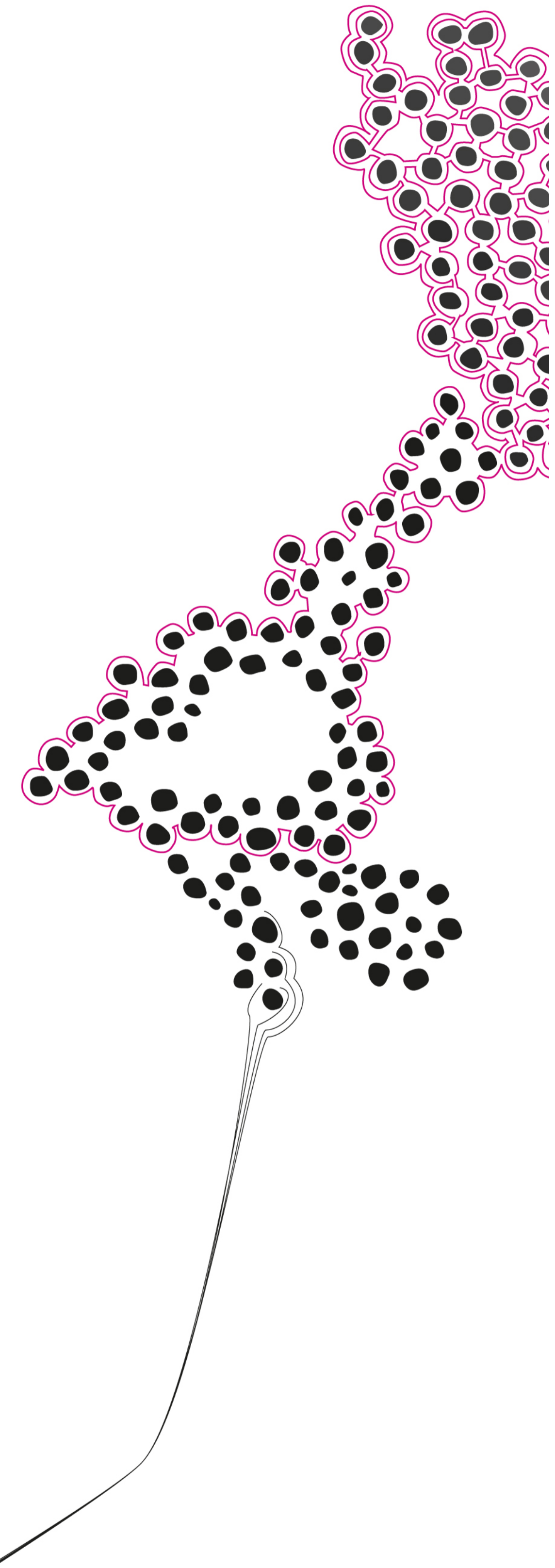

IT-BASED RISKS IN ADVERGAME CAMPAIGNS

A focus on fairness and privacy

UNRESTRICTED VERSION

R.E.J. de Groot

SCHOOL OF MANAGEMENT AND GOVERNANCE
INFORMATION SYSTEMS & CHANGE MANAGEMENT (ISCM)

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS & COMPUTER SCIENCE
INFORMATION SYSTEMS (IS)

SUPERVISORS
Dr. A.J.B.M. Wijnhoven (ISCM)
Dr. P.A.T. van Eck (IS)

DOCUMENT VERSION
Final Restricted (e-version)

UNIVERSITY OF TWENTE.

30-08-2012

UNRESTRICTED VERSION

Important: This version of this thesis is unrestricted and does not contain confidential chapters. The full version of this thesis is not available to the general public.

Master Thesis Jogchem de Groot

IT-based Risks in Advergame Campaigns

A focus on fairness and privacy

Amsterdam, August 30 2012

Author

Jogchem de Groot

Programme	Business Information Technology (MSc), School of Management and Governance, University of Twente
Student number	0047376
E-mail	r.e.j.degroot@alumnus.utwente.nl

Graduation committee

Fons Wijnhoven

Department	School of Management and Governance, University of Twente
E-mail	a.b.j.m.wijnhoven@utwente.nl

Pascal van Eck

Department	Faculty of Electrical Engineering, Mathematics and Computer Science University of Twente
E-mail	p.a.t.vaneck@utwente.nl

UNIVERSITY OF TWENTE.

Management summary

Purpose

The last few years have seen a rise in the use of advergames, games designed around a brand or product that are specifically created to communicate advertising messages. The combination of a brand with the fun and entertainment from playing a game results in outstanding performance with regard to brand memory, persuasion and visitor retention. However, due to their interactive, technological and data-oriented nature they have a risk profile different from other advertising methods, as they are also exposed to IT-based risks. The purpose of this study is to investigate what and how IT-based risks affect advergames, what their prevalence is, and to design a control to mitigate some of these risks.

Results

Four key areas of IT-based risk have been identified for advergames: security, where threats can target the brand owner's assets or visitors; fairness, where cheating can harm a brand's reputation and undermine the fun perceived by players, the key catalyst of advergame success. privacy, where threats are concerned with the loss of privacy sensitive data of players; and quality of experience risks, which can frustrate players and undermine fun as well.

A model for fairness and privacy threats that is operationalized for the technological context of advergames, consisting of detailed descriptions of eighteen different threats that are categorized according to what part of an advergame's architecture they target has been developed. This model enables advergame developers to understand these threats and to assess their own advergames.

This model has been used to perform a risk assessment of sixteen existing advergame campaigns and the results indicate overall high vulnerability for fairness threats, with medium to low impact and medium to high overall risk. The results indicate a serious and structural problem with fairness risks in advergames which significantly reduces the reliability and predictability of advergames as advertising instrument.

An effective solution to mitigate fairness risks has been designed that supports the detection of almost all forms of cheating, and that can be integrated in existing or new advergames with relatively low effort.

Recommendations

For brand-owners:

1. Include risk factors in the evaluation of whether the choice for an advergame is appropriate.
2. Consider the risk track record of an advergame company before selecting one.
3. Include risk management requirements in contracts negotiated with advergame creators. But always monitor and maintain an active position within the process, it is after all your brand at stake!
4. Allocate a realistic budget and time-frame on top of the basic advergame budget to assess and mitigate the risks for an advergame campaign.

For advergame creators:

1. Present a realistic view about risks to the client and include risk-management (costs) in proposals. Do not accept a project to create an advergame without a budget for risk management, it is after all also your reputation as an advergame creator at stake!
2. Make the advergame creation process risk aware: involve risk evaluation and communication at every stage of the advergame creation process.
3. Implement the proposed solution in new and existing advergames. Do not rely on your own anti-cheating controls and do not rely on anti-reverse engineering techniques as they are unlikely to be effective.
4. Perform a vulnerability assessment of both client-side and server-side code before going live. For the client-side code the threat model developed as part of this research should be used.
5. During live time, continuously monitor the campaign for incidents and respond appropriately.

Table of Contents

Acknowledgements

During this final project's research process I was helped and supported by many people, which I hereby want to thank for their support. First and foremost, I wish to thank my university supervisors, Fons Wijnhoven and Pascal van Eck, for their interest in supervising a thesis on this subject, their sharp discussions, helpful suggestions and guidance during the complete thesis process.

Next, I would like to thank Koen Voermans and Vincent Tuitman for their advice and feedback, Cornelis ten Napel for his initial support and I'm grateful to Josh Sisco, Wayne Janoe and Alice Kreiner for their help in proof-reading my thesis.

Last, but not least, I would like to thank my parents and friends for their help and support.

Chapter 1: Introduction

This chapter presents an overview of what advergames are and how and why they are used, followed by the problem description that presents the rationale for this thesis, and then proceeds with the conceptual model and the approach used for this research and an outline of the report.

1.1 Advergames

Over the last years interactive online games have seen an ever increasing use within advertising campaigns. Such games are called advergames, and they are designed around a brand, product or even a viewpoint and are specifically created to communicate advertising messages (Kretchmer, 2005). In simpler terms, advergames are interactive games that a company distributes for consumers to play. The objective of an advergame is to provide promotional incentives with the goal of obtaining a response from consumers, and in turn increasing consumer awareness. From an organizational perspective, advergames can stand by themselves, be featured in their own surrounding marketing campaign, called advergame campaign, or just be a part of a larger multi-channel marketing campaign. The practice of using games for advertising is sometimes also called advergaming. Advergames are considered a class of serious games (Alvarez et al., 2007) and a subcategory of branded entertainment (Winkler & Buckner, 2006). It is important to distinguish advergames from in-game advertizing, which “more closely resembles traditional product placement, but within a game, whereas for an advergame, the game is specially made to promote the brand” (Cauberghe & De Pelsmacker, 2010, p. 1).

Advergames are, just like any other game, played for entertainment and fun and seek to combine this with the brand. Most advergames are easy to learn and simple to play and offer quick rewards with a forgiving gameplay, which makes a fun experience. Research shows that the high level of brand interactivity and the association of a brand with the fun of playing a game results in better performance in brand awareness, message exposure, message association and buying intention (Gurău, 2008; Mau et al., 2006; Smith, 2007; Ping et al., 2010). This - in combination with possibilities for viral and opt-in marketing - causes fast-growing popularity within the advertising industry. The most recent figures by Boston-based Yankee Group estimated the size of the United States advergame industry to be worth US $312.2 million in 2009, up from $83.6 million in 2004. And with the current size of the worldwide video game market estimated at $67 billion in 2012 and forecasted to reach $82 billion by 2017[1], this media is not going away anytime soon.

Advergames are different from other (advertising) media in that the audience is in active control of the entertainment process and its outcomes (interactivity). As a result the experience varies every time the game is played. As a consequence the (branded) content of the game can be experienced and processed differently for every engagement. Additionally, advergames are different from traditional forms of advertising in that the advertising messages they contain are not broadcasted and pushed into the attention of consumers, but that consumers actively seek to engage with the advergame experience and choose to be exposed in exchange for entertainment. Consequently, advergames are good in retaining the attention of people (once they have started playing your advergame), but poor at acquisition (having people find your advergame in the first place). And therefore advergames (initially) need to be advertized for using other advertising media, an activity called seeding. Once enough momentum has been reached, the campaign might self-sustain or self-acquire new players through viral marketing mechanisms such as tell-a-friend, engaging friends in the game play, or by social media integration (Kaplan & Haenmein, 2011).

Advergames can be found for different platforms and devices and they can vary from simple re-skinned existing games (such as classic arcade games) to fully customized games specifically created for the brand or product. They are usually available for free, but sometimes require a product to be bought first, in order to play. Advergames are currently most often distributed on internet websites, in which case they are played within the player's internet browser. They can for example be placed on an existing company, brand or product website; or on a website specifically created for the advergame campaign (Buckner et al., 2002). In recent years there has been a trend towards integrating advergames with social media or distributing them through social media websites, taking advantage of the social infrastructure they provide and utilizing a

1 http://www.prweb.com/releases/2012/7/prweb9701884.htm (last accessed: August 20, 2012)

player's social network for promotion or within the game. Another recent development is the distribution of advergames through mobile media (Okazaki & Yagüe, 2011), in which case they may have just been made accessible for mobile devices or they might even have been specifically designed for them, relying on features such as the availability of location information. Such mobile advergames are often distributed exclusively through mobile app markets. Historically, but now seldom seen due to higher costs and longer development times, advergames were sometimes distributed on CD-rom discs, for game consoles or as downloadable and installable files.

1.1.1 Advergame classification

Brand integration in advergames comes in many different types and forms. Chen & Ringel (2001) created a three-step classification based on how a brand or product is featured within an advergame:

- **Associative**: The brand or product is associated with a lifestyle or a theme in the game, but not directly featured in the actual game-play. This is done by finding a combination of the brand and game theme, where the self-image of the brand's target groups and the game's aesthetics match.
- **Illustrative**: The brand or product is featured in such a way within the game that the player in some way interacts with it.
- **Demonstrative**: The game-play and game narrative are designed in such a way that they show, or that the player directly interacts with, the specific characteristics of the individual product within the game.

Even though this classification is often referred to in literature and practice, it has a couple of limitations. The categories are not mutually-exclusive, e.g., a game can be associative as well as illustrative. But more importantly it doesn't say anything about the dominance in the relationship between the game play and the brand-message. Svahn (2005) proposes a classification using a four-step ordinal scale that aims to take this into account:

1. **Type one** games do not contain any brand messages themselves but are placed on websites or social media pages, that do contain brand messages, to attract more visitors. For our thesis we do not consider these advergames.
2. **Type two** games contain superficial brand/product placements within the game or game-play. The brand or product placement is not necessary for the game to exist, and is theoretically interchangeable for another brand. An example would be a branded Tetris game.
3. **Type three** games are specifically created to communicate brand or product advertising messages and the brand/product and the messages are dominantly placed within the game and its narrative but are not a core part of the game-play. With some reworking the game could still be reused for another, similar brand.
4. In **Type four** games the game-play itself is the message, and game and message can not be detached from each other. These games are entirely custom made for the brand/product and its advertising messages.

These classifications are important because research shows that advergames that are highly thematic with respect to the brand, and games with a high game-product fit result in superior brand effects (Wise et al., 2008; Gross, 2010; Winkler & Buckner, 2006). But a stronger integration of the brand within the game potentially also means more brand impact from risks.

1.1.2 Advergame creation

Several types of companies, that can take up different roles with different types of expertise and responsibilities, can be involved during the planning, realization and operation of an advergame campaign. The principal companies involved are:

- **Brand owner** – Sometimes also referred to as game sponsor or publisher, is the company that commissioned the advergame and that owns the brand or product the game features and that it wishes to promote. It is responsible for setting the campaign requirements, goals and the allocated budget. The initiative is usually taken by the marketing or communication department responsible for promoting the brand.
- **Game developer** – Responsible for the game narrative, -play and technical realization of the

advergame. Development can be subdivided in front-end development consisting of the development of the actual game and campaign website, and back-end development which consists of all server-side logic and storage needed for the game to function that is not directly visible to the player.

Additionally, the following types of companies are also frequently involved:

- **Advertising agency** – Or creative agency, is often employed by a brand owner to assist in creating, planning and handling advertising, and can provide advice on overall marketing and branding strategy. Using an advergame for promoting a brand or product, possibly within an existing advertising campaign, is often suggested by them and they are typically responsible for the creative setting within which the advergame must be designed.
- **Media agency** – Helps a brand owner realize their marketing and communication strategies within the communication medium they are specialized in, such as interactive/internet technology, social media or mobile platforms. They are often responsible for seeding the advergame on the media of their expertise.
- **Hosting service** – Is responsible for hosting the front-end advergame campaign website and providing and operating the required infrastructure for the back-end components and storage services that the advergame campaign depends on.
- **Distributor** – Help with distributing an advergame on media different than a brand or campaign website. They include social media websites such as Facebook, game portals or in the case of a mobile advergame through market places such as Android Market or Apple App Store.

It's important to realize that not all these different companies might be present in every advergame campaign and that the roles they represent can overlap and thus do not always have to be embodied by different companies. Many different arrangements are possible in practice and some commonly seen ones are:

- In a minimal arrangement, just the brand owner and a game developer, experienced with advergames, are involved. In this case the brand owner will do parts of its own in-house marketing and advertising and the game developer typically has some media and advertising expertise.
- A full-service advertising agency might employ or incorporate their own media agencies, either integrated or as subsidiaries, which in turn might include game or interactive development services, each focusing on a different part of the advergame creation process.
- A campaign might be hosted by a separate company, which might be different from the company hosting the brand's corporate website, but this role is often taken up by one of the other companies involved, or even the brand owner self.
- For very large multi-channel campaigns each of these company types might be present and the game developer role might even be divided over different companies specializing in front-end or back-end programming, graphics design, sound design, etc..

The life cycle of an advergame campaign can be subdivided in three phases: creation, execution and completion. The creation phase starts by determining the campaign goals and targets, as well as the planning and allocated budget. It is then followed by creative design in which the advergame is outlined on an idea inspired by the brand or product. What then follows is the technical design and implementation of the game as well as setting up the campaign organization and initial seeding. While not formalized, the advergame creation process often most closely resembles an agile development process, such as extreme programming or scrum, with short time-boxed development cycles, prototyping, informal and volatile requirements and a high level of brand owner involvement (Iuppa & Borst, 2010). The next phase, execution starts with the launch of the advergame campaign and is supported by campaign monitoring and response. Finally, when the campaign completes the outcomes and results are evaluated and final closing actions are taken.

1.1.3 Advergame technology

If one takes a high-level architectural view of a stand-alone game, four different components can be identified in any game (see Figure 8):

- **Graphical representation**. This component is responsible for the graphical representation of the game on the player's computer screen. Its main function is to draw (or render) all the visual items that the game consists of on the screen area dedicated to the game and update them accordingly when the game transitions during game play. This component takes care of what a player sees of the game.

- **Input processing**. This component is responsible for capturing and processing player inputs to the game. All games take inputs so that the player can assert control over how the game transitions, such as keyboard presses, mouse motion and clicks, or even voice inputs. These inputs, once captured, have to be interpreted and processed into higher-level game user actions or commands that can be passed onto the main game logic. This component takes care of how the player can control the game.

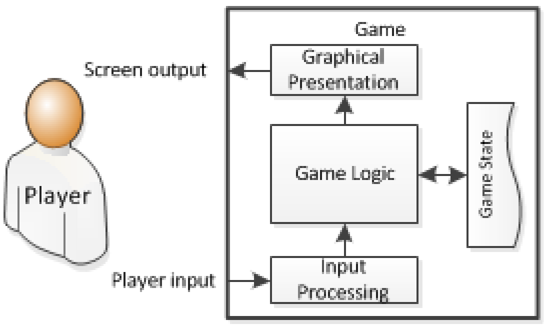

Figure 1: Structure of game components

- **Game state**. This component is responsible for representing the state of the game system. The game state consists of all dynamic data and variables that together represent the game at a specific moment in time. This data can for example consist of the users current score, the number of lives left, the list of game items collected, but also how far the player progressed within the game, a list of all enemies within the game and their current positions and directions of movement, and when the next enemy is due to be spawned, etc..
- **Game logic**. This component is responsible for all the game rules and transitions that together make up the workings of the game. It decides what game actions can take place within the game, and which ones can not. Its core typically consists of a periodically executed main game loop that operates on the user input events (from the input processing component), timer events and the game state, to compute a new game state according to its internal game rules and programmed game transitions, and signals the graphical representation to update accordingly. Examples of tasks that the game logic is concerned with include collision detection, game artificial intelligence, movement of game entities and the game score function. The score function decides under what game conditions the score is changed (thus how to gain points or lose points), and therefore implicitly defines what range of scores is possible within the game. Overall the game logic is responsible for the game's gameplay.

Most advergames use this basic architectural layout but also communicate player registration information and game results, among other things, with the campaign website in a client-server architecture. More on advergame client-server architecture can be found in section 4.1.

The two most important technology platforms for developing and distributing advergames are Adobe Flash and JavaScript. Flash-based advergames run within the player's internet browser and require the presence of the Flash player plugin, which currently has a market share of 95,36%[2] and which is available free of charge for various computer systems and devices. Flash is a platform for adding interactive and animated multimedia elements to web pages and supports game development features such as vector, raster and 3D graphics; and bidirectional streaming of video and audio (Adobe, 2012). It is currently the most popular platform for advergame development. Games developed using Flash are programmed in ActionScript, an object-oriented scripting language very similar to JavaScript. The latest version, ActionScript 3.0 (AS3) (Adobe, 2007c), is a significant redesign and improvement over the previous version, ActionScript 2.0 (AS2) (Adobe, 2007b), which is still used, and better suited for developing complex applications by allowing better control and code reusability. AS3 comes with an extensive standard API and many external libraries and components have been developed that can be reused in new applications. Flash content is distributed using the Adobe Flash file format called SWF. A SWF file packs the multimedia data, vector graphics and ActionScript code necessary to play the content in a binary format (Adobe, 2008). This data is organized within the SWF file using various tags that either contain encoded representations of data items (such as vector shapes, bitmaps, texts, fonts), actions (such as compiled ActionScript code), or control information. Compiled ActionScript code is executed within the Flash player using a virtual machine with its own native instructions. Two versions of the ActionScript Virtual Machine (AVM) exist, each having their own binary instruction format, one for AS2 (AVM1) and one for AS3 (AVM2). Both AVMs have a stack-oriented execution model and instruction format. Instructions for the AVM2 are contained within the DoABC tag of the SWF file format and are encoded in the ActionScript Byte Code (abc) format (Adobe, 2007a), which

2 http://www.statowl.com/flash.php (last accessed: August, 5 2012)

includes, next to encoded instructions, data necessary for loading and linking at runtime such as a constant pool containing strings, namespaces, class definitions, method signatures, etc. For older AVM1 flash files, the binary instruction format has a much simpler representation, but is scattered over several different tags within the SWF (Adobe, 2008).

Flash dominance is predicted to decline in the future in favor of HTML5 (Ankeny, 2011), the newest revision of the HTML standard by the World Wide Web Consortium (W3C), but it is currently still under development (W3C, 2012). HTML5 builds further on JavaScript, which is the premier scripting language used for adding dynamic, interactive, animated elements to web-pages and is often used to create browser-based (adver)games. Basic JavaScript is supported by all major browsers and executed using the browser's native JavaScript engine and therefore does not require a separate plugin or player. With JavaScript-based games the distinction is often made between those that require the improvements and extensions provided by the new HTML5 standard-under-development, which require a recent browser version (often called HTML5-games), and those that rely on pre-HTML5 technology that will work in most existing browsers (often simply called JavaScript games or Dynamic HTML (DHTML) games). DHTML games combine the use of various web standard technologies such as a static markup language (HTML), a client-side scripting language (JavaScript), a presentation definition language (CSS) and the Document Object Model (DOM) (W3C, 2005). Animated elements are created by accessing and manipulating the (graphical) objects that the document consists of, through the DOM programming API. However, different browsers support slightly different JavaScript versions and APIs and this makes it harder to get a uniform experience across different browsers. Freely available JavaScript libraries such as jQuery[3] and Dojo Toolkit[4] can make this process easier. JavaScript is a simple scripting language offering object-oriented and imperative programming styles and is formalized in the ECMAScript language standard (ECMA, 2011). JavaScript code can be directly embedded within a HTML document, either as snippets of code within <*script*> elements, or within the event attributes of various HTML elements, such as “onclick”; or it can be loaded as a separate JavaScript file (with the .js extension).

HTML5 is mostly an improved superset of the set of technologies that are collectively referred to as DHTML It promises better interoperability and better cross-platform support with a focus on supporting mobile applications. With respect to games the biggest improvements are the addition of some new features such as the <*canvas*> element that allows a script to directly draw on a designated area within the browser screen, instead of having to go through the DOM; and the <*video*> and <*audio*> elements that allow for direct integration of video and audio content within the page, without requiring a plugin. New markup and application programming interfaces allow for the development of complex web applications (including games). Due to these improvements games developed using HTML5 technology can potentially have a graphical performance that is on par with or even better than Adobe Flash, but DHTML games typically have a more basic look and feel, simpler game controls and are less snappy than most Flash games. Technologies that have formerly been used for advergame development but are nowadays rarely used include Java Applets, Adobe Shockwave and Microsoft Silverlight.

1.2 Advergame motivation & theory

A brand owner can decide to launch an advergame campaign for a number of different reasons and with a number of different goals. A few of the most important reasons and goals are grouped together and listed in the following table:

Grouping	**Reasons & Goals**
Brand image	• Wishing to associate a brand with the fun of playing a game, therefore improving brand attitude with consumers. • Reaping the benefits of the unique advantages that advergames can have on advertising measures like brand awareness, familiarity and recall. • Increasing the online presence of a company in a novel and interactive way. • Opening up to new audiences normally not reached.

3 http://www.jquery.com (last accessed: August 5, 2012)
4 http://www.dojotoolkit.org (last accessed: August 5, 2012)

	• Avoiding the saturation-effect that comes with (involuntary) exposure to standard forms of (e-)marketing, such as internet banners.
Persuasion & Education	• Being able to communicate persuasive messages more effectively by total brand immersion. • Interactively educating people about a brand, product or viewpoint.
Traffic	• Attracting more visitors to a brand website and having them spend more time • Receiving more “likes” or finding more “followers” for a social media account.
Financial	• Lower cost than some other forms of advertising such as TV commercials. • Expectancy of a higher campaign-wide return on investment.
Data collection & Marketing research	• Collecting contact information of people potentially interested in a brand for future contact (such as for a mailing list). • Building up marketing profiles of people using the personal data they provided or shared with the game through their social media profile. • Building up market intelligence by tracking choices made by players throughout the game.

Table 1: Reasons & goals for using advergames

Advergames perform especially well with regard to brand memory and persuasion and most research on advergames has investigated this relation. The rest of the section explains why advergames perform so well on these two aspects by discussing the results from some key scientific advergame studies and the psychological processed involved. But we start by providing a high-level overview of the psychological processing of entertainment in games, which forms the basis for both brand memory and persuasion.

Playing games can lead to strong arousal, a state of affection and activation. This excitement is part of the entertainment value of games. This is the result of mainly two important psychological processes (Nelson & Waiguny, 2012):

- **Emotional engagement.** An entertainment experience can consist of a huge variety of different emotions and expectancies for emotions, that can affect the information processing as well evaluation of media. Emotions such as joy and excitement, but also boredom or frustration, may transfer directly to the brand inside the game (Gurau, 2008; Wise et al., 2008).
- **Cognitive immersion**. The rich, multisensory media environment of games requires concentration, allocation of significant mental resources for gameplay understanding and solution finding, as well as physical control of the game situation. This full cognitive immersion may lead to a sense of *flow* for the game player, a pleasurable experience which is a state or sensation of total involvement, disconnected from your surroundings. Flow is an optimal situation for a person, which is achieved when the challenge meets the individual's skill, and does therefor not automatically occur (Chen, 2007). *Telepresence* is one construct of flow which is defined as the feeling of being “there”, inside the game world. Psychological effects of flow and telepresence are increased enjoyment, involvement, persuasion and memory which can lead to enhanced product knowledge, brand attitudes and purchase intentions (Li et al., 2002).

In conclusion, games combine cognitive, emotional and physical activities, that in an optimal game-player context lead to the experience of feeling flow, telepresence and strong emotions. When such emotions are expected, an enjoyable experience for the player occurs which, in an advergame context, will have an effect on brand memory and persuasion.

Advergames & brand memory

Memory-based measures such as recognition, implicit & explicit recall of the brands in a game are often used as an estimate of advertising effectiveness (Nelson, 2005), and acquire fairly high results for advergames. Several studies investigated this relationship. (Winkler & Buckner, 2006) found that in a study where participants played one of three different advergames each featuring a single brand, 86% of respondents remembered seeing the brand logo, even after a delay. Gross (2010) found that 100% of the players recalled what the game was about and brand featured of a highly congruent brand-game advergame a week after playing, and even for an incongruent version, 95% could remember what the game was about and 80% remembered the brand. (Yang & Wang, 2008) reported similarly high levels of recall with 87% of respondents able to recall the products featured with free recall measures, and 97% with aided recall measures. Brands placed more prominently within the game, and brands that are central to the gameplay are more likely to be recalled than those that are not (Lee & Faber, 2007; Schneider & Cornwell, 2005).

Advergames & persuasion

The persuasive power of a brand message (measured by for example brand attitude, brand choice/preference, sales, etc..) in a game is (1) related to the design of game and gameplay, and how the brand message is included in the game, and (2) the level of entertainment. Two different psychological phenomena explain how this takes places (Nelson & Waiguny, 2012):

Conditioning & affect transfer

Most advergames do not include explicit information about products, instead persuasion occurs passively (Smith & Just, 2009): relevant information is integrated into game elements or gameplay, or the entire gameplay is thematically around the intended message (Wise et al., 2008). From this perspective the persuasion effects of embedded brands depend mostly on two different processes.

- **Conditioning**. If the positive experience of the game (emotional stimulus) is frequently combined with the brand (neutral stimulus), the emotional experience may be remembered together with the brand, and as a consequence positive feeling towards the brand increases.
- **Affect transfer**. A separate but similar process may take place when persons in a good mood or with a positive feeling tend to evaluate subjects and objects more positively. A pleasing gaming situation will not only let players evaluate the game more positively, but maybe also let them evaluate the embedded brand more positively. This classic affect transfer is well known in advertising research, and in case of advergames as assume to be rather strong in comparison with traditional advertising formats, as the potential to produce positive emotions using entertainment is much higher (Nelson & Waiguny, 2012).

For these effects to happen the game should be evaluated as fun, and fun is therefore the most important attribute for the evaluation of games. Fun is found the strongest motivation to play advergames in the first place (Youn & Lee, 2005), and the strongest antecedent of attitude towards the game (Hernandez, 2008). Games are evaluated most positively if they get the user in a state of flow or telepresence and several studies have reported a positive relationship between the attitude towards the game and attitude towards the brand (Mau et al., 2008; Nelson et al., 2006; Wise et al., 2008).

Implicit learning effects

Embedded brand persuasion can also happen through implicit learning effects. Advergames typically give the brands embedded in them a much longer exposure time, and more repeat exposure than traditional advertising media. Such repeat exposures may influence the audiences' perceptions of the brand through incidental and implicit learning, and perceptual *fluency* (ease of processing presemantic or visual features of stimuli) towards the stimulus (the embedded brand) may occur. More familiar stimuli are processed quicker, which in turn generally leads to more positive evaluation (Nelson & Waiguny, 2012). Several studies investigated this relation. For example, Hang and Auty (2011) conclude that the synergistic effects of exposure and interactivity led to superior brand evaluations in a study where they manipulated the exposure time to brands in a game setting. In addition to perceptional processing fluency, advergames also provide the possibility to communicate messages rhetorically or procedurally, which allows learning effects through conceptual fluency (ease of processing language or conceptual information) (Smith & Just, 2009).

1.3 Problem description & rationale

With the unique advertising potential that advergames have to offer, it is easy to understand why an increasing number of companies have used or experimented with advergames for their marketing communication over the past few years. However, many companies jumping on the advergame bandwagon have found that their campaigns did not achieve the results they hoped for, did not go as planned, or even resulted in damage to their brand image. The unique interactive nature of advergames results in a class of risks that are different from traditional risks associated with advertizing, but similar to risks well-known in the area of information technology. These risks include topics as security and privacy, quality, availability and conformance.

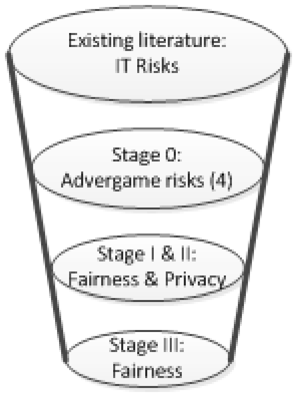
Existing literature: IT Risks
Stage 0: Advergame risks (4)
Stage I & II: Fairness & Privacy
Stage III: Fairness

Figure 2: Scope tunnel applied during research

When consumers are involved, the same psychological processes that are responsible for the positive performance of advergames, have the potential to amplify the negative effects of such risks, as for example conditioning and affect transfer do not only occur for positive experiences, but also for negative ones. Negative feelings and emotions experienced during the game may be transferred to the brand embedded in the game, potentially leaving the player behind with a more negative attitude towards the brand than before. Therefore any risk that results in negative experiences during gameplay, or results in a negative attitude towards the game or the campaign may negatively impact the brand on any of the measures discussed and this effect is strengthened by repeat exposures.

As an example of a risk in advergames related to security, many advergame campaigns offer attractive and valuable prizes to the best performing players in order to stimulate participation. But in practice many advergames are insufficiently secured and are easy to manipulate. Dishonest players are able to submit scores that have not been achieved in a fair way with the goal of winning prizes. This is often done in an unsubtle manner, and other players, who notice, might get frustrated and disgruntled. The advergame campaign that was once regarded as sympathetic, might look a lot less sympathetic to them now, and this will influence the effectiveness of the campaign. In worst-case scenario's it might even have a negative impact on the reputation of the brand or the company, leaving the company off worse than before. The advergame industry does not seem to know how to deal with such manipulation or how to prevent it. Knowledge and expertise in security is often missing and there has been very little research on security risks and techniques for the most important advergame technology platforms.

In general, we expect that there is insufficient knowledge about what risks exist and that advergame and advertising companies have insufficient experience with managing them. Incidents often catch them by surprise, and when they are expected, advergame managers are often not able to respond effectively. Fixed budgets and relatively short time schedules often mean that there is insufficient focus on risk during design and development, as developers and advertisers rather spend their time on making the game just a bit more creative, or just a bit nicer looking, something that helps them persuade their client in choosing them. Risks are then almost fully transferred to the brand owner, because it is their brand at stake, but they are not aware of this as they are insufficiently informed by the developers.

The advergame industry has seen a rapid growth over the last years, but now, as more and more clients have had negative experiences with advergames, this growth might be under pressure. Especially in times of economic uncertainty, companies will prefer more well known and proven methods for advertising, with a clearer risk profile. Advergame companies have or should come to the conclusion that managing and reducing the risks that campaigns are exposed to, is of vital importance for continued growth. Therefore there should be a high demand within advergame companies to be able to deal with such risks, but an extensive literature search showed that a systematic understanding of risks in this context has not been developed so far. However, we consider this necessary to mature advergames into a proven instrument within the marketing mix that can deliver the reliable and predictable results that brand owners look for.

1.4 Conceptual model

Risks in advergame campaigns are the subject of this thesis and Figure 3 presents the conceptual model showing causal relations relevant to this research. The model has been subdivided in three sections and each of these sections will be explained from bottom to up. A full discussion of advergame risk and risk concepts can be found in chapter 2.

At the bottom we see the basic risk model that follows the risk formula that is often used for IT-based risks:

*Risk = Likelihood * Impact* where *Likelihood = Threat * Vulnerability*

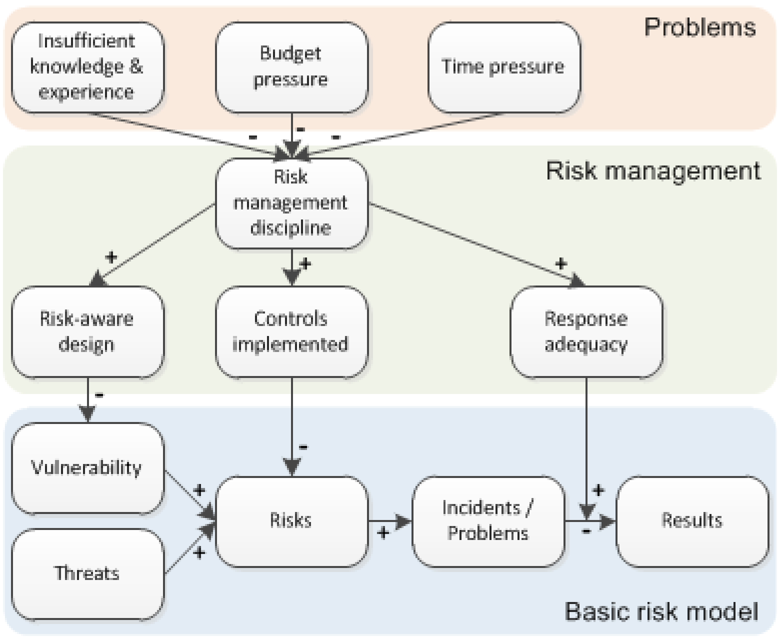

Figure 3: Conceptual model showing causal relations for this research

According to this formula, the likelihood of an advergame risk depends on the threat but also the vulnerability or susceptibility of the advergame campaign to this threat. When risks become true they cause or problems that negatively impact the results of an advergame campaign or the brand owner. Risks can be managed by risk management processes that aim to reduce the likelihood of risk occurrence and the impact when they do occur, and this is represented by the middle part of the model. This research is concerned with three ways to manage and mitigate risk (from left to right):

1. Being aware of the threats that affect an advergame campaign and taking them into account during design has a positive effect on the vulnerability of the campaign to those threats.
2. The implementation of specific controls or countermeasures that protect against certain threats or reduce their impact will reduce the amount of risk an advergame campaign is exposed to. A control is any technical, administrative, management, or legal method installed to reduce risk.
3. Finally, when incidents and problems do occur, responding appropriately on discovery can reduce their impact.

The degree to which these risk mitigation methods are implemented in an advergame, and thus to what amount of risk an advergame is exposed, depends on the risk management discipline taken by the advergame developers and managers. And this brings us back to the problems that we defined in section 1.3, as depicted in the top section of the model. There is insufficient knowledge about advergame risks, and insufficient experience with managing them and this problem is further intensified by the budget and time pressure under which advergame campaigns are created. Even though budget and time pressure in advergame projects are unlikely to go away, having knowledge of the what risks exists and what is needed to manage and mitigate will allow a rational trade-off to be made.

1.5 Research approach

Since no previous research has been performed on the subject of IT-based risks in advergame campaigns, the following two working hypotheses are formulated for this thesis:

H1: *Existing advergame campaigns are exposed to a large amount of IT-based risk.*

H2: *This amount of risk is unnecessarily large and controls can be designed to mitigate (some of) these risks.*

This is used to define the research goal:

To develop detailed insight in how different IT-based risks can affect advergames, their their prevalence in existing advergame campaigns; and to design a control to mitigate (some of) these risks

This research goal can be subdivided in three separate research objectives:

1. Identify and provide a model and detailed technical descriptions for the different IT-based threats to advergames, that will allow us to understand how these threats work and when advergames are vulnerable.
2. Investigate the actual manifestation and prevalence of risks in advergames by performing a risk assessment on existing advergame campaigns using the knowledge about threats from the first objective.
3. Design a control to mitigate one or more of the risks identified.

In order to fulfill the research objectives and goal, the following research questions have to be answered:

1. What are advergames? Why and how are they used?
2. To what IT-based risks are advergames potentially exposed?
3. How are the different threats from these IT-based risks manifested in existing advergame campaigns?
4. What is the prevalence of these risks in existing advergame campaigns?
5. How can a solution (technical or organizational) be designed to mitigate these risks?
6. What is the validity of this research?

1.5.1 Research structure & scope

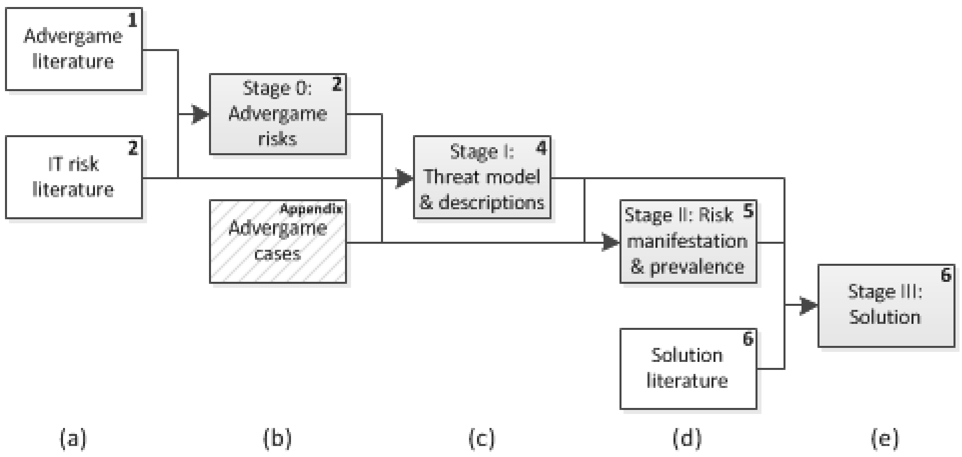

Figure 4: Research structure

From the three objectives it follows that this research consists of three consecutive stages (I to III), where the outcome of each state is needed as input for the next stage. Figure 4 shows the structure of the outcomes needed in order to fulfill the research goal according to the technique described by (Verschuren & Doorewaard, 1999). Research results are depicted in gray, supporting literature analyses in white and empirical sources are striped. The corresponding chapters in this thesis are shown in the corners of the blocks. From this figure it follows that *(a)* a confrontation between advergame literature and IT-risk literature resulted in *(b)* the identification of a number of potential IT-based risk areas for advergames. Given that no existing literature about risks in advergames exists, yet identifying these risk areas is a necessary precondition for developing threat descriptions, we've named this the stage 0 result of our research, and it corresponds with the exploratory nature of this research topic. Two of these advergame risk areas have been selected and a number of existing advergames have been analyzed to identify how threats derived from these two risk areas can affect them. Subsequently, the result of stage I consists of *(c)* a threat model and detailed technical threat descriptions. In stage II this threat model and the detailed threat descriptions are used to perform a risk assessment on a number of existing advergame campaigns to determine *(d)* the prevalence and manifestation of risk in these advergames. Finally in stage III, using the outcomes from stage I and II and an investigation of solutions described in literature, *(e)* a solution has been designed for the most urgent risk in advergames: fairness.

Due to the fact that this subject had been previously unexplored in literature it was unclear in the beginning what results to expect at each stage and how to continue from there. Consequently, scoping decisions have been made at each stage during the research process and this scope tunnel is shown in Figure 2. Additionally, to keep this research project focused and manageable within the assigned time frame the following general scope was applied:

- Strictly, only advergame campaigns are considered. Related and similar interactive advertising campaigns such as (non-game) quizzes and in-game advertizing are left out of scope.
- Many categories of risk can be identified for advergames, but this research project strictly considers risks from the IT/IS discipline. With regard to risk, we ultimately take the perspective from the brand owner. However, in practice many of these risks will be shared with or even transferred to the companies involved in creating the advergame campaign and this research will thus be very much of interest to them as well. Out of scope are:
  - Risks pertaining to marketing communication demands. We will not consider the question what a good or a bad advergame is, or how marketing messages can be most effectively communicated and any risks related to this.
  - Risks pertaining to corporate responsibility (is it ethical to promote candy to preteens using advergames?)
  - Risks from the perspective of the players (consumers). Some of these risks are likely shared with the brand owner, but will have a different impact (for example: a privacy leak will result in an individuals private data being compromised, but results in reputation damage, claims and possible legal/regulatory consequences for the brand owner).
  - Risks related to misinforming consumers in advergames.
- We will not try to be exhaustive with regard to identifying advergame threats and risks, we will limit ourselves to those risks that currently pose the biggest threat (as concluded from part *(b)* of our research, see research structure section).

1.5.2 Methodology

To answer the different research questions and fulfill the research objectives, different research methods are used and Table 2 provides an overview.

Research question	Stage	Methods
1. What are advergames? Why and how are they used?	0	Literature research
2. To what IT-based risks are advergames potentially exposed?	0	Literature research. Scenario analysis
3. How are the different threats from these IT-based risks manifested in existing advergame campaigns?	I	Exploratory, qualitative empirical research. Grounded theory. Reverse engineering. Literature research
4. What is the prevalence of these risks in existing advergame campaigns?	II	Descriptive, qualitative empirical research. Risk assessment. Scenario analysis.
5. How can a solution (technical or organizational) be designed to mitigate these risks?	III	Design science research. Prototype development. Literature research.
6. What is the validity of this research and the proposed solution?	-	Literature research. Critical evaluation.

Table 2: Research methods used to answer research questions

All stages in this research follow a qualitative approach. The results from stage 0 are based on literature research and confrontation. Stage I and II both rely on the analysis of empirical data collected from 16 different Dutch advergame campaigns. The main analysis technique used here is reverse engineering. But where the research from stage I is exploratory in nature, using an approach similar to grounded theory to induce theory (a threat model) from empirical observations, is stage II descriptive and deductive in nature, using the developed theory to perform a risk assessment on existing advergame campaigns. Stage III is concerned with solution design and development and follows a design science research methodology. For full details on the methodology, data collection and analyses used, see chapter 3.

1.5.3 Impact and relevance

From a practical perspective the outcomes of this research help advergame companies and brand owners to better understand and manage the risks that an advergame campaign is exposed to, and the results with regard to risk prevalence and manifestation should raise awareness of the urgency of the problem. The threat-model and detailed threat descriptions can be used by advergame developers to assess the vulnerability of their advergames and as a guideline for more risk-aware design. The proposed solution can be used for existing and new advergames to significantly mitigate the problem of fairness in advergames, giving advergame developers more time and opportunity to focus on the other risks identified during this thesis. In summary, this research contributes to the maturity of advergames as a reliable and predictable advertising instrument.

From an academic perspective we provide a contribution to literature by identifying IT-based risks in the context of advergames, a subject previously unexplored. Existing IT risk frameworks are not directly usable because traditional information systems or software are different from advergames, which are not evaluated within a functional domain, but within the advertising domain. Thus, even though some of the risks within IS have their counterparts within advergame campaigns, their consequences are different. Also from a technical perspective, due to the architectural design of most advergames, the risk model is different from that of, for example, traditional web security.

1.6 Structure of report

The structure of the rest of the report is as follows:

- Chapter 2 discusses risk & risk management foundations and identifies four IT-based risk areas that pose a serious threat to the success of advergames (stage 0).
- Chapter 3 presents the methodology and data collection methods used, and the analyses performed for the three main research stages of this thesis.
- Chapter 4 presents the analytical results from research stage I, centering upon the threat model and detailed descriptions of fairness & privacy threats.
- Chapter 5 presents and discusses the risk assessment results from research stage II, revolving around the prevalence of fairness (& privacy) risks in existing advergames.
- Chapter 6 proposes a solution that can significantly mitigate the impact of fairness risks in advergames (stage III).
- Chapter 7 evaluates the validity and reliability of the three research stages.
- Chapter 8 presents the conclusions, contributions and suggestions for future work of this research.

Chapter 2: Advergame risks

This chapter describes the four most significant areas of IT-based risk that can negatively contribute to the success of an advergame campaign or the brand owner. These risk areas have been identified by confronting the advergame literature (see sections 1.1 and 1.2) with existing literature on IT-based risks. Two approaches from the *Risk IT Practitioners Guide* (ISACA, 2009) have been used to identify relevant risks: A top-down approach that starts from the overall advergame objectives and performs an analysis of the most relevant and probable IT-risk scenarios impacting these business objectives, and a bottom-up approach that starts with generic IT-risk scenarios and identifies which ones are also applicable to the context of advergames.

2.1 IT-risk fundamentals

Before we start our discussion of risks in advergames, it is necessary to introduce some important concepts related to IT-risk first. Generally speaking a risk can be defined as the product of the likelihood and the impact of a potential (negative) event, in formula form:

*Risk(e) = Likelihood(e) * Impact(e)*

This should be understood as: the risk associated with event *e* is the likelihood that event *e* takes place multiplied by the expected impact when event *e* does indeed occur. For IT-based risks it is more common and insightful to express the likelihood in terms of threat and vulnerability, according to the following formula (Caballero, 2009):

*Risk(e) = Threat(e) * Vulnerability(e) * Impact(e)*

The ISO/IEC 27005:2011 standard (International Organization for Standardization (ISO), 2011) provides the following definitions.

- **Risk**: “a combination of the consequences that would follow from the occurrence of an unwanted event and the likelihood of the occurrence of the event”.
- **Threat**: “An event that is a potential cause of an incident, that may result in harm of systems and organization”. The term **threat-level** is sometimes used as a measure of the likelihood that this particular threat event takes place.
- **Vulnerability**: “A weakness of an asset or group of assets that can be exploited by one or more threats “. In combination with a particular threat, it refers to the susceptibility to that threat, thus the chance that when a threat occurs it will be successful in triggering a particular vulnerability.
- **Impact**: “adverse change to the level of business objectives achieved”. In other words, the negative consequences (and amount of loss associated with them) coming from the occurrence of a threat-event.

Both threat and vulnerability are probabilistic in nature (just like likelihood in the first formula), but the impact expresses the amount of loss involved. Schlarman (2009) groups impacts into four categories depending on how they affect the objectives of an organization:

- **Financial** impacts are risks that cost the organization money such as the loss of assets or incurring extra costs for dealing with unforeseen circumstances.
- **Regulatory** or **legal** impacts lead to fines and possibly prison for example due to non-compliance with regulation or law.
- **Business** impacts generally hinder the companies (or project's) overall success.
- **Reputation** impacts cause harm to an organization's prestige, brand, goodwill or influence.

Although these categories are not mutually exclusive (for example a regulatory outcome might also negatively impact reputation), they do provide a useful high-level way of categorizing impacts.

A risk rarely consists of just a single threat and corresponding vulnerability (i.e. the risk of theft of privacy-sensitive data), but often consists of many different threats and corresponding vulnerabilities with different impacts that together make up the total risk for such an event. This total risk can be computed by taking the sum over all the likelihood-impact pairs for each specific threat to which there is some degree of vulnerability:

$$Risk_{total}=\sum_i Likelihood_i * Impact_i \quad \text{where} \quad Likelihood_i = Threat_i * Vulnerability_i$$

Risk is a well-understood concept within the IS/IT discipline and numerous IT-risk management frameworks are described in literature and used in practice. IT-risk management is defined as "the process of identifying vulnerabilities and threats to the information resources used by an organization in achieving business objectives, and deciding what countermeasures, if any, to take in reducing risk to an acceptable level, based on the value of the information resource to the organization" (ISACA, 2006). The most important frameworks for IT-risk management are the Risk IT framework by ISACA (ISACA, 2009), the NIST SP 800 30 framework (Stoneburner et al., 2001) and the ISO/IEC standard 27005 reference model (ISO, 2011), but many more exist. These frameworks describe a set of reference processes and activities with regard to risk management that should be implemented and integrated with the systems development life cycle (SDLC). Two of these activities are relevant to this thesis and we define them as follows:

- **Risk assessment**: The identification, evaluation, and estimation of the levels of risks involved in a particular situation and it involves the analysis of the vulnerabilities, threats, likelihood, loss or impact, and theoretical effectiveness of controls.
- **Risk mitigation**: The efforts taken to reduce either the probability or consequences of a threat, either by reducing vulnerability or by implementing appropriate risk-reducing controls. In this context, a **control** or **countermeasure** is any technical, administrative, management, or legal method installed to reduce risk.

2.2 Security

As one of the biggest present-day IT risks, security-related risks also pose a major threat to advergame campaigns, due to their interactive, technological and data-oriented nature. Advergames have a completely different security risk profile than traditional advertising methods, including existing internet-based advertising methods such as internet banners. Security is typically not on the radar of the marketeers that organize and plan advergame campaigns, and even for the more technically-inclined game developers, security and secure design & development are often overlooked or low on their priority list.

Threats

Security threats against advergame campaigns can be roughly subdivided according to what or who they target. Security threats can target the advergame organizers, such as their campaign website, or their (corporate) infrastructure and data:

- Theft of valuable visitor data is a major risk as advergames often attract a lot of visitors that leave behind sensitive personal information. Even when the information they leave behind is just an email-address and a password to login to their account, attackers might be interested as many people reuse their password for other accounts. This often also has severe privacy consequences for visitors.
- But adversaries might also be after the data and infrastructure of the advergame organizer. Even though advergame websites are not always directly connected to the brand owner's corporate systems, advergame managers will likely login to the campaign back-end from those systems, and compromising the back-end system might be the starting point for further intrusions.
- The advergame website might be defaced with a political, activist or abusive message.
- The campaign might be sabotaged by causing service disruptions (denial of service attacks).
- Finally, adversaries might try to influence the competition outcome of an advergame campaign by obtaining confidential operational information or by making modifications to the server logic or score databases.

But adversaries can also specifically target the unwary visitors of an advergame campaign by abusing security problems that exist in an advergame campaign. For example, the website might be modified to directly try to steal or trick the visitor into disclosing sensitive information, by making them believe that this is necessary in order to play the game, or by using the website to distribute malware that links a visitor's computer to a botnet and that intercepts sensitive information such as credit card payments once installed.

Vulnerability

Vulnerability to these security threats can result from general website vulnerabilities such as out-of-date or insecure software, security misconfiguration, insufficient access restrictions, etc.. or can specifically result from security flaws or programming mistakes that exist in the server-side logic custom-made for the advergame campaign. The Open Web Application Security Project (OWASP, 2010) publishes a 3-yearly top-10 of most critical web security risks, most of which are also applicable to advergame campaign websites and their server side logic. Vulnerabilities like SQL injection flaws and insecure direct object references can result in the theft of data. Other vulnerabilities like cross-site scripting (XSS), cross-site request forgery, and unvalidated redirects or forwards, can enable adversaries to execute scripts in a victims browser to hijack sessions, distribute malicious content or redirect the victim to a malicious websites. Finally, broken authentication or session management, and insufficient transport layer protection may lead to attackers compromising players passwords, keys, session tokens or other sensitive private information they shared with the advergame website.

Impacts

The impact of security threats in advergame campaigns can be among the following four categories:

1. News coverage of security breaches can result in serious reputation impacts, such as damage to a companies brand image, goodwill and influence.
2. Financial impacts might result from for example, asset loss, the costs of urgently fixing security-related problems, incident investigation, and possibly financial compensation.
3. Regulatory impacts might come from fines or judicial actions coming from non-compliance with security regulations, especially when related to sensitive information from visitors.
4. Business impacts can for example result from players seriously reconsidering participation in a current or future campaign, or doing business with your company in general, after news of a privacy breach.

Even though security issues pose a major risk to advergame campaigns, these risks are not significantly different from general IT or website security risks. There already exists a large body of academic research and practice-oriented literature on website security risks and their prevalence, for example the OWASP project mentioned. The biggest issue here seems to be a lack of awareness with advergame organizers. Consequently, we do not expect to find significant new academic insights or results for the context of advergames, and therefore general security risks are left outside of scope for the rest of this research.

2.3 Fairness

Many advergame campaigns feature competitive elements in one way or another. These elements provide an incentive for repeat play and add to the experience value of advergames and are therefor popular with advertisers. Regardless of whether these elements are highscore rankings, friend-challenges or achievement badges, the perceived fairness of these elements is of vital importance to the experience players have. Perceived unfairness can lead to a loss of enjoyment and motivation, the key catalysts for advergame effectiveness, frustration, cynicism towards participation and even a negative attitude towards the game or campaign (Duh & Chen, 2009), especially when valuable prizes are at stake. Threats against fairness are probably the most prolific of all possible threats against advergame campaigns and many campaigns in the past had to deal with them, often suffering serious impacts, such as campaign under-performance, incurring significant extra costs in trying to deal with them, campaign shutdown or even damage to brand image beyond the advergame campaign.

Merriam-Webster defines fairness as “marked by impartiality and honesty” and “conforming with the established rules”. For this study an (adver)game is considered fair when its benefits are distributed according to an individual's legitimate participation and performance within the game. Benefits in this context can refer to anything from high-score rankings to tangible prizes. And with legitimate in this context means: according to the explicit (and implicit) rules set by the advergame's game logic and the campaign's terms & conditions. This definition does not cover the situation where the rules themselves can be considered unfair, such as when multiple prizes might be given away, but no limit is set on how many prizes one can win. One player (whether he played fairly or not) might then be able to win multiple, or even all prizes, for example by winning a daily prize every day. This can be perceived as very unfair by other players.

Threats

For advergames, fairness generally means absence of cheating, which is defined by (Yan & Randell, 2009) as "any behavior that a player uses to gain an advantage over his peer players or achieve a target in an online game is cheating if, according to the game rules or at the discretion of the game operator […], the advantage or the target is one that he is not supposed to have achieved". Thus, threats to fairness come from the different techniques and opportunities that exist for cheating in an online advergame campaign, which often center around manipulating technical elements of the advergame. The subjects of cheating and cheat prevention in online games are relatively well recognized in literature. The most thorough and complete attempt at systematically listing and categorizing opportunities for cheating is provided by (Yan & Randell, 2009). However, not all of the categories they identify are applicable to the context of advergames, which are almost exclusively simple browser-based games. Of the fifteen categories they identify, the following are most applicable (in order):

- **Exploiting misplaced trust**. Cheats that involve tampering with code and/or data on the client side. This touches the core problem with cheating in advergames, which often rely on a client-server architecture where the entire game logic is executed on the players computer and only results are communicated back to the server. Developers place too much trust in the client, which in fact, cannot be trusted at all, because players have total control over their game client.
- **Exploiting lack of secrecy**. Data exchanged between the advergame and the server is often transmitted in plain-text or with insufficient encryption or integrity protection, which makes it easy for cheaters to modify important status updates or game commands to their advantage.
- **Exploiting machine intelligence**. Cheaters can sometimes use artificial intelligence techniques in the form of bots of scripts that automatically solve a puzzle or play a simple game under perfect conditions and speed. Sometimes this can be as simple as scripting a few mouse clicks to happen at the right moment in time.
- **Exploiting a bug or loophole**. Here cheaters exploit a bug or loophole in the game design or game play without having to modify any game code or data. Once discovered it might give them a major advantage over other players.

Other categories listed that might also be of influence to cheating in advergames include: **collusion**, **abusing game procedures**, **denying service to peers**, **compromising game servers** and **social engineering**. However, threats are not just limited to technical game manipulation. For example, because of insufficient verification of the identity of winners, it might be possible for a single player to win many prizes by playing for his friends, or by taking on false identities, despite limits set. In the age of social media, where networks of friends are made explicit on the web, this is often easy to notice for other players.

Vulnerability

The vulnerability of an advergame to a specific fairness threat (cheat) depends on how susceptible to cheating the design and structure of the game are and to what extent this is prevented by specific protective measures (controls) that are taken. Absence of cheating can generally be accomplished by two mutually supportive approaches (Yan & Choi, 2002):

- **Prevention**: Making cheating impossible or more difficult, either by design or structure, or by implementing certain controls.
- **Detection**: Being able to detect when cheating does occur, so that actions can be taken to restore fairness.

Both approaches can have a technological and an organizational dimension, for example detection might be facilitated by technical measures, but still requires the task to be performed by an actual person. The attitude of the advergame managers towards cheating can make a big difference in actual and perceived fairness and is therefore an import moderating variable (see Figure 5). With regard to attitude, the motivation to make an advergame campaign fair can be extrinsic (motivated by the potential negative effects when externally observed) or intrinsic (motivated by fairness as a virtue, independent of whether a violation can be externally observed). Thus the vulnerability of an advergame to fairness threats depends on the vulnerability of the general design and

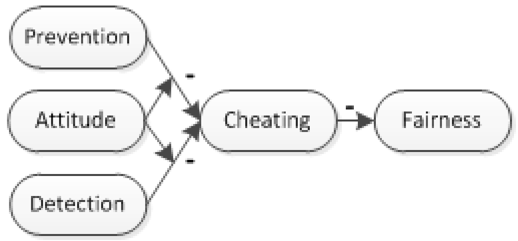

Figure 5: Conceptual model of fairness

architecture of the advergame to fairness threats. Whether there are preventive or detective controls in place that can impede the successful exercise of these vulnerabilities and their theoretical effectiveness, and the level of skill and effort required to successfully exercise any remaining vulnerability.

<u>Impacts</u>

Fairness related threats in advergame campaigns can have an impact on the following three impact categories:

1. Reputation impacts cause harm to brand image and goodwill. For example players might be left with a negative attitude towards the brand after perceiving the game as unfair. In a worst case scenario a company or brand in a context-sensitive industry might sustain serious reputation damage, that reaches well beyond the scope of the advergame campaign after news coverage of the problems.
2. Business related impacts hinder the success of the advergame campaign project. For example the campaign goals might not be achieved because of players that are put off by the amount of cheating in the advergame, or in a worse scenario, the campaign might even have to be preliminarily stopped.
3. Financial impacts cost the company organizing the advergame money. For example, unforeseen costs might incur during the campaign from having to spend extra man-hours to filter out cheaters, to improve the cheat-resistance of the advergame or to replace expensive prizes unjustly given away.

Issues related to fairness are relatively unique to games and have not previously been researched in the context of advergames, even though they pose a major threat to their success and even to brand reputation. Existing research related to this topic is focused on cheating in multiplayer games and provides little connection to the context of advergames. And although cheating is possibly the largest risk to advergame campaigns, until now there has been no investigation into the extent of the problem in advergames as far as we know. As a consequence, focusing this research on fairness can provide a novel and unique contribution to the academic field as well as practical knowledge that can support the creation of more robust advergame campaigns with regard to this subject. Therefore, we will include fairness within our scope and make it the main focus of the next stages of this research.

2.4 Privacy

Most advergames nowadays collect data about their players. This can be done passively by tracking choices and progress made throughout the game. Or actively, by asking players to register and share personal information before, during or after they play the advergame. Active data collection can be mandatory: the game can not be played before providing the requested information. Or voluntarily, often with an incentive, for example if you want to keep updated about the game or the brand, to be eligible to win prizes or get a discount or sample, or possibly even to unlock new parts of the game, such as bonus levels to play. In recent years, with the rise of social media, advergames have integrated the new possibilities they provide, and sharing your information might mean signing in with your social media account and sharing parts of your profile and social data with the advergame campaign, or providing the advergame permission to post status updates on your profile.

Consequently, advergames offer many opportunities to collect data from players, however this data is often privacy-sensitive. This can range from providing just an email-address to full contact details including address and phone-number to even almost every detail about ones life in the case of unrestricted social media profile access. In unauthorized hands this information could for example be used for spamming people, illegal profiling, identity theft and for social-engineering based attacks, such as fishing for banking details. This is potentially very valuable data for adversaries with such intentions, and advergame campaigns are therefore exposed to the threat of this sensitive data being stolen or abused. The number of large-scale leaks of privacy sensitive information has significantly increased over the last few years, increasing the urgency of this risk. Privacy in this context is mostly concerned with maintaining the following two properties (and together with availability they make up the CIA triad, one of the core principles in information security) (Solove, 2006):

- **Confidentiality**: Privacy sensitive data must be free from disclosure (intentional or accidental) to unauthorized individuals or systems.
- **Integrity**: Privacy sensitive data must be free from modification (intentional or accidental) by unauthorized individuals or systems.

Every advergame campaign should be designed to take measures to protect the privacy of its players to guarantee that these two properties maintain satisfied.

At the same time when sensitive data is being collected and processed, different privacy laws and regulations aimed at protecting consumer privacy and privacy on the internet, may apply, depending on jurisdiction. These regulations state what can and what cannot be collected, how this data is allowed to be processed and under what conditions, how this data needs to be stored, how long this information may be stored, who needs to be informed about what information is stored and how and for what purpose, and how this information can be modified, updated or removed. Non-compliance with these regulations can lead to fines and intervention. Privacy-policies are often mandated to communicate how personal information is collected, stored and used.

In general, sharing privacy sensitive information is a matter of trust placed in the advergame organizer, by the player. If the player finds out that his privacy, and therefore his trust is violated, regardless of whose fault this is, this will most likely have a strong negative and lasting impact on the the players attitude towards the brand. The risk also works in the other direction. With the increasing amount of data being collected by advergames, if a player feels that he has to share too much information, or that this information is too sensitive, or that he doesn't trust the client company with this information, he might decide not to play at all. If this happens often, it will have a negative effect on the number of players, and therefore on campaign effectiveness.

Threats

Thus, the first group of threats to privacy are related to the different techniques and opportunities for adversaries that result in the violation of the two properties for privacy sensitive data. The most important threat to privacy in advergames is the unauthorized access (disclosure or modification) to privacy-sensitive information (violation of confidentiality). But also the case where adversaries can modify data of other players, or even pretend to be another player can pose a major threat (violation of integrity). The most important privacy threats to advergames in this group are:

- Sensitive data from players can be intercepted during communication by others because of insufficient transport level security.
- Sensitive data from players might be leaked through the mechanisms by which the advergame operates, or might be easy to obtain through poor authentication mechanisms (such as identifying and loading somebodies profile on the basis of just their e-mail and being able to assume their identity on this basis).
- Sensitive data might be insufficiently protected or cryptographically insecurely stored by the advergame organization, so that in case of server compromise, this information is easily stolen (a security risk with a privacy consequence).

The second group of of threats comes from the incorrect use of privacy-sensitive data:

- Insufficient communication on how sensitive information is collected, processed, shared and stored, or non-compliance with privacy regulations could lead to fines and judicial interventions.
- Violating ones own privacy policy with regard to collecting, processing, sharing and storing privacy-sensitive data could lead to lawsuits and regulatory non-compliance.

Vulnerability

The vulnerability of an advergame to privacy threats depends on to what extent the advergame has been designed with privacy in mind, absence of certain implementation bugs and what specific protective measures (controls) that are taken. Unlike with fairness, protection against privacy threats can only typically rely on prevention. Detection, while still useful, would be too late, as the privacy violation has then already occurred, and can typically no longer be corrected (contrary to what is often possible for fairness).

Impacts

The impact of privacy threats in advergame campaigns can be among the following four categories:

1. Reputation impacts might for example result from damage to a companies brand image and goodwill after news coverage of a privacy leak in a campaign.
2. Financial impacts might result from the costs of having to urgently fix privacy-related problems, legal defense, or having to financially compensate victims of a privacy breach in a campaign.
3. Regulatory impacts might come from fines or judicial actions coming from non-compliance with

privacy laws, or come for example in the form of being legally required to notify all participants who have been involved in a privacy breach in a campaign.

4. Business impacts can for example result from players seriously reconsidering participation in a current or future campaign, or doing business with your company in general, after news of a privacy breach. But even if everything seems fine, requesting too much private information may put off potential players.

Even though the topic of privacy has been well researched, it remains a “hot” topic both in research and in practice. In advergames, privacy-sensitive data is not only collected and stored as (aggregate) profiles in databases owned by the advergame organization, but sometimes also used and integrated within the game and gameplay, which can be relatively unique for advergames. This justifies investigating to what privacy threats advergames are exposed, and therefore we will include the first group of privacy threats within the scope of this research for research stage I and II.

2.5 Quality of experience

Risks pertaining to quality are widely understood and researched within the information systems field, and influence every part of the system development life cycle. But although advergames are software as well, their quality is primarily evaluated from a marketing perspective, which focuses more on the entertainment experience than on whether they fulfill all functional requirements. Therefore, many of the quality aspects and measures that are relevant to regular software development, are less important for advergames. But since the entertainment experience is the driving factor behind the effectiveness of advergames, any IT-based risk that can affect the quality of this experience is important.

System requirements

Advergame audiences differ from traditional computer game audiences in that their computer's computational and graphical abilities are often very average, whereas specific game enthusiasts often try to stay close to the state of the art with regard to hardware, in order to achieve the best possible graphics and performance. For game developers, who often come from these game enthusiast communities, it is natural to show their technical ability by incorporating state of the art visual elements within the games they develop, with equally demanding system requirements and this also applies to advergames. The brand owner, unaware of this, is happy to see a highly promising game with visually stunning elements during during development, but when the advergame campaign goes online, a significant share of the players will find that the game doesn't play smoothly as their system is too old or slow. This can have a strong negative effect on their experience of the game, which they will stop playing at best, or frustrated about at worst, which might transfer to the brand. When the brand owner finds out that a significant share of the target audience can not be reached by this game, it will either face a significant impact on campaign goals, or face high additional costs in changing the game to be less demanding.

Usability

The typical audience of advergames consist of consumers and casual gamers. In order to attract those people to your branded game, the game should be inviting, accessible and easy to play. Game mission, game play and game controls should be easy to understand and the game interface should be designed to facilitate and communicate this. Unlike the players of high-end computer games, advergame users are typically not willing to invest much time in studying complex instructions and practicing a complex set of controls. The game should be instantly appealing as a fun experience, and not as a complex one, or they might not even start playing it. When during the advergame creation process, game designers are spending a lot of time on creatively designing and implementing the game, the game play and mechanisms may start to look too trivial to them, and as a result new elements are often added to the game play. What still looks as a simple, but fun game to them, might then be experienced as complex and difficult to play to the advergame audience, and they might not even be tempted. Similarly, games that do seem appealing to their audience, but turn out to be too difficult to control or play, might lead to frustration, which is detrimental to player experience therefore to project success (Isbister & Schaffer, 2008).

Quality

Even though advergames are pieces of software, and software can be evaluated on its functional and structural quality, these traditional notions have less meaning in the context of advergames. Advergames are ultimately seen as advertising instruments and their quality is therefore assessed by whether it can communicate the necessary advertising messages within the entertainment experience it provides. Whether the advergame fully adheres to its original functional specifications is therefore less relevant, as it can still be effective in providing an entertaining experience. However, showstopping bugs that reveal themselves during game play - for example by making the game hang, crash or clearly misbehave - can have a significant harmful effect on player experience, especially if they occur frequently. Unfixed, this can have a significant effect on campaign results, but fixing them might be costly.

Availability

For the advergame campaign to be effective, it needs to be available. If an advergame campaign has too much downtime, visitors and players will not be able to play the game, during which none of the advergame campaign goals can be realized. Problems with availability can come from problems with the website (not being able to load the game), with the server logic (not being able to start a game session or submit your score), or from problems from integration with numerous other subsystems (for example not being able to invite a friend when this is required to participate). Threats to availability can be internal (faults, errors and failures) or external (for example denial of service attacks, see section 2.2) (Avižienis et al., 2004). Lack of availability can be a frustrating experience for players, especially if the advergame involves a competition, and absence of availability monitoring can prolong the time the campaign is unavailable.

Issues related to quality of experience and the risk they pose for advergames are interesting from an academical perspective, because the way they impact the success of advergame campaigns is different from regular IT systems, and potentially even different compared to regular games because of the advertising context. However, research on this subject is likely resource and time consuming (especially using empirical methods), as it will rely on extensive testing and monitoring, where at the same time it could potentially lead to relatively few results. For instance, if we take the subject of quality, we could for example test games for show-stopping bugs ourselves and monitor the comments and complaints from players on publicly visible media about these games. However, this can easily lead to few and insignificant results and is thus not reasonable in comparison to the time and effort this requires. Therefore we won't investigate this topic any further, and leave it outside the scope of this research.

2.6 Conclusion

This chapter introduced the subject of risk to the context of advergames, and described four areas of IT-based risk that can negatively affect the success of an advergame campaign. The interactive, technological and data-driven nature of advergames results in a higher risk-profile than for other types of advertising and creating awareness of the importance of these risks is the first step in managing them. We argued that the four risk areas often have a different impact for advergames than for IT-systems in general and an overview of the types of impact that the different risks have for advergames is found in Table 3. Because of practical considerations and time constraints, we do not have the time to focus on all identified risk areas in this research. Consequently, fairness and privacy have been selected as the focus of this research and the next chapter presents the methodology used to investigate these risks.

	Financial	**Regulatory**	**Business**	**Reputation**
Security	x	x	x	x
Fairness	x		x	x
Privacy	x	x	x	x
Quality of Experience	x		x	

Table 3: IT-based risks in advergames and their potential impact categories

Chapter 3: Methodology, data collection and analysis

This chapter presents the methodology used by this research project. In the previous chapter we selected fairness and privacy as the two IT-based risk areas that we focus this research on. Section 1.5 defines three different objectives for this research, and we explained that these objective correspond with the three consecutive research stages (I to III) that this thesis consists of. An overview of these research stages and their inter-dependencies can be seen in Figure 6.

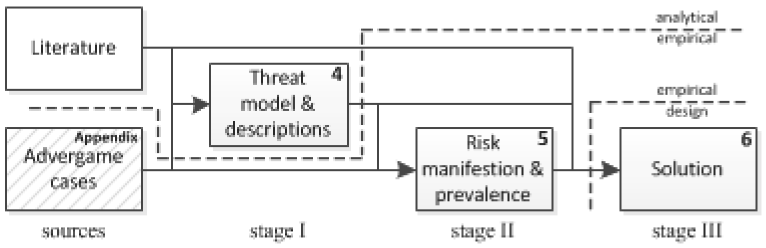

Figure 6: Research structure, showing analytical, empirical and design parts

The outcome of the first stage, which corresponds to the first objective, is of analytical nature, and consists of insight in the problem domain through a threat model consisting of detailed technical descriptions of fairness and privacy threats that advergames are exposed to. The outcome of the second stage, which uses this newly developed theory to analyze the manifestation of, and exposure to, risk in actual advergame campaigns, is of empirical nature and consists of insight in the prevalence of fairness and privacy risks in existing advergame campaigns. Both stages rely for their analysis for a significant part on the use of the same empirical evidence, data from sixteen different existing advergame campaigns. But where the first research stage uses this empirical data for the development of the theory necessary to understand and assess fairness and privacy threats in advergames; the second stage uses this empirical data for descriptive research, by describing the actual manifestation and prevalence of these risks in advergames. Because this is done by using the theory of the first stage as input, this theory will then also be empirically tested, however using the same empirical evidence to form and test a theory has some consequences for its validity, see chapter 7. Both stage I and II of this study will be performed using a qualitative research approach, but where the first research stage is inductive (form theory from empirical observation), is the second stage partially deductive (use theory to analyze particular advergame campaigns). Finally, in the third and last stage a solution to mitigate fairness risks is designed, the most prevalent and urgent risk area as indicated by the results of stage II, and this stage uses requirements derived from the threat model and descriptions from stage I.

3.1 Threat theory development methodology (stage I)

The objective for the first stage of this research is to investigate the problem domain of fairness and privacy threats and is formulated as:

> To identify and develop a model of, and detailed technical descriptions for, the different threats to fairness and privacy in advergames, that will allow us to understand how these threats work and when advergames are vulnerable.

This is qualitative research that seeks to form new insight and theory by developing detailed technical descriptions of the different threats to fairness and privacy in advergames. The research method we used to is similar to grounded theory, defined by Martin & Turner (1986) as “an inductive, theory discovery methodology that allows the researcher to develop a theoretical account of the general features of a topic while simultaneously grounding the account in empirical observations or data.” and the empirical data used for this stage consists of a number of existing advergame campaigns. Each advergame is analyzed using reverse engineering techniques (see section 3.5) in order to identify possibilities for cheating and privacy-related vulnerabilities. This analysis is supported by various sources of information such as scientific literature, internet resources, technical documentation, etc. It is important to stress at this point that the analysis of each case is not performed as a single-pass linear process, but more closely resembles a multi-pass process where cases are re-analyzed iteratively as new insights are gained during research. This mode of analysis has some resemblance to the hermeneutic circle (Myers, 2004). In this way patterns of similar vulnerabilities (and related threats) will start to emerge that can ultimately be used to induce a set of generic fairness and privacy threats within the context of advergames. Once these generic threats have been identified, a model (and categorization) of these threats is developed that provides further insight in how they are interrelated. Thus although the data sources are empirical, the outcome of this stage more analytical or theoretical in nature. For each identified threat a detailed, standardized threat description will be developed according to the format described in Table 4.

Section	**Required information**
Description	This section provides an overview of the what, why and how of a threat and will discuss the technical impact that a successful execution of a threat can have.
When vulnerable?	This section discusses under what conditions an advergame campaign is to what extent vulnerable. It provides guidelines for assessing the vulnerability of an advergame campaign.
How to perform?	This section provides detailed instructions on how such a threat can be performed and has the following goals: • to provide more advanced understanding of how the threat works • to understand the level of skill and effort required to perform the threat • to provide guidelines on how to test a threat against an actual advergame.
Threat-level	This section will assess the threat-level that a threat poses to an advergame campaign. The threat-level depends on the following aspects: • The **technical impact** that results from the successful exercise of a threat (thus without taking contextual or business aspects into account). For fairness this means the degree of fairness violation: will the successful execution of the cheat give a cheater only a minor indirect advantage? Or will it allow him to fully compromise fairness? For privacy this means the amount of loss of privacy properties. • The **prevalence** or **awareness** of a particular threat. Is this type of threat well known and widely used in practice? Or is it obscure and rarely used? • The amount of **skill** required to perform a particular threat. A threat that can be performed even by novice computer users will pose a much bigger threat than one that requires intricate knowledge of certain technologies and advanced programming skills. • The amount of **time and effort** required to exercise the threat? A threat that is easy and quick to perform will pose a much bigger threat than one that requires a significant investment of time and resources. The threat-level will be valued by the researcher on an ordinal scale with values low, medium and high and a motivation will be provided.
Example(s)	This section gives an example using one (or more) of the advergame cases that are vulnerable to this particular threat.

Table 4: Format of detailed threat description

As a direct result of our passive and non-invasive data collection method (see section 3.4), anything on the server-side of the advergame cases can not be analyzed. Therefore, threats that target the server-side directly in a way that does not involve a regular client-server interaction will be excluded.

3.2 Risk assessment methodology (stage II)

The objective for the second stage of this research is:

> To investigate the actual manifestation and prevalence of fairness and privacy risks in advergames by performing a risk assessment on existing advergame campaigns.

This stage uses the newly developed theory about threats from the first research stage to perform the risk assessments. It is the second empirical stage of this research as it also involves the analysis of existing advergame campaigns and is thus based on observation, but the results will be descriptive in nature, as they describe the present-day state of fairness and privacy risk exposure of advergames.

The risk assessment for each advergame is performed using the research model depicted in Figure 7 and is used for both fairness and privacy risks. The model corresponds with the concepts and definitions that we discussed in chapter 2 and follows the basic risk formula where *Risk = Likelihood * Impact*. The likelihood is a composite measure that depends on how much motivation the campaign provides for potential adversaries and on the total assessed level of vulnerability of a campaign for either fairness or privacy threats. This total vulnerability level depends on the amount of skills and the amount of effort required to successfully take advantage of one or more vulnerabilities. For impact, an estimate is made for the amount of potential asset and mission loss that can result from the successful exploitation of one or more of the advergame campaigns vulnerabilities, a simplification of the four impact categories from section 2.1.

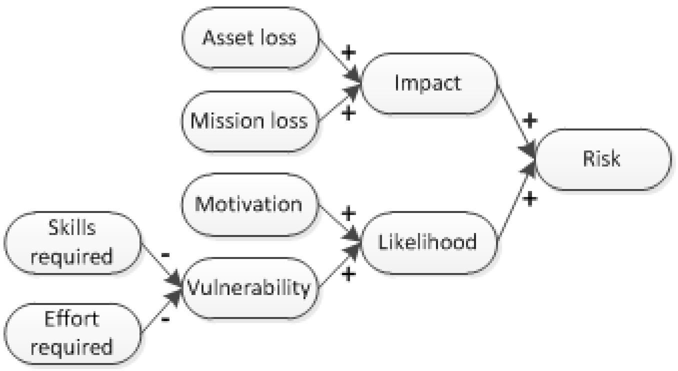

Figure 7: Research model used for advergame risk assessment

3.2.1 Scale

Risk assessment is the process of estimating (the value of) risk and this can be done from a quantitative and qualitative perspective. With quantitative risk assessment, risk is computed by the application of numerical measures for the two components of risk: the magnitude of the potential loss (the impact) and the probability that the loss will occur (the likelihood, or the product of threat and vulnerability). As a consequence from a quantitative perspective the value of risk is expressed as the expected loss. However it is not always possible to assign (sufficiently accurate) quantitative values to the likelihood or impact. Qualitative risk assessment offers an alternative approach where the components of risk are determined by qualitative scales and detailed descriptions (Pritchard, 2010). For this research we've selected qualitative risk assessment for the determination of risk in the different advergame campaigns examined during this study. The reasons for selecting qualitative assessment over quantitative assessment are:

1. We do not have enough data to make accurate numerical estimates, this is, for example, because we generally do not have enough information about the budgets or investments associated with the individual campaigns, or most costs resulting from risk impact. These are highly specific to the specific advergame and the type of company involved.
2. Some impacts are very hard to quantify numerically such as reputation damage, even though they are very relevant.
3. Descriptions lend themselves better to explain the details of the reasons for the existence of the risks as well as the conditions which contribute to risk severity. This aids our second objective to understand how risk is manifested within existing advergame campaigns.

We've chosen to use an ordinal scale to express all variables in we take into account for our risk assessment. This will allow us to maintain an ordering for the different items we measure and the risk outcomes we assign, without having to say anything about the relative size or degree of difference between the items we measured. This corresponds with the situation that we do not have enough data to make accurate numerical estimates, and therefore an ordinal scale will work well with qualitative risk assessment. In order to assign meaningful ordinal values to the variables we measure, we will use a simple three-step ordinal scale: low, medium and high. For composite variables that depend on one or more other variables in an ordinal scale, we will assign a similar label of low, medium and high to ranges of ordinal values, with the goal of making these outcomes easier to understand.

3.2.2 Operationalization

This section provides operational definitions for the concepts in Figure 7 and explains the methodology used to measure them. We start with the left side of the model.

Vulnerability

The concept of vulnerability here refers to the total vulnerability level or the sum of all vulnerabilities of an advergame campaign to fairness or privacy threats and is assessed by using the threat model and threat descriptions that resulted from the first stage of this research. It is assessed using the following questions:

- How vulnerable is the general design and architecture of the advergame to each threat?
- For each threat to which the advergame's design and architecture is vulnerable, are there preventive or detective controls in place that can impede the successful exercise of the vulnerability to that threat and how effective are they?
- For any remaining vulnerability, how much skill, time and effort is required to successfully exercise that vulnerability (possibly by circumventing a preventive control or by avoiding detection)?

The level of vulnerability is then expressed as a combination of the skills and effort required. For these two variables we can assign an ordinal value from 1 to 3 (high to low), according to the following cut-offs:

- *Skill*: (1, high) good programming/network/security penetration skills are required; (2, medium) some technical skills are required that correspond with a more advanced computer user; (3, low) only basic computer skills and no specific knowledge is required.
- *Effort*: (1, high) requires a significant effort in time or resources to perform (multiple days); (2, medium) requires a medium-large effort in time or resources to perform (less than 1 day); (3, low) can be accomplished in less than 2 hours.

Using the values from these two variables, we now assign an ordinal value to the vulnerability level according to the 3x3 matrix in Table 5. As can be seen we treat these two variables as equally important. We label the vulnerability-level for ordinal values 1 and 2 as low, 3 and 4 as medium and 5 and 6 as high.

	1	**2**	**3**
1	1	2	3
2	2	4	5
3	3	5	6

Table 5: Vulnerability

Likelihood

The likelihood of fairness and privacy risks does not only depend on the vulnerability of the campaign, but also on how motivated potential adversaries are to actually try and attack the campaign. For example, with regard to fairness, if high value prizes are at stake, especially if there's an opportunity to win multiple prizes, cheaters will be much more motivated than if there's just a high-score ranking between friends. For privacy it depends on how much privacy-sensitive data is collected by the campaign. For motivation we also assign an ordinal value from 1 to 3 (low to high), according to the following cut-offs:

- *Motivation related to fairness*: (1, low) just personal pride (from high-score rankings) or very low-value prizes such as discount coupons; (2, medium) prizes with a value up to EUR 500; (3, high) prizes with a value above EUR 500 or the possibility to win multiple (lower-valued) prizes to a value that exceeds EUR 500.
- *Motivation related to privacy*: (1, low) only limited privacy-sensitive data is collected; (2, medium) full personally identifiable privacy-sensitive information is collected(3, high) valuable full personally identifiable privacy-sensitive information is collected from a large number of users.

These cut-offs for fairness are benefit oriented for a potential adversary and therefore not complete as there could be various other reasons that motivate a potential adversary, such as wanting to cause harm to a certain company without direct personal benefit. As such motivations are highly-specific and impossible to estimate, we will not consider them for our study. Using the value from motivation together with the value from vulnerability, we can now assign an ordinal value to the likelihood according to the 3x6 matrix in Table 6. In this table the top row represents the values for vulnerability and the left column represents the values for motivation. Our position is that motivation as more important than vulnerability.
This is because we think that even if an advergame campaign is relatively well protected, and exploiting its vulnerability requires a significant amount of skill and effort. If the campaign provides enough motivation, eventually skilled enough adversaries will be attracted. This relative importance of motivation over vulnerability is reflected in the assignment of ordinal values in the table. For ease of understanding and communication we label the likelihood for ordinal values 1 to 4 as low, 5 to 8 as medium and 9 to 12 as high.

	1	**2**	**3**	**4**	**5**	**6**
1	1	2	3	4	5	6
2	2	4	6	7	9	10
3	3	6	8	10	11	12

Table 6: Likelihood

Impact

The impact of fairness and privacy threats depends on the amount of mission loss and asset loss they can cause for the owner of an advergame campaign (see sections 2.3 and 2.4). Mission impacts hinder the success of the company's goals and projects and asset impacts cost the company organizing the advergame money or resources. Estimation of the two different impacts is performed using scenario analysis, and will take the context of the campaign into account. As it's impossible for us to estimate impacts in absolute terms, we will assess them in relative terms. For both impact dimensions we assign an ordinal value from 1 to 3 (low to high) according to these cut-offs:

- *Mission loss*: (1, low) small to medium impact expected on results of campaign; (2, medium) significant impact expected on results campaign and/or some damage to brand-image; (3, high) mission loss that reaches beyond the scope and context of the advergame campaign (such as the potential for serious damage to brand-image or goodwill).
- *Asset loss*: (1, low) value of asset loss is relatively small (fraction of cost of campaign); (2, medium) value of asset loss is significant (up to the full cost of the campaign). (3, high) value of asset loss is expected to exceed the total costs of the campaign.

Using the values from these two impact variables, we can now assign an ordinal value to the impact according to the 3x3 matrix in Table 7. In this table the top row represents mission loss and the left column represents asset loss. We argue that mission loss is potentially more serious than asset loss, even though it is harder to quantify. Assets can simply be replaced, where mission loss such as damage to brand-image or reputation can last a long time. This is reflected in the assignment of ordinal values in the matrix. We label the impact for ordinal values 1 to 3 as low, 4 to 6 as medium and 7 to 9 as high for ease of understanding.

	1	2	3
1	1	3	6
2	2	5	8
3	4	7	9

Table 7: Impact

Risk

Now that we have operationalized all concepts on which risk depends in the research model used, it is now possible to determine the overall fairness or privacy risk of an advergame campaign using the 9x12 matrix in Table 8 . In this table the top row represents the likelihood and the left column represents the impact, and both are treated as equally important. We label the overall risk for ordinal values 1 to 17 as low, 18 to 35 as medium and 36 – 53 as high for ease of communication. We can further elaborate this by labeling the bottom five values (1 to 5) as very low, and the top five values (49 – 53) as very high.

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	13	15	16	17	19	20
3	3	6	9	12	14	16	18	20	22	24	26	28
4	4	8	12	15	17	20	23	25	28	29	31	33
5	5	10	14	17	21	24	27	29	32	35	37	39
6	6	12	16	20	24	28	30	33	36	39	42	44
7	7	13	18	23	27	30	34	38	40	43	45	48
8	8	15	20	25	29	33	38	41	44	46	49	51
9	9	16	22	28	32	36	40	44	47	50	52	53

Table 8: Overall risk

3.2.3 Research process

In order to analyze each case in the same structured and systematic way, and to make the research process repeatable, a detailed description for each step in the research process is provided here. A uniform format for reporting case results has been made that follows these steps. An overview of the analysis steps performed for each case during the second research stage can be found in Table 9.

Research step	**Activities and analyses performed**
Step 1: Describe case	Provide a concise description of: • the advergame (client company industry, type of brand, description of game and gameplay, marketing message) • campaign organization and context (competition type, prizes, campaign period)
Step 2: Determine technical characteristics	Determine the technology, architecture and internal workings of the game with a focus on the game-server interactions (communications) and the score function (according to section 3.5). This step involves the following

	activities: • Capturing and analyzing data communicated between game client and server to determine game-server interaction and to find out what game program files and other relevant data is loaded at runtime. • Determine technology used and reverse engineering game program files to uncover architecture and internal workings. • Analyze score function to establish exact game scoring rules and range and pattern of valid scores. Only report on what is essential for understanding the risk assessment performed.
Step 3: Determine motivation	Determine the motivation of potential threat sources with respect to fairness and privacy. Analyze the terms&conditions and identify privacy sensitive information that is collected, then assess value of motivation according to the operationalization in section 3.2.2.
Step 4: Determine vulnerability	This step consists of multiple sub-steps: 1. Identify the controls that are built into the game to protect against possible threats, in the form of specific countermeasures or choices with regard to architecture or game-server interaction. Also look at organization level controls. 2. For each threat identified as a result of the first stage of this research (see section 4.4), assess the vulnerability to this threat for this particular advergame. Identify if vulnerability is reduced or prevented by the implemented controls, and identify if there are possibilities for detection. 3. For any remaining vulnerability, identify how much skill, time and effort is required to successfully exercise that vulnerability. 4. Determine value of vulnerability using Table 5 by assessing values of skill and effort required for the easiest path to exercise a violation of fairness or privacy using operationalization in section 3.2.2.
Step 5: Determine likelihood	Determine value of likelihood using the assessed values for motivation and vulnerability using Table 6. For fairness, use (optional) supportive data to analyze if cheating has indeed occurred.
Step 6: Predict potential impacts	Use scenario analysis (taking into account the contextual factors of the campaign) to predict potential mission and asset loss that could result from the identified vulnerabilities. Assess value of mission and asset loss according to cut-off definitions in section 3.2.2, base assessment on worst-case scenario that is still likely. And then assess the value of the potential impact using Table 7.
Step 7: Determine overall risk	Determine overall risk severity from the assessed values of likelihood and impact according to Table 8.

Table 9: Seven steps of analysis in advergame risk assessment research process.

3.2.4 Limitations

As a direct result of our passive and non-invasive data collection method (see section 3.4) it not possible to collect any server-side data or have any insight in the procedures used by the companies behind an advergame. This has consequences for the completeness of our research for both fairness and privacy.

With regard to fairness, the consequences are relatively small, as most information required can be deduced entirely by analyzing the game client and the client-server interactions. However, one important limitation comes from the fact that we have incomplete overview of what detective controls are taken by an advergame campaign. Most detective controls are positioned on the server-side (both technical and organizational), and we cannot access them. We attempt to work around this limitation by trying to recognize the opportunity for cheat detection during our assessments and assessing the vulnerability accordingly. However the fact that an opportunity for detection exists, does not mean that such detective controls are actually in place and whether they are used effectively. As a consequence, the chance exists that the vulnerability of an advergame campaign is underrated.

With regard to privacy, the limitations are much more serious. The biggest threat to privacy is what happens with privacy-sensitive data on the server, once collected. This is where the privacy sensitive data of all players is brought together. Is this data well protected using secure cryptographic storage? Are the servers the data is stored well protected? Are the authentication mechanisms used to access this data by the organization solid enough? All these questions can not be answered, but do have a major impact on the level of vulnerability with respect to privacy. All that we can assess is that there are no privacy related vulnerabilities in the game client and its game-server interaction, only a small portion of the whole privacy system. As a consequence, the fact that no privacy sensitive vulnerabilities are found in the parts that we can assess, does not mean that no significant privacy-related vulnerabilities (and therefore risks) exist. Therefore a privacy risk (vulnerability and impact) rating will only be provided if a significant privacy vulnerability is found in the case date that we have access to, because otherwise such a risk rating would not be meaningful.

3.3 Solution development methodology (stage III)

The objective for the third stage of this research is:

> To design a control to mitigate one or more of the risks identified.

This design process is executed within the paradigm of design science research and follows the design process from the general methodology as outlined by Peffers et al. (2007). The seven guidelines for design science research in information systems as formulated by Hevner et al. (2004) are also followed. The solution designed is for the most prevalent and urgent risk area as indicated by the results of stage II: fairness and the solution objectives, consisting of a set of requirements, are derived from the threat model developed during stage I. During design and development of the solution special attention is paid to maintaining the design argument and to assert the internal validity of our designed solution using the guidelines from Wieringa (2009). A proof-of-concept prototype has been built to demonstrate, test and validate the developed solution, which is then thoroughly evaluated.

3.4 Data requirements and collection

For this study, as can be seen from Figure 6, two sources of data are required:

1. Data from existing advergame campaigns (primary data) used by stage I and II.
2. Literature to support the theory development (I) and solution design (III) stages (secondary data).

Advergame campaign cases

For the first category, which is used by stages I and II, data needs to be collected from a number of existing advergame campaigns. This number should be large enough for the conclusions drawn from this research to inspire confidence and we have collected data from 16 different existing advergame campaigns (see Appendix B: Cases). Advergame campaigns have been selected to availability according to the following selection criteria:

- The advergame must be a computer-based game (not a real life game); must have a non-deterministic, variable outcome which is not entirely knowledge-based (for example, no quizzes); must be available online (and should therefore not require installation); and must be playable on a regular computer (not just available for mobile devices for example).
- The advergame must specifically target the Dutch market (but is allowed to target other markets as well).

Data collection is performed using non-invasive and passive methods, thus without involvement or intervention from parties connected to one of these advergames. Consequently, this will limit the data that can be collected about these campaigns to data that is publicly accessible, on the advergame campaign website, or elsewhere. This means in particular that we will not have access to data such as:

- Server-side logic and storage. Thus data about server-side components and game session or player information collected at the server-side throughout the campaign, can not be collected.
- Information about procedures used by the organization behind an advergame campaign, other than what's described in the terms & conditions. For example, what procedures are used to manually detect cheating (if any), and how to select winners.

For each selected advergame case the following data is collected:

- All the advergame program files, including external data loaded at runtime that affects game execution (such as configuration files or level-data), but not including any data items loaded at runtime that do not directly affect game execution (such as multimedia files).
- A capture of all data (HTTP(S) and non-HTTP) communicated between game client and server (if any) for at least one game play. This trace is captured using a program called Fiddler[5] for data transmitted over HTTP/HTTPS and using the Wireshark[6] network protocol analyzer for data transmitted using any other protocol.
- Data that provides us with information of the organization and setup of the advergame campaign, such as rules and conditions, privacy policies, how to play information, etc..
- Supportive data gathered from observing the campaign during its live time, such as high score lists, discussions on social media or internet forums, etc.. This data is not systematically collected, and its value is therefore only anecdotal, but can still provide some information about for example the actual occurrence of cheating.

Literature

Sources from literature are used by stages I and III. The general approach used to search and select relevant literature comes from the first three steps according to the method by Wolfswinkel et al. (2011). With regard to scientific literature, we limit literature to papers published in peer-reviewed journals preferrably after 2004 from the Information Systems, Computer Science and Mathematics disciplines. Databases used to search for articles include Scopus, Web of Science and Google Scholar. Of the articles that we deemed relevant, forward and backward citations were checked.

For the first stage the literature used consists of any information that is available to support the analysis and identification of threats. The only criteria for source selection was utility, and thus any sort of non-scientific information was allowed as well (as not much scientific information on the subject is available despite a rigorous literature search). This required all information to be verified for accuracy. Sources used next to scientific literature include software, technology reference and instruction-set manuals; online tutorials and forum discussions, Youtube videos, etc..

For the third stage we looked for potentially applicable solutions to the problem of fairness in advergames by analyzing the existing scientific literature on solutions for cheating in online games. Only scientific literature was searched for and search criteria used include various permutations of the terms: game, cheat, fairness, security, detection and prevention.

3.5 Analysis technique: reverse engineering

For , it is necessary that, in order perform the analyses required to gain understanding in and assess the threats to fairness and privacy in advergame campaigns, we need to have enough insight in how a particular advergame functions, how it is designed and how it is implemented. However we haven't designed or implemented these games ourselves, nor do we have access to any documentation about how they are created, and neither can we inquire with the designers/developers for the purpose of this study. So we have to uncover the design elements by analyzing the artifacts of which the advergame consists as this is all we have access to. This process is called reverse engineering, and is defined as “the process of discovering the technological principles of a device, object, or system through analysis of its structure, function, and operation“ (Eilam & Chikofsky, 2007).There can be many reasons for reverse engineering, often related to maintenance (Elliot & Cross, 1990), but for this study the reason is design recovery with the purpose of

5 http://www.fiddler2.com/fiddler2/ (last accessed: August, 5 2012)
6 http://www.wireshark.org/download.html (last accessed: August, 5 2012)

security/risk auditing. Reverse engineering is the main analysis method used in this research and both research stages strongly depend on it. This justifies that considerable attention is given here to describe how it is performed as part of this research for the technologies used in advergames.

When reverse engineering games for the purpose of this research, we are not concerned with understanding the full workings and component inter-dependencies of the games that are scrutinized. We only have to focus on those components and algorithms that affect the fairness and privacy of a game. What exactly these components and algorithms are for individual games varies case by case, but the following areas are typically of importance:

- The high-level technical architecture of the advergame campaign, especially the distribution of responsibilities between the client-side and the server-side.
- The interaction and communication between the client-side game components and the server, and the code within the game implementing the client side of this communication.
- The score function of the game and the game logic that has influence on it.

General approach

Before an overview is provided of reverse engineering techniques for Flash and JavaScript, the technology-independent first steps of the reverse engineering process are explained first. When analyzing a new advergame campaign, the first step is to determine:

- The technology platform used.
- What (program) files together make up the client-side of the advergame
- What data is communicated to where, during game play and when a score/user is registered.

Several tools exist to do this for a game loaded within a web-browser, such as the free Fiddler program, that have to be attached to the browser process when visiting the advergame website and playing the game. These tools will trace and display any HTTP(S) data exchanged when the game is loaded and played. If there's a suspicion that the game also communicates non-HTTP(S) data, a full network analyzer has to be used that is capable of intercepting any data communicated over the network during game loading and game play, such as Wireshark. The results of these traces can then be analyzed to see what files are downloaded and which of them are part of the advergame program files, what the format of these files is and thus what technology is being used, and what data is being exchanged between the game client and the server and when. Analysis of this data can already provide important clues about the high-level architecture of the game and the distribution of responsibilities between client and server. It also provides a useful indication on where to start with reverse engineering the game program files (by looking for the code implementing the communication, and continuing from there on). We will now look at the specifics of reverse engineering Flash and JavaScript based games.

Reverse engineering Adobe Flash

As discussed in section 1.1.3, Adobe Flash content is distributed using the SWF format, and a game implemented in Flash will consist of one or more .swf files. These files are in a binary format that is difficult to analyze directly, so ideally we would like to turn them into a higher-level representation that is much easier to analyze, such as ActionScript source code. This process of translating a file with a low level of abstraction (such as the computer-readable, but not human readable SWF file) into a a form with a higher level of abstraction (such as ActionScript code which is human readable) is called decompilation, the reverse of compilation. Decompilers exist for SWF files (such as SourceTec Sothink SWF Decompiler[7] or Action Script Viewer[8]) and they do a very good job at reconstructing AS code for both AVM1 (AS2) and AVM2 (AS3), recovering the full source code with the exception of the original names of local variables. This is due to the simple stack-based nature of the instruction set of both virtual machines, which results in clearly recognizable instruction set patterns for all higher level AS (control flow) constructs. In fact these decompilers not just reconstruct the full AS source codes of the original project, but also all of its other source material including shapes, animations, texts, fonts, etc. Consequently, instead of a binary file format, the source code can now be analyzed, which is a lot easier, especially since AS code is typically easy to follow and understand. The source code can then be searched for strings exchanged between the game client

7 http://www.sothink.com/product/flashdecompiler/ (last accessed: August, 5 2012)
8 http://www.buraks.com/asv/ (last accessed: August, 5 2012)

and server, to identify the communication functions, and reversely track the path of execution from there until enough understanding is gained to assess the vulnerability to fairness and privacy threats.

Reverse engineering JavaScript

JavaScript is the scripting language used for games implemented using both DHTML and HTML5 and its files are distributed over the web as plain text files, and not using a binary format. This makes JavaScript code in principle human readable by default, however it might be distributed over many different files and be embedded within HTML documents at several places, making it sometimes a hassle to locate what you're looking for. But in general JavaScript based games are naturally easy to reverse engineer.

Chapter 4: Analytical results (stage I)

This chapter presents the insights and analytical results from the first research stage (see section 3.1 for the methodology used). It starts with a detailed discussion of the two root causes of fairness problems in advergames and then presents the advergame threat model and detailed technical threat descriptions.

4.1 Advergame architectures & fairness implications

Figure 1 in section 1.1.3 depicted the general component structure of games. Although stand-alone advergames can be found, most advergames communicate with a server in a client-server setting. This is for example necessary to register the personal data and score of a player with the advergame website, so that a high score listing can be maintained. Such a basic client/server advergame architecture, where the client is fully responsible for running the game, and the server is only responsible for registering the user's information and (intermediate) game results, can be seen in Figure 8. Please observe that we positioned the game state as a part of the game logic, for reasons of diagram simplification. This architecture introduces the following three new components:

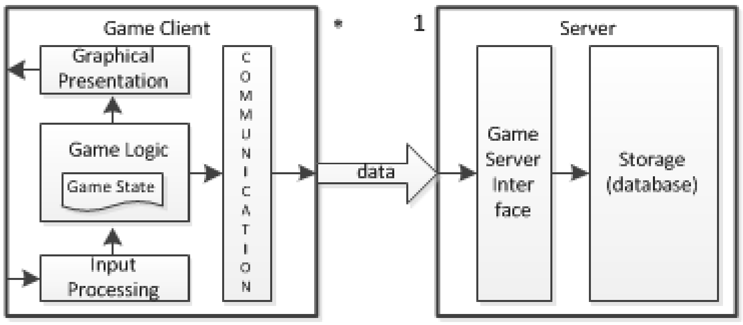

Figure 8: Client/Server architecture with game logic in client

- **Communication (client)**. This component is responsible for communicating with the server. It takes a message to send to the server from the game logic, and will then structure and encode it according to the format expected by the server. If controls that protect the confidentiality or integrity of the message are in place, they will be added to the message here. It will then connect to the server's interface and transfer the message.
- **Game server interface (server)**. This component that is located on the advergame website (or game server, not necessarily the same) is responsible for receiving messages from the game clients of different players and storing the user and score registration information in the game database. It does this by exporting an interface by which a game client can submit a message. It will then decode the message, verify it's integrity (and act accordingly if tampered), and store it in the servers game database.
- **Storage (server)**. This component (often a database) is responsible for storing the registration information and game results of all the players of the advergame.

This simple client-server based architectural arrangement is the easiest to implement and therefore very popular with advergame developers, but positioning the game logic and game state fully within the game client has potentially serious consequences for maintaining the fairness of an advergame. The problem lies in the fact that the two components on whose integrity fairness mostly depends, the game logic and the game state, are fully located within the game client. And the execution of the game client is fully under the control of the player, as this takes place on the players own computer (see Figure 9). This can present a problem to fairness if too much trust is placed by the game developers in the game client, which in fact, can not be trusted at all.

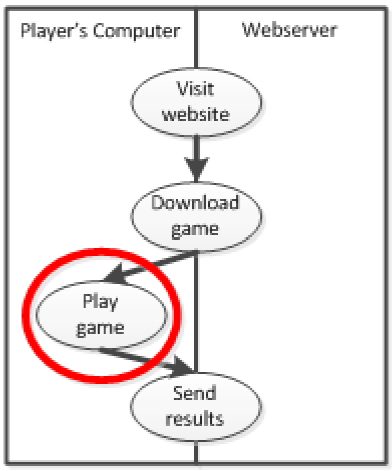

Figure 9: Root cause fairness problem

One way to counter this potential fairness problem could be to use a different architectural client/server arrangement by moving the game logic and the game state components from the game client to the server. The resulting architectural arrangement can be seen in Figure 10. In this arrangement the game client is only responsible for input processing and the graphical representation of the game. Player input is captured, processed and turned into messages by the game client, which are then sent to the game logic on the server through the clients communication component and the servers game interface. The game logic then computes the game transitions and updates the game state accordingly (including possibly the score), and sends graphical update commands back to the game client which are then animated and rendered by its graphical presentation component. When the game is finished, results are directly stored into the database by the game logic and never sent from the client to the server. Since the game logic and game state are no longer under direct control of the player, game results can now be trusted and only input events have to be validated. Even though this architectural arrangement provides very good protection against most fairness threats, it comes with some significant drawbacks which make it less suitable for many advergames:

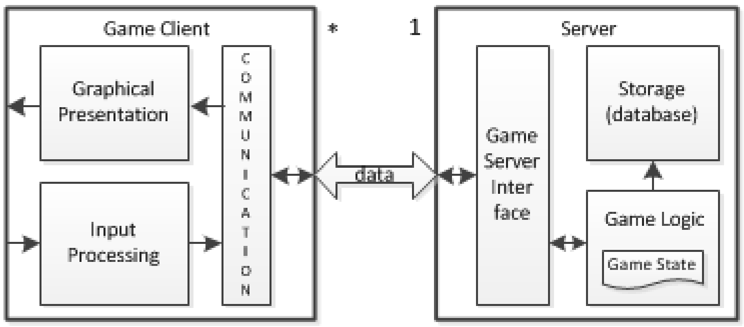

Figure 10: Client/server architecture with logic in server

1. It is generally more difficult to design and implement a game that has its game logic and game state located on the server. Every possible interaction between the input processing / graphical presentation components and the game logic has to be formalized into a message and asynchronously processed (for performance reasons), something that requires a significant shift of thinking for most game developers.
2. Even if a game can be completely designed to have its game logic and game state on the server, network latency and delays might make smooth gameplay impossible. This is especially the case for response- and skill-based games where one has to react quickly to events taking place within the game. For such games locating the game logic fully at the server might simply be unfeasible. This is why in practice most games that use this architectural arrangement are puzzle, simulation or role-playing games that do not require a very quick response time.

Dividing the game logic and game state over both the client and server can provide a solution to the potential drawbacks of using a client/server architecture with the game logic fully located at the server. This architectural arrangement can be seen in Figure 11. In an ideal configuration those parts of the game logic and game state whose integrity directly affects the fairness of the game (such as the score function and important game variables), are located within the server, and those parts that are needed for a quick response time (such as collision detection, movement of game entities and parts of game artificial intelligence) are located within the client. The game client may keep a local working copy of the server game state within the client for performance reasons, but the authoritative version is maintained at the server. Inevitably once a part of the logic and game state is moved back to the client, it is again under full control of the player, and manipulation of these client side components can still indirectly affect game fairness. That's why the server should validate messages arriving from the game client to detect potential fairness violations. In practice this architectural arrangement and how the game logic and game state are divided over the client and server are often a developer's trade-off between guaranteeing fairness on one side and performance, complexity and available development resources at the other side.

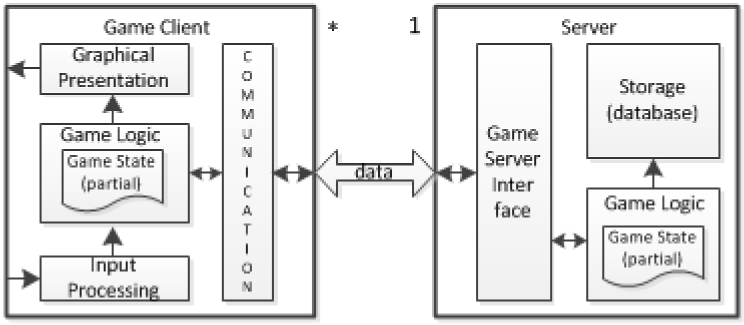

Figure 11: Client/server architecture with logic divided over client and server

In conclusion, the basic client/server advergame architecture, with the game logic and game state fully located within the client, is most vulnerable to fairness violations, as its execution is fully under the control of the player, and therefore its results can not be directly trusted. An architectural arrangement with the game logic and state fully located at the server-side provides good protection against fairness violations, but comes with several drawbacks. Dividing the game logic and state over the client and server can lift some of these drawbacks, but how well it protects the fairness of a game depends on the design and trade-offs made.

4.2 Reverse engineering

Section 3.5 presented reverse engineering as the main analysis technique for this research project, where it was used for design recovery. However, the possibility for reverse engineering can also pose a threat and some of the fairness threats that we've identified and that we will present in section 4.4 depend on such techniques. In this case the attackers reason is to “acquire sensitive data by disassembling and analyzing the design of a system component” (Shirey, 2007). Knowledge of a (partially) recovered design can also be used to perform game client modifications that violate the integrity of the advergame. In fact, assessing the difficulty of performing these tasks are an important part of the risk assessment performed for each advergame. Therefore, the subject of reverse engineering is revisited here, in the context of the threats it can pose to advergames.

As established in section 3.5 it is generally fairly easy to reverse engineer Flash and JavaScript based games. For Flash based games a decompiler can be used that can reconstruct a source code representation of a Flash file and for JavaScript based games the source code can be directly accessed. Once the source code of a game is obtained two general attack methods are possible:

1. The source code can be searched for any sensitive data, such as encryption keys or hash secrets, that are often easy to spot even without programming skills. More advanced adversaries would be able to analyze the source code for any controls implemented, such as hidden interactions, score function patterns, etc. and adjust their path of attack accordingly.
2. Access to the source code makes it much easier to modify the game client in any way an attacker desires. Most decompilers are capable of reconstructing Flash archive files (.fla extention) that can be loaded within Flash authoring software and subsequently recompiled without modification, making this attack easy to perform, even for adversaries with only minor programming skills.

4.2.1 Obfuscation

However, when reverse engineering advergames as part of our analysis it was observed that some games would not decompile to sensible source code. It turned out that many game developers attempt to protect their program files against decompilation by using an obfuscation tool (a protective control). Collberg & Thomborson (2002) define obfuscation as “transforming a program into an equivalent one that is harder to reverse engineer”. It is a form of security by obscurity and it will never provide guaranteed protection against reverse engineering, but in theory it can make reverse engineering much harder, up to a point where it is no longer economical for an adversary (but this of course depends on the value of the code). Collberg & Thomborson define five important properties that good obfuscation transformations should have:

1. They should be **semantics-preserving**, thus the observable behavior of the obfuscated program should be the same as the original program.
2. They should provide **obscurity**, reverse engineering the obfuscated program should be significantly more time consuming than the original
3. They should be **resilient**, meaning that it should be difficult to construct an automatic tool to undo the transformations, or that it would be very resource-intensive
4. They should be **stealth**, it should not be easy to easily identify what predicates are added by the transformations
5. They should have a low **cost**: the execution time/space penalty incurred should be minimized.

Obfuscation software can use various techniques to frustrate reverse engineering:

- **Lexical transformations**, such as identifier scrambling which renames every identifier (variable name, class name, etc..) within the the program to make it much harder to comprehend.
- **Control flow transformations** that break-up the flow of control of a procedure. For example opaque predicates, that are control flow decisions that are known at obfuscation-time but hard for a deobfuscator to deduce, can be used to do such code breakup.
- **Decompiler traps**, such as insertion of misaligned jumps that are never executed, but do prevent decompilers from making an accurate rendition of the code.
- Program **encryption**. The program code can be encrypted in its binary file representation, and be decrypted by a small loader at runtime.

4.2.2 An investigation of Flash & JavaScript obfuscators

During this research several advergames were observed that use obfuscation technology in order to protect against reverse engineering. In order to properly understand the impact of these obfuscators on the vulnerability to reverse engineering for these games, a small investigation of commercial obfuscation tools was performed to analyze the strength of the transformation they perform. Several commercial obfuscation tools exist for Flash that support both AVM1 as AVM2 based SWF files, and Amayeta SWF Encrypt[9], Kindisoft SecureSWF[10] and DoSWF[11] seem to be popular choices.

Amayeta SWF Encrypt

SWF Encrypt claims to "encrypt" your SWF and uses "dense ActionScript obfuscation" using "unique algorithms". A test shows that it does indeed prevents the decompiler used (Sothink SWF Decompiler) from showing the AS code, and in fact, it often crashes the decompiler. Further analysis using the AVM2 disassembler that comes as part of the Tamarin[12] package shows that all the obfuscator does is inserting a few decompiler traps in every method body. These take the form of (for example):

```
pushbyte 1
pushbyte 0
ifne L1        ; jumps to L1 if top two values on the stack are unequal (0 and 1)
…              ; some garbage instructions that trap the decompiler are positioned here.
L1: original instructions
```

Nothing else, certainly no encryption. It would be trivial to automatically detect and remove such traps from the SWF and therefore automatically remove the obfuscation layer. A quick search on the Internet found such a tool for free[13]. Clearly this obfuscator does not satisfy the conditions by Collborg & Thomborson. Even if you don't use this tool, the SWF can still be reverse engineered relatively easily by looking at the assembly listings and searching for certain strings or method names in them. It will only be effective against adversaries that give up after decompiler failure and that don't look further.

Kindisoft SecureSWF

SecureSWF claims to perform several obfuscation techniques that "make decompilation virtually impossible" and "an extraordinarily difficult process". When tested using the Sothink SWF Decompiler, the AS code shown is indeed quite transformed and illegible. However when analyzed with the AVM2 disassembler it shows similar traps to Amayeta, possibly slightly more advanced, and an attempt at opaque predicates, but totally ineffective as the decision outcome can be instantly inferred (the value of variables tested in the opaque predicates are simply assigned at the beginning of a method). Both are very easy to manually spot and remove, and with some effort this can also be done automatically. The most effective feature of this obfuscator is most likely its identifier renaming, which will frustrate the understanding of the code as meaningful variable names are removed. Again, anybody who doesn't give up when a decompiler fails to deliver results right away and who has the skill to use a disassembler, will be able to reverse engineer a SWF file with this obfuscation applied without much trouble.

DoSWF

DoSWF claims to be a "professional encryption tool" for Flash, and is the only obfuscator we tested that actually does what it claims to do. The original SWF is encrypted using their encryption algorithm and packed as a data tag within a new container SWF file. When the container SWF file is loaded at runtime, it will decrypt the encrypted the data, load it as a SWF at runtime and pass control to it. However for experienced computer users it's easy to come up with an effective attack that does not involve reverse engineering the encryption algorithm used by DoSWF (which does not seem too difficult to do) in order to extract and decrypt the original SWF. When the SWF loaded and the original SWF is decrypted and played, the decrypted SWF should be contained within the the Flash Player's process memory. By attaching a debugger to this process and searching through memory to find the SWF file signature, we have been able to

9 http://www.amayeta.com (last accessed: August, 5 2012)
10 http://www.kindisoft.com (last accessed: August, 5 2012)
11 http://www.doswf.com (last accessed: August, 5 2012)
12 http://developer.mozilla.org/en/Tamarin (last accessed: August, 5 2012)
13 http://www.swfdecrypt.com (last accessed: August, 5 2012)

locate the decrypted SWF every time. Extracting it from the process' memory and saving it to a file was enough to completely bypass the obfuscation applied.

JavaScript obfuscation

Obfuscation software also exists for JavaScript and we've encountered one game that uses it. Obfuscation transformation for JavaScript have to be source code transformations, and cannot be binary code transformations, which are potentially more powerful. Most of them work by identifier renaming and making the code less readable by removing all white space space or by encoding the JavaScript code and decoding it at runtime (often as simple as turning it into a hexadecimal representation). It is easy to understand that these transformations are not very effective and will only slightly hinder dedicated reverse engineers. In fact powerful JavaScript deobfuscators are widely available online (often also written in JavaScript), such as JSBeautifier[14], which worked very effectively against the obfuscated JavaScript source code of the single advergame campaign that used this, that we encountered.

4.2.3 Conclusion

At this point it can be concluded that reverse engineering Flash and JavaScript based advergames is generally very easy to do, for example with decompilers that produce almost perfect reconstructions of the original source code code. We've observed that some games rely on the use obfuscation in order to attempt to frustrate reverse engineering, however these obfuscators are very weak, as they do not satisfy any of the properties defined by Collberg & Thomborson, and are therefore easy to remove or bypass. Sometimes the use of a Flash disassembler may be needed to reverse engineer the game using its assembly code. If you're goal is to just search for a hash secret or encryption key used by the game, this often doesn't take much time or effort at all even when assembly code is all that is available. In conclusion, reverse engineering advergames is practically always possible, even if protective anti-reverse engineering controls are applied.

4.3 Threat model

This section presents a model for advergame fairness and privacy threats (see Figure 12). We've decided to create a new categorization, different from the taxonomy of cheating in online games presented by Yan & Randell (2009), because we feel that many of the cheat categories defined in their categorization are insufficiently operational for the context of advergames, as they are too broad to provide enough insight and understanding into the problem. This is because Yan & Randell's taxonomy tries to categorize any possible cheat in the whole eco-system of online games, where advergames typically belong to a specific niche within this universe. We propose a model that categorizes threats based on what particular area of the client/server based architecture of the advergame they target, as we think that such a categorization provides advergame developers with the highest amount of insight in how different weaknesses can affect their campaign's fairness and privacy.

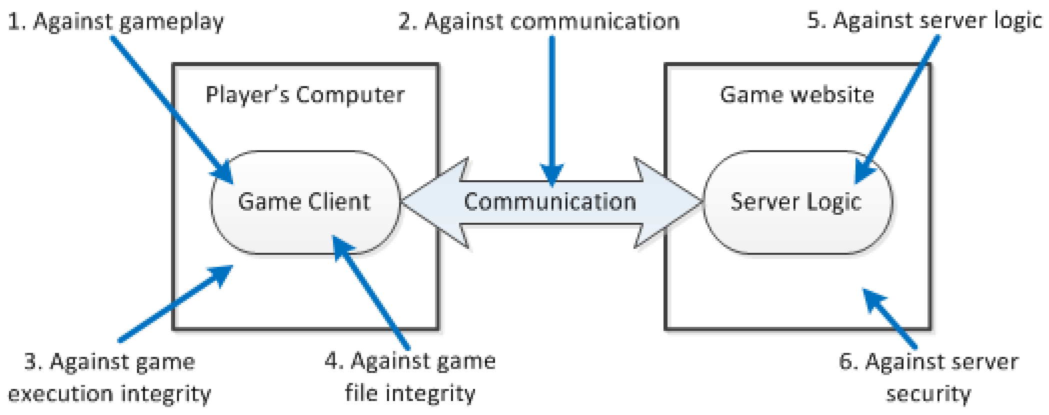

Figure 12: A model categorizing threats according to area of weakness targeted

14 http://www.jsbeautifier.org (last accessed: August 5, 2012)

For this categorization we've defined the following six categories:

1. **Against gameplay**. Threats in this category target the gameplay of the game without modifying or bypassing the game client. Due to the relatively simple gameplay of many advergames software-based scripts or agents might be able to automatically play (parts) of your game using automation techniques or artificial intelligence leading to a significant advantage over human players. Next to that, glitches in the gameplay can sometimes lead to significant advantages for those who know.
2. **Against communication**. Threats in this category direct target the communication between the game client and the server. By modifying data communicated between the game client and the server (such the submission of game results) cheaters can often directly influence the results of a game. Additionally during communication privacy sensitive information can be leaked or might be intercepted by others.
3. **Against game execution integrity**. Threats in this category target the execution integrity of the game, that is, they target the runtime representation of the game on the players computer, and not the integrity of the static files the game consists of. Since the execution environment of the game client is under full control of the cheater he might be able to change the speed of a game or directly modify important variables in the game state of the game through memory or a debugger.
4. **Against game file integrity**. Threats in this category target the integrity of the files the game consists of, by modifying them in such a way that a cheater gains an advantage over other players, or by replacing them all together with a script.
5. **Against server logic**. Threats in this category target the server logic directly through its game server interface. Insufficient validation of interface calls may allow a cheater to influence his game results, or to obtain information that he can only retrieve in this way. Weak player authentication may allow adversaries to access or modify privacy sensitive information or affect the results of others.
6. **Against server security**. A cheater/adversary might be able to take advantage of programming mistakes and security holes in the server logic or software running on the server, to obtain direct access to the database with score entries and privacy sensitive information of other players, as well as other sensitive assets. This is the only attack that we list that is actually punishable by criminal law in The Netherlands. This category is considered out of scope for this thesis.

Table 10 provides an overview of all fairness and privacy threats identified during our research as well as their classification according to our categorization.

No.	**Category**	**Name**	**Type**	**Threat-level**
1.1	Gameplay	Automation cheats	Fairness	Medium
1.2	Gameplay	Artificial Intelligence / Bot play	Fairness	Low
1.3	Gameplay	Game glitches / Emergent gameplay	Fairness	Low
2.1	Communication	Outgoing communication tampering	Fairness	High
2.2	Communication	Leaking privacy sensitive information	Privacy	Medium
2.3	Communication	Incoming communication tampering	Fairness	Low
2.4	Communication	Insufficient transport level security	Privacy	Low
3.1	Execution integrity	Speedhack	Fairness	High
3.2	Execution integrity	Memory manipulation	Fairness	High
3.3	Execution integrity	Debugger manipulation	Fairness	Medium
4.1	File integrity	Game logic modification using source code	Fairness	Medium
4.2	File integrity	Replace game client with script	Fairness	Medium
4.3	File integrity	Simulation against detection	Fairness	Low
4.4	File integrity	Game logic modification without source code	Fairness	Low
4.5	File integrity	Non-logic game modifications	Fairness	Low

5.1	Server logic	Weak player authentication	Fairness & Privacy	Medium
5.2	Server logic	Interface manipulation	Fairness	Low
6.1	Server security	Server security breach	Fairness & Privacy	High

Table 10: List of all fairness and privacy threats identified during this study

4.4 Detailed threat descriptions

In this section we provide detailed technical descriptions and discussions of the all the potential threats against fairness and privacy we identified during this study. Most of the detailed descriptions haven't been filled in yet, but all threats we identified are listed.

4.4.1 Threats against gameplay

Threats in this category target the gameplay of the game without modifying or bypassing the game client.

No. 1.1	**Name**: Automation cheats	**Type:** Fairness	**Threat:** Medium
Description: The gameplay of many advergames is based on relatively simple skill-based tasks that do not depend on decision-making or strategy, but on correct task execution and speed. Examples of these include using the mouse or keyboard to correctly reproduce onscreen patterns on time, responding as quickly as possible to onscreen indicators using mouse clicks or key presses, achieving the highest number of clicks or correctly formed movement/key patterns within a certain time-frame, etc. There are many possible variations, but what have all in common is that they combine repetitive elements and speed. Such elements are often relatively easy to automate using keyboard or mouse scripting software that can help the player to achieve an excellent performance without having to be able to do it himself.			
When vulnerable? Any game that mostly relies on simple (repetitive) skill-based aspects for its gameplay without requiring the player to make (strategic) decisions during game play that are important to its outcome, is potentially vulnerable. The exact vulnerability then depends on how easy or difficult it then is to actually create such automated scripts.			
How to perform: Key-press or mouse-click recording functionality can be used to automatically generate a script for the game, which then only has to be adapted and replayed by the user to achieve a better score (this can be as easy as just changing the delay-timings between clicks or key-presses). More advanced cheaters might be able to use the often advanced scripting possibilities (including support for recognizing visual items on the screen to trigger actions) to automate even less trivial skill-based games. Many software packages exist that can be used for this, such as AutoHotKey[15] (free), WinAutomation[16] and AutomationBox[17].			
Threat-level: The basic version of this cheat is relatively easy to perform and does not require much computer skill or any programming skills, as scripts can simply be recorded. More advanced versions of this cheat might require some clever thinking to detect the possibility, some programming skills and a time investment to make it work. Some awareness of this kind of cheat exists in the form of Youtube video demos. This cheat typically allows a cheater to significantly enhance his performance, but will typically not allow him to achieve any result he wants. Therefore we rate the threat-level as medium.			
Examples: During our study we found one advergame that is highly susceptible to this attack (case #2). In this game you have to click as quickly as possible on onscreen click patterns as they appear, and later in the game, simply click on an area as often as you can within a certain time-frame. Making an automation script for this game would be relatively easy to do. The sumo game, one of the two games in case #7, and the games in cases #1 and #8 are also potentially vulnerable, but writing a successful automation script for these games requires more effort.			

15 http://www.autohotkey.com (last accessed: August 5, 2012)
16 http://www.winautomation.com (last accessed: August 5, 2012)
17 http://www.automationbox.com (last accessed: August 5, 2012)

No. 1.2	**Name**: Artificial Intelligence / Bot play	**Type:** Fairness	**Threat:** Low
Description: Many games are based on concepts that rely on the player's ability to devise a strategy, to plan, to solve (a puzzle) or to memorize, often in a timed manner. Sometimes it is possible to use artificial intelligence based agents to automatically play or solve them (botting or bot play) or to assist the player in playing/solving the game (Schluessler et al., 2007). Computer agents have almost unlimited resources for game event memorization and can often computationally plan many steps ahead to devise the best strategy. Some games even have an optimal game play strategy that is usually not humanly possible to achieve (such as memorizing all previous cards in a deck for black jack game), but trivial for AI agents. This can lead to significant advantages over regular players and often even in superhuman performance.			
When vulnerable? Games built around well-known existing strategy, planning or memorization games are most vulnerable as often AI-based agents for such games have already been developed and can be made to work with your advergame.			
How to perform: Existing AI-based bots can often be found on the internet for various different well-known existing games. Sometimes they have to be adapted to work with a new game to recognize its different visual elements. For games for which no existing AI-agent exists yet, programming one can sometimes be much easier than you would expect if it's easy to automatically recognize the game's visual elements. If the game architecture is such that the game state and game logic is at the server side, it might also be possible to program an AI-based agent that operates directly with the game's server-side interface, bypassing the graphical layer, this is a composite attack that also involves replacing the original game client with a script that interacts directly with the server (see threat no. 4.2).			
Threat-level: This attack, if successfully performed, can have a serious impact on the fairness of a game and numerous videos on Youtube exist that demonstrate the possibility, thus some public awareness exists. The attack might be very easy to perform if an AI-based agent already exists for the type of game that you want to cheat (that is publicly available, which is often the case for game classics), and if it works with the game's visual elements without adaptation (less often the case as it's unlikely to be exactly the same game version as the existing AI-based agent was made for). Having to adapt an existing agent or to create a new one often requires (significant) programming skills, but more importantly it often requires a significant time investment, making the cheat less attractive to perform. Therefore we rate the threat-level as low.			
Examples: The concentration games (memory) we encountered in our study (cases #4, #5, #6 and #15) are potentially easily susceptible to using an AI-based solver. Bots for this game can be found on the internet and can most likely be made to work with these specific versions (if they don't already work out of the box). The same applies for the classic Tetris game we found in case #3, for which numerous agents can be found on the web.			

No. 1.3	**Name**: Game glitches / Emergent gameplay	**Type:** Fairness	**Threat:** Low
Description: Unintentional bugs or glitches that exist in the game, where the game continues to work normally when triggered, can have unexpected or unforeseen results that can sometimes be to the advantage of a player (Bainbridge & Bainbridge, 2007). Players who discover such glitches might devise a strategy around them to maximize their personal advantage, and sometimes also at the cost of other players. This is a form of emergent gameplay (Lewis et al., 2010).			
When vulnerable? There are no specific conditions for vulnerability other than that systematic and extensive testing of the game can reduce the occurrence of such glitches.			
How to perform: Depends on the specific glitch.			
Threat-level: This using such glitches to your advantage is typically not very difficult, but they are often hard to detect and to recognize as an opportunity for advantage. Existence of such glitches in advergames is generally low as advergames are often relatively simple games with a simple gameplay where such glitches are more easily found during game testing, as opposed to regular video games that contain many complex interactions and therefore more opportunity for glitches to remain undetected during development. Therefore we rate the threat-level as low.			

Examples: We've encountered such a glitch in the game of case #16. When the game is playing but left out of focus (window minimized), the game arena will eventually clutter with hundreds of moving teeth (that are spawned over time). At which point it becomes almost impossible to miss anymore, and you can pretty much freely shoot to gain points, without the risk of missing and losing lives. If the game is left in focus (which is usual when a player plays a game, the number of teeth that can be spawned seems to be restricted).

4.4.2 Threats against game communication

Threats in this category target the communication between the game client and the advergame server.

No. 2.1	**Name**: Outgoing communication tampering	**Type:** Fairness	**Threat:** High
Description: With most advergames data that is relevant to game outcome is communicated to the server during gameplay. What exactly is communicated is different from game to game, and for example depends on the architectural setup of the game. For games where most of the game logic resides on the server, data communicated must include the game actions taken by the user that have to be processed by the server to update the game state, but does not typically include the final game results as they are computed on the server. But for games where the game logic is implemented within the game client (which is the case for most advergames), in order for the game to register the (intermediate) game results, such as the final score, with the server, this data has to be communicated from the client to the server. This data can be tampered with by the player during communication, and a player might for example be able to change the original score submitted to the server to any value he wants. Even when security measures are put in place to protect the integrity of the client-server communication, such as a cryptographic hash with a secret value, or by using symmetric key encryption, communication might be successfully tampered with. This is because of the fact that the secret key must reside somewhere within the game client on the client's computer, and can therefore for example be retrieved by reverse engineering the game client. And reverse engineering with the goal of just obtaining a secret value is typically much easier to perform than reverse engineering with the goal of modification. Once this secret value is obtained, the hash can now be recomputed to match the updated data communicated, or this data can be successfully decrypted, modified and re-encrypted. But often the measures taken to protect the communication are so weak and naive or incorrectly implemented that obtaining the secret value through reverse engineering isn't even necessary, as the protective measures can be directly attacked. Instead of proper symmetric key encryption some kind of encoding scheme is often used and assumed to be sufficient, such as base64, uuencode or even rot13-encoding, or by using for example the binary encoding scheme used for Flash Remoting: Action Message Format (AMF), all of which can be trivially modified. Naive and weak, self-designed hash or checksum functions are often used instead of cryptographically strong ones. And even if cryptographically strong encryption is used, it is sometimes used in the wrong way, making it easy to attack, such as reusing RC4 key streams. Even in the case of an architectural setup where the scoring function does not reside in the game client, and therefore the game result is not directly communicated, cheating through outgoing communication tampering might be possible by modifying intermediate actions or operational variables communicated on which the final game result depends, such as input timings or for example the number of hits or misses.			
When vulnerable? Any advergame that communicates data between the client and the server that directly or indirectly influences the final game result as registered on the server is potentially vulnerable. Actual vulnerability depends on how difficult it is to make meaningful modifications to the data being communicated in such a way that it will lead to a better game result (for example it is very easy to make a meaningful modification to a final game score being submitted, but a lot more difficult to meaningfully modify raw input coordinates of screen clicks captured during game play that are submitted to the server game logic for processing), and on the ability to detect such modifications (either by integrity protecting measures or by (automatic) analysis). The strength of such integrity protecting measures depends on the cryptographic quality and on the difficulty of obtaining the secret value that they depend on.			

How to perform: This cheat is performed by using tools to intercept and modify data communicated between the browser and the server in real-time. One of the most popular of such tools is TamperData[18] a Firefox browser plugin that enables you to intercept browser HTTP/HTTPS POST requests and allows you to change the contents of any data submitted. Similar plugins are available to other browsers such as Chrome. Alternatively one can use a HTTP debugging proxy server through which requests can be intercepted and modified, such as Fiddler2. Programs such as Fiddler2 offer an advanced scripting interface which can be used to write scripts that modify certain HTTP requests fully automatically, without user intervention. This allows for powerful attacks using outgoing communication tampering that can defeat many detection mechanisms that rely on the timely submission on many intermediate results. In most cases, that isn't necessary and a tool like TamperData is powerful enough. If integrity protecting measures are in place, reverse engineering might be necessary to obtain the required secret value, but this is often simple to do (see section 4.2 about reverse engineering), and online MD5 or SHA1 hash forms can for example be used to compute a new, valid hash value.

Threat-level: This is one of the most often performed cheats in practice, and most likely the first one any potential cheater will try on your game. Tons of videos on Youtube show how to perform this cheat using TamperData or other tools for plain-text score submissions and more advanced video tutorials show how one can find a secret key using for example a Flash decompiler and how to recompute a hash, reducing the amount of skill required to beginner's level. This cheat results in full violation of fairness as a cheater can typically submit any score he wants. If no integrity protection is in place the cheat is easier to perform than any other cheat and if your data is only being protected by a secret-value based hash stored in your game client without obfuscation, it's only marginally harder to perform. Therefore we rate the threat-level as high.

Examples: Of the games we analyzed in our study, all games that have their game logic located at the client (that is 13 of the 15 analyzed) are potentially vulnerable. Most of them take protective measures by adding a hash to the data communicated to protect its integrity (case #12 is the only one without integrity protection), but in most cases the hash can be easily obtained by reverse engineering the game client.

No. 2.2	**Name**: Leaking privacy sensitive information	**Type:** Privacy	**Threat:** Medium

Description: Due to the interactive and often social experience that advergames try to create for their players, information about other players is frequently sent from the server to the game client to make this possible, for example high score data. Care has to be taken that these game/server interactions, at the communication level, do not (accidentally) exchange or contain privacy sensitive information of other players that is not intended to be disclosed, even if it will not be directly presented within the game. If this does occur, the game interactions contain a leakage of privacy sensitive information and even though this happens under the surface, interested adversaries might be able to look at the communication level of the game and harvest such data.

When vulnerable? A game whose game/server interactions exchange privacy sensitive data of players other than that of the player currently playing, that is not intended to be presented within the game environment.

How to perform: An interested adversary could use a protocol analyzer to capture the communication between the game and the server when he plays the game (or initiates the specific interaction), and is then able to extract the privacy sensitive information of other users that he's interested in. Depending on the specific vulnerability the adversary might be able to initiate many of such game interactions in order to obtain privacy sensitive information of as many users as possible. A more advanced adversary could simply write a script that directly interacts with the game-server to initiate such interactions and harvest the privacy sensitive information it leaks.

Threat-level: If such a vulnerability exists it's generally easily detected as many players (potential cheaters for example, but they might not be interested) will be looking at what data is being communicated between the game client and the server, but general awareness of such privacy risks is generally low. Performing the attack is generally not very hard, but does require some knowledge on how to use specific tools such as a

18 http://tamperdata.mozdev.org (last accessed: August 5, 2012)

packet sniffer and some insight in how to trigger the leaky interactions. Writing a script to automatically perform the attack requires some programming skills. Therefore we rate the threat-level as medium.

Examples: The advergame in case #1 contains a very serious example of such a privacy sensitive information leak. When a player opens the highscore screen within the game, the game requests the highscores from the server, but the server response does not only contain the name and score of users, but their full registration information. Even though this isn't displayed within the game score screen, it is still communicated. In this specific example an adversary is able to obtain the registration information of all users by browsing through the entire high-score table and capturing the communication between the game client and the server.

No. 2.3	**Name**: Incoming communication tampering	**Type:** Fairness	**Threat:** Low

Description: Not only data that is communicated towards the server can be tampered with, game clients often rely on data that is communicated from the server side interface or dynamically loaded from the server for their execution, that may well affect the outcome of the game. For example games sometimes dynamically load configuration data from the server that may contain important game parameters such as the number of points obtained per certain game event, or the maximum playtime and sometimes even entire level definitions. Modifying such parameters before they are loaded by the game can give a cheater a significant advantage. Also for games with an architecture where the game logic mostly resides on the server, the server may send game events or game actions to the game client, which a cheater can then again modify to his advantage, and if the server side logic doesn't verify that the input coming from the game client corresponds to the game events and actions it sent out before, this may go undetected as well.

When vulnerable? Any game whose runtime execution and game outcome depends on data dynamically loaded from the server logic is potentially vulnerable.

How to perform: This cheat is typically performed by using a HTTP debugging proxy server such as Fiddler2, in between the game client and the server that can be scripted to automatically change data according to certain rules in real time. Sometimes this can be as simple as dynamically redirecting the loading of configuration data from the server logic to a file on disk.

Threat-level: Even though this cheat can be relatively easy to perform when a game is vulnerable, it's not very common as it's less well-known than some other cheats and most games do not dynamically load data from the server that directly influences its game execution to start with. More advanced versions of the cheat may require scripting skills and significant insight in how data communicated from the server to the game client may affect game execution. Therefore we rate the threat-level as low.

Examples: No examples of advergames vulnerable to this technique have been observed during our study.

No. 2.4	**Name**: Insufficient transport level security	**Type:** Privacy	**Threat:** Low

Description: Many advergames require players to enter potentially privacy sensitive information when participating or when registering their game results. This data is then communicated to the server, so that it can be stored. If this data is communicated with insufficient transport level security, thus not over HTTPS, its confidentiality and integrity can not be guaranteed and it might be eavesdropped on, intercepted or modified by others, violating the users privacy. This is especially likely to happen when the game is played over an unencrypted Wi-Fi network, something that is very common nowadays.

When vulnerable? Any game that does not communicate privacy sensitive data to the server over HTTPS is vulnerable. Using your own game-level encryption to communicate data to the server is typically not a full solution as this often relies on symmetric key cryptography and the keys for this have to be stored within the game client, and as such can be reverse engineered by dedicated adversaries.

How to perform: An eavesdropper might have setup a network sniffer on the players network (only affecting players from that network), the advergame servers network (affecting all players for that advergame and therefore much more serious) or anywhere in between.

Threat-level: This vulnerability is very easy to detect, but in order to specifically target an eavesdropping attack against an advergame campaign, the advergame server's network (or that of the hosting company) needs to be compromised so that a packet sniffer can be installed on that specific network. This is a specialized and high-profile attack that is not so likely to be specifically targeted against an advergame campaign (it makes more sense to target the advergame server's database directly in that case). In practice most eavesdropping will not be specifically targeted against the advergame, but occur as by-catch on public networks that have an active network sniffer. Therefore the threat-level is low.
Examples: Of the games we analyzed in our study, 9 out of 15 games do not transmit privacy sensitive data over HTTPS. One of these 9 campaigns, case #8, uses its own encryption function to safeguard the data, but as mentioned before, this does not actually provide full protection.

4.4.3 Threats against game execution integrity

Threats in this category target the execution integrity of the game, that is, they target the runtime representation of the game on the players computer, and not the integrity of the static files the game consists of.

No. 3.1	**Name**: Speedhack	**Type:** Fairness	**Threat:** High
Description: Time often plays an important and deciding role in games, for example achieving the best time might be the games objective, or there might be a time bonus for quick completion. The problem is that time measurements on a players computer can not be trusted. Since a cheater fully controls the environment in which the client executes he might be able to influence the perception of time by the game to make it run much faster or much slower. This cheat is called speedhack, and it allows cheaters to speed up or slow down the execution by a factor of their choice. Even if the score does not directly depend on the time, it will certainly do so indirectly, as a game that is difficult to play might be a lot easier when everything moves much slower. The cheat is accomplished by using cheating software that attaches itself to the game process and then hooks and intercepts time related system/API calls to return a speed factor adjusted value.			
When vulnerable? Any game that relies on time measurements done at the client computer is potentially vulnerable. This especially applies to games that use an architecture where the game logic is located in the game client (as opposed to the server). The exact vulnerability depends on to what extent the server takes measures to detect speedhack cheats by doing server side time measurements and comparing those with time and score submissions from the client.			
How to perform: This cheat is performed by using external software such as the free open-source CheatEngine[19]. Such program will abstract all the technical details away. One simply selects the process (or window) for which the speedhack has to apply, fills in the time factor (for example 0.5x for slowing the game down 50%) and starts the speedhack. This speed factor can be adjusted at any moment, and the game speed will change accordingly. When the user is finished, he can simply close the CheatEngine program, or set the speed factor value back to 1x.			
Threat-level: This cheat is as easy to perform as it seems and it is probably the most common cheat performed on advergames due to this simplicity. A search for speedhack on Youtube comes up with tens of thousands of results including many detailed video tutorials on how to do it, that aim at even the most basic computer users. Just trying the cheat is enough to detect if it works on a particular game. In summary, it's not just very quick and easy to perform, it also requires no skills or knowledge, but it does give cheaters a very significant advantage over non-cheaters, therefore we rate the threat-level as high.			
Examples: Of the games we analyzed in our study, all games that have their game logic located at the client (that is 13 of the 15 analyzed) are vulnerable. However three of them potentially take countermeasures with the opportunity for detection by server-side time measurement. This leaves 10 out of 15 advergames at full risk, an incredible number considering how well-known speedhack is.			

19 http://www.cheatengine.org (last accessed: August, 5 2012)

No. 3.2	**Name**: Memory manipulation	**Type:** Fairness	**Threat:** High

Description: During game play a game has to keep account of many important variables that determine the coarse of the game, collectively called the game state. Some of these variables directly determine or influence the score or outcome of a game, such as the current score or the number of lives left. If the game state is located on the client's computer (as opposed to on the server) it is under his influence, which means that in theory he can change the contents of any of these variables. A player can then search through the memory of the process of the game he plays to localize and modify important game variables. The technique of using this to your advantage in games is called game memory manipulation, or sometimes memory hack, and can often be used to obtain any score that one wishes.

When vulnerable? Any game that maintains its game state on the client and that takes no specific countermeasures to make this attack more difficult, but that does put trust in client-side results, is vulnerable. Score-based games are typically more vulnerable than time-based games (where you have to achieve the fastest time). Even games that do take measures to protect certain game critical variables against manipulation (which usually means, making them impossible to find in the process memory), might be vulnerable to some extent if they oversee variables that indirectly affect the score. Games that maintain the game state on the server are typically not vulnerable. Also this cheat is difficult (if not impossible) to detect at the server-side if no specific measures for detection are taken.

How to perform: This cheat can be performed using CheatEngine as well, but other programs exist. You start by attaching this program to your game process. Now suppose you want to change the game score, currently at 1355 points to a value of you liking, you start by doing an initial memory search for the value 1355 (you have the option to select what data type and representation, but let's assume the default 4 byte integer type works). This search might show up many different locations in that process' memory that contain this value. So you continue with your game till your score changes to another value, say 1365. Now you can perform a "next scan" for the updated value of your score, and any location that formerly had the value 1355 and now has the value 1365 is shown. Several such iterations might have to take place before it is narrowed down to exactly one location, which you can at that point update to any value you want, and see this reflected within the game. Several more advanced scanning options exist that make localizing the variable you're looking for easier, such as searching for a range or values that increased or decreased, etc..

Threat-level: Even though this threat might initially seem difficult to perform, it is actually very easy if no counter-measures are taken. Again thousands of videos on Youtube show you how it's done, you don't even have to understand what it is that you're actually doing. If successfully performed, a cheater can typically change the game score to any value he wants, therefore fully violating the game's fairness. In practice it's just slightly more difficult and time consuming to perform than a speedhack, but it still requires very few skills or computer knowledge. Therefore we also rate this threat as high.

Examples: Of the advergames we examined during our study, we found six of them to be highly vulnerable to memory manipulation without significant opportunities for detection. Several more were vulnerable in theory but performing the cheat might turn out to be relatively difficult, and easier methods for cheating for those games existed. For example in the highly vulnerable advergame case #8, performing this cheat might save you hours of laborious and monotonous game play.

No. 3.3	**Name**: Debugger manipulation	**Type:** Fairness	**Threat:** Medium

Description: Similarly to the memory manipulation technique mentioned above important game variables can also be changed during game execution using a debugger. A debugger can be used to examine and change the execution of a program in real time. One of the core features of any debugger is to query and change the runtime values of variables during execution. A cheater can use this functionality to change the value of important game variables such as the score to any value he wishes. More advanced features that

can be used by cheaters include setting a breakpoint at any place in the game code (for example just before the score is being submitted, so that it can be modified), or actually changing the code of the program under execution. This is especially applicable to JavaScript and HTML5 games as they are executed directly by the browser process and as such they can always be debugged right away from the browser window in which the game executes, using the browsers built-in (or plugin-provided) JavaScript debugging functionality. Using a debugger on Flash games is not always possible, as they have to be compiled with debugging support enabled, which is typically not the case. And even if they are, debugging them in real time is a lot less convenient than with JavaScript/HTML5 based games as they requires external debugging software that has to be connected to the Flash player (a special debugging version) that executes the game.
When vulnerable? Any JavaScript- / HTML5-based game can be debugged using the browsers debugger, and is therefore potentially vulnerable. The actual extent of the vulnerability depends on what game state variables are maintained by the game client and how changing them potentially affects the final game result. Flash-based games are in theory only vulnerable if they are distributed with debugging support enabled.
How to perform: The newest versions of all major browsers ship with JavaScript/DOM debugging functionality included such as Web Console[20] and Scatchpad[21] in Firefox, Dragonfly[22] in Opera and the Developer Tools[23] in Chrome. More powerful and advanced functionality can sometimes be installed as an add-on such as the very popular Firebug for Firefox[24]. All of these present a debugging console through which the execution of the game can be examined and changed and which allow JavaScript code to be directly executed.
Threat-level: This technique is relatively simple to perform for games that are vulnerable to it as it can be as simple as querying and changing a single variable, but more advanced versions might require debugging skills and being able to read and understand JavaScript code. The technique still seems relatively unknown, but more and more instructional videos on Youtube are appearing how to cheat specific games using this technique. If successfully performed, it typically results in full violation of fairness. With the predicted shift from Flash to HTML5 games, this technique will likely become more widely used, as every HTML5 game is potentially susceptible. Therefore we rate the threat-level as medium.
Examples: In our study we found 5 games that are based on JavaScript. However the games from cases #6 and #10 are not vulnerable to this cheat as they do not maintain any useful game state variables on the client. The other three: cases #4, #12 and #14 can be manipulated directly through the browsers debugger in various ways.

4.4.4 Threats against game file integrity

Threats in this category target the integrity of the files the game consists of, by modifying them or by replacing them all together with an alternative game client.

No. 4.1	**Name**: Game logic modification using source code	**Type:** Fairness	**Threat:** Medium
Description: A powerful class of cheats works by modifying the source code of the game client to change the game logic to the advantage of the cheater. The game can then be rebuilt (if necessary) and deployed instead of the original game client. If the player now plays the game, he can take advantage of whatever modifications he made. Obtaining the source code of a game client is often relatively easy using reverse engineering techniques (see section 4.2). Once a cheater has access to the source code anything can be modified and the possibilities are almost unlimited, some examples: • The game can be changed to submit any hard coded score on game over. • Preventive anti-cheating countermeasures can be removed or disabled. • Cheat detection countermeasures that work by keeping track of game progress through validating various game parameters that are sent as intermediate results during the game can be made			

20 http://developer.mozilla.org/en-US/docs/Tools/Web_Console (last accessed: August 5, 2012)
21 http://developer.mozilla.org/en-US/docs/Tools/Scratchpad (last accessed: August 5, 2012)
22 http://http://www.opera.com/dragonfly/ (last access: August 5, 2012)
23 http://developers.google.com/chrome-developer-tools/ (last accessed: August 5, 2012)
24 http://www.getfirebug.com (last accessed: August 5, 2012)

ineffective by subtly changing game behavior that is not validated, but that will make the game much easier to play, such as disabling collision detection with game entities that would otherwise lose you points or lives when touched.

There's typically no way for the server to detect if the game client has been modified (at least, that can not be bypassed).

When vulnerable? Any game that is distributed by its source code (such as JavaScript games) or whose source code can be (easily) obtained through reverse engineering techniques is potentially vulnerable. The actual extent of the vulnerability depends on how much trust is placed in the validity of results submitted by the game client and the difficulty of reverse engineering. Especially client/server based games with its game logic placed (partially) in the game client as opposed to the server are potentially vulnerable (see section 4.1)

How to perform: A cheater can attempt to use the reverse engineering techniques from section 4.2 to obtain the game client's source code. For JavaScript based games the source can be directly obtained from the HTML document or from the JavaScript files loaded, and for Flash games a decompiler can often be used. Once a cheater has his hands on the source code, he can analyze it, looking for pieces of code that give him an advantage when changed. Once the code is modified, it has to be rebuilt for Flash based games, using a Flash compiler (free ones are available, such as the Adobe Flex SDK[25] and therefore the full Adobe Flash Professional authoring program is not required). The modified game then has to be deployed instead of the original one. Two easy methods are often used:

1. Use a HTTP debugging proxy such as Fiddler2 to transparently redirect the location of the original game client to a the modified game client on disk.
2. Deploy the modified game client by replacing the original game client file(s) in the browser cache and reload the page from cache.

Threat-level: This is a potentially very powerful class of cheats that has the ability to bypass various anti-cheating countermeasures. Reverse engineering the game to obtain the source code often doesn't require much time or skill as tools are available and many people will be able to do this (using online video tutorials for example). Using this technique typically results in full fairness violation as the game can be adapted in any possible way. However successfully performing this cheat does require programming (or at least code reading and adapting) skills and insight in how modifications affect the game results and its detectability. Simple versions of the cheat such as submitting a fixed score on game over might require relatively little skill and time to perform, where more advanced versions may require more skill and time to implement and test. Since JavaScript and ActionScript are some of the most widely used scripting languages, in/ practice many people will be able to perform this cheat and therefore we rate its threat-level as medium.

Examples: Out of all 15 advergames in our study, 13 have their game logic located within the game client. Eight of those do not use any form of obfuscation and are therefore highly vulnerable to this technique. Of four out of five games that do use obfuscation, the full source code can be recovered if the obfuscation layer is stripped. Only for case #7 is obtaining a re-compilable source code more difficult due to the game's obfuscation.

No. 4.2	**Name**: Replace game client with script	**Type:** Fairness	**Threat:** Medium

Description: Instead of modifying the original game client, the game client can also be replaced altogether by a script that pretends to be the game client when interacting with the server. Such a script does not have to be executed within the browser, but can stand on its own and can now directly submit (intermediate) results to the server according to the format it expects. It can easily remain undetected by mimicking a regular browser by setting its User-Agent string as such and by including HTTP Referer headers (from the original game client URL location). Communication integrity and confidentiality controls can be recreated within the script (after the necessary secrets have been reverse engineered). Using a script offers powerful opportunities to avoid server side cheating detection. For example if time is measured on the server side and correlated with the score achieved, the script can be programmed to emulate exactly the right timings that

25 http://www.adobe.com/devnet/flex/flex-sdk-download.html (last accessed: August 5, 2012)

correspond to a real player achieving that score. Possibilities are endless. This technique can sometimes be used in a composite attack together with the threat no. 1.2 (artificial intelligence) against games that have their client logic located at the server, to automatically play a game against the server with a better performance than what the cheater would have been able to do by himself.

When vulnerable? Any game that places any trust in (intermediate) results or commands that come from the game client is potentially vulnerable. Actual vulnerability depends on how difficult it is to get the game interaction "right", in order to not get detected (if detective controls are in place).

How to perform: It is necessary to understand the interaction between the game client and the server in order to perform this cheat. This can often be easily monitored using HTTP debugging tools such as Fiddler2. Reverse engineering the game client might be necessary to gain full understanding or to obtain the cryptographic secrets required for communication (see section 4.2 on reverse engineering). Writing a script that directly communicates with the server can then be done in almost any scripting or programming language that the cheater favors. Scripting languages like php, perl, python and visual basic make this process easy by providing powerful libraries for HTTP-based communication. A script that then simply submits a score with the correct hash value and correct timings does not have to take more than a few lines of code.

Threat-level: Performing the basic version of this cheat requires programming skills, but other than that, such scripts can often be quickly created, especially for games with their game logic fully in the client that only submit the game result at the end of the game. But more advanced scripts for games with more complex game/server interactions (especially composite attacks together with AI) might require significantly more skills and resources. Overall there will be enough of people who have enough programming skills in any language to create such a script for almost any given advergame and therefore we rate the threat-level as medium.

Examples: Simply put, every advergame we encountered in our study is to some degree vulnerable to this threat. For 9 out of 15 advergames this is especially easy as the required cryptographic secrets can be easily obtained by reverse engineering and the game/server interaction is very simple with limited to no opportunities for detection. For 3 (case #1, #3 and #9) out of the remaining 6 the only reason why it is more difficult is because of the obfuscation it's a bit harder to obtain the required cryptographic secrets. For the game in case #7, obfuscation makes obtaining the secrets more difficult and more advanced detection by correlating scores to timings is possible (the cheater needs to get this right). For the remaining two advergames (cases #6 and #10) the fact that the game-logic lies at the server makes this attack more difficult, but considering the fact that both games are very simple puzzle based-games, performing a composite attack would still be possible.

No. 4.3	**Name**: Simulation against detection	**Type:** Fairness	**Threat:** Low

Description: This is a more advanced extension of the previous threat that attempts to further minimize detection. The organization behind an advergame could potentially attempt to detect cheaters by analyzing their playing patterns, such as how often they played and what scores they achieved, and compare this with what they think is a natural pattern for a dedicated and enthusiastic player (that somebody hits the highest score on the very first time he plays a new game, is for example rather unlikely). Once a script has been created that can directly interact with the game server can and submits results, it is possible to write a second script (or to integrate in within the original script) that automatically simulates an enthusiastic human player dedicated to win by playing the game frequently and simulating natural playing patterns such as a learning curve for the game with scores improving over time. This will most likely (but unjustly) increase the confidence by the organization in that a potential cheater is actually a genuine player.

When vulnerable? Every game for which a script that replaces the original game client can be made is vulnerable to this threat as well.

How to perform: An algorithm needs to be devised that simulates a dedicated and improving player for a particular advergame. This might be as simple as generating a set of slowly increasing scores (within the valid scoring range of the advergame) and in between timings, but might be more complex for games where the scripts that replaces the original game client needs a complex set of inputs. Such an algorithm can then be implemented using the same programming language as the replacement script.

Threat-level: The difficulty of devising such an simulation algorithm might change from game to game, but at least some programming skills are required. If well implemented, detection using side-channel techniques would be absolutely impossible, resulting in full breach of fairness. Since this is a further extension of the previous threat (and thus requires a script to be developed first), and possibly not very well known, we consider the threat-level to be low.

Examples: Since it is theoretically possible to write a replacement script for any advergame we encountered in our study, it's also possible for every advergame to write a simulation script around it, and for most games it would be quite easy to do so.

No. 4.4	**Name**: Game logic modification without source code	**Type:** Fairness	**Threat:** Low

Description: Sometimes it is not possible to obtain the source code of a game client through reverse engineering (or at least a re-compilable or executable one) for example because of obfuscation. Several other, more advanced techniques for modifying the game logic can still be used. Two general approaches are:

- Instead of using a decompiler, a disassembler might be used to obtain machine instruction listings of the game client's code. While significantly harder to understand, these instruction listings can then be analyzed to uncover fairness critical elements of the game logic, that can then be modified and reassembled into a modified game client.
- Sometimes using a disassember might not (fully) work because of obfuscation traps that result in incorrect assembly listings or disassembling errors. Or disassembly is possible (with errors), but because of these traps, it's not possible to reassemble again. In this case it's still possible to modify the game client by binary patching. Using a hex-editor the structure of the game client files can be manually dissected and instructions can be patched in their binary form. This sounds more complicated than it is, especially if you know what you're looking for. You might for example use a hex-editor to execute a search for the binary representation of a few consecutive instructions that you know are important and patch them when found.

Since it's generally complicated to make complex changes using these approaches (especially for binary patching), modifications are often limited to:

- Disabling code by overwriting the original instructions with NOP instructions that do nothing when executed. For example you might disable the code that lets you lose a life when you a certain game entity hits you.
- Changing conditionals either by changing the corresponding conditional jump instruction to one that always jumps independent of the condition, reversing the condition for the jump.
- Changing simple arithmetic instructions, for example within the score function so that the score is calculated differently.

If none of that is possible you're still left with at least two options (specifically for Flash files):

- You can read out the binary constant tables within the DoABC tag and modify values of which you think that they might affect game to your advantage.
- One particularly advanced attack is as follows: One can inject code into a Flash file by adding a new class to the DoABC tag that contains a static initializer for that class. A static class initializer is called before the main execution of the flash file starts. Within the static initializer you put code that starts a timer that will call a method of your class again in a few seconds, thus when the game has started. Within this method you can now dynamically obtain a runtime representation of all classes, class instantiations (with their class members), and global variables that exist within the game using the built-in ActionScript3 reflections API by recursively querying everything down from the "root" object that every Flash program has. Once you have obtained all this information, you can analyze it to find which runtime objects or variables are of interest, and then inject the game client in a similar way again with code that updates these objects of variables with any value you want.

When vulnerable? Any game is potentially vulnerable, even if powerful obfuscation is applied. The actual extent of the vulnerability depends on how much trust is placed in results submitted by the game client. Especially client/server based games with its game logic placed (partially) in the game client as opposed to the server are potentially vulnerable (see section 4.1 on architecture).

How to perform: Flash disassemblers for AVM1 (for example, Flasm[26]) and AVM2 (such as Tamarin) are freely available and most decompilers also have the option to list assembly listings (which is often more reliable if the source code obfuscations let the decompiler crash). No further instructions are given here.

Threat-level: These threats require a significant amount of skill and intimate knowledge of the SWF file format as well as the AVM1 or AVM2 instruction set. Even though making simple changes in the codes assembly listings if they disassemble and reassembly correct is potentially not too difficult, not many people will be able to perform the more advanced versions of these threats right away. However there exist very vivid communities of people who crack software and games for fun using similar binary techniques that are actually more complex for real executable files than for Flash files. If your campaign attracts their attention, they might well give it a shot, and although threat occurrence is not very likely, it is therefore also not unlikely. Therefore we rate the threat-level as low.

Examples: The advergame in case #7 is a good example of a game where such a technique might be used. Due to SecureSWF obfuscation (which is actually not strong at all), decompilation does not result in re-compilable sources. However disassembling the advergame does result in assembly listings that can be reassembled. Modifications can therefore be made at the assembly level.

No. 4.5	**Name**: Non-logic game modifications	**Type:** Fairness	**Threat:** Low

Description: A game client rarely only contains logic, but often also packs together various other non-logic data items that are necessary for the game, such as multi-media files or data that is used and interpreted by the game logic during gameplay (such as definitions for levels or other game elements, in for example binary or XML representations). This data can also be modified, if unprotected by controls such as encryption, or integrity verification within the game logic. If modification of the game logic is not possible (for example because the game logic is located on the server) or too difficult (for example because of effective client-side obfuscation), it might still be possible to gain an unfair advantage through the modification of these non-logic parts of the game client. It is easy to understand how the modification of game-critical data such as level-definitions or definitions of the graphical representation of in-game enemies and other objects to avoid (for example by reducing their size) can lead to an advantage to cheaters, but even the modification of pure, uninterpreted multimedia data might lead to advantages (such as by enhancing contrast to make the game easier to play).

When vulnerable? Any game that contains unprotected non-logic data whose modification can provide an advantage to a cheater is potentially vulnerable. Actual vulnerability depends on how difficult it is to successfully modify this data and how big the advantage is that a cheater gets.

How to perform: This threat is easy to perform for JavaScript games as they typically consist of separate files that are loaded from the game website when needed, and can therefore be easily downloaded and modified. For Flash games, non-logic data items such as shapes, images, animations, or user defined data are contained in separate tags within the SWF file. Most SWF decompilers can list and extract such data items from SWF files and often even replace them. If a decompiler does not work (for example because it crashes due to obfuscation) various other tools exist (such as SWFTools[27]) that can list the tags and the data they contain within an SWF file and subsequently extract and replace this data or the entire tag, if necessary. SWF-format binary encoded shape or animation data can be prepared using the Adobe Flash Professional authoring tool (or any other authoring tool) after which the relevant tags (after extraction) can then replace the original shape/animation data within the game client SWF file, thus without having to deal with the binary representation of such data.

Threat-level: This is typically an attack that takes some time to prepare and perform, but where the game is sometimes only marginally influenced. It's rarely possible that a cheater can directly influence the score this

26 http://www.nowrap.de/flasm (last accessed: August 5, 2012)
27 http://www.swftools.org (last accessed: August, 5 2012)

way, and the effect is typically only indirect, but can in some cases lead to a significant advantage over other players. Detecting the opportunity for gaining an unfair advantage through non-logic game modifications might be the most difficult part of this attack, as almost each case where it's applicable is unique. Once the opportunity is spotted, successfully performing the attack typically requires some more advanced computer skills (such as being able to create or modify multimedia files), but programming isn't necessary. Therefore we rate this threat as low.

Examples: The memory-based advergame in case #6 is a good example of a game where it is impossible to modify the game-logic (as it's located on the server), but where changing the images used within the game can make gameplay significantly easier. The last level of the memory game is made deliberately difficult by using images that look very much alike, but replacing these images with images that have contrasting colors and patterns, will make this level much easier to play.

4.4.5 Threats against server logic

Threats in this category target parts of the server logic directly through its game server interface but do not actually breach the security of the server by exploiting bugs in the server logic.

No. 5.1	**Name**: Weak player authentication	**Type:** Fairness & Privacy	**Threat:** Medium

Description: Most advergames require a player to register with some personal information before the game can be played or before submitting a result, but this often happens without requesting the player to create a password protected player account. Instead, when the player's session has expired, the player is re-identified by submitting his details again. Either the player's email-address (most frequently), phone-number, or sometimes even his (nick-)name is then often used as primary identifier for the player. Even though not being requested to register a password-protected account might be convenient for a potential player, such an authentication method is weak and results in the following privacy and fairness problems:

- From a privacy-perspective: Anybody who knows the email-address of another player or who is able to guess it or look it up on the internet, is then able to resubmit the player's personal information and subsequently play for that player. Thus, this allows unauthorized modification of a player's personal information and is therefore a violation of the integrity of this information. If during the game this weak player identifier is also used to load a player's profile or game history, then this may additionally result in the loss of the confidentiality of a player's private information.
- From a fairness-perspective: There are two ways a potential adversary/cheater might abuse this. If a cheater knows or can guess the email-address of a top-scoring player he might be able to update parts of his contact information with his own, hoping that a possible prize might be sent directly to his address instead. Alternatively, using a particularly vicious method to manipulate the game's fairness, a cheater might try to discredit another player by submitting obviously false/fake scores under his name/e-mail address, with the hope of having him disqualified, improving the cheater's own chances.

In a similar way, advergames often integrate with social media websites, and rely on such platforms for player authentication. When a player then plays such an advergame he's asked for permission to share some of his personal information with the advergame, and this information is then sent directly to the advergame server. The social media platform's unique ID or user name is then typically used as the primary identifier for a player, but if these parameters are not properly authenticated as coming from the social media platform (often a HMAC-SHA signature as used for example by Facebook) when sent during a game, adversaries might be able to manually change them for the ID or user name of any other player or social media user. This results in the same privacy and fairness problems as listed above, with the main difference that it's much easier obtain somebodies social media ID or user name (this is considered public information), which makes the attack easier to perform.

When vulnerable? Any game that requires a player to register some personal information for highscore or contacting reasons but that does not require the user to setup a password protected account is potentially vulnerable, as well as any game that uses a social media platform for player authentication, but does not actually authenticate the origin and integrity of user information.

How to perform: All that is necessary to perform this threat is to guess or find the email-address (or anything else that is used to identify a player) of the player whose account you would like to access or manipulate. In case of poor social media authentication, only the users social media ID or name is necessary, and this can for example often be directly obtained from the advergame's highscore ranking.

Threat-level: This attack often requires no technical skills to perform and it is easy to spot when a campaign is vulnerable. Impact is typically low to moderate, but can in some cases be high, therefore we rate this threat as medium.

Examples: Vulnerability to this threat is very common with 9 out of 16 campaign vulnerable. Most often only an email address is required for authentication, which is in many cases trivial to obtain. Some campaigns have insufficient authentication when they integrate with social networks, such as case #4, where only somebody's Facebook ID (and this is public knowledge) is used to authentication users.

No. 5.2	**Name**: Interface manipulation	**Type:** Fairness	**Threat:** Low

Description: A game server typically offers a game server interface through which the game client communicates with the game server. It is often assumed that the game client will be the only entity communicating with this interface according to the interaction-sequences that the developers had in mind. As a consequence the parameters and order of interface calls are not always properly validated, which, if abused by a cheater, may result in an advantage with regard to outcome or knowledge for the cheater. We calls this class of threats interface manipulation, and see it as an extension of what Webb and Soh (2008) define as “invalid commands”. This is especially occurs with advergames that divide the game state and logic over the client and the server with the goal of decreasing trust put in the client and therefore (hopefully) improving the fairness of the game. In such a situation the game server typically maintains an authoritative version of some parts of the game state, and listens for game event messages through the game server interface that may affect the game state. If such interface calls are not properly validated to be in line with permitted or expected game progress, a cheater may still be able to cheat such a game. For example if in a card game a player can only play a certain card once, playing it twice within the same game session should not pass validation, if it does it may lead to a significant advantage for a clever cheater. In another scenario a cheater might gain information that may aid him in performing more successful cheat attempts through creative use of the game server interface, for example some games that give away prizes to the best player do not show an absolute highscore ranking but only provide you with your relative position (with the intention of not giving away what the current high score is to make cheating more difficult). Often the game client queries the server for this relative position using the game interface, by providing the absolute score that the player achieved as parameter. A smart cheater can then create a script that determines the current highscore by repeatedly sending slowly increasing scores to the game server interface until position 1 is returned.

When vulnerable? Any game where the game server interface (or game server logic) does not properly validate the calling parameters and order of interface calls by assuming that they will be only be interacted with as the game developer intended is potentially vulnerable.

How to perform: This attack is typically performed in a composite attack together with threat no. 12 (replace game client with script) so that the order and parameters of interface calls can be specifically programmed to make optimal use of the vulnerability, but sometimes it's also used in a composite attack together with threat no. 11 (game logic modification with source code), where the original game client is modified to take advantage of the lack of validation within the game server interface or game logic.

Threat-level: In the case of advergames, vulnerability to this threat is not very common because the basic client/server based architecture used does not provide much opportunity for vulnerability and when a campaign is vulnerable it typically requires some technical skills and insight to take advantage of and therefore we rate the threat-level as low.

Examples: The advergame in case #11 we encountered during our study is susceptible to this attack. Players are only entitled to play the game as long as they have credits, which can be earned by inviting friends. When a player starts the game it will make an call to the server to check and decrement the number of credits left for that player (identified by his e-mail). Suppressing this call allows the player to never use up his credits but keep playing the game. Additionally, each time the game is played the user earns between 1 and 4 digital lottery tickets, depending on the score, which are used for a prize draw at the end of the game period. The game client makes an interface call to the server with the number of lottery tickets earned as parameter, which can never be more than 4, but it doesn't validate this, allowing the player to generate any number of lottery tickets, significantly enhancing his chances during the prize draw.

4.5 Conclusion

We've now developed a model for fairness and privacy threats that is operationalized for the technological context of advergames, based on the analysis of vulnerabilities in actual advergame campaigns. Threats are categorized according to what part of the game's architecture they target, independent of what kind of condition they exploit, as is the case with existing categorizations. We believe that this model offers a lower level of conceptual abstraction, and a better fit with practice than existing models. Detailed threat descriptions have been developed that aid practitioners in understanding why and how these threats work and help them to assess their own campaigns.

In addition to that, that two supportive theories have been developed on the fairness implications of different architectural arrangements and the role of reverse engineering that provide the foundations for our threat model and help in better understanding the problem domain.

The second research stage tests this threat model in practice by using it to perform a risk assessment on 16 different advergame campaigns which will provide some insight in its validity and usefulness. The results of this risk assessment are reported in the next chapter.

Chapter 5: Empirical results (stage II)

This chapter presents the empirical results from research stage II: a risk assessment of 16 Dutch advergame campaigns (see section 3.2 for methodology). Section 5.1 presents results related to the technology platforms and architectures used, and the success of reverse engineering. Section 5.2 presents results about the prevalence of vulnerability to the 17 different threats identified. Section 5.3 presents the overall risk assessment results for all advergames. Finally, conclusions are drawn in section 5.4.

5.1 Technology, architecture and reverse engineering

This section discusses the results related to the technology platforms and architectures found in use and reverse engineering success. The first 5 columns of Table 11 present the data for these results and the last column, taken from Table 14 in section 5.3, is added to relate these results to the observed level of vulnerability.

Case #	Technology	Architecture	Obfuscation	RE success	Vulnerability
1	Flash (AVM2)	CS/client	Yes	Full source	High (6)
2	Flash (AVM2)	CS/client	No	Full source	High (6)
3	Flash (AVM2)	CS/client	Yes	Full source	Medium (4)
4	JavaScript	CS/client	Yes	Full source	High (6)
5	Flash (AVM1)	CS/client	No	Full source	High (6)
6	JavaScript	CS/server	No	Full source	Low (2)
7	Flash (AVM2)	CS/client	Yes	Partial source	Medium (4)
8	Flash (AVM1)	CS/client	No	Full source	High (6)
9	Flash (AVM2)	CS/client	Yes	Full source	High (6)
10	JavaScript	CS/server	No	Full source	Low (1)
11	Flash (AVM1)	CS/client	No	Full source	High (6)
12	JavaScript	CS/client	No	Full source	High (6)
13	Flash (AVM2)	CS/client	No	Full source	High (6)
14	HTML5	CS/client	No	Full source	High (5)
15	Flash (AVM1)	CS/client	No	Full source	High (6)
16	Flash (AVM2)	CS/client	No	Full source	Medium (4)

Table 11: Technology related findings for advergame cases

Technology platforms

Table 12 summarizes the results about the technology platform found in use from Table 11. Despite predictions by numerous analysts about the decline of Adobe Flash in the near future in favor of HTML5, see for example Goldman (2011) and Vaughan-Nicols (2011), Flash was found to be the dominant technology platform for advergame development in this study, followed by JavaScript. In fact, we only found one game that was developed using HTML5 technology. This has most likely to do with the fact that Flash still has the best overall market penetration for desktop computers (95,36% according to StatOWL[28]) as compared to a HTML5 desktop market share of just 58.1% at the end of 2011 (Ozer, 2011). Other factors that are likely to contribute to this situation are, that there is existing experience in Flash and switching to HTML5 requires learning new a new platform and new skills, and the fact that desktop HTML5 based games often require more resources (Gullen, 2011).

Technology	Number	Percentage
Flash (AVM2)	7	43.8%
Flash (AVM1)	4	25.0%
JavaScript (DHTML)	4	25.0%
JavaScript (HTML5)	1	6.2%

Table 12: Technology platforms found in our study

28 http://www.statowl.com/flash.php (last accessed: August, 5 2012)

Architectures

The architecture column in Table 11 shows that all but two advergame cases used the simple client/server based architecture with the game logic fully located within the game client as depicted in Figure 8 on page 41 (denoted in the table as “CS/client”) am its popularity can be explained by the relative simplicity of implementing an advergame using this architecture. Only two cases (#6 and #10) used an architectural arrangement where the game logic is fully located on the server, as depicted in Figure 10 on page 42 (denoted in the table as “CS/server”) and this small number can most likely be explained by the drawbacks this type of architectural arrangement has. Now if we look at the vulnerability column it can be seen that the two cases with the game logic on the server are the only two cases with their vulnerability assessed as low, where all the other advergames have their vulnerability assessed at at least medium, and often even at the highest possible vulnerability value (6). This is in line with our prediction that having the game logic located on the server provides better resistance against fairness threats than the basic client-server architecture. This does however not mean the two cases are invulnerable to cheating, as some threats are still possible.

Reverse engineering success

Reverse engineering is the main analysis technique used for this research (see section 3.5), but also a prerequisite technique for many threats against fairness (see section 4.2). The 'RE success' column in Table 11 shows our success in reverse engineering the different advergame cases for this research and the 'Obfuscation' column shows whether an obfuscation control was used. For all but one advergame it was possible to obtain the full (reconstructed) source code, despite obfuscation. Only for one advergame, case #7, we have not been able to recover a full, accurate and re-compilable source representation within a reasonable time frame because of obfuscation. However, we could still recover enough parts of the source code, together with disassembled instruction listings, to fully understand the relevant parts of the game and retrieve the game's secrets. For the three other cases that used obfuscation, we had no problem working around it to retrieve the full source code. These findings support our prediction that reverse engineering Flash and JavaScript based games is easy thanks to tools like decompilers, and our prediction that existing obfuscation tools for Flash and JavaScript are weak and do not effectively protect a game against reverse engineering.

5.2 Threat vulnerability prevalence

Table 13 shows the prevalence of vulnerability to the 17 threats identified in section 4.4 for the advergame cases analyzed. For each particular threat the value 0 means no vulnerability, value 1 means reduced vulnerability, because of a design choice or due to a control, which means that successful exploitation of the vulnerability using that particular threat requires more skill and effort. And value 2 means full vulnerability to that particular threat. The following can be observed from these results:

- The three fairness threats that we assigned a high generic threat-level to (threat #2.1, #3.1 and #3.2) have indeed a high occurrence of vulnerabilities. However, what really makes their threat-level high, is how easy they are to perform and the technical impact that results from successful exploitation. It can be seen that a large number of cases are fully vulnerable to these threats.
    - Almost every advergame campaign is in principle vulnerable to threat #2.1, however most advergame cases use a secret-based mechanism to protect the integrity of communication. Performing this threat then requires reverse-engineering the game client to obtain this secret, hence the large number of reduced vulnerability ratings. All advergame cases, except for those two that have the game logic located on the server, are vulnerable to speedhack (threat #3.1). For the four cases with reduced vulnerability, there's either an opportunity for server-side detection, or a speedhack cheat doesn't result in a significant advantage.
    - Many advergames are vulnerable to memory manipulation (threat #3.2) as well, but vulnerability is often reduced because the score is not a discrete value, or because performing a successful memory manipulation attack requires changing multiple variables in order to avoid detection. For reverse-engineering based threats related to tampering with the game logic (#4.1 #4.2, #4.3 and #4.4) all campaigns except for those that have the game logic located on the server are vulnerable to some degree. However vulnerability is sometimes reduced because obfuscation controls are in place (even though they are weak).A significant share of advergame cases is to some degree vulnerable to automation (threat #1.1) and artificial intelligence based threats

No.	Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Σ
1.1	Automation cheats	1	2	0	0	0	0	1	1	0	0	0	2	1	0	0	0	**8**
1.2	Artificial Intelligence / Bot play	0	0	1	2	2	2	0	0	0	1	0	0	0	0	2	0	**10**
1.3	Game glitches	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	1	**1**
2.1	Outgoing communication tampering	1	1	1	2	1	0	1	1	1	0	2	2	1	1	2	1	**18**
2.2	Leaking privacy sensitive information	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**
2.3	Incoming communication tampering	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**
2.4	Insufficient transport level security	0	2	0	0	2	2	2	1	0	2	2	2	2	2	2	2	**23**
3.1	Speedhack	2	2	1	1	2	0	1	1	2	0	2	2	2	2	2	2	**24**
3.2	Memory manipulation	1	2	1	0	1	0	1	2	2	0	1	0	1	2	1	2	**17**
3.3	Debugger manipulation	0	0	0	2	0	0	0	0	0	0	0	2	0	2	0	0	**6**
4.1	Game logic modification with source	1	2	1	1	2	0	0	2	1	0	2	2	2	2	2	2	**22**
4.2	Replace game client with script	1	2	1	1	2	1	1	2	1	1	2	2	2	2	2	2	**25**
4.3	Simulation against detection	1	2	1	1	2	0	1	2	1	0	2	1	2	2	2	2	**22**
4.4	Game logic modification without source	1	2	1	0	2	0	2	2	1	0	2	0	2	0	2	2	**19**
4.5	Non-logic game modifications	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	**2**
5.1	Weak player authentication	2	0	2	2	0	0	0	2	2	2	2	0	2	0	2	0	**18**
5.2	Interface manipulation	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	**2**

Table 13: Threat prevalence among cases (0 = not vulnerable, 1 = reduced vulnerability, 2 = vulnerable)

(threat #1.2). All four cases that are rated vulnerable to threat #1.2 are memory-based games, for which numerous automatic solvers exist on the internet.

- The three games vulnerable to debugger manipulation (threat #3.3) are all JavaScript based, which can be easily manipulated directly from the browser.
- With regard to privacy, a large number of cases (12 out of 16) does not transmit privacy-sensitive data over HTTPS. Also a large number of campaigns has no or insufficient authentication for players (threat #5.1), which could allow attackers to play using the account of somebody else.
- With regard to game glitches (threat #1.3), the fact that we didn't detect them, does not mean that they aren't there. That's why the value is set to no answer for those cases.
- For each threat we found at least one vulnerable advergame case, except for threat #2.3.

We can conclude that the results from Table 13 generally support the predictions we made with regard to the threats we identified in section 4.4.

5.3 Risk assessment results

Table 14 shows the results from the risk assessment of each advergame case. These results are only with respect to fairness as insufficient data was available to fully assess the privacy risks of the advergames. A privacy-related risk rating has only been established for one advergame, case #1 which is exposed to a large privacy risk. See Appendix B: Cases for full details and for detailed information about the risk assessment of all other cases. Details about the risk assessment process and scoring tables can be found in section 3.2.2. The following can be observed from these results:

- Most advergame cases in our study provide a strong motivation for potential cheaters to cheat in the form of expensive prizes. Of the 16 cases, 11 (68.9%) are assessed as high, 4 (25%) as medium and only 1 (6.2%) as low with regard to motivation. Both the mode and median are high.
- Most advergame cases are highly vulnerable to fairness threats, with 11 (68.8%) of the cases assessed as high, and 3 (18.7%) as medium. Only two (12.5%) campaign (those that have the game logic at the server) are assessed to have a low vulnerability. Both value of both the mode and median is 6 (high).
- As a consequence, the likelihood for fairness related impacts is high as well, with 13 (81.3%) out of 16 cases assessed as high, 1 (6.2%) as medium, and 2 (12.5%) as low. The mode is 12 (high, and the highest value on the ordinal scale used) and the median is 11 (high).

Case #	Motivation	Vulnerability	Likelihood	Impact	Risk
1	High (3)	High (6)	High (12)	High (8)	Very high (51)
2	Medium (2)	High (6)	High (10)	Low (3)	Medium (24)
3	High (3)	Medium (4)	High (10)	Medium (5)	Medium (35)
4	High (3)	High (6)	High (12)	Medium (5)	High (39)
5	Medium (2)	High (6)	High (10)	Low (1)	Low (10)
6	Medium (2)	Low (2)	Low (4)	Medium (5)	Low (17)
7	High (3)	Medium (4)	High (10)	High (8)	High (46)
8	High (3)	High (6)	High (12)	Low (3)	Medium (28)
9	High (3)	High (6)	High (12)	Medium (5)	High (39)
10	Medium (2)	Low (1)	Low (2)	Low (3)	Low (6)
11	High (3)	High (6)	High (12)	Medium (5)	High (39)
12	High (3)	High (6)	High (12)	Low (2)	Medium (20)
13	High (3)	High (6)	High (12)	Low (3)	Medium (28)
14	Low (1)	High (5)	Medium (5)	Low (3)	Low (14)
15	High (3)	High (6)	High (12)	Medium (5)	High (39)
16	High (3)	Medium (4)	High (10)	Medium (5)	Medium (35)

Table 14: Risk assessment results (fairness, see section 3.2.2 for explanation of values)

- With regard to impact, only two (12.5%) advergames have a high potential impact, due to the sensitivity of the brand involved to cheating. For the other advergame cases the potential impacts are equally divided between low and medium (both 43,8%). The value of both the median and the mode is 5 (medium).
- The overall risk distribution is diverse, with one advergame campaign rated as being at very high risk (51, almost the highest value possible). Five others are rated as high (31.3%), 6 as medium (37.5%) and 4 as low (25%). The overall risk rating of most cases is reduced by the impact, which is generally rated lower than the vulnerability. For risk, the mode is 39 (high) and the median is 31,5 (medium).

5.4 Conclusion

The results from research stage II support our initial hypothesis that advergames are exposed to a significant amount of risk. High vulnerability levels and often strong incentives for cheating mean that the likelihood of cheating is almost uniformly high for all advergames assessed. And this is supported by our (anecdotal) observations that occurred for almost every campaign, and often at a large scale (see case descriptions). While the impact of individual cheating in a campaign varies from case to case, it has the potential to be significant for most cases with a median of 31,5 (on a scale from 1 to 53), leaving these campaigns to the mercy of chance of whether a campaign's worst impact scenario becomes true, as it often requires only one frustrated player to bring the problem under the attention of a news website. In conclusion, this is indicative of a serious and structural problem with regard to the awareness and understanding of fairness risks in advergames.

The threat model and descriptions developed as part of sections 4.3 and 4.4 proved useful, adequate and complete in supporting the assessment of advergame risks. For every threat, except one, at least one case was found vulnerable and those threats that we rated with a high threat-level also showed the highest level of vulnerability prevalence.

Finally, our predictions with regard to the inherently higher level of vulnerability for the simple client/based server architecture, and the better protection provided by a client/server architecture with the game logic and state located at the server are supported by the data. Unfortunately, no advergames using an client/server architecture that divides the game logic and state over the client and server were found in use. Additionally, our predictions with regard to reverse engineering advergames and the ineffectiveness of obfuscation are supported by the results.

Chapter 6: Solution (stage III)

**** This chapter is confidential ****

Chapter 7: Evaluation

This chapter evaluates the research performed during this thesis. The research from stages I and II is evaluated against the four criteria for qualitative research defined by Lincoln & Guba (1985). The design science research from stage III is evaluated against the seven guidelines from Hevner et al. (2004).

7.1 Evaluation of qualitative research (stages I and II)

Because the four criteria from the validity framework (internal & external validity, reliability and objectivity) are rooted in the positivist scientific tradition associated with quantitative research, and since our study is based on qualitative research (which takes a naturalistic perspective), it is difficult to use them to directly evaluate our research (Golafshani, 2003; Morse et al., 2002; Lincoln & Guba, 1985). We will use the framework for establishing the trustworthiness of qualitative research proposed by Lincoln and Guba (1985) instead. They propose four widely adopted alternative criteria that better reflect qualitative research analogous to the quantitative criteria, and explain how they are related to the quantitative criteria.

7.1.1 Internal validity (credibility)

The internal validity of a study is the degree to which the research truly measures what it intends to measure and refers to the rigor with which the study was conducted (Golafshani, 2003). Lincoln and Guba (1985) propose credibility as an analogue criteria for qualitative research, which refers to the confidence in the truth of the data and the interpretations of them. Morse et al. (2002) define four strategies to achieve and verify credibility in a qualitative research project

Methodological coherence refers to the congruence of the research question and the components of the methods used: the question must match the method, which must match the data and analytic procedures. We believe that stage I of our research was methodologically coherent because our analysis method (reverse engineering) allowed advergames to be analyzed for vulnerabilities (and related threats) from the same position and point of view as potential attackers. With respect to stage II, we believe that the methods and analyses used to perform a risk assessments are coherent with answering our research question (investigate manifestation and prevalence). However, we did have a problem with our data, that prevented us from obtaining credible results with regard to privacy risk. Since we did not have access to server-side data, we could only assess a very small portion of an advergame's vulnerability to privacy threats, but most of the vulnerability with regard to privacy comes from the server-side. Thus, the fact that we did not identify a vulnerability in a particular campaign, does not allow us to conclude that the campaign is indeed at low risk. Consequently, we have left privacy risks out of the risk assessment and therefore we have only limited results with regard to privacy, but what we have is credible. For fairness this did not pose as much of a problem, as most fairness threats originate from the client-side, and we explicitly excluded the server-side from our scope. In conclusion, we believe that this research has been methodologically coherent.

Appropriate sample size. We believe that the number of cases used (16) is enough to draw valid conclusions. With regard to the first research stage, both saturation and replication was achieved with the threats we identified. Most threats were already identified with the first few cases, and replicated many times with the remaining cases, which allowed us to verify these threats, and gave us a sense of comprehension and completeness. The same applies for the stage II. While 16 is not a very large number of cases, it provides enough insight into the prevalence of threats and risks in existing advergame campaigns with a clear trend towards high vulnerability and medium to low impact. Therefore, we believe this property is maintained.

The last two, *Thinking theoretically* and *theory development* refer to the cycle where ideas that emerge from data are reconfirmed in new data, and new ideas from new data should be verified with data already collected. The deliberate movement between a micro perspective of the data and the macro conceptual/theoretical understanding that this requires is necessary for theory development. And this is exactly the iterative process that occurred during the first stage of this research, where individual threats were identified, abstracted to generic threats, and verified with other (past and future) data. The final result where the threats identified are tested on all cases (as can be seen in Table 13 in section 5.2) provides credibility to the validity of these threats, but can not be regarded as an empirical test of these threats as they are tested on the same data as where they are induced from, and therefor such an approach would be flawed.

In conclusion, we think that the results presented by this research are credible and thus internally valid.

7.1.2 External validity (transferability)

The external validity of a research study is the degree to which the research results are generalizable, thus to what extent they are still valid for a larger, possibly slightly different population. The analogue criteria as proposed by Lincoln and Guba for qualitative research is transferability, which is concerned with to what degree the results of qualitative research can be generalized or transferred to other contexts or settings. According to Lincoln and Guba, a researcher can not be specify the transferability for his own qualitative research, and all he can do is to provide sufficient descriptive data in the research report so that other researchers can evaluate the applicability of the data to other contexts. We believe this research satisfies this property as the context, assumptions and limitations of this research have been well specified throughout the document. Nevertheless, we would like to say a bit more about transferability.

With regard to the results from the first stage of this research (as presented in chapter 4), we believe that the fairness implications from the basic architectural arrangements as presented in section 4.1 are transferable to all other advergames, and possibly even to online games in general. The basic assumption: the client can not be trusted is equally valid such settings. With regard to the threat descriptions developed in section 4.4, we believe that they will be transferable to most advergames or regular games within a similar context (browser-based, not real time multiplayer), because such games would have an architectural structure similar to the one on which our threat categorization is based (see section 4.3). With regard to online games that do not have a similar context, such as games that need to be installed, or real-online multiplayer games, we believe that the basic theory behind the different threats identified is still equally valid, but that these threats will then only form a subset of the total set of threats that such games could be exposed to. And as a consequence the categorization we developed would be very much incomplete.

With regard to the results from the assessment of risk in existing advergame campaigns (as presented in chapter 5), we expect the results to be generalizable to at least the population of similar advergames in The Netherlands. Because we expect that the different game studios that created the advergames we investigated, are responsible for a significant share of all advergames developed in The Netherlands. The rate and type of vulnerabilities found are indicative of a more structural problem in the understanding of advergame threats, which makes it likely that other advergames developed by the same studios would have similar vulnerabilities. Since no data was collected about advergame campaigns outside of The Netherlands, estimating the transferability to a European or worldwide population would be guesswork. For example, it might be the case that, possibly because of historic reasons, only in The Netherlands high value prizes are common for advergame campaigns (which result in high values for motivation), where this might be uncommon for other markets. This will have a direct result on the likelihood of these risks in those markets. However it seems likely that outside of The Netherlands, many advergames will also be vulnerable to the threats identified, but the prevalence might be different.

7.1.3 Reliability (dependability)

The reliability is the extent to which the measuring procedures used during research yield the same result on repeated trials. With other words, it is concerned with whether the results of a study can be reproduced using a similar methodology, thus whether it is repeatable or replicable (Golafshani, 2003). The analogue criteria as proposed by Lincoln and Guba (1985) is dependability. The dependability question is: would the findings of a study be repeated if it were replicated with the same (or similar) cases in the same (or similar) context? Thus, dependability depends on the repeatability of a study and the consistency of the findings.

For the first, theory-forming stage of this research dependability is relatively difficult to evaluate, as a different researcher, when presented with the same cases, might come up with a (slightly) different categorization and list of threats, as this is an analytical process of which the outcome depends on the investigator's responsiveness to the data (Morse et al., 2002). For stage II of this research we can evaluate the dependability by looking at how carefully we documented the methodology used for gathering data and measuring our variables. All the measured concepts in this risk assessment, either variables directly assessed or composite measures, have been operationalized using cut-off definitions and the steps required for assessing them have been carefully documented. However, this is still partially subjective. We do not expect this to be a problem for the variables measured in relation to likelihood (as they are grounded in the factual data), however for the impact related variables, these are assessed based on scenario analyses (which are predictions), and different researchers might come up with slightly different predictions. One way to increase the dependability of the measures used and the measurements made is to have two or more researchers perform the same measurements on the same cases and compare their findings. This would increase the dependability of our measurements and say something about the measures we used (called the inter-rater reliability), however since the research for this project is required to be performed by one person, this was not possible. Nonetheless, the choices made have been made with consistence and are clearly documented, and therefore even if other researchers do not agree with the value assessed, the assessment itself is transparent. Therefore we believe that our research provides enough dependability to be of academic value.

7.1.4 Objectivity (confirmability)

The objectivity of a research study is the degree to which the results are based on scientific facts and proof, rather than the researcher's perception of a result (which might be biased). The analogue criteria as proposed by Lincoln and Guba is confirmability, which is the degree to which the results could be confirmed by others. The goal of true objectivity is never fully possible and a realistic aim is to remain impartial to the outcome of the research. We've tried to remain as impartial as possible with regard to measuring the different values during the risk assessment that is part of this study. And we've to some degree confirmed some of our results with people from practice using informal discussions, but these can not be regarded as true confirmations. But since this research was ultimately performed by just one investigator, it's hard to exclude the possibility of the unconscious introduction of subjectivity. We therefore welcome other researchers to confirm our findings.

7.2 Evaluation of design science research (stage III)

The design science research in stage III is evaluated against the seven guidelines for understanding and performing design science research by Hevner et al. (2004):

1. Design as an artifact: During this design science research project we've produced both a method to improve the possibility of enforcing fairness of advergame and a proof-of-concept prototype (an instantiation).
2. Problem relevance: The problem relevance is motivated by the risk assessment results (section 5.3)
3. Design evaluation: The design of our solution is extensively validated using informed arguments that show the effectiveness and utility of the solution. Additionally, a prototype has been built that has undergone functional testing, experimental evaluation (threat simulation and controlled experiments) and static analysis to uncover what is required to integrate our solution with an existing game, so as to assess the complexity of this integration and the effort required.
4. Research contributions: The solution proposed is our research contribution and we've demonstrated that it is “implementable” by building a prototype. The solution is also a “clear contribution to the business environment, solving an important, previously unsolved problem”, namely the problem of maintaining fairness in advergames.
5. Research rigor: We've used rigor throughout our design process as the knowledge base has been effectively used by means of a rigorous literature search. The claims we've made about our solution are supported by rigorous analysis of the properties of our solution, and demonstrated and evaluated by means of a proof-of-concept prototype.
6. Design as a search process: When designing our solution, the basic idea (game session replay) has been gradually extended and improved using knowledge from the application domain (requirements and constraints) and solution domain (technical and organization aspects), to meet our requirements. This has been an heuristic search process, which has led to a solution that is good enough and usable.
7. Communication of research: The technical details necessary for implementing this solution in practice have been well described as well as the rationale for using it as a control to guarantee fairness in advergames with little effort and resources required.

Chapter 8: Conclusions

This chapter presents the main conclusions from this research. Section 8.1 presents the main analytical findings and contributions to theory by answering the research questions described in chapter 1. Section 8.2 discusses the implications for practice. And section 8.3 discusses the limitations of this research and opportunities for further research.

8.1 Answers to research questions

The goal of this research was defined as:

> *To develop detailed insight in how different IT-based risks can affect advergames, their their prevalence in existing advergame campaigns; and to design a control to mitigate (some of) these risks*

And this has been achieved by answering the research questions, which are briefly answered here:

> *1. What are advergames? Why and how are they used?*

The first research question is addressed by sections 1.1 and 1.2. Advergames are computer games designed around a brand or product and are specifically created to communicate advertising messages. They seek to combine the brand with entertainment through an interactive experience that better holds the attention of consumers than with traditional advertising media. They perform especially well with regard to brand memory and persuasion, but are also used to attract traffic and to collect data from players. They are often simple, online browser-based games developed in Adobe Flash or JavaScript.

> *2. To what IT-based risks are advergames potentially exposed?*

This research question is addressed in chapter 2. The interactive, technological and data-driven nature of advergames results in a higher risk-profile than for other types of advertising and four key areas of IT-based risk have been identified that advergames are exposed to. Security, where threats can target the brand owner's assets or visitors. Fairness, where cheating can harm a brand's reputation and undermine the fun perceived by players, the key catalyst of advergame success. Privacy, where threats are concerned with the loss of privacy sensitive data of players. And quality of experience risks, which can frustrate players and undermine fun as well. This has been the first study that investigated the IT-based risks that can affect the success of an advergame campaign, as until now, the academic literature on the subject of advergames solely covers the subject from the perspective of advertising, communication studies or consumer psychology.

> *3. How are the different threats from these IT-based risks manifested in existing advergame campaigns?*

This question is answered by chapter 4. A model for fairness and privacy threats that is operationalized for the technological context of advergames, based on the analysis of vulnerabilities in sixteen existing Dutch advergame campaigns has been developed. We believe that this model offers a lower level of conceptual abstraction, and a better fit with practice than existing models, such as the model by Yan & Randell (2009). Eighteen different threats are categorized according to what part of an advergame's architecture they target and detailed threat descriptions have been provided that explain when an advergame is vulnerable and why and how these threats work.

> *4. What is the prevalence of these risks in existing advergame campaigns?*

The answers to this question are presented in chapter 5 and have been obtained by performing risk assessments on 16 existing advergame campaigns. The results indicate overall high vulnerability for fairness threats, with medium to low impacts and medium to high overall risk and this supports the hypothesis that advergames are exposed to a significant amount of risk. In conclusion, the results are indicative of a serious and structural problem with regard to the awareness and understanding of fairness risks in advergames.

> *5. Can a solution (technical or organizational) be designed to mitigate these risks?*

The answer to this question is presented in chapter 6. A technical solution has been designed that allows for the detection of most fairness related advergame threats identified in the threat model we developed. The solution supports the detection of cheating by allowing an advergame manager to fully reconstruct and replay a game session so that its reported results can be compared to the actual results. A proof-of-concept prototype has been developed to assess the implementation requirements and to demonstrate the feasibility of this solution and the results indicate that replay verification is indeed an effective solution against cheating that can be integrated in existing or new advergames with relatively low effort. Thus the second hypothesis is supported as well.

6. What is the validity of this research?

This question is answered in chapter 7. The first two stages of this research have been evaluated against the criteria for trustworthiness by Lincoln and Guba (1985), the qualitative analogue to the validity framework, and have been found credible (internally valid). We expect that the threat model is transferable to most other advergames, where the results about prevalence will at least transfer to the population of advergames in the Dutch market. With regard to reliability or dependability, we've carefully documented our methodology to include enough detail to make the research repeatable, however some of the measurements based on prediction could potentially be subjective, because the research was performed by just a single researcher. Therefore, we welcome other researchers to confirm our findings. Overall, we think that this research satisfies the criteria for being considered trustworthy. The solution has also been found internally valid and externally valid for similar advergames and the design science research process for the solution has been satisfactorily evaluated against the seven criteria for design science research by Hevner et al. (2004).

8.2 Implications for practice

The success of an advergame campaign does not just depend on its creative and advertising aspects such as how much fun the game is to play or how effectively it communicates the intended advertising messages, but also on how well it manages the risks that it is exposed to. General IT-based risks such as those coming from security and privacy, and more context specific IT-derived risks such as those related to fairness and quality of experience can present a large threat to the success of an advergame campaign. The first step in managing these risks is being aware of their existence and the impact they can have, as this seems to be often overlooked when advergame campaigns are created. This thesis can raise the awareness of these risks with brand owners, marketeers and game developers by discussing how they can affect an advergame campaign and what their potential impact might be.

The fairness & privacy threat model for advergames, the underlying theories and the detailed threat descriptions can help game developers to develop a deep understanding of the problem of maintaining fairness in advergames. They will be able to understand if and why their current advergame campaigns are vulnerable to fairness and privacy threats and what techniques potential attackers will use against their games. Game developers often overestimate the difficulty of performing such cheats and using techniques such as reverse engineering. Detailed information on how such threats are performed and what tools are used, as well as information about their prevalence and threat-level will help them obtain a realistic view of the likelihood that such threats affect their campaigns. Only when the likelihood can be correctly estimated, will it be possible to make a reasonable estimate of the risk a campaign is exposed to. The risk assessment method we developed can provide a starting point for advergame developers to evaluate the risk of their own games and campaigns.

At the same time the results with regard the prevalence of these risks in existing advergame campaigns might be an eye-opener to the advergame industry in its entirety as these results indicate a serious and structural problem with the awareness and understanding of the problem of fairness in advergames. Many campaigns are exposed to significant and unnecessary amounts of risk, and when these risks do occur the advergame brand-owners will be the primary victims. Chances are that advergames will be considered a risky advertising method and as a consequence, brand owners might instead choose other, more traditional, but also more predictable and reliable, methods for bringing their brand under the attention of consumers. This may result in less future market demand for advergames, and as a consequence less revenue and profit for advergame developers and marketeers specialized in advergames. Advergame developers might have to regain the confidence of brand owners that advergames can actually provide a predictable and reasonable-risk approach to advertising.

The solution developed as part of this research can significantly reduce the amount of vulnerability to fairness threats within advergames when correctly integrated in a game and campaign procedures. Successful implementation of replay validation can reduce the problem of fairness from its current state (out of control) to a state that is fully manageable without significant effort. Adoption of this solution in practice can therefore provide a significant contribution to the success of advergames as advertising instrument, and subsequently to the success of advergame developers.

8.2.1 Recommendations

The following recommendations for practice can be given derived from this research:

<u>For brand-owners:</u>

5. Include risk factors in the evaluation of whether the choice for an advergame is appropriate.
6. Consider the risk track record of an advergame company before selecting one.
7. Include risk management requirements in contracts negotiated with advergame creators. But always monitor and maintain an active position within the process, it is after all your brand at stake!
8. Allocate a realistic budget and time-frame on top of the basic advergame budget to assess and mitigate the risks for an advergame campaign.

<u>For advergame creators:</u>

1. Present a realistic view about risks to the client and include risk-management (costs) in proposals. Do not accept a project to create an advergame without a budget for risk management, it is after all also your reputation as an advergame creator at stake!
2. Make the advergame creation process risk aware: involve risk evaluation and communication at every stage of the advergame creation process.
3. Implement replay verification and the complementary controls within your advergame. Do not rely on your own anti-cheating controls and do not rely on anti-reverse engineering techniques as they are unlikely to be effective.
4. Perform a vulnerability assessment of both client-side and server-side code before going live. For the client-side code the threat model developed as part of this research should be used.
5. During live time, continuously monitor the campaign for incidents and respond appropriately.

8.3 Limitations and further research

This research has several important limitations, that at the same time provide opportunities for further research on this subject. The first set of limitations comes from the fact that the research has been performed using a naturalistic approach: all data about existing advergame campaigns was collected using unobtrusive and passive collection methods, without any contact with the campaign organization. This has two important limitations:

1. Only data that was publicly accessible was used within this research. Most notably there was no access to any server-side code, no knowledge of risk management procedures taken at the organization side that were not communicated, no access to any game session data, etc. As a consequence a large number of threats that more strongly affect the server-side than the client side have not been investigated. This has been especially a problem with regard to privacy.
2. Since there was no contact with the advergame campaigns the impact estimates were entirely based on outsider predictions. Brand-owners would be better at the severity of different possible risk scenario's.

This provides opportunities for further research. A future research project might choose the involvement of brand-owners and advergame creators to:

1. Extend the threat model with server-side threats operationalized for the context of advergames by analyzing server-side advergame code, infrastructure and organizational risk management procedures and develop detailed descriptions of these server-side threats
2. Extend and improve the risk assessment on existing advergame campaigns by including server-side vulnerability assessment (which will allow privacy risks to be assessed) and by obtaining better and

more accurate impact ratings by involving brand owners

3. Obtain and analyze game session data from players to determine the actual prevalence of cheating within advergames (as opposed to the prevalence of vulnerability to cheating).

Other limitations are for example related to the scope of this research. We've left the risk areas of security and quality of experience out of our investigation even though these are still academically interesting areas. While most security threats for advergames will most likely be equivalent to generic web security threats, it would be very interesting to assess the vulnerability of advergame campaigns to such threats and compare this with the prevalence of web-security issues in general. It would also be interesting to replicate the study on the prevalence of risk in advergames in different geographic regions, with a larger sample size or at a different moment in time. A follow-up study could for example assess whether there has been any improvement with regard to vulnerability in for example 2 years.

Finally, there exist several possibilities for further research with regard to the developed solution. Replay verification should be further validated and evaluated in practice. The solution could possibly be extended to work with slightly more complex games than typical advergames. And finally, analyzing replay traces obtained from live advergame campaigns might offer a lot of possibility for analysis of the cheating attempts performed on those campaigns.

References

Adobe Systems Inc. (2007a). *ActionScript Virtual Machine 2 (AVM2) Overview.* Available online at: http://www.adobe.com/content/dam/Adobe/en/devnet/actionscript/articles/avm2overview.pdf (last accessed: 11 June 2012).

Adobe Systems Inc. (2007b). *Learning ActionScript 2.0 in Flash*. Available online at: http://livedocs.adobe.com/flash/9.0/main/flash_as2_learning.pdf (last accessed: 11 June 2012)

Adobe Systems Inc. (2007c). *Programming ActionScript 3.0.* Available online at: http://livedocs.adobe.com/flash/9.0/main/flash_as3_programming.pdf (last accessed: 11 June 2012).

Adobe Systems Inc. (2008). *SWF File Format Specification Version 10*. Available online at: http://www.adobe.com/content/dam/Adobe/en/devnet/swf/pdf/swf_file_format_spec_v10.pdf (last accessed: 11 June 2012).

Alvarez, J., & Rampnoux O. (2007). Serious Games: Just a question of posture?. *Artificial & Ambient Intelligence, AISB'07*, pp. 420-423

Ankeny, J. (2011). HTML5's rise spells Adobe Flash Player for mobile's demise. Available online at: http://www.fiercedeveloper.com/special-reports/year-review-2011-stories-shaped-mobile-app-development/html5s-rise-spells-adobe-flas

Avižienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing, 1(1)*, pp. 11-33

Bainbridge, W.A. & Bainbridge, W.S. (2007). Creative uses of software errors: Glitches and cheats. *Social Science Computer Review, 25(1)*, pp. 61-77

Baughman, N.E., Liberatore, M., & Levine, B.N. (2007). Cheat-proof playout for centralized and peer-to-peer gaming. *IEEE/ACM Transactions on Networking, 15(1)*, pp. 1-13

Bellare, M. Canetti, R., & Krawczyk, H. (1996). Keying hash functions for message authentication. *Advances in Cryptology – Crypto'96, Lecture Notes in Computer Science, Vol. 1109*, pp. 1–15.

Bethea, D., Cochran, R. A., & Reiter, M. K. (2011). Server-side verification of client behavior in online games. *ACM Transactions on Information Systems Security, 14(4)*, article 32

Buckner, K., Fang, H., & Qiao, S. (2002). Advergaming: A new genre in Internet advertising. *SoCbytes Journal, 2(1)*

Caballero, A. (2009). *Computer and Information Security Handbook*. Morgan Kaufmann Publications, chapter 14, pp. 323.

Cauberghe, V., & De Pelsmacker, P. (2010). Advergames: The impact of brand prominence and game repetition on brand responses. *Journal of Advertising, 39(1)*, pp. 5–18

Chen, J. (2007). Flow in games (and everything else). *Communications of the ACM, 50(4)*, pp. 31-34

Chen, J., & Ringel, M. (2001). Can Advergaming be the future of interactive advertising?. *<kpe> Fast Forward 2001*

Chen, K.T., Pao, H.K.K., & Chang, H.C. (2008). Game bot identification based on manifold learning. *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games,* pp. 21–26

Collberg, C.S. & Thomborson, C. (2002) Watermarking, tamper-proofing, and obfuscation - tools for software protection. *IEEE Transactions on Software Engineering , 28(8)*, pp. 735- 746

Cronin, E., Filstrup, B., & Jamin, S. (2003). Cheat-proofing dead reckoned multiplayer games. *Proceedings of the 2nd International Conference on Application and Development of Computer Games.*

Duh, H.B.L., & Chen, V.H.H. (2009). Cheating Behaviors in Online Gaming. *Online Communities and Social Computing, Lecture Series in Computer Science vol. 5621/2009*, pp. 567-573

ECMA. (2011). *ECMAScript Language Specification – ECMA-262 Edition 5.1*, Available online at: http://ecma-international.org/ecma-262/5.1/ (last accessed: 11 June 2012).

Eilam, E., & Chikofsky, E.J. (2007). *Reversing: secrets of reverse engineering*. John Wiley & Sons, ISBN 978-0-7645-7481-8, p. 3

Elliot J.C. & Cross J.H. (1990). Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software, 7(1)*, pp. 13-17

Gauthier-Dickey, C., Zappala, D., Lo, V., & Marr, J. (2004). Low latency and cheat-proof event ordering for peer-to-peer games. *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video (NOSSDAV '04)*, pp. 134-139

Golafshani, N. (2003). Understanding reliability and validity in qualitative research. *The Qualitative Report, 8(4)*, pp. 597–607.

Goldman, D. (2011). *The beginning of the end of Adobe's Flash*, CNN Money, available online at http://money.cnn.com/2011/11/10/technology/adobe_flash/ (last accessed: 11 June 2012).

Goodman, J. & Verbrugge, C. (2008). A peer auditing scheme for cheat elimination in MMOGs. *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 9-14.

Gross, M. L. (2010). Advergames and the effects of game-product congruity. *Computers in Human Behavior, 26(6)*, pp. 1259–1265

Gullen, A. (2011). *HTML5 2D gaming performance analysis*. available online at: http://www.scirra.com/blog/58/html5-2d-gaming-performance-analysis (last accessed: 11 June 2012).

Gurau, C. (2008). The Influence of Advergames on Players' Behaviour: An Experimental Study. *Electronic Markets (18:2),* pp. 106-116

Hang, H., & Auty, S. (2011). Children playing branded video games: The impact of interactivity on product placement effectiveness. *Journal of Consumer Psychology, 21(1)*, pp. 65-72

Hernandez, M.D. (2008). Determinants of children's attitudes towards advergames: the case of Mexico. *Young Consumers, 9(2)*, pp. 112-120.

Hevner, A.R, March, S.T, Park, J; & Ram, S. (2004). Design science in information system research. *MIS Quarterly, 28(1)*, pp. 75-105

ISACA. (2009). *The Risk IT Practitioner Guide*, ISACA, ISBN 978-1-60420-116-1

International Organization for Standardization (ISO). (2011). *ISO/IEC 27005:2011, Information technology – Security techniques – Information security risk management.*

Isbister, K, Schaffer, N. (2008). *Game Usability: Advancing the Player Experience*, Morgan Kaufmann Publications, ISBN 978-0-12-374447-0

Iuppa, N., & Borst, T. (2010). *End-to-End Game Development: Creating Independent Serious Games and Simulations from Start to Finish*, ISBN 978-0240811769, Focal Press, Waltham, USA

Kaplan, A.M., & Haenlein M. (2011). Two hearts in three-quarter time: How to waltz the social media/viral marketing dance. *Business Horizons, 54(3)*, pp. 253-263

Kretchmer, S.B. (2005). Changing Views of Commercialization in Digital Games: In-Game Advertising and Advergames as Worlds in Play. *Proceedings of DIGTAR conference: Changing Views: Worlds in Play*, Vancouver, Canada.

Lee, M., & Faber, R.J. (2007). Effects of product placement in on-line games on brand memory. *Journal of Advertising, 36(4)*, pp. 75-90

Lewis, C.; Whitehead, J., & Wardrip-Fruin, N. (2010). What went wrong: A taxonomy of video game bugs. *Proceedings of the Fifth International Conference on the Foundations of Digital Games, ACM Digital Library*, pp. 108-115.

Li, L., Daugherty, T., & Biocca, F. (2002). Impact of 3-D advertising product knowledge, brand attitude, and purchase intention: the mediating role of presence, *Journal of Advertising, 31(3)*, pp. 43-57

Lincoln, Y. S. & Guba, E. G. (1985). *Naturalistic inquiry*. Beverly Hills, CA: Sage.

Mau, G., Silberer, G., & Constien, C. (2008). Communicating brands playfully: The impact of ad placements in computer games. *Journal of Advertising, 27(5)*, pp. 827-851.

Martin, P.Y., & Turner, B.A. (1986). Grounded Theory and Organizational Research. *The Journal of Applied Behavioral Science, 22(2)*, pp. 141-157.

References

Mitterhofer, S., Platzer, C., Kruegel, C., & Andkirda, E. (2009). Server-side bot detection in massive multiplayer online games. *IEEE Security & Privacy, 7(3)*, pp. 18–25.

Morse, J. M., Barrett, M., Mayan, M., Olson, K., & Spiers, J. (2002). Verification strategies for establishing reliability and validity in qualitative research. *International Journal of Qualitative Methods, 1(2)*

Mönch, C., Grimen, G., & Midtstraum, R. (2006). Protecting online games against cheating. *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games, NetGames '06*

Myers, M.D. (2004). Hermeneutics in Information Systems Research. In J. Mingers and L.P. Willcocks (eds.), *Social Theory and Philosophy for Information Systems,* John Wiley & Sons, Chichester, pp. 103-128.

Nelson, M.R. (2005). Exploring consumer response to advergaming. In C.P Haugtvedt, K.A. Machleit, & R. Yalch (Eds.), *Online consumer psychology: Understanding and influencing consumer behaviour in the virtual world*, pp. 167-194, Lawrence Erlbaum

Nelson, M.R., & Waiguny, M.K.J. (2012). Psychological processing of In-Game advertising and Advergaming. In L.J. Schrum (ed.), *The Psychology of Entertainment Media: Blurring the Lines Between Entertainment and Persuasion 2nd edition*, pp. 93-144, Routledge

Nelson, M.R., Yaros, R.A., & Keum, H. (2006). Examining the influence of telepresence on spectator and player processing of real and fictitious brands in a computer game. *Journal of Advertising, 35(4)*, pp. 87-99

Okazaki, S., & Yagüe, M.J (2012). Responses to an advergaming campaign on a mobile social networking site: An initial research report. *Computers in Human Behavior, 28(1)*, pp. 78-86

Ozer, J. (2011). HTML5 Desktop Market Share at 58.1% Max, on *Streaming Learning Center*, available online at http://www.streaminglearningcenter.com/articles/stat-of-the-week-html5-desktop-market-share-at-581-max.html (last accessed: 11 June, 2012)

Park, S.; & Miller, K. (1988). Random Number Generators: Good Ones are Hard to Find. *Communications of the ACM, 31(1)*, pp. 1192-1201

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems, 24(3)*, pp. 45-77.

Ping, J.W., Goh, K.Y., & Teo, H.H. (2010). Engaging Consumers With Advergames: An Experimental Evaluation Of Interactivity, Relevancy And Expectancy. *ICIS 2010 Proceedings,* paper 221

Pritchard, C.L. (2010). *Risk Management: Concepts and Guidance 4th edition*. ESI International, ISBN 1890367559

Rivest, D. (1992). *RFC1321: The MD5 Message-Digest Algorithm.* Internet Engineering Task Force, available online at: http://tools.ietf.org/html/rfc1321 (last accessed: 11 June, 2012).

Ross, Seth. (1999). *UNIX System Security Tools*. McGraw-Hill Companies, September 1999

Schlarman, Steve (2009). IT Risk Exploration: The IT Risk Taxonomy and Evolution. *ISACA Journal, 2009(3)*

Schluessler, T., Goglin, S., & Johnson, E. (2007). Is a bot at the controls?: Detecting input data attacks. *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07)*

Schneider, L.-P., & Cornwell, T.B. (2005). Cashing in on crashes via brand placements in computer games. *International Journal of Advertising, 24(3)*, pp. 321-343

Shirey, R. (2007). *RFC 4949: Internet Security Glossary, Version 2*. Internet Engineering Task Force, available online at: http://tools.ietf.org/html/rfc4949 (last accessed: 11 June, 2012).

Smith, A. (2007). Exploring advergaming and its online advertising implications. *International Journal of Business Information Systems (2:3)*, pp 298-311.

Smith, J.H., & Just, S.N. (2009). Playful persuasion: the rhetorical potential of advergames. *Nordicom Review, 30(2)*, pp 53-68

Solove, D.J. (2006). A taxonomy of privacy. *University of Pennsylvania Law Review, 154(3)*, pp. 477-560

Stoneburner G., Goguen A., Feringa A. (2001). *Risk management guide for information technology systems. National Institute of Standards and Technology Special Publication, 800(30)*

Svahn, M. (2005), Future-proofing Advergaming: A Systematisation for the Media Buyer. *Proceedings of the Second Australasian Conference on Interactive Entertainment, ACM International Conference Proceeding Series*, pp. 187-191

Vaughan-Nichols, S.J. (2011). Flash is dead. Long live HTML5.on *ZDNet Networking*, available online at http://www.zdnet.com/blog/networking/flash-is-dead-long-live-html5/1633 (last accessed: 11 June 2012)

Vikram, K. Prateek, A., & Livshits, B. (2009) Ripley: automatically securing web 2.0 applications through replicated execution. *Proceedings of the 16th ACM conference on Computer and communications security (CCS '09).*

W3C. (2005). *Document Object Model*, available online at: http://www.w3.org/DOM/ (last accessed: 11 June 2012)

W3C. (2012). *HTML5: A vocabulary and associated APIs for HTML and XHTML: W3C Working Draft 29 March 2012*, available online at: http://www.w3.org/TR/html5/ (last accessed: 11 June 2012).

Webb, S. & Soh, S. (2008). A survey on network game cheats and P2P solutions. *Australian Journal of Intelligent Information Processing Systems, 9(4)*, pp. 34-43.

Wieringa, R. (2009). Design science as nested problem solving. *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, DESRIST '09*, art. no. 8

Winkler, T., & Buckner, K. (2006). Receptiveness of Gamers to Embedded Brand Messages in Advergames: Attitudes towards Product Placement. *Journal of Interactive Advertising, 7(1)*, pp. 37-46

Wise, K., Bolls, P., Kim, H., Venkataraman, A., & Meyer, R. (2008). Enjoyment of Advergames and Brand Attitudes: The Impact of Thematic Relevance. *Journal of Interactive Advertising, 9(1)*, pp. 27-36.

Wolfswinkel, J.F; Furtmueller, E.; & Wilderom, C.P.M. (2011). Using grounded theory as method for rigorously reviewing literature. *European Journal of Information Systems, advance online publication November 2011*

Yampolskly, R.V. & Govindaraju, V. (2007). Embedded noninteractive continuous bot detection. *ACM Computers in Entertainment, 5(4)*, pp. 1–11.

Yan, J.J. & Choi, H.J. (2002). Security issues in online games. *The Electronic Library, 20(2)*, pp. 125-133

Yan, J.J. & Randell, B. (2009). An Investigation of Cheating in Online Games. *IEEE Security & Privacy, 7(3)*, pp. 37-44

Yang, H.L., & Wang, C.S. (2008). Product placement of computer games in cyberspace. *Cyberpsychology and Behavior, 11(4)*, pp. 399-404

Youn, S., & Lee, M. (2005). Advergame playing motivations and effectiveness. In M.R. Stafford & R.J. Faber (Eds.), *Advertising, promotion and new media*, pp. 320-347, M.E. Sharp

Appendix A: Source code

**** This appendix is confidential ****

Appendix B: Cases

Case #1

Client company industry	Lottery
Product/brand	Special edition lottery
Campaign period	12 December 2011 – 28 December 2011
Competition	Yes
Prizes	1x Trip to Bahama's (EUR 16.000), 100x lottery ticket (EUR 30)
Competition-type	Main prize: Lottery among top-20 Remaining prizes: Lottery among all players

Advergame promotion for a large lottery company for one of their special edition lotteries. Simple skill-based game where one has to go through four consecutive levels in which one has to perform a successful limbo-dance: using mouse controls maneuver the game avatar under a limbo-stick. The objective is to finish all four levels as quick as possible, without dropping the stick and while keeping your balance. If you drop the stick or fall over, it's game over. At the end of the game period one winner from the top-20 is randomly selected to win a trip to the Bahama's with six friends worth EUR 16.000 in a ultra luxury villa to experience "what life is like when you are a millionaire".

Technical characteristics

Technology	Flash (AVM2)
Obfuscation	Yes, container
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash 10. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server in plain-text but integrity protected with a hash. No intermediate results communicated, but server is signaled when next level is played. Final results communicated at the end, including user registration and total time.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	High (3)

An extremely high-value prize is given away to one of the top-20 scorers, leaving potential cheaters highly motivated. Cheaters could manipulate the game to submit a score high enough to be in the top-20 and be eligible for the lottery. Since the prize is for 6 persons, cheaters could easily position their 5 friends in the top-20 as well, to increase chances. Dedicated attackers might even be motivated to completely fill in the top-20 with friends or different identities of themselves in order to be almost certain of the prize. Therefore we rate the motivation for fairness as high.

Participants have to submit substantial private information including their name, age, address, phone-number and email. Considering that this information is obtained for a lottery campaign, this can be considered extra sensitive (as this information can be extra valuable for lottery spammers). Therefore we rate the motivation with regard to privacy as high.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Low (3)	Low (3)	High (6)
Privacy	Medium (2)	Medium (2)	Medium (4)

<u>Control analysis</u>

1. Communication is protected by a

challenge-based hash implemented by nl.backend.Communicator.getChecksum() that hashes the data that is sent out together with the hash of the previous challenge value.

2. The flash file is protected against decompilation / secret reverse engineering by embedding the actual game logic in a separate Flash file (SWF) contained as raw data in the main SWF . The container SWF will once started, dynamically load the game logic through the Runtime Shared Library (RSL) interface.
3. Data is communicated over HTTPS.
4. Organizational: A top-20 lottery is used to spread the risk of cheating.

Vulnerability analysis

Prevention

- An automation attack using mouse recording (threat #1.1) might be possible, but will require a lot of effort.
- It is possible to tamper with the score data when it's communicated to the server (threat #2.1). However this requires that the hash key is recovered through reverse engineering in order to generate a valid MAC to avoid detection.
- There's a very serious privacy leak in the rpc/top server-request. When high-scores are loaded, full account information of users is transmitted back to the game, as part of regular implementation, including all privacy sensitive information. Using this, the private information of all game participants could be trivially obtained (threat #2.2).
- The game is fully vulnerable to speedhack (threat #3.1), without possibility for server-side detection.
- The game has limited vulnerability to score manipulation through memory (threat #3.2), but this is difficult to perform for this game as locating a time value in memory is more difficult than a score value.
- The chosen obfuscation method will prevent off-the-shelf decompilers from decompiling the game logic of this game. However this obfuscation can be removed: It is possible to extract the SWF containing the game-logic from the container SWF if you possess knowledge of the SWF binary format. This SWF can then be decompiled by off-the-shelf decompilers without problem. And thus reverse-engineering based attacks (threat #4.1, #4.2, #4.3 and #4.4) are possible.
- Player account info is not password protected, and player information is easy to learn from the previously mentioned data leak, which can lead to abuse (threat #5.1).

Detection:

- The minimalist game-server interactions make server-side detection of cheating for individual runs (even by analyzing game results afterwards) impossible.

Skill & effort estimation

The easiest way to cheat this game is by using a speedhack cheat against which the game provides absolutely no protection, and this cheat, having a high threat-level, requires low skill and effort to perform. This is enough to fully compromise fairness, but more advanced cheaters are potentially able to remove the obfuscation layer, but this requires intimate knowledge of the SWF binary format and therefore a significant amount of skill, however people who do possess this knowledge can put it off in about 5 to 10 minutes, and if you're capable of doing that, you will also be capable of writing a simple gameplay simulation script that makes cheating (especially under multiple names) a lot easier. There will be enough people capable of doing this and motivated to do this, if they come across this game.

Compromising users privacy in this game is relatively easy, as this information is leaked automatically when the top-scores are requested. For this, simply intercepting the communication between the game and the Webserver is enough (which requires medium skill and effort), but a script can also be written that automates this (requiring more skill but less effort).

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	High (6)	High (12)
Privacy	High (3)	Medium (4)	High (10)

Using these outcomes we can determine the likelihood for both fairness and privacy to be high. This corresponds with our observations. Analysis of the top-20 scores we observed shows that cheating at this game has happened at a large scale: the entire top-20 consists of scores that are not possible by genuine play because they are too fast. Even if the organizers of the game realize this, due to the lack of detection options, there's no way for them to tell which score is real and which score is not. Smart cheaters might realize that the current top-20 scores are impossible, and position themselves at the edge of what might be possible.

Impact prediction

	Mission	Asset	→ Impact
Fairness	High (3)	Medium (2)	High (8)
Privacy	High (3)	Medium (2)	High (8)

The most obvious impact potential of these risks is that the company will give away their EUR 16.000 prize to an unfair winner, a considerable asset that will take up a large share of the total campaign cost. Therefore we rate the asset loss as medium. With respect to mission loss, the consequences of these threats can be much wider: If the news gets out that this company, a large national lottery company, can not guarantee fairness for a simple online lottery and gives away prizes to cheaters, this can have very serious consequences for the image of this lottery. At a smaller level, the impossibly low-scores will demotivate honest players who think they are impossible to beat and give up. At the same time, many of them will and have (complaints the companies Facebook page) realize(d) that those scores can not possibly be right, which can seriously affect the attitude people have towards the game and campaign, as the company did not seem to respond to it. Therefore we rate the potential mission loss as high.

The problem of leaking of privacy sensitive information could lead to a full compromise of the sensitive details of all participants. News coverage of this, or just the fact that this is possible, could seriously harm the reputation of this company. There's even a possibility for lawsuits or regulatory fines. In conclusion, the impact for both fairness as well as privacy potentially very high.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (12)	High (8)	High (51)
Privacy	High (10)	High (8)	High (46)

Taking into account the high likelihood and high impact for both fairness and privacy, this advergame was at a very high risk.

Case #2

Client company industry	Cheese
Product/brand	Cheese brand
Campaign period	1 April 2012 – 31 May 2012
Competition	Yes
Prizes	61x large inflatable swimming pool (value unknown)
Competition-type	Daily high-score wins

Promotion by cheese company for one of their cheese brands. Simple game where one has to correctly make different mouse/slide movements, as well as click/tap on positions, according to an onscreen pattern as quickly as possible. If a movement is made correctly, then its speed determines the score for that movement. Scores for different movements add up and the goal is to achieve the highest score. The highest score at the end of each day (23:59) during the campaign day wins a large inflatable, branded, swimming pool.

Technical characteristics

Technology	Flash (AVM2)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

The game is distributed online as well as a download for mobile platforms (Android and iOS). We've only analyzed the online version. Browser-based game entirely developed in Flash 10. Simple client-server based architecture where the game client fully implements the game, and the server-logic is just for user and score registration. Communication with server in plain-text but integrity protected with a hash. No intermediate results communicated, only the final result together with user registration.

Likelihood determination

Motivation

	Motivation
Fairness	Medium (2)
Privacy	Low (1)

While the monetary value of the prizes given away might not be very high, the fact that they are limited editions, might make them a collectors item or give them a higher sale value, therefore the motivation with regard to fairness is medium.

With regard to privacy, when one achieves the daily top-score one has to to submit substantial private information including their name, age, address, phone-number and email. However, since this information is only requested if the score obtained by the player is the highest score at that moment, the amount of people privacy sensitive information is collected from is very limited, therefore we rate the motivation for privacy breach as low.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Low (3)	Low (3)	High (6)

Control analysis

1. Communication is protected by a MD5 hash implemented by nl.lvqr.zwembad.model.ApiModel.insertWinner() that hashes the data is sent out together with the a secret value ("IN10DeMonsters").
2. Organizational: Privacy sensitive data is only transmitted if the player achieves currently the highest score.

Vulnerability analysis

Prevention

- An automation attack using mouse recording (threat #1.1) might be relatively easy, due to the large number of very simple time-based repetitive elements.

- It is possible to tamper with the score data when it's communicated to the server (threat #2.1). However this requires that the hash key is recovered through reverse engineering in order to generate a valid MAC to avoid detection.
- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).
- The game is fully vulnerable to speedhack (threat #3.1), without possibility for server-side detection.
- The game is vulnerable to score manipulation through memory (threat #3.2) and this is relatively easy to perform as the score is a discrete numeric variable in memory.
- The lack of obfuscation means that it is possible to decompile the game SWF using any available flash decompiler. This makes the game vulnerable to reverse-engineering based attacks (threat #4.1, #4.2, #4.3 and #4.4) are possible. Finding the hash secret this way is also easy, and requires very limited code reading skills.

Detection:

- Server side cheat detection other than by making an estimate if the score falls in the possible range is not possible due to very minimalistic server interaction.

<u>Skill & effort estimation</u>

The game is vulnerable to three different easy to perform fairness threats without significant opportunity for detection. The speedhack threat is by far the easiest to perform, requiring very limited skill and time. Score manipulation through memory is also easy possible, but requires slightly more skill and effort than the speedhack. Also recovering the hash secret is trivial, which then enables the quickest and easiest of all attacks: modifying the score when it's sent to the server. Many people are capable of performing any of these cheats. Thus the required skill and effort are both low.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	Medium (2)	High (6)	High (10)

Using these outcomes we can determine the likelihood for privacy to be high. This corresponds with our observations. From analysis of the scores we observed during the campaign, we determined that cheating has occurred (scores outside of possible range).

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Low (1)	Low (3)

The campaign is at risk of giving away prizes to unfair winners, which might then have to be replaced, but the value of these assets is relatively low. From a mission perspective, other players might get demotivated or frustrated by the (unrealistically) high scores obtained by cheaters which might result in less participation or reflect on the image of the brand, therefore we rate this as medium.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (10)	Low (3)	Medium (24)

Taking into account the high likelihood and low impact, this advergame was at medium risk with regard to fairness.

Case #3

Client company industry	Food
Product/brand	Instant coffee
Campaign period	6 December 2011 – 31 January 2012
Competition	Yes
Prizes	1x 3-day trip to Rome for 4 persons (EUR 2500)
Competition-type	Highest score at end of period wins

A Facebook-based advergame was used to support such a campaign for sample distribution and mail-in rebate of an instant coffee brand. A simple branded version of the well known Tetris game. The player with the highest score at the end of the game period wins a 3-day trip to Rome for 4 persons worth EUR 2500.

Technical characteristics

Technology	Flash (AVM2)
Obfuscation	Yes, DoSWF
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash 10. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server through several scripts in plain-text but integrity protected with a checksum value. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Low (1)

A high value prize is given away to the player with the highest score, providing strong motivation for cheaters that might try to manipulate the game to submit a false score. Therefore the threat-level to fairness is high.

The motivation for privacy is low as the only data requested and collected through Facebook are the player's full name and user ID (both of which are considered public by Facebook) and its e-mail address.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Medium (2)	Medium (2)	Medium (4)

<u>Control analysis</u>

We've observed two different versions of this game. With the first version a significant amount of cheating occurred which most likely stimulated the developers to create a second version, a few days before the end of the campaign, that added several extra controls in order to combat cheating. For each control we list we specify in what version it was implemented.

1. The score submitted is protected by a simple checksum value implemented by Blox.gameOver() that uses a basic arithmetic formula (score * 87654 + 3 >> 4) % 54321) for its computation. In both versions, but the parameters of the formula changed in the second version.
2. In the second version obfuscation was added to protect against reverse engineering the flash file using the DoSWF obfuscator.
3. The second version updated its score function from a simple linear points per line played formula that is easy to predict, to a formula that contains several irregularities for higher scores (at several times when a certain number of lines is played a different value is additionally added to the score), with the intention of being able to detect cheaters manipulating the score (but who have not detected the changes in the score function).
4. The second version also submits the number of lines played, and the level reached together with the

score to the server. This is information necessary to determine whether the submitted score adheres to updated formula in the second version, and thus to detect potential cheaters.

5. The second version makes an extra server-call to (cron.php) at the start of the game, with the goal of being able to measure the playtime for each game played. This can again be used to detection potential cheaters.
6. Data is communicated over HTTPS.

Vulnerability analysis

Since there are two versions we focus our analysis on the updated second version. The biggest difference between them being that the second version allows for much better detection of cheaters, reducing the vulnerability-level from high in the first version to medium in the second.

Prevention:

- The simple Tetris based game is vulnerable to artificial intelligence attacks as automatic Tetris playing bots exist, but they may need some modification (threat #1.2).
- It is possible to tamper with the score data when it's communicated to the server (threat #2.1). However this requires that the hash key is recovered through reverse engineering in order to generate a valid MAC to avoid detection.
- The game is fully vulnerable to speedhack (threat #3.1). But the second version offers limited opportunities for detection (see detection)
- The first version of the game is vulnerable to score manipulation through memory (threat #3.2), but the second version offers significant opportunities for detection (see detection).
- The obfuscation method in the second version will prevent off-the-shelf decompilers from decompiling the game logic of this game. Removing the obfuscation layer requires significant technical knowledge, but a cheater who possesses this knowledge can accomplish this in about 15 minutes. Finding the hash secret this way is also easy, and requires very limited code reading skills. Without the possibility to reverse engineer the updated checksum function can not be determined, and the concealed irregularities to the score function will most likely also remain undiscovered. This reduces vulnerability to reverse-engineering based attacks (threat #4.1, #4.2, #4.3 and #4.4), but they are still possible.
- The game uses very weak player authentication (threat #5.1), the player's Facebook ID is used which can be changed to any other Facebook ID (which is publicly available).

Detection:

- The first version had no opportunities for server-side speedhack detection, but the second version adds a limited opportunity by allowing the time of a game session to be measured on the server side. This improvement is not as effective as it seems, as the time signature for a certain score can legitimately vary significantly due to differences in playing speed and style. Thus a smart cheater that only slows the game down by a relatively small percentage (say 20-30%) still gets a significant edge over fair players without the possibility of being detected. Scores submitted by speedhack will always have their checksum correct and satisfy any checks on score validity (see next point).
- The first version had no opportunities for server-side detection of memory based score manipulation. However with the updated score function introducing irregularities to high-performance scores cheaters might be tricked in changing their score into a score that the score function can no longer legitimately generate and can therefore be detected by checking the combination of score, level and number of lines plaid (now communicated). The time signature of a memory-manipulated score must also fall within a reasonable range for that score. Thus detecting memory-based score manipulation is now most likely possible, unless perfectly executed.

Skill & effort estimation

The first version was highly vulnerable to three attacks to three different fairness threats without opportunity for detection. The second version was a significant improvement by making most cheats significantly more difficult, but left a gaping hole with the still easy to perform speedhack attack. Preventing detection requires limited skill and effort, and therefore we rate both as medium.

This together with a high threat-level makes the likelihood for risk occurrence high for the version version, and medium for the second version.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	Medium (4)	High (10)

Using these outcomes we can determine the likelihood for privacy to be high. This corresponds with our observations. Large scale cheating was observed with the first version, and sometimes all-too-obvious cheaters were removed. After the installation of the second version (a few days before the campaign ended) cheating initially declined but several highly suspicious scores arrived again on the last day, some of which were provably impossible in terms of limits to human reaction time or could not have been generated by the new score function.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Medium (2)	Medium (5)

The campaign is at risk of giving away a valuable prize to an unfair winner, not just a loss of assets but also potentially damaging to the organizations reputation. But there is no context-sensitivity to either to the brand. From a mission perspective, other players might get demotivated or frustrated by the (unrealistically) high scores obtained by cheaters which might result in less participation or reflect on the image of the brand, therefore we rate this as medium.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (10)	Medium (5)	Medium (35)

Taking into account the high likelihood and medium impact, this advergame was at medium risk with regard to fairness.

Case #4

Client company industry	Online shoe retail
Product/brand	Company name + brands of shoes they sell
Campaign period	28 November 2011 – 30 December 2011
Competition	Yes
Prizes	165x Brandos Gift certificate (value varies, average value 150 EUR)
Competition-type	Daily top-5 wins

An international online shoe store based in Sweden, organized an advergame campaign for all its international subsidiaries, including the Dutch subsidiary. The game was a simple shoe-themed 4x4 memory game, accessible through the different country-specific Facebook pages, which was branded by a daily changing brand from the collection of the shop. The goal of the game is to solve the memory puzzle in as few turns as possible, and in the least amount of time. At the end of each day those players with a top-5 highscore win a gift certificate with a value (ranging from EUR 95 to EUR 269) equal to the shoe on display from the brand that sponsored that days game.

Technical characteristics

Technology	Javascript (DHTML)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

Browser-based game developed in JavaScript (DHTML). Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server through several scripts at the campaign server in plain-text. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Low (1)

Attractive and valuable prizes are given away to players with a top-5 score each day, providing strong motivation for cheaters that might try to manipulate the game to submit a false score. Successful cheaters will most likely attempt to win more gift certificates by playing on friend names or by using fake identities and the digital distribute of the prizes (gift certificate code mailed to winners) facilitates this as no physical address is required. Therefore the motivation to fairness is high.

The threat-level to privacy is low as the only data requested and collected through Facebook are the player's full name and user ID (both of which are considered public by Facebook), its country of residence and its e-mail address.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Low (3)	Low (3)	High (6)

<u>Control analysis</u>

1. A request is made when the game starts and when it completes, which potentially makes it possible to compare the playtime submitted with the playtime measured at the server-side. It seems however that this is not actually performed.
2. The Javascript source code has been obfuscated.
3. When the result is communicated to the server, a key is also communicated, that is computed (rblob() in memory.js) based on a value that has been obtained as a result from the request when the game starts. This key however does not protect any of the critical score values (time and number of turns), and is therefore ineffective.

4. Data is communicated over HTTPS.

Vulnerability analysis

Prevention:

- AI-based solvers for memory-based games are available online and can be used (threat #1.2).
- The game is highly vulnerable to tampering with the communication (threat #2.1) as the key variables (number of turns and playtime) in the result submitted to the server can be modified to any value without having to recompute the key, as it does not depend on those values.
- The game is highly vulnerable to speedhack (threat #3.1) with regard to time, but this does not influence the number of turns required to solve the game.
- The game is fully vulnerable to debugger manipulation (threat #3.3).
- The JavaScript source code obfuscation can be trivially removed with online tools such as the one available at http://jsbeautifier.org enabling reverse engineering attacks (threat #4.1 and #4.1). Once the workings of the rblob() function have been uncovered it is possible to write a simple script emulating a player playing this memory game many times (threat #4.3), beating any kind of player-behavior analysis done by the organization.
- The game uses very weak player authentication (threat #5.1), the player's Facebook ID is used which can be changed to any other Facebook ID (which is publicly available).

Detection:

- In theory it is possible that server-side time measurement was implemented to verify that the submitted playtime corresponds with the measured playtime, however it seems that this was either not implemented or not performed.

Skill & effort estimation

The game is vulnerable to two different easy to perform fairness threats without significant opportunity for detection. Tampering with the communication is by far the easiest to perform, requiring very limited skill and time. Score manipulation through the debugger is also easy possible, but requires slightly more skill and effort than communication tampering. Many people are capable of performing any of these cheats. Thus the required skill and effort are both low.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	High (6)	High (12)

Using these outcomes we can determine the likelihood for fairness to be high. This corresponds with our observations. Analysis of the top-scores and the lists of winners showed that the game has been manipulated at a very large scale. Sometimes this could be directly inferred from the number of turns submitted (you can not solve a 4x4 memory game in less than 8 turns) or by an impossibly large time. But more interestingly a very high share of all winners were people with perfect scores (solving the memory in exactly 8 turns). Out of 165 total winners, 80 won the game the game with such a score. However if we compute the chance for achieving a perfect score for a 4x4 memory game:

$$P(turns=8)=\frac{1}{15*13*11*9*7*5*3}=\frac{1}{2027025}$$

We see that this chance is smaller than 1 in 2 million, an incredibly small chance, yet 80 out of 165 winners managed to achieve this score. According to the developer website, the game has been played about 500000 times in total. With other words: this is statistically impossible. Even the chance of solving this game in 9 turns is very small and yet 45 winners had this score. And therefore cheating occurred at a very large scale. Only the very first scores, of the very first few days have a chance of being real, with a reasonable number of turns and playtime.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Medium (2)	Medium (5)

The campaign is at risk in giving away valuable prizes to unfair winners and this might lead to a loss of assets, declined campaign performance and reputation damage. This campaign was especially at risk for unfair players to grab away many prizes using different prizes. However the the company organizing the campaign is not in a context-sensitive business, thus the campaign can be thoroughly ruined (and it was), but it will most likely stay within the scope of the campaign. Therefore the mission and asset impact is medium.

Many people complained on the different country Facebook pages about cheaters and these statistically unlikely scores. While the retailer removed any score that was definitely impossible (less than 8 turns, or a play time lower than 6.00 secs), it took the position that all the 8 turns scores could potentially have been real and that they can therefore not remove them, as they had no possibility for distinguishing genuine 8 turn scores from cheated ones. Lots of people criticized this, and actually showed that many of the winners were using fake, setup for the campaign, Facebook accounts, so that cheaters could win multiple prizes. The brand owner clearly did not seem to know how to deal with the situation, and its Facebook walls were flooded with negative comments and discussions. high scores obtained by cheaters which might result in less participation or reflect on the image of the brand, therefore we rate this as medium.

Risk

Taking into account the high likelihood and medium impact, this advergame was at high risk with regard to fairness.

	Likelihood	**Impact**	**→ Risk**
Fairness	High (12)	Medium (5)	High (39)

Case #5

Client company industry	Supermarket
Product/brand	Fresh produce
Campaign period	30 January 2012 – 4 March 2012
Competition	Yes
Prizes	1750x Tefal kitchen appliance (EUR 15-30)
Competition-type	Daily top-50 players win a Tefal kitchen appliance of choice

To promote the fresh produce assortment of a supermarket, a simple 6x3 memory game, themed each week with a different category of fresh produce, was developed. The goal is to solve the memory game as quickly as possible. A top-50 best time on a game during the game period wins you a prize (a choice between 6 different Tefal kitchen appliances). Maximum one prize per address per week, allowing each player to win up to 5 prizes for this competition. No high-score rankings are shown.

Technical characteristics

Technology	Flash (AVM1)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server in plain-text but integrity protected with a hash. No intermediate results communicated, only the final result with separate user data registration URL. The score is expressed as the number of flips and the total playtime, but only the playtime is relevant for the ranking.

Likelihood determination

Motivation

	Motivation
Fairness	Medium (2)
Privacy	Medium (2)

A large number of prizes are given away that are individually not very valuable, but as players are allowed to win 5 prizes per address during the campaign period, it might still provide a relatively strong motivation for potentially dishonest players to cheat. Since so many prizes are given away it's also tempting to also win for your friends or family, adding to the threat. Therefore the the motivation for fairness is medium.

The motivation privacy is medium as the data requested and collected is relatively substantial consisting of the player's full name, e-mail address, phone-number and province of residence.

We've also observed several bugs that would frequently occur doing game-play resulting either in cosmetic glitches (visually annoying, but you can continue the game) or even a hung game which needs to be restarted. This can be a threat to the quality of experience for players.

Vulnerability

	Skill	Effort	→ Vulnerability
Fairness	Low (3)	Low (3)	High (6)

<u>Control analysis</u>

1. Communication is protected by a MD5 hash implemented in the MainMovie that hashes the score and playtime together with the a secret value (“coop”).

<u>Vulnerability analysis</u>

Prevention

- AI-based solvers for memory-based games are available online and can be used (threat #1.2).
- It is possible to tamper with the score data when it's communicated to the server (threat #2.1). However this requires that the hash key is recovered through reverse engineering in order to generate a valid MAC to avoid detection, which is very easy to do.

- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).
- The game is fully vulnerable to speedhack (threat #3.1), without possibility for server-side detection.
- The game is vulnerable to score manipulation through memory (threat #3.2), but since this involves a time value, it is more difficult to perform.
- The lack of obfuscation means that it is possible to decompile the game SWF using any available flash decompiler. This makes the game vulnerable to reverse-engineering based attacks (threat #4.1, #4.2, #4.3 and #4.4) are possible. Finding the hash secret this way is also easy, and requires very limited code reading skills.

Detection:

- Server side cheat detection other than by making an estimate if the score falls in the possible range is not possible due to very minimalistic server interaction.

Skill & effort estimation

The game is vulnerable to two trival to perform fairness threats without significant opportunity for detection. The speedhack threat is by far the easiest to perform, requiring very limited skill and time. Also recovering the hash secret is trivial, which then enables the quickest and easiest of all attacks: modifying the score when it's sent to the server. Many people are capable of performing any of these cheats. Thus the required skill and effort are both low.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	Medium (2)	High (6)	High (10)

Using these outcomes we can determine the likelihood for fairness to be high. Since no high-score rankings are listed within the game or campaign, we can not observe if cheating actually occurred. But it's highly likely that there was at least some cheating.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Low (1)	Low (1)	Low (1)

Even though there is a fair chance that the game is being manipulated, we consider the impact low. No high-score rankings are shown, so for other players there's no way to see if people are playing dishonestly. So cheating will most likely not cause any reputation damage or player demotivation, and thus not significantly affect the campaign. Of course there's a good chance that prize assets are unfairly awarded to cheaters, but since there are so many prizes, and 50 prizes are given away every day, fanatical and capable fair players will most likely still be able to secure a prize for themselves. So Coop is probably factoring in some amount of loss due to cheating, but is willing to accept this risk.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (10)	Low (1)	Low (10)

Taking into account the high likelihood and the very low impact, this advergame was at relatively low risk with regard to fairness.

Case #6

Client company industry	Supermarket
Product/brand	Promotional mascot for supermarket
Campaign period	5 May 2012 – 16 June 2012
Competition	Yes
Prizes	1x Weekend to Paris, 60x soccer shirt
Competition-type	Main prize: Best time at end of period wins trip to Paris Week prizes: Top-10 at end of each week wins soccer shirt

To support a Euro Soccer Championship related promotion, a supermarket created a themed memory game consisting of three 4x4 levels. The goal is to complete all levels, and to complete the third level as soon as possible. The player with the best time at the end of the game period wins a 2-night trip to Paris. Each week the top-10 players get a soccer shirt.

Technical characteristics

Technology	JavaScript (DHTML)
Obfuscation	No
Architecture	Client/Server, game state & logic in server, view in client

One of the few advergames that fully implement the game state & logic on the server. The client takes care of the graphical presentation and communicates input events to the server. The server processes them and sends graphical updates back to the client. The server side of the game is developed using ASP.NET and the client side is developed in JavaScript using the AjaxControlToolkit library that works together with ASP.NET. Communication with the server is in plain text. The game is also available for mobile platforms.

Likelihood determination

Motivation

	Motivation
Fairness	Medium (2)
Privacy	Medium (2)

The campaign is exposed to the threat of cheating, the week prizes are not very valuable and might not motivate many cheaters to attempt to manipulate the game. The trip to Paris however has some significant value is might provide enough motivation for cheaters to try and influence game outcome. Therefore the motivation for fairness is medium.

Participants have to submit substantial private information including their name, age, address, phone-number and email and therefore the motivation related to privacy is medium.

A “Validation Error” coming (related to ASP.NET/AjaxControlToolkit data exchange) occurs frequently during game play, with the consequence that you have to start all over. A very frustrating experience that poses a serious threat to the quality of experience.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	High (1)	Medium (2)	Low (2)

<u>Control analysis</u>

1. Server-side game state & logic protects against most threats as there is almost nothing a potential cheater can influence on the client side.

<u>Vulnerability analysis</u>

Prevention

- Server-side game state & logic make speedhack, memory based score manipulation and tampering with results while they are being communicated impossible.
- AI-based solvers for memory-based games are available online and can be used (threat #1.2), but will need some adaptation. Such a bot can also be programmed to communicate directly with the server-interface, bypassing the game's graphical representation (a composite attack with threat #4.2)

- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).
- Advantages can be gained by adapting some non-logic parts of the client (threat #4.5). The memory-tiles can be changed to have significantly more contrast between each other, which makes memorization easier. This is especially useful in the third level (the only level that counts for your score) where the items on the tiles are all very similar. This kind of modification is impossible to detect and just requires visually adapting one jpeg file and a setup that loads the adapted file instead of the original. It's also possible to remove some of the animations and delays in the client side Javascript code, which might allow a cheater to flip tiles slightly faster than without modification. While both attacks are not hard to perform to a user with good computer skills, they require more out of the box thinking as they deviate from the usual threat opportunities.

Skill & effort estimation

The most effective way to manipulate this advergame is by artificial intelligence, which requires a significant amount of skill and some time. Therefore we rate skill as high and effort as medium.

Likelihood

	Motivation	**Vulnerability**	**→ Likelihood**
Fairness	Medium (2)	Low (2)	Low (4)

Using these outcomes we can determine the likelihood for fairness to be low. Despite the low likelihood we did actually observe some cheating, most likely using AI based solvers.

Impact prediction

	Mission	**Asset**	**→ Impact**
Fairness	Medium (2)	Medium (2)	Medium (5)

If successfully cheating does take place there's a high chance of the trip to Paris being given away to unfair players and if obvious, this could lead to some reputation damage. Also unrealistically high times in the high-score table might demotivate other players from participating, which might affect campaign success. Thus the impact for mission and asset are both medium.

Risk

	Likelihood	**Impact**	**→ Risk**
Fairness	Low (4)	Medium (5)	Low (17)

Taking into account the low likelihood and medium impact, this advergame was at relatively low risk with regard to fairness.

Case #7

Client company industry	IT security
Product/brand	Anti-virus
Campaign period	5 December 2011 – 16 January 2012
Competition	Yes
Prizes	1x Luxurious trip to Shanghai for 2 people 6x Samsung Galaxy Tab 10.1 + Antivirus product (EUR 500)
Competition-type	Main prize: Highest tagteam score at end of period Week prizes: Highest score in individual game at end of each campaign week

For the launch of their new anti-virus product a Facebook social advergame campaign was developed to illustrate the combination of “muscle power” and “cloud intelligence” of their new product using game elements. The game is played in a team of two Facebook friends where one player plays a “strength” game and the other player plays an “intelligence” game. The “strength” game involves clicking on randomly appearing viruses as soon as possible (the sooner you click on one, the higher the number of points you get) where the total score is the addition of the all the points for each virus. The “intelligence” game involves solving a sudoku-like mathematical puzzle (with some elements of chance) as quickly as possible. The game involved several rounds, and a bonus round against another team, that gave opportunities to earn bonuses or have their total team score doubled. The player with the highest individual game score (either strength or intelligence) at the end of each campaign week wins a Samsung Tablet and the best performing team at the end of the game period wins a luxurious trip to Shanghai.

Technical characteristics

Technology	Flash 10 (AVM2)
Obfuscation	Yes, SecureSWF
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash 10. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server through interface in plain-text but integrity protected with a challenge-response based hash, using a GET request (data contained within URL). No intermediate results communicated, except for a periodic (every 60 seconds) empty keep-alive call for the session, but the start of every game is signaled to the server (and used to obtain the hash-challenge).

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Medium (2)

Very high value prizes are given away during this campaign, most significantly the main prize, a luxurious trip to Shanghai. The week prizes are also valuable and together they will strongly motivate potential dishonest players to attempt to cheat. Therefore the motivation for fairness is high.

In order to play the game a user has to give permission for the game to collect the users full name, Facebook ID, e-mail address and location, as well as his friends list, both during game play as offline. Permission also needs to be granted for posting updates on the players wall as well as on his friends wall. Even though the amount of personal data collected is relatively limited (but some of it privacy-sensitive), the permissions for posting status updates could be seriously abused when obtained by adversaries. Therefore the motivation for privacy is a firm medium.

Vulnerability

	Skill	Effort	→ Vulnerability
Fairness	Medium (2)	Medium (2)	Medium (4)

Control analysis

1. Communication is protected by a challenge-based hash implemented by a (identifier scrambled) method in the games.gameOverScreen class that hashes the score (multiplied by 4) together with a secret ("apennootjes_") and the value from the challenge. This makes tampering more difficult (easier to detect). The dynamic challenge makes it also impossible to replay the a particularly high score with corresponding hash for later rounds.
2. The flash file is protected against decompilation / secret reverse engineering by the SecureSWF obfuscator.
3. The hash secret is not contained within the SWF as a literal string, but is dynamically constructed at runtime, which makes extraction more difficult, especially with obfuscation.
4. Signaling when a game starts being played makes it potentially possible to do server-side time measurements which can be used to analyze score submissions and detect cheating.

Vulnerability analysis

Prevention

- The game is highly vulnerable to speedhack (threat #3.1) which makes it possible to achieve a very high score for both games. No client-side countermeasures against speedhack are taken, but server side detection might be possible (see detection).
- The game is highly vulnerable to memory-based score manipulation (threat #3.2), no client-side countermeasures exist.
- The chosen obfuscation method will prevent off-the-shelf decompilers from decompiling the game logic of this game (thus limited vulnerability to threat #4.1 and #4.2). However this obfuscation does not prevent people with knowledge of the AVM2 instruction set from reverse engineering the game and retrieving its hash secrets as the game can be disassembled (threat #4.3), and the obfuscation is rather weak.. This however, requires significant skills.
- It is possible to tamper with the score data when it's communicated to the server (threat #2.1). However this depends on recovering the hash function and its secret by reverse engineering, which requires significant skills.
- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).

Detection:

- The design of the score function results in a specific relation between the height of the score and relatively specific bounds for the playtime under which it can be achieved, especially for high scores. The game interaction is designed so that the playtime can be measured at the server side. This allows for powerful opportunities for cheat detection by analysis of score submissions which can potentially detect most speedhack and memory manipulation cheats. It is likely that this was designed on purpose and that detection is done at the server side, which reduces the vulnerability to medium. If no such advanced manual score verification is done by the company, the vulnerability would be high.
- However any potential cheater who analyzed the score function and properly understands this relation is able to emulate this. This is impossible to do with a speedhack cheat, and very difficult with a memory manipulation cheat, but easy to time if you have recovered the hash function including its secret. It's now easy to write a script that submits any score with a correct hash and with the right timings. If well executed this is absolutely impossible to detect by the company, but it also requires significant skills.

Skill & effort estimation

The skill and effort required could be either medium or high depending on whether the vulnerability is high, but since we expect that score analysis is done, we estimate the vulnerability to be medium. In fact performing a successful and undetectable attack requires a significant amount of skill and insight, but since the campaign is available to players from 16 different countries it's a certainty that some potentially dishonest players will have the right skill-set.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	Medium (4)	High (10)

Using these outcomes we can determine the likelihood for fairness to be high. We have observed several teams that clearly cheated throughout the game because of impossibly high scores, or because they were using "fake" Facebook profiles. We've also observed such cheaters being actively removed from the high score lists.

Impact prediction

	Mission	Asset	→ Impact
Fairness	High (3)	Medium (2)	High (8)

The most obvious impact potential of these risks is that the company will give away prizes worth thousands of euro's to unfair winners, a significant loss of assets. Consequences of these risks can be much wider though: If the news gets out that, the second biggest anti-virus and security company in the world, can not even secure its own game and provide fairness for a simple online advergame and gives away prizes to cheaters, this can have very serious consequences for the reputation of this company. Even though the development and execution of the campaign was outsourced to a different company, it is the company's name and reputation at stake. Next to this, observed cheating and impossible-to-beat scores will demotivate and frustrate fair players, which might result in less participation or reflect on brand image, and therefore may have a significant impact on campaign effectiveness. Because of all this we rate the mission impact as high.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (10)	High (8)	High (46)

Taking into account the high likelihood and high impact, this advergame was at high risk with regard to fairness.

Case #8

Client company industry	Meat produce
Product/brand	Sausage brand
Campaign period	1 December 2011 – 11 January 2012
Competition	Yes
Prizes	1x Weekend Barcelona for 2 42x Set of tapas products
Competition-type	Main prize: Player with highest score at end of period Daily prize: Random draw

An advergame campaign created to promote that dried sausages are also a good starting point for Spanish tapas. It's a simple skill-based endurance game where one has to keep feeding three men tapas on time. Each time one of the men requests a piece of tapas, the player has to use his mouse to bring it to his mouth on time, and the time between requests will get shorter over time as the game progresses. Five failures (not servicing a request on time, or dropping the tapas at the wrong place) are allowed, and then it's game over. The score is the number of tapas served. The highest score at the end of the period wins a luxurious tapas-themed weekend to Barcelona. On each campaign day, every participant has a chance of being randomly selected to win the day prize: a tapas set.

Technical characteristics

Technology	Flash 9 (AVM1)
Obfuscation	No
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash 9, using the older AVM1. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. No intermediate results are communicated. Score information is encoded and submitted at the end of the game to the campaign site using a GET request.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Low (1)

A very attractive and valuable prize is given away so the player with the highest score, which will provide good motivation for potentially dishonest players to cheater. Therefore the motivation to fairness is high.

The motivation for privacy is low as the data requested and collected only consists of the player's name and e-mail address.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Low (3)	Low (3)	High (6)

Control analysis

1. The game result (score, name and e-mail, together with the literal string “fairplay”) is encoded with some rudimentary “encryption” function with the goal of providing fairness.

Vulnerability analysis

Prevention

- The game is vulnerable to speedhack (threat #3.1) which can make the game a lot easier to play when the interval between requests decreases. Although by slowing the game down, it will take even longer for this endurance game to achieve a high score.
- This key can be relatively easily automated using mouse automation scripts (threat #1.1) but this will

require skill and effort.

- The game is trivially vulnerable to memory based score manipulation (threat #3.2). The implementation of the game and its score function provide for the easiest setting to perform such a cheat.
- This encoding function is so weak that just by analysis of how your scores are being encoded, you might be able to determine how it's done (without access to the source). The "key" is computed by adding up the ASCII values of the players textual IP-address string. Individual characters from the result string are then encoded numerically by taking their ASCII value and adding this to the key. The key is then used to compute a numeric separator which joins each individual numerically encoded character from the results string. Thus the same separator is used for the whole encoded data string, and therefore this is very easy to recognize, leaving numeric values that only need a fixed constant (the key) subtracted to reveal their original ASCII value.
- The lack of obfuscation means that it is possible to decompile the game SWF using any available flash decompiler. This is easy to perform, and only limited code reading skills are required to then localize the "encryption function" and to understand how the results string is encoded. Thus vulnerable to threats #4.1, #4.2, #4.3 and #4.4.
- The game is trivially vulnerable to tampering with score data when it's submitted to the server (threat #2.1). Because a GET request is used, communication does not need to be intercepted but only the URL loaded has to be modified (or entered manually). Encoding a valid score string perhaps requires some minor programming skills but can also be done manually.
- Private data not communicated over HTTPS but encoded with previously mentioned "encryption function" (threat #2.4)
- The uses weak player authentication as it just relies on an e-mail address and a name (threat #5.1).

Detection:

- There are no opportunities for detection whatsoever, in principle any score is possible if one lasts the game long enough. No server-side time measurement is possible, so they can not match a high score to a long playtime.

<u>Skill & effort estimation</u>

Since this game is vulnerable to the highly trivial to perform speedhack and memory-based score manipulation threats, the skill and effort are both low.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	High (6)	High (12)

Using these outcomes we can determine the likelihood for fairness to be high. Since any score could potentially be legitimate with respect to range of scoring function, it's hard to determine whether cheating actually occurred. However we do have one very important indicator that it did happen: Only a few minutes before the deadline a new high score, by a new player not seen before, arrived. This is kind of last-minute high score submission (similar to last-second sniper-bidding with online auctions) is a very typical cheating pattern of behavior.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Low (1)	Medium (2)	Low (3)

The campaign is at risk of giving away a valuable prize to an unfair winner, not just a loss of assets but also potentially damaging to the brand's reputation. However, the industry the company operates in, is not context-sensitive at all, so the extent of potential reputation damage will be limited. But players might get potentially demotivated or frustrated by (unrealistically) high scores obtained by cheaters which might result in less participation or a negative attitude towards the campaign. Therefore we rate mission as low and impact as medium.

For this game the luxurious trip to Barcelona was most likely awarded to a cheater. Even the company must have realized that this was probably occurring (although without a way to prove it, they didn't want to interfere it seems) and they decided to create a second prize worth EUR 500, for the runner-up who had been extensively playing this game, and actively participating on Facebook about it for most of the campaign period. A clear example of a financial risk result.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (12)	Low (3)	Medium (28)

Taking into account the high likelihood and low impact, this advergame was at medium risk with regard to fairness.

Case #9

Client company industry	Consumer foods
Product/brand	Tea
Campaign period	5 April 2012 – 15 June 2012
Competition	Yes
Prizes	3x Luxury champagne breakfast for two in hot air balloon flight (EUR 550)
Competition-type	Highest three scorers at end of period win

A Facebook-based advergame was used to support such a campaign for a tea brand. In the game the player has to climb a large pyramid shaped mountain of tea by pressing the arrow keys according to an onscreen pattern. There are five levels and with each level the maximum allowed interval between two successive arrow key presses is decreased. If you're too slow, you ll drop and lose a live (and you have five). Also with each higher level, extra distractive features are added. The three players with the highest score at the end of the game period win 2 tickets each for a champagne breakfast during a hot air balloon flight (worth EUR 550).

Technical characteristics

Technology	Flash (AVM2)
Obfuscation	Yes, DoSWF
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash 10. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server through several scripts on the website in plain-text, but integrity protected with a checksum value. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Medium (2)

A high value prize is given away to the three players with the highest score, providing strong motivation for cheaters that might try to manipulate the game to submit a false score. Therefore the motivation for fairness is high.

The amount of private data collected is low as the only data requested and collected through Facebook are the player's full name and user ID (both of which are considered public by Facebook) and its e-mail address. However the game also asks permission to post messages on your wall, which can pose a risk if the permission token compromised, therefore we rate the motivation for privacy as medium.

Vulnerability

	Skill	Effort	→ Vulnerability
Fairness	Low (3)	Low (3)	High (6)

<u>Control analysis</u>

1. The score submitted is protected by a simple checksum value implemented in the game over function (name is scrambled due to obfuscation) that uses a basic arithmetic formula ((score * 8321 + 2 >> 4) % 331) for its computation.
2. The flash file is obfuscated with DoSWF to protect the game against reverse engineering.
3. Some extra game variables (level reached, number of lives left, score before bonus and average reaction time) on which the final score depends are submitted together with the score that can possibly aid in the detection of false scores.
4. A call to a script is made at the start of the game which allows for the measurement of the playtime on the server-side, possibly used for detection.

5. Data is communicated over HTTPS.

Vulnerability analysis

Prevention

- The game is highly vulnerable to speedhack (threat #3.1) which makes achieving a very high score possible without difficulty, with no client-side countermeasures in place.
- The game is highly vulnerable to score manipulation through memory (threat #3.2), no client-side countermeasures in place.
- The obfuscation method in the second version will prevent off-the-shelf decompilers from decompiling the game logic of this game. Removing the obfuscation layer requires significant technical knowledge, but a cheater who possesses this knowledge can accomplish this in about 15 minutes. Without the possibility to reverse engineer the game the checksum function can not be determined directly. Thus reduced vulnerability for threat #4.1, #4.2, #4.3 and #4.4.
- The checksum function is weak. It only takes into account the total score and none of the other variables or a challenge, which means it's vulnerable to a replay attack. If a particularly high score is achieved the checksum value can be recorded, and can be used to replay the same score with a valid checksum for two of your friends. The checksum has also only 331 different outcomes.
- The game is vulnerable to tampering with the score data (threat #2.1), but the attacker must understand how the checksum is calculated in order not to be detected. This could require the skills to deobfuscate and reverse engineer the game.

Detection:

- The server-side time measurement is too coarse and therefore useless in detecting a speedhack or memory manipulation. Players are legitimately allowed to pause between levels and therefore there's no relation between the play time and the score achieved. A long play time for a high score could simply mean a legitimate player taking a break between two levels.
- The extra game variables submitted might only help in detecting memory-based score manipulation, where the score is changed through memory, but the other game variables are not. In this case the submitted score might not be in line with the other game variables (the other game variables as input to the scoring function would result in a different score). For tampering with the score submission when communicated, the cheater must understand the score function to also change the other game variables submitted to match with the score. If the cheater has reverse engineered the game, this is not difficult to do. Finally, scores influenced by speedhack will always have their checksum and other game variables correct and satisfy any checks on score validity, and therefore there is again absolutely no possibility to detect a speedhack here.
- Any cheater that has fully reverse engineered the game and obtained full knowledge of the score and checksum function can create an intelligent script for gameplay simulation.

Skill & effort estimation

The game is vulnerable to the trivial speedhack without opportunity for detection, and vulnerable to memory-based score manipulation with some opportunity for detection. Even though the game is obfuscated, somebody with the right skills, although significant, can remove this obfuscation within about 15 minutes and employ array of more sophisticated attacks. Thus both the skill and effort required are low.

Likelihood

	Motivation	**Vulnerability**	**→ Likelihood**
Fairness	High (3)	High (6)	High (12)

Using these outcomes we can determine the likelihood for fairness to be high. Large scale cheating has been observed with many scores at the theoretical maximum (which requires an average reaction time of under 50msec without any mistake, not possible by humans) or very close to it (most likely speedhack) and even scores that are above the theoretical maximum and can therefore never be achieved (memory hack or tampering with communication data without understanding the score function). No cheaters have been removed so far.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Medium (2)	Medium (5)

The campaign is at risk of giving away a valuable prize to an unfair winner, not just a loss of assets but also potentially damaging to the organizations reputation. This impact is bound to occur, as there's simply no possibility for the organization to draw the line between what scores have been honestly achieved and what scores haven't. But there is no context-sensitivity to the brands. At the same time players might get demotivated or frustrated by the (unrealistically) high scores obtained by cheaters which might result in less participation or reflect on their brand image. Therefor we rate the impact for both asset and mission as medium.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (12)	Medium (5)	High (39)

Taking into account the high likelihood and high impact, this advergame was at high risk with regard to fairness.

Case #10

Client company industry	Online marketplace
Product/brand	Daily deals
Campaign period	9 Jan 2012 – 16 Jan 2012
Competition	Yes
Prizes	7x Flight lesson (value EUR 150)
Competition-type	Daily high-score wins

An advergame was created to promote a new daily deals service of an online market place. In the game one is presented three series of random pictures for 30 seconds which one has to remember, and then a quiz is presented with a few questions about these pictures which have to be answered as quickly as possible. The score is determined by the number of correct answers and your time. The best score of the day wins a flight lesson worth EUR 150.

Technical characteristics

Technology	JavaScript (DHTML)
Obfuscation	None
Architecture	Client/Server, game state & logic on server

One of the few advergames that fully implement the game state & logic on the server. The client takes care of the graphical presentation and communicates input events to the server. The server processes them and sends graphical updates back to the client. Communication with the server is in plain text.

Likelihood determination

Motivation

	Motivation
Fairness	Medium (2)
Privacy	Low (1)

The game provides relatively strong motivation because of the valuable and attractive prizes. Therefore we rate the motivation for fairness as medium.

The only data requested is a name and e-mail address, and therefore the motivation with respect to privacy is low.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	High (1)	High (1)	Low (1)

Control analysis

1. Server-side game state & logic protects against most threats as there is almost nothing a potential cheater can influence on the client side.

Vulnerability analysis

Prevention

- Server-side game state & logic make speedhack, memory based score manipulation and tampering with results while they are being communicated impossible.
- An AI-based attack is possible that automatically plays, tries and learns the correct quiz answers for each of the randomly selected pictures. This can be done by replacing the game client with an intelligent script that communicates directly with the server game interface. This has to be custom written and this will require a significant amount of effort and skill. Thus some vulnerability to the composite attack of threat #1.2 and #4.2.
- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).
- The uses weak player authentication as it just relies on an e-mail address and a name (threat #5.1).

Skill & effort estimation

The only viable attack is to use custom program an artificial intelligent script that can automatically learn the right answers and then play the game automatically. This requires both high skill and high effort, and thus this game has low vulnerability.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	Medium (2)	Low (1)	Low (2)

Using these outcomes we can determine the likelihood for fairness to be very low. And correspondingly no (obvious) cheating was observed during this campaign.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Low (1)	Low (3)

The campaign is at risk of giving away prizes to an unfair winner, not just a loss of assets but also potentially damaging to the organizations reputation. However the total value of the prizes is most likely only a fraction of the cost for the whole campaign and therefore the asset impact is low. The potential for some damage to the campaign and the companies reputation exists and therefore the mission impact is rated as medium.

Risk

	Likelihood	Impact	→ Risk
Fairness	Low (1)	Low (3)	Low (6)

Taking into account the low likelihood and low impact, this advergame was at low risk with regard to fairness.

Case #11

Client company industry	Not for profit
Product/brand	Battery collection
Campaign period	28 November 2011 – 28 December 2011
Competition	Yes
Prizes	1x Travel check (EUR 1500) 400x gift certificate of choice (EUR 100)
Competition-type	Highest score at end of period wins travel check Weekly raffle for 100 gift certificates

To promote the awareness of old battery collection an advergame was created that involved a quiz game on the subject. Random questions were presented and had to be answered as quickly as possible. Depending on your score you would get between 1 and 4 raffle tickets used for a weekly raffle. You can only play the game once, but for every friend you invite you get another opportunity. The best score at the end of the period won a travel check worth EUR 1500.

Technical characteristics

Technology	Flash (AVM1)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server through several scripts on the website in plain-text. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Low (1)

A very high value prize is given away at the end of the campaign which will strongly motivate cheaters to participate, therefore the motivation for fairness is high.

To participate only your name and an email-address are required and thus the motivation for privacy is low.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Low (3)	Low (3)	High (6)

<u>Control analysis</u>

1. No controls exist.

<u>Vulnerability analysis</u>

Prevention

- The game is vulnerable to tampering with the score data (threat #2.1) and no integrity mechanism exists.
- The game is highly vulnerable to speedhack (threat #3.1).
- The game is vulnerable to score manipulation through memory (threat #3.2), but this is difficult considering it is a time value.
- The lack of obfuscation means that it is possible to decompile the game SWF using any available flash decompiler. This makes the game vulnerable to reverse-engineering based attacks (threat #4.1, #4.2, #4.3 and #4.4) are possible.
- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).

- The uses weak player authentication as it just relies on an e-mail address and a name (threat #5.1).
- When a player plays the game, a call is made to a server script to reduce the number of plays that a player has left (one for every person invited). However, by suppressing this call, the game can be played as often as a cheater wishes, allowing him to generate many raffle tickets. The game is thus vulnerable to interface manipulation (threat #5.2).

Detection:

- The game provides no opportunity for detection.

<u>Skill & effort estimation</u>

This game is vulnerable to 2 highly trivial to perform threats, the easiest of which is just changing the score to any value you want on submission, therefore, on the basis of these two threats, the requires skill and effort are both low.

Likelihood

	Motivation	**Vulnerability**	**→ Likelihood**
Fairness	High (3)	High (6)	High (12)

Using these outcomes we can determine the likelihood for fairness to be high. This was supported by our observations that showed a large amount of cheating with many impossibly low timings.

Impact prediction

	Mission	**Asset**	**→ Impact**
Fairness	Medium (2)	Medium (2)	Medium (5)

The campaign is at risk of giving away many prizes to an unfair winners including the high value main prizes, not just a loss of assets but also potentially damaging to the organizations reputation. The asset value is likely to make up a significant share of campaign costs and therefore the asset rating is medium. The potential for some damage to the campaign and the companies reputation exists and therefore the mission impact is rated as medium as well.

Risk

	Likelihood	**Impact**	**→ Risk**
Fairness	High (12)	Medium (5)	High (39)

Taking into account the high likelihood and medium impact, this advergame was at high risk with regard to fairness.

Case #12

Client company industry	Cheese
Product/brand	Cheese brand
Campaign period	2 Jan 2012 – 27 February 2012
Competition	Yes
Prizes	8x EUR 400 travel check
Competition-type	Weekly high-score wins

To promote a cheese brand a simple advergame was developed where one had to click on the differences between two pictures as quickly as possible. In order to participate, you need to buy the product to get a participation code. The quickest time at the end of each week wins a travel check worth 400 euro.

Technical characteristics

Technology	JavaScript (DHTML)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

Browser-based game developed in JavaScript (DHTML). Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server through several scripts at the campaign server in plain-text. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Medium (2)

The high value prizes, and the possibility to win more than one, provide a high motivation for cheating.

When participating, a substantial amount of privacy information had to be entered including name, address, mail, phone number and birth date. This provides medium motivation for privacy related threats.

Vulnerability

	Skill	Effort	→ Vulnerability
Fairness	Low (3)	Low (3)	High (6)

Control analysis

1. No controls exist.

Vulnerability analysis

Prevention

- Since the locations where to click in the find the differences game are always the same, it would be very easy to automate these mouse clicks. Therefore the game is vulnerable to threat #1.1.
- The game is vulnerable to tampering with the score data (threat #2.1) on communication, and no countermeasures exist.
- The game is highly vulnerable to speedhack (threat #3.1) with regard to time.
- The game is fully vulnerable to debugger manipulation (threat #3.3).
- The lack of obfuscation means that it is possible to view the source. This makes the game vulnerable to reverse-engineering based attacks (threat #4.1, #4.2) are possible. Vulnerability to simulation (#4.3) is reduce as product codes are required.
- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).

Detection:

- There are no opportunities for detection.

Skill & effort estimation

This game is vulnerable to a number of trivial to perform threats which require very little skill or effort. Therefore the vulnerability is high.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	High (6)	High (12)

Using these outcomes we can determine the likelihood for fairness to be high. However since no high scores are being shown, it is impossible to determine how much cheating actually occurred.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Low (1)	Medium (2)	Low (2)

The biggest impact for this campaign is to give away prizes to unfair winners, and these also have a considerable value, thus the asset impact is medium. However, since no high scores are shown, there's little potential impact for reputation or campaign damage, as potential problems are simply not visible. Therefore the mission impact is low.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (12)	Low (2)	Medium (20)

Taking into account the high likelihood and low impact, this advergame was at medium risk with regard to fairness.

Case #13

Client company industry	Coffee & Tea retail
Product/brand	Coffee & Tea shop
Campaign period	12 October 2011 – 16 October 2011
Competition	Yes
Prizes	10 different prizes with a total value of EUR 600.
Competition-type	Top-10 best players at end of campaign win

To promote a coffee & tea shop, a simple advergame was created where one has to shake a cup of tea as quickly as possible for the tea to get enough color without any tea spilling around the borders. In three levels the tea cup takes different shapes that make shaking more difficult. The top-10 players at the end of the game win different relaxation and tea related prizes with a total value of EUR 600.

Technical characteristics

Technology	Flash (AVM2)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server through several scripts on the website in plain-text. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Low (1)

The relatively high-value prizes can provide relatively high motivation for cheaters to try and influence the game outcome. It would be possible to win multiple prizes by having multiple identities in the top-10.

With regard to privacy, only the players nickname and email-address are collected, thus the motivation for privacy is low.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Low (3)	Low (3)	High (6)

<u>Control analysis</u>

1. The score is “encrypted” using a simple transformation in Main.encryptScore().

<u>Vulnerability analysis</u>

Prevention

- The game might be vulnerable to some automation attack for the shaking in the different levels, which can be recorded with mouse recording software and then optimized (threat #1.1).
- The game is vulnerable to tampering with the score data (threat #2.1), but the attacker must understand how the score is transformed by encryptScore().
- The game is highly vulnerable to speedhack (threat #3.1).
- The game is vulnerable to score manipulation through memory (threat #3.2), but this might be difficult considering it is a time value.
- The lack of obfuscation means that it is possible to decompile the game SWF using any available flash decompiler. This makes the game vulnerable to reverse-engineering based attacks (threat #4.1, #4.2, #4.3 and #4.4) are possible. Finding the workings of the “encryption” function this way is also

easy, and requires very limited code reading skills.
- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).
- The uses weak player authentication as it just relies on an e-mail address and a name (threat #5.1).

Skill & effort estimation

Considering that this game is vulnerable to the trivial to perform speedhack cheat, the skill and effort required to cheat this game are both low. And thus the vulnerability is high.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	High (6)	High (12)

Using these outcomes we can determine the likelihood for fairness to be low. This corresponds with our observations, as we observed some unlikely scores in the highscore list, that are likely the result of cheating.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Low (1)	Low (3)

The biggest impact for this campaign is to give away prizes to unfair winners, but their value is likely limited in comparison to the full costs of the campaign and thus the asset impact is low. Due to the prominent high-score table cheating can be quite obvious, reducing the fun that people have from participating in this game, and possibly causing damage to the companies brand image. Therefore we rate the mission impact as medium

Risk

	Likelihood	Impact	→ Risk
Fairness	High (12)	Low (3)	Medium (28)

Taking into account the high likelihood and low impact, this advergame was at medium risk with regard to fairness.

Case #14 optimel

Client company industry	Dairy
Product/brand	Drink yoghurt
Campaign period	1 – 17 June 2012
Competition	Yes
Prizes	Air balloon trip (EUR 300)
Competition-type	Lottery among top-100 scores

To add some fun to their website, a dairy company created a simple father's day advergame called super-dad. A simple 8-bit console like game where one has to use the arrow keys to navigate super-dad through a concourse where obstacles have to be avoided and yoghurt items have to be caught for points. At the end of the period, a player randomly selected from the best 100 players wins an air balloon trip (EUR 300).

Technical characteristics

Technology	JavaScript (HTML5)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

The only HTML5 game found in use during this study. Simple browser-based game entirely developed in JavaScript. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server in plain-text. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	Low (1)
Privacy	Medium (2)

Considering the low-value of the prize and the fact that it is a lottery among the top-100 best players, we estimate the motivation for cheaters to be low.

In order to participate an account has to be created on the companies website, which requires substantial privacy sensitive information such as name, address, age and email-address. Therefore we consider the motivation for privacy to be medium.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Low (3)	Medium (2)	High (5)

Control analysis

1. The score is protected using a MAC
2. An organizational control is in place which in the form of selecting the winner random from the top-100 players.

Vulnerability analysis

Prevention

- The game is vulnerable to tampering with the score data (threat #2.1), but the attacker must understand how the checksum is calculated in order not to be detected.
- The game is highly vulnerable to speedhack (threat #3.1).
- The game is highly vulnerable to score manipulation through memory (threat #3.2), no client-side countermeasures in place.
- The game is fully vulnerable to debugger manipulation (threat #3.3).
- The lack of obfuscation means that it is possible to view the game source. This makes the game vulnerable to reverse-engineering based attacks (threat #4.1, #4.2, #4.3) are possible. Finding the hash secret this way is also easy, and requires very limited code reading skills.

- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).

Detection:

- No opportunities for detection exist.

<u>Skill & effort estimation</u>

The game is technically very vulnerable, with many ways to cheat the game without requiring much skill or effort. However the lottery-control means that in order for a cheater to increase his chances, he will have to participate under many different accounts, which does mean a considerable effort. Therefore we rate the skill as low, and effort as medium. Giving the game a high vulnerability.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	Low (1)	High (5)	Medium (5)

Using these outcomes we can determine the likelihood for fairness to be medium. Some cheating was indeed observed in the high-score lists with clearly impossible scores.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Low (1)	Low (3)

The chance of giving away the prize to an unfair player is reduced by the fact that it depends on chance, but it still exists. However the value is relatively low, therefore we rate the asset impact as low. With regard to mission, the fact that scores are published in a high-score table makes it obvious to see that the game is cheated on, which can reduce the effectiveness of the campaign and potentially lead to reputation damage. Therefore we rate the mission impact as medium.

Risk

	Likelihood	Impact	→ Risk
Fairness	Medium (5)	Low (3)	Low (14)

Taking into account the medium likelihood and low impact, this advergame was at low risk with regard to fairness.

Case #15

Client company industry	Yoghurt brand
Product/brand	Yoghurt
Campaign period	10 April 2012 – 31 April 2012
Competition	Yes
Prizes	1x Giftfor2 voucher (EUR 339) and 9x Giftfor2 voucher (EUR 45)
Competition-type	Top-10 best players wins

To promote a brand of yoghurt an advergame was made based on a memory game themed with items related to feeling good. The memory game consists of three levels with increasing grids. The goal is to complete all the levels as soon as possible. The fastest players at the end of the game period get a prize.

Technical characteristics

Technology	Flash (AVM1)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server using in plain text. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	Low (1)

The prizes are valuable and there's an opportunity for repeat winning, therefore the motivation with regard to fairness is high.

With respect to privacy, the amount of information collected is fairly limited consisting of users name and email-address. Thus the motivation with respect to privacy is low.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Low (3)	Low (3)	High (6)

Control analysis

1. No controls exist.

Vulnerability analysis

Prevention

- The simple memory-based game is vulnerable to automatic solving using an artificial intelligent memory-game solver (threat #1.2).
- The game is fully vulnerable to tampering with the score data (threat #2.1).
- The game is highly vulnerable to speedhack (threat #3.1).
- The game is vulnerable to score manipulation through memory (threat #3.2), but this is difficult considering it is a time value.
- The lack of obfuscation means that it is possible to decompile the game SWF using any available flash decompiler. This makes the game vulnerable to reverse-engineering based attacks (threat #4.1, #4.2, #4.3 and #4.4) are possible. But this is unnecessary as it contains no secrets.
- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).
- The uses weak player authentication as it just relies on an e-mail address and a name (threat #5.1).

Detection:

- No possibilities for detection exist.

Skill & effort estimation

The game is vulnerable to three trivial to perform threats without any opportunity for detection. The easiest of which is simply changing the time value when it's communicated to the server. Therefore we rate both the skill and effort required as low, and the vulnerability as high.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	High (6)	High (12)

Using these outcomes we can determine the likelihood for fairness to be high. This corresponds with our observations. We observed several impossibly low timings that are a clear indication of cheating. No such cheaters have been removed.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Medium (2)	Medium (5)

The most obvious impact is giving away prizes to unfair winners. The total value of all the prizes will make up quite a considerable share of the total campaign costs, and therefore we rate the potential asset impact as medium. With regard to mission, the fact that scores are published in a high-score table makes it obvious to see that the game is cheated on, which can reduce the effectiveness of the campaign and potentially lead to reputation damage. Therefore we rate the mission impact as medium.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (12)	Medium (5)	High (39)

Taking into account the high likelihood and medium impact, this advergame was at high risk with regard to fairness.

Case #16

Client company industry	Chewing gum
Product/brand	Chewing gum brand
Campaign period	6 March 2012 – 19 September 2012
Competition	Yes
Prizes	26x X-box 360 Kinect (EUR 232) daily: 2 cinema tickets (EUR 15)
Competition-type	Xbox: randomly selected from weekly top-20

An advergame was created to promote the fact that a brand of chewing gum refreshes your breath. Clouds of bad breath have to be shot by pieces of chewing gum while packages of chewing gum have to be avoided. The daily high score wins 2 cinema tickets, and every week a winner is randomly drawn from the top-20 players who wins an X-box 360 Kinect (EUR 232).

Technical characteristics

Technology	Flash (AVM2)
Obfuscation	None
Architecture	Client/Server, game state & logic located in client

Browser-based game entirely developed in Flash and available through Facebook. Simple client-server based architecture where the game client fully implements the game, and the server-logic is only used for user and score registration. Communication with server using flash remoting protocol with integrity protected by hash. No intermediate results communicated, only the final result together with user registration information.

Likelihood determination

Motivation

	Motivation
Fairness	High (3)
Privacy	

The prize is of considerable value, but what makes it even more attractive is that so many of these prizes are given away, which gives ample opportunity for repeat winning. Therefore the motivation with regard to fairness is high.

With respect to privacy, the amount of information collected is fairly limited consisting of only the users facebook id, name and email-address. Thus the motivation with respect to privacy is low.

Vulnerability

	Skill	**Effort**	**→ Vulnerability**
Fairness	Medium (2)	Medium (2)	Medium (4)

<u>Control analysis</u>

1. Data is communicated using Flash Remoting Protocol which makes it slightly harder to modify, and protected with various MACs.
2. Additional score data is sent on completion, which can enables some basic forms of detection.
3. Organizational control (lottery among top-20).

<u>Vulnerability analysis</u>

Prevention

- The game has a glitch where if the game screen is kept out of focus for a long time, bad breaths to shoot will accumulate on the screen, and at that point it's almost impossible to miss them anymore (threat #1.3).
- The game is vulnerable to tampering with the score data (threat #2.1), but the attacker must understand how the MAC is calculated in order not to be detected.
- The game is vulnerable to speedhack (threat #3.1) which makes the game a lot easier.
- The game is vulnerable to score manipulation through memory (threat #3.2), but multiple variables

have to be changed. No client-side countermeasures in place.

- The lack of obfuscation means that it is possible to decompile the game SWF using any available flash decompiler. This makes the game vulnerable to reverse-engineering based attacks (threat #4.1, #4.2, #4.3 and #4.4) are possible. Finding the hash secrets this way is also easy, and requires very limited code reading skills.
- Privacy sensitive data is not transmitted over HTTPS and could be intercepted (threat #2.4).

Detection:

- Some additional score data is sent on game completion which can aid a little in detecting cheaters, such as the game time and the number of bad breaths hit. However the game time isn't of much use as the game can be paused without consequence, and the number of bad breath hits required for a certain score is easy to guess.

Skill & effort estimation

The game is vulnerable to speedhack and memory manipulation, both of which are easy to perform, but for memory manipulation there is a complication that make it slightly more difficult as multiple score variables have to be changed using memory manipulation. The lottery among the top-20 means that a cheater has to try many weeks, or with multiple identities to increase his chances. Therefore we rate the skill and effort as medium for both.

Likelihood

	Motivation	Vulnerability	→ Likelihood
Fairness	High (3)	Medium (4)	High (10)

Using these outcomes we can determine the likelihood for fairness to be high. We've indeed observed a number of cheaters. For example because the score did not match with the number of bad breaths shot.

Impact prediction

	Mission	Asset	→ Impact
Fairness	Medium (2)	Medium (2)	Medium (5)

The most obvious impact is giving away prizes to unfair winners. The total value of all the prizes will make up quite a considerable share of the total campaign costs, and therefore we rate the potential asset impact as medium. With regard to mission, the fact that scores are published in a high-score table makes it obvious to see that the game is cheated on, which can reduce the effectiveness of the campaign and potentially lead to reputation damage. Therefore we rate the mission impact as medium.

Risk

	Likelihood	Impact	→ Risk
Fairness	High (10)	Medium (5)	Medium (35)

Taking into account the high likelihood and medium impact, this advergame was at medium risk with regard to fairness.