be applied to reach a given goal.

3. Provide students with a first-hand experience with the strengths and weaknesses of security mechanisms from the physical, digital and social domain.

As part of the class, we provide a practical assignment where the students take the point of view of an adversary.

The practical assignment is divided in three exercises, (1) physical penetration exercise using social engineering, (2) offline attacks on a laptop and (3) online attacks on a vulnerable server (Figure 8.2).

8.2.1 Physical and social engineering attacks

The goal of the physical penetration exercise (1), is to make the students aware of the social engineering and physical activities an attacker can use to get sensitive data, by stealing a laptop. After this exercise the students should have knowledge of social engineering and physical security, and know the threats that arise from them. For the first part of the assignment we used the methodologies described in Chapter 6.

In 2008 we performed a pilot study with 9 students divided in 3 groups. The groups performed 3 penetration tests using the EF methodology. After positive feedback and applying the lessons learned in the pilot study, we gave the practical assignment to all students in the class of 2009, 11 groups of 3 students. In the second year, the students performed the penetration tests using the CF methodology.

Each laptop the students obtained was protected with at least three layers of access control: the entrance of the building, the entrance to the office of the employee and a Kensington lock. After setting up the environment, we gave each of the teams the location of a single laptop they should obtain. First, each team scouted their location and collected as much information as possible about the employee and the

security mechanisms in place. Then, each team proposed a list of attack scenarios they wanted to conduct. Each scenarios was approved by us and the security management. The students had two weeks to gain possession of the laptop. The actions of the teams were logged using the web-cameras we positioned in the offices of the employees and through recording devices carried by the students, such as mobile phones. We used such comprehensive recordings to be sure the employees were treated with respect by the students. After each successful or failed attempt, the teams provided an attack trace listing which mechanisms they circumvented and, in case of failed attempts, which mechanism caused the attack to fail.

8.2.2 Offline attacks

The second part of the practical assignment (2) consists of offline attack on encrypted data. The goal of the offline attacks, is to make the students aware of the strength of the current encryption mechanisms, as well as the state of the art hacking tools. After this exercise the students should have knowledge of most commonly used tools for encryption of data and their vulnerabilities.

During the pilot study, on each of the laptops marked for stealing we put three copies of a file containing the IP address of one vulnerable server. One copy was encrypted with WinZip[1], the other copy with TrueCrypt[2] and the third with BitLocker[3].

During the actual assignment, to avoid privacy breach of the custodians, we installed the copies of the file on laptops provided by the students rather than on the laptops marked for stealing.

In this part of the assignment, the students used offline attacks, such as the Cold-boot attack [53] and/or password cracking tools such as John the Ripper[4] and Hydra[5] to obtain the IP address from the encrypted file.

[1] www.winzip.com
[2] www.truecrypt.org
[3] bitlocker.
[4] www.openwall.com/john
[5] freeworld.thc.org/thc-hydra

8.2.3 Online attacks

During the digital penetration testing exercise, the students need to use the IP address obtained from the second part of the exercise as an attack vector in online attack on a vulnerable server (3). The goal of the exercise was to give the students an overview of the current online techniques in compromising a system. After the exercise, the students should to be able to use the common tools used in penetration testing, and know their capabilities.

As a target server, we used a number of virtual machines running Windows and Linux operating systems. On each machine we set a web site in which we introduced a vulnerability, either SQL injection or buffer overflow.

In the last exercise the students used Backtrack[6] to attack the vulnerable server and obtain a protected file.

In this chapter we focus only on the first exercise of the assignment, the physical penetration test. The second and third exercise can be considered as a reproduction of exercises reported by other authors [37, 68, 13].

8.3 Implications

Running the practical assignment is challenging. Most of the issues we faced we discuss in Section 6.4.6 and Section 6.5.6 in Chapter 6.

During the design of the course, we had four major concerns. First, the execution of the assignment might violate the law. Second, the assignment is executed in an environment where the outcome cannot be controlled. Thirdly, the assignment teaches students to steal and lie. Finally, the assignment includes deceiving employees. In the rest of this section we explore the implications.

8.3.1 Legal implications

To ensure that we were operating within the law, before deploying the assignment we consulted the legal department of the university. Following the advice from the legal department, we forbade all scenarios that included (1) theft of any object besides the laptop, (2) searching through the belonging of the employees and (3) impersonation of officials of the university, police, fire department, etc.

[6] www.backtrack-linux.org

8.3.2 Reducing unexpected outcomes

The assignment was executed on the campus of the university. This is a semi-controlled environment, where we could neither fully control the behavior of the students nor the behavior of the employees and the security personnel. We applied the following principles to the design of the exercise:

1. *Limit the scope of the activity.* The students were not allowed to use intimidation, violence nor to put the employees or themselves at risk. They were also forbidden to cause any physical damage, except for cutting the Kensington locks. The laptops were clearly marked and the students were allowed to gain possession only of a specific laptop. All students signed the rules of engagement before the scouting phase.

2. *Control the risk.* All attack scenarios were approved in advance by us, and only minor deviations in the execution were allowed. The web-cameras and the students recorded with video/audio of all their activities. The recordings allowed us to see if the employees were treated with respect or were put at risk. Finally, all students were given a lecture on the ethical aspects of social engineering and theft.

3. *Reduce the impact of the exercise.* Just after the execution of the task, the students reported to us. When the laptops were stolen without knowledge of the employees, we tried to inform the employees before they found the laptop gone. The employees were briefed from the beginning not to store any sensitive, private nor critical information on the laptops, and to use them only for gaming and surfing. At the end of the assignment, we properly debriefed all the employees, and provided small gifts for the participation.

4. *Introduce escape clauses.* The employees and the students participated voluntarily in the study. Both groups were aware, and signed an informed consent stating they can stop with the activity at any time. All students were given ”get out of jail” cards in case they are caught by the security guards. The cards contained phone numbers of the security management and the lecturers. The security management had information about all the target employees and all the students who participated in the exercise.

8.3.3 Ethical implications for the students

During the design of the first part of the assignment, the physical penetration tests, we were concerned (1) whether the students would feel comfortable with the activity and (2) whether we were training a future generation of criminals.

Benefits	Risks
1. Practical overview of all aspects of security. 2. Awareness of the strengths and weaknesses of security mechanisms.	1. Reduced comfort level. 2. Misuse of the knowledge.

Figure 8.3: Risks and benefits from the assignment to the students

8.3.3.1 Comfort level of the students

Before the start of the assignments, all the students were given the opportunity to perform an experiment of their choice instead of participating in the assignment. The only limitation was that the experiment should have the same workload as the assignment. All the students decided to join the assignment.

Survey among the students

To evaluate their level of comfort we devised two questionnaires, one before the exercise and one after the exercise. We had 31 respondents (94%) for each of the questionnaires. The students graded on a scale of 1 (strongly disagree) to 5 (strongly agree) how much they agree with a statement.

The majority of students felt comfortable during the assignment. The number increased from 65% before the execution of the exercise, to 77% after the exercise. Most students thought the assignment would be fun, but the percentage dropped from 71% to 65% after conducting it. At the beginning students felt less comfortable because they were not sure what kind of attacks they would perform. After we approved only the low risk scenarios, the comfort level increased, but the fun part of the activity decreased. 68% said they would repeat the assignment if they were again given the chance. The results are summarized in Figure 8.4.

The students filled out another questionnaire after finishing the course, as part of the standard quantitative evaluation of courses in the university. The results show the satisfaction of the students increased as well as their attendance in class compared to the previous year (Figure 8.5). The grade of the course increased a whole point, from 6.8 in 2008 to 7.8 in 2009, thanks to the enthusiasm of the students for the practical assignment. The average grade of the rest of the courses were 7.2, both in 2008 and 2009.

Question	Av.	SD
Before the exercise:		
The exercise will be fun	4.6	0.7
The exercise is useful	4.1	0.8
I feel fine about the exercise in general	3.8	1.1
I feel fine about the ethical implications of the exercise	3.5	1.2
After the exercise		
The exercise was fun	4.5	0.7
The exercise was useful	4.1	0.9
I feel fine about the exercise in general	4.1	1.0
I feel fine about the ethical implications of the exercise	3.6	1.2
I would do the exercise again	3.9	1.2
I am now more aware of physical and social security	3.9	0.9

Figure 8.4: Results from the students before and after the first part of the assignment

Year:	2008	2009
Respondents:	40 (100%)	28 (90%)
1. The course was well organized	7.4	8.6
2. My attendance was above 80%	7.3	8.9
3. I liked the practical assignment	5.6	8.2
4. Overall grade of the course	6.8	7.8

Figure 8.5: Results from the students after passing the course

8.3.3.2 Risks of teaching students to steal

Checking if we are teaching the next generation of criminals is a more subtle issue. The benefit of educating students in the adversarial aspect of security is widely discussed and implemented in many security courses. Pashel [76] and Logan and Clarkson [64] discuss the ethical implications of teaching students to hack and the possibility of misusing the acquired knowledge. We show that the arguments in favor of teaching digital penetration testing also hold for the physical domain, by establishing an analogy between digital and physical penetration testing.

Analogy to teaching digital penetration testing

Question	Average	SD
1: Deceiving the subject	3.5	1.2
2.1: Physical damage - Discomfort	2.3	1.1
2.2: Physical damage - Injury	1.3	0.8
2.3: Physical damage - Death	1.0	0.0
3.1: Material damage - Emotional	2.4	1.2
3.2: Material damage - Financial	2.8	1.3
3.3: Material damage - Production loss	2.1	1.4
4.1: Psychological damage - Threats	2.1	1.0
4.2: Psychological damage - Deception	4.2	0.8
5.1: Privacy - Assume identity	4.0	1.0
5.2: Privacy - Access sensitive informa.	3.6	1.3
5.3: Privacy - Destroy information	2.3	1.1
5.4: Privacy - Theft of information	3.2	1.3

Figure 8.6: Ethical acceptability of damage according to students

According to Pashel [76] and Logan and Clarkson [64], teaching hacking to students is mostly justified, because to provide the best security defense, a system administrator must possess the same skills as the attacker. We consider this to be the *Locksmith Argument*. For any locksmith to be able to create decent locks they also need to have the ability to break locks (or at least have extensive knowledge on the techniques of a lock picker). The same argument can be applied to teaching physical penetration testing. The only way a student is able to secure physical objects is to have extensive knowledge on how attackers penetrate organizations, buildings and so forth. Letting them gain experience from an attacker's point of view will positively affect this knowledge.

Another argument in favor of teaching students digital penetration testing is that these skills are useful in discovering weaknesses in the security of a system [64]. The same argument can be applied to physical penetration testing. For example, an insurance company needs to review the physical security (such as cameras and security guards) and digital security (the network infrastructure that controls the cameras, the locks and the alarms) of a museum before determining the insurance premium.

Survey among the students

In the questionnaires we gave to the students, we also asked for their opinion on ethical issues. The students were told to assume there were no rules in the assignment, the only objective was to obtain the laptop.

Benefits	Risks
1. Increased awareness of the employees. 2. Checks the security mechanisms in the university.	1. Employees are deceived. 2. Employees or their data might be put at risk.

Figure 8.7: Risks and benefits from the assignment to the employees

We asked what type of damage the students are willing to inflict on the employee or the surroundings: physical, material, psychological damage and invasion of privacy. Each type of damage consists of some subtypes that contain examples of such damage, varying from light to severe (in our perception). In this way, we could identify the ethical sensitivity of types of damage, in the perception of the students.

The results from this survey are shown in Figure 8.6. The scale is from 1 (uncomfortable / I will never do that) to 5 (comfortable / I have no problem doing that). The questionere was filled out by 28 students (85%).

Physical damage is a sensitive matter. Even light physical damage or emotional damage was rated only around 2. The roots for this rating can be found in the basic ethics of society e.g. ”do not hurt people” and ”respect your fellow human beings”. The students are less concerned with material damage than physical damage. Ratings are rather spread with this type of damage: some students do not care about material damage at all while other students do feel very uncomfortable causing material damage. It is surprising to see that the students feel uncomfortable with causing production loss.

Furthermore, threats and intimidation are again sensitive according to the students: The rating averages around 2. Deception however does not seem to be such a big problem, most students have no difficulty in feeling comfortable with deceiving the employee. It is also surprising to see that privacy issues do not make the students feel very uncomfortable. Destroying information does rate as very uncomfortable, but other types of privacy issues tend to rate toward comfortable.

8.3.4 Ethical implications for the employees

We describe the ethical implications of the penetration tests to the employees in Section 6.6 from Chapter 6. Physical penetration testing using social engineering can never be completely respectful because it is based on deception. However, the deception used in the assignment presented in this chapter is justifiable.

Question	Av.	SD
1. I found the exercise interesting	4.2	0.8
2. The exercise increased my awareness	3.8	1.2
3. The exercise should be done more often	3.8	0.9
4. During the exercise I found myself stressed	1.9	1.1
5. I find the assignment ethical	3.6	1.0
6. These exercises are harmful	2.1	0.9
7. These exercises will benefit students	3.7	0.8
8. The security awareness of students and employees can be improved through such exercises?	4.4	1.0

Figure 8.8: The view of the employees

Besides the 11 penetration tests we executed as part of the presented assignment, we orchestrated additional 18 penetration tests in 2010. After the debriefing, we asked the employees to fill in a survey. Twenty four employees (83%) from which the students obtained the laptop filled a questionnaire after the debriefing.

They answered multiple questions, on a scale of 1 (I strongly disagree) to 5 (I strongly agree) and yes/no questions. The results are summarized in Figure 8.8.

Most of the employees said that the university should continue letting graduate students perform these assignments (71%) or did not have opinion on the topic (29%), but none of the employees was for stopping the training. 94% of the employees, also agreed that these kinds of assignments can improve the security awareness, both that of the students and of the university employees.

8.4 Using Portunes to produce attack scenarios

In 2010 we ran the practical assignment again with two modification. First, the students had to steal a laptop twice. Once using attack scenarios generated by Portunes, and once using attack scenarios generated through brainstorming. Second, because of the extra work for the students, we removed the online attacks on vulnerable servers from the practical assignment.

The goal of these changes was to focus on the suitability of Portunes to generate attack scenarios for penetration testers and to compare the quantity and quality of the scenarios produced by Portunes with the attack scenarios the testers can come up by traditional ways (brainstorming).

8.4.1 Setup of the practical assignment

In 2010 we had 9 teams of 3 students. The teams first got a target and scouted the target for one week. Simultaneously they were reading on how to use the Portunes tool to produce models and generate attack scenarios.

After the scouting phase, the teams met for 3 hours in a lab. Here the teams were divided in two groups. The first group of 5 teams was supposed to brainstorm as many attacks as possible for their target, while the second group of 4 teams was supposed to generate a Portunes model and automatically produce the attack scenarios.

After the 3 hour lab exercise, the teams had one day to write the results in a general template so it is impossible to see whether the attack scenarios they are produced by Portunes or brainstorming. We collected these reports and randomly distributed them to the teams. Upon receiving a set of attack scenarios, the teams had to grade them and then execute them.

During the second round of penetration tests, the setup was the same, but the groups changed their roles. The group that generated the attack scenarios using Portunes in the first round of penetration tests, generated them through brainstorming, and vice versa, the group of teams that generated the scenarios through brainstorming generated them using Portunes.

8.4.2 Unanticipated difficulties

During the execution of the exercise, we faced a few unanticipated difficulties, which hindered us in drawing conclusions.

First, during the lab work it become apparent that most of the teams did not read the instructions how to use Portunes. For example, the teams were downloading the Portunes tool for the first time during the lab work, and did not know the concepts of Portunes nor how to use the tool to start building a model. Without this knowledge they could not produce any meaningful result within the allocated 3 hours. Thus 2 of the 4 teams that were supposed to generate attack scenarios using Portunes were allowed to use brainstorming.

Second, we faced a few bugs in the Portunes GUI. For example, when generating all the attack scenarios, the model gets modified. There was no button to revert to the original Portunes model. This forced the teams to save the Portunes model every time before they generated the attack scenarios. These bugs together with

the time limitation of 3 hours, limited the size of the model the students could generate.

Third, during the second round of generating attack scenarios, the teams that generated scenarios using brainstorming (7 out of the 9 teams), produced models such that generate the same attack scenarios they produced using the brainstorming. We did not have a clear policy on this before the start of the assignment. In addition, the time between handing in the reports and the deadline for starting the execution of the penetration tests was one day, leaving no time to repeat the lab work. Thus, we accepted these scenarios as valid.

Finally, we had no clear policy how the quality of an attack scenario should be graded. The positive or negative outcome of an attack depends from many variables which are not part of the attack scenario. Thus, we asked for subjective grading from the students based on the likelihood of the scenarios to be successful. We found this reasonable, because the students were the ones that will eventually execute the scenarios. However, each set of attack scenarios contained a scenario where they mimic an ICT employee. In this scenario they state to the custodian there is a virus in the laptop and they need to take the laptop for inspection. When executing the scenarios, the teams most often ignored the other produced scenarios and executed the ICT Desk attack scenario (14 out of the 18 tests). Thus it is hard to determine how realistic were the other scenarios produced both by Portunes and through brainstorming.

With the above mentioned limitations of the assignment, we could distill the following conclusions:

- Portunes can generate realistic, executable scenarios similar to the ones generated by brainstorming. The students were able to generate the same scenarios from the brainstorming using Portunes. The average number of scripts is almost equal (13 for manual against 12 for Portunes). There was a non-significant difference in the average grading with 1 point (6.5 for manual against 5.5 for Portunes).

- Using Portunes, a penetration tester can generate a set of attack scenarios within 3 hours. All scenarios the teams generated were within the allocated slot of 3 hours.

8.5 Conclusion

To make students aware of physical and social aspects of security, these aspects need to be included in security courses, both from a theoretical and a practical point of view. We presented a practical assignment consisting of three steps. First, the students needed to steal a laptop from an unaware employee, then decrypt a document from the laptop, and finally use the information from the document to attack vulnerable servers.

As part of the assignment the students used the Portunes framework to automatically generate physical penetration tests to execute. However due to unanticipated difficulties in the setup of the experiment, we were not able to draw conclusions on the usability of Portunes in generating attack scenarios.

The students enjoyed the assignment and their awareness of the physical and social aspects of security increased. During the course they learned the strengths and weaknesses of common physical, digital and social mechanisms used to secure sensitive information. Such experience is essential for the future security architects and chief security officers. Furthermore, the practical assignment increased the overall attendance in the course and improved the course grade.

However, the assignment is challenging to administer and draws ethical and legal implications. The students might feel uncomfortable to execute the attacks or the employees might not be treated with respect. The students might also abuse the new skills in illegal actions.

Surveys among students and employees indicate that the risks can be managed. Employees who participated in the exercise did not feel stressed nor considered the exercise harmful. Moreover, the arguments used in favor of digital penetration testing also apply for physical penetration testing. Therefore, we believe the benefits of the practical assignment outweigh the mentioned risks. This assignment can be used by other universities in introducing computer security to graduate students.

Chapter 9

Conclusions

In the introductory chapter, we introduced the term alignment of organizational security policies. In the same chapter we introduced three research questions that would improve the policy alignment and enforcement. In this chapter we summarize the contributions of this thesis, in relation to the research questions and show how our contributions can help solve practical problems in the industry. We also highlight future research directions both related to the work introduced in this thesis as well as the area of aligning security policies across the physical, digital and social domain in general.

Historically, policy alignment is either observed from management perspective, where managers align the security requirements with the business requirements, or from technical perspective, where security professionals align policies in a specific software, computer or computer network. Recently, policy alignment is being seen more as a tool that allows the security professionals to refine security requirements into technical security specification.

Policy alignment can be horizontal, when policies at the same level of abstraction are aligned, or vertical, when more abstract policies are refined into low-level policies on objects and data. During a horizontal alignment, the main goal is to make the policies jointly *exhaustive* and mutually *consistent*, which means that every behavior is either allowed or forbidden, and there is no behavior that is both

allowed and forbidden. During a vertical alignment, the main goal is to ensure that low-level policies are *complete* with respect to the high-level policies, which means that every behavior that is allowed by the high-level policies is allowed by the low-level policies, and that every behavior that is forbidden by the high-level policies is also forbidden by the low level policies.

The main goal of policy alignment research is to provide enforcement of high-level policies into security mechanisms, taking into consideration the physical, digital and social aspects of security. To achieve this goal, this research domain needs to provide (1) a consistent and exhaustive horizontal alignment of high-level security policies, (2) complete vertical alignment between high-level and low-level security policies and (3) complete enforcement of the low-level security policies into security mechanisms.

The first focus in this thesis is on helping organizations to have complete vertical alignment by providing formal methods for proving completeness of vertical alignment, and is addressed in Chapter 2, 3, 4 and 5. We show how the results from the first part of the thesis can be applied in mitigating the insider threat. Insiders have intimate knowledge of the policies of the organization, and may use them to their advantage to achieve a malicious goal. As a running example, we used the road apple attack, where an insider deceives an employee to plug an infected dongle into a server residing in a restricted area, enabling the transfer of sensitive information to a remote server.

After the policies are horizontally and vertically aligned by the HR and IT and physical security departments, they are enforced into security mechanisms. Security mechanisms cannot guarantee 100% enforcement of the policies because of their nature (locks may break, people may reveal confidential information). Therefore testing the security mechanisms by taking an adversarial role plays an important role in assurance that the deployed mechanisms are sufficient and operational.

The second focus in this thesis is on helping organizations have complete enforcement of the security policies by providing methodologies for testing and improving the effectiveness of security mechanisms in organizations, and is addressed in Chapter 6, 7 and 8. We show how the results from the first part of the thesis can be applied in mitigating the laptop theft threat. Laptops are mobile devices that are easy to cloak and steal, and may contain whole databases of sensitive information. A loss of a single laptop can cost an organization loss of productivity, restitution cost to clients and loss of intellectual property. As a working example, we use the scenario of stealing a laptop from an employee in a restricted area within the premises of the organization.

9.1 Scientific contributions

In the introductory chapter we formulated the following research question:

> **Main research question:** How can we align and enforce security policies spanning the physical, digital and social domain?

We refined the main research question into three research question that could help in aligning and enforcing security policies spanning the physical, digital and social domain. Below we summarize the contributions of this thesis for each of the research questions from practical and theoretical perspective.

Research question 1: *How can we represent the policies from the three domains in one formal framework?*

- We specified the three domains in one formal framework, Portunes. The framework is able to express: 1) physical properties of elements, 2) mobility of objects and data, and 3) trust and delegation between people. The framework is also able to represent low-level policies on objects, locations and data, such as identity, credential and location based access control (Chapter 3).
- We specified high-level policies formally. We defined a modal logic, designed to specify a set of desired and undesired behaviors and states of Portunes models. These behaviors and states are represented as properties of Portunes models and can also be used to describe adversarial goals (Chapter 5).

Research question 2: *How can we efficiently discover all cross-domain threats caused by policy misalignment?*

- We provided a set of algorithms that from a Portunes model generate a behavior that is allowed by the low-level policies but forbidden by a high-level policy (Chapter 4). The algorithms can be used to test the completeness of the vertical alignment across the physical, digital and social domain.

Research question 3: *How can we test and improve the enforcement of the low-level policies?*

- We developed two methodologies for testing the enforcement of the low-level policies through physical penetration tests (Chapter 6).
- We analyzed the logs from actual thefts and 32 penetration tests to determine the effectiveness of most commonly used security mechanisms in protecting against laptop theft (Chapter 7).
- We designed a practical assignment for graduate students in introducing the basic concepts in information security using live penetration tests (Chapter 8).

9.2 Practical contributions

Besides the scientific contribution, the results from the research can help addressing practical issues in the industry. In the introduction of this thesis we provided a motivating example where the management faced two issues in their information security program:

Problem 1: *How can the management be sure that the total set of low-level policies produced by the physical, IT and HR departments matches their high-level policy?*

Problem 2: *How can the three departments be sure that the security mechanisms in place follow the design specifications of the low-level policies?*

We approached the first issue using formal methods by providing a formal model and a logic and were able to define low-level policies, high-level policies and behaviors in a single formalism. From a practical perspective, we implemented the algorithms and partially the logic in a single proof of concept tool, which is freely available. Now the security departments can build a model that will represent the environment of interest, input the current low-level and high-level security policies and generate and simulate behaviors that are allowed by the low-level policies but forbidden by the high-level policies. The tool can be used at various cycles of development, for example, before system deployment to analyze what-if scenarios and after high-level policy modification, to check whether the low-level policies are still complete with respect to the high-level policies. These contributions can help the management to obtain formal assurance that the low-level policies defined by the physical, IT and HR departments are properly refined from the high-level policies and are complete with respect to them.

We addressed the second issue from the motivating example by providing an extensive overview of methods used to protect assets in an organization from theft, and testing methodologies to assure the security departments that the low-level policies are properly implemented. The security departments can now run physical penetration tests using social engineering with methodologies that have been scrutinized by the scientific research community and take into consideration the ethical implications of the tests. The results we obtained from orchestration 32 penetration tests provide valuable first hand information on the advantages and limitations of the proposed methodologies, as well as empirical information on the effectiveness of the most commonly used security mechanisms in open organizations.

9.3 Future work

The results in this thesis open several possible research directions.

- The constructs used to describe the three security domains in Portunes are carefully chosen with respect to their expressiveness and the complexity they add in the automatic generation of the behaviors. One can envision extending the Portunes framework with constructs such as negotiation between people, behavior templates (sequence of action templates that represent a generic behavior) or logging mechanisms, to increase the level of detail at which behaviors can be presented.

- In this thesis we presented only one analysis of Portunes models: automatic generation of behaviors. With small modifications, the Portunes framework is suitable for other types of analysis:

  Quantitative analysis. Using Portunes we can get a number of scenarios that lead to the violation of a high-level policy. The number of scenarios can be related to how well this policy is refined. For example, there is difference whether 1000 behaviors violate the policy rather than only 2. However, not all behaviors are equally likely. With addition of probabilities to the model, by giving to each action the likelihood of occurrence, we can provide quantitative analysis of Portunes models. Thus, the chance that someone forgets to lock a door or would be susceptible to social engineering can be encoded within the model and used for ranking the produced behaviors.

  Logging. Portunes can describe only preventive security mechanisms, such as firewalls, encryption, passwords and physical locks. By adding logging constructs on the policies, the Portunes framework can be extended

with detective security mechanisms, such as cameras, intrusion detection sensors, guards, infrared sensors etc. The analysis can then be extended to search for behaviors that are not detectable with the current positioning of the detective security mechanisms, but still violate the high-level policy. Another usage of logging can be to provide a minimal set of low-level policies that log the actions the guard, so no malicious behavior can occur undetected.

- The penetration testing methodologies proposed in this thesis were used to orchestrate 32 penetration tests. Throughout the execution of the tests we identified two issues that we consider to require further research.

  Perceived importance of the asset. During the penetration tests we noticed that some of the employees might have guarded the provided laptops much less than the laptops they work on. In other cases, the situation was reversed. Because the employees were entrusted with a new laptop, they seemed to guard these laptops much more than their own. It would be interesting to see how the perceived importance of the asset affects the behavior of the custodians. In an experiment, the custodians can be separated in two groups. One of the groups can be informed that the laptop contains information critical for the organization. Through another round of penetration tests we could see the difference of behavior between the groups.

  Safety as a requirement. The penetration tests are usually performed in office buildings. However, when they need to be performed in potentially hazardous environments, such as chemical, biological or nuclear laboratories safety becomes an important requirement for the methodology. It is interesting to investigate the aspect of *safety* of both the employees and the testers and include it into the penetration testing methodologies.

- During the analysis of the effectiveness of security mechanisms we used the logs from thefts from two universities and the results from the penetration tests. A more general picture would be obtained if we complement these results with logs from thefts from other institutions, as well as results of penetration tests performed at different premises.

9.4 Application of the results to other research areas

The results of this thesis can assist in multiple areas of information security:

Penetration testing. Until recently, the focus of penetration tests was finding vulnerabilities in computer networks. With the realization that the employees are the weakest link in security, the number of penetration tests increases where social engineering is used as a tool. The thesis provides a methodology the penetration testers can use to execute penetration tests. The Portunes tool can assist in the penetration test by providing the testers an automatically generated attack scenarios.

Risk assessment. Risk assessment usually takes as input attack scenarios where an asset or process is interrupted, stolen or exposed, and calculates the probability of these scenarios happening and the impact on the organization if such scenario occurs. The quality of the risk assessment depends directly on the quality and quantity of the attack scenarios used in the analysis. The Portunes tool can help presenting and generating these scenarios.

Security awareness training. Educating the future security professionals and the employees is becoming a standardized process in universities and organizations. The assignment provided in the thesis allows students to learn first-handedly the weaknesses of the most commonly used security mechanisms. The Portunes tool also provides step by step simulation of attack scenarios. These scenarios can be used as part of educating employees during security awareness trainings.

Appendix A

Comparison of related models

In this appendix we analyze in greater detail the Ambient Calculus, the model of Scott and the model of Dragovic. We analyze the models using the device tampering, coldboot attack and the road apple attack, which were presented in the case study in Chapter 2. Using the attacks from the case study, we point and discuss the shortcoming of the presented models.

Ambient calculus

For the semantics of the ambient calculus refer to the original paper [23]. The results below have been validated on the Basic Ambient Factory tool. Here we show why the ambient calculus can not present tampering with a device, and an approach to model the coldboot attack and the road apple attack.

Tampering:

1. Without any information about the tamper resistance of the device, we can present the data leaving the device as.

   device[data[out device]]

2. We can present the increased tamper resistance by using *N* layers of nested ambients.

   device[pLayer1…[pLayerN[data[out pLayerN… out pLayer1.out device]]]…]

The capability of the data leaving the device resides with the data, not with the device. Through this presentation, for every change on the stack of layers, the capability of the data needs to be changed dynamically.

A simple example is:

device[pLayer1[pLayer2 [data[out pLayer2.out pLayer1.out device]]]]

If pLayer1 is removed from the device (the adversary circumvents one layer of defense or reduces the strength of the resistance of the device), the data ambient will never be able to get out of the device because out pLayer1 capability will never be executed.
(NO)

Coldboot attack:

To present the attack, we model 2 laptops, the data of interest, the RAM which is being moved, and the destination hard drive. In steps (1) and (2), the ram moves from laptop1 to laptop2. In steps (3) and (4) the data moves from the ram to the hdd of laptop2. Every current action is presented in bold font.

laptop1[ram[**out laptop1**. in laptop2. data[out ram.in hdd]]] | laptop2[hdd[]] (1)
→laptop1[] | laptop2[hdd[]] | ram[**in laptop2**. data[out ram.in hdd]] (2)
→laptop1[] | laptop2[ram[data[**out ram**.in hdd]] | hdd[]] (3)
→laptop1[] | laptop2[ram[hdd[] | data[**in hdd**]]] (4)
→laptop1[] | laptop2[ram[hdd[data[]]]] (5)

(YES)

Road apple 1:

adversary[in cafeteria.usb[out adversary. in employee. in laptop. rootkit[]] | out cafeteria] | employee[in cafeteria | laptop[open usb] | out cafeteria] | cafeteria[]

Trace when the adversary succeeds in the attack:

adversary[**in cafeteria**.usb[out adversary. in employee. in laptop. rootkit[]] | out cafeteria] (1)
| employee[in cafeteria | laptop[open usb] | out cafeteria] |cafeteria[]
→ employee[in cafeteria | laptop[open usb] | out cafeteria] (2)
| cafeteria[adversary[usb[**out adversary**. in employee. in laptop. rootkit[]] | out cafeteria]]
→ employee[in cafeteria | laptop[open usb] | out cafeteria] (3)
| cafeteria[adversary**[out cafeteria**] | usb[in employee. in laptop. rootkit[]]]
→ employee[**in cafeteria** | laptop[open usb] | out cafeteria] (4)
| cafeteria[usb[in employee. in laptop. rootkit[]]] | adversary[]
→ cafeteria[employee[laptop[open usb] | out cafeteria] | usb[**in employee**. in laptop. rootkit[]]] | adversary[] (5)
→ cafeteria[employee[usb[**in laptop**. rootkit[]] | laptop[open usb] | out cafeteria]] | adversary[] (6)
→ cafeteria[employee[laptop[**open usb** | usb[rootkit[]]] | out cafeteria]] | adversary[] (7)
→ cafeteria[employee[laptop[rootkit[]] | **out cafeteria**]] | adversary[] (8)
→ cafeteria[] | employee[laptop[rootkit[]]] |adversary[] (9)

Besides being able to present the coldboot attack, the calculus is able to present concurrent actions. In this scenario, there are two other trance which we just mention. The first branch generating new trace can occur in (2), where the adversary enters the cafeteria but does not leave the dongle (out cafeteria capability in adversary executes before out adversary in usb). The second branching can occur in (5), where the employee enters the cafeteria but does not pick up the dongle (out cafeteria capability in employee executes before in employee in usb).
(YES)

Road apple 2:

In this scenario, the adversary sends a message to the employee to plug in the dongle. After approval, the dongle goes from the adversary to the laptop of the user and infects the laptop.

One trace is:

adversary[m1[**out adversary**.in employee] |open m2.usb[out adversary. in employee.in laptop.rootkit[]]] (1)
| employee[open m1.m2[out employee.in adversary] | laptop[open usb]]
→ adversary[open m2.usb[out adversary. in employee.in laptop.rootkit[]]] (2)
| employee[open m1.m2[out employee.in adversary]| laptop[open usb]] | m1[**in employee**]
→ adversary[open m2.usb[out adversary. in employee.in laptop.rootkit[]]] (3)
| employee[m1[] | **open m1**.m2[out employee.in adversary] | laptop[open usb]]
→ adversary[open m2.usb[out adversary. in employee.in laptop.rootkit[]]] (4)
| employee[m2[**out employee**.in adversary] | laptop[open usb]]
→ adversary[open m2.usb[out adversary. in employee.in laptop.rootkit[]]] (5)
| employee[laptop[open usb]] | m2[**in adversary**]
→ adversary[m2[] | **open m2**.usb[out adversary. in employee.in laptop.rootkit[]]] (6)
| employee[laptop[open usb]]
→ adversary[usb[**out adversary**. in employee.in laptop.rootkit[]]] (7)
| employee[laptop[open usb]]
→ adversary[] | employee[laptop[open usb]] | usb[**in employee**.in laptop.rootkit[]] (8)
→ adversary[] | employee[usb[**in laptop**.rootkit[]] | laptop[open usb]] (9)
→ adversary[] | employee[laptop[**open usb** | usb[rootkit[]]]] (10)
→ adversary[] | employee[laptop[rootkit[]]] (11)
(YES)

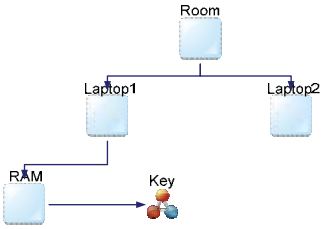
Model of Scott

The model of Scott is capable of presenting the coldboot attack and the road apple attack with indirect interaction between the adversary and the employee, but can not present device tampering and the road apple attack with direct interaction between the adversary and the employee.

Tampering:

Similarly to the ambient calculus, we can assume multiple layers of protection over the data. But, the defined command over data (*emit/receive)* uses "teleporting" approach and ignores the layers between the source path and destination path. Because of the teleporting, any obstacles placed between the data and the adversary can be ignored, and thus, no tamper resistance of a device can be presented.
(NO)

Coldboot attack:



Initial:

room[laptop1[RAM[key[0]]] | laptop2[0]]

Transformations:

$$\frac{\dfrac{laptop1 \xrightarrow{RAM} room; room \xrightarrow{RAM} laptop2}{laptop1 \xrightarrow{RAM} laptop2}}{\eta[laptop1] \xrightarrow{RAM} \eta[laptop2]}$$

Result:

room[laptop1[0] | laptop2[RAM[key[0]]]]

The transformation above shows the RAM leaving laptop1 and going to laptop2, through the room entity.

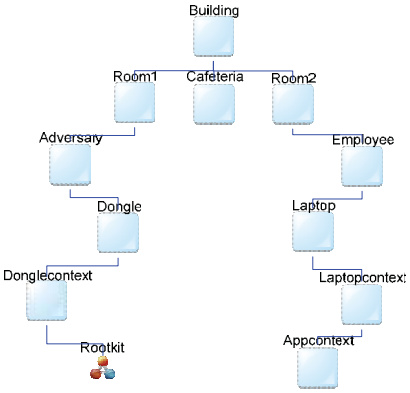
$$laptop1 \xrightarrow{RAM} room; room \xrightarrow{RAM} laptop2$$

This derivation can be read as: the RAM moves from the position of laptop1 to the position of laptop2. Here we point out the teleportation effect, where all entities in between which might restrict the movement (in this case room) are ignored in the formula.

$$\eta[laptop1] \xrightarrow{RAM} \eta[laptop2]$$

(YES)

Road apple 1:



Initial:

building[room1[adversary[dongle[donglecontext[rootkit[0]]]]] | cafeteria[0] | room2[employee[laptop[laptopcontext[appcontext[0]]]]]]

Transformations:

$$\frac{room1 \xrightarrow{adversary} building; building \xrightarrow{adversary} cafeteria}{\dfrac{room1 \xrightarrow{adversary} cafeteria}{\eta[room1] \xrightarrow{adversary} \eta[cafeteria]}}$$

$$\frac{adversary \xrightarrow{dongle} cafeteria}{\eta[adversary] \xrightarrow{adversary} \eta[cafeteria]}$$

$$\frac{cafeteria \xrightarrow{adversary} building; building \xrightarrow{adversary} room1}{\dfrac{cafeteria \xrightarrow{adversary} room1}{\eta[cafeteria] \xrightarrow{adversary} \eta[room1]}}$$

$$\frac{room2 \xrightarrow{employee} building; building \xrightarrow{employee} cafeteria}{\dfrac{room2 \xrightarrow{employee} cafeteria}{\eta[room2] \xrightarrow{employee} \eta[cafeteria]}}$$

$$\frac{cafeteria \xrightarrow{dongle} building; building \xrightarrow{dongle} laptop}{\dfrac{cafeteria \xrightarrow{dongle} laptop}{\eta[cafeteria] \xrightarrow{dongle} \eta[laptop]}}$$

$$\frac{donglecontext \xrightarrow{emit(rootkit)} laptopcontext; laptopcontext \xrightarrow{recieve(rootkit)} appcontext}{\dfrac{donglecontext \xrightarrow{rootkit} appcontext}{\eta[donglecontext] \xrightarrow{rootkit} \eta[appcontext]}}$$

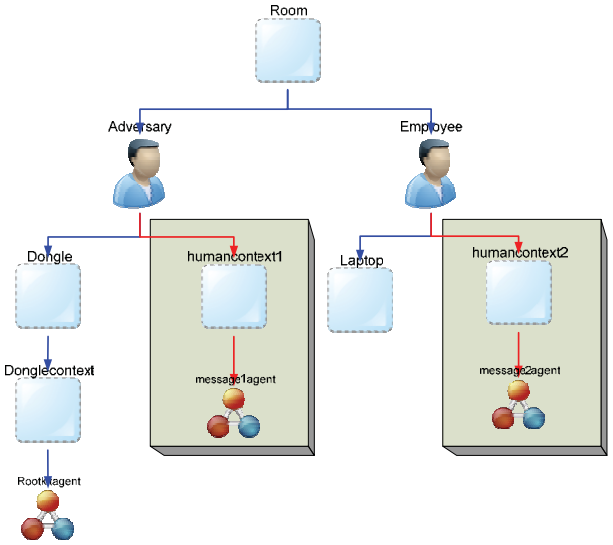
Result:

building[room1[adversary[0]] | cafeteria[0] |
room2[employee[laptop[dongle[donglecontext[0]] |
laptopcontext[appcontext[rootkit[0]]]]]]

(YES)

Road apple 2:

Although the model can present the physical/digital transitions, the model can not present the interaction employee-adversary. A similar approach as presented in the ambient calculus example will require generating new rules besides *pick up/leave down*. We show an approach that might be used to present the attack to a greater extent. First, we add context to people. Second, we use special agents which will present messages and reside in the context in people. The world for the road apple attack with direct interaction between the employee and the adversary would be:



room[adversary[usb[usbcontext[rootkit[0]]] |humancontext1[message1agent[0]] |
employee[laptop[laptopcontext[0]] | humancontext2[message2agent[0]]]

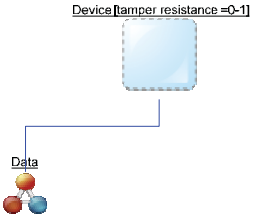
Next, we need to add rules for exchanging messages between people, such as:

$$\frac{humancontext1 \xrightarrow{emit(message1agent)} employee; employee \xrightarrow{recieve(message1agent)} humancontext2}{\dfrac{humancontext1 \xrightarrow{message1agent} humancontext2}{\eta[human1context] \xrightarrow{message1agent} \eta[humancontext2]}}$$

(NO)

Model of Dragovic

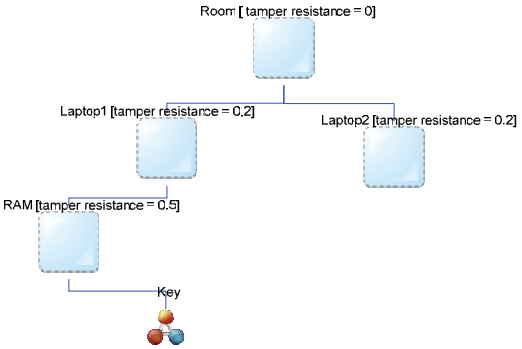
Tampering:



leave(Data)

The model of Dragovic, besides providing attributes that define the tamper resistance of a device, provides information for the sensitivity of the data inside the device, the level of exposure of this data, as well as additive protection of multiple types of containers. The additional information is not provided because it is out of the scope of the paper. For further information refer to [35,36].
(YES)

Coldboot attack:



Initial:

tr = tamper resistance
room:tr=0;[laptop1:tr=0.2;[RAM:tr=0.5;[key:tr=0[0]]] | laptop2:tr=0.2;[0]]

Transformations:

leave(Room/Laptop1/RAM)
update(RAM, tamper resistance = 0.8)
enter(RAM, Room/Laptop2)
update(RAM, tamper resistance = 0.2)

Result:

room:tr=0;[laptop1:tr=0.2; | laptop2:tr=0.2;[RAM:tr=0.5;[key:tr=0;[0]]]]

Besides providing information about the movement of the RAM with the data, the model provides information for the degradation of the data inside the RAM when the RAM is removed from the laptop.
(YES)

Road apple 1:

Initial:

For brevity, we assume the containers to have no attributes.

building [room1[adversary[dongle [rootkit[0]]]] | cafeteria[0] | room2[employee[laptop[0]]]]

Transformations:

migrate (building/room1/adversary, building/cafeteria)
migrate (building/cafeteria/adversary/dongle, building/cafeteria)
migrate (building/cafeteria/adversary, building/room1)
migrate (building/room2/employee, building/cafeteria)
migrate (building/cafeteria/dongle, building/cafeteria/employee/laptop)
migrate (building/cafeteria/employee/laptop/dongle/rootkit, building/cafeteria/employee/laptop)

Result:

building[room1[adversary[0]] | cafeteria[0] | room2[employee[laptop[dongle [rootkit[0]]]]]]

(YES)

Road apple 2:

Initial:

room[adversary[dongle[rootkit[0]]] | employee[laptop[0]]]

Transformations:

migrate (Room/Adversary/Dongle, Room/Employee/Laptop)
migrate (Room/Employee/Laptop/Dongle/Rootkit, Room/Employee/Laptop)

Result:

room[adversary[0] | employee[laptop[dongle[0] |rootkit[0]]]]

This model also does not present interaction between people. A workaround will be similar as in modeling the road apple example with Scott's model.

(NO)

Appendix B

Rules of engagement

Rules of engagement

I, _____ (name of student) agree to perform penetration tests for _____ (name of researcher)

I understand that the participation of is completely voluntary. At any time, I can stop my participation.

I fully oblige to the following rules of engagement:

1. I will only execute attacks that are pre-approved by the researcher and only to an assigned target.
2. I am not allowed to cause any physical damage to university property, except for Kensington locks.
3. I am not allowed to physically harm any person as part of the test.
4. I will video or audio record all my activities while interacting with people during the penetration test as a proof that no excessive stress or panic is caused to anyone.
5. If I am caught by a guard of a police officer, I will not show any physical resistance.

Signature of researcher:_____ Date: _____

Signature of student: _____ Date: _____

Appendix C

Informed consent

Informed consent

I, _____ (name of employee) agree to participate in the study performed by _____ (name of the research group).

I understand that the participation of the study is completely voluntary. At any time, I can stop my participation and obtain the data gathered from the study, have it removed from the database or have it destroyed.

The following points have been explained to me:

1. The goal of this study is to gather information of laptop usage. Participation in this study will yield more information concerning the habits people have in using mobile devices.
2. I shall be asked to work for 5 min every day on a laptop for one month. The laptop can be monitored and/or recorded using a keylogger and a web-camera. At the end of the study, the researcher will explain the purpose of the study.
3. No stress or discomfort should result from participation in this study.
4. The data obtains from this study will be processed anonymously and can therefore not be made public in an individually identifiable manner.
5. The researcher will answer all further questions on this study, now or during the cause of the study.

Signature of researcher:_____ Date:_____

Signature of employee:_____ Date:_____

Appendix D

Sample report of a laptop theft

Report 2009/0670, 6-07-2009, 1/1

Date of the incident:	Wednesday, 01 July 2009
Time of the incident:	14:00h
Place:	4th floor, main building
Person reporting:	Alice

Description:

On the above date, Mr."Bob" came to the security post and informed me that his laptop and the laptop of his colleague were stolen from room Yellow. Mr."Bob" informed me that at 13:45 they both left the room Yellow without anyone inside. After returning to the room at 14:00, both laptops were gone. Both laptops were locked to a table with a Kensington lock. The locks were cut. Room Yellow is accessible only by using an access pass.

Actions taken:

I called to the Facility management and asked if someone used an access pass to enter the room Yellow between 13:45 and 14:00. I was informed that only the pass of Mr."Dave" was used. However, the pass of Mr."Dave" was not used for entering or leaving the building, and we cannot reach him. We searched the camera images from this period, but we could not find any useful information.

Appendix E

Get out of jail card

Get out of jail card

The student _____ is performing a penetration test in the period between xxth and xxth of September. The test is approved by xxxxxxx (researcher name) and the security management of xxxxxxx. In case of being caught while executing the penetration test, please contact xxxxxxx (researcher) tel: xxxxxxx or xxxxxxx (security manager) tel: xxxxxxx at any time of the day.

Signature of researcher: _____

Appendix F

Note left from the testers

Dear Sir/Madam

Your laptop is taken by a team of physical penetration testers. The laptop is not stolen nor damaged in any way.

These penetration tests are a joint effort between the security management of University of Twente and the Distributed and Embedded Security Group (DIES). The goal of the tests is to examine the effectiveness of the physical security in the campus of the University. The tests will help us reduce the number of the increasing thefts of laptops, computer equipment and lab equipment in the campus.

All the data stored in the laptop will be treated as confidential and will be available to you upon request. After the all tests are finished, we will invite you for a debriefing session where we will describe you the methodology we used and the results we obtained. The same results (anonymized) will be used as case studies in the training of the security personnel in the University. If you have any questions, please contact us by mail or by phone at any time.

DIES Group:

Dimkov Trajce	**Wolter Pieters**	**Pieter Hartel**
t.dimkov@utwente.nl	w.pieters@utwente.nl	p.hartel@utwente.nl
Zilverling 3006	Zilverling 3023	Zilverling 3001
Tel: xxxxxxxxxxx	Tel: xxxxxxxxxxx	Tel: xxxxxxxxxxx

Security management:
xxxxxxxxxxxxxxxx
xxxx@fb.utwente.nl
Tel: xxxxxxxxxx

Appendix G

Successful and unsuccessful attempts during the penetration tests

In this appendix we present a summary of all penetration tests performed by the students, both the successful and unsuccessful.

Table G.1 shows the actions that contributed to the successful penetration tests, ordered according to the frequency they were used in the successful attempts. For example, there were 27 successful attempts, where the testers entered the building during working hours, when the outside doors are open to the public. Thus, from the frequency of the actions, the easiest way for the tester to obtain the laptop is to go into the building during working hours, while the custodian is in his office, and provide a reason why she needs to unlock the laptop and give it to the tester.

Similarly, in Table G.2 we summarize the mechanisms that contributed to the failed attempts. For example, because of the presence of the employees, the testers were 7 times not able to enter the office, while they were not able to steal the laptop 3 times because of the Kensington lock. Some of the reasons why the attempts

Enter building	**Enter office**	**Unlock laptop**	**Leave**
working hours (27)	someone inside (20)	custodian unlocks (19)	working hours (26)
obtained a card (2)	key from employee (5)	bolt cutter (5)	using access card (3)
asked an employee (1)	room was unlocked (4)	laptop was unlocked (3)	emergency door (1)
during social event (1)	cleaning lady (1)	detach from desk (2)	during social event (1)
	secretary opens (1)	found key in office (1)	
		took the desk with the laptop (1)	

Table G.1: Actions used in successful attempts

Enter building	**Enter office**	**Unlock laptop**	**Leave**
locked door (1)	employee (7)	custodian (6)	
	custodian (2)	Kensington lock (2)	
	secretary (2)	employee (3)	
	door lock (2)	lab officer (2)	
	guards (1)	laptop not found (3)	

Table G.2: Reasons why attempts failed

failed were purely coincidental and if repeated could have most likely succeeded. For example, on few occasions, the custodians despite the rules took the laptop home. Thus when the testers approached them, although they were willing to hand over the laptop, they did not have it in the office. Similarly, on some occasions the mere presence of employees in the office was sufficient to abort the attack, because it was not matching the scenario the testers were allowed to execute.

There were 31 successful and 31 unsuccessful attempts in total. We distinguished six elements from each attempt: (1) how the testers entered the building, (2) how did the testers entered the office where the laptop resides, (3) how the testers circumvented the Kensington lock, (4) the role the testers took as a disguise and (5) a qualitative analysis on how high the security awareness of the affected employees was.

During some attempts, the testers had to circumvent the CCTV surveillance or record the attempt. In those attempts, we also include this information. During the unsuccessful attempts, we include the successful actions until the attempt failed, and the reason why the attempt failed (in bold font).

TEAM	IRA	Cornelissen	Donker
Enter building	working hours	working hours	working hours
Enter office	custodian inside	custodian inside	custodian inside
Unlock laptop	custodian unlocks	custodian unlocks	custodian unlocks
Leave	working hours	working hours	working hours
Recording	hidden camera		
Role of pentester	coordinator assistant	game players/ students	ICT Desk
CCTV	scouting and hiding	hoods	scouting and hiding
Resistance	MEDIUM. The custodian tried to call the coordinator but failed.	HIGH. Testers faked an email granting them permission on the laptop. The testers promised to return the laptop in a few hours.	HIGH: The custodian asked to talk to the coordinator for permission. After talking to a fake coordinator through a tester phone, the custodian accepted to give the laptop.

TEAM	Byte Philosophy	Laverman	Veen
Enter building	working hours	working hours	working hours
Enter office	secretary	custodian inside	custodian inside
Unlock laptop	custodian unlocks	custodian unlocks	custodian unlocks
Leave	working hours	working hours	working hours
Recording			camera
Role of pentester	coordinator assistants	coordinator assistants	ICT Desk
CCTV	scouting and hiding	scouting and hiding	Hoods
Resistance	LOW. A tester sent a fake email. The custodian packs the laptop and leaves it in the office. The secretary unlocked the door, and gave the laptop to the tester.	LOW. The custodian did not get suspicious and handed in the laptop.	LOW: The custodian gave the laptop easily after being told by the testers he will get the laptop back in a few hours.

TEAM	Nicked	X	Damhuis
Enter building	working hours	social engineered a card	waits after working hours
Enter office	custodian inside	cleaning lady	key from janitor
Unlock laptop	custodian unlocks	bolt cutter	bolt cutter
Leave	working hours	using the same card	through emergency door
Recording	open camera	web-camera	
Role of pentester	coordinator assistant	PhD researcher	students
CCTV	used newspapers	scouting and hiding	scout and hide
Resistance	LOW. The custodian believed in the fake email and handed in the laptop.	LOW: The employee easily gave the night card to the tester. The cleaning lady easily believed the tester is a PhD researcher and let him inside the office.	LOW: The janitor gave the testers a key from the room after working hours.

TEAM	He	Hafidz	Team X	A
Enter building	during social event when doors are open	working hours	working hours	working hours
Enter office	custodian inside	custodian inside	room was unlocked	key from janitor
Unlock laptop	custodian unlocks	custodian unlocks	laptop was unlocked	laptop not locked
Leave	same door	working hours	working hours	working hours
Recording			camera	web camera
Role of pentester	students	coordinator assistants	delivery man	student
CCTV	did not hide	scout and hide		scout and hide
Resistance	VERY HIGH: The custodian got a (fake) identification, but asked them to sign a document they will return the laptop.	LOW: After getting the fake email, the custodian handed in the laptop.	LOW: The testers phoned the only present employee in the office, stating that there is a package for him. The employee left the room unlocked. The laptop was also unlocked.	MEDIUM: The janitor opened the door for the testers after being shown a fake mail. However, he escorted them during the theft.

TEAM	Awesome	093	Pasta
Enter building	working hours	working hours	night pass from employee
Enter office	key from janitor	key from janitor	use master key
Unlock laptop	laptop not locked	found key in the desk	not locked properly
Leave	working hours	working hours	night pass from employee
Recording	web camera	web camera	camera
Role of pentester	student	student	students preparing a party
CCTV	scout and hide	scout and hide	
Resistance	MEDIUM: The janitor opened the door for the students after being shown a fake mail. However, he escorted them during the theft.	MEDIUM: The janitor opened the door for the students after being shown a fake mail. He did not escort them to the room.	LOW: The security officer/janitor gave the students the master key of the building.

TEAM	Clerro	Flickr	Team 8
Enter building	working hours	working hours	working hours
Enter office	employee did not lock	custodian inside	custodian inside
Unlock laptop	bolt cutter	custodian unlocks	custodian unlocks
Leave	working hours	working hours	working hours
Recording	camera	camera	camera
Role of pentester	ICT Desk	ICT Desk	ICT Desk
Resistance	MEDIUM: The employee was convinced to turn in the laptop, but could not find the key from the Kensington lock. While he left the room to search for the key with one of the testers, two other testers entered the room and cut the Kensington lock.	LOW: The secretary introduced the tester to the custodian, thus the custodian did not doubt anything. Secretary did not ask for any identification, nor checked the fake phone number and email.	HIGH: The employee contacted the helpdesk. Although they informed him they are not aware of virus spread, the custodian still gave the laptop to the testers. The custodian required document signed by the testers that they take the laptop.

TEAM	MCN Team	Outcasts	Clerro
Enter building	working hours	working hours	working hours
Enter office	employee inside	custodian inside	custodian left unlocked
Unlock laptop	detach from desk	custodian unlocks	bolt cutter
Leave	working hours	working hours	working hours
Recording	camera	camera	Camera
Role of pentester	ICT Desk	ICT Desk	/
Resistance	LOW: An employee gave the laptop without checking the identity of the testers.	LOW: The custodian gave the laptop without any request for identification.	N/A: There was no social engineering used in this scenario.

TEAM	Team 8	Team X	Big Brothers
Enter building	working hours	working hours	working hours
Enter office	custodian inside	custodian inside	custodian inside
Unlock laptop	custodian unlocks	custodian unlocks	custodian unlocks
Leave	working hours	working hours	working hours
Recording	audio	audio	camera
Role of pentester	ICT Desk	ICT Desk	ICT Desk
Resistance	MEDIUM: the custodian asked for credentials, but was not insisting.	LOW: the custodian did not challenge the employees. They presented him only with a report stating there is a virus in the laptop.	LOW: The team sent an email claiming the laptop has a virus. The custodian gave the laptop without asking any question.

TEAM	Pasta	Flickr	The Insiders
Enter building	night pass from employee	working hours	working hours
Enter office	room was unlocked	custodian inside	custodian inside
Unlock laptop	take the desk with the laptop	custodian unlocks	custodian unlocks
Leave	use a night pass	working hours	working hours
Recording	camera	camera	camera
Role of pentester	Student	ICT Desk	ICT Desk
Resistance	N/A: There was social engineering used only to get the night pass. The testers used the night pass to get to the laboratory. They took the whole desk, and brought it with 5 other students in Zilverling. They had also access to this building.	LOW: The secretary introduced the tester to the custodian, thus the custodian did not doubt anything. Secretary did not ask for any identification, nor checked the fake phone number and email.	HIGH: The custodian wanted a receipt. The laptop was kept in a locked closed, locked with a Kensington lock. Thus, she was the only person in the office that could access it.

TEAM	MCN Team	Big Brothers	The Insiders
Enter building	working hours	working hours	working hours
Enter office	custodian inside	custodian inside	custodian inside
Unlock laptop	custodian unlocks	custodian unlocks	bolt cutter
Leave	working hours	working hours	working hours
Recording	camera	Camera	camera
Role of pentester	ICT Desk	ICT Desk	ICT Desk
Resistance	LOW: The custodian did not ask any questions.	LOW: The team sent an email claiming the graphic chipset is faulty. They came 20min later to pick up the laptop. The custodian did not ask any questions.	LOW: The tester managed to cut the Kensington lock while there was another person in the office. The tester was not challenged on what he is doing.

TEAM	Cornelissen failed	X failed	X failed
Enter building	working hours	working hours	working hours
Enter office	custodian inside	custodian inside	**spotted guards and aborted**
Unlock laptop	**custodian forgets key**	**custodian declines**	
Leave	working hours	working hours	
Recording			
Role of pentester	game players	ICT Desk	
Resistance	LOW. However, sent email to coordinator asking about the game. Coordinator says he has never heard of it	MEDIUM: the custodian asks for an email to confirm that the laptop needs to be replaced.	

TEAM	X failed	Veen failed	Veen failed
Enter building	working hours	working hours	working hours
Enter office	employee inside	employee inside	custodian inside
Unlock laptop	**employee declines**	**employee agrees, but no laptop**	**custodian did not have the laptop**
Leave			
Recording		camera	camera
Role of pentester	master student	ICT Desk	ICT Desk
Resistance	MEDIUM: the employee opens the door for the tester, but is reluctant to search for key for the Kensington lock, and instead asks the tester to talk to the custodian	LOW: the employee is willing to give the laptop of the custodian. The custodian breaks the agreement, and takes the laptop with him when he leaves the office.	LOW: the custodian is willing to give the laptop, but leaves it to his girlfriend in Belgium.

TEAM	Damhuis failed	Damhuis failed	Damhuis failed
Enter building	working hours	working hours	working hours
Enter office	key from janitor	**secretary rejected**	**employee inside**
Unlock laptop	**employee enters while stealing the laptop**		
Leave			
Role of pentester	students	students	PhD researcher
Resistance	LOW: the janitor handed in the key without problem. When the employee enters the office and spots the tester, it did not challenge him.	HIGH: instead of giving the key, the secretary asked the custodian to come to work and talk to the students	

TEAM	A failed	A failed	093 failed
Enter building	working hours	working hours	working hours
Enter office	**locked**	**custodian inside**	**secretary**
Unlock laptop			
Leave			
Role of pentester	student	student	student
Resistance	LOW: An employee got a key from the secretary to help the tester, but for some reason the key did not work.	HIGH: The custodian got suspicious and did not allow any theft in his presence.	HIGH: The secretary asked them to talk to security for getting the key.

TEAM	Clerro failed	Flickr failed	The Insiders failed
Enter building	working hours	working hours	working hours
Enter office	**custodian inside**	**too many employees inside the room**	custodian inside
Unlock laptop			**custodian rejects**
Leave			
Role of pentester	delivery person		ICT Desk
Resistance	N/A: The testers called the custodian trying to tell her that there is a delivery waiting downstairs. The custodian did not pick up the phone.	N/A: After seeing that there are many people in the office, the testers left the area.	HIGH: the custodian rejected to unlock the laptop or give it to the tester. The custodian insisted the laptop should be fixed on the spot.

TEAM	MCN Team failed	MCN Team failed	Outcasts failed
Enter building	working hours	**cannot enter the building**	working hours
Enter office	left room unlocked		employee inside
Unlock laptop	**too short time to cut the lock**		**laptop not present**
Leave			
Role of pentester			ICT Desk
Resistance	N/A: The room was left empty only for a short time. The testers did not have enough time to cut the Kensington lock.	N/A: The testers tried to enter the building after working hours and social engineer the cleaning lady. The cleaning ladies are present only in the morning.	LOW: The employee was willing to give the laptop of the custodian, but they could not find it.

TEAM	Outcasts failed	Pasta failed	Pasta failed
Enter building	working hours	working hours	working hours
Enter office	custodian insider	custodian inside	**too many people inside**
Unlock laptop	**laptop not present**	**stopped by lab officer**	
Leave			
Role of pentester	ICT Desk	Students	students
Resistance	LOW: The custodian was willing to give the laptop, but she took it home and the laptop was not at work.	HIGH: The employees in the lab were suspicious on the testers. The man responsible for the security in the lab stopped the testers and asked them to leave the lab.	N/A: The room was occupied with many people, making it impossible for the testers to steal the laptop.

TEAM	Pasta failed	Pasta failed	Pasta failed
Enter building	working hours	with a pass	working hours
Enter office	**too many people inside**	unlocked office	**could not see laptop from window**
Unlock laptop		**could not find laptop**	
Leave			
Role of pentester	students	Students	students
Resistance	N/A: The room was occupied with many people, making it impossible for the testers to steal the laptop.	N/A: The room was empty, but the testers could not locate the laptop.	N/A: The testers could not locate the laptop.

TEAM	Pasta failed	Pasta failed	Pasta failed
Enter building	working hours	working hours	working hours
Enter office	unlocked office	**too many people**	custodian inside
Unlock laptop	**could not find laptop**		**stopped by lab officer**
Leave			
Role of pentester	students	Students	students
Resistance	N/A: The testers could not locate the laptop.	N/A: The room was occupied with many people, making it impossible for the testers to steal the laptop.	HIGH: The employees in the lab were suspicions. The person responsible for the security in the lab stopped the testers and asked them to leave the lab. This time he asked them to speak with the custodian.
TEAM	**Outcasts failed**	**Team X failed**	**MCN Team failed**
Enter building	working hours	working hours	working hours
Enter office	employee lets them in	**employee locks door behind**	**too many employees inside the room**
Unlock laptop	**cannot find the key**		
Leave			
Role of pentester	ICT Desk	delivery person	
Resistance	HIGH: After the failed attempt, the custodian contacted the coordinator. The testers did not try any other attempt.	HIGH: The testers called the employee and told him there is a package for him in the reception. When the employee left, he locked the door behind.	N/A: After seeing that there are many people in the office, the testers left the area.
TEAM	**Flickr failed**		
Enter building	working hours		
Enter office	secretary let him in		
Unlock laptop	**laptop was unlocked but no positive ID**		
Leave			
Role of pentester	ICT Desk		
Resistance	MEDIUM: The secretary let the testers in the office and let them run software from a USB drive. She also left them for a few minutes. However, the tester could not get a positive identification whether the laptop is the target and did not steal the laptop.		

Appendix H

Variables used in the quantitative analysis

N	Variable	Description	Encoding	
			Yes	No
1	SocialEng	An individual was social engineered	1	0
2	PhysicalTheft	A physical theft took place	1	0
3	WorkingHours	The students entered the building during working hours	1	0
4	SocialEngCard	The students entered the building using an access card	1	0
5	AskEmployee	The students entered the building by asking an employee	1	0
6	DuringSocialEvent	The students entered the building during a social event	1	0
7	SomeoneInside	The students entered the office while someone was inside	1	0
8	KeyFromEmployee	The students entered the office with a key from an employee	1	0
9	UnlockedRoom	The students entered an unlocked office	1	0
10	CleaningLady	The students entered the office with help of a cleaning lady	1	0
11	Secretary	The students entered the office with the help of a secretary	1	0
12	CustodianUnlocks	The custodian unlocks the Kensington lock	1	0
13	BoltCutter	The students circumvent the lock using a bolt cutter	1	0
14	NotLockedKL	The laptop was not locked with a Kensington lock	1	0
15	DetachFromDesk	The Kensington lock was detached from the desk	1	0
16	FindKeyInDesk	The students found the key from the Kensington lock	1	0
17	TakeDesk	The students circumvented the Kensington lock by taking the desk where the laptops is lock	1	0
18	WorkingHoursL	The students left the building during working hours	1	0
19	UsingAccessCardL	The students left the building using an access card	1	0
20	DuringSocialEventL	The students left the building during a social event	1	0
21	EmergencyDoor	The students left the building through the emergency exit	1	0
22	ICTEmployee	The students took the role of an ICT employee	1	0
23	Student	The students took the role as students	1	0
24	CoordinatorAss	The students took the role as coordinator assistants	1	0
25	PhDStudent	The students took the role as PhD Students	1	0
26	DeliveryPerson	The students took the role as delivery person	1	0
27	Custodian	The students approached the custodian	1	0
28	Employee	The students approached an employee	1	0
29	Janitor	The students approached the janitor	1	0
30	CleaningLadyApp	The students approached the cleaning lady	1	0

Figure H.1: Independent variables

Bibliography

Author's Publications

— Refereed Conferences —

[1] T. Dimkov, W. Pieters, and P. Hartel. Effectiveness of physical, social and digital mechanisms against laptop theft in open organizations. In *IEEE/ACM International Conference on Cyber, Physical and Social Computing*, pages 727–732. IEEE, 2010. (Subsumed by Chapter 7 of this thesis).

[2] T. Dimkov, W. Pieters, and P. Hartel. Portunes: representing attack scenarios spanning through the physical, digital and social domain. In *Proceedings of the 2010 joint conference on Automated reasoning for security protocol analysis and issues in the theory of security*, ARSPA-WITS'10, pages 112–129, Berlin, Heidelberg, 2010. Springer. (Subsumed by Chapter 3 of this thesis).

[3] T. Dimkov, W. Pieters, and P. Hartel. Training students to steal: a practical assignment in computer security education. In *Proceedings of the 42nd ACM Technical Symposium on Computer science education*, SIGCSE '11, pages 21–26, New York, NY, USA, 2011. ACM. (Subsumed by Chapter 8 of this thesis).

[4] T. Dimkov, Q. Tang, and P. H. Hartel. On the inability of existing security models to cope with data mobility in dynamic organizations. In *Proceedings of the Workshop on Modeling Security*, pages 1–13. CEUR Workshop Proceedings, 2008. (Subsumed by Chapter 2 of this thesis).

[5] T. Dimkov, A. van Cleeff, W. Pieters, and P. Hartel. Two methodologies for physical penetration testing using social engineering. In *Proceedings of*

the 26th Annual Computer Security Applications Conference, ACSAC '10, pages 399–408, New York, NY, USA, 2010. ACM. (Subsumed by Chapter 6 of this thesis).

[6] A. van Cleeff, T. Dimkov, W. Pieters, and R. Wieringa. Realizing security requirements with physical properties: A case study on paper voting. In *Prooceedings of the International Conference in IT Convergence and Security*, Suwon, South Korea, 2012. Springer. In print.

— Book Chapters —

[7] C.W. Probst, M.A. Sasse, W. Pieters, T. Dimkov, E. Luysterborg, and M. Arnaud. *European Data Protection: In good health?*, chapter Privacy penetration testing: How to establish trust in your cloud provider? Springer, 2012. In print.

— Professional Publications —

[8] T. Dimkov and W. Pieters. Physical Penetration Testing: A Whole New Story in Penetration Testing. *PenTest Magazine*, 1(3):20–23, 2011.

Bibliography

[9] M. Abramowitz and I.A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. Dover publications, 1964.

[10] M. Abrams and D. Bailey. *Abstraction and refinement of layered security policy*, pages 126–136. IEEE Computer Society Press, 1995.

[11] W. Allsopp. *Unauthorised Access: Physical Penetration Testing For IT Security Teams*. Wiley, 2009.

[12] M. AlZarouni. The reality of risks from consented use of usb devices. In *Proceedings of the 4th Australian Information Security Conference*, pages 5–15, 2006.

[13] J. R. Aman, J. E. Conway, and C. Harr. A capstone exercise for a cybersecurity course. *Journal in Computing in Small Colleges*, 25(5):207–212, 2010.

[14] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based net-

work vulnerability analysis. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 217–224. ACM, 2002.

[15] T. Ashish. Social engineering: An attack vector most intricate to tackle. Technical report, Infosecwriters, 2007.

[16] N. Barrett. Penetration testing and social engineering hacking the weakest link. *Information Security Technical Report*, 8(4):56–64, 2003.

[17] R. Baskerville and M. Siponen. An information security meta-policy for emergent organizations. *Logistics Information Management*, 15:337–346, 2002.

[18] D. Baumrind. Research using intentional deception. Ethical issues revisited. *The American psychologist*, 40(2):165–174, 1985.

[19] L. Bettini, M. Loreti, and R. Pugliese. An infrastructure language for open nets. In *SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 373–377. ACM, 2002.

[20] C. Braghin, A. Cortesi, R. Focardi, and S. Bakel. Boundary inference for enforcing security policies in mobile ambients. In *TCS '02: Proceedings of the IFIP 17th World Computer Congresse*, pages 383–395, Deventer, The Netherlands, 2002. Kluwer, B.V.

[21] M. Bugliesi, G. Castagna, and S. Crafa. Access control for mobile agents: The calculus of boxed ambients. *ACM Transactions on Information and System Security*, 26(1):57–124, 2004.

[22] L. Cardelli, G. Ghelli, and A.D. Gordon. Types for the ambient calculus. *Information and Computing*, 177(2):160–194, 2002.

[23] L. Cardelli and A.D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[24] E.M. Chan, J.C. Carlyle, F.M. David, R. Farivar, and R.H. Campbell. Bootjacker: compromising computers using forced restarts. In *CCS '08: 15th ACM conference on Computer and communications security*, pages 555–564, NY, USA, 2008. ACM.

[25] F. Chen, J. Su, and Y. Zhang. A scalable approach to full attack graphs generation. In *ESSoS '09: Proceedings of the 1st International Symposium on Engineering Secure Software and Systems*, pages 150–163, Berlin, Heidelberg, 2009. Springer-Verlag.

[26] F. Chen, R. Tu, Y. Zhang, and J. Su. Two scalable analyses of compact attack graphs for defending network security. In *Proceedings of The International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 1, pages 627–632, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[27] L. Chen, D. Feng, and L. Ming. The security threats and corresponding measures to distributed storage systems. In *Proceedings of the 7th international conference on Advanced parallel processing technologies*, volume 4847, pages 551–559. Springer, 2007.

[28] Y. Chen, B. Boehm, and L. Sheppard. Value driven security threat modeling based on attack path analysis. In *Hawaii International Conference on System Sciences*, pages 1530–1605, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[29] R. Chinchani, A. Iyer, H. Ngo, and S. Upadhyaya. Towards a theory of insider threat assessment. In *Proceedings of the International Conference on Dependable Systems and Networks*. IEEE Computer Society, 2005.

[30] J. Clark, S. Leblanc, and S. Knight. Compromise through usb-based hardware trojan horse device. *Future Generation Computer Systems*, 27(5):555 – 563, 2011.

[31] J. Clark, S. Leblanc, and S. Knight. Risks associated with usb hardware trojan devices used by insiders. In *2011 IEEE International Systems Conference (SysCon),*, pages 201 –208, 2011.

[32] J. Concato, A.R. Feinstein, and T.R. Holford. The risk of determining risk with multivariable models. *Annals of Internal Medicine*, 118(3):201–210, 1993.

[33] J. Concato, P. Peduzzi, T.R. Holford, and A.R. Feinstein. Importance of events per independent variable in proportional hazards analysis i. background, goals, and general strategy. *Journal of clinical epidemiology*, 48(12):1495–1501, 1995.

[34] D. B. Cornish. The procedural analysis of offending and its relevance for situational prevention. *Crime Prevention Studies*, 3:151–196, 1994.

[35] D.B. Cornish and R.V. Clarke. Opportunities, precipitators and criminal decisions: A reply to Wortley's critique of situational crime prevention. *Crime Prevention Studies*, 16:41–96, 2003.

[36] R. De Nicola and M. Loreti. A modal logic for mobile agents. *ACM Transactions on Computer Logic*, 5(1):79–128, 2004.

[37] L.L. DeLooze. Counter hack: Creating a context for a cyber forensics course. In *FIE'08: Frontiers in Education Conference*, pages 1–6, New York, 2008. IEEE.

[38] J. DePoy, J. Phelan, P. Sholander, B.J. Smith, G.B. Varnado, G.D. Wyss, J. Darby, and A. Walter. Critical infrastructure systems of systems assessment methodology. Technical Report SAND2006-6399, Sandia National Laboratories, 2007.

[39] T. Dimkov, W. Pieters, and P. Hartel. Laptop theft: a case study on the effectiveness of security mechanisms in open organizations (extended abstract). In *CCS '10: Computer and Communications Security*, pages 666–668, NY, USA, 2010. ACM.

[40] B. Dragovic and J. Crowcroft. Information exposure control through data manipulation for ubiquitous computing. In *NSPW '04: Proceedings of the 2004 workshop on New security paradigms*, pages 57–64. ACM, 2004.

[41] B. Dragovic and J. Crowcroft. Containment: from context awareness to contextual effects awareness. In *Proceedings of 2nd Inernational Workshop on Software Aspects of Context*. CEUR Workshop Proceedings, 2005.

[42] B. Endicott-Popovski and D.L. Lockwood. A Social Engineering Project in a Computer Security Course. *Academy of Information and Management Sciences Journal*, 9(1):37–44, 2006.

[43] M. Felson. Those who discourage crime. *Crime and place*, 4:53–66, 1995.

[44] M. Felson and L. Cohen. Human ecology and crime: A routine activity approach. *Human Ecology*, 8:389–406, 1980. 10.1007/BF01561001.

[45] P. Finn and M. Jakobsson. Designing ethical phishing experiments. *Technology and Society Magazine, IEEE*, 26(1):46–58, 2007.

[46] P.R. Finn. *Research Ethics: Cases and Materials*, chapter The ethics of deception in research, pages 87–118. Indiana University Press, 1995.

[47] National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. The Belmont report: Ethical principles and guidelines for the protection of human subjects of research. Technical report, 1978.

[48] J. A. Goguen and J. Meseguer. Security policies and security models. In *3rd Symposium on Security and Privacy (S&P)*, pages 11–20. IEEE Computer Society, 1982.

[49] B.L.A. Goodman. Snowball sampling. *The Annals of Mathematical Statistics*, 32(1):148–170, 1961.

[50] D. Gorla and R. Pugliese. Resource access and mobility control with dynamic privileges acquisition. In *Proceedings of 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, pages 119–132. Springer, 2003.

[51] C. Greenlees. An intruder's tale-[IT security]. *Engineering & Technology*, 4(13):55–57, 2009.

[52] D. Ha, S. Upadhyaya, H. Ngo, S. Pramanik, R. Chinchani, and S. Mathew. Insider threat analysis using information-centric modeling. In *IFIP International Conference on Digital Forensics*, pages 55–73. Springer, 2007.

[53] J.A. Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum, and E.W. Felten. Lest we remember: Cold boot attacks on encryption keys. *USENIX Security*, pages 45–60, 2008.

[54] W.R. Hartmann, P. Manchanda, H. Nair, M. Bothner, P. Dodds, D. Godes, K. Hosanagar, and C. Tucker. Modeling social interactions: Identification, empirical methods and policy implications. *Marketing letters*, 19(3):287–304, 2008.

[55] R. Hasan, S. Myagmar, A. J. Lee, and W. Yurcik. Toward a threat model for storage systems. In *1st ACM Workshop on Storage Security and Survivability (StorageSS)*, pages 94–102. ACM, Nov 2005.

[56] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of ACM*, 32(1):137–161, 1985.

[57] M. A. Jackson. *Problem Frames: Analysing and Structuring Software Development Problems*. Addison-Wesley, 2001.

[58] Wayne A. Jansen, Serban I. Gavrila, and Vlad Korolev. Proximity-based authentication for mobile devices. In *Proceedings of The 2005 International Conference on Security and Management*, pages 398–404, 2005.

[59] X. Jiang, J.I. Hong, and J.A. Landay. Approximate information flows: Socially-based modeling of privacy in ubiquitous computing. In *Proceedings of the 4th international conference on Ubiquitous Computing*, pages 176–193. Springer, 2002.

[60] X. Jiang and J.A. Landay. Modeling privacy control in context-aware systems. *IEEE Pervasive Computing*, 1(3):59–63, 2002.

[61] G. Kitteringham. Lost laptops = lost data: Measuring costs, managing threats. Crisp report, ASIS International Foundation, 2008.

[62] I. Kotenko, M. Stepashkin, and E. Doynikova. Security analysis of information systems taking into account social engineering attacks. In *Proceedings of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 611 – 618, 2011.

[63] D. Lacey. Inventing the future–the vision of the Jericho forum. *Information Security Technical Report*, 10:186–188, 2005.

[64] P.Y. Logan and A. Clarkson. Teaching students to hack: curriculum issues in information security. In *SIGCSE'05: Special Interest Group on Computer Science Education*, pages 157–161. ACM, 2005.

[65] M. Marshall, M. Martindale, R. Leaning, and D. Das. Data loss barometer. Technical report, KPMG, UK, 2008.

[66] S. Mathew, S. Upadhyaya, D. Ha, and H.Q. Ngo. Insider abuse comprehension through capability acquisition graphs. In *Proceedings of the 11th International Conference on Information Fusion*, pages 1–8, 2008.

[67] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing. Ranking attack graphs. In *Recent advances in intrusion detection*, pages 127–144. Springer, 2006.

[68] A.M. Minkley. Cyberattacks: a lab-based introduction to computer security. In *SIGITE '06: Proceedings of the 7th conference on Information technology education*, pages 39–46. ACM, 2006.

[69] K.D. Mitnick and W.L. Simon. *The Art of Deception: Controlling the Human Element of Security*. Wiley, 2002.

[70] R. De Nicola, G. L. Ferrari, and R. Pugliese. KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on software engineering*, 24(5):315–330, 1998.

[71] Code of Federal Regulations. Title 45: Public welfare department of health and human services. part 46: Protection of human subjects. Technical report, 2005.

[72] E. Oladimeji. Security threat modeling and analysis: A goal-oriented approach. In *Proceedings of the 10 th IASTED International Conference on Software-Engineering and Applications*. ACTA Press, 2006.

[73] I. M. Olson and M. D. Abrams. *Infomation Security Policy*, pages 160–169. IEEE Computer Society Press, 1995.

[74] X. Ou, S. Govindavajhala, and A.W. Appel. Mulval: a logic-based network security analyzer. In *Proceedings of the 14th conference on USENIX Security Symposium*, page 8. USENIX Association, 2005.

[75] G. Palmer. De-perimeterisation: Benefits and limitations. *Information Security Technical Report*, 10:189–203, 2005.

[76] B.A. Pashel. Teaching students to hack: ethical implications in teaching students to hack at the university level. In *InfoSecCD '06: Proceedings of the 3rd annual conference on Information security curriculum development*, pages 197–200, NY, USA, 2006. ACM.

[77] J. Pauli and D. Xu. Threat-driven architectural design of secure information systems. In *Proceedings of the 7th International Conference on Enterprise Information Systems*, pages 136–143. ICEIS, 2005.

[78] P. Peduzzi, J. Concato, A.R. Feinstein, and T.R. Holford. Importance of events per independent variable in proportional hazards regression analysis ii. accuracy and precision of regression estimates. *Journal of clinical epidemiology*, 48(12):1503–1510, 1995.

[79] P. Peduzzi, J. Concato, E. Kemper, T.R. Holford, and A.R. Feinstein. A simulation study of the number of events per variable in logistic regression analysis* 1. *Journal of clinical epidemiology*, 49(12):1373–1379, 1996.

[80] W. Pieters. Representing humans in system security models: An actor-network approach. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 2(1):75–92, 2011.

[81] L. Ponemon. The human factor in laptop encryption. Technical report, Ponemon Institute, 2008.

[82] L. Ponemon. Cost of a lost laptop. Technical report, Ponemon Institute, 2009.

[83] A.A. Prayogi, J. Park, and E. Hwang. Selective role assignment on dynamic location-based access control. In *Proceedings of International Conference on Convergence Information Technology*, pages 2136–2135, 2007.

[84] C. W. Probst, R. R. Hansen, and F. Nielson. Where can an insider attack? In *Workshop on Formal Aspects in Security and Trust (FAST 2006)*, pages 127–142. Springer, 2006.

[85] C.W. Probst and R.R. Hansen. An extensible analysable system model. *Information Security Technical Report*, 13(4):235–246, 2008.

[86] M.R. Randazzo, M. Keeney, E. Kowalski, D. Cappelli, and A. Moore. Insider threat study: Illicit cyber activity in the banking and finance sector. *U.S. Secret Service and CERT Coordination Center Software Engineering Institute*, pages 1–25, 2004.

[87] I. Ray and N. Poolsapassit. Using attack trees to identify malicious attacks from authorized insiders. In *Preceedings of the European Symposium on Research in Computer Security*, pages 231–246. Springer, 2005.

[88] J. Rees, S. Bandyopadhyay, and E.H. Spafford. Pfires: a policy framework for information security. *Communications of the ACM*, 46:101–106, 2003.

[89] T. Ristenpart, G. Maganis, A. Krishnamurthy, and T. Kohno. Privacy-preserving location tracking of lost or stolen devices: cryptographic techniques and replacing trusted third parties with dhts. In *Proceedings of the Usenix Security Symposium*, pages 275–290. USENIX Association, 2008.

[90] F. B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, 2000.

[91] D.J. Scott. *Abstracting Application-Level Security Policy for Ubiquitous Computing*. PhD thesis, University of Cambridge, 2004.

[92] D.J. Scott, A. Beresford, and A. Mycroft. Spatial policies for sentient mobile applications. In *Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks*, pages 147–157, USA, 2003. IEEE Computer Society.

[93] A. Shah and J. Giffin. Analysis of rootkits: Attack approaches and detection mechanisms. Technical report, Georgia Institute of Technology, 2008.

[94] C. Soghoian. Legal risks for phishing researchers. In *eCrime Researchers Summit, 2008*, pages 1–11. IEEE, 2008.

[95] J. Sommers. Educating the next generation of spammers. In *SIGCSE'10: Special Interest Group on Computer Science Education*, pages 117–121, Wisconsin, USA, 2010. ACM.

[96] F. Swiderski and W. Snyder. *Threat Modeling*. Microsoft Press Redmond, USA, 2004.

[97] S. Trpe and J. Eichler. Testing production systems safely: Common precautions in penetration testing. In *Proceedings of Testing: Academic and Industrial Conference (TAIC PART 2009)*, pages 205–209. IEEE Computer Society, 2009.

[98] S. Türpe, A. Poller, J. Steffan, J.P. Stotz, and J. Trukenmüller. Attacking the BitLocker Boot Process. In *Trust '09: Proceedings of the 2nd International Conference on Trusted Computing*, pages 183–196. Springer-Verlag, 2009.

[99] E. Van Den Berg, S. Uphadyaya, P.H. Ngo, M. Muthukrishnan, and R. Palan. Mitigating the insider threat using high-dimensional search and modeling. Technical report, DTIC Research Report ADA450159, 2006.

[100] J. Walker. The extended security perimeter. *Information Security Technical Report*, 10:220–227, 2005.

[101] L. Wang and S. Jajodia. An approach to preventing, correlating, and predicting multi-step network attacks. *Intrusion Detection Systems*, pages 93–128, 2008.

[102] L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 2006.

[103] L. Wang, A. Singhal, and S. Jajodia. Measuring the overall security of network configurations using attack graphs. In *Proceedings of the 21st annual IFIP WG 11.3 working conference on Data and applications security*, pages 98–112. Springer, 2007.

[104] R. Wieringa. Conceptual modeling in social and physical contexts. Technical Report TR-CTIT-08-40, Centre for Telematics and Information Technology, University of Twente, 2008.

[105] R. Willison and M. Siponen. Overcoming the insider: reducing employee computer crime through situational crime prevention. *Communications of the ACM*, 52(9):133–137, 2009.

[106] J.M. Wing. *Scenario Graphs Applied to Network Security*, chapter 9, pages 247–277. Morgan Kaufmann, 2007.

[107] M. Workman. Gaining access with social engineering: An empirical study of the threat. *Information Security Journal: A Global Perspective*, 16(6):315–331, 2007.

[108] D. Xu and K.E. Nygard. Threat-driven modeling and verification of secure software using aspect-oriented petri nets. *IEEE Transactions on Software Engineering*, 32(4):265–278, 2006.

Web references

[109] J. Butler and S. Sparks. Windows rootkits of 2005, part two. www.securityfocus.com/infocus/1851, 2005. Last accessed: 08.07.2011.

[110] Infosec Research Council. Hard problem list. www.infosec-research.org/docs_public/, 2005. Last accessed: 08.07.2011.

[111] J. Heasman. Implementing and detecting a pci rootkit. Presented at Black Hat Europe, 2006.

[112] J. Heasman. Implementing and detecting an acpi rootkit. Presented at Black Hat Federal, 2006.

[113] P. Herzog. OSSTMM 2.2–Open Source Security Testing Methodology Manual. www.isecom.org/osstmm, 2006. Last accessed: 08.07.2011.

[114] P. Kleissner. Stoned bootkit. Presented at Black Hat USA, 2009.

[115] Microsoft. Microsoft threat analysis and modeling v3. www.archive.msdn.microsoft.com/tam, 2007. Last accessed: 08.07.2011.

[116] B. Rudis. Protecting road warriors: Managing security for mobile users, part one. www.securityfocus.com/infocus/1777, 2004. Last accessed: 08.07.2011.

[117] B. Rudis. Protecting road warriors: Managing security for mobile users, part two. www.securityfocus.com/infocus/1781, 2004. Last accessed: 08.07.2011.

[118] J. Ryder. Laptop security, part one: Preventing laptop theft. www.securityfocus.com/infocus/1186, 2001. Last accessed: 08.07.2011.

[119] J. Ryder. Laptop security, part two: Preventing information loss. www.securityfocus.com/infocus/1187, 2001. Last accessed: 08.07.2011.

[120] P. Saitta, B. Larcom, and M. Eddington. Trike v.1 methodology document [draft]. www.octotrike.org/papers/, 2005. Last accessed: 08.07.2011.

[121] Absolute Software. Lojack for laptops. www.lojackforlaptops.com. Last accessed: 07.08.2011.

[122] S. Stasiukonis. Social engineering the usb way. http://www.darkreading.

com/security/perimeter-security/208803634, 2006. Last accessed: 08.07.2011.

[123] Seagate Technology. Can your computer keep a secret? www.trustedstrategies.com/papers/, 2007. Last accessed: 08.07.2011.

Titles in the IPA Dissertation Series since 2005

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting*. Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs*. Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations*. Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data*. Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space*. Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization*. Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML*. Faculty of

Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models*. Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications*. Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition*. Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems*. Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use*. Faculty of Science,

Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems*. Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML*. Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery*. Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding*. Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks*. Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data*. Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy*. Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS*. Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems*. Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing,*

and Assimilation of Language Conglomerates. Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines*. Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras*. Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit*. Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks*. Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments*. Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design*. Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems*. Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms*. Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications*. Faculty of

Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation*. Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems*. Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification*. Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development*. Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation*. Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean*. Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques*. Faculty of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata The-*

ory and Modal Logic. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07

A. Mesbah. *Analysis and Testing of Ajax-based Single-page Web Applications*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08

A.L. Rodriguez Yakushev. *Towards Getting Generic Programming Ready for Prime Time*. Faculty of Science, UU. 2009-9

K.R. Olmos Joffré. *Strategies for Context Sensitive Program Transformation*. Faculty of Science, UU. 2009-10

J.A.G.M. van den Berg. *Reasoning about Java programs in PVS using JML*. Faculty of Science, Mathematics and Computer Science, RU. 2009-11

M.G. Khatib. *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12

S.G.M. Cornelissen. *Evaluating Dynamic Analysis Techniques for Program Comprehension*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13

D. Bolzoni. *Revisiting Anomaly-based Network Intrusion Detection Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-14

H.L. Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange*. Faculty of Mathematics and Computer Science, TU/e. 2009-15

M.R. Czenko. *TuLiP - Reshaping Trust Management*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-16

T. Chen. *Clocks, Dice and Processes*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-17

C. Kaliszyk. *Correctness and Availability: Building Computer Algebra on top of Proof Assistants and making Proof Assistants available over the Web*. Faculty of Science, Mathematics and Computer Science, RU. 2009-18

R.S.S. O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory*. Faculty of Science, Mathematics and Computer Science, RU. 2009-19

B. Ploeger. *Improved Verification Methods for Concurrent Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-20

T. Han. *Diagnosis, Synthesis and Analysis of Probabilistic Models*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-21

R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization*

and Their Applications to Medical Image Analysis. Faculty of Mathematics and Natural Sciences, UL. 2009-22

J.H.P. Kwisthout. *The Computational Complexity of Probabilistic Networks*. Faculty of Science, UU. 2009-23

T.K. Cocx. *Algorithmic Tools for Data-Oriented Law Enforcement*. Faculty of Mathematics and Natural Sciences, UL. 2009-24

A.I. Baars. *Embedded Compilers*. Faculty of Science, UU. 2009-25

M.A.C. Dekker. *Flexible Access Control for Dynamic Collaborative Environments*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-26

J.F.J. Laros. *Metrics and Visualisation for Crime Analysis and Genomics*. Faculty of Mathematics and Natural Sciences, UL. 2009-27

C.J. Boogerd. *Focusing Automatic Code Inspections*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2010-01

M.R. Neuhäußer. *Model Checking Nondeterministic and Randomly Timed Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-02

J. Endrullis. *Termination and Productivity*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-03

T. Staijen. *Graph-Based Specification and Verification for Aspect-Oriented Languages*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-04

Y. Wang. *Epistemic Modelling and Protocol Dynamics*. Faculty of Science, UvA. 2010-05

J.K. Berendsen. *Abstraction, Prices and Probability in Model Checking Timed Automata*. Faculty of Science, Mathematics and Computer Science, RU. 2010-06

A. Nugroho. *The Effects of UML Modeling on the Quality of Software*. Faculty of Mathematics and Natural Sciences, UL. 2010-07

A. Silva. *Kleene Coalgebra*. Faculty of Science, Mathematics and Computer Science, RU. 2010-08

J.S. de Bruin. *Service-Oriented Discovery of Knowledge - Foundations, Implementations and Applications*. Faculty of Mathematics and Natural Sciences, UL. 2010-09

D. Costa. *Formal Models for Component Connectors*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-10

M.M. Jaghoori. *Time at Your Service: Schedulability Analysis of Real-Time and Distributed Services*. Fac-

ulty of Mathematics and Natural Sciences, UL. 2010-11

R. Bakhshi. *Gossiping Models: Formal Analysis of Epidemic Protocols*. Faculty of Sciences, Department of Computer Science, VUA. 2011-01

B.J. Arnoldus. *An Illumination of the Template Enigma: Software Code Generation with Templates*. Faculty of Mathematics and Computer Science, TU/e. 2011-02

E. Zambon. *Towards Optimal IT Availability Planning: Methods and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-03

L. Astefanoaei. *An Executable Theory of Multi-Agent Systems Refinement*. Faculty of Mathematics and Natural Sciences, UL. 2011-04

J. Proença. *Synchronous coordination of distributed components*. Faculty of Mathematics and Natural Sciences, UL. 2011-05

A. Moralı. *IT Architecture-Based Confidentiality Risk Assessment in Networks of Organizations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-06

M. van der Bijl. *On changing models in Model-Based Testing*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-07

C. Krause. *Reconfigurable Component Connectors*. Faculty of Mathematics and Natural Sciences, UL. 2011-08

M.E. Andrés. *Quantitative Analysis of Information Leakage in Probabilistic and Nondeterministic Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2011-09

M. Atif. *Formal Modeling and Verification of Distributed Failure Detectors*. Faculty of Mathematics and Computer Science, TU/e. 2011-10

P.J.A. van Tilburg. *From Computability to Executability – A process-theoretic view on automata theory*. Faculty of Mathematics and Computer Science, TU/e. 2011-11

Z. Protic. *Configuration management for models: Generic methods for model comparison and model coevolution*. Faculty of Mathematics and Computer Science, TU/e. 2011-12

S. Georgievska. *Probability and Hiding in Concurrent Processes*. Faculty of Mathematics and Computer Science, TU/e. 2011-13

S. Malakuti. *Event Composition Model: Achieving Naturalness in Runtime Enforcement*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-14

M. Raffelsieper. *Cell Libraries and Verification*. Faculty of Mathematics and Computer Science, TU/e. 2011-15

C.P. Tsirogiannis. *Analysis of Flow and Visibility on Triangulated Terrains*. Faculty of Mathematics and Computer Science, TU/e. 2011-16

Y.-J. Moon. *Stochastic Models for Quality of Service of Component Connectors*. Faculty of Mathematics and Natural Sciences, UL. 2011-17

R. Middelkoop. *Capturing and Exploiting Abstract Views of States in OO Verification*. Faculty of Mathematics and Computer Science, TU/e. 2011-18

M.F. van Amstel. *Assessing and Improving the Quality of Model Transformations*. Faculty of Mathematics and Computer Science, TU/e. 2011-19

A.N. Tamalet. *Towards Correct Programs in Practice*. Faculty of Science, Mathematics and Computer Science, RU. 2011-20

H.J.S. Basten. *Ambiguity Detection for Programming Language Grammars*. Faculty of Science, UvA. 2011-21

M. Izadi. *Model Checking of Component Connectors*. Faculty of Mathematics and Natural Sciences, UL. 2011-22

L.C.L. Kats. *Building Blocks for Language Workbenches*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2011-23

S. Kemper. *Modelling and Analysis of Real-Time Coordination Patterns*. Faculty of Mathematics and Natural Sciences, UL. 2011-24

J. Wang. *Spiking Neural P Systems*. Faculty of Mathematics and Natural Sciences, UL. 2011-25

A. Khosravi. *Optimal Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2012-01

A. Middelkoop. *Inference of Program Properties with Attribute Grammars, Revisited*. Faculty of Science, UU. 2012-02

Z. Hemel. *Methods and Techniques for the Design and Implementation of Domain-Specific Languages*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2012-03

T. Dimkov. *Alignment of Organizational Security Policies: Theory and Practice*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2012-04