Masarykova University

Faculty of informatics

A Client Honeypot

Master's Thesis

Bc. Vladimír Bartl

Brno, fall 2014

STATEMENT OF AUTHORSHIP

Hereby I declare that this thesis is my original work, which I accomplished independently. All sources, references and literature I use or refer to are accurately cited and stated in the thesis.

Signature:..

Acknowledgement

I wish to express my gratitude to RNDr. Marian Novotny, Ph.D. for governance over this thesis. Furthermore, I wish to thank my family and my girlfriend for their support throughout the process of writing.

Tutor: RNDr. Marian Novotny, Ph.D.

Abstract

This thesis discusses a topic of malicious software giving emphasis on client side threats and vulnerable users. It gives an insight into concept of client honeypots and compares several implementations of this approach. A configuration of one selected tool is proposed and tested in the experiment.

Keywords

client honeypot, low-interaction, high-interaction, malware, attacker, exploit, vulnerability, Cuckoo Sandbox

TABLE OF CONTENTS

1 INTRODUCTION

The trends of the modern age of humanity are making a tremendous use of computer technology as well as of the Internet. The escalation of everyday use is exponential. A different kind of computer can be seen almost in every type of an electronic device, but not limited only to these devices. Computers are being operated by a vast amount of people with greatly diverse levels of knowledge and operation skills, beginning with children and unskilled daily users through educated staff, up to highly skilled experts. To prevent misuse, leak of information, personal damage, or even financial harm, we need to ensure respective security of these systems. The security of systems highly depends on the skills and awareness of the user and the proper utilization of a computer. The goal is to achieve equal level of security for every possible user interacting with a computer. At this point, a complex problem begins.

There needs to be an appropriate promotion and education of computer abilities and habits amongst users because the sufficient level of confidential security is not possibly achievable just by securing the systems, although it should be the first step to begin with. Security teams of skilled engineers are constantly trying to improve systems and provide desired security. The spectrum of techniques and tools available is extensive. One of these techniques is a *Honeypot* on the servers' side of the network or a *Client Honeypot* on the clients' side, which is the main topic of this thesis.

The work is divided into six chapters. The chapter following the introduction is about giving the basic explanation of honeypots and categorizing them into several groups. Afterwards, essential information about attackers is presented as well as details on malware (malicious software) regarding types, threats posed and tools to protect from it or analyze it. The third part of the thesis is about a more detailed description of selected available client honeypots referring to architecture signatures and operation principles. At the end of the section there is a comparison of attributes of these selected honeypots. The fourth chapter is dedicated to an exhaustive explanation of the framework that was selected to host the experiment. It contains a thorough description of the project, deployment and configuration means. The fifth chapter defines the experiment to prove the configuration of honeypot environment, shows the process of experiment and presents the evaluation of the data collected during the experiment. The final chapter summarizes the whole thesis.

1.1 Motivation

The motivation for this work is simple enough. It is the spreading of cyber-crime targeting vulnerable users who often do not even suspect something unwanted or even harmful is happening to them. The purpose of the thesis is to explore options on how to contribute to detection and prevention of various threats waiting for end-users on the Internet. The gained knowledge can be further processed to create and apply necessary measures to improve users' security in the wild of the Internet.

In the time frame of the past 10 years the number of Internet users has grown massively – from 14.1% of the population in 2004 amounting to a bit less than 1 billion users to more than 38% in 2014 which stands for 2.75 billion users [13]. Nowadays, theoretically every third person living on the Earth is connected to the Web either by using a desktop computer, a laptop, a tablet or a smartphone. Due to this fact, the attack surface[1] is growing bigger and bigger and it is easier for criminal minds to exploit and take advantage of assaulted users. Hand in hand with users, the number of webpages is growing as well as the number of opportunities to attack users directly or by using different, third party, webpages they visit. In regards to worldwidewebsize.com the estimated size of the indexed web is at least 2.06 billion webpages [31]. Although not all of these pages are active all the time, there are plenty of opportunities for users with hostile intentions.

Even though the focus of attackers might have expanded to mobile platforms as they became more popular in the last 2-3 years, the number of computer threats has not lowered significantly. "As of January 2013 the National Vulnerability Database [2] (NVD) listed 53 489 vulnerabilities affecting 20 821 software products from 12 062 different software vendors. [27]" In the past 10 years on average 4 660 security vulnerabilities were

[1] attack surface refers to sum of such points in OS or application software that could be used by an unauthorized user to penetrate into the system

[2] http://nvd.nist.gov

disclosed to general public audience per year. Figure 1.1 depicts the tendency in vulnerabilities disclosure, number of products and vendors showing a decreasing trend after the peak year of 2006. Nevertheless, year 2012 shows

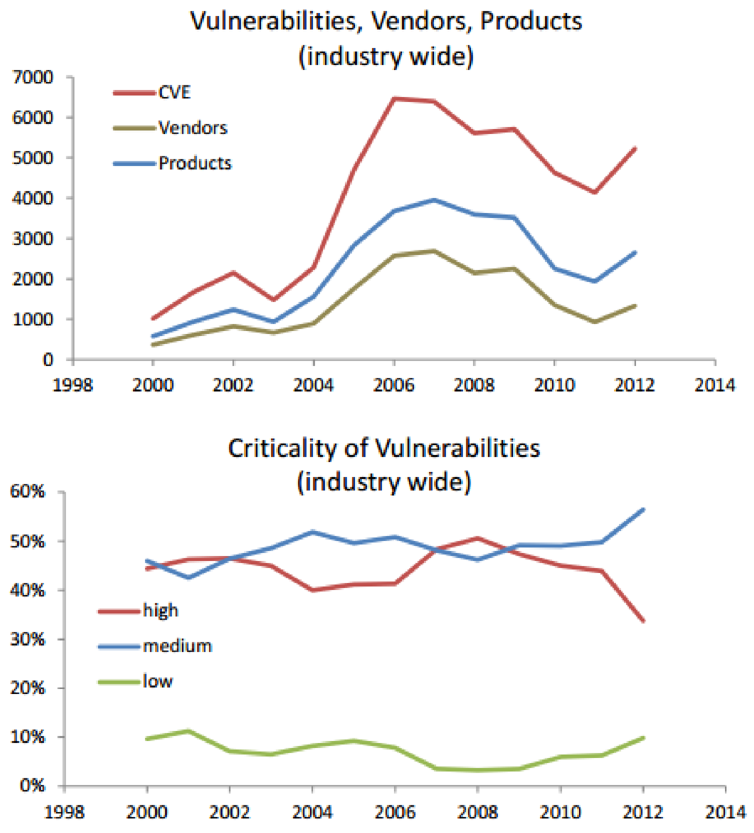


Figure 1.1 Distribution of Vulnerabilities and Criticality [27]

a turnaround back to an increasing count of vulnerabilities. Also, an important fact to mention is that not all of the vulnerabilities pose the same danger. The second chart in the figure 1.1 shows rates of the criticality of vulnerabilities – the lower is the better.

The referred report also introduces the distribution of vulnerabilities amongst software vendors where a small number of vendors are in possession of the majority of disclosures. Mainly vendors of popular, everyday use, software are on the top spots of this chart, e.g. Adobe (Flash, Reader), Mozilla (Firefox, Thunderbird), Oracle (Java), Microsoft (Word, Excel) and others. [27] Based on these facts, there is a strong need to search for and fight against all of these threats to make the Internet a safer place mainly for unskilled users.

1.2 Contribution

The contribution of the thesis is to clearly lay out and summarize the theory that stands behind the entire field of honeypotting – how it originated and what the trends and developments are. The thesis will properly explain the definition of a *honeypot*, but more importantly the definition of a *client honeypot*, in hand with the principal division of honeypots based on the interaction level. Explanations of matters that are closely associated with the area of cyber-attacks are given. The thesis describes cyber-criminals, malware and threats that are posed to vulnerable users, but also tools that can be used to contribute to user's protection against these threats and tools that can check web resources in a sense of maliciousness.

Following the essential-theory part, the chapter describing several selected honeypots gives a closer look to the actual implementation of the honeypot hypothesis as it is seen by various security experts and enthusiasts. This part covers low-interaction as well as high-interaction solutions. The chapter is concluded with a comparison of reviewed honeyclients.

Throughout the Internet there is a quantity of information available about all sorts of various aspects connected to the problem of cyber-security, yet none of them gives an easily-understandable summary of essential terms and concepts in a complementary security level like client honeypot is. The contribution of this thesis is to create a text that is easy to understand and is sufficiently fetching to motivate a common user to join the efforts in malware research using honeypots. In the thesis we will show that even a user without expert-like skills is able to set-up his own test environment at home. The thesis also shows how a test environment using a honeyclient can be configured. The setup is subsequently used to perform an experiment. There is a chapter devoted to describe the experiment and the results extracted from it.

2 BACKGROUND THEORY

The chapter is dedicated to the necessary theoretical foundation behind honeypots. At first, a proper definition of the term is presented in hand with an explanation what a honeypot is, because the idea of a honeypot can be interpreted in several manners depending on the reader's point of view. Then a division of honeypots based on the logic of operation is shown. Namely we discuss an older server-side approach as opposed to a newer client-side, followed by a division based on the interaction level of honeypots. Later this chapter covers basic information about a, examining their motives and targets. The last part of the second chapter is addressed to the information about malware, specifically types, threats and tools to mitigate risks or analyze it.

At the beginning of the Internet era, websites consisted only of static content so the visitor could only view what was embedded in websites hosted on servers. Thanks to that reason attackers aimed at assaulting these servers, e.g. tried to tamper the information that was displayed or steal non-public content. Around year 2004 a term Web 2.0 was established [42]. It does not refer to a particular software or technology rather to a set of applications and mainly a point of view, in general, how the Internet is seen. It means that the Internet became dynamic, more interactive and the user is offered an opportunity to create a certain part of the content on his own, e.g. interactive responses to web application queries, blogs, forums, Wikipedia. In other words, a greater collaboration between users themselves became possible. For that purpose technology on the client's side such as Ajax, Flash, JavaScript and others was introduced. Such an approach established a wide space for new attack types as well as the target of attacks started to switch from servers to users. An attacker can decoy a code into a webpage or a web application that will execute itself after a user navigates to that web resource or performs the attacker's pre-defined action while browsing. The trick is that the execution is hidden and happens in the background without the user's consent. In this way, a successful attack is capable of making a user to perform disadvantageous actions or recover diverse intelligence.

2.1 What is Honeypot

First of all, the main idea of a honeypot is a computer security tool, which is flexible and widely adjustable to many different scenarios. The simplest solution, in the form of a low-interaction server-side honeypot, starts only with the emulation of chosen services, e.g. HTTP, Telnet, FTP, to give an attacker essential responsiveness and basic interactivity. The goal of the set-up is to search for well-known attacks and exploits. This approach merely collects data of a specific narrow profile, strictly bound to the type of service that is being emulated. On the other side stands a high-interaction honeypot which exposes a complete, properly functional operating system to an attacker. Thus, the attacker is allowed to cause significant harm to the exposed system if the system is not kept under close surveillance. This might even involve a sequence of events leading to a connection of the attacked system into the attacker's botnet[3] in order to spread malware, send spam messages or abuse other systems in a similar manner. The advantage of this strategy is the scope of collected data, which is rather wide and gives security personnel a better insight into the attacker's ambitions. However, analysis of such data is more of a nuisance and requires a greater amount of time and experience. This kind of honeypot set-up has a possibility to catch a zero-day attack[4]. Server-side honeypot is a dedicated system deployed on the network; it has no production value which is the main difference in comparison to Intrusion Detection System (IDS)[5]. [28] No production value means that no traffic should reach the honeypot for the purpose of ordinary communication. For that reason, all the traffic spotted on a honeypot is, with high probability, an attempted attack, simply the actions of attacker. This is an advantage, because the amount of captured data is considerably lower, hence the need to filter a vast amount

[3] botnet is a network of compromised computers that is remotely controlled by the attacker

[4] zero-day attack stands for using a new system or application vulnerability to breach into the system, which was previously not known by security community

[5] IDS is a security software, which automates the intrusion detection process of monitoring the events happening on network that may be violating predefined security policies or well-known best practices [30]; usually they are deployed on the systems with production value, so the amount of data to control is extensive and high number of false detects appears

of data to find symptoms of intrusion or attack diminishes. Still, mistakes are possible and occasionally a false positive alert may be raised. False positive refers to mislabeling the traffic as an attack although it is benign communication traffic. The rate of triggering false positives principally depends on the configuration of the honeypot.

Important fact to know is that honeypots are definitely not there to replace any other known security measure. They bring the chain of security measures to a higher level. Honeypots help to study the signs and behaviors of malware and thus contribute to the creation of new definitions for security software like antivirus engines or to creation of tools to prevent from, or alternatively remove the malware infection.

2.1.1 Definition of a Honeypot

Honeypots have a wide range of applications which depend on the type of data about attacks we demand to collect. For that reason the meaning of the term *honeypot* may not be uniform. One of the first publications on the topic of honeypots, written by Lance Spitzner, has defined a honeypot as follows:

"A honeypot is security resource whose value lies in being probed, attacked, or compromised. [33]"

The definition says that a honeypot is a tool to lure assailants so they believe they are interacting with a real system or a real user that they can compromise them and gain some kind of benefit. Yet, most often, it is a dedicated system with the purpose to mislead the attacker and uncover the actions he would normally perform to invade a system. An analysis of collected tracks helps to understand such behavior and design appropriate countermeasures to better defend against threats and attackers, not only for researchers, but mainly for ordinary users.

2.1.2 Server vs. Client honeypot

The initial viewpoint of honeypots was the use on the server's side. It means that an attacker was interested in exploiting a server which might be utilized to store sensitive private data or even confidential corporate intelligence. Thus, server-side honeypot served as a quiet decoy on the network waiting for an adversary to connect and bring the criminal thoughts into action. Upon this connection the honeypot was observing actions performed

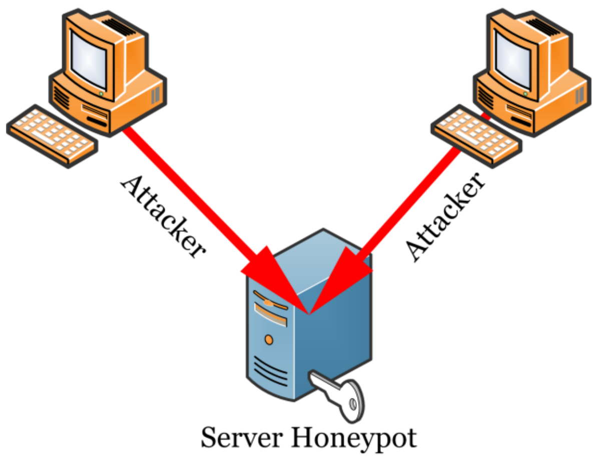


Figure 2.1 Server honeypot scheme

and recording all relevant details. Figure 2.1 is a schematic illustration of this perspective. It is useful, for example, in catching computer malware known as worm[6] because a honeypot is passively waiting for an adversary (malware) to reach it. Such was the initial motivation for honeypots forming around year 2000 when most famous worms were caught in the wild, e.g. Melissa (1999), ILOVEYOU (2000), Nimda (2001) or CodeRed (2001) [35].

On the other side stands a client honeypot, which is often referred to as *honeyclient*. The principal idea of honeypot's definition remains the same for client honeypot as well. As a consequence of following security research and vendors evolving their software, servers became more secure and therefore harder to exploit as well as the change of attitude towards the Web with introduction of Web 2.0, as mentioned before. Naturally, attackers tried to find easier targets and their aim extended to ordinary unskilled end-users of the

[6] worm is a class of malicious code which is capable of self-replicating without any user interaction; the main areas of assault are system memory and hard-disk drive where it replicates endlessly causing the sluggish responsiveness of the computer by draining its resources and occasionally denial of service (DoS); after infecting a computer, the worm takes over the control of information-transport features and propagates to other computers on the network [5]

Internet. The number of possible victims in a form of users is much greater than the number of servers on the network, it is practically immeasurable. Thus, the attack surface is outspread because of diverse environments existing on the network and multiple attack vectors[7] discovered which method to use to breach into a system.

On the contrary to passive server-side honeypot, a honeyclient is an active security resource which crawls the Web. The ambition is to find malware waiting on the websites for users' interaction as shown in figure 2.2. This kind of malware may be embedded into a webpage on purpose, with clear malicious intention, or it may be injected by attacker into, otherwise benign, website without any knowledge of webpage's administrator. Honeyclients emulate certain behavioral patterns of users in order to trigger malware execution since assaulters try to protect themselves from being detected by using various evasion techniques.

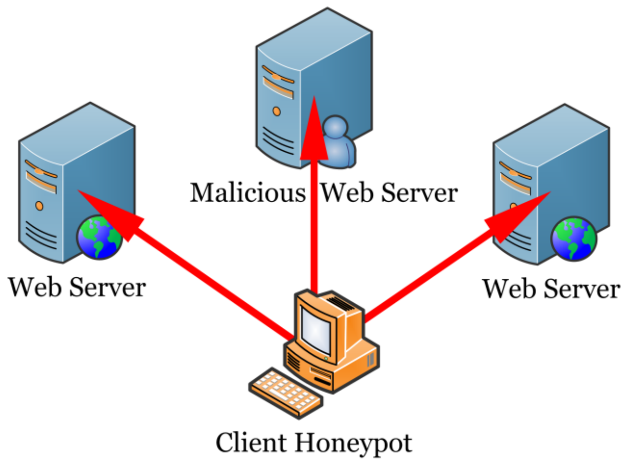


Figure 2.2 Client honeypot scheme

2.1.3 Honeypot by interaction level

Honeypots are divided into two main groups by level of interaction they provide to the perpetrator. The usage of appropriate level strongly impacts the amount and scope of data collected about probes. The basic approach

[7] attack vector is a mechanism used by attacker to gain access to a computer

is to provide only low level of possible interaction, giving an attacker only suitable responses. That is achieved by emulating system services or chosen vulnerabilities in order to catch desired malware. The evaluation of threats happens by matching suspicious behavior with pre-defined signatures. In case actions match precisely the threat is detected otherwise it cannot be classified with confidence. Due to the necessity of pre-defined signatures, this type of honeypot cannot be used to reveal zero-day attacks, i.e. it can only detect known problems. Such attitude is beneficial in a sense of speed because honeypot is capable of evaluating many more webpages than high-interaction honeypot within the same time frame. The data collected is of specific profile and is suitable for attack detection rather than examination of attacker habits and intentions. Furthermore, an experienced attacker is able to detect such set-up and may change his behavior or most likely leave without revealing anything about him. Despite that, low-interaction honeypots are easier to deploy as there is no need to install and configure additional services. Administration and maintenance is also easier because the host system cannot get infected as it is emulating the services and not directly exposing them. Thus, after detecting a threat the honeypot does not need to be cleaned and reverted to healthy state which is usually a time consuming procedure. The use of low-interaction approach can also be called *static analysis*. That is due to the fact that malware is only being evaluated based on the pre-defined signatures, which are stationary and have precisely defined boundaries of what can be detected.

High-interaction honeypot utilizes entire operating system's functionality including its services, applications, other components and exposes them to the attacker which gives him more freedom in choices where and how to strike. The system is being closely monitored, e.g. system registry entries, process creation/termination, file system, network activity, etc. in order to spot differences resulting from attacker - honeypot interaction. That is why the scope of collected data is broad and the amount is voluminous, hence the analysis is more challenging. It requires time and experience, but the outcome is the ability to capture zero-day attacks and holding such knowledge is required in a process of signatures creation that are used in low-interaction and intrusion prevention paradigms. This kind of activities can be labeled as *dynamic analysis*. Thanks to the complete operating system behind the setup where a honeypot is arranged, there are practically no limits for the possibilities of detection. The researcher can dynamically investigate the changes made to the system while the malware is being executed.

One of the drawbacks is the effort needed to set up and run high-interaction honeypot. The work of administrator is more demanding due to high scalability of environment where we can install additional vulnerable applications which may come in different versions. Additionally, maintenance is more complex as well because the need to reset a healthy state after examining malicious activity emerges. Table 2.1 summarizes the outline differences between high and low interaction approach. The setup of a combination of both interaction approaches is also possible in order to create a versatile tool for analysis.

	low-interaction	high-interaction
services	emulated	real
set up	easy	complex
maintenance	easy	demanding
attacks	well-known	zero-day
velocity	speedy	slow

Table 2.1 Summary of main differences by interaction level

2.1.4 Honeypot by deployment environment

A honeypot can be deployed into different environment types. That can be either physical setting or using virtual software emulation of hardware.

One outlook is to set up a honeypot directly into an operating system of a physical computer yet this attitude has a big disadvantage. That drawback shows up as soon as the computer gets infected by any malware. Because there is no fast and easy-to-use revert option to previous healthy state available, the re-installation of the whole operating system and all the services is a necessity. The alternative to omit reverting to clean state before navigating to any other website is out of the question for the reason that the ability to impartially determine the harmfulness of the next website has changed, thence the threat may be overlooked. This line of action is truly time and energy consuming. Although it is not entirely pointless because virtual environment can be detected (e.g. checking system registry keys, running system processes) causing the attack not to trigger. This is one of methods how attackers try to avoid unmasking as it is unusual for a normal user to use virtual emulation of an operating system while browsing the Internet.

A virtually emulated environment appears to be more useful for the sake of honeypots. It has some advantages to make the honeypot deployment and usage more effective. The most notable is the ease of machine's state changing. Furthermore, the number of realizable set-ups in identical or different configurations we are able to deploy and thus save time, labor and resources is an important advantage as well. From just single set-up up to distributed networks of honeypots identified as *honeynet*. Because the intention of honeypot is to get infected in order to gather information about malware, good computer systems' administration is essential. Since malware is capable of leaving unnoticeable tracks, the need to revert a healthy state is obligatory otherwise abilities of honeypot may be corrupted. Virtual environment tools change state by using saved snapshots of the system which is an automated process and is significantly faster than manual revert. Nowadays, a few tools for virtual emulation are available, e.g. VMware, VirtualBox, User-Mode Linux. The decision depends on the honeypot administrator or honeypot developer if any virtualization tool is implemented directly into the honeypot solution.

2.2 Attackers

In order to effectively prevent and defend from danger on the Internet it is useful to know who may be the source of malign activities. In the context of computer security, users with malicious intentions are called *hackers*[8]. However, these users can be further classified as:

White hats. Security experts who break a system's security for the purpose of penetration testing and revealing vulnerabilities, often labelled as ethical hackers. White hats have permissions to access computer system.

Black hats. These are users who have malicious plans for the sake of gaining profit from the assaulted system or damaging the system's functionality, i.e. cyber-criminals. Black hats have no permissions to access computer system.

...and divided into groups by the level of their knowledge:

Script kiddies. These attackers (mostly youths) lack extensive knowledge about computing and use mainly tools or scripts that were made by skilled hackers, i.e. these attackers do not understand adequately what is actually happening in the background. Thanks to automation of the hacking process the quantity overcomes the quality of assaults.

Advanced black hats. Skilled attackers with extensive knowledge about systems and computing who are capable of finding new threats in applications, creating scripts and exploits used by script kiddies, or can aim to penetrate a desired system of higher value (probably better secured). Thanks to the experience these black hats are often excellent at covering tracks of their attacks. Moreover, black hats are not used to reveal their achievements or share tools and techniques. Thus, if such expertise is disclosed it may not be applicable to another skilled hacker. [33]

Naturally, there is also a layer between these two groups where hackers with moderate skills and knowledge are, who can roughly understand what scripts do and how programs work, but are not able to code them on their own. Due to the fact that pre-made programs are easy to use, the number of script kiddies is significantly higher than the number of advanced hackers. However, due to the level of severity, attacks by advanced hackers are considered to be far more dangerous for the reason that such attacks do not necessarily follow well-

[8] the original meaning of the term says that a *hacker* is a user with advanced skills in computing/programming, not only the user with malicious intentions

known, easier-to-detect patterns and are easily missed by security systems. Due to the overwhelming growth of the Internet users, the quantity of attacks by script kiddies on weakly secured systems should not be underestimated.

2.2.1 Targets of attacks

It is important to realize that everyone connected to the network can be a target for any kind of attack. Assuming that one's computer is not valuable enough to become a target is wrong. Even if a computer does not possess valuable information it can be utilized by an attacker, for example to execute sub-sequential attacks or to store some previously stolen information. By studying victims a lot of useful information can be learned. Mainly motives and tactics can be discovered which may be helpful to predict future occurrences or future targets of the attacks. Targets are divided into two main groups that are thoughtfully linked to the kind of attackers abusing them:

Targets of Opportunity. The goal is to hack as many systems as possible. Generally, these are the targets for less sophisticated assailants who tend to use automated tools to scan large networks in pursuance of a single (or a small set) vulnerability to exploit it.

Targets of Choice. The goal is to penetrate a desired system while chasing certain higher value information, e.g. credit cards, corporate confidentialities, government espionage. Such targets are mostly followed by skilled hackers. Considering that black hats are distinguished in covering their tracks, they can reside in a system for a longer period of time without being discovered. Even after finding out about the residence, it is no trivial process to track the attacks down to the actual source.

2.2.2 Possible motives

The spectrum of different attackers' motives is wide and is truly known only to the attacker himself. Despite that, it is meaningful to make an effort and investigate the attacker's motivation. Such investigation may be profitable in the sense that if the victim is solely a target of random (impersonal) attack than the incident may be considered the last as well. On the other hand, if the reasons for the attack were more personal a victim might need to await reoccurrence of the attack in the future. Therefore, the victim may carry out all means necessary to better protect him. Another notable fact is that attackers often invade computers with the motivation to gain accessibility for later visits.

An acronym MECEES, established by Dr. Max Kilger from Honeynet Project can be applied to divide motivation as follows: [15]

Money	probably the most driving motivation for black hats
Ego	satisfaction of overcoming the technical (or security) barriers, powerful code creation, innovations
Entertainment	personal amusement which is feasible after exploitation
Cause	in other words hacktivism, i.e. pursue for promotion of certain political view, ideology, or information ethics (form of a protest)
Entry to social group	showing off a level of expertise to gain access into a group
Status	acknowledgement amongst the hacking community, besides money the strongest motivator

2.2.3 Structure of an attack

The composition of an attack can be split into 4 steps that occur during the attack.

1. <u>Investigation and enumeration</u>

   The point of investigation is to recognize possible security flaws in the focused system. There are several techniques available, such as social engineering, (spear) phishing, pharming. Skilled attacker can even obtain such security-threatening information just by reading user's (corporation's) website. Enumeration process is sorting out the useful part of investigated information.

2. Intrusion

   This is the phase of penetration into the system or network when an intruder gains control over the assaulted system or network.

3. Malware injection

   After a successful penetration, an attacker can inject malicious code into the system in order to ensure continuous control over the system, or alternatively to achieve the goal why he attacked the system.

4. Clean up

   When the goal is accomplished or the system is set up for further visits, attacker works on deleting the evidence of his visit. This can be done by, for example, deleting event logs, upgrading of outdated software, or similar.

2.3 Malware

The word malware is an abbreviation of expression *malicious software*, sometimes a word *badware* is used as well. This term gives reference to a wide range of different software, most likely of poisonous nature. The purpose of such programs is to cause harm to an unprotected (vulnerable) computer system and gain some sort of leverage. It may be served in varying forms, for instance virus, Trojan, spyware, adware, to follow certain scenarios or pursue specific kind of information on the assaulted user's computer. Malware is strongly connected with terms *vulnerability* as well as *exploit*. In this place the definitions of these terms are presented as Microsoft interprets them:

"Vulnerabilities are weaknesses in software that enable an attacker to compromise the integrity, availability, or confidentiality of the software or the data that it processes. Some of the worst vulnerabilities allow attackers to exploit the compromised system by causing it to run malicious code without the user's knowledge. [22]"

Simply said, vulnerability is a weak point which is being focused on by assailants in order to deliver exploit and execute it. Every known vulnerability has a standardized number, for easier referral and data exchange amongst software, and is listed in *Common Vulnerabilities and Exposures* (CVE) list maintained by MITRE Organization[9].

"An exploit is malicious code that takes advantage of software vulnerabilities to infect, disrupt, or take control of a computer without the user's consent and typically without their knowledge. Exploits target vulnerabilities on operating systems, web browsers, applications, or software components that are installed on the computer. [22]"

In this subchapter we introduce relevant types of such software and attack vectors used to deploy.

2.3.1 Types of malware

Malicious software can be partitioned into three different classes by the maliciousness of their nature against users. [37]

1. Nuisance malware – Spyware, Adware

   Spyware. It is a kind of badware which is not necessarily malicious in a manner of posing some direct threat to users. By design, it is software which monitors user's Internet behavior and browsing habits and sends gathered information back to assaulter who can misuse it on his own or sell it to third party. The main focus of spyware is to collect confidential data such as usernames, passwords, or identity details. Leakage of such information is dangerous and misuse can be menacing.

   Adware. That is a subcategory of spyware family. It collects browsing habits and adjusts advertisements accordingly showing various pop-up windows. It is often able to redirect user to different webpages, or it can take over browser's home page attribute and change it which makes a user navigate to unwished-for websites.

   Presence of such software in user's computer is vexatious and may have a significant impact on the computer's performance and stability.

[9] http://cve.mitre.org

2. Controlling malware – Trojan, Rootkit, Ransomware

   Trojan horse. A Trojan is a harmful application designed to trick a user to believe that a file is of desirable or beneficial content. It is designed to provide attacker with remote access, or even create a backdoor [10] on the assaulted system. Unlike worm, Trojan does not reproduce nor does it self-replicate. Thus, it must be carried as a part of another program, unwanted download as a result of visiting a compromised webpage, or received via messaging and subsequently activated, i.e. by opening message attachment. After successful infection, the attacker may alter or steal data, for example credit card details, personal identity information, or install and launch other undesired software such as input keylogger in order to track user's input like login credentials. Trojans are versatile in the sense that they provide multiple types of infection after executing and create a lot of options for perpetrator to command and abuse the system. For that reason Trojan-like malware is currently a leading threat on the Internet.

   Rootkit. More advanced than Trojan. It is a piece of malware that benefits from taking over a user account within the system that has administrator rights. This way, the attacker has full permissions to execute any desired operation. The advantage against Trojans is that rootkits are crafted in accordance with the ability to hide themselves inside of the system on the sub-OS level. Rootkit possesses functionality to avoid detection by conventional means (e.g. hiding running process from the system's processes list) and thus retaining attacker's option to abuse a system repeatedly. Certain variations are possible to defend against removal by re-starting killed processes or re-creating deleted parts of the package. It is no simple routine to remove a rootkit from a system; occasionally a full reinstallation may be needed.

   Ransomware. It is a malware that uses Trojan methodology to infiltrate a user's system. After infecting, malware either encrypts selected users' files, or completely restricts the admission to the system. By such means abusers try to extort ransom fee from the assaulted person who wants to regain

[10] backdoor is a method to bypass usual access control procedure

access to his files or system. That is a powerful variety of malware due to the fact that encryption is based on asymmetrical formula; therefore the decryption key is known only to the abuser. This means that by refusing to pay the blackmail fee a user will surrender all affected files due to inability to decrypt them without the decryption key owned by attacker.

3. Destructive malware

   This category covers malware which is distributed with intention to disrupt systems' operation, for example by erasing data stored on the computer's hard-drive, or making hardware inoperable (wiping BIOS flash memory). Destructive malware is a label indicating programs with such objectives. History showed us that a computer worm (as described in section 2.1.2) has frequently had a destructive nature. Besides, ransomware or a Trojan carrying a deadly payload can be classified under this category.

2.3.2 Attack vectors

The majority of malware attacks rely on the so-called click fraud scenario. A cyber-criminal crafts some content which looks to be of certain value, however on the backend a deceptive action takes place. It is a common way to obfuscate malicious plans and seduce an unexperienced user to trigger vicious actions. *Attack vector* denotes a mechanism, vulnerability that is abused by attacker to exploit the system or network and gain access to resources needed to accomplish desired actions. This section presents some of the well-known attack practices used in the wild.

Phishing. It is a fraudulent technique that makes use of messaging environment like email or instant messaging. The hacker sends out thousands of messages (emails) to a number of recipients as big as possible. Messages are likely to include links which will navigate users to a disguised website that was pre-made to look just like the genuine website, however maintained by the attacker who is able to collect trusting users' information.

Spear phishing. It is a phishing practice that focuses on selected persons only. The messages are crafted directly for particular recipients and are sent solely to those people. The attacker often looks for publicly accessible information about the victims he wants to address. This method is popular to deliver payload when targeting a specific organization as one of incautious employees is likely to execute the malicious content, attachment of the email and thus invite malware into the organization's systems.

Email. Emailing is a popular channel for malware distribution. Practically, anybody can send an email message to anybody else. There are no restrictions by default, taking into account the fact that email can be considered publicly available information, thus attacker can contact his victim directly. On the other hand, there is no genuine sender identification who thence can pretend to be whoever. Emails can transmit attachments of various types and make it easier to distribute malicious files. Email scams bet on users to fail in reviewing the authenticity of the content and mainly of the sender and to click on embedded redirection link or attachment beforehand. One advantage of email channel is that malware cannot activate itself; it is dependent on user's interaction.

Weak Authentication. This is a common problem of the secured areas all over the Internet. Users tend to protect their accounts using passwords that are easy-to-remember, e.g. birthdays, pet names, common phrases. These passwords are easily guessable in several minutes by brute-force using dictionary attack [11] . The other deficiency is that systems', websites' administrators do not demand the use of strong passwords. Even worse is the situation when the security mechanism is implemented incorrectly and has various security flaws to abuse.

Internet Social Engineering. A term used to describe various fraudulent techniques used by attackers who try to trick users into revealing the personal information about them that can help attackers in conducting a successful assault and achieve the desired goal. For example, above-mentioned phishing is one of the Internet's social engineering techniques.

[11] Dictionary attack is an assault using pre-built list of known passwords and common words, phrases for a particular language; it is a trial-and-error approach

2.3.2.1 HTML

HyperText Markup Language is a defined standard for webpages creation. Rather than programming language, HTML is a markup language defining the logical structure of websites. Thus, it does not have enough potential to exploit vulnerabilities on its own; despite the fact that there were several issues known in the past. [21] The language is used to create structure for website's content and to embed various files into website, which is later displayed to a visitor. For such purpose HTML tags are used that denote elements within the page. Furthermore, scripts can also be embedded into pages in order to provide some sort of reactive behavior to interact with user's actions. [45] A script does not need to be included directly into a page, a URL pointing to the actual script can be provided which is then accessed remotely. In addition to objects embedding, there is a possibility to insert another webpage into the original page. Due to the capability of setting the display attributes of such inserted webpage, it can be completely hidden from a vulnerable visitor. That is often used when the inserted page is of malicious nature.

Web browsers are user applications that can read and interpret the content of HTML page in order to display it in human-readable form. Based on the fact that HTML is not a programming language, HTML does not provide enough possibilities for attackers to exploit vulnerabilities. Rather it gives attackers possibilities to decoy malicious resources into webpages that have the potential to exploit users' systems.

2.3.2.2 JavaScript

JavaScript is an object oriented interpreted scripting language that was developed to provide responsive content to users' actions within webpages and is executed solely within a web browser. [43] JavaScript is written in text-form and needs to be embedded directly into HTML body and therefore gives a user extensive set of possible actions. Due to such large scale of actions, JavaScript gives an attacker a lot of scenarios how to gain advantage over assaulted user. For instance, by using JavaScript it is possible to steal session cookie and impersonate an authenticated user, it is possible to invoke a URL redirection to an arbitrary destination address, or manipulate the properties of objects in the Document Object Model (DOM)[12] tree. [9] Because DOM standardization defines many different event handlers, an attacker can use these as a trigger to kick off the malicious action, e.g. start the redirection on *mouseover* event over some picture on the webpage, supply the drive-by download on a *keypress* on the keyboard, or steal the session cookie when a web form is submitted. Even an existing object can be manipulated and misused to download malicious content from a remote site; such can be achieved by adjusting the property of an object pointing to a malicious URL. JavaScript is also able to work with objects, i.e. files of various types that can be embedded into webpages. Due to the variety of file types, the supplementary browser plug-ins exist that help a browser to properly display the content to a visitor. Every available plug-in has a unique identification number, which makes it easier for an attacker to provide malicious content to a browser and summon a desired plug-in which has a known vulnerability to exploit. [40]

This fact makes JavaScript versatile and complex enough to attack not only web browser engines, but also add-ons thus broadening the possible attack surface. Thanks to all the possibilities, JavaScript is a favorite attack vector for client-side attacks.

2.3.2.3 SQL Injection

SQL Injection (SQLI) attack is a wide-ranging issue in modern era of dynamic web content. On the backend of every meaningful web application, there is a database which stores relevant data needed for proper functionality. The data is often created by users who use various input fields throughout the webpage in order to forward information to the table. SQLI is a kind of attack that targets these databases in order to steal or manipulate stored intelligence. What is more, in certain cases attacker is able to erase contents

[12] DOM is an API for HTML and XML documents; DOM defines the logical structure of documents and is in a form of a tree or forrest (connected trees); it also defines the way a document is manipulated, i.e. accessed, changed, deleted, or added; throughout the development lifecycle a lot of functionality was added, today's Level 3 DOM specification supports various event handlers, such as click, mouseover, drag, keyboardpress, resize, change, submit and similar [46]

of the table. In simple words, the problem dwells in the web application that is programmed insecurely and does not validate user's input sufficiently.

Web application is a mediator between a user and a backend database. Authenticated users are allowed to view or alter records of the database. On the other hand, a hacker has no such authenticity, but still wants to interact with the database and its records. There are multiple ways how to determine if the particular database is vulnerable for some type of SQL injection. Afterwards, the attacker delivers a masterminded SQL command using one of available input areas. Due to the reason that the inputted SQL command is not properly reviewed and sanitized, the SQL query containing user input is passed directly to the backend database where it is executed. This way a hacker fools SQL interpreter to execute unplanned commands.

This attack can be prevented only by proper input validation, encoding and use of up-to-date software where known security flaws are fixed. The following Cross-site scripting attack can be partially seen as a variation of SQL injection.

2.3.2.4 Cross-site scripting

In the recent years Cross-site scripting (XSS) became one of the most used client-side attacks. Several sources report that nearly half of the overall amount of attacks was based on XSS. At first, an attacker needs to abuse a legitimate webpage and entangle a malicious script inside the page. The attack itself takes place when a user visits a webpage and a dynamically generated response is sent to the user's browser where it is interpreted and executed. The malicious trap is either reflective or persistent. [3]

Persistent cross-site scripting attack is illustrated in the figure 2.3. The attacker abuses a web server in order to store a malicious script. The script is not harmful for the server, yet it is dangerous for users visiting the server. This security flaw occurs mainly when the web application does not properly inspect and sanitize user input. As a common example an arbitrary message board can be considered, how a Facebook's user profile wall can be seen as well. The attacker posts a message and appends a script code to the end of his

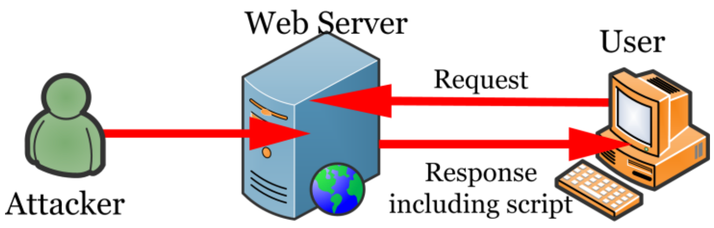


Figure 2.3 Persistent XSS

message. The script code is hidden due to the reason that it is embedded in HTML tags that are by default not displayed to visitors. However, when a visitor requests to view the webpage a response containing all HTML code is sent to his browser. Upon receiving, the browser interprets all content received and executes the script since it came from a trusted source. The script redirects a user somewhere else, or steals his session cookie[13] for example. This way various kinds of unwanted behavior can be achieved in order to assault unaware users.

Reflective cross-site scripting attack can begin when a user clicks on a hyperlink received via email, for instance. The hyperlink may look trustworthy, yet it encapsulates the malicious script. Nowadays, services to shorten web URLs are broadly used, thus it is even easier for attacker to prepare a malicious hyperlink and broadcast it to users. It is enough to introduce the link with some video in order to invoke user's curiosity who will

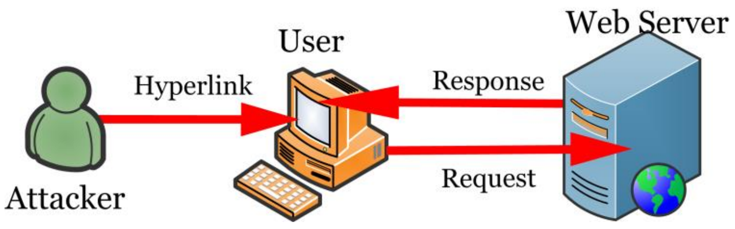


Figure 2.4 Reflective XSS

eventually click on the link. After the web resource, to which the link points, is opened, user's web browser interprets and executes the script. This attack is called reflective due to the fact that the web application only reflects the malicious script back to the user's browser where the action is executed as shown in the figure 2.4.

[13] Session cookies are often being stolen by attackers so they can impersonate the victim and pretend to be a legitimate user

2.3.2.5 Buffer Overflow

Buffer overflow defect is caused by imprecise application coding in hand with deficient input validation. It means that the input of larger size is passed to the buffer than is the allocated size of the buffer. As a consequence, data stored in memory addresses behind the buffer boundaries are overwritten. This way attacker's arbitrary injected code can be granted the same system privileges as the application whose data were attacked. The *C, C++* programming languages are often targeted by buffer overflow attacks due to the fact that languages do not have a by-default checking mechanism. [17] Merely, the first to exploit buffer overflow vulnerability was the well-known *Morris worm* in 1988, which exploited UNIX finger service. [32]

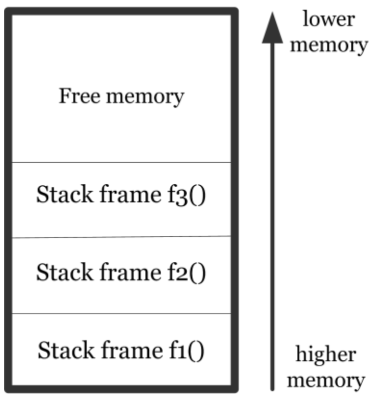


Figure 2.5 A stack

For the purpose of explanation, we will use the stack-based buffer overflow due to the reason that the vulnerability's principle is, in fact, very similar in other buffers as well. A buffer is a continuous block of memory that holds a multiple instances of the same data type. As shown in the figure 2.5, a stack is a *Last-In-First-Out* (LIFO) logical structure for the buffer - the last appended record to the top of the stack will be the first record to be taken for processing in following appropriate step. The stack can either grow down (from higher memory addresses towards lower), or up depending on the implementation based on the CPU manufacturer. The functionality of a stack is to allow recursive function calls, i.e. the stack stores return address, function arguments and local variables. The set of data belonging to the same function is called *stack frame*. When a function is called by a program, a new stack frame is created and pushed to the top of the stack. A register called *stack pointer* holds the current address of the top of the stack. Moreover, there is an auxiliary *frame pointer* register that holds a fixed position within the stack, in order to provide easier access to the stored variables. It is helpful due to the fact that the top of the stack is constantly changing by pushing and removing of new stack frames.

As mentioned, stack frame also stores the return address of a function. That means when a function is called, its instructions are stored on a different address within the memory. Therefore the actual function who invoked the call needs to store the address of the instructions where to follow after the return from invoked function. The stack-buffer overflow vulnerability happens when the unchecked longer buffer input rewrites the variables' allocated space. This way, the input will rewrite the stored return address within the stack frame. After a return to the function, this fact will either cause a program crash, or in case the input was specifically crafted it will allow the attacker to jump to a desired address in memory where the arbitrary code can be executed. Thus, an attacker can gain privileged control of the system. However, an attacker can aim only on rewriting the contents of variables, or arguments to conduct malicious actions. [9]

To avoid this attack from occurring, a programmer should use safe libraries that re-implement vulnerable functions of *C* language and ensure proper input validation as well as buffer size verification during, for example, copying of buffers. Likewise, the implementation of the stack may be helpful in terms to disallow a direct code execution from within the stack boundaries, or an early detection of attempted attack can mitigate the risk of an actual attack.

2.3.2.6 Drive-by download attack

A drive-by download attack is a complex process that encapsulates several afore-mentioned techniques into a single continuous event. Mostly, all of the attackers aim to launch a drive-by download on the victim's computer. Drive-by download attack occurs when arbitrary content is downloaded into user's computer without user's consent and happens during the action that acts to be benign [6]. This way attacker abuses unsuspecting users and executes malware program in the victim's system in order to steal information or connect the assaulted PC to the botnet, for example. Figure 2.6 illustrates events that happen during an attack. There are 4 main phases that can be distinguished in most drive-by attacks.

1. Malware placement

The first activity of an attacker is to place a malicious script somewhere on the Internet {1}. This can be done either by crafting a dedicated webpage or application for such purpose, or by abusing a genuine website and injecting a malicious script. In case a webpage is created, the attacker needs to lure Internet users to visit the page. For this purpose, email spam containing redirection link can be used as well as blog or forum posts. Otherwise, attackers try to abuse popular pages with high amounts of visitors that are often returned amongst best results of search engines.

2. Webpage visit

When a user navigates to such webpage {2}, the content of the webpage is sent back to him in hand with the embedded malicious script {3}. The received data is processed by user's web browser; displayed and executed {4}. During the procedure, if prepared in such manner, the script can scan user's system to obtain information about versions of operating system, installed browsers, plug-ins and other software. That happens because assaulters try to serve matching exploits that are specific for particular vulnerabilities. Moreover, information about geolocation or visit uniqueness, for example, can be used to make selection whether to attempt the attack or supply a benign webpage to avoid detection of malicious intents. In case an attack is undertaken, a chain of redirections may occur to hide traces of malicious sources {5}. The final spot is the attacker's web storage where an exploit is saved and from which it is distributed to user's machine {6}.

3. Vulnerability exploit

When a user's visit is evaluated by an attacker, desired vulnerability is found and the decision to serve the exploit is made, the exploit is delivered and executed {6}. Thus, the control over user's system can be acquired. The successful exploitation can be accomplished, for instance, by making the processor to jump to a shellcode, a delivered payload, which was injected into the memory space allocated for browser.

4. Malware execution

At this point, the shellcode instructs the system to visit attacker's storage {5} to download the malicious content together with the content the user was looking for while initiated the visit {3}. Once the malicious payload was executed, the attacker is able to perform nearly arbitrary actions.

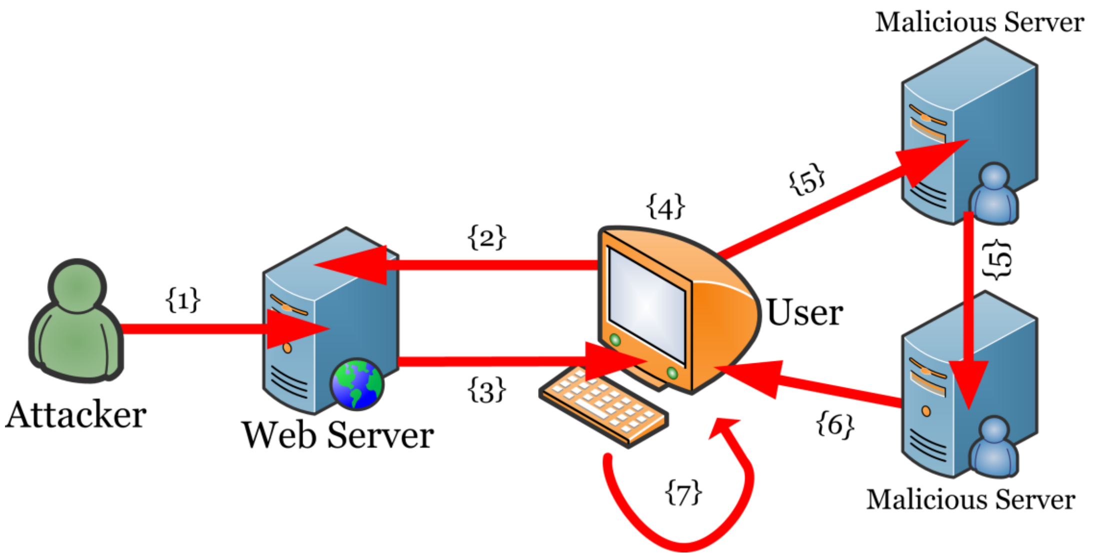


Figure 2.6 Drive-by download attack

As an illustration we mention the ability to start a keylogger to steal credentials, or download additional malware to preserve steady control over the system {7}.

The drive by attack has some challenges for attackers as well as security personnel trying to detect the attack and prevent it from happening. There are various detection approaches, but all of them have the common biggest challenge which is performance. Malware researchers try to achieve as good performance as possible in order to be able to evaluate higher number of websites. The amount of websites on the Internet is changing every minute and the change is rather fast and dynamic. Once a website has been evaluated and flagged as benign there is no guarantee that the website will stay benign for the rest of its lifespan, thus the need of re-visiting and re-evaluating of websites is demanding. Moreover, advanced blackhats are able to craft their malicious web resources in a way to serve and attempt exploitation only under specific circumstances, e.g. particular version of browser and plug-in combination, and otherwise serve a benign variation of the website. That is why a single visit with a single setting may miss the awaiting malware. As already mentioned, another noteworthy challenge for security community is the ability to overcome and unveil attackers' masquerading techniques. Attackers tend to use obfuscation of JavaScript code so the Intrusion Detection Systems are not able to evaluate the meaning of transmitted code in plain text and miss the detection, i.e. a false negative. Recently, also encryption of exploit code starts to appear making the detection even more difficult and demanding. In addition, the encryption makes it also much harder to analyze the intercepted code which is useful in helping to understand attacker's plans. The level of analysis difficulty is determined by the encryption type and encryption key used by attacker. In some cases attackers use location-related parameters as a part of decryption key, thus when a malware is deployed from a different location than it is meant to, the decryption key is computed incorrectly and decrypted exploit code does not make sense.

From attacker's point of view, there are several challenges as well. The foremost challenge is to seduce a user to visit the malicious webpage. This can be achieved in various ways; nonetheless the goal is to lure as many unique visitors as possible. Following challenge is to correctly evaluate user's system in order to find vulnerabilities. For advanced hackers, finding unknown vulnerabilities and launching zero-day attacks is a challenge. During an actual exploit attempt an attacker is challenged to locate the shellcode he injected into the memory and is desired to be executed for the sake of a successful exploit delivery. To ease this task, attackers append shellcode injection with NOP[14] (no operation) instruction sequences. Ultimately, the ability to avoid detection is a great challenge. For example, skills to disclose a visit by virtual machine that pretends to be a genuine user are supporting for such task as well as delayed exploitation, or visit uniqueness.

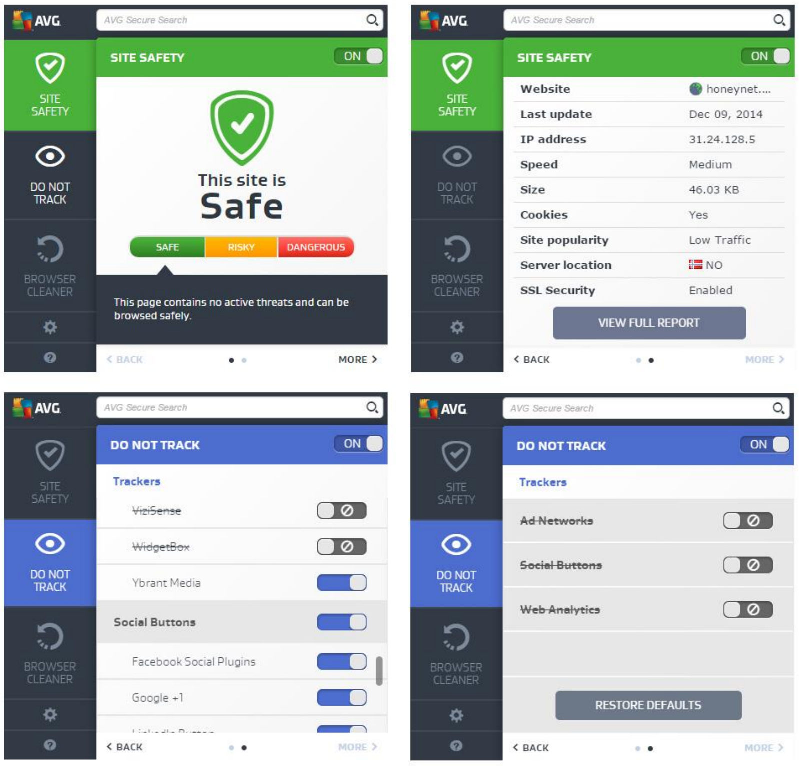
[14] NOP instruction has no operational value besides upon executing this instruction, the pointer is passed to the following instruction in the queue for processing

2.3.3 Protection tools

These tools are just another layer in computer’s security that can be achieved by a user. Such tools are guiding users where it is safe to browse or otherwise. Moreover, a conventional antivirus engine should be running on the user’s system. There is a plenty of vendors implementing their detection techniques and operation principles for the purpose of achieving users’ respective security. The antivirus engine is able to monitor system’s behavior and the files that are in transmission between user’s computer and external sources. However, such engines can only detect previously-known threats that are defined using signatures issued by engine’s developers and behavioral heuristics to the certain extent. Thus, antivirus engines are not prone to detect all malicious activity and can be circumvented. The biggest struggle for antivirus companies is the fact that viruses are constantly evolving. This way an antivirus may miss to detect slightly altered virus which can attempt an attack before the definition was added to vendor’s database and updated in end-user’s computer.

2.3.3.1 AVG Web TuneUp

AVG Company offers a free browser extension that can help users to navigate the Internet more safely and brings some features as well. Currently, the plug-in can be downloaded for Windows based computers running Internet Explorer, Google Chrome or Mozilla Firefox. There is a database behind this plug-in which bears rating information gathered from continuous scan of the Internet. The rating is later displayed to a user and has three levels of severity as figure 2.7 shows. More information can be display for a website a user is viewing. Web TuneUp plug-in is capable of blocking popular information trackers that collect browsing data of users. There are three areas of possible tracking that can be blocked. Additionally, a browser clean up functionality is also a part of the plug-in which makes the deletion of files related to browsing history more straightforward.



SITE SAFETY ON

This site is Safe

SAFE RISKY DANGEROUS

This page contains no active threats and can be browsed safely.

Website	honeynet....
Last update	Dec 09, 2014
IP address	31.24.128.5
Speed	Medium
Size	46.03 KB
Cookies	Yes
Site popularity	Low Traffic
Server location	NO
SSL Security	Enabled

DO NOT TRACK ON

Trackers

~~ViziSense~~

~~WidgetBox~~

Ybrant Media

Social Buttons

Facebook Social Plugins

Google +1

DO NOT TRACK ON

Trackers

~~Ad Networks~~

~~Social Buttons~~

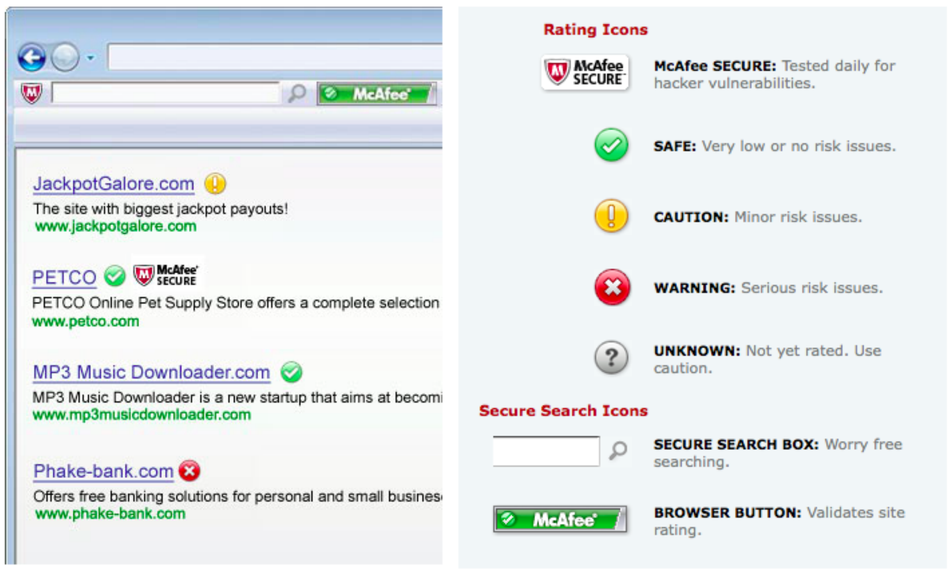
~~Web Analytics~~

Figure 2.7 AVG Web TuneUp

2.3.3.2 McAfee SiteAdvisor

McAfee SiteAdvisor comes in a form of a web browser plug-in, specifically for Mozilla Firefox, Microsoft Internet Explorer and Google Chrome. McAfee uses a set of computers to crawl the Internet and scans for malicious activity. The service classifies visited sites with 4 different categories. Classification is stored in a database. After user installs the plug-in into his browser and navigates the web, a classification based on a record from the database is shown for every visited website and helps the user to determine the safety of website. The plug-in also shows notifications near every hyperlink embedded in a website as figure 2.8 illustrates.

The McAfee site ratings are determined by analyzing multiple areas such as downloads that occur, emails that are received after signing up, browser exploits, links redirection. There is a subscription option available for website developers, which provides their website with scans by McAfee on a regular basis. Often popular e-commerce websites sign in for this feature to provide security for visitors, because after passing the test a website is marked *McAfee-SECURE* site. SiteAdvisor also works with most popular web search engines like Yahoo, Bing, Google, etc. displaying security ratings on search result pages to help protect users. [19]


JackpotGalore.com
The site with biggest jackpot payouts!
www.jackpotgalore.com

PETCO
PETCO Online Pet Supply Store offers a complete selection
www.petco.com

MP3 Music Downloader.com
MP3 Music Downloader is a new startup that aims at becomi
www.mp3musicdownloader.com

Phake-bank.com
Offers free banking solutions for personal and small busines
www.phake-bank.com

Rating Icons

McAfee SECURE: Tested daily for hacker vulnerabilities.

SAFE: Very low or no risk issues.

CAUTION: Minor risk issues.

WARNING: Serious risk issues.

UNKNOWN: Not yet rated. Use caution.

Secure Search Icons

SECURE SEARCH BOX: Worry free searching.

BROWSER BUTTON: Validates site rating.

Figure 2.8 McAfee SiteAdvisor [19]

The advantage of this service is its easy installation and immediate availability as well as it covers a broad range of websites. However, it is not prone to false positives or false negatives which may occur. This drawback happens because of a long time-span between repeated scans; therefore it is not guaranteed that the website is still safe.

2.3.3.3 Sandboxie

Sandboxie is a piece of software designed to secure computer user's operating system and mitigate the risk of getting infected by malware. The tool creates a virtual layer inside the operating system, a so-called sandbox. As illustrated in the figure 2.9, standard behavior is that applications have access to computer's memory resources with operating system's consent and are allowed to make changes. These changes, such as file creation or modification, are stored in persistent memory distributed throughout different memory addresses. On the contrary, Sandboxie allocates a monolithic block of memory where all system modifications are recorded under a close supervision by the toolkit; the areas of operating system are as follows: files, hard-disk services, registry keys, processes and threads, drivers, and objects of inter-process communication. The full list in hand with detailed hierarchy can be found on the tool's website. [29] This way, user applications (web browsers, email and chat clients, games, etc.) are launched within the boundaries

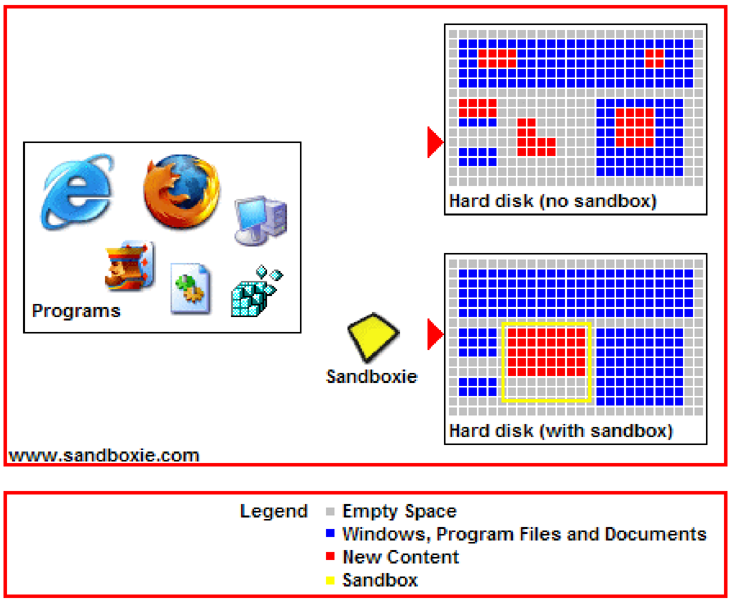


Figure 2.9 Sandboxie memory use logic [29]

of a sandbox where it is easier to monitor actions that take place while particular program is running.

Developers also implemented necessary mechanisms to prevent events when a potentially malicious application, running in a sandbox, could hijack non-sandboxed programs and thus infiltrate the system. Moreover, programs in sandboxed mode are prohibited to load system drivers, which is useful to avoid installation of rootkits. Thanks to the highly customizable set of settings, it is possible to exclude user files that are allowed to be read by programs from sandbox environment.

A user can create multiple sandbox instances at the same time, which may be useful to isolate different programs and diminish the impact on the system that may occur in case of malware execution. If malicious software is noticed to be running, it is easy to clean-up the sandbox with a few clicks of a mouse. In this manner, all files in the particular sandbox are deleted and the threat is dismissed. In computer security context, Sandboxie may be seen as a sort of a virtual machine embedded into the system due to possibilities it offers. Although, this tool may be powerful in defending against threats it is not wise to abandon conventional security measures like anti-virus engines considering the fact that security holes occur time to time that allow attacker to bypass sandboxes environment and to penetrate into the actual operating system.

For malware researchers an add-on called *Buster Sandbox Analyzer*[15] can be of interest. This package is built upon the actual Sandboxie and gives a spectator overview of actions happening in the system during an application run.

2.3.4 Analysis tools

There are tools available for download and offline use after installation, as well as there are tools that can be used right after opening the webpage of the respective service. Hereby are presented some tools that work on demand.

2.3.4.1 urlQuery.net

By visiting the website http://urlquery.net, you can access a free online service to test a given URL for suspicious or malicious behavior. Developers try to contribute their own detection engine in addition to use of other detection tools, for instance Intrusion Detection Systems using a default set of signatures yet leaving out unrelated services and protocols of IDS like FTP, SMTP, ICMP and similar. [36] Figure 2.10 shows a sample result page after scanning a webpage containing malicious content. The report is split into categories of information regarding the threat discovered. At first, general information about the submitted URL and *UserAgent*[16] field of the browser used to visit the URL are shown. Later on, a documentation concerning security is displayed, fields giving more detailed explanation about

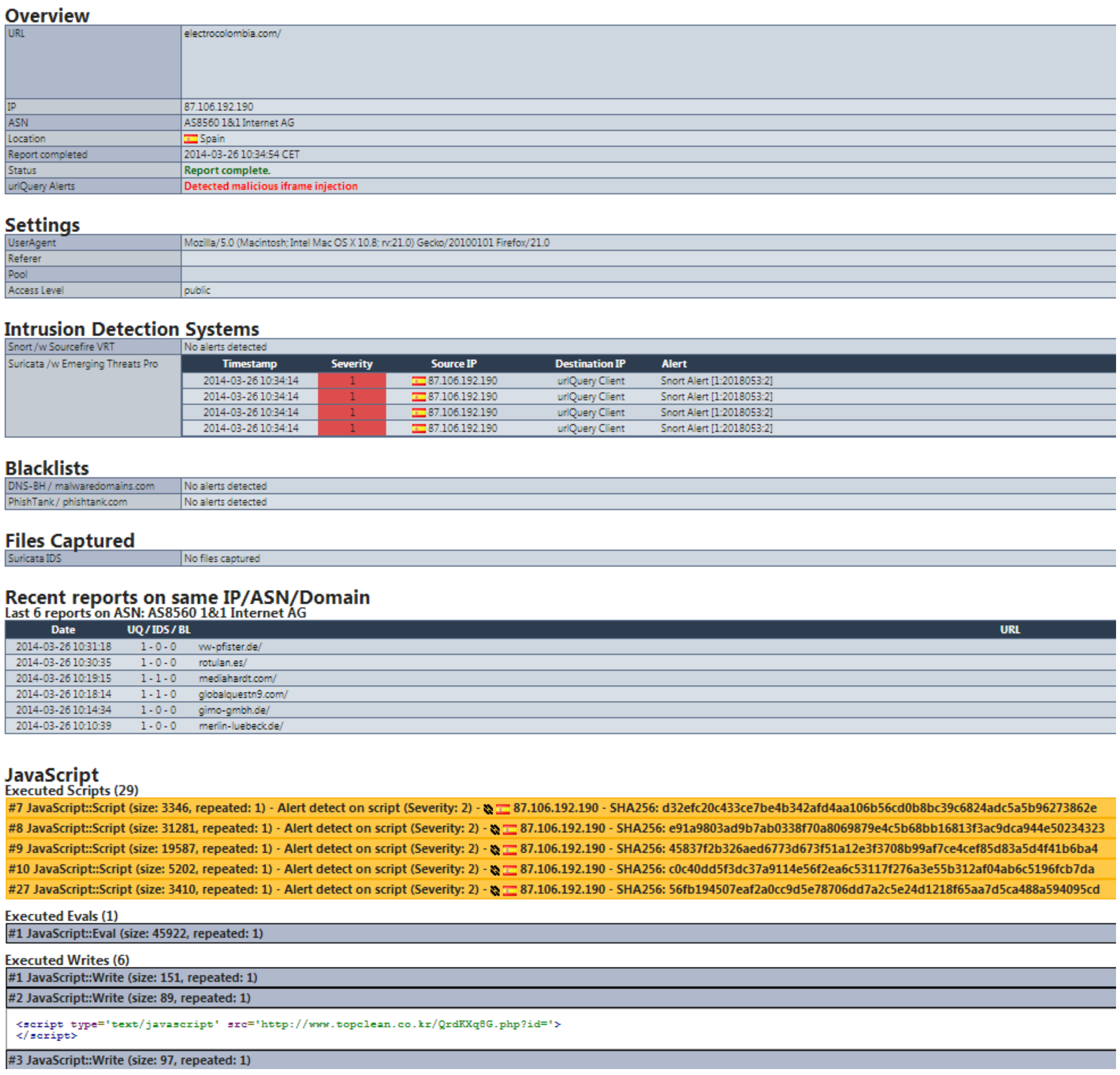
- alerts raised by IDS
- whether the URL was found on a blacklist of URLs
- if a file is offered for download upon visiting a URL, accompanied by VirusTotal rating in case a file is present

[15] http://bsa.isoftware.nl/

[16] UserAgent field carries identification information regarding the version of user's web browser; it is often used for sorting the web content passed to users due to different capabilities of a particular browser

- section devoted to JavaScript code found on the visited webpage where a user can see the complete code of executed scripts just by clicking on a chosen branch
- section (not shown in the figure) observing all HTTP transactions (request – response) is present and depicts all redirects during a visit of a website

At the moment of writing of this thesis, developers were working on providing users with an API, but it was still closed for beta testing.



Overview

URL	electrocolombia.com/
IP	87.106.192.190
ASN	AS8560 1&1 Internet AG
Location	Spain
Report completed	2014-03-26 10:34:54 CET
Status	Report complete.
urlQuery Alerts	Detected malicious iframe injection

Settings

UserAgent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:21.0) Gecko/20100101 Firefox/21.0
Referer	
Pool	
Access Level	public

Intrusion Detection Systems

Snort /w Sourcefire VRT: No alerts detected

Suricata /w Emerging Threats Pro:

Timestamp	Severity	Source IP	Destination IP	Alert
2014-03-26 10:34:14	1	87.106.192.190	urlQuery Client	Snort Alert [1:2018053:2]
2014-03-26 10:34:14	1	87.106.192.190	urlQuery Client	Snort Alert [1:2018053:2]
2014-03-26 10:34:14	1	87.106.192.190	urlQuery Client	Snort Alert [1:2018053:2]
2014-03-26 10:34:14	1	87.106.192.190	urlQuery Client	Snort Alert [1:2018053:2]

Blacklists

DNS-BH / malwaredomains.com	No alerts detected
PhishTank / phishtank.com	No alerts detected

Files Captured

Suricata IDS	No files captured

Recent reports on same IP/ASN/Domain
Last 6 reports on ASN: AS8560 1&1 Internet AG

Date	UQ / IDS / BL	URL
2014-03-26 10:31:18	1 - 0 - 0	vw-pfister.de/
2014-03-26 10:30:35	1 - 0 - 0	rotulan.es/
2014-03-26 10:19:15	1 - 1 - 0	mediahardt.com/
2014-03-26 10:18:14	1 - 1 - 0	globalquestn9.com/
2014-03-26 10:14:34	1 - 0 - 0	gimo-gmbh.de/
2014-03-26 10:10:39	1 - 0 - 0	merlin-luebeck.de/

JavaScript
Executed Scripts (29)

#7 JavaScript::Script (size: 3346, repeated: 1) - Alert detect on script (Severity: 2) - 87.106.192.190 - SHA256: d32efc20c433ce7be4b342afd4aa106b56cd0b8bc39c6824adc5a5b96273862e
#8 JavaScript::Script (size: 31281, repeated: 1) - Alert detect on script (Severity: 2) - 87.106.192.190 - SHA256: e91a9803ad9b7ab0338f70a8069879e4c5b68bb16813f3ac9dca944e50234323
#9 JavaScript::Script (size: 19587, repeated: 1) - Alert detect on script (Severity: 2) - 87.106.192.190 - SHA256: 45837f2b326aed6773d673f51a12e3f3708b99af7ce4cef85d83a5d4f41b6ba4
#10 JavaScript::Script (size: 5202, repeated: 1) - Alert detect on script (Severity: 2) - 87.106.192.190 - SHA256: c0c40dd5f3dc37a9114e56f2ea6c53117f276a3e55b312af04ab6c5196fcb7da
#27 JavaScript::Script (size: 3410, repeated: 1) - Alert detect on script (Severity: 2) - 87.106.192.190 - SHA256: 56fb194507eaf2a0cc9d5e78706dd7a2c5e24d1218f65aa7d5ca488a594095cd

Executed Evals (1)

#1 JavaScript::Eval (size: 45922, repeated: 1)

Executed Writes (6)

#1 JavaScript::Write (size: 151, repeated: 1)
#2 JavaScript::Write (size: 89, repeated: 1)

```
<script type='text/javascript' src='http://www.topclean.co.kr/QrdKXq8G.php?id='>
</script>
```

#3 JavaScript::Write (size: 97, repeated: 1)

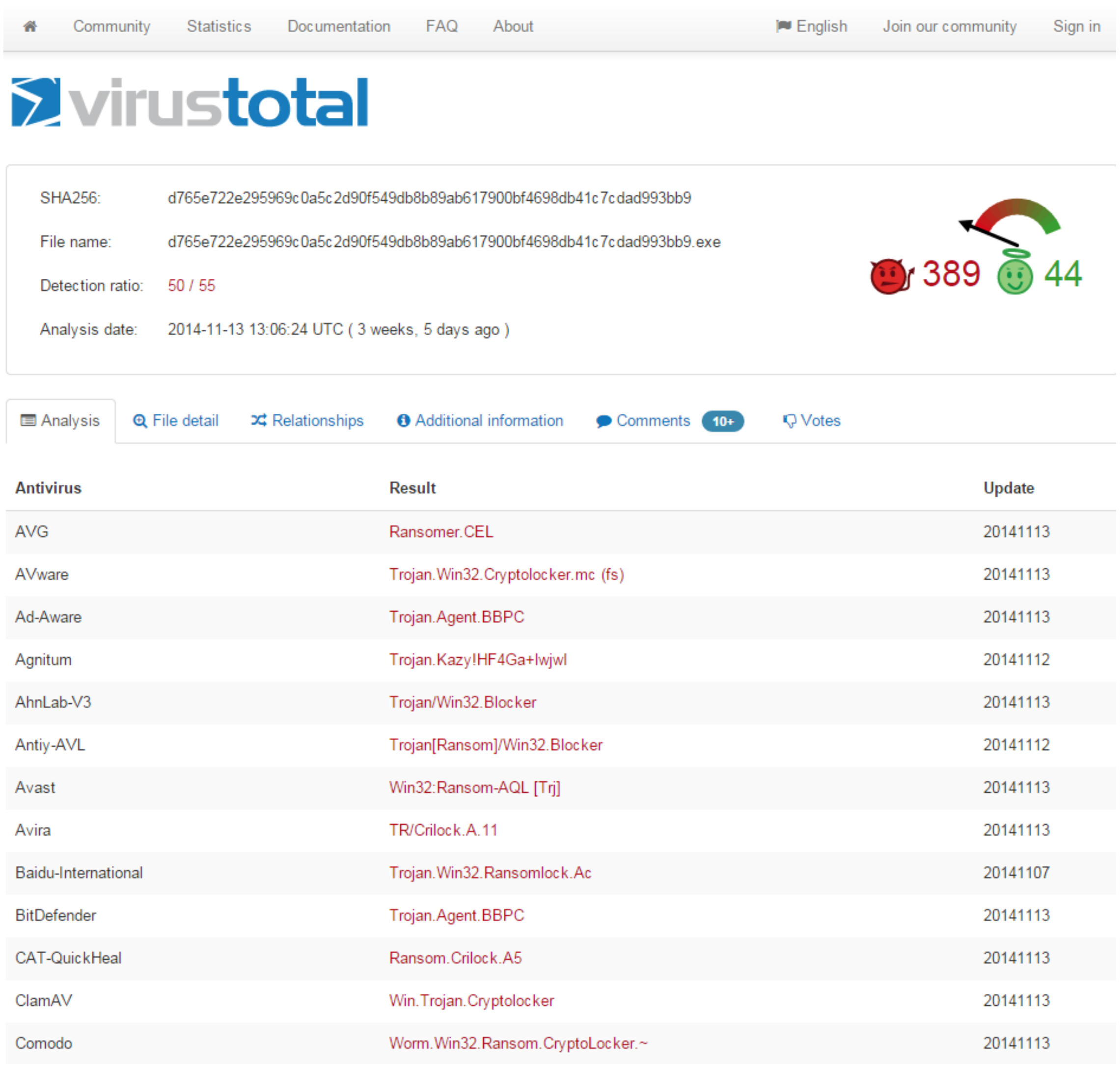
Figure 2.10 urlQuery.net report page

2.3.4.2 VirusTotal.com

VirusTotal is another free online service where you can submit a suspicious link or a file to conduct detailed analysis in order to determine whether the resource is or is not malicious. However, VirusTotal works more as an information aggregator than a scanning framework itself and neither does it work like a conventional antivirus software solution. It utilizes information gathered by various antivirus products, website scanning engines or file characterization tools of which antiviruses like AVG, ESET, Avira, Symantec or Kaspersky can be mentioned. [39] The full list of antiviruses, scanners and tools utilized can be found on the VirusTotal's website. Moreover, a community network was started in 2010. It allows users to comment on files and URLs which is a good contribution to improve system's accuracy, for example avoiding false positives by users conducting a deep malware analyses and sharing their findings with other users.

As an outcome of VirusTotal's design, the service is supporting wide range of file formats for scan, e.g. Windows executables, PDFs, images, JavaScript. Another advantage is that malware signatures are as fresh as they are issued by antivirus developers. Every single scan generates dataset that is stored in database. After submitting a resource that has been scanned any time before, the latest report is shown to the user. Additionally, there is an option to re-scan the resource at the moment of submission. User also has an option to search the database for a malware based on its hash string, specifically MD5, SHA1, SHA256 functions. There are several possibilities how to submit a resource for scan. The main method is to use web interface where you can choose a desired option and receive results as quick as possible because the web interface has the highest scanning priority assigned. The next method to submit a resource is by using an email where a suspicious file is sent as an attachment and the report is replied to the user by email. Furthermore, the project's team has implemented extensions for Mozilla Firefox, Google Chrome and Internet Explorer to integrate its functionality within web browsers and make the usage faster. Standalone software is available as well - a file uploader that makes files submission more straightforward. For the purpose of automating the process, an API is ready to use. However, the free variant is limited to 4 requests per one minute, making a total of 5760 requests a day. It is possible to ask for a private

API key which has no such request limitations in addition to providing a bigger set of data concerning the scanned resource. [38]



SHA256: d765e722e295969c0a5c2d90f549db8b89ab617900bf4698db41c7cdad993bb9
File name: d765e722e295969c0a5c2d90f549db8b89ab617900bf4698db41c7cdad993bb9.exe
Detection ratio: 50 / 55
Analysis date: 2014-11-13 13:06:24 UTC (3 weeks, 5 days ago)

Antivirus	Result	Update
AVG	Ransomer.CEL	20141113
AVware	Trojan.Win32.Cryptolocker.mc (fs)	20141113
Ad-Aware	Trojan.Agent.BBPC	20141113
Agnitum	Trojan.Kazy!HF4Ga+lwjwl	20141112
AhnLab-V3	Trojan/Win32.Blocker	20141113
Antiy-AVL	Trojan[Ransom]/Win32.Blocker	20141112
Avast	Win32:Ransom-AQL [Trj]	20141113
Avira	TR/Crilock.A.11	20141113
Baidu-International	Trojan.Win32.Ransomlock.Ac	20141107
BitDefender	Trojan.Agent.BBPC	20141113
CAT-QuickHeal	Ransom.Crilock.A5	20141113
ClamAV	Win.Trojan.Cryptolocker	20141113
Comodo	Worm.Win32.Ransom.CryptoLocker.~	20141113

Figure 2.11 VirusTotal.com report page

Figure 2.11 shows how a report of a malicious resource looks like after submitting it via web interface. In this case, the sample was already scanned and upon submitting the hash was found in the database hence the report was loaded and displayed.

2.3.4.3 Malwr.com

Malwr.com online scanning service is also free for public use. The goal is similar to the goal of VirusTotal.com website, yet it utilizes different scanning and operations principle. This website is founded by security researchers that are developing high-interaction Cuckoo Sandbox honeypot, which is described in more detail in section 3.2.3. Due to this fact, Malwr.com website uses mainly this honeypot to conduct a dynamic analysis of submitted samples, but also compares the sample against collected intelligence of VirusTotal service and appends the findings to the analysis result. By the words of founders, it is a non-commercial project that does not make any profit from files uploaded by users and aims at public sharing of these files. Despite that, developers value user's privacy and the decision whether a file is shared (available for download by other users) or not, depends on the uploader's choice. As an addition to submitting files directly from the website's interface, there is an API available for automation of submission process. Upon creating a user's account, a private API key is generated for the account. The API is not limited to a number of submissions, yet it is desirable to treat this opportunity wisely and not to drain resources of the offered service.

Samples submitted to the service are identified based on their MD5 hash. After signing in, users are able to search for previously submitted malware. The main search criterion is the hash string of MD5, SHA1, SHA256 and SHA512 functions. Additionally, it is possible to specify other search criterions, such as *filename, file type, signatures* or *string contained, opened registry keys* and others. While visiting the website, users are able to browse results of recent submissions that are ordered chronologically beginning from the latest one. The list contains hash string, filename, file type and number of detections by antivirus engines.

The particular analysis result contains extensive information about the sample. The output is split into several areas:

Quick Overview. Shows basic file details, gives possibility to download the sample (if shared by uploader), behavioral signatures, screenshots of virtual machine during the analysis and files, registry keys and mutexes[17] accessed by the file.

[17] Mutex is a synchronization object that is responsible to ensure mutual exclusion of multiple threads trying to access the same shared file at once

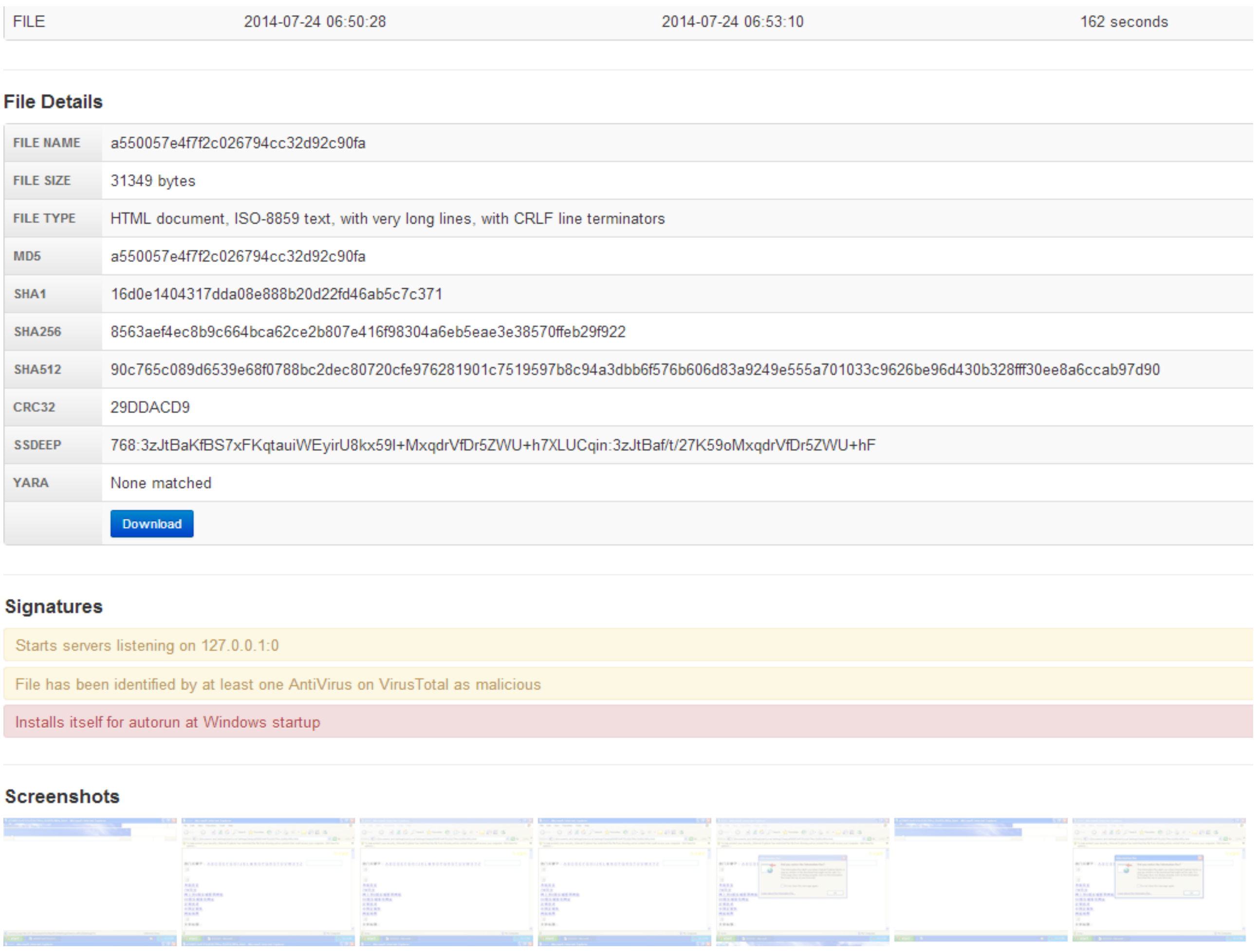
Static Analysis. Shows memory addresses admissions, imports of libraries, strings discovered and antivirus engines detections.

Behavioral Analysis. Shows exhaustive details about actions taken by executables in sense of network, filesystem, registry, services, synchronization activity by processes captured on the system level.

Network Analysis. Shows details about contacted network places, specifically domains, hosts, HTTP, IRC or SMTP.

Dropped Files. Hash strings, file types information of files which download was invoked during sample execution and stored in local storage.

Community comments.



FILE	2014-07-24 06:50:28	2014-07-24 06:53:10	162 seconds

File Details

FILE NAME	a550057e4f7f2c026794cc32d92c90fa
FILE SIZE	31349 bytes
FILE TYPE	HTML document, ISO-8859 text, with very long lines, with CRLF line terminators
MD5	a550057e4f7f2c026794cc32d92c90fa
SHA1	16d0e1404317dda08e888b20d22fd46ab5c7c371
SHA256	8563aef4ec8b9c664bca62ce2b807e416f98304a6eb5eae3e38570ffeb29f922
SHA512	90c765c089d6539e68f0788bc2dec80720cfe976281901c7519597b8c94a3dbb6f576b606d83a9249e555a701033c9626be96d430b328fff30ee8a6ccab97d90
CRC32	29DDACD9
SSDEEP	768:3zJtBaKfBS7xFKqtauiWEyirU8kx59l+MxqdrVfDr5ZWU+h7XLUCqin:3zJtBaf/t/27K59oMxqdrVfDr5ZWU+hF
YARA	None matched
	Download

Signatures

Starts servers listening on 127.0.0.1:0

File has been identified by at least one AntiVirus on VirusTotal as malicious

Installs itself for autorun at Windows startup

Screenshots

Figure 2.12 Malwr Quick Overview page

Figure 2.7 partially illustrates the *Quick Overview* info tab of a submission of a malicious file. There is a helpful feature available – user may receive an email notification after the analysis of user's submitted file is completed. That may be advantageous because submissions are processed depending on their priority and the queue may become quite lengthy.

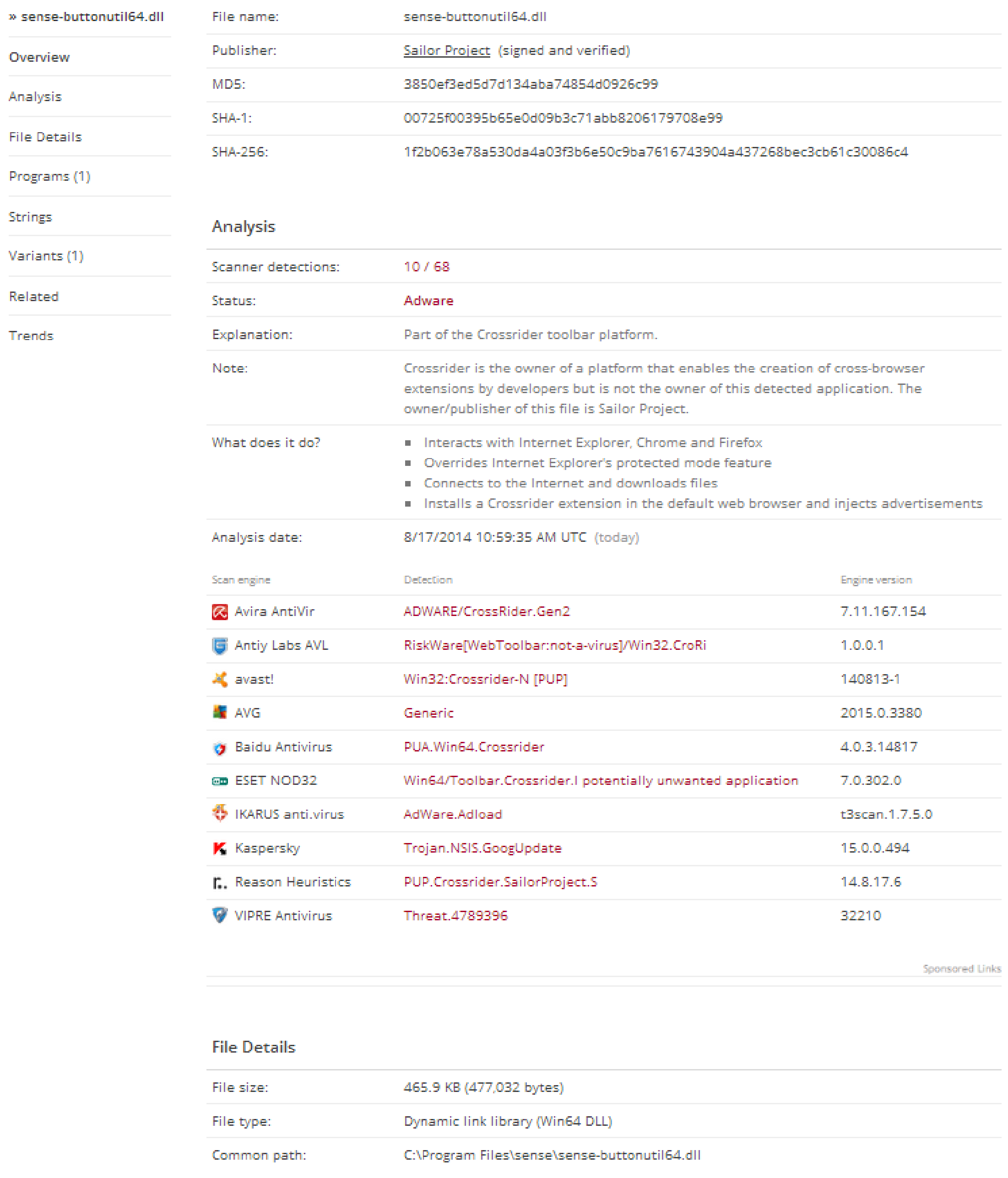
2.3.4.4 herdProtect.com

The project called herdProtect is approaching the malware-defense problem a bit differently. Due to the fact that a single antivirus engine is not able to be 100% effective, herdProtect crafted a platform that utilizes 68 antimalware engines to scan and protect user's computer. Despite that, herdProtect is not a full-plan antimalware protection tool. It is designed to serve as a complementary level of protection to one of actual antivirus engines which should be protecting a user's computer. At the moment of writing, the tool was only capable of scanning on demand, i.e. no real-time protection in the background was a part of the tool. In addition to small antivirus companies, practically all of the major vendors are present. The complete list of used engines can be seen on the project's website.

Likewise the previously-mentioned online services in this section, also herdProtect website offers possibility to search for stored results in the database of already conducted analyses. Although this is not the main purpose of the service, there is a section of the website called knowledge base. It gives a user not only the search capability but also the possibility to browse stored threats that are split into several categories such as detections, URLs, domains, or publishers. On the contrary to the previously-mentioned web services, herdProtect does not offer a possibility to submit a voluntary user input, i.e. URLs to visit, single or multiple files to upload for malware verification. It only collects files that are detected to be suspicious or unknown during the computer scans. Similarly, there is a community section which is held in a *Questions & Answers* fashion. Anyone visiting the website is free to post questions and to provide answers to existing questions.

The main goal of the project is to build a platform that provides scanning functionality. The service can be downloaded directly from the project's website.

After the installation is completed, the tool is able to start scanning. The scan monitors active objects within the operating system. Active objects, as defined by developers, are processes, modules, drivers and similar, that are running or have the ability to automatically execute. The tool takes a snapshot of such file and removes user's personal information. The scan process consists of 4 steps: [10]



» sense-buttonutil64.dll

- Overview
- Analysis
- File Details
- Programs (1)
- Strings
- Variants (1)
- Related
- Trends

File name:	sense-buttonutil64.dll
Publisher:	Sailor Project (signed and verified)
MD5:	3850ef3ed5d7d134aba74854d0926c99
SHA-1:	00725f00395b65e0d09b3c71abb8206179708e99
SHA-256:	1f2b063e78a530da4a03f3b6e50c9ba7616743904a437268bec3cb61c30086c4

Analysis

Scanner detections:	10 / 68
Status:	Adware
Explanation:	Part of the Crossrider toolbar platform.
Note:	Crossrider is the owner of a platform that enables the creation of cross-browser extensions by developers but is not the owner of this detected application. The owner/publisher of this file is Sailor Project.
What does it do?	▪ Interacts with Internet Explorer, Chrome and Firefox ▪ Overrides Internet Explorer's protected mode feature ▪ Connects to the Internet and downloads files ▪ Installs a Crossrider extension in the default web browser and injects advertisements
Analysis date:	8/17/2014 10:59:35 AM UTC (today)

Scan engine	Detection	Engine version
Avira AntiVir	ADWARE/CrossRider.Gen2	7.11.167.154
Antiy Labs AVL	RiskWare[WebToolbar:not-a-virus]/Win32.CroRi	1.0.0.1
avast!	Win32:Crossrider-N [PUP]	140813-1
AVG	Generic	2015.0.3380
Baidu Antivirus	PUA.Win64.Crossrider	4.0.3.14817
ESET NOD32	Win64/Toolbar.Crossrider.I potentially unwanted application	7.0.302.0
IKARUS anti.virus	AdWare.Adload	t3scan.1.7.5.0
Kaspersky	Trojan.NSIS.GoogUpdate	15.0.0.494
Reason Heuristics	PUP.Crossrider.SailorProject.S	14.8.17.6
VIPRE Antivirus	Threat.4789396	32210

Sponsored Links

File Details

File size:	465.9 KB (477,032 bytes)
File type:	Dynamic link library (Win64 DLL)
Common path:	C:\Program Files\sense\sense-buttonutil64.dll

Figure 2.13 herdProtect.com result page

1. At first, herdProtect simply compares the hashes of the sample with the stored signatures in the database.
2. If no match is found, the tool extracts static and behavioral information about the active object and compares it with the database of relevant information.
3. In case no match is found in previous step, the tool will analyze the sample in the sandbox environment with all of 68 engines. Due to the fact that all scanning takes place in the cloud environment, the sample is sent from user's computer to the herdProtect machines and is reported back to the user once the scan is completed.
4. Additionally, herdProtect uses standard industry scanning techniques for detecting offline and binary patched files and rootkits.

If a match is found during the scan process, herdProtect does additional verification in order to exclude false positive detections. Otherwise, the file is flagged for further observation and is re-analyzed when the signatures of antivirus engines are updated. The figure 2.13 depicts results of an analysis of a randomly chosen malicious file. If relevant, the result contains a section where variations of the file or related files are listed.

2.3.5 Malware resources repositories

The Internet offers a quantum of various malware repositories where a user can find and download samples so he can look into the behavior and analyze the threats. Most commonly, samples are look-able based on the file hashes. The download may be conditional – based on the uploaders decision to share the sample or not, for example. Honeypots are complementing matter for computer security, because they help to study the malware in order to form new signatures for instance, and thus help to strengthen the security against malware. Here we mention some of the websites that give a user a chance to download samples.

malwr.com. Gives user a possibility to search by hashes of samples as well as browse recent submissions. The sample sharing depends on the uploader whether he allows his submission to be downloadable.

virustotal.com. Gives user possibility to search for samples, yet by default the sharing is not allowed However, user can request for admission to the download section.

contagiodump.blogspot.cz/2010/11/links-and-resources-for-malware-samples.html. A blogpost containing links to multiple repositories.

malware-traffic-analysis.net/index.html. A blog with descriptions about updates regarding the newest threats that are thoroughly analyzed. URLs where threats had been spotted can be found in the posts as well as links to alternative repositories sharing samples and analyses.

zeustracker.abuse.ch/monitor.php. List of domains that are known to serve ZeuS[18] infection and files associated with this threat.

forums.malwarebytes.org/index.php?/forum/51-newest-malware-threats/. Official forum of Malwarebytes anti-spyware software vendor where community shares and comments can be found.

exploit-db.com. User friendly collection of malware samples.

vxheaven.org. A portal collecting not only samples but also an extended intelligence about viruses such as magazines, whitepapers, tutorial and various utilities.

virusshare.com. Large repository of malware samples to support malware analysis and security community. There are several millions of samples stored.

malware.dontneedcoffee.com. A research community's blog where detailed description about exploits is posted regularly.

shadowserver.org. A project focused on collecting information regarding viruses and cyber-threats rather than sharing samples. The website also bears various statistical data attacks, botnets, scan, viruses and similar.

[18] ZeuS is a wide-spread Trojan horse malware

3 Hands on Client Honeypot

The following chapter presents several honeypot solutions which were selected for a closer look. The presented solutions cover static analysis approach of low-interaction honeypots as well as dynamic analysis approach of high-interactions honeypots.

3.1 Low Interaction

In this section there are three low-interaction client honeypots presented in more detail. Implementations called *HoneyC*, *Thug* and *Yalih* are described.

3.1.1 HoneyC

HoneyC is one of the first honeyclients, which started to form this sector of computer security. It is a project of Christian Seifert that originated in 2006. Nowadays, this project is not alive but a downloadable version is still present on the website of the project. We state this honeyclient for a reference and to see how the entire field developed since then. The honeyclient operates in a manner of low-interaction attitude and instead of having a fully-fledged system in the background it only emulates some services to pretend basic user interaction.

The client honeypot consists of three main components: *Queuer*, *Visitor* and *Analysis Engine*. The responsibility of the *Queuer* is to arrange a set of webpages that the *Visitor* will visit and gather data for analysis. The *Visitor* is a unit that actually opens webpages and interacts with servers. The *Analysis Engine* compares received responses with a set of Snort[19] signatures in order to determine the maliciousness of a visited website. [2]

Due to the fact that HoneyC is published under General Public license, a user is free to alter the components and to craft them to better suit the needs of the user. For example, to make *Queuer* build the set of servers by crawling the Internet, or by interacting with API of one of the available search engines

[19] Snort is a Intrusion Detection System software; https://www.snort.org

and query the API for search results that may be attractive to find exploits. Alternatively, criterions to evaluate maliciousness can be changed in *Analysis Engine*. However, the implementation is now outdated and there are more complex solutions available, which are presented in subsequent sections.

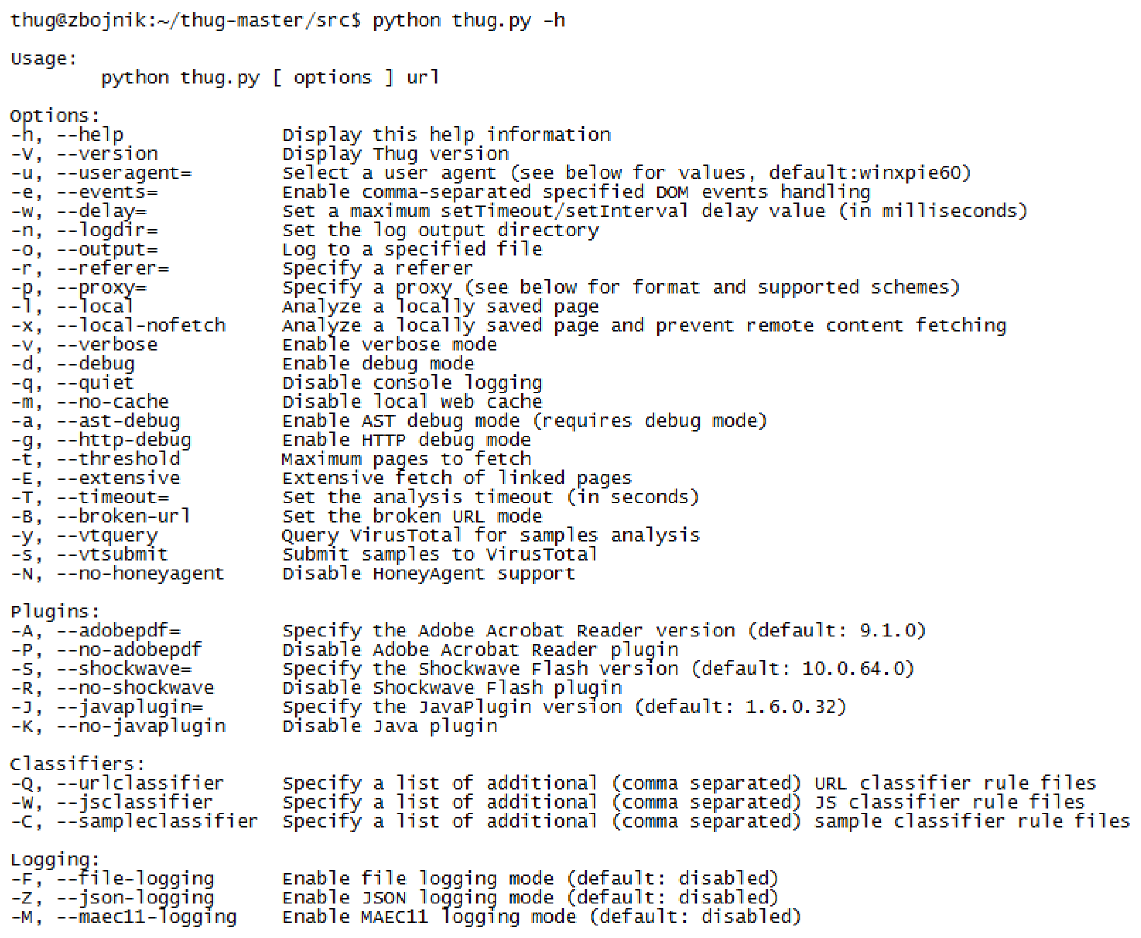
3.1.2 Thug

Thug is a low-interaction honeyclient project that is written in Python programming language and is still in development by Angelo Dell'Aera. Thug's focus is to emulate browser's behavior in order to detect client-side attacks. It utilizes signature matching principle for the sake of analysis. The implementation uses Google V8[20] JavaScript engine to analyze malicious JavaScript code and Libemu[21] library to detect shellcode. As mentioned, the principal area for Thug is browser-application emulation. It focuses on emulation of four most popular browsers these days, i.e. Internet Explorer, Google Chrome, Mozilla Firefox and Safari. The UserAgents, called browser personalities in this particular case, are available in different release versions to broaden the scope of analysis possibilities, which is in line with the fact that attackers tend to serve different content to different browser versions. To supplement the variety, personalities are even available with a different underlying operating system, e.g. Chrome is available as it would be installed in Windows XP, Windows 7, Linux, MacOS X. In order to keep-up with the most recent threats, the personalities of web browser on portable devices are available for emulation. Currently, there are several Android OS devices and Apple iPad that can pretend different browsers and various versions of iOS. In regards to browser plug-ins that experience the highest number of attack attempts, Thug is able to emulate Adobe PDF, Shockwave Flash and Java in versions specified by Thug's administrator. Moreover, a user can define DOM-based events through a parameter for submission query and thus extend the emulation capabilities.

[20] V8 is Google's open source JavaScript engine used in Chrome web browser; more information can be found at https://code.google.com/p/v8/

[21] Libemu is a library for basic x86 emulation and shellcode detection specifically designed for use in IDS and honeypots; more information can be found at http://libemu.carnivore.it/

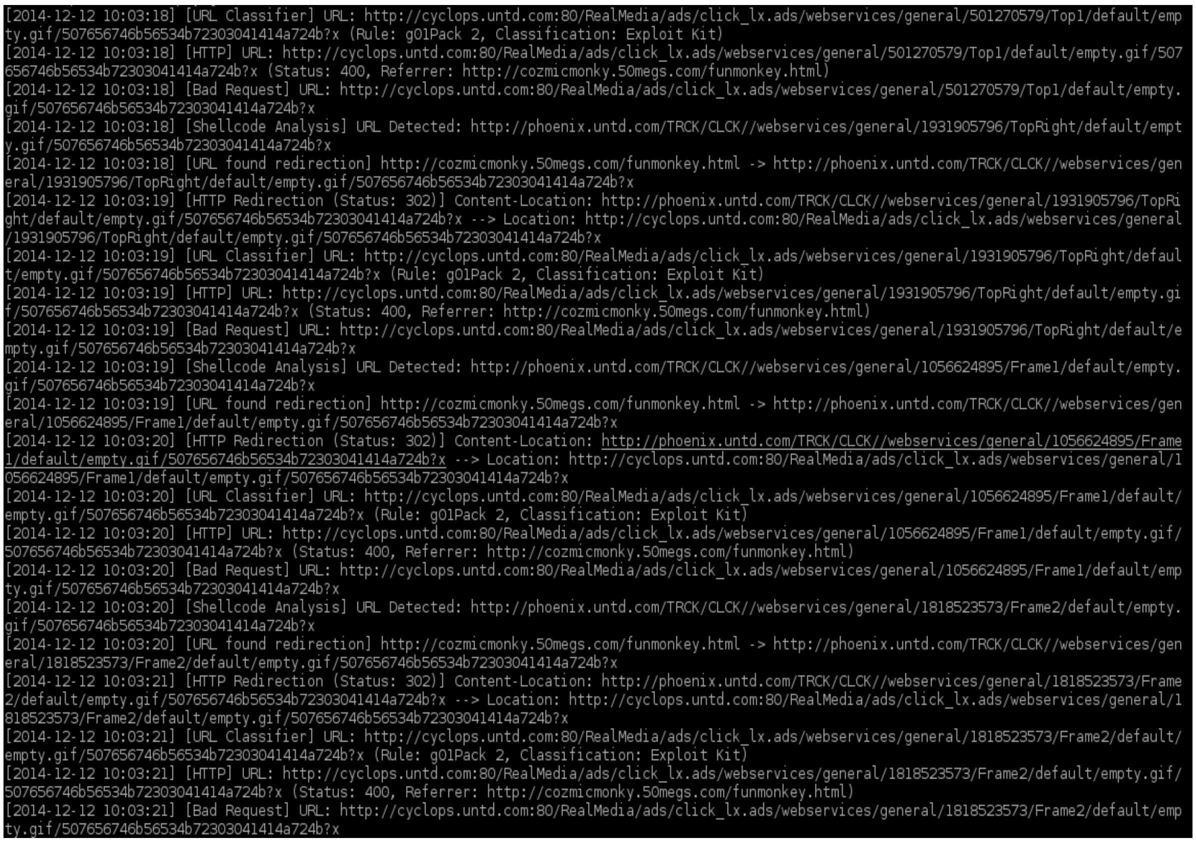
The installation process is a bit lengthy due to the higher amount of dependencies, but there are user-created scripts that can be found on the Internet to make the installation easier. Once the required packages are installed, the honeypot is ready to run right after it has been unpacked from distribution archive. The usage is straightforward and listing 1 shows an output from *help* query to demonstrate available options.



```
thug@zbojnik:~/thug-master/src$ python thug.py -h

Usage:
        python thug.py [ options ] url

Options:
-h, --help                 Display this help information
-V, --version              Display Thug version
-u, --useragent=           Select a user agent (see below for values, default:winxpie60)
-e, --events=              Enable comma-separated specified DOM events handling
-w, --delay=               Set a maximum setTimeout/setInterval delay value (in milliseconds)
-n, --logdir=              Set the log output directory
-o, --output=              Log to a specified file
-r, --referer=             Specify a referer
-p, --proxy=               Specify a proxy (see below for format and supported schemes)
-l, --local                Analyze a locally saved page
-x, --local-nofetch        Analyze a locally saved page and prevent remote content fetching
-v, --verbose              Enable verbose mode
-d, --debug                Enable debug mode
-q, --quiet                Disable console logging
-m, --no-cache             Disable local web cache
-a, --ast-debug            Enable AST debug mode (requires debug mode)
-g, --http-debug           Enable HTTP debug mode
-t, --threshold            Maximum pages to fetch
-E, --extensive            Extensive fetch of linked pages
-T, --timeout=             Set the analysis timeout (in seconds)
-B, --broken-url           Set the broken URL mode
-y, --vtquery              Query VirusTotal for samples analysis
-s, --vtsubmit             Submit samples to VirusTotal
-N, --no-honeyagent        Disable HoneyAgent support

Plugins:
-A, --adobepdf=            Specify the Adobe Acrobat Reader version (default: 9.1.0)
-P, --no-adobepdf          Disable Adobe Acrobat Reader plugin
-S, --shockwave=           Specify the Shockwave Flash version (default: 10.0.64.0)
-R, --no-shockwave         Disable Shockwave Flash plugin
-J, --javaplugin=          Specify the JavaPlugin version (default: 1.6.0.32)
-K, --no-javaplugin        Disable Java plugin

Classifiers:
-Q, --urlclassifier        Specify a list of additional (comma separated) URL classifier rule files
-W, --jsclassifier         Specify a list of additional (comma separated) JS classifier rule files
-C, --sampleclassifier     Specify a list of additional (comma separated) sample classifier rule files

Logging:
-F, --file-logging         Enable file logging mode (default: disabled)
-Z, --json-logging         Enable JSON logging mode (default: disabled)
-M, --maec11-logging       Enable MAEC11 logging mode (default: disabled)
```

Listing 1 Output from Thug's help query

At the time of writing of this thesis, Thug is only able to accept a single URL as an input. There is no automation mechanism as a part of the implementation. This can be solved easily. For this purpose, for example, a simple *bash* script that accepts a *.txt* file as an input, where a single URL-per-line is stored, reads the URL and passes it to Thug for processing is completely sufficient. The figure 3.1 shows a sample output during Thug run upon visiting a malicious webpage that was launched in the test environment during experimentations with honeyclient. Additionally, Thug package that is available for download contains sample exploit files that may help a user to test functionality of the honeypot.



```
[2014-12-12 10:03:18] [URL Classifier] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/501270579/Top1/default/emp
ty.gif/507656746b56534b72303041414a724b?x (Rule: g01Pack 2, Classification: Exploit Kit)
[2014-12-12 10:03:18] [HTTP] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/501270579/Top1/default/empty.gif/507
656746b56534b72303041414a724b?x (Status: 400, Referrer: http://cozmicmonky.50megs.com/funmonkey.html)
[2014-12-12 10:03:18] [Bad Request] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/501270579/Top1/default/empty.
gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:18] [Shellcode Analysis] URL Detected: http://phoenix.untd.com/TRCK/CLCK//webservices/general/1931905796/TopRight/default/empt
y.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:18] [URL found redirection] http://cozmicmonky.50megs.com/funmonkey.html -> http://phoenix.untd.com/TRCK/CLCK//webservices/gen
eral/1931905796/TopRight/default/empty.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:19] [HTTP Redirection (Status: 302)] Content-Location: http://phoenix.untd.com/TRCK/CLCK//webservices/general/1931905796/TopRi
ght/default/empty.gif/507656746b56534b72303041414a724b?x --> Location: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general
/1931905796/TopRight/default/empty.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:19] [URL Classifier] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1931905796/TopRight/defaul
t/empty.gif/507656746b56534b72303041414a724b?x (Rule: g01Pack 2, Classification: Exploit Kit)
[2014-12-12 10:03:19] [HTTP] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1931905796/TopRight/default/empty.gi
f/507656746b56534b72303041414a724b?x (Status: 400, Referrer: http://cozmicmonky.50megs.com/funmonkey.html)
[2014-12-12 10:03:19] [Bad Request] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1931905796/TopRight/default/e
mpty.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:19] [Shellcode Analysis] URL Detected: http://phoenix.untd.com/TRCK/CLCK//webservices/general/1056624895/Frame1/default/empty.
gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:19] [URL found redirection] http://cozmicmonky.50megs.com/funmonkey.html -> http://phoenix.untd.com/TRCK/CLCK//webservices/gen
eral/1056624895/Frame1/default/empty.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:20] [HTTP Redirection (Status: 302)] Content-Location: http://phoenix.untd.com/TRCK/CLCK//webservices/general/1056624895/Frame
1/default/empty.gif/507656746b56534b72303041414a724b?x --> Location: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1
056624895/Frame1/default/empty.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:20] [URL Classifier] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1056624895/Frame1/default/
empty.gif/507656746b56534b72303041414a724b?x (Rule: g01Pack 2, Classification: Exploit Kit)
[2014-12-12 10:03:20] [HTTP] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1056624895/Frame1/default/empty.gif/
507656746b56534b72303041414a724b?x (Status: 400, Referrer: http://cozmicmonky.50megs.com/funmonkey.html)
[2014-12-12 10:03:20] [Bad Request] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1056624895/Frame1/default/emp
ty.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:20] [Shellcode Analysis] URL Detected: http://phoenix.untd.com/TRCK/CLCK//webservices/general/1818523573/Frame2/default/empty.
gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:20] [URL found redirection] http://cozmicmonky.50megs.com/funmonkey.html -> http://phoenix.untd.com/TRCK/CLCK//webservices/gen
eral/1818523573/Frame2/default/empty.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:21] [HTTP Redirection (Status: 302)] Content-Location: http://phoenix.untd.com/TRCK/CLCK//webservices/general/1818523573/Frame
2/default/empty.gif/507656746b56534b72303041414a724b?x --> Location: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1
818523573/Frame2/default/empty.gif/507656746b56534b72303041414a724b?x
[2014-12-12 10:03:21] [URL Classifier] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1818523573/Frame2/default/
empty.gif/507656746b56534b72303041414a724b?x (Rule: g01Pack 2, Classification: Exploit Kit)
[2014-12-12 10:03:21] [HTTP] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1818523573/Frame2/default/empty.gif/
507656746b56534b72303041414a724b?x (Status: 400, Referrer: http://cozmicmonky.50megs.com/funmonkey.html)
[2014-12-12 10:03:21] [Bad Request] URL: http://cyclops.untd.com:80/RealMedia/ads/click_lx.ads/webservices/general/1818523573/Frame2/default/emp
ty.gif/507656746b56534b72303041414a724b?x
```

Figure 3.1 Sample from Thug run

After visiting a webpage, Thug creates a report that differs based on the content of the particular webpage and the configuration set by administrator. It is capable of saving HTML content as well as CSS content of the website, images or JavaScript elements executed on the website. In case the submitted link points to a downloadable content like *.zip* or *.exe* files and similar, the honeypot stores the content on the hard-drive and thus allows the user to proceed with investigation of potentially malicious content with the help of other tools serving for this purpose. At the same time, Thug performs an analysis and saves the outcome. The honeyclient creates a file using JSON template [22] that stores exact configuration used to visit a webpage and states the behavior or exploits discovered on the webpage. Furthermore, another file

[22] A structured text file to store high amouts of information based on a pre-defined template; https://code.google.com/p/json-template/

in *.xml* structured format regarding to the analysis can be created using MAEC language[23]. There is also a graph of redirections which may be hidden from a user, which occurred during the visit of a webpage. You can see an example in the figure 3.2.

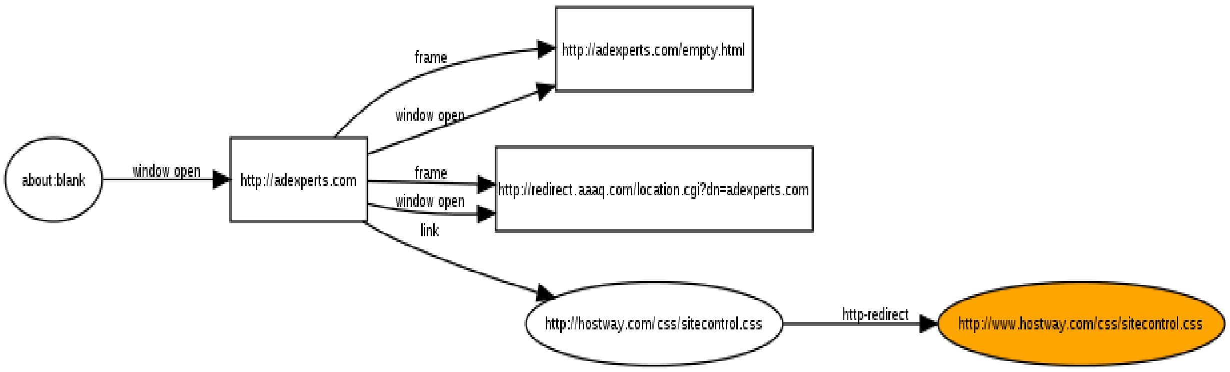

about:blank
window open
http://adexperts.com
frame
window open
http://adexperts.com/empty.html
frame
window open
link
http://redirect.aaaq.com/location.cgi?dn=adexperts.com
http://hostway.com/css/sitecontrol.css
http-redirect
http://www.hostway.com/css/sitecontrol.css

Figure 3.2 Graph of redirections

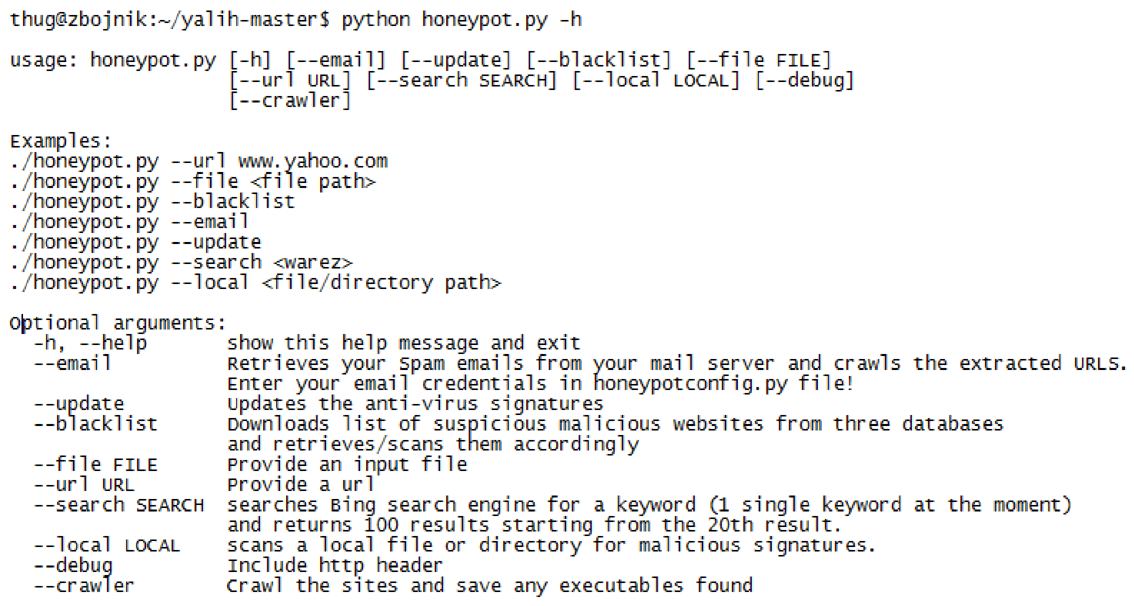
3.1.3 Yalih

Yalih is an abbreviation that stands for "*Yet another low interaction honeyclient*". The name is self-explanatory in this case. Honeyclient is written in Python programming language and has several capabilities, and all of them match the description of low-interaction honeypots, which we clarified previously. It is designed to detect malware mainly by looking for familiar patterns. [18] Signatures are downloaded from databases of AVG and ClamAV antivirus engines that were chosen by developers and are complemented by signatures from *Yara* tool (more information in section 4.1.1).

As we can see in the listing 2, Yalih is able to accept different types of input. It is possible to provide a single URL link as well as a file containing a set of URLs, or a local folder containing single or multiple files for analysis.

[23] MAEC is a standardized language (free for public use) for sharing structured information about malware based upon attributes such as behaviors, artifacts, and attack patterns. MAEC aims to improve communication about malware by eliminating the inaccuracy that exists in malware descriptions and by reducing reliance on signatures. [24] For exhaustive information look into the referenced paper.

Moreover, honeyclient has extended functionality and is able to scan provided email account, extract URLs from messages in mailbox folders and visit discovered links in order to check for malicious activity. User needs to supply login credentials into the configuration file and provide the mailbox address, e.g. *imap.google.com* for Gmail account. Another interesting feature is the ability to query search engine of Bing search service. User inputs a keyword that will be searched by Bing and afterwards links from the first 100 results will be examined; there is also a setting to determine a number of links from the beginning of search results that will be omitted from examination, as it is highly probable that the most popular links are benign. In case user does not have a particular input for the honeyclient, there is an option to scan malware URLs retrieved from blacklist databases. Yalih queries three different websites for their database of malicious or suspicious webpages and scans them



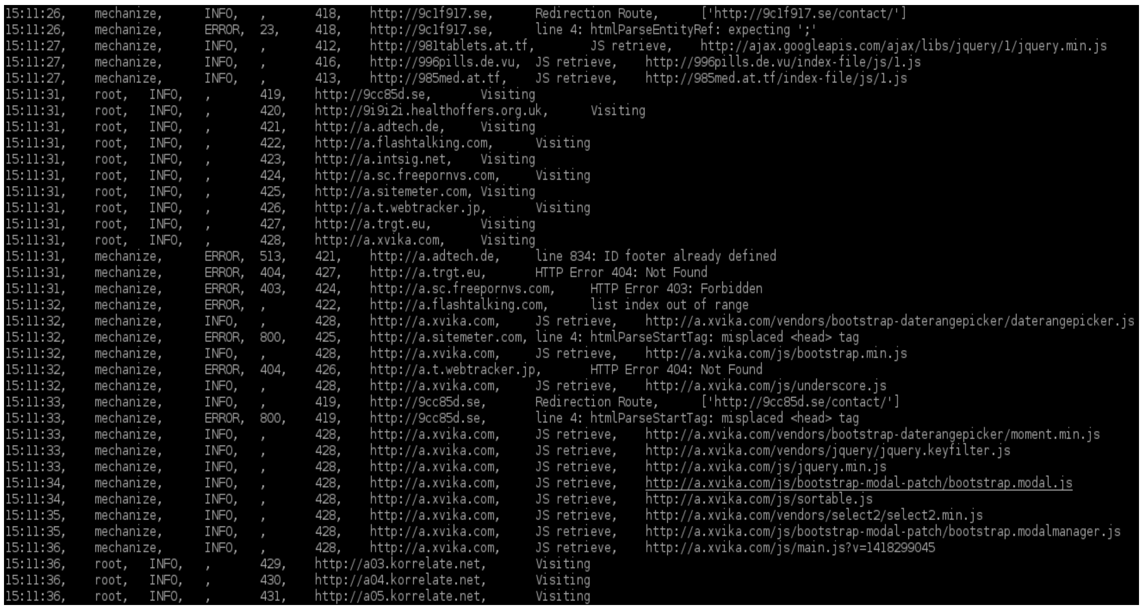
```
thug@zbojnik:~/yalih-master$ python honeypot.py -h

usage: honeypot.py [-h] [--email] [--update] [--blacklist] [--file FILE]
                   [--url URL] [--search SEARCH] [--local LOCAL] [--debug]
                   [--crawler]

Examples:
./honeypot.py --url www.yahoo.com
./honeypot.py --file <file path>
./honeypot.py --blacklist
./honeypot.py --email
./honeypot.py --update
./honeypot.py --search <warez>
./honeypot.py --local <file/directory path>

optional arguments:
  -h, --help       show this help message and exit
  --email          Retrieves your Spam emails from your mail server and crawls the extracted URLS.
                   Enter your email credentials in honeypotconfig.py file!
  --update         Updates the anti-virus signatures
  --blacklist      Downloads list of suspicious malicious websites from three databases
                   and retrieves/scans them accordingly
  --file FILE      Provide an input file
  --url URL        Provide a url
  --search SEARCH  searches Bing search engine for a keyword (1 single keyword at the moment)
                   and returns 100 results starting from the 20th result.
  --local LOCAL    scans a local file or directory for malicious signatures.
  --debug          Include http header
  --crawler        Crawl the sites and save any executables found
```

Listing 2 Output from Yalih's help query

accordingly. The list of webpages is saved in the computer and it can be used later as an input for another honeypot, for example.

Likewise in the previous honeypot, also Yalih is able to emulate several different web browsers, but the list is not so rich. Mozilla Firefox and Internet Explorer have the most versions amongst others. When a downloadable executable is encountered, it is stored for the sake of further investigation.

A possibility to configure a proxy is available that is useful to tamper with the geolocation assigned to the IP address that is used to run Yalih. That is



```
15:11:26,    mechanize,     INFO,  ,      418,    http://9c1f917.se,     Redirection Route,     ['http://9c1f917.se/contact/']
15:11:26,    mechanize,     ERROR, 23,    418,    http://9c1f917.se,     line 4: htmlParseEntityRef: expecting ';'
15:11:27,    mechanize,     INFO,  ,      412,    http://981tablets.at.tf,      JS retrieve,    http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js
15:11:27,    mechanize,     INFO,  ,      416,    http://996pills.de.vu, JS retrieve,    http://996pills.de.vu/index-file/js/1.js
15:11:27,    mechanize,     INFO,  ,      413,    http://985med.at.tf,   JS retrieve,    http://985med.at.tf/index-file/js/1.js
15:11:31,    root,  INFO,  ,      419,    http://9cc85d.se,      Visiting
15:11:31,    root,  INFO,  ,      420,    http://9i9i2i.healthoffers.org.uk,      Visiting
15:11:31,    root,  INFO,  ,      421,    http://a.adtech.de,    Visiting
15:11:31,    root,  INFO,  ,      422,    http://a.flashtalking.com,      Visiting
15:11:31,    root,  INFO,  ,      423,    http://a.intsig.net,   Visiting
15:11:31,    root,  INFO,  ,      424,    http://a.sc.freepornvs.com,     Visiting
15:11:31,    root,  INFO,  ,      425,    http://a.sitemeter.com, Visiting
15:11:31,    root,  INFO,  ,      426,    http://a.t.webtracker.jp,       Visiting
15:11:31,    root,  INFO,  ,      427,    http://a.trgt.eu,      Visiting
15:11:31,    root,  INFO,  ,      428,    http://a.xvika.com,    Visiting
15:11:31,    mechanize,     ERROR, 513,   421,    http://a.adtech.de,    line 834: ID footer already defined
15:11:31,    mechanize,     ERROR, 404,   427,    http://a.trgt.eu,      HTTP Error 404: Not Found
15:11:31,    mechanize,     ERROR, 403,   424,    http://a.sc.freepornvs.com,     HTTP Error 403: Forbidden
15:11:32,    mechanize,     ERROR, ,      422,    http://a.flashtalking.com,      list index out of range
15:11:32,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/vendors/bootstrap-daterangepicker/daterangepicker.js
15:11:32,    mechanize,     ERROR, 800,   425,    http://a.sitemeter.com, line 4: htmlParseStartTag: misplaced <head> tag
15:11:32,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/js/bootstrap.min.js
15:11:32,    mechanize,     ERROR, 404,   426,    http://a.t.webtracker.jp,       HTTP Error 404: Not Found
15:11:32,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/js/underscore.js
15:11:33,    mechanize,     INFO,  ,      419,    http://9cc85d.se,      Redirection Route,     ['http://9cc85d.se/contact/']
15:11:33,    mechanize,     ERROR, 800,   419,    http://9cc85d.se,      line 4: htmlParseStartTag: misplaced <head> tag
15:11:33,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/vendors/bootstrap-daterangepicker/moment.min.js
15:11:33,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/vendors/jquery/jquery.keyfilter.js
15:11:33,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/js/jquery.min.js
15:11:34,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/js/bootstrap-modal-patch/bootstrap.modal.js
15:11:34,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/js/sortable.js
15:11:35,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/vendors/select2/select2.min.js
15:11:35,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/js/bootstrap-modal-patch/bootstrap.modalmanager.js
15:11:36,    mechanize,     INFO,  ,      428,    http://a.xvika.com,    JS retrieve,    http://a.xvika.com/js/main.js?v=1418299045
15:11:36,    root,  INFO,  ,      429,    http://a03.korrelate.net,       Visiting
15:11:36,    root,  INFO,  ,      430,    http://a04.korrelate.net,       Visiting
15:11:36,    root,  INFO,  ,      431,    http://a05.korrelate.net,       Visiting
```

Figure 3.3 Sample from Yalih run

helpful when a user would like to investigate malware presence that may be hiding when a visit from incorrect (from the malware's point of view) location happens. Figure 3.3 shows an output from running Yalih.

Yalih implements some good ideas, which absent in other available implementations, e.g. retrieving and scanning of links from blacklists, or following links found in an email account, but there is a lot of work still to be done in the future. Better optimization of the process, unification of output produced by honeyclient. For example, more sophisticated reporting could be implemented, because at the moment only a common debug log is produced and files are being retrieved from visited websites. Alternatively, more antivirus databases can be added in order to supply broader set of signatures for scanning.

3.2 High Interaction

This subchapter presents solutions that focus on high-interaction paradigm providing the malware with complete operating system to interact with. These tools utilize dynamic analysis in order to evaluate submitted samples.

3.2.1 CaptureHPC

CaptureHPC is a honeyclient developed at Victoria University of Wellington, New Zealand. It was originally released in 2006 as one of the first implementations of high-interaction client side honeypots. The main signature of this honeypot is a server – client architecture. The central server component is responsible for event handling and tasks distribution amongst multiple client components which actually conduct the work. i.e.visit webpages, or utilize any user application in common in order to classify the interaction. Thus, it is not difficult to extend framework's performance and it is done by adding client component instances. Figure 3.4 demonstrates the CaptureHPC set-up along

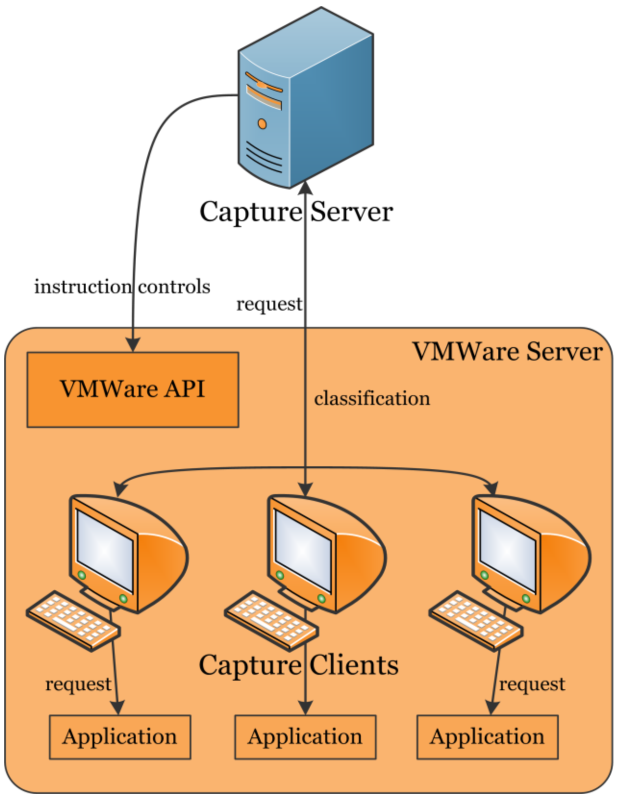


Figure 3.4 CaptureHPC framework scheme

with data flow inside of the framework environment. Client components are launched as virtual machines running Windows operating system. Thanks to various settings possibilities, administrator is able to define, for example, how long a visit would last before the machine shuts down and proceeds to the next URL.

Detection principle has the same outlines as high-interaction honeypots have. The operating system is closely monitored, in this case processes, filesystem, registry entries, in order to spot changes during the application run. Naturally, there are events that are of normal benign behavior, e.g. file manipulation in browser cache folder. These events need to be excluded from the final classification of a resource. For this purpose, framework supports *exclusion lists*. [1] An example of such list regarding file modifications made by Capture client's process is shown in the listing 3. Entries in the list are constructed as regular expressions. Malicious classification then depends on the occurrence of system modifications outside the excluded area.

```
#capture
+       Write   .*      C:\\program files\\capture\\logs\\.+
+       Delete  C:\\program Files\\capture\\captureclient\.exe  C:\\program files\\capture\\.+\.zip
+       Delete  C:\\program Files\\capture\\captureclient\.exe  C:\\program files\\capture\\logs
+       Delete  C:\\program Files\\capture\\captureclient\.exe  C:\\program files\\capture\\logs\\.*
+       Write   C:\\program Files\\capture\\captureclient\.exe  C:\\program files\\capture\\capture\.log
+       Write   C:\\program Files\\capture\\7za\.exe    C:\\program files\\capture\\capture\.log
+       Write   C:\\program Files\\capture\\7za\.exe    C:\\program files\\capture\\.+\.zip
+       Delete  C:\\program Files\\capture\\7za\.exe    C:\\program files\\capture\\.+\.zip
+       Write   C:\\program Files\\capture\\7za\.exe    C:\\progra~1\\capture\\capture\.log
+       Write   C:\\program Files\\capture\\7za\.exe    C:\\progra~1\\capture\\.+\.zip
+       Delete  C:\\program Files\\capture\\7za\.exe    C:\\progra~1\\capture\\.+\.zip
```

Listing 3 File operations exclusion list

Notwithstanding, CaptureHPC was introduced earlier; it still is a powerful tool to detect malicious behavior of user applications. Due to the fact that the framework monitors a complete set of events, which are hidden from unaware user, happening inside of a system, it collects relevant data. Gathered intelligence usually needs a human analysis to support the final decision. This honeyclient implementation is utilized by the HoneySpider Network 2.0 framework that is also described later in this chapter.

The honeyclient implementation is powerful due to the fact that it gives a broad observation of events that happened in the system. The cost for the amount of information is the time needed to conduct a single analysis.

Present-day implementations have much better overall process handling and give even more details about application execution.

3.2.2 Strider HoneyMonkey

Strider HoneyMonkey is a research project founded by Microsoft for the purpose of detecting and analyzing websites hosting malicious code. The framework makes use of high-interaction honeypots interconnected via virtual environment on several physical machines. Thanks to benefits of virtualization (mentioned in section 2.1.4) the project utilizes various configurations of exposed systems, from completely unpatched to (nearly) fully up-to-date systems. HoneyMonkey operates in three stages:

1. Each HoneyMonkey starts by visiting the same large list of URLs in one unpatched VM (virtual machine) with redirection detection switched off. In case of an exploit is detected the machine switches into one-URL-per-VM mode to re-test suspicious links.
2. Found exploit-URLs from the previous stage are being rescanned while recursive redirection tool is enabled, in order to resolve all URLs connected to the exploit.
3. Last stage scans URLs from stage 2 on updated machines to detect threats trying to exploit latest vulnerabilities (optionally zero-day attacks). [41]

This way, researchers are able to look for sites that focus on specific vulnerabilities. Additionally, even zero-day attacks can be detected due to the use of (nearly) fully patched systems. When an updated system is being successfully exploited, it means a zero-day attack has been found.

As researchers state, the detection is being held in so-called *black-box*. After a HoneyMonkey visits a URL it waits for a predefined period of time to allow the exploit to trigger, as it may be delayed in order to hide from detection. During the visit of a webpage the system is monitored for file creation outside the black-boxed area, process creation, registry entries changes and a report is generated which signals about exploitation. [41]

The whole process is supported by some other tools like *Strider GhostBuster Rootkit Detection* which helps to detect hidden processes, i.e. rootkits, and similar. Unfortunately, HoneyMonkey project is not available for download and public use as it is an internal project of Microsoft to help the company to improve security and develop secure applications.

3.2.3 Cuckoo Sandbox

Cuckoo Sandbox is a live open-source high-interaction honeypot project and is still being actively developed and improved. The framework is written mainly in Python language, but also takes advantage of C language. Developers of the project started a website based on Cuckoo Sandbox - www.malwr.com as described in subsection 2.3.4.3, which is available for public use to analyze malware samples and URL links.

Cuckoo is built on the host – guest (server – client) paradigm which is characteristic for high-interaction honeypots and the environment architecture is shown in the figure 3.5.

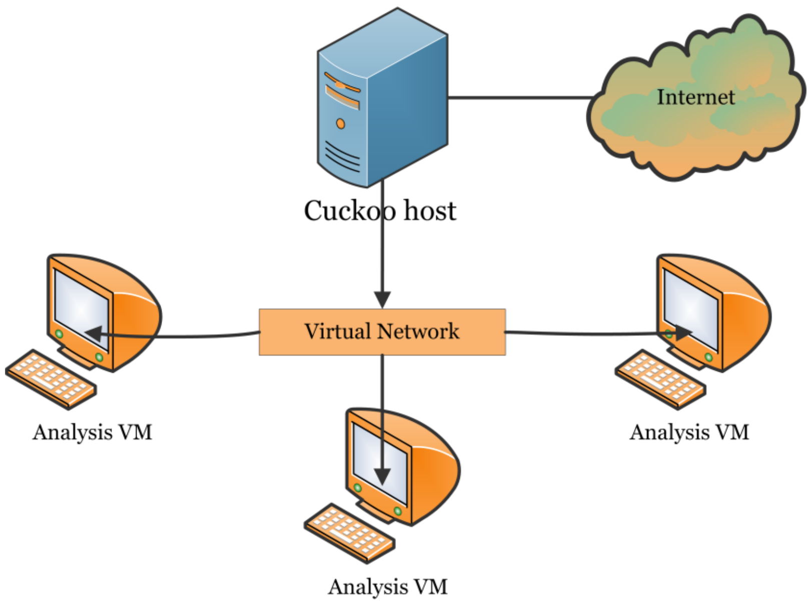


Figure 3.5 Cuckoo Architecture

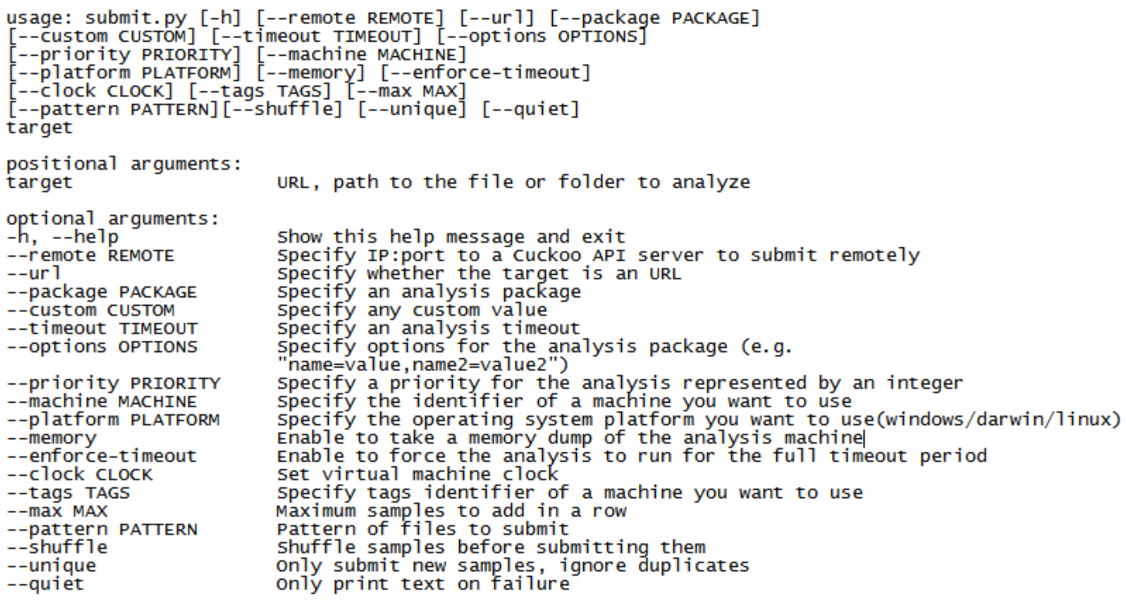
The Cuckoo host machine is connected to both Internet and internal virtual network which interconnects guest virtual machines and acts as the central processing unit that manages the overall malware analysis. The host machine submits malware samples and distributes them amongst guest stations where the actual analysis takes place. The host is also responsible for analysis reports creation and holds the reported data in raw format as well as parsed into database structure (if configured). The guest system is a pre-crafted operating system with Python programming language environment and instances of vulnerable software packages which are to be monitored for exploits. For the ease of use, a snapshot of guest system is made and is restored to the default state before every single analysis in order to be capable to monitor every single activity of a particular malware execution. Such analysis environment makes it possible to deploy multiple instances of analysis stations for the sake of improving speed and vulnerabilities variability of analyses. The Cuckoo honeypot's monitoring competences are designed to capture various data so one may evaluate submitted samples and the competences are as follows:

- win32 API calls raised by processes belonging to malware
- files modification
- memory dumps of processes belonging to malware
- network traffic dumps in PCAP format
- screenshots of VM during the malware execution
- full memory dumps of virtual machines

Above-mentioned information is automatically processed and overall result is presented to a Cuckoo user. Additionally, a complementary analysis can be made by investigating recorded raw data which may be beneficial to avoid false positives or negatives or to uncover malware's behavior, yet it requires advanced knowledge. [8]

The installation process is thoroughly described in the user's manual available on the project's website. The Cuckoo host machine is preferred to have a GNU/Linux based operating system and for the sake of best performance the guest stations are preferred to have Windows XP Service Pack 2/3, but also newer versions of Windows OS will work. The subsequent configuration consists of creating a default state snapshot(s) for virtual machine(s) and proper configuration of the host machine. The whole framework is based on core components which are extended with modules, thus it is possible to integrate almost arbitrary functionality into the framework and enhance the overall solution. Due to this fact, the configuration possibilities highly depend on the possibilities of modules integrated. By default, there are 6 main configurable areas: basic settings for framework itself, settings of interaction with virtualization software, settings for Volatility[24] tool regarding memory dump analysis, configuration of processing modules that define in which way the raw data will be analyzed and lastly settings regarding reporting of acquired information.

Once the Cuckoo framework is successfully installed, a user can submit a malware sample for analysis. The submission can be achieved in several ways. Listing 4 shows available options for submission of a file using */cuckoo/utils/submit.py* utility.



```
usage: submit.py [-h] [--remote REMOTE] [--url] [--package PACKAGE]
[--custom CUSTOM] [--timeout TIMEOUT] [--options OPTIONS]
[--priority PRIORITY] [--machine MACHINE]
[--platform PLATFORM] [--memory] [--enforce-timeout]
[--clock CLOCK] [--tags TAGS] [--max MAX]
[--pattern PATTERN][--shuffle] [--unique] [--quiet]
target

positional arguments:
target                   URL, path to the file or folder to analyze

optional arguments:
-h, --help               Show this help message and exit
--remote REMOTE          Specify IP:port to a Cuckoo API server to submit remotely
--url                    Specify whether the target is an URL
--package PACKAGE        Specify an analysis package
--custom CUSTOM          Specify any custom value
--timeout TIMEOUT        Specify an analysis timeout
--options OPTIONS        Specify options for the analysis package (e.g.
                         "name=value,name2=value2")
--priority PRIORITY      Specify a priority for the analysis represented by an integer
--machine MACHINE        Specify the identifier of a machine you want to use
--platform PLATFORM      Specify the operating system platform you want to use(windows/darwin/linux)
--memory                 Enable to take a memory dump of the analysis machine
--enforce-timeout        Enable to force the analysis to run for the full timeout period
--clock CLOCK            Set virtual machine clock
--tags TAGS              Specify tags identifier of a machine you want to use
--max MAX                Maximum samples to add in a row
--pattern PATTERN        Pattern of files to submit
--shuffle                Shuffle samples before submitting them
--unique                 Only submit new samples, ignore duplicates
--quiet                  Only print text on failure
```

Listing 4 Cuckoo submission options

As a part of the framework, there is a small web interface, which serves mainly for displaying results of analyses, but it also provides a possibility to submit a sample and supply some arguments as well. For the automation purpose, developers implemented an API and an option to manage tasks using Python functions in case an advanced user would like to create his own scripts.

File types for analysis are defined as packages in order to preserve framework's modularity. Thus, it is possible to create user packages and extend file types analysis compatibility. Upon submitting a sample for analysis, a user

[24] For description see section 4.1.1

is advised to pass a parameter to the submission query in order to identify the package and make Cuckoo adjust monitoring and analysis settings properly. Nonetheless, Cuckoo is able to determine which package to use, but this mechanism is not necessarily reliable. Most common file types have already been created by developers and Cuckoo is ready to analyze these packages: [8]

- *applet, jar* – analyzing Java applets and Java JAR files
- *bin* – analyzing binary data like shellcode
- *dll* – analyzing dynamically linked libraries
- *doc, xls* – analyzing Microsoft Word and Excel documents
- *exe* – analyzing Windows executables; in case of an installer, Cuckoo is capable of detecting the win32 buttons of the installation process and mimic user's mouse movements and install provided software package; this is also useable when an executable is retrieved from a provided URL
- *html, ie* – analyzing Internet Explorer's behavior while opening a supplied HTML file, URL respectively
- *pdf* – analyzing PDF documents
- *ps1, vbs* – analyzing PowerShell language, VisualBasic scripts
- *zip* – analyzing zip archives (accepts password-protected archives as well)

Every package has some options that can be switched on in order to alter the nature of the analysis done using the package.

Results of conducted analyses are stored in raw format as well as parsed by reporting modules. Moreover, Cuckoo is possible to compare results against pre-defined signatures and mark analyzed samples for better results interpretation and orientation. Signatures can be created by users and currently there is a set of user-created signatures available for download from Cuckoo's community space; a script for downloading these signatures is a part of the framework. As a full report several files are created: an overall debug log file of actions that took place during execution, a network dump file (if enabled), a full memory dump of the analysis machine (if enabled), all files that appeared in Cuckoo during the analysis, raw logs, reports (as defined in configuration) and screenshots. Cuckoo is able to create report files in JSON format, HTML, MAEC and export to MongoDB, which is utilized to display results using a webpage.

3.2.4 HoneySpider Network 2.0

Honeyspider Network 2.0 is a joint project of Polish and Dutch cyber-security centers. The team puts their efforts to build a highly-scalable system capable of crawling the web and monitoring systems, which are used to visit webpages, to spot exploits and uncover malware websites. The main focus is intensified on web browsers as a vulnerable user application and a drive-by download attack vector. However, the framework is tailored with an intention to possess abilities to analyze different file types such as .pdf, .exe, .swf, .doc.

The framework is built on modularity principle that combines functionality of low-interaction as well as possibility to add functionality of high-interaction client honeypots and is illustrated in the figure 3.6. As we can see, there is a central controlling unit that makes use of attached modules that are utilized for the actual analysis.

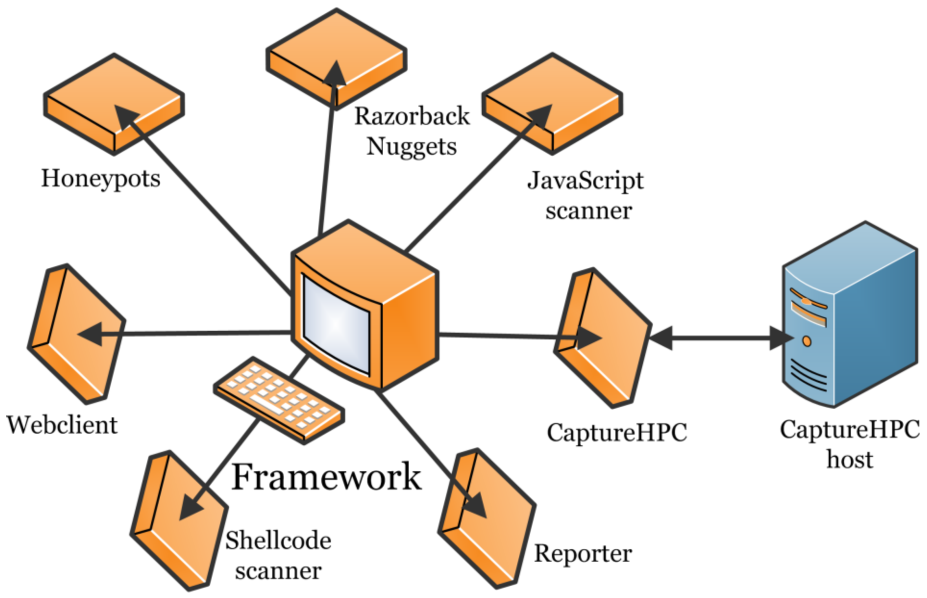


Figure 3.6 HSN 2.0 architecture

The overall operation of the framework is controlled by so-called services, which are the components of the framework that accept some input, process it and create an output for subsequent service, in case the service's nature is the creation of any output. The pattern of a single input stream, called *job*, is defined by a *workflow*, i.e. sort of an XML-structured document. An example workflow is shown in the listing 5. The job is divided into smaller tasks that are performed on the objects, which is a set of attributes. Services can add attributes to existing objects or create new objects. [12]

```xml
<?xml version="1.0"?>
<workflow>
        <description>
        VirusTotal analysis.
        </description>
  <process id="main">
    <service name="feeder-list" id="feeder">
      <parameter name="uri">file.txt</parameter>
      <parameter name="domain_info">true</parameter>
      <output process="process_url"/>
    </service>
  </process>
  <process id="process_url">
    <service name="webclient" id="webclient0" ignore_errors="DEFUNCT">
      <parameter name="link_click_policy">0</parameter>
      <parameter name="link_limit">2</parameter>
      <parameter name="redirect_limit">20</parameter>
      <parameter name="save_html">true</parameter>
      <parameter name="save_images">false</parameter>
      <parameter name="save_objects">false</parameter>
      <parameter name="save_multimedia">false</parameter>
      <parameter name="save_others">false</parameter>
      <output process="report"/>
    </service>
    <service name="reporter" id="reporter1">
      <parameter name="serviceName">webclient</parameter>
      <parameter name="template">webclient.jsont</parameter>
    </service>
    <service name="reporter" id="reporter2">
      <parameter name="serviceName"/>
      <parameter name="template">url.jsont</parameter>
    </service>
    <service name="rb-virustotal" id="virus1"/>
  </process>
  <process id="report">
    <service name="reporter" id="reporter3">
      <parameter name="serviceName">webclient</parameter>
      <parameter name="template">webclient.jsont</parameter>
    </service>
    <service name="reporter" id="reporter7">
      <parameter name="serviceName"/>
      <parameter name="template">url.jsont</parameter>
    </service>
    <service name="rb-virustotal" id="virus1"/>
  </process>
</workflow>
```

Listing 5 Sample HSN2.0 workflow

There are several tags used in the workflow that define the particular part of the processing stream. A bit more detailed description is:

Description. An explanation of the actual workflow shown in the web interface.

Process. Delimits a possible way of objects' flow and is used to pass objects to a specific flow. A process is partitioned into specific services.

Service. Determines which service treats an object. Basic approach is that information added to objects by a service is subsequently used by another service. The service uses additional tags to control the flow – *parameter* and *output*. The complete list of supported parameters can be found in *Data Contract* document [16]. The output tag redirects newly created objects to a particular process.

Conditional. Tag supports the flow control by adding a conditional expression *expr* which needs to be met by any object in order to proceed to the service defined inside of the tag.

As mentioned before, the framework consists of various services that are responsible for processing data in a sense of management or evaluation. Full list of supported services can be found in *Data Contract* document [16]. Nonetheless, we mention some of the essential services.

File Feeder service. This service takes a text file containing a list of URLs (single URL per line) as an input and creates a separate object for every line as an output for the subsequent process. [11]

Web Client service. Webclient is a crawler service, i.e. it emulates behavior that pretends to be a visit of an ordinary user. It is capable of downloading resources embedded in a webpage such as HTML, JavaScript, images, documents, or clicking on hyperlinks. For every downloaded file a new object is created, hence it can be further analyzed by another suitable service. The configuration possibilities are extensive; therefore we are able to adjust the behavior accordingly to a desired scenario. JavaScript present in any webpage is responsible for a part of dynamic content that originates after a script execution. The functionality of webclient service allows intercepting such scripts before they are executed by interpreting engine. Moreover, arguments of *eval*[25] call can be saved at runtime in order to observe de-obfuscated subject of scripts. This approach helps to create suitable data

[25] Eval function interprets contents of the script and executes them with the priviliges of the user who initiated the call; improper use may create security holes that can be abused by code injection atacks

to process by analyzers. In the previous description it was mentioned that the service is capable of creating new objects. These objects are of two different classes:

- objects that originated from links and redirects that turned up after initial visit, which are of type *url* due to the relation to succeeding webpages,
- objects that are created for different files that were downloaded, which are of type *file* that is supported by MIME[26] type specification of the content as determined by service's internal mechanism, not simply copied the classification from the visited website.

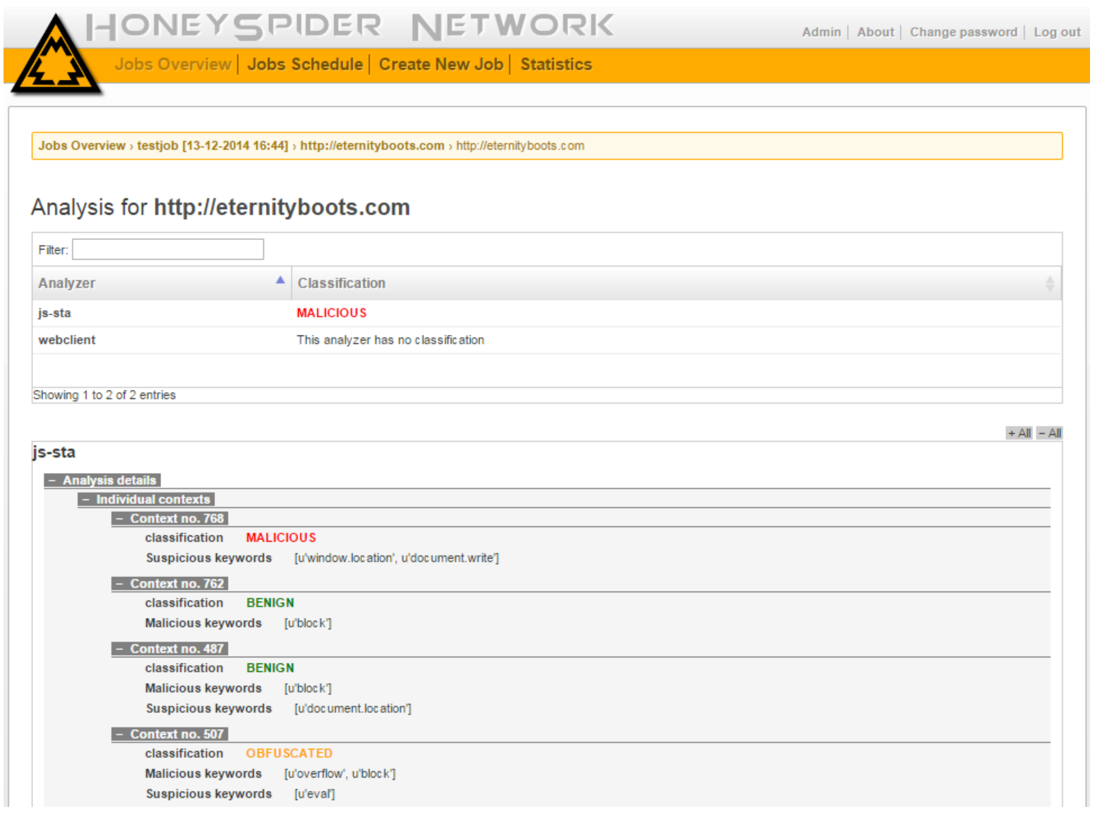
Due to the ability to push attributes from parent to child objects, a certain combination of settings makes it possible to effectively emulate a single browsing session, when in fact the processing is carried out by distinct instances of webclient service. [16]

Reporter service. Reporter is responsible for saving data in *CouchDB* database behind the framework. Information about the object is stored as a JSON document based on a pre-defined JSON template. The service must be present in every section where any data about processed objects needs to be recorded. This service does not generate any new objects. [11]

JavaScript Analyzer service. The service is capable of analyzing JavaScript source code without the need of code execution. Analyses are conducted upon contexts of JavaScript code. The service makes use of *Weka Toolkit*[27] and pattern matching techniques. The service examines contexts of code in order to find *suspicious* or *malicious* keywords inside of the code. It is possible to extend the list of such keywords by supplying parameters to the service inside of the utilized workflow. Furthermore, it is possible to provide a *whitelist* of context hashes that will be omitted from suspicious or malicious classification. Service adds some attributes to processed objects, for example the final classification of harmfulness. [14] Figure 3.7 shows malicious classification by JavaScript analyzer of a visited web link.

[26] standardization of formatting non-ASCII files in the Internet in order to provide files transfer regardless of the operating system in use; http://tools.ietf.org/html/rfc2046

[27] http://www.cs.waikato.ac.nz/ml/weka/


HONEYSPIDER NETWORK
Admin | About | Change password | Log out
Jobs Overview | Jobs Schedule | Create New Job | Statistics

Jobs Overview › testjob [13-12-2014 16:44] › http://eternityboots.com › http://eternityboots.com

Analysis for **http://eternityboots.com**

Filter:

Analyzer	Classification
js-sta	MALICIOUS
webclient	This analyzer has no classification

Showing 1 to 2 of 2 entries

+ All − All

js-sta

− Analysis details

− Individual contexts

− Context no. 768

classification	MALICIOUS
Suspicious keywords	[u'window.location', u'document.write']

− Context no. 762

classification	BENIGN
Malicious keywords	[u'block']

− Context no. 487

classification	BENIGN
Malicious keywords	[u'block']
Suspicious keywords	[u'document.location']

− Context no. 507

classification	OBFUSCATED
Malicious keywords	[u'overflow', u'block']
Suspicious keywords	[u'eval']

Figure 3.7 JavaScript analyzer malicious classification

Shellcode scanners. Framework can make use of two different components in order to detect shellcode injected by malware by executing the binary content. One uses *scizzle* package which is no longer available for free and the other uses *scdbg* [28] application which can be downloaded from the referenced website.

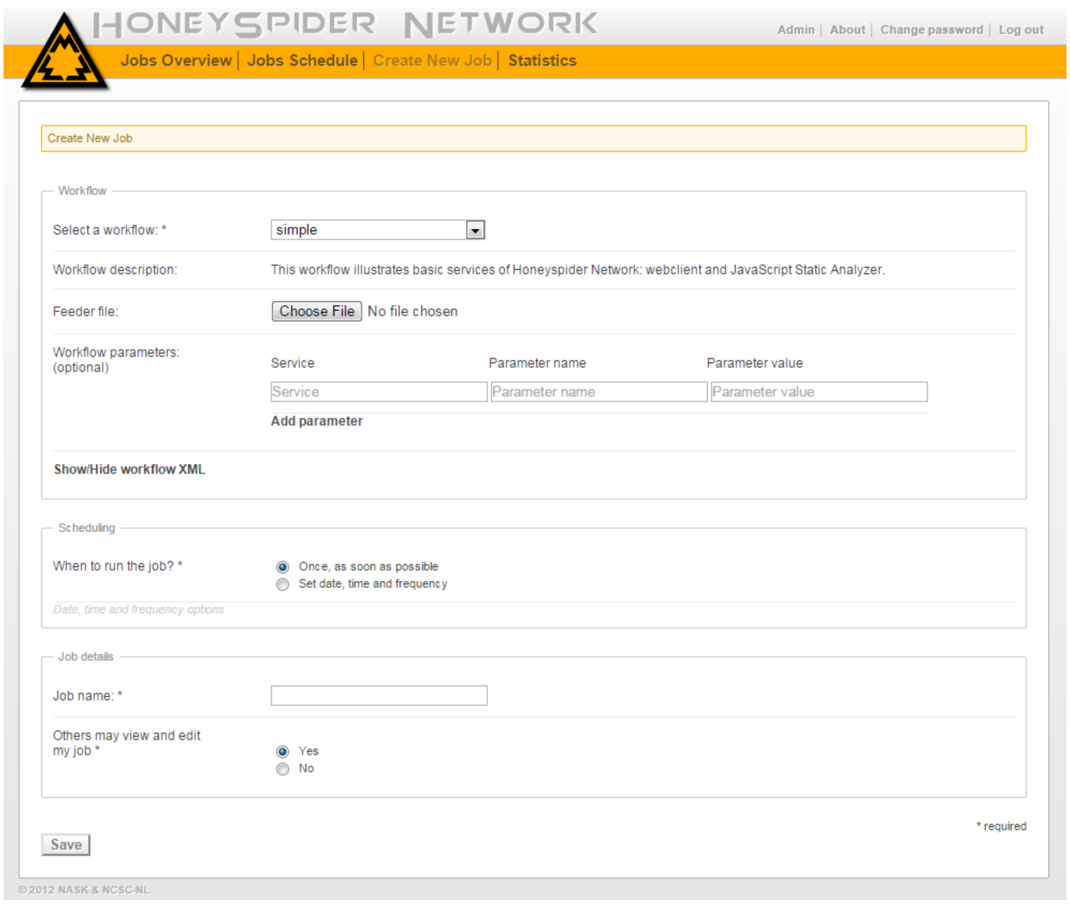
Razorback nuggets. The component employs an open-source collection of utilities that are meant to ease data processing in order to detect various events. The HSN 2.0 framework utilizes 6 nuggets out of the whole set. These are capable of - scanning .swf and .pdf files, MS Office files, passing the files for scan using ClamAV antivirus engine, comparing the MD5 hash against VirusTotal database, extracting files from archives for further processing. However, the Razorback framework has more utilities to offer; more specific description can be found in the referenced source. [25]

Honeyclient services. The framework is prepared to accept and cooperate with Thug and/or Cuckoo honeyclients during malware classification. A user needs to have these honeypots installed and configured, in hand with respective

[28] http://sandsprite.com/blogs/index.php?uid=7&pid=152

package inside of the HSN 2.0 framework to gain the possibility to utilize them as framework's services, which only passes the input for processing to these honeypots.

Capture-HPC service. This service utilizes slightly modified implementation of CaptureHPC high-interaction honeyclient that was described in section 3.2.1. In order to utilize this service, a user needs to add a dedicated computer which will run this service and configure the framework accordingly to interconnect both machines to be able to collect results effectively and interpret them using the web interface provided. The service needs to be added to the workflow respectively, so the analysis process can utilize the functionality of the high-interaction honeypot solution.


HONEYSPIDER NETWORK
Admin | About | Change password | Log out
Jobs Overview | Jobs Schedule | Create New Job | Statistics
Choose File
Add parameter
Show/Hide workflow XML
Save
© 2012 NASK & NCSC-NL

Create New Job

Workflow

Select a workflow: *	simple
Workflow description:	This workflow illustrates basic services of Honeyspider Network: webclient and JavaScript Static Analyzer.
Feeder file:	No file chosen
Workflow parameters: (optional)	Service / Parameter name / Parameter value

Scheduling

When to run the job? * [✓ Once, as soon as possible] Set date, time and frequency

Date, time and frequency options

Job details

Job name: *

Others may view and edit my job * [✓ Yes] No

* required

Figure 3.8 Web interface submission utility

The installation and configuration process is described on the project's webpage[29]. Configuration of Capture-HPC component is a bit more demanding, but a link to a website with detailed description is to be found on the webpage as well.

There are two possible ways how to operate HSN 2.0. One is a command-line interface that allows user to submit jobs or query for jobs details, manage workflows. The alternative way is to use web interface which is depicted in the figure 3.8. The latter way gives a user ways to submit jobs, schedule jobs or view results of analyses.

There is a pre-made virtual appliance available for download from the project's website. This virtual machine image was pre-built by developers and is configured accordingly so the user is ready to utilize the framework right after download.

3.3 Comparing honeyclients

In previous subchapters we have introduced several selected honeypots. Firstly we described HoneyC, Thug and Yalih that operate on low-interaction principle and utilize the emulation of services to detect malicious web resources. Subsequently, there are high-interaction honeyclients, namely CaptupeHPC, Strider HoneyMonkey, Cuckoo Sandbox and Honeyspider Network 2.0. Every implementation has some advantages as well as drawbacks. The comparison is summarized in the table below where plus (+) and minus (-) signs are used for classification. Number of signs means the overall assessment regarding the respective criterion. For example, Cuckoo Sandbox collects the broadest set of data regarding the analysis and therefore the *amount of data* criterion has +++ score. On the contrary, HoneyC project is abandoned for a long time and for that reason *status* criterion has --- score.

[29] http://www.honeyspider.org/Installation.html

		ease of use	threats	stability	speed	amount of data	status
low interaction	**HoneyC**	-	---	-	+	--	---
	Thug	+	+	+	++	++	+++
	Yalih	++	+	-	+++	+	++
high interaction	**Capture HPC**	--	++	-	--	++	---
	Honey Monkey	n/a	++	+++	-	+++	n/a
	Cuckoo	-	+++	++	--	+++	+++
	Honey spider	---	++	---	--	+	-

Table 3.1 Comparison of selected client honeypots

Previous comments on all of chosen honeypots in their respective subchapters give more information grounding the evaluation provided by this table. HoneyC is one of the first honeypots and now is long outdated, therefore it cannot be measured with most up-to-date solutions. Thug proved to be a good current low-interaction honeypot, although there is still work to do. Development of Yalih is somehow stalled, due to the fact that the honeypot's repository was not updated since the first publication. Yalih is the fastest of reviewed solution which was able to scan roughly 8000 links in several hours, but did not produce that much data for further analysis. CaptureHPC is similar to HoneyC as for the development status, yet this solution can still be used. Thanks to the principle it uses, where there is no need for signatures, but a strong skill of the operator who is responsible to observe gathered information and evaluate its harmfulness. HoneyMonkey project was referred to give a notice that also corporate sphere is engaging in this area of security. HoneySpider Network started as a potentially strong project with big ideas, but unfortunately it looks that developers have postponed it at the moment. The scalability was robust, which reflected on lower stability and slight troubles with operation configuration. Cuckoo is the most up-to-date project that is still under development and people responsible are continuously contributing to the solution in order to bring new features and performance improvements. Moreover, the community of enthusiasts around Cuckoo is growing bigger and people start to take into account the need for such sort of research in order to bring security to a higher level. The speed of Cuckoo is significantly slower compared to Thug, for example, where scanning of several hundreds of samples/links may take up to a whole day (this also depends on the time-out settings of the setup). On the other hand, the amount of collected information is large-scaled and gives the researcher required data to dig in and make proved conclusions. Cuckoo Sandbox was chosen to conduct an experiment which is a part of the thesis and is described in the fifth chapter.

4 CUCKOO SANDBOX

As mentioned in the previous chapter, Cuckoo Sandbox is an open-source project that is still being actively developed and improved by computer-security research enthusiasts. It has a growing community and a number of people who are contributing to the project and thus are extending the honeypot's functionality with new modules. This honeypot was chosen for the sake of the experiment in this thesis for four reasons, i.e. functionality, stability, user-friendly interaction and development status.

4.1 Detailed description

The main framework's architecture was outlined in section 3.2.3. We will utilize this principle in the experiment. In the present, some members from community are working to make it possible to utilize Cuckoo and deploy several server machines controlled centrally and thus multiply the operational set-up to achieve higher performances. However, in the experiment we will stay with the conventional architecture and deploy a solo server machine to control analyses on three virtual machine clients.

4.1.1 Architecture & modularity

Cuckoo Sandbox is organized into 6 main elements. Every element is configurable and is responsible for respective part of operations. The sandbox is divided into:

Cuckoo – general behavioral configuration; for example, defines which virtualization software is used, bears information about result-server address, limits for number of processing threats and processing time-outs

Auxiliary – subsidiary modules that are run synchronously with malware analysis; at the moment, by default only *tcpdump* is available

Machinery – defines aspects for selected virtualization software; list of details regarding used client machines

Memory – settings for *Volatility* module's plug-ins that parse and analyze memory dumps saved during malware execution process

Processing – switching on/off of processing modules that dissolve raw data from malware execution; e.g. behavioral, static, network analyses, or dropped files

Reporting – switching on/off of modules responsible for human-readable reports generation; Json, HTML, Maec, MongoDB

Cuckoo has the ability to function with VirtualBox, VMware, ESx and KVM virtual emulation software. The choice is completely up to the administrator. Moreover, the configuration files are easily editable to append functionality of user-created modules and support the settings or switching of such modules.

There are supplementary packages worth mentioning that bring additional functionality to overall performance of Cuckoo.

tcpdump[30]**.** Linux distribution of network packet analyzer with ability to dump network communication and store it in .pcap format; wide range of various information can be extracted from dumps; we will use this package in the Cuckoo setup in order to monitor network activity of malware.

volatility[31]**.** Open-source bundle for forensic analysis of RAM memory; it can analyze both Windows and Linux memory dumps and has an extensive set of plugins that denote possibilities of the package; the memory analysis is rather complex and demanding process and is a topic on its own, thus it will not be extensively covered in the following experiment.

yara[32]**.** Tool with the goal to unify malware classification through the definition of common patterns that are followed by malware families and creation of signatures based on these unveiled rules; formulation of rules is out of scope of this thesis, but we will use some signatures that can be downloaded from Cuckoo's community.

[30] http://www.tcpdump.org/
[31] https://github.com/volatilityfoundation/volatility
[32] http://plusvic.github.io/yara/

zer0m0n[33]**.** It is a supplementary driver which is able to perform a kernel-level analysis of malware execution, thus strengthening the detection abilities of Cuckoo in case the malware is smart enough to detect the VM mockery.

vmcloak[34]**.** Software package from one of Cuckoo's developers, which value lies in automation of the process of VM generation; this way it helps researcher to save significant amount of time that is needed to set up guest VMs and makes large deployments much easier with the help of wide configuration possibilities it has; however, the setup of our environment is not big enough for this tool to be significantly useful.

4.1.2 Starting an analysis

There are five ways how a user can start the actual analysis through Cuckoo. The most straightforward is to utilize the *Submission utility* in command line environment, supply the resource for scan and append some arbitrary parameters in order to specify the actual sample or the nature of analysis, e.g. forced time-out interval, package type, machine selection, operating system time of the guest machine, memory dump creation (if disabled by default). Following is an option to launch analysis using *REST API* which comes useful when a user wants to automate the submission process, e.g. in order to set-up a webpage where another user could scan his samples. The service starts listening on a given port of an IP address and forwards tasks to Cuckoo framework in terms of creating and extracting acquired information, as well as reporting the current state and setup of the whole framework. This functionality can be combined with a simple script for example, which will help a user to avoid one by one submission of resources to scan. The possibility to submit a sample for analysis is also present in both web interfaces that can be used with Cuckoo (more information in section 4.1.4). Lastly, developers implemented a possibility for Cuckoo to integrate with SQL databases such as MySQL, PostregSQL to maintain the stream of samples using Python functions that can be, for example, embedded into scripts.

33 https://github.com/conix-security/zer0m0n
34 http://vmcloak.org/

4.1.3 An analysis

The type of analysis is defined by analysis package that is selected. To a certain extent, Cuckoo is capable of determining the package type in case this information is not provided by user. However, it is better to supply this analysis parameter, especially when the sample is a file, not a URL. Packages that are available in Cuckoo have been described in section 3.2.4. The project's documentation also contains some notes regarding the creation of user-defined packages to supplement the efforts of enthusiasts. In our experiment set-up we modified the *ie.py* package that is suitable for examination of Internet Explorer's behavior upon visiting a webpage. We added path to Mozilla Firefox executable as well as Google Chrome in order to gain ability to open desired webpages in these web browsers and monitor events. Few lines were added into the file in *cuckoo/analyzer/windows/modules/packages/* named *ie.py*:

```
#Mozilla Firefox
iexplore = os.path.join(os.getenv("ProgramFiles"), "Mozilla Firefox", "firefox.exe")
#Google Chrome
iexplore = os.path.join(os.getenv("USERPROFILE"), "Local Settings", "Application
Data", "Google", "Chrome", "Application", "chrome.exe")
```

In a situation when a visited web location presents some downloadable content, Cuckoo is able to click the dialogue windows and submit the file for analysis. What is more, in case of an executable Cuckoo is also able to proceed with the installation and actually install the downloaded software package and monitor what is happening during the installation process. This functionality is encoded in module called *human.py*.

The actual analysis is supported by *cuckoomon.dll* which is a dynamic link library file that is injected into Windows guest system before every analysis begins. This file is responsible for the Windows-side monitoring and reports findings and interceptions to the Cuckoo host system.

4.1.4 Output of an analysis

The analysis process creates an amount of files depending on the actual nature of the malware sample submitted and the configuration set for the processing of gathered data. Files are stored in a dedicated folder, where a new subfolder is created for every single analysis. The structure is illustrated in the figure 4.1.

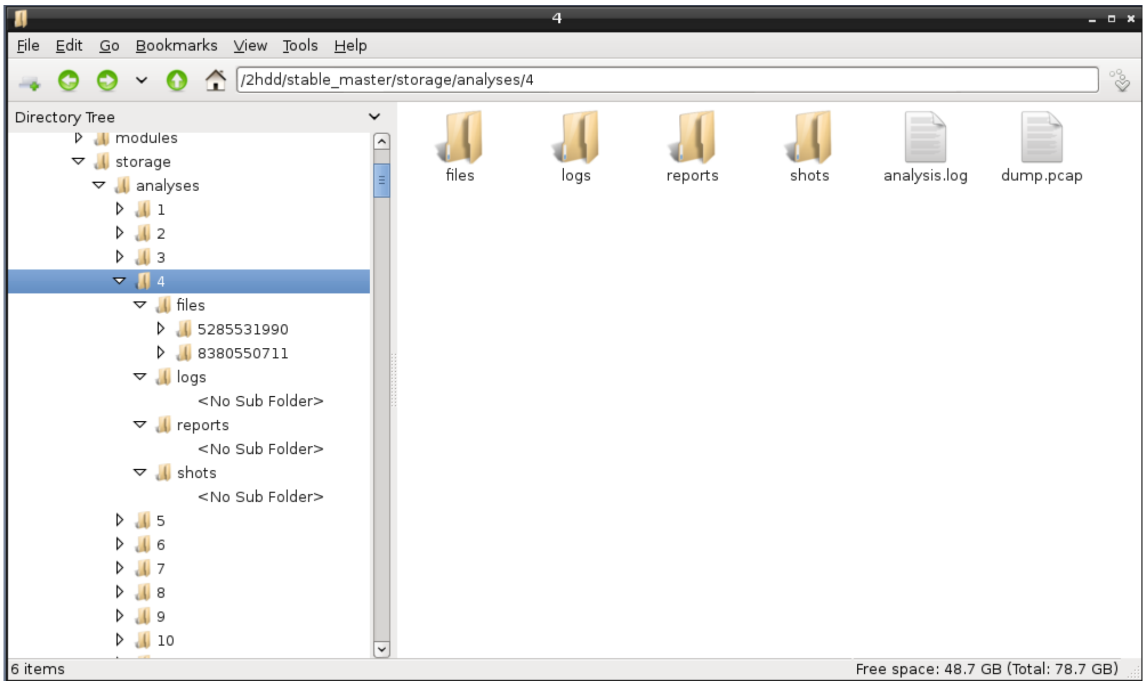


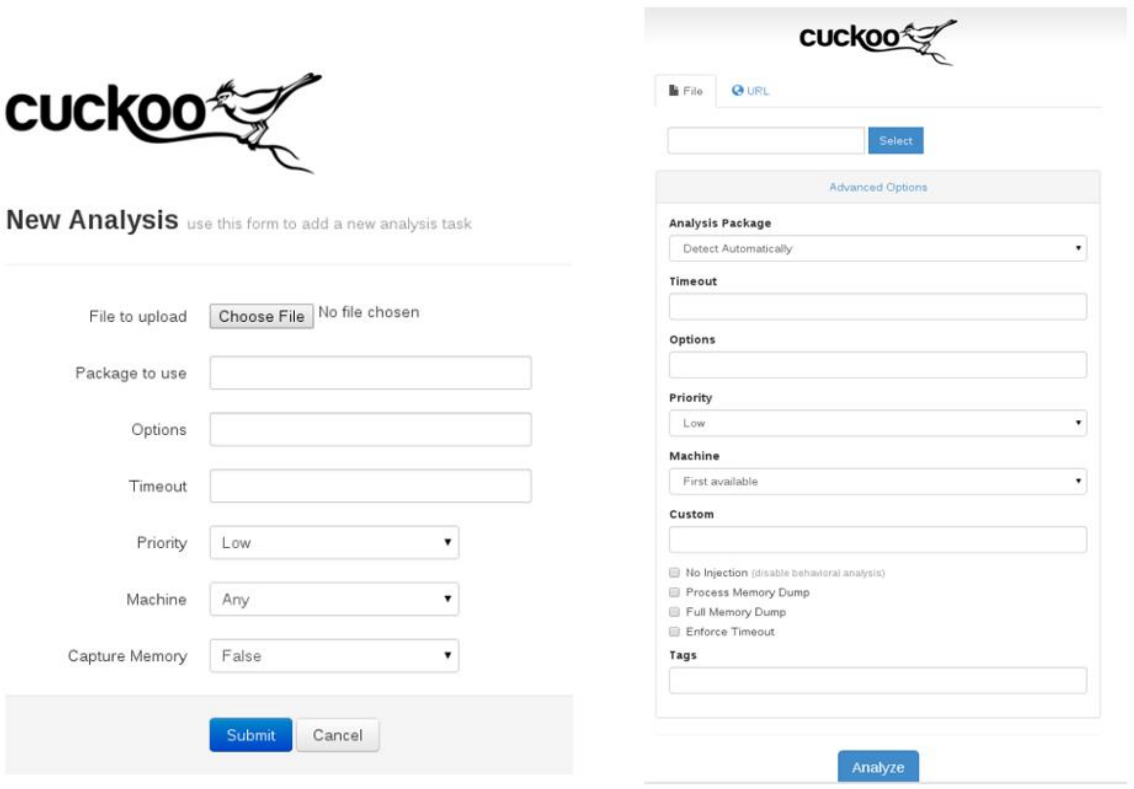
Figure 4.1 Results folder tree

File named *analysis.log* is a general debug log file that bears information about events such as process spawning, file creation, or errors that might have occurred during the analysis run. A file named *dump.pcap* stores information collected by the network sniffer and bears records about network communication, in case the network analysis is enabled in overall configuration. If created, the *memory.dmp* file is a full memory dump of the execution machine, which can be deeply examined for events that took place while the sample was executed. Inside of the *files* directory, there are files which were collected by Cuckoo that were processed, created or appeared on the guest machine in any other way, e.g. JavaScript, cookies. The subfolder *logs* stores raw logs generated by Cuckoo in .bson format. On the other hand, *reports* folder holds reports generated from the raw data and the contents of the folder depend on the report processors enabled in the framework's configuration. Lastly, the folder *shots* is used to save screenshots that were captured during the analysis, which are helpful to see what was happening on the screen while the sample is executed. The time delay between screenshots is, by default, set to 1 second, but can be easily altered.

4.1.5 Web front-end

Cuckoo Sandbox can be enriched with two different web interfaces that help user to display results in more fancy way and to maintain tracks of conducted analyses.

The older, simpler interface is easy to launch, with no need of installation or configuration, directly from the Cuckoo application directory under */utils* subfolder. Figure 4.2 shows the submission page of both web interfaces, with the older one located on the left side.



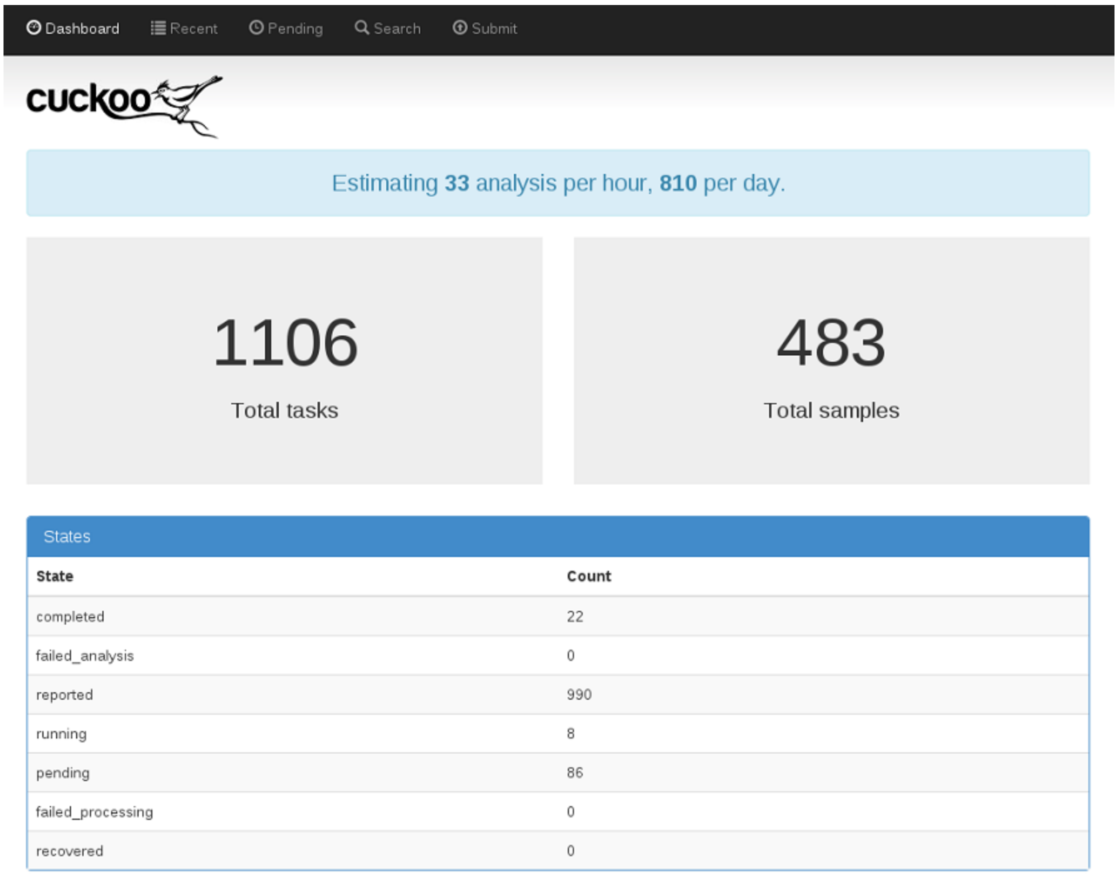
New Analysis use this form to add a new analysis task

File to upload	Choose File No file chosen
Package to use	
Options	
Timeout	
Priority	Low
Machine	Any
Capture Memory	False

Analysis Package
Detect Automatically
Timeout
Options
Priority
Low
Machine
First available
Custom
No Injection (disable behavioral analysis)
Process Memory Dump
Full Memory Dump
Enforce Timeout
Tags

Figure 4.2 Sumbission page of both interfaces

There is also a bit more complex interface available, which utilizes *Django* application and accumulates data in *MongoDB* on the background. Therefore, it is necessary to have these packages installed and configured as well



Estimating **33** analysis per hour, **810** per day.

1106
Total tasks

483
Total samples

States

State	Count
completed	22
failed_analysis	0
reported	990
running	8
pending	86
failed_processing	0
recovered	0

Figure 4.3 Homepage of new interface

as allowed within the settings of Cuckoo. The newer interface allows a user to search amongst analysis results. Moreover, in assistance of WSGI[35] interface this web interface can be deployed on web servers like *Apache*, *Unicorn*, or *Nginx* in order to give users access to the platform via web. The configuration of infrastructure needed for this deployment is out of scope of this thesis. The homepage of new interface is depicted in the figure 4.3.

[35]Python‘s Web Server Gateway Interface to interconnect web server and web application or framework; https://www.python.org/dev/peps/pep-0333/

4.2 Deploying the honeypot

The following subchapter explains the actual process of installation of underlying systems needed to set up the experimental framework.

4.2.1 Host operating system and software

The operating system for the *Host* machine is chosen to be Debian 7.7.0 (debian-7.7.0-amd64-lxde), the latest stable release with LXDE desktop environment. The Cuckoo Sandbox is intended to have the best performance on GNU/Linux system such as Ubuntu or Debian. [4] The installation of the OS is extensively covered on the distribution's website[36]. Operating system is installed on a mainstream home PC with following parameters:

Asus P8Z77-M PRO
Intel Core i5-3470 @ 3.20 GHz
8 GB RAM DDR3
80 GB hard-drive for OS
80 GB hard-drive for results storage

After the OS is installed, there is a need to complement the functionality with additional packages. Many of packages are installed automatically by package manager *aptitude* because the necessary packages are dependent on the chosen ones. Packages installed via aptitude manager using *$sudo apt-get install* command are as follows:

python, python-pip, python-dev, python-dpkt, python-markupsafe python-magic, python-gridfs, python-sqlalchemy, python-bson, libtool, automake, autotools-dev, m4, ia32-libs, curl, tcpdump, dkms, virtualbox-4.3

An alternative python-oriented package manager holds several of used distributions that are installed using *$sudo pip install* command:

jinja2, pymongo, bottle, pefile, maec==4.0.1.0, django==1.7.1, chardet, vmcloak, pycrypto

All of the packages can be downloaded in source code from projects' websites, then compile the code and install the package manually.

[36] https://www.debian.org/index.html

Some of required packages are not available in *aptitude* or *PyPI* managers and need to be added manually. These packages are:

pydeep and dependency *ssdeep-2.12*[37]*, yara 3.1.0, volatility 2.3.1, distorm3*[38]

Finally, necessary dependencies for Cuckoo Sandbox are installed. The Cuckoo package is available for download on the project's website (in stable and development builds). It is sufficient to extract the downloaded archive and grant permissions to a (created) system user to operate with VirtualBox software; at this point Cuckoo is ready to start working.

The particular configuration of Cuckoo Sandbox used for experimenting is described later in subchapter 5.1.

4.2.2 Guest VM systems installation

Preparation of the *Guest* system, that will actually do the exploratory part of the work, requires patience due to the high time heftiness. The proper arrangement should not be underestimated, due to the fact that advanced malware is able to detect the mock-up and avoid the attack attempt. Moreover, in terms of software it is also important to install various outdated versions with vulnerabilities. In order to plausibly reproduce a genuine computer system it may be useful to store some user files in the system, e.g. pictures, photos, videos, documents and similar. Additionally, the presence of software that is not often a target of attacks and is not intended to be monitored may be complementary in order to create virtual machine impersonation. Such pre-attack scans are, in case of targets of opportunity, automated. That is thanks to the high number of attacks that cyber-criminals try to conduct. In case of targets of choice, the perpetrator is considered to be skilled sufficiently to determine a counterfeit, thus such cloaking cannot be sufficient.

The Guest operating system used in virtual machines is Microsoft Windows 7. An outdated Windows XP was considered to be a part of experimental set-up, but during the recent half a year the usage shares dropped drastically from about 30% in January 2014 to as low

37 https://github.com/kbandla/pydeep; http://ssdeep.sourceforge.net/
38 https://code.google.com/p/distorm/

as 14% November 2014. [26] This may be a consequence of end of official support for this system from Microsoft that was closed with product's lifecycle on 8.4.2014. [44] This decision was made not only because Cuckoo has the best performance and stability with this kind of OS, but mainly due to the fact that Windows is the most used operating system in the present day. Based on the data from StatCounter website, the Windows OS has more than 85% of overall market share. [34]

Windows 7

For the sake of experiment on Windows 7 platform we will use three virtual guests with identic configuration. A clean installation of Windows 7 is performed. Automatic *Windows updates* are turned off as well as *Windows Firewall* feature as advised by developers. This is due to the fact that firewall may be disrupting the communication between Cuckoo client and Cuckoo server. To ensure correct operations following software is necessary:

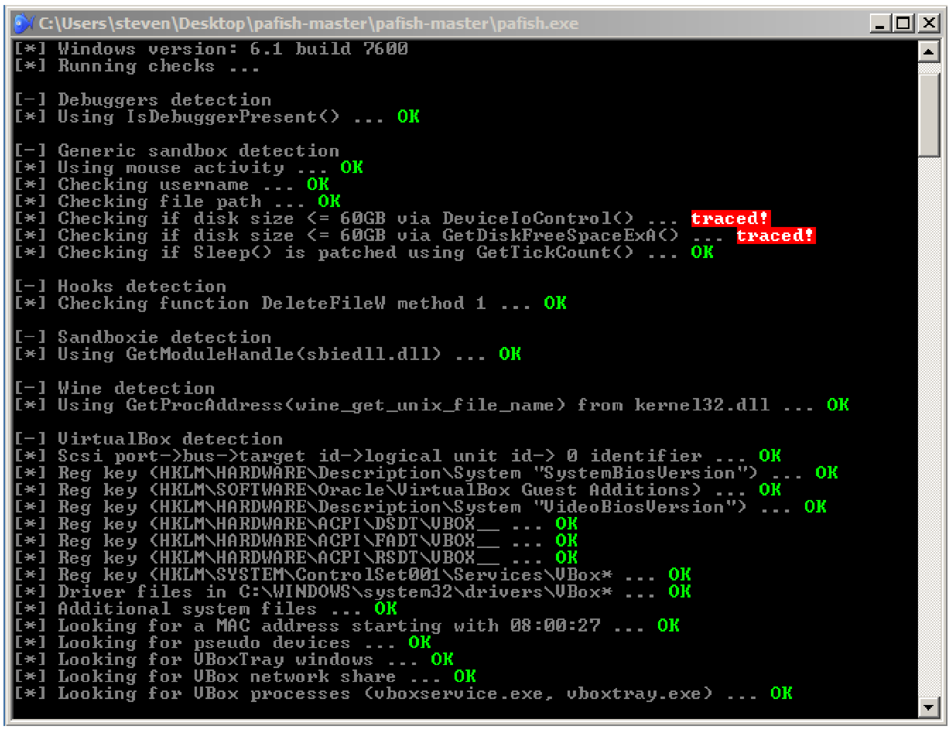
python 2.7.7, python image library 1.1.7 (PIL), internet explorer (IE) 8.0.6001.18702 (default version)

From the Cuckoo environment's point of view, there is a need to copy *agent.py* file from Cuckoo's distribution package and paste it into the guest system. It is necessary to execute this file afterwards, because it ensures the communication between host and guest stations. Moreover, user software was installed to grant the honeypot possibilities to work with user files. Installed applications are:

adobe reader 10.1.4, microsoft office 2007 (12.0.4518.1014)

We also adjusted settings of *Internet Explorer* browser and *Internet Options* of the operating system in order to lower security measures that could prevent execution of malware.

To avoid easy detection of VirtualBox emulated environment the installation of *VirtualBox Guest Additions* software package was omitted. The reason for omitting is the fact that the package adds several drivers, e.g. video, shared folders, to the system, which are useful for a basic user to ease interaction with VM. On the other hand, the presence of the package makes it easier to detect virtual environment thanks to presence of particular registry keys or processes that are created by the package. Figure 4.4 shows an output from *pafish* [39] utility. Multiple factors that reveal presence of virtual environment are successfully masked besides the actual size of hard-drive in guest systems, which is smaller than 60 GB in this case. This is due to the limitations of available physical hard-drive which is not big enough to store three virtual machines with 60 GB virtual hard-drive for each machine.


C:\Users\steven\Desktop\pafish-master\pafish-master\pafish.exe

```
[*] Windows version: 6.1 build 7600
[*] Running checks ...

[-] Debuggers detection
[*] Using IsDebuggerPresent() ... OK

[-] Generic sandbox detection
[*] Using mouse activity ... OK
[*] Checking username ... OK
[*] Checking file path ... OK
[*] Checking if disk size <= 60GB via DeviceIoControl() ... traced!
[*] Checking if disk size <= 60GB via GetDiskFreeSpaceExA() ... traced!
[*] Checking if Sleep() is patched using GetTickCount() ... OK

[-] Hooks detection
[*] Checking function DeleteFileW method 1 ... OK

[-] Sandboxie detection
[*] Using GetModuleHandle(sbiedll.dll) ... OK

[-] Wine detection
[*] Using GetProcAddress(wine_get_unix_file_name) from kernel32.dll ... OK

[-] VirtualBox detection
[*] Scsi port->bus->target id->logical unit id-> 0 identifier ... OK
[*] Reg key (HKLM\HARDWARE\Description\System "SystemBiosVersion") ... OK
[*] Reg key (HKLM\SOFTWARE\Oracle\VirtualBox Guest Additions) ... OK
[*] Reg key (HKLM\HARDWARE\Description\System "VideoBiosVersion") ... OK
[*] Reg key (HKLM\HARDWARE\ACPI\DSDT\VBOX__ ... OK
[*] Reg key (HKLM\HARDWARE\ACPI\FADT\VBOX__ ... OK
[*] Reg key (HKLM\HARDWARE\ACPI\RSDT\VBOX__ ... OK
[*] Reg key (HKLM\SYSTEM\ControlSet001\Services\VBox* ... OK
[*] Driver files in C:\WINDOWS\system32\drivers\VBox* ... OK
[*] Additional system files ... OK
[*] Looking for a MAC address starting with 08:00:27 ... OK
[*] Looking for pseudo devices ... OK
[*] Looking for VBoxTray windows ... OK
[*] Looking for VBox network share ... OK
[*] Looking for VBox processes (vboxservice.exe, vboxtray.exe) ... OK
```

Figure 4.4 Paranoid fish anti-vm detection

[39] Paranoidfish is a tool that checks several aspects within Windows system in order to detect presence of virtual environment; https://github.com/a0rtega/pafish

5 EXPERIMENT

The fifth chapter is discussing the experiment that was conducted on the configured environment which is described in the previous chapter and also showcases the actual utilization of Cuckoo Sandbox malware research tool. It illustrates several malware samples and findings that came up from analysis.

5.1 Configuration of Cuckoo instance

This subchapter shows the specific reading of configuration files in Cuckoo Sandbox as they are used in the setup for experiment. As mentioned in honeypot's description, there are 6 configuration files to adjust, yet we omit one that contains features of memory dump analysis. This area of analysis is not considered in the experiment due to the high consumption of resources such as storage space and timespan which takes to process a single analysis. Nonetheless, memory dump analysis is a powerful domain in sense of unveiling malicious activity, but it takes a skilled expert in order to announce reliable conclusions and additionally it is more useful in investigation and classification of zero-day threats. Important values of configurable aspects as well as those that were changed in comparison to pre-set values are as follows:

cuckoo.conf

```
version_check = off           #Cuckoo update check on start-up
machinery = virtualbox        #virtual environment in use
memory_dump = off             #creation of memory dumps
reschedule = off              #re-scheduling of incompletely processed tasks
default_timeout = 150 sec     #amount of seconds for the analysis, killed afterwards
critical_timeout = 300        #max interval before VM is terminated no matter what
```

auxiliary.conf

```
sniffer_enabled = yes         #creation of network communication dump
tcpdump = /path/to/package    #location of sniffer package within the host system
interface = vboxnet0          #network interface to sniff on
```

virtualbox.conf

```
mode = gui                          #VirtuaBox mode in which guests will run
machines = cuckoo1, cuckoo2, cuckoo3    #list of VMs
[cuckoo1]                           #section for specific VM
label = wse1                        #name of VM in VirtualBox
platform = windows                  #OS in VM
ip = 192.168.56.110                 #IP address of VM
snapshot = final                    #snapshot's name that should be restored on every run
(optional) tags = win7, ...         #user specified tags to better categorize large setups
[cuckoo2], [cuckoo3] sections respectively as they are of same setup as [cuckoo1]
```

processing.conf

```
analysisinfo = yes                  #information about analysis, e.g. time, machine
behavior = yes                      #behavioral info about processes running
debug = yes                         #events captured in guest VM, e.g. process spawning
dropped = yes                       #information about dropped files on guest VM
memory =no                          #memory dumps
network = yes                       #info about network communication
static = yes                        #imports into .dll files
strings = yes                       #strings embedded in examined files
targetinfo = yes                    #additional info about sample, e.g. hashes, size
virustotal = yes                    #evaluation of file discovered on VirusTotal
```

reporting.conf

```
jsondump = yes                      #creation of .json report
reporthtml = yes                    #creation of .html report
mmdef = no                          #report findings from memory dump
maec40_enabled = no                 #report creation in MAEC format
mangodb_enabled = yes               #store reports into MongoDB
```

For the sake of automation of start-up of Cuckoo's services a simple bash script was created and is sufficient to accomplish this task. After the configuration of Cuckoo is completed, we can execute the script and launch the framework so it can wait for submission of desired resources we want to analyze. Automation of submission of samples for analysis is also simply achievable with help of a bash script.

5.2 Defining the experiment

The goal of the experiment is to demonstrate that the prepared set-up and configuration is proper and works correctly giving a user data regarding performed analyses that can be investigated. Moreover, the experiment shows possibilities of Cuckoo and reveals some statistics gathered from collected data. The statistics were extracted with a help of a script written in Python language, which searched through created .json reports and collected desired data.

Resources for experiment in a form of URLs were retrieved from http://malwareurls.joxeankoret.com/normal.txt list[40] on 10.12.2014. For the sake of downloading malware, the VirusTotal's repository was utilized. With the default public access account a user does not have admission to these resources, yet it is possible to request permission. Moreover, a tiny utility called *maltrieve*[41] was also used to retrieve malicious content. It is capable of scanning malware repositories, retrieve knowledge and store it for further processing. With the correct configuration, the utility can push downloaded content directly to the running instance of Cuckoo Sandbox.

5.3 Process of the experiment

During the experiment we analyzed a total number of 2464 resources out of which 1303 were potentially malicious URLs and 1161 were various malware samples. Out of the overall amount only 19 analyses did not finish successfully, i.e. the created .json report was incomplete and findings from these reports were not accounted in the final report. This split is depicted in the figure 5.1 as well as the distribution between particular virtual machines. Cuckoo1 was responsible for 904, Cuckoo2 for 722 and Cuckoo3 did 778 analyses.

[40] The list is provided by an enthusiast malware researcher for non-commercial use and is daily updated

[41] https://github.com/technoskald/maltrieve

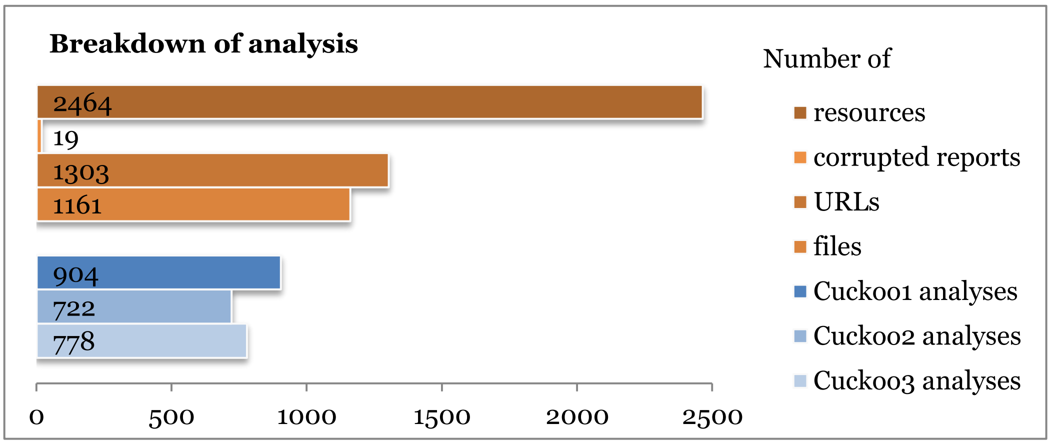


Figure 5.1 Split of resources

The breakdown of file types of analyzed samples can be seen in the figure 5.2. Based on the computed SHA256 hash of files analyzed, there were 991 unique files and 170 were duplicates of already analyzed files. The highest amount of analyses were Windows 32bit executables, which reflects the reality in a sense that .exe file format is the one which is the most popular to transmit malware over the Internet.

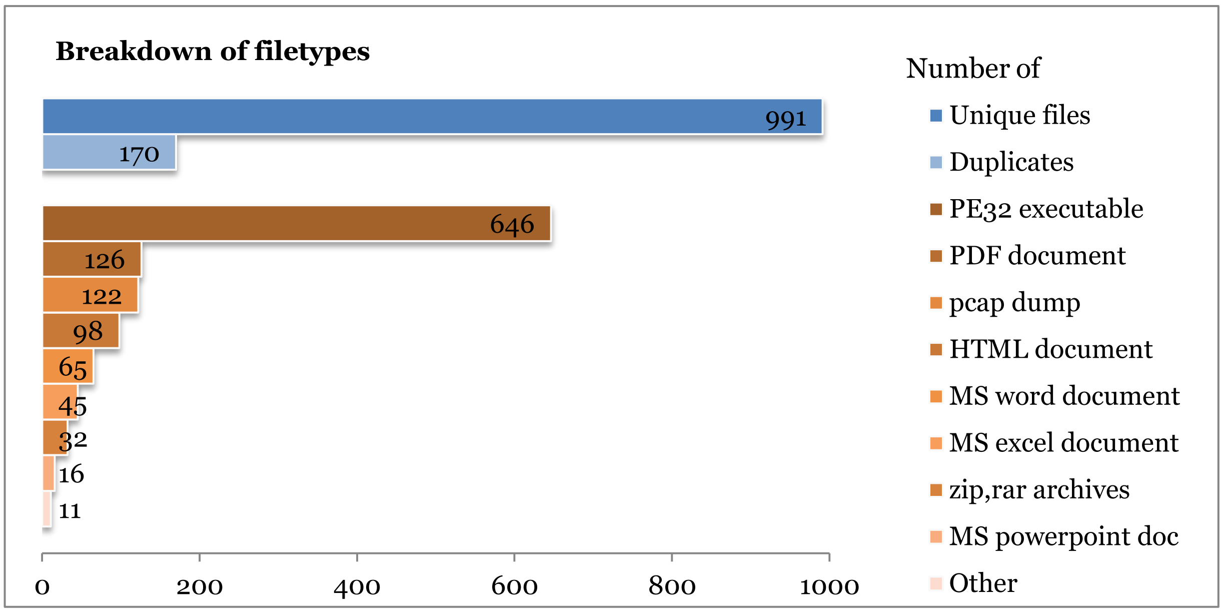


Figure 5.2 Filetypes of resources in experiment

In the end, the framework did on average 28 analyses per hour which amounts to 672 analyses per day. However, while processing URLs the framework was a bit faster processing around 42 links per hour, but on the other hand processing of files averaged around 22 files per hour.

5.4 Output

Results extracted from the experiment are presented in this subchapter. The overall process of experiment produced 116.7 GB of data in two independent runs in forms of collected files, executed binaries, raw reports and processed reports in human-readable form.

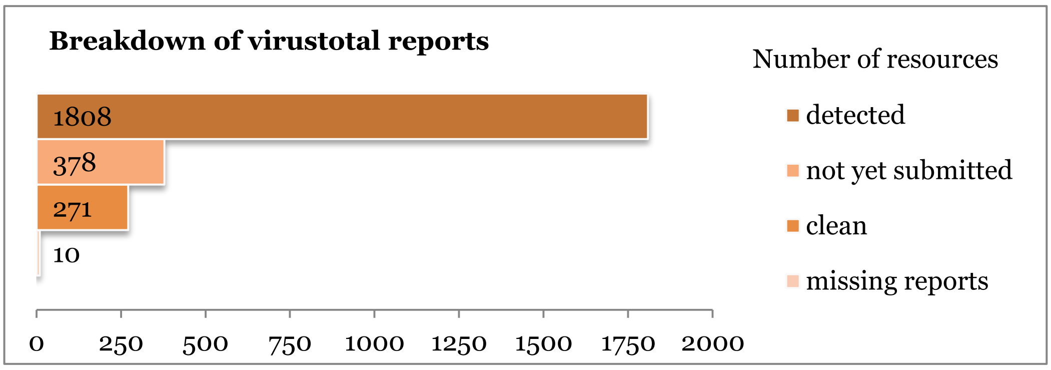


Figure 5.3 Scanned resources on VirusTotal

Thanks to the integration of VirusTotal's findings into created reports, we can assess scanned resources with more confidence. The figure 5.3 shows results compared against VitusTotal, 1808 resources had at least one positive detections by any of present antivirus engines, 378 resources were not yet submitted for scan or previously scanned at the moment of query for results, 271 resources had zero positives which makes them clean with high probability. In total, all malicious resources had 24006 positive detections which approximates for 13.28 detections per malicious resource. As previously mentioned in the framework's description, there are some signatures available for download which may mark recognized patterns from samples examination that are present in final reports. The figure 5.4 shows the split of all signatures matched during the experiment. In total, 1441 of resources performed a check of machine's system preferences; however this behavior was also observed with benign resources. More than 500 resources carried out steps that try to ensure automatic launch at Windows start-up, which may, for example, help an attacker to restore access to the computer automatically after a reboot. Almost 300 resources stole personal information stored within user's web browser application. On 17 occasions, examined malware checked for presence of anti-debug or forensic tools which are often used by malware researches

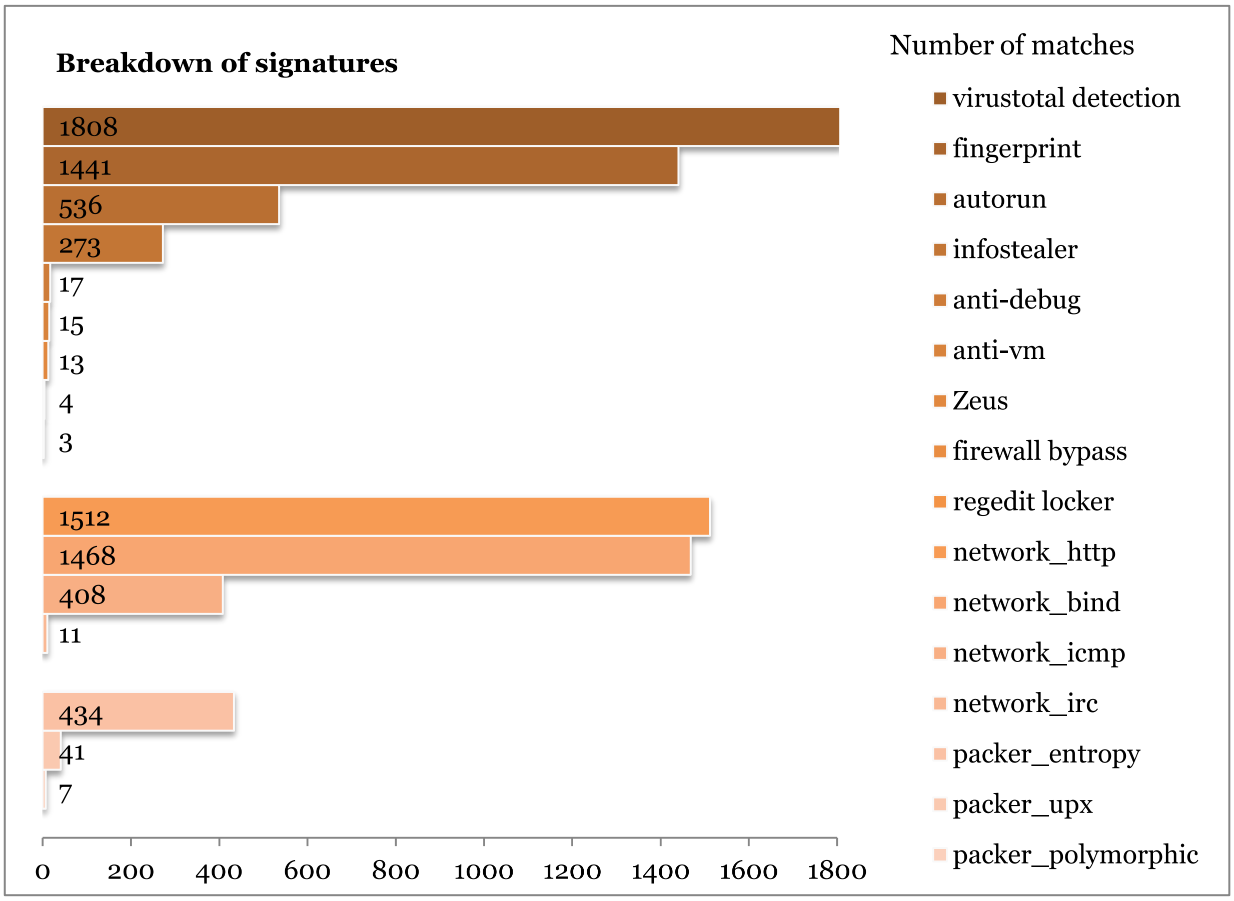


Figure 5.4 Signatures matched

for the sake of investigation. This way attackers try to avoid detections. Moreover, in 15 cases a check for virtual machine environment was encountered. The actual Zeus banking Trojan was noticed 13 times. Settings of local firewall were attempted to be changed 4 times and 3 times malware locked user's access into Windows registry keys editor. Significant amount of resources performed some network communication, specifically in 1512 cases at least one HTTP request was made, in 1468 cases the examined resource started listening on a network port in order to intercept some traffic, in 408 cases the resource sent a message via ICMP[42] protocol and in 11 cases a resource attempted to connect to IRC channel, which may sign an attempt to connect the machine into an existing botnet. In the process of experiment an overall number of 3115 unique domain addresses was contacted by resources examined. The recent phenomenon with attackers trying to mask the actual content of malicious files was also encountered in around 20% of all analyses. In 434

[42] ICMP is a network protocol used to exchange messages amongst devices regarding service state and is not typically used by user applications [23]

binaries excessive entropy was noticed to be present, which is a sign that the file may contain either compressed or encrypted data. The UPX packer [43] for executables compression was found in 41 files and 7 times the binary replicated itself in a slightly modified version within the system.

Additionally, created .json reports hold information about particular antivirus engine detections in case of scanned files. In case of potentially malicious URLs information about the presence of the URL in blacklists or similar databases can be extracted from reports. There are 57 different antivirus engines and 42 different databases or URL scanning engines present in all collected reports. Figure 5.5 shows numbers of detections of files where top 5 vendors by number of detections are present and are complemented with popular vendors amongst users.

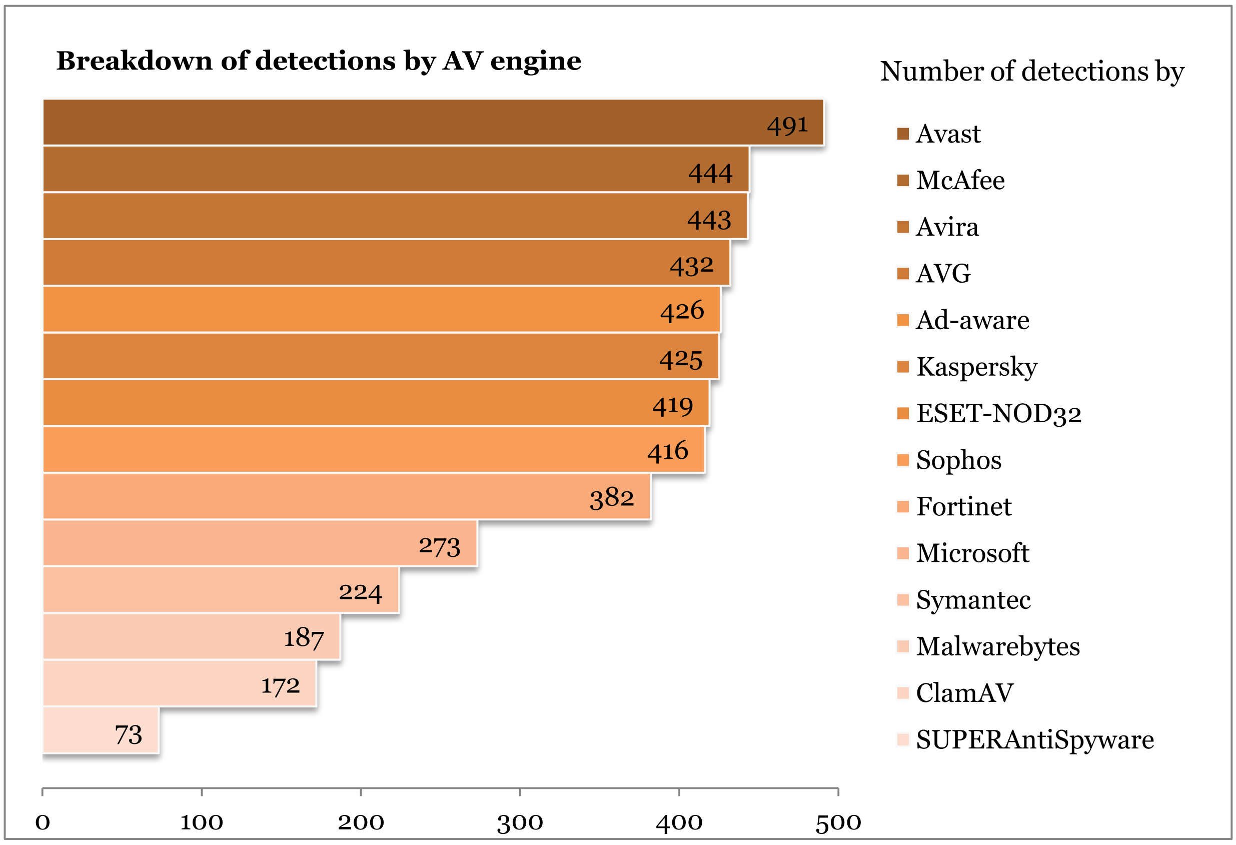


Figure 5.5 Detections of files by AV engine

[43] http://upx.sourceforge.net/

Similarly, figure 5.6 displays detections of scanned URLs as identified by scanning engines or malware blacklists.

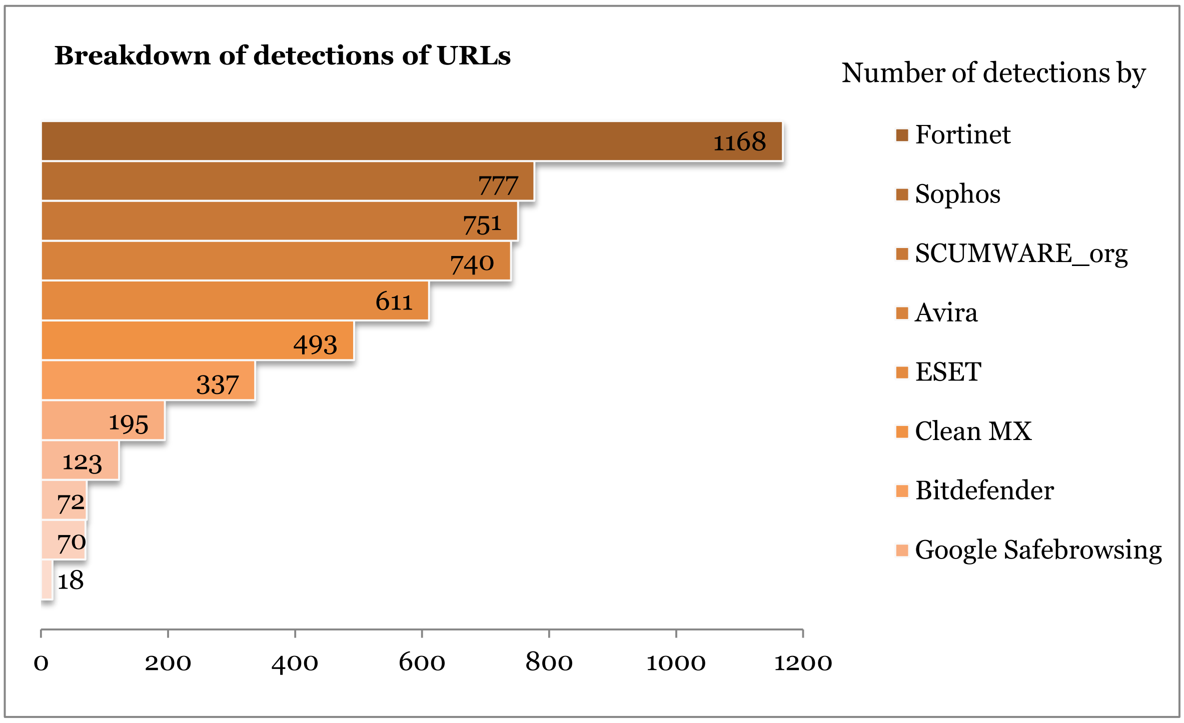
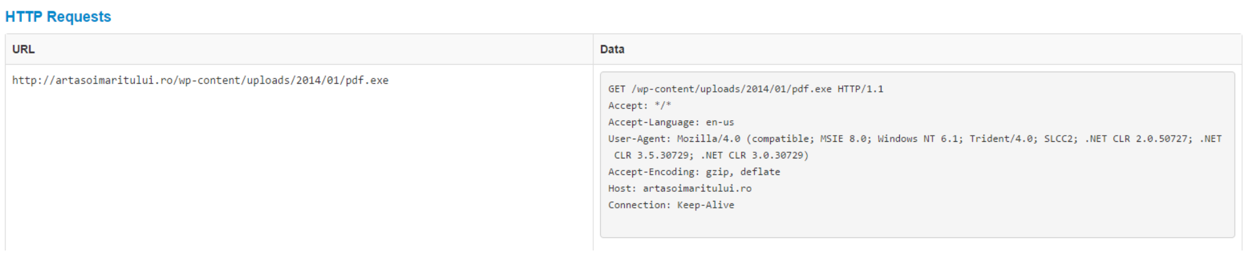


Figure 5.6 Detections of URLs

5.5 Sample Analysis

For the reason that a proper analysis in Cuckoo Sandbox should be accompanied by some evaluation of a researcher, in this subchapter we will have a closer look on a chosen threat that was encountered amongst resources that were scanned during the process of experiment.



HTTP Requests

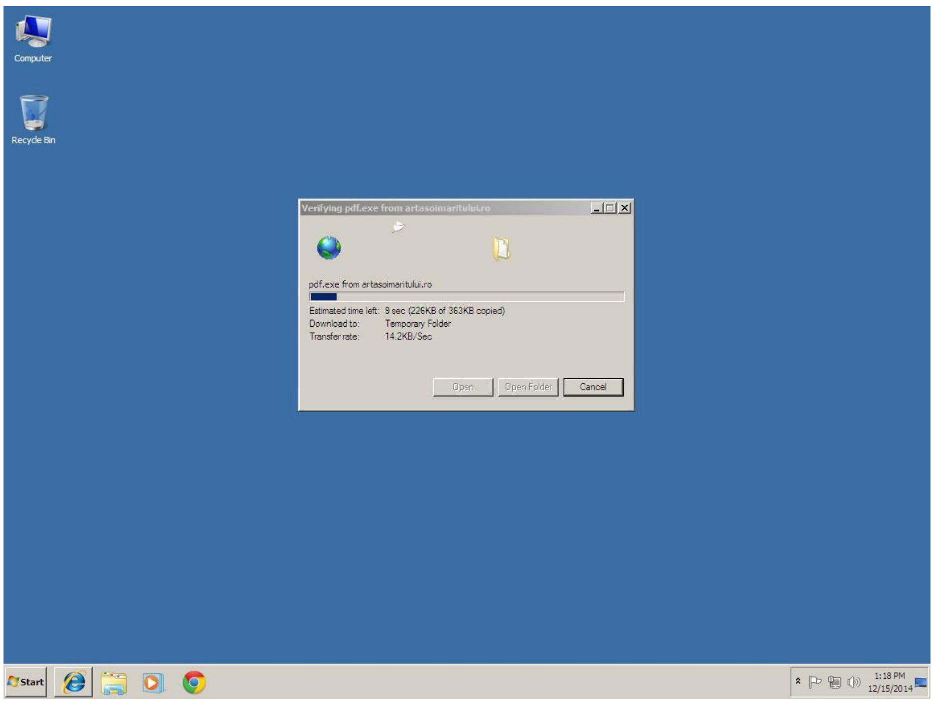
URL	Data
http://artasoimaritului.ro/wp-content/uploads/2014/01/pdf.exe	GET /wp-content/uploads/2014/01/pdf.exe HTTP/1.1 Accept: */* Accept-Language: en-us User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729) Accept-Encoding: gzip, deflate Host: artasoimaritului.ro Connection: Keep-Alive

Figure 5.7 HTTP request to remote location

A resource for download was offered during the visit to a potentially malicious URL located at:

http://artasoimaritului.ro/wp-content/uploads/2014/01/pdf.exe

Cuckoo Sandbox has distinguished this activity and proceeded with the download and execution of presented executable. The HTTP request used to contact the remote location of malicious file is illustrated in the figure 5.7.



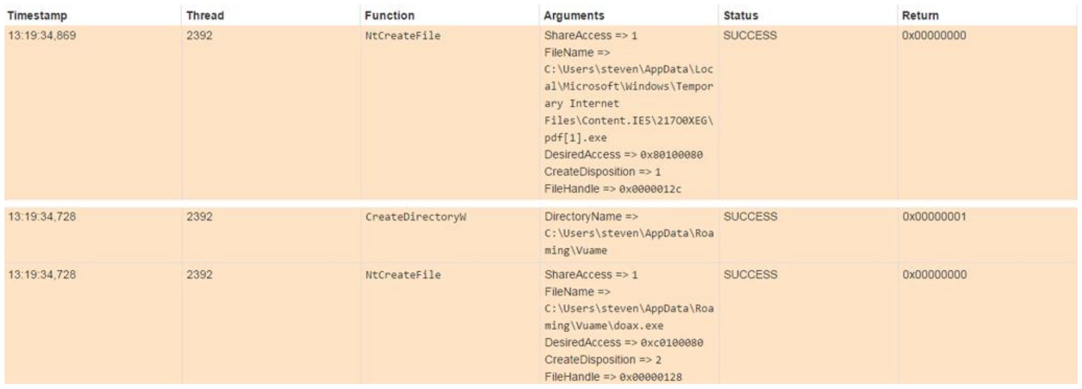
Verifying pdf.exe from artasoimaritului.ro

pdf.exe from artasoimaritului.ro

Estimated time left: 9 sec (226KB of 363KB copied)
Download to: Temporary Folder
Transfer rate: 14.2KB/Sec

Figure 5.8 Download process of offered resource

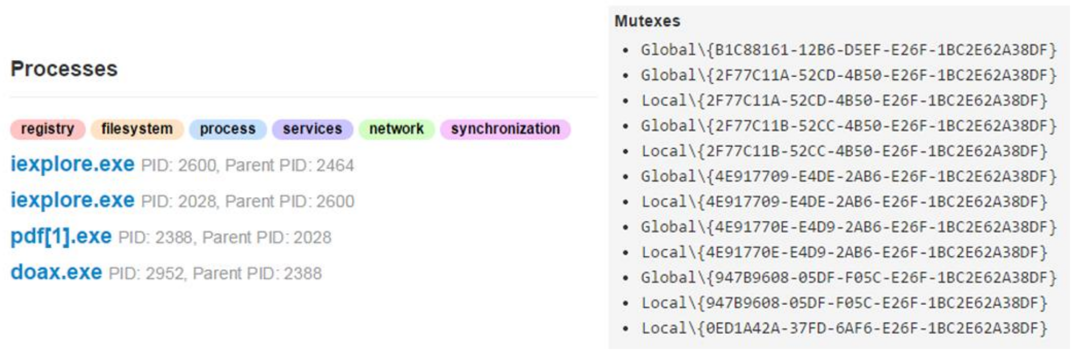
The figure 5.8 is a screenshot captured on the machine which illustrates the described activity. The link was detected by 13 different intelligence collectors



Timestamp	Thread	Function	Arguments	Status	Return
13:19:34,869	2392	NtCreateFile	ShareAccess => 1 FileName => C:\Users\steven\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\21700XEG\pdf[1].exe DesiredAccess => 0x80100080 CreateDisposition => 1 FileHandle => 0x0000012c	SUCCESS	0x00000000
13:19:34,728	2392	CreateDirectoryW	DirectoryName => C:\Users\steven\AppData\Roaming\Vuame	SUCCESS	0x00000001
13:19:34,728	2392	NtCreateFile	ShareAccess => 1 FileName => C:\Users\steven\AppData\Roaming\Vuame\doax.exe DesiredAccess => 0xc0100080 CreateDisposition => 2 FileHandle => 0x00000128	SUCCESS	0x00000000

Figure 5.9 Function calls for files creation

to be of malicious nature. After the download finished, 3 different files were stored on the local hard-drive, 2 out of which were executables which spawned new processes. The figure 5.9 shows functions calls recorded that were responsible for creation of files on the local storage. Due to the fact that the resource was instructed to be run directly after the download, the file was stored in the location for temporary Internet Explorer files. The second dropped



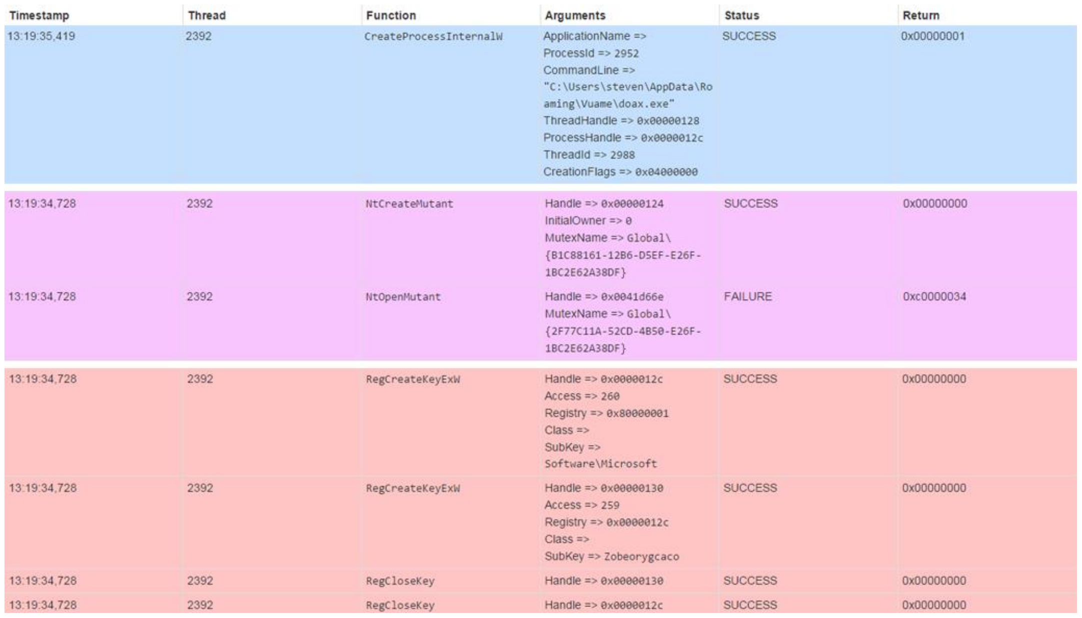
iexplore.exe PID: 2600, Parent PID: 2464
iexplore.exe PID: 2028, Parent PID: 2600
pdf[1].exe PID: 2388, Parent PID: 2028
doax.exe PID: 2952, Parent PID: 2388

Mutexes

- Global\{B1C88161-12B6-D5EF-E26F-1BC2E62A38DF}
- Global\{2F77C11A-52CD-4B50-E26F-1BC2E62A38DF}
- Local\{2F77C11A-52CD-4B50-E26F-1BC2E62A38DF}
- Global\{2F77C11B-52CC-4B50-E26F-1BC2E62A38DF}
- Local\{2F77C11B-52CC-4B50-E26F-1BC2E62A38DF}
- Global\{4E917709-E4DE-2AB6-E26F-1BC2E62A38DF}
- Local\{4E917709-E4DE-2AB6-E26F-1BC2E62A38DF}
- Global\{4E91770E-E4D9-2AB6-E26F-1BC2E62A38DF}
- Local\{4E91770E-E4D9-2AB6-E26F-1BC2E62A38DF}
- Global\{947B9608-05DF-F05C-E26F-1BC2E62A38DF}
- Local\{947B9608-05DF-F05C-E26F-1BC2E62A38DF}
- Local\{0ED1A42A-37FD-6AF6-E26F-1BC2E62A38DF}

Figure 5.10 Spawned processes and created mutexes

file was stored into the *ApplicationData* folder of the system. The figure 5.10 presents spawned processes as well as created mutexes for executables in order to ensure necessary resources of the computer's CPU. Mutex is often used by malware to help avoid double infection, or alternatively to ensure synchronization of multiple malware elements. It can be used to distinguish malicious nature of executed files thanks to the names of these mutexes which



Timestamp	Thread	Function	Arguments	Status	Return
13:19:35,419	2392	CreateProcessInternalW	ApplicationName => ProcessId => 2952 CommandLine => "C:\Users\steven\AppData\Roaming\Vuame\doax.exe" ThreadHandle => 0x00000128 ProcessHandle => 0x0000012c ThreadId => 2988 CreationFlags => 0x04000000	SUCCESS	0x00000001
13:19:34,728	2392	NtCreateMutant	Handle => 0x00000124 InitialOwner => 0 MutexName => Global\{B1C88161-12B6-D5EF-E26F-1BC2E62A38DF}	SUCCESS	0x00000000
13:19:34,728	2392	NtOpenMutant	Handle => 0x0041d66e MutexName => Global\{2F77C11A-52CD-4B50-E26F-1BC2E62A38DF}	FAILURE	0xc0000034
13:19:34,728	2392	RegCreateKeyExW	Handle => 0x0000012c Access => 260 Registry => 0x80000001 Class => SubKey => Software\Microsoft	SUCCESS	0x00000000
13:19:34,728	2392	RegCreateKeyExW	Handle => 0x00000130 Access => 259 Registry => 0x0000012c Class => SubKey => Zobeorygcaco	SUCCESS	0x00000000
13:19:34,728	2392	RegCloseKey	Handle => 0x00000130	SUCCESS	0x00000000
13:19:34,728	2392	RegCloseKey	Handle => 0x0000012c	SUCCESS	0x00000000

Figure 5.11 Functions calls for process spawning, creation of mutex and regkey

tend to have irregular names that are characteristic for certain malware. [47] The figure 5.11 displays functions calls that arranged spawning of processes, creation of mutexes and registry keys, where malware's necessary information is stored.

The dropped executable was marked by 50 antivirus engines on VirusTotal's website as a Trojan horse malware that is used to spread Zeus banker infection. The described example detection reflects known patterns that are used by this malware in terms of location where the dropped file is stored, the location of created registry keys and also mutexes, which are named with random regular expressions in length of 32 characters. [7][20] The severity of this threat is high due to its capabilities to steal user's personal information regarding credentials for emails, baking applications and other various accounts. Such is achieved via, for example, deployed key-logger, capturing of screenshots, cookies and more.

5.6 Evaluation

The experiment demonstrated how Cuckoo Sandbox operates and what the options it provides to its user are. It is a powerful engine that accumulates wide spectrum of modular extensions in order to provide comprehensive overview of events that take place in the system when a file is processed, i.e. executed, loaded, opened or visited for example. It is implemented in a way to closely cooperate with databases of already collected intelligence and compare the analyzed resources against this knowledge. This way it can quickly propose an evaluation of a sample and direct the researcher towards the conclusion, however the final resolution should be made by the researcher. Thanks to the nature and scope of the data it collects, it gives the analyst a possibility to examine unknown suspicious resources and occasionally reveal zero-day vulnerabilities. Additionally, in such cases collected data can serve as a basis for creation of signatures for revealed malware.

6 SUMMARY

The main topic of the thesis is a *Client Honeypot*, which is a tool that helps users to closely investigate potentially malicious resources that can be found circulating over the Internet posing threat to vulnerable users. The thesis discusses theoretical aspects of this area of computer security and gives thorough background on the most important parts, thus accumulates the intelligence in one place. There are several of publicly known implementations described in the thesis, which gives a reader an overview on the current state of art. For the reference also outdated solutions, which were forming the field as it is known today, are presented to deliver a possibility to compare features now and then. After the description of honeyclients, a single solution was chosen to serve a reader with very close review of a powerful honeypot as well as to show off the setup process and propose a configuration of environment. Later, an actual experiment is conducted using the configured environment that shows how actual operations work and what are the outcomes of them.

The experiment revealed that the problem of maliciousness on the Internet is ongoing as well as the diversity of techniques used by assailants in order to abuse vulnerable users and the applications they use on daily basis. Moreover, it proved the suggested configuration of the environment as well as the usefulness and necessity of malware research, which does not require powerful computer and can be done, in fact, by everyone interested.

The main problem, which is a reason for tools like honeyclients to originate, is malicious software that abuses Internet users all over the world. The thesis tries to engage unfamiliar users to join the efforts of malware analysis and investigation. Awareness about the problem is one of the key elements in the fight against malware. The penetration of modern computer technology is becoming bigger and more and more criminals convert into abusing cyber-space. Moreover, the arriving young generation of users is better aligned with computers and see many opportunities in utilization of World Wide Web, thus the number of cyber-criminals raises daily. In order to keep-up with criminals, the research of their actions needs to become more robust as well. The more people join the research the better the defense can be implemented in antivirus engines for example. On the other hand, to mitigate risks of getting attacked by malware, users need to be educated about proper conduct on the Internet. Additionally, users can fortify themselves by using different secure passwords on every respective account they have, update the software they use as soon as the newer version is released, or audit the permissions of their user accounts within the system to lower the number of options an attacker has after he penetrates into the system.

The complexity and every-day updating of malware opens wide space for possible future work. The fight against malicious software needs to keep up and adjust dynamically in hand with the actual trends in malware. The field of honeypotting can be, for instance, enriched with new techniques to mitigate the rate of successful escape of malware from triggering when a virtual environment is detected. Augmented implementation of functionality to scan email inboxes carefully and investigate links retrieved, as abundant share of malware is distributed via spam messages, can be beneficial, or the field can be enhanced by utilizing large distributed setups in order to strengthen the performance and reliability of malware research.

BIBLIOGRAPHY

[1] *About Capture Client Honeypot* [online]. [quoted 4.4.2014] Available from <https://projects.honeynet.org/capture-hpc/wiki/AboutCapture>

[2] *About HoneyC* [online]. [quoted 2.12.2014] Available from: <https://projects.honeynet.org/honeyc/wiki/AboutHoneyC>

[3] *Cross Site Scripting (XSS) Attack* [online]. [quoted 27.9.2014] Available from <https://www.acunetix.com/websitesecurity/cross-site-scripting/>

[4] Cuckoo Foundation. *Preparing the Host* [online]. [quoted 24.11.2014] Available from: <https://cuckoo.readthedocs.org/en/latest/installation/host/>

[5] *The Difference between a Computer Virus, Worm and Trojan Horse* [online]. 28.10.2004. [quoted 28.3.2014] Available from <http://www.webopedia.com/DidYouKnow/Internet/virus.asp>

[6] EGELE, Manuel; ENGIN, Kirda; KRUEGEL, Christopher. *Mitigating Drive-by Download Attacks: Challenges and Open Problems.* Available from <http://www.iseclab.org/papers/inetsec09.pdf>

[7] *ESET Virus Radar: Win32/Spy.Zbot.AAU* [online]. [quoted 4.1.2015]. Available from <http://www.virusradar.com/en/Win32_Spy.Zbot.AAU/description>

[8] GUARNIERI, Claudio; TANASI, Alessandro; BREMER, Jurriaan; SCHLOESSER, Mark. *Cuckoo Sandbox Book, Release 1.1.* 29.10.2014. Available from <https://readthedocs.org/projects/cuckoo/downloads/pdf/stable/>

[9] GÖBEL, Jan Gerrit; DEWALD, Andreas. *Client-Honeypots: Exploring Malicious Websites.* Oldenbourg Wissenschaftsverlag GmbH, 2011. ISBN 978-3-486-70526-3

[10] *herdProtect Anti-malware: About* [online]. [quoted 19.4.2014] Available from <http://www.herdprotect.com/about.aspx>

[11] *Honeyspider Network 2: Services* [online]. [quoted 15.4.2014] Available from <http://www.honeyspider.net/Services.html>

[12] *Honeyspider Network 2: Workflows* [online]. [quoted 15.4.2014] Available from <http://www.honeyspider.net/Workflows.html>

[13] *Internet live stats: Internet Users* [online]. [quoted 25.3.2014]. Available from <http://www.internetlivestats.com/internet-users/>

[14] JACEWICZ, Pawel. *Specification of HSN 2.0 JavaScript Static Analyzer.* Last edited 8.11.2012. Available from <http://www.honeyspider.net/docs/HSN2_js_static_analyzer.pdf>

[15] KILGER, Max. *The Honeynet Project: Motivations for Malicious Online Behavior and Consequent Emerging Cross-National Cyberthreats*. July, 2010.Available from <http://icc.ite.gmu.edu/csga2010/Max_Kilger.ppt>

[16] LASOTA, Krzysztof; PAWLINKSI, Pawel. *Data Contract Specification for HSN 2.0 Services*. Last edited 5.12.2012. Available from <http://www.honeyspider.net/docs/HSN2_data_contract.pdf>

[17] LEVY, Elias (Aleph One). *Smashing the Stack for Fun and Profit* [online]. Phrack Magazine No. 49. 11.8.1996. Available from <http://phrack.org/issues/49/14.html#article>

[18] MANSOORI, Masood; WELCH, Ian; FU, Qiang; *Yalih, Yet Another Low Interaction Honeyclient*. 2014. Available from <http://crpit.com/confpapers/CRPITV149Mansoori.pdf>

[19] *McAfee website: How it works* [online]. [quoted 14.3.2014]. Available from <http://www.siteadvisor.com/howitworks/index.html>

[20] *Microsoft Malware protection Center: Win32/Zbot* [online]. [quoted 4.1.2015] Available from <http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32/Zbot>

[21] *Microsoft Security Bulletin MS08-078*. 18.12.2008. Available from <https://technet.microsoft.com/en-us/library/security/ms08-078.aspx>

[22] *Microsoft Security Intelligence Report, Volume 15*. January through June 2013. Available from <http://www.microsoft.com/en-eg/download/details.aspx?id=40871>

[23] *Microsoft Support: Internet Control Message protocol (ICMP) Basics* [online]. [quoted 19.12.2014] Available from <http://support.microsoft.com/kb/170292>

[24] MITRE CORPORATION, The. *The MAEC Language Version 4.0.1 Specification*. 15.11.2013. Available from <http://maec.mitre.org/language/version4.0.1/MAEC_Language_Specification_11-15-2013.pdf>

[25] MULLEN, Patrick; PETNEY, Ryan. *Razorback*. Last updated 4.8.2010. Avaliable from <http://sourceforge.net/projects/razorbacktm/files/Presentations/razorback-slides-1.1.pdf/download>

[26] *NetMarketShare: Desktop Top Operating System Share Trend* [online]. [quoted 15.12.2014] Available from <http://www.netmarketshare.com>

[27] NSS Labs. *Vulnerability Threats Trends: A Decade in Review, Transition on the Way*. February 2013. Pages 4-8. Available from <https://www.nsslabs.com/reports/vulnerability-threat-trends>

[28] PROVOS, Niels; THORSTEN, Holz. *Virtual Honeypots: From Botnet Tracking To Intrusion Detection*. Pearson Education Inc., 2008. ISBN 0-321-33632-1

[29] *Sandboxie* [online]. Available from <http://www.sandboxie.com>

[30] SCARFONE. Karen; MELL, Peter. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. National Institute of Standards and Technology Special Publication 800-94, February 2007. Available from <http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf>

[31] *The Size of the World Wide Web (The Internet)* [online]. [quoted 25.3.2014]. Available from <http://www.worldwidewebsize.com/>

[32] SPAFFORD, Eugene H. *The Internet Worm Program: An Analysis*. 8.12.1988. Available from <http://spaf.cerias.purdue.edu/tech-reps/823.pdf>

[33] SPITZNER, Lance. *Honeypots: Tracking Hackers*. Pearson Education Inc., 2003. ISBN 0-321-10895-7

[34] *StatCounter Global Stats: Top 7 desktop Oss on Nov 2014* [online]. [quoted 24.11.2014] Available from <http://gs.statcounter.com/?PHPSESSID=drd7u86f66gr1djns6fki1k0i0#desktop-os-ww-monthly-201411-201411-bar>

[35] *Top 10 Worms* [online]. [quoted 28.3.2014] Avialable from <http://www.secpoint.com/Top-10-Worms.html>

[36] *urlQuery website: About urlQuery* [online]. [quoted 26.3.2014]. Available from <http://urlquery.net/about.php>

[37] VERACODE. *Common Malware Types: Cybersecurity 101* [online]. 12.10.2012. [quoted 27.9.2014] Available from <https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101>

[38] *VirusTotal website: About VirusTotal* [online]. [quoted 25.3.2014]. Available from <https://www.virustotal.com/en/about/>

[39] *VirusTotal website: Credits & Acknowledgements* [online]. [quoted 25.3.2014]. Available from <https://www.virustotal.com/en/about/credits/>

[40] *W3schools.com: HTML Plug-ins* [online]. [quoted 11.10.2014] Available from <http://www.w3schools.com/html/html_object.asp>

[41] WANG, Yi-Min et al. *Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities*. Microsoft Research, Redmond. Available from <http://research.microsoft.com/en-us/um/redmond/projects/strider/honeymonkey/NDSS_2006_HoneyMonkey_Wang_Y_camera-ready.pdf>

[42] *Web 2.0 (or Web 2)* [online]. March 2011. [quoted 5.4.2014] Available from <http://whatis.techtarget.com/definition/Web-20-or-Web-2>

[43] *What is JavaScript?* [online]. [quoted 6.10.2014] Available from <https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript>

[44] *Windows lifecycle fact sheet* [online]. [quoted 15.12.2014] Available from <http://windows.microsoft.com/en-us/windows/lifecycle>

[45] *The World Wide Web Consortium (W3C): HTML & CSS* [online]. [quoted 11.10.2014] Available from <http://www.w3.org/standards/webdesign/htmlcss>

[46] *The World Wide Web Consortium (W3C): What is the Document Object model?* [online]. 1.10.1998. [quoted 6.10.2014] Available from <http://www.w3.org/TR/REC-DOM-Level-1/introduction.html>

[47] ZELTSER, Lenny. *SANS Digital Forensics and Incident Response Blog: Looking at Mutex Objects for malware Discovery and Indicators of Compromise* [online]. 24.7.2012. [quoted 4.1.2015]. Available from <http://digital-forensics.sans.org/blog/2012/07/24/mutex-for-malware-discovery-and-iocs>