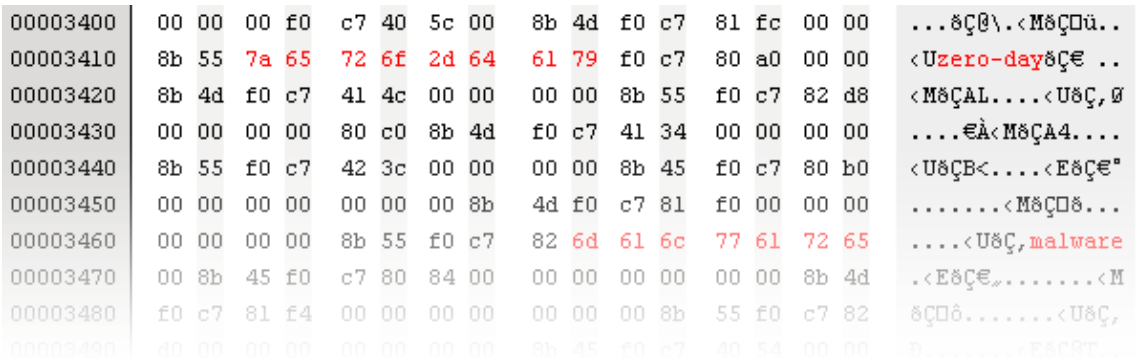
NTNU
Innovation and Creativity

Zero-day Malware

Project Assignment



Trondheim, December 17, 2008

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Telematics

Finn Michael Halvorsen, Rune Walsø Nergård and Håvard Vegge

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING

PROJECT ASSIGNMENT

Students:	Finn Michael Halvorsen, Rune Walsø Nergård and Håvard Vegge
Course:	TTM4530
Title:	**Zero-day Malware**

Description:

The current trend in malware is increased stealth for the purpose of creating large, undetected botnets. Coupled with the closing gap between time of vulnerability detection to time of available exploit, this leads to an increasing lag time for anti-malware vendors. The task is two-fold:

1. Design a laboratory testbed consisting of updated Microsoft Windows PCs, with updated anti-malware software installed, and then expose these systems to known suspicious sites, file-sharing systems, etc.
2. At a given time afterwards perform an offline malware search of the system with updated anti-malware tools to determine whether the system was infected with zero-day malware.

Deadline:	December 17, 2008
Submission date:	December 17, 2008
Carried out at:	Department of Telematics
Supervisor:	Martin Gilje Jaatun, SINTEF ICT
Co-Supervisor:	Jostein Jensen, SINTEF ICT

Trondheim, December 11, 2008

Danilo Gligoroski
Professor

Abstract

There has been an enormous increase in malware *variants* during the last year. This has made it even more difficult for the anti-malware vendors to maintain protection against the vast amount of threats. Various obfuscation techniques, such as polymorphism, contribute to this trend. The ongoing battle between malware creators and anti-virus vendors causes an increasing signature lag, which leads to vulnerable end-systems for home users as well as in corporate environments.

We define *zero-day malware* as malware that is able to infect a host because of the absence of matching malware signatures or other malware detection techniques in anti-virus programs. There has been a lack of scientific research in this field, and as far as we are aware of, no studies have been conducted to map the prevalence of zero-day malware. This report documents one of the first attempts to use an empirical approach to identify the prevalence of zero-day malware.

A laboratory environment consisting of Microsoft Windows computers was set up with updated anti-virus software, and exposed to various potentially malicious sources. The identification of zero-day malware is derived from a baseline malware scan compared to a final scan performed one month later. During this month the computers were turned off. Malware that was detected only in the final scan would be classified as zero-day malware, as this implies that the anti-virus software had gained detection capability for this malware during this dormant phase.

Our methodology proved effective for detecting zero-day malware and our results show a remarkably high presence of such malware. A total of 124 zero-day malware instances were identified during the experiment. This shows that even with updated anti-virus software there is a relatively high risk of getting infected.

This report has been the basis for the paper *Where Only Fools Dare to Tread: An Empirical Study on the Prevalence of Zero-day Malware*, submitted to the The Fourth International Conference on Internet Monitoring and Protection, ICIMP 2009, in Venice, Italy [29]. This paper is included in Appendix F.

Preface

This report serves as a specialization project in Information Security in the 9th semester of the Master's Programme in Communication Technology at The Norwegian University of Science and Technology, NTNU. The assignment was given by SINTEF ICT, and the project has been carried out at a laboratory located at their premises in S. P. Andersens vei, Trondheim.

This project has been a challenging and inspirational task in an emerging field of malware research. Working on a topic with limited previous research has been motivating and exciting, we also enjoyed the practical aspect of the experiment.

We would like to thank our supervisor Martin Gilje Jaatun for his valuable input and weekly feedback. We appreciate his enthusiasm and that he has been a driving force throughout the project. His contribution to the scientific paper based on our project is also appreciated. Also thanks to Jostein Jensen, our Co-Supervisor, for his helpful comments and insightful ideas.

Finally we would like to thank our professor Danilo Gligoroski, and wish him good luck in the SHA-3 competition.

Trondheim, December 17, 2008

Rune Walsø Nergård Finn Michael Halvorsen Håvard Vegge

Abbreviations

AV Anti-Virus

BART Bootable Antivirus and Recovery Tool

CPU Central Processing Unit

DNS Domain Name Server

DDoS Distributed Denial of Service

EULA End User License Agreement

EXE Executable (file extension)

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IP Internet Protocol

IT Information Technology

KVM Keyboard, Video, Mouse

MBR Master Boot Record

MSN Microsoft Network

NOP No Operation

P2P Peer-to-Peer

SP Service Pack

TSR Terminate and Stay Resident

TTL Time To Live

UAC User Access Control

UDP User Datagram Protocol

URL Uniform Resource Locator

VLAN Virtual Local Area Network

WMF Windows Meta File

Definitions

BitTorrent tracker A BitTorrent tracker is a server that holds information about torrents (defined below) and connected peers. A tracker should be differentiated from a BitTorrent index by the fact that it does not necessarily list files that are being tracked.

Black Hat In computer security a *Black Hat* is a person which conduct unauthorized penetration of information systems. In the field of malware research this term is used to describe persons which create and distribute malware. Also see *White Hat*, defined below.

Desktop malware search A desktop malware search is performed while the operating system is running, this will most commonly be performed by anti-virus software installed on the computer.

In the wild For malware to be considered *in the wild*, it must be spreading as a result of normal day-to-day operations on and between the computers of unsuspecting users. The opposite of in the wild is *zoo* - defined below.

Offline malware search An offline malware search is a search performed without booting the host operating system, for instance by using a bootable anti-virus CD or scanning the hard drive with another computer. This type of scan can detect malware that is using various hiding techniques.

Patch A patch is a small piece of software designed to fix problems with or update a computer program or its supporting data. Though meant to fix problems, poorly designed patches can sometimes introduce new problems.

Servent Servent is a term used by Gnutella to describe peers. Peers in Gnutella act as both SERvers and cliENTS, thus the name SERVENT.

Signature lag Signature lag is the time between an anti-virus vendor is aware of the existence of a malware type, until a signature is added to their database.

Torrent A torrent is a a metainfo file containing information about the data you want to download and the tracker. It is typically named *filename*.torrent. The BitTorrent client needs the .torrent file to connect to the tracker and download what the user requests.

Vulnerability window Seen from a malware researcher's point of view, the vulnerability window is the time between a malware instance, for example a virus, is spread on the Internet to when the a patch or signature is released by the anti-virus vendors.

Warez Warez is a term used in the underground scene to describe pirated and/or cracked software.

White Hat In computer security a *White Hat* or *Ethical Hacker* is a person which conduct authorized penetration testing of information systems. In the field of malware research this term is used to describe persons which create malware for proof-of-concept or test use.

Zero-day exploit A zero-day exploit is an exploit used in a malicious computer attack that takes advantage of a security hole before the vulnerability is known to the software developers or before the developers manage to make a patch, even though they are aware of the vulnerability.

Zero-day malware Zero-day malware is a type of malware that is able to infect a host because of the absence of malware signatures or other malware detection techniques in the anti-virus program. A zero-day malware do not need to involve a zero-day exploit.

Zoo A zoo is a collection of malware that merely exist, typically only in virus and anti-virus labs, not in the wild. They are not spreading.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

IT organizations are constantly fighting to keep their systems patched and updated, while malware creators try to translate vulnerability discoveries into new exploits. Looking back in history, the time between vulnerability detection and time of available exploit code has decreased from months, to weeks, to zero days. This development is also illustrated in Table 1.1, which shows an overview of the most widespread worms over the last years.

Patch	**Malware**	**Patch availability**	**Worm attack date**	**Days for worm to appear**
MS01-020	Nimda	Oct 17th, 2000	Sept 18th, 2001	335 days
MS02-061	Slammer	July 24th, 2002	Jan 25th, 2003	185 days
MS03-026	Blaster	July 16th, 2003	Aug 11th, 2003	26 days
MS04-011	Sasser	Apr 13th, 2004	Apr 30th, 2004	17 days
MS05-039	Zotob	Aug 09th, 2005	Aug 14th, 2005	5 days
MS06-040	Mocbot	Aug 08th, 2006	Aug 12th 2006	4 days

Table 1.1: The time between the patch was available and when malware creators had incorporated the vulnerability into a worm candidate.

For IT administrators, this leaves very little time to apply updates and patches to all computers in their networks. Even worse, what happens if someone creates malicious software by exploiting a vulnerability that is not discovered by the security community? Having such an unknown piece of malware running wild inside your organization is certainly a real threat.

Incidents where malware has infected big companies and caused damage are not uncommon. The US Department of Defense has recently banned the use of USB drives, after an attack by a worm identified as *Agent.btz* [20]. Another example is from the biggest bank in Norway, DnB Nor, where a worm *viking.gt* infected a total of 11.000 computers through the whole country in November 2007 [15]. The bank was using an anti-virus solution from Norman and apparently there was an error in the real time protection mechanism. In retrospect DnB Nor has estimated their economic loss to be somewhere in between 20 and 80 million NOK, due to cleanup and indirect expenses.

1.1 Motivation

As already mentioned, it is possible to get infected by malware, even though all your programs are up to date and all security patches are installed. Falling victim to an attack which none of your security systems discover is a real threat, and has also been discussed by the IT administrator and security professionals at SINTEF ICT.

A second motivation is the fact that, although the concept of zero-day exploits and malware have been around for years, no major studies or scientific articles seem to have been published. Literature about the topic consists primarily of loose web articles with limited details. This makes our project very interesting, but also a challenging task.

1.2 Related Work

There is no doubt that downloading files from file-sharing networks constitute a great risk. According to a study of malware prevalence in Kazaa by Shin et al. [SJB06], 15% of the 500,000 files that were downloaded were infected by malware. A similar study of malware in both the Gnutella and OpenFT network was carried out by Kalafut et al. [KAG06]. Their report showed that from over one month of data collection, 68% of all downloadable responses in LimeWire/Gnutella contained malware, while the corresponding number for OpenFT was only 3%. While the mentioned peer-to-peer networks have been thorougly examined, little has been done in BitTorrent so far. A computer security project by Andrew Berns at The University of Iowa [Ber08], however, showed that 70 out of 379 downloads from the BitTorrent network contained malware (18.5%).

Many companies or web sites test different anti-virus software on a regular basis. Two of the biggest actors in this area are AV-Comparatives.org and AV-Test.org. As such companies are comparing anti-virus vendors, their methodology is not the same as in this project where we are searching for zero-day malware. Still, a proactive/retrospective test performed by AV-Comparatives [Cle08], can give indications of what results to expect. A retrospective test is used to test the proactive detection capabilities of scanners. It gives an idea of how much new malware a scanner can detect (for example by heuristic/generic detection), before a signature is provided for the malware.

Several studies have reported that static analysis of malware have severe limitations [CJS+05, CJ04, MKK07, Per08] and some goes as far as proclaiming that traditional anti-virus is dead [Blo06, Jaq07]. Because of the increasingly number of new malware instances and also more sophisticated code, researchers are forced to come up with improved solutions for detecting these threats. An approach based on behavioral software analysis is proposed by Jostein Jensen from SINTEF ICT. In his article [Jen08] a novel way of using honeypot technology is proposed to build a testbed for analyzing current threats. The laboratory environment presented is based on the ideas of sandboxed environments [Nat01], but also allow controlled connections to the Internet during the analysis phase.

1.3 Objectives

This project consists of two main parts. First a thorough theoretical background study in the field of malicious software is to be performed. Secondly we will develop a method and

testbed for detection of zero-day malware. By implementing this, we aspire to achieve the following objectives:

- Map the prevalence of zero-day malware.
- If found, from which sources do zero-day malware originate?
- Is it possible to say something about the vulnerability window, and the signature lag of anti-virus vendors?

1.4 Limitations

Zero-day is a broad term and can be applied to various areas of information security. Often people associate zero-day with software vulnerabilities which are not known to the public, and the creation of zero-day exploits. This report is focused towards zero-day malware, that is, malicious software which is not detected by anti-virus programs due to the absence of virus signatures or other malware detection techniques in the anti-virus program. This means that we are not exclusively looking for malware which is using zero-day exploits, but also malware that is not detected due to the lack of existing virus signatures etc. A thorough analysis of malware is not the focus of this project but a short explanation of some of the malware found, is given in Section 6.2. Aspects like trading of zero-day exploits and the community around it, are out of the scope of this project. Refer to Section 2.2 for a more detailed description of zero-day malware.

Although we are using several different anti-virus packages, this is not a comparison of such programs. The reason for having different anti-virus engines is to verify that results are real, and not just a peculiarity in one of the engines. As an example, if one anti-virus engine recognizes a piece of malware to be zero-day, but all the other engines do not, the conclusion would be that this is zero-day malware to that specific engine only.

According to SANS Institute [Ins07], all operating systems and all software applications are vulnerable to zero-day vulnerability discovery and exploitation. This project is limited to detection of zero-day exploits and malware threatening Windows XP and Internet Explorer. Actually we have installed Windows Vista on one computer, mostly to indicate to what extent this newer operating system is exposed to zero-day malware compared to Windows XP. Another possibility could have been to install different web browsers, media players, office software, etc., but this has not been our focus.

It was originally planned to use spam e-mail as an additional infection source, and we set up an e-mail account for this use. While we did receive a lot of spam, time constraints made us unable to implement this in our experiment and this approach is left as further work (See Section 7.10).

The procedure of the experiment is carried out manually. Together with a strict time period, this makes the amount of visited web sites and number of downloaded files somewhat limited. In order to increase the extent of system exposure, crawlers or other techniques could be used. Such strategies are described in Section 7.10.

1.5 Method

The method we used to find zero-day malware can roughly be divided into four parts:

- Laboratory setup
- System exposure phase
- Dormant phase
- Compare results

An outline of the method can be seen in Figure 1.1. For a more thorough overview of our methodology see Chapter 5. First the laboratory computers were installed with a fresh copy of Microsoft Windows which was updated with all the latest security updates, anti-virus and anti-spyware were also installed. A malware-scan was performed to ensure that the laboratory setup was clean of any known malware. In the system exposure phase the laboratory was exposed to various sources of malware, including peer-to-peer networks and the web. After this phase was completed the computers were scanned with two updated anti-virus programs to produce a baseline result, then all the laboratory computers were shut down for a month. After this month had passed, additional malware-scans were performed with updated anti-virus definitions, and the results of these new scans were compared to the baseline scans performed a month earlier. Any detections made in the final scan, but not in the baseline scan, would indicate the presence of zero-day malware. The dormant phase of one month was chosen due to time-constraints and we believed it to be a reasonable time frame.

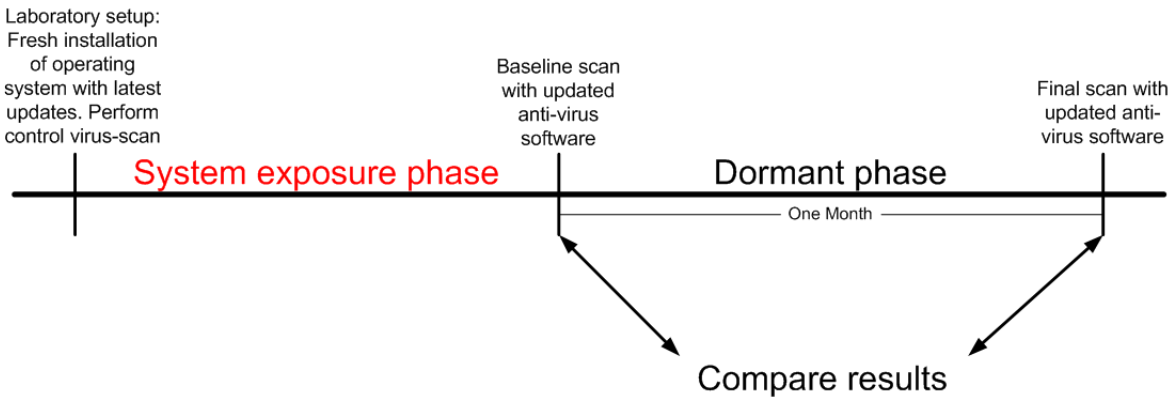


Figure 1.1: An outline of the method used to find zero-day malware.

1.6 Document Structure

The remainder of this report is organized as follows:

Chapter 2: Background presents theory related to malware, different sources of infection and techniques that anti-virus software use to detect such threats. This is also the chapter, where we define our understanding of the term *zero-day*.

Chapter 3: Laboratory Environment describes the laboratory environment at SINTEF ICT, together with our chosen hardware and software.

Chapter 4: Pre-study is an overview of experiences we made early in our work, and which were incorporated in the experiment procedure. In the risk section, we try to emphasize some of the challenges that could arise in a project of this character.

Chapter 5: Experiment Procedure describes how we carried out the practical work in the project. This includes some preparations, a period of system exposure and malware search.

Chapter 6: Results presents the main findings of the experiment which is further evaluated in Chapter 7. Along with hard facts from the anti-virus scans we include our experiences with both the file-sharing networks and the different anti-virus packages.

Chapter 7: Discussion evaluates the experiment procedure and the results. Finally, we present some ideas for future work.

Chapter 8: Conclusion summarizes the major results and findings, and concludes the project.

Additionally, the following appendices are included:

Appendix A: Suspicious Sites lists our initial list of suspicious web sites.

Appendix B: Offline Anti-virus CDs contain step-by-step descriptions for obtaining and using avast! BART CD and Knoppix Live Linux CD/F-PROT.

Appendix C: List of Zero-day Malware lists all the 124 zero-day malware infections found in Avast! BART and F-PROT. The same list with detection names reported by F-Secure and Symantec are also included.

Appendix D: Logs from VirusTotal for the Wimad example includes the logs after scanning a malware using the Wimad exploit.

Appendix E: Attached CD lists the content (scan logs etc.) of the enclosed CD.

Appendix F: Paper includes the scientific article based on this report which has been submitted to the Fourth International Conference on Internet Monitoring and Protection, ICIMP 2009.

Chapter 2

Background

In this chapter theory related to malware and especially zero-day malware and exploits are presented. Also our understanding of the term *zero-day* and *zero-day malware* is defined. The chapter continues with an thorough description of malware sources, with emphasis on file-sharing networks. Finally different malware detection techniques are reviewed.

2.1 Malware

Malware, also known as Malicious Software, is software that is intentionally included or inserted in a computing system for a harmful purpose. To get a common understanding of the terms, this section is an overview of such software threats. The classification of malware types is not perfect, in the sense that the groups often overlap and the difference is not always obvious.

2.1.1 Viruses

Definition: *"A computer virus is a program that can infect other programs by modifying them to include a possibly evolved copy of itself."* [Coh87]

A virus is a computer program or script that can copy itself and infect a computer whenever the infected computer comes in contact with an uninfected piece of software. Viruses require human interaction to propagate and are able to spread using a variety of methods, for instance by sending it over a network or by users who are exchanging disks, CDs or other storage devices. Typically a virus attaches itself to another program and executes when the host program is run. It can then perform similar tasks as other programs, for instance deleting or moving data files.

There are a lot of different types of viruses, and as anti-virus software vendors find countermeasures, new and more advanced viruses are developed. Stallings suggests the following main categories [Sta06]:

- *Parasitic viruses* attach themselves to programs, also known as executables. When that program is launched, the virus is executed first, and tries to infect other executable files.
- *Memory-resident viruses* (also called TSR for Terminate and Stay Resident) load in

main memory in order to infect programs that execute. This as opposed to non-resident viruses which infect programs found on the hard drive.

- *Boot sector viruses* infects the boot sector of a floppy disk or hard drive. It can also infect the master boot record (MBR). Once the boot code on the drive is infected, the virus will be loaded into memory on every startup.
- *Stealth viruses* are designed to hide themselves from anti-virus software. Typically, when an anti-virus program runs, a stealth virus hides itself in memory, and uses various tricks to hide changes it has made to files or boot records. A stealth virus could also use compression or encryption to hide itself.
- *Polymorphic viruses* are able to mutate with every infection, which makes it hard to detect a virus by its signature. Such viruses create copies during replication that are functionally equivalent, i.e. the original algorithm is intact, but the bit patterns are distinctly different.
- *Metamorphic viruses* are even more difficult to detect as they rewrite themselves completely. Both behavior and appearance can be changed. In order to detect such viruses, anti-virus software will need some kind of emulation to analyze the code behavior.

The *Brain* virus [21] and the *Melissa* virus [57] are two examples, from 1986 and 1999, respectively. Brain is considered the oldest PC virus known and infected the boot sector of floppy disks. Even though it was the first virus, it had some stealth capability. When an attempt was made to read the infected boot sector, the virus would show you the original boot sector instead. Melissa was a macro-virus arriving in Microsoft Word documents. If an infected file was opened, then the macro in the document would run and attempt to mass mail itself from the e-mail client Microsoft Outlook.

2.1.2 Worms

Definition: *"A worm is a self-replicating piece of code that spreads via networks and usually doesn't require human interaction to propagate."* [SZ04]

In addition to sending copies of itself to other nodes, a worm usually performs some unwanted function. This specific action is written in the worm's *payload*, which is a chunk of code executing on behalf of the attacker on the target system. Some worms are only designed to spread, but they can still cause damage by disrupting networks and using bandwidth.

There are a lot of possible options to include in a payload. One is to open a backdoor, as described in section 2.1.3, which gives the attacker complete control of the target system remotely. Another possibility is planting a distributed denial of service flood agent, also known as a zombie or bot. This is a special kind of backdoor which waits for the attacker to send a command to possibly flood another victim machine.

Compared to viruses, the biggest difference is that a worm can replicate itself, without human interaction. To replicate, a network worm uses some sort of network vehicle. Stallings suggests three examples [Sta06]:

- *Electronic mail facility:* A worm mails a copy of itself to another system.
- *Remote execution capability:* A worm executes a copy of itself on another system.
- *Remote login capability:* A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

Two examples are *SoBig* and *MyDoom* [WBMW04] from 2003 and 2004, respectively. They are both email worms and disrupted normal network operations and caused unexpected downtime and increase in associated IT expenses. Unlike a scanning worm, which needs to aggressively search for new victims to compromise, an email worm has much higher hit rate because it obtains targets from victim machines. SoBig and MyDoom utilized spreading techniques such as harvesting legitimate domain names from victim machines (e.g., by scanning web caches and hard disks) then attempting to construct probable addresses. Although categorized as mass-mailing worms, both SoBig an MyDoom also replicated over file sharing clients (e.g. Kazaa).

2.1.3 Backdoors

Definition: *"A backdoor is a program that allows attackers to bypass normal security controls on a system, gaining access on the attacker's own terms."* [SZ04]

Backdoors are created by developers either for debugging purposes as a way to bypass authentication or setup procedures, or as a hidden way of gaining unauthorized access to the system after it has been deployed. Debug backdoors should be removed from production software, but can be left there unintentionally by mistake, and can therefore be described as a vulnerability in the software. Software with backdoors that are left intentionally, as a way to gain unauthorized access, can be categorized as a Trojan Horse. The backdoor can be triggered by using special login credentials, or some special and unlikely sequence of input to the system [Sta06].

2.1.4 Trojan Horses

Definition: *"A Trojan Horse is a program that appears to have some useful or benign purpose, but really masks some hidden malicious functionality."* [SZ04]

The term is derived from the classical story of the Trojan Horse, where an army managed to sneak right through a highly fortified gate. In a similar fashion, today's Trojan horses try to bypass for instance firewalls, by employing similar trickery. According to Skoudis and Zeltser [SZ04] Trojan horse programs are used for the following goals:

- Tricking a user or system administrator into installing the Trojan horse in the first place. The Trojan horse and the ignorant user will then become the entry vehicle for the malicious software on the system.
- Blending in with the normal programs running on the machine. By camouflaging itself to appear to belong on the system, users and administrators will continue their activity, unaware of the malicious code's presence.

Typically, Trojan horses are contrasted with viruses and worms in that Trojan horses do not replicate. Beyond that, the categories of malicious code are not mutually exclusive. Remote access Trojan horses with backdoor capabilities are becoming more common [KE03]. When executed, the Trojan horse enables some form of remote access and control to the compromised system by unauthorized persons.

2.1.5 Rootkits

Definition: *"A rootkit is a malicious software designed to modify the underlying operating system of an infected computer to hide other malicious programs from the user of the system."* [51]

The word *rootkit* is composed of *root* and *kit*. *Root* refers to the administrator account on Unix and Linux systems and *kit* refers to a set of programs that allow someone to maintain root-access to a computer. A rootkit allows a legitimate or a malicious user to get control over a computer system, without system management utilities, anti-virus programs or spyware blockers knowing about it. Rootkits are useless by themselves and are therefore used by other forms of malware to hide. By having root-access provided by the rootkit you will be able to execute files, changing system configuration, accessing log files and monitoring activity on the target system [51, 49] [HB06].

2.1.6 Bots and Botnets

Definition: *"A bot, which is short for robot, is a type of malware which allows an attacker to gain control over the affected computer."* [50]

More specific a bot is an automated software program that can execute certain commands and with that carry out specific tasks when it receives a specific input. Computers that are infected with a bot are referred to as *drones* or *zombies*. There are numbers of computers on the Internet which is infected with some type of bot, and many do not even realize it. This is confirmed by Figure 2.1. The graph is based on actual zombie/bot counts from botnets tracked by Shadowserver [54], on the networks they are monitoring, during 2008. The 30-days entropy, mentioned in the figure, means that if no activity on a specific IP is seen within 30 days, that IP should be considered dead for the purposes of counting infected systems. The reports are updated by Shadowserver every 15 minutes [53]. The bots are good at hiding and the attacker do not want you to discover it so that you can remove or disable the running process. The processes or files often have the similar or identical names as normal system file names and processes so that the user would not react when they see the process in the task manager. Agobot, SDBot, SpyBot and GT Bot are all examples of known bots [50, 52].
Definition: *"A botnet is a collection of zombies, connected to the Internet, that interact to accomplish some distributed task."* [52]

A botnet is typically used for illegal purposes even though it can also be used for constructive applications. The person who controls the botnet is called a *bot herder*. Two examples of what a botnet can be used for are Distributed Denial of Service (DDoS) attacks and spam.

Botnets can be used in a DDoS attack against other machines on the Internet, for ex-

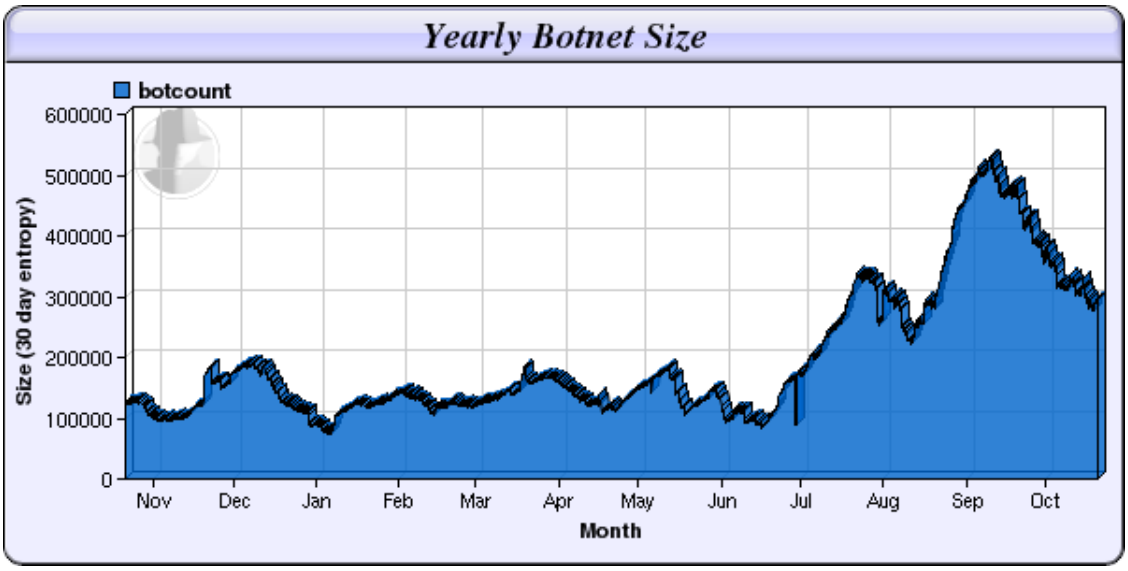


Figure 2.1: Zombie/bot counts from botnets tracked by Shadowserver during 2008 [53].

ample a server, to saturate its bandwidth or other resources. Such attacks can prevent access to a particular site for a long time and this may be critical to the financial situation for a corporation since they are unable to reach out to their customers on the web. There have also been episodes where criminals have demanded payment from the company to stop the DDoS attack against them. The reason why DDoS attacks are that powerful is because the botnet gives the criminal extremely many network resources through the compromised computers in the botnet. And because of this it may be very difficult to prevent or mitigate this type of attack. If anyone should try to trace the attack back to its source, they will find an unwitting compromised computer instead of the true attacker.

Botnets are also used to propagate spam. This works in the fashion as DDoS attacks, but instead the zombies now forward spam emails. Instant messaging accounts may also be utilized to send malicious links or advertisements to everyone on the victim's contact list. This means that gathering zombies is a key task for the bot herder. The more zombies the bot herder is controlling, the more impact the botnet can have on the Internet. As a consequence, the bots often contain software that automate the task of scanning IP addresses to find vulnerabilities in software. Once a vulnerability is found the vulnerable machines are infected with the bot. And the process repeats itself. It is worth noting that the difference between a bot and a conventional worm is the existence of a unified control system. This brings us over to the Command and Control server which is used by the bot herder to control the zombies. The collection of compromised computers are useless without this control mechanism. The Command and Control server can be seen upon as the interface between the botnet and the bot herder. The bot herder enters command to the server which in turn commands the botnet [52].

2.1.7 Exploits

Definition: *"An exploit is a program or technique that takes advantage of a vulnerability in software and that can be used for breaking security, or otherwise attacking a host over the network."* [45]

A common way to classify exploits is to distinguish between a local and a remote exploit. A local exploit requires prior access to the vulnerable system. Therefore, since the local exploits do not by themselves allow intruders into the system, an ordinary user account is needed first. Usually, a local exploit increases the privileges of the person running the exploit past those given by the system administrator. It is possible to use several exploits, first to get access to the system at a low level, and then use another exploit to get high-level access, like the system administrator has. A remote exploit on the other hand, works over a network and exploits the security vulnerability without having prior access to the vulnerable system. Even though exploits are typically included in easy-to-use tools, it would still be convenient for an attacker to keep the exploit as a stand-alone program in order to be easily organized and accessed [HM04].

Another way of classifying exploits is by the action the exploits take against the vulnerable system. Unauthorized data access and code execution are two examples. There also exist exploits against specific client applications. When the client application accesses the modified server, the server sends an exploit to the client application. Exploits against client applications may also require interaction with the user and thus may be used to trick the user to give away secret information. Techniques used to manipulate people into giving away secret information and performing actions that benefit an unauthorized person is called *social engineering*. In most cases, an exploit only takes advantage of one system vulnerability. A single attacker that makes an exploit may share it with others and in that way increase the probability that the exploit will be used. It is also important to be aware of that white hats, people without malicious intentions, may create an exploit for the purpose of analyzing the level security of a system or program through penetration testing. In other words, the threat of an exploit depends on the intention of the attacker [Szo05].

2.1.8 Ad- and Spyware

Definition: *"Adware is software that produces advertisements on the computer to which it is installed [11]. A spyware is a program that monitors and gathers user information for different purposes [30]."*

Adware is often bundled with other pieces of software, mostly freeware, and is commonly installed without the user's knowledge. It displays its advertisements though pop-up windows, URL redirection or other means. The advertisements are sometimes deliberately made to annoy the user, and there is often an option to buy another version of the software that is ad-free. There are also examples of this "full version" just being a fraud, for instance the freeware might be a fake virus-scanner and the user is told to pay for the full version which is claimed to include virus-removal capability [12]. There are examples of legit usage of adware, where the software simply displays an ad-banner to the user to compensate for offering the software free of charge. Adware becomes a type of malware when it installs itself without the user knowing about it, obstructs normal usage of the computer, contributes to fraud or uses spyware techniques to display personalized ads.

Spyware programs usually run as background processes, unknown and unnoticeable by the user. Spyware can also use known malware techniques as rootkits to further hide and complicate removal. Spyware gathers information about the user, such as web-surfing

habits or personal information, or more crucial information such as keystrokes, financial information or passwords. Both ad- and spyware are known to degrade both network and general system performance, and in some cases can deteriorate it such that the computer becomes practically useless. Spyware and adware usually works together, with the adware using information gathered by the spyware to display personalized ads [11, 30].

A common strategy for installing the ad- and/or spyware is to make the user agree to an End User License Agreement (EULA) where they accept this type of software being installed on their systems. This has lead to a legal issue for anti-malware vendors as it is problematic to classify this type of installations as malware as the users (though mostly inadvertently) have agreed to the installation. There have been examples of spyware creators that have filed (and won) lawsuits against anti-malware vendors that has labeled their software as malicious [24]. This can be one of the reasons why traditional anti-malware vendors avoided this market for a long time and many of the most popular anti ad- and spyware programs are free of charge.

2.2 Zero-day Malware

In Section 2.1, we described various types of malware, which is essential to have an understanding of in our project. The main idea of this experiment is to find zero-day malware and as a consequence of that we have to take a closer look at the term *zero-day*. Then we will describe the vulnerability life-cycle followed by some previous zero-day malware examples.

2.2.1 Zero-day

A security issue can be made known the same day as the computer attack is released. In this case the software developer has *zero* days to prepare for the security breach and must work as quickly as possible to develop a patch or update that fixes the problem. This is why we use the term *zero-day*. And because of this possibility new malware may be difficult to prevent, even if you have anti-virus software installed on your computer. It takes some time for the anti-virus vendors to analyze, make and distribute anti-virus signatures [14].

The term *zero-day* is often used in the context of exploits. An exploit is already defined in Section 2.1.7. The reason why we often talk about exploits when we use the *zero-day* term is that the exploit is the code that take advantage of the vulnerability and that the exploits may be used by viruses, trojan horses, worms or other malicious software to propagate and infect hosts. Web browsers and media players are popular targets since they receive files from the Internet and can have access to system functions [14]. It is important to note that a virus for example, does not need to contain an exploit and therefore may not take advantage of a specific vulnerability. This is why we define both zero-day exploit and zero-day malware as separate concepts. This is our definition of these terms:

Definition: *A zero-day exploit is an exploit used in a malicious computer attack that takes advantage of a security hole before the vulnerability is known to the software developers or before the developers manage to make a patch, even though they are aware of the vulnerability.*

Definition: *Zero-day malware is a type of malware that is able to infect a host because of the absence of malware signatures or other malware detection techniques in the anti-virus program. Zero-day malware does not need to involve a zero-day exploit.*

In Figure 2.2 we illustrate the stages malware goes through, from it is created to the anti-virus software is updated with the specific signature.

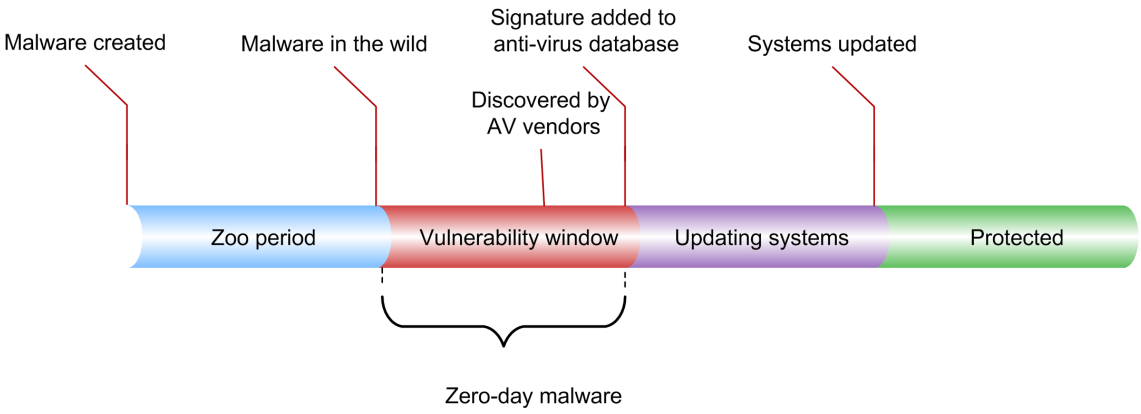


Figure 2.2: Malware timeline, seen from a malware researcher's point of view.

The stage that we have called *Zoo period* in Figure 2.2 is the period after the malware is created but before the malware is considered as being in the wild. *In the wild* means that the malware is spreading on the Internet. The zoo period is the period when the malware is considered as zoo malware. *Zoo malware* is malware which only exists in a closed environment, typically in a lab, and is not spreading on the Internet. Therefore zoo malware can be seen as the opposite of malware that is in the wild. We are not separating between black and white hats when it comes to zoo malware. A possible scenario may be that a white hat creates zoo malware to be able to show his friends. Even though the white hat has showed the malware to his friends it is not considered as in the wild until someone spreads the malware on the Internet.

The next stage is called *Vulnerability window*. In the context of malware, the vulnerability window can be seen from both a system owner's point of view and a malware researcher's point of view. We have chosen to see it from the malware researcher's point of view and define the vulnerability window as *the time between a malware instance, for example a virus, is spread on the Internet to when a patch or signature is released by the anti-virus vendors*. If we had seen it from the system owner's point of view the vulnerability window would not be over when the anti-virus vendors have *released* the malware signature but rather when the systems are *updated*. Because of the closing gap between the time when the attackers find a way to spread malware to when they have the malware ready this makes it harder for the anti-virus vendors to keep their software up to date. The anti-virus vendors' goal is to keep this vulnerability window as narrow as possible. As can be seen in Figure 2.2 the discovery of malware by the anti-virus vendors, called *Discovered by AV vendors* in the figure, has been put relatively close to when the signature is added to the anti-virus database. The reason for this is that it is likely that the time from the vendors discover the malware to when they have a signature ready is short. We

refer to this period as *the signature lag*. As already mentioned, the vendors want to keep their anti-virus software up to date. Referring to our definition of zero-day malware above, malware that is in the stage called *Vulnerability window* in Figure 2.2 is called *zero-day malware*.

It is important to note that we distinguish between the time a signature is added to the anti-virus database by the anti-virus vendor and the time when every system using that anti-virus software is updated with the signature. The reason why we make this distinction is that even after a signature is available it takes some time before all the users have installed the update. We have called this stage *Updating systems*. The following stage called *Protected* is a bit optimistic. We do not try to say that every system is protected after this update but rather is protected against this specific malware.

2.2.2 Vulnerability Life-cycle

In this section we present a life-cycle model that shows the states a vulnerability may enter during its lifetime. First we are considering the number of intrusion during the life-cycle of a vulnerability before we describe the states a vulnerability may enter. At last we will present which states a *system* may be in during its lifetime.

The highest number of intrusions, according to Figure 2.3, is some time after a patch for the vulnerability is released. And the reason why the number of intrusion is increasing even after a patch is available is the fact that it takes some time for the patch to spread and for users to install it. So the attackers will continue to exploit the vulnerability until most of the users have installed the patch. Some attackers even wait for a patch to be released so that they can see what the patch does and then make an exploit that take advantage of the vulnerability before the patch is installed on the host machines. Then, as we can see from Figure 2.3, the number of intrusions is decreasing. Most of the hosts are now updated and there is no point in continuing this particular attack. It is also worth mentioning that attackers often use this patch to make new exploits based on what the patch does not fix and also if the patch itself leads to new vulnerabilities. If we take a look at the time of *disclosure*, we see that the number of intrusions is increasing dramatically after this point compared to the time between *discovery* and *disclosure*. The reason is that even though you assumed that hackers did know about the vulnerability before it was disclosed, you can now be certain that even more hackers know about the weakness and will try to take advantage of it. But we need to highlight that undisclosed weaknesses also are a real danger and a way to protect oneself against attacks targeting vulnerabilities is to apply a heuristic or behavior-based defense.

Usually, two groups of experts research and discover vulnerabilities and create new exploits: Those who are interested in fixing the vulnerable software, and those who are interested in exploiting it. Among those who are interested in fixing the vulnerable software there are also two groups. On one side we have those who believe in telling the vendor about the vulnerability before releasing the information to the public, such that the vendor gets time to fix the problem before everybody knows about it. On the other side we have those who believe that the vendors do not react fast enough when they encounter a problem. They also argue that the hackers probably already know about the vulnerability. So because of this they believe that announcing it is the quickest way to get the vendor to make a patch that fixes the problem. The vendors do not of course like

when this is being announced because it gives them a bad reputation [32] [AFM00].

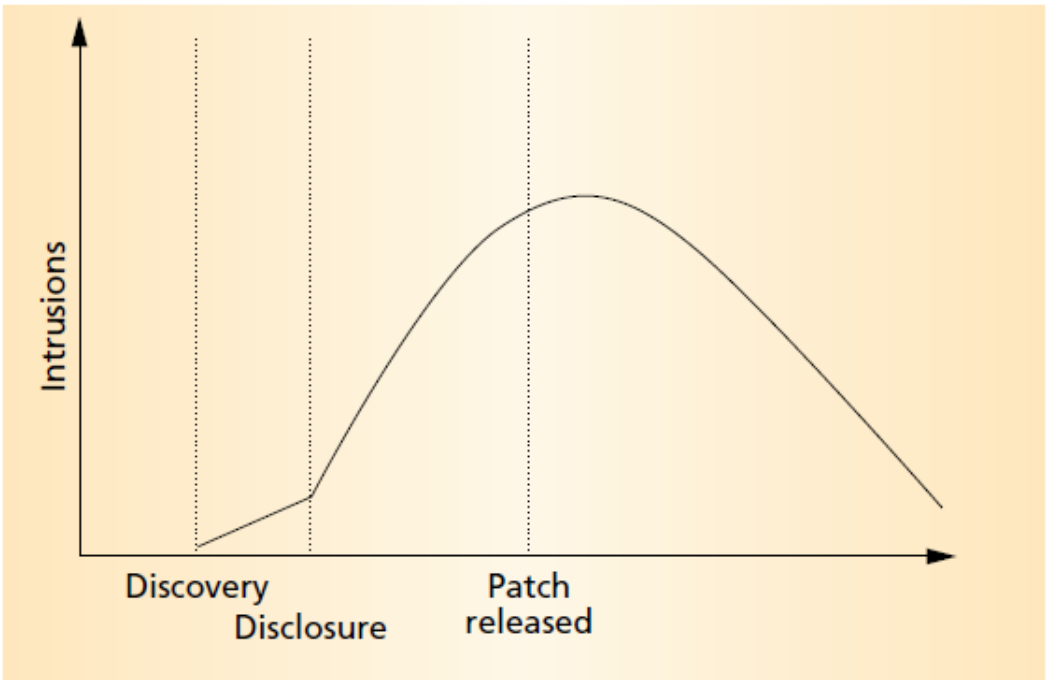


Figure 2.3: A consideration of the number of intrusion during the life-cycle of a vulnerability [AFM00].

To be more specific a vulnerability can be in the following states [AFM00]:

- *Birth:* Birth occurs either unintentionally during development or intentionally by a hacker. When the birth occurs intentionally or maliciously, birth and discovery coincide. When birth occurs unintenionally birth occurs before discovery. Discovery is marked in Figure 2.3
- *Discovery:* This is the state when the flaw really becomes a vulnerability, when someone discovers that the product has security implications. We do not divide between malicious intentions or not.
- *Disclosure:* The vulnerability is disclosed when details about the problem is revealed to a wider audience. Disclosure is illusustrated in Figure 2.3.
- *Correction: "A vulnerability is correctable when the vendor or developer releases a software modification or configuration change that corrects the underlying flaw"* [AFM00].
- *Publicity:* The vulnerability may enter this state if the disclosure gets out of control. In other words, this is when the vulnerability becomes known on a large scale. Likely this leads to more intrusion attempts. This stage thus occurs between disclosure and when the patch is released.
- *Scripting: "This phase applies to any simplification of intrusion techniques that exploit the vulnerability, such as cracker cookbooks or detailed descriptions on how to*

exploit the vulnerability" [AFM00]. This leads to that also unskilled people can exploit the vulnerability. There are several web sites that publish exploits and descriptions on how to use these exploits. An example of such a website is *milw0rm.com*. *Metaploit* is an example of a framework which can be used to produce exploits quite easy. Scripting may occur when the exploit is in the wild, consequently after disclosure.

- *Death:* A vulnerability dies when the system at risk is patched and/or the hackers lose interest in exploiting the vulnerability. It is important to note that death is not when the patch is released but when the majority of the systems have installed the update.

It is important to note that a vulnerability does not need to enter each state, but the first three states need to occur in order. This is logical since a vulnerability needs to be born before it can be discovered. The next four states *may* be entered in any order [AFM00].

Just like a vulnerability, a system can be in different states during its lifetime. As can be seen from Figure 2.4, we distinguish between *hardened, vulnerable and compromised*. Figure 2.4 is based on a figure in [AFM00] but with some modifications since the figure illustrated in [AFM00] is incomplete.

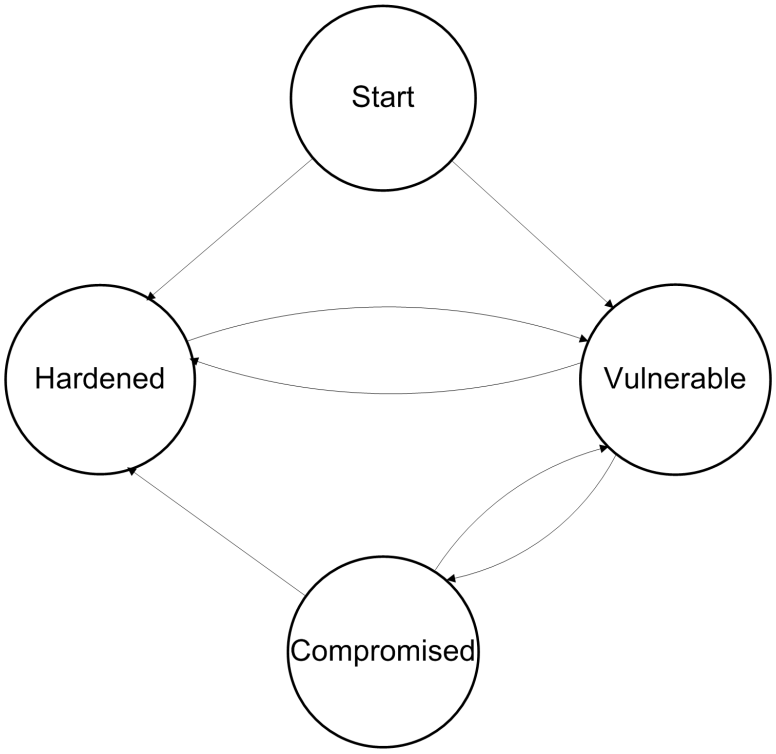


Figure 2.4: System life-cycle.

A system is in the *hardened state* when all security-related corrections, for example patches, have been installed and in the *vulnerable state* when at least one security-related correction has not been installed. When a system has been successfully exploited it enters the *compromised state*. We can see from Figure 2.4 which states that are connected. The state labeled *Start* illustrates the birth of the system. When a system is created it can either

enter the hardened state or the vulnerable state, depending on whether a vulnerability is found right after the system is created or not. If a vulnerability is found right away, the system will enter the vulnerable state. There are potentially many attackers that tries to find weaknesses with a *new* system. But it is more likely that a system first enters the hardened state before it possibly enters the vulnerable state. This is because it is unlikely that a vendor releases a system with known vulnerabilities. Note that it is not mentioned how long a system stays in a specific state. When you are in the vulnerable state you can end up in either the hardened or the compromised state, depending on whether the system is patched before it gets compromised or not. From the compromised state a system can enter the vulnerable or the hardened state. When a system is compromised this may lead to the discovery of new vulnerabilities and therefore lead the system to enter the vulnerable state. If the vendors release a patch for the specific vulnerability and no new vulnerabilities are discovered, the system enters the hardened state. The goal of the system managers is to reduce the time a system is in the vulnerable and the compromised state [AFM00].

2.2.3 Examples

This section presents two examples of previous zero-day malware. The first is exploiting a zero-day vulnerability, while the second one is not directed towards any special vulnerability, but still it managed to infect a lot of computers before anti-virus vendors noticed it at all.

2.2.3.1 Vulnerability in Windows Metafile Files (WMF)

In December 28, 2005, F-Secure reported that a zero-day vulnerability related to Windows Metafile Files (WMF) was exploited [58]. At that time, several trojan horses exploiting the vulnerability were available on the Internet and no patch or fix existed. F-Secure reported the following threats:

- Trojan-Downloader.Win32.Agent.abs
- Trojan-Dropper.Win32.Small.zp
- Trojan.Win32.Small.ga
- Trojan.Win32.Small.ev

According to the Danish security company, Secunia [48], the vulnerability was caused due to an error in the handling of Windows Metafile files (.wmf) containing specially crafted SETABORTPROC *Escape* records. Such records allow arbitrary user-defined functions to be executed when the rendering of a WMF file fails. Opening a malicious WMF file in *Windows Picture and Fax Viewer* or previewing it in explorer would cause the code to execute. More critically, just by visiting a web site that had an image file containing the exploit, you could get infected. With Microsoft Internet Explorer users automatically got infected, while most versions of Firefox and Opera prompted the user for where to open the image file.

2.2.3.2 Rustock.C rootkit

In May 2008, the Russian security company Dr. Web announced that its experts had detected a new rootkit called Ntldrbot or Rustock.C [Rus08]. According to Dr. Web, the

rootkit had been in the wild and undetected since October 2007, eluding all anti-virus vendors.

It was suggested that Rustock.C was used to create one of the largest botnets for sending spam, but this has been partly rejected during later analysis done by Kapersky Lab [27]. Dr. Web obtained a sample of Rustock.C in March 2008, but it took them over a month to analyze and create tools for detecting and removing the rootkit. The author of the rootkit had included sophisticated polymorphic self-protection which made its extraction and analysis extremely difficult. Although Rustock.C was not using a zero-day exploit or some other unknown vulnerability, it still falls under our definition of zero-day malware. The rootkit was in the wild in the end of 2007, but no anti-virus program detected it until March 2008.

2.3 Infection Sources

The enormous increase in Internet usage has made it *the* primary source for malware infections, in this section we will present the most common ways users get infected with malware over the Internet.

2.3.1 The Web

The web browser has become the most common tool for Internet usage, and it is no surprise that malware creators are using the web as an attack vector. Web delivered malware can be divided into two main categories: various social engineering techniques and so called drive-by-downloads [PMRM08].

Social engineering techniques are designed to deceive the user into downloading malware by various methods. The most traditional way of doing this is to make the user download files containing malware, this can be achieved by naming the malware as something else or including some kind of malware in a piece of software. Other, more advanced, techniques include tricking the user into installing fake browser plug-ins or Active-X controls. A common way of doing this is to claim that the user needs to accept the download to be able to view some special content, for example videos, on the web page. Another trick is to claim that the user is infected or unprotected against malware, and make him install some kind of fake anti-virus software, which can contain various types of malware.

The second technique has become more common in the later years, and does not rely on any kind of user interaction except from browsing the web. This is achieved by exploiting vulnerabilities in the web browser or plug-ins, like for instance Adobe Flash or Windows Media Player. The most common attack vector is to use Javascript attacks on the browser itself, these scripts are often hidden in zero-pixel iframes which makes them invisible on the web page. While there is an increased risk of infection from sites containing *gray content* (Pornography, pirated software and similar) compared to *normal* pages, there is still a risk of infection from these pages as well. The malicious code can be added by hackers to compromised servers, or the code can be injected into web sites that allow user contribution (Comment fields, forums or blogs). Another method, that has seen increased popularity, is to use infected advertisements as an infection vector. The increase of this method can be attributed to so called syndication, where advertisers rent out their

advertising space to other parties, which in turn can rent it to another, and so on. This method is especially powerful as even trusted and protected web servers can be used for drive-by-downloads [PMRM08].

As an example, the Norwegian home page of MSN[1] hosted an infected Honda advertisement. The advertisement exploited a vulnerability in Adobe Flash and installed a trojan on the user's computer when the ad was displayed. It was estimated that thousands of users were infected, as MSN is Norway's second most popular web site, and is the default start page of all new Windows PCs [34].

2.3.2 File-sharing

File-sharing networks are known to be a significant source of malware [SJB06], and therefore it is important to expose the experiment to these networks. According to a recent survey [6] the most popular network is the Gnutella network with networks based on the BitTorrent protocol as number two, see Figure 2.5. These two networks combined have almost 70 percent of the total peer-to-peer market share.

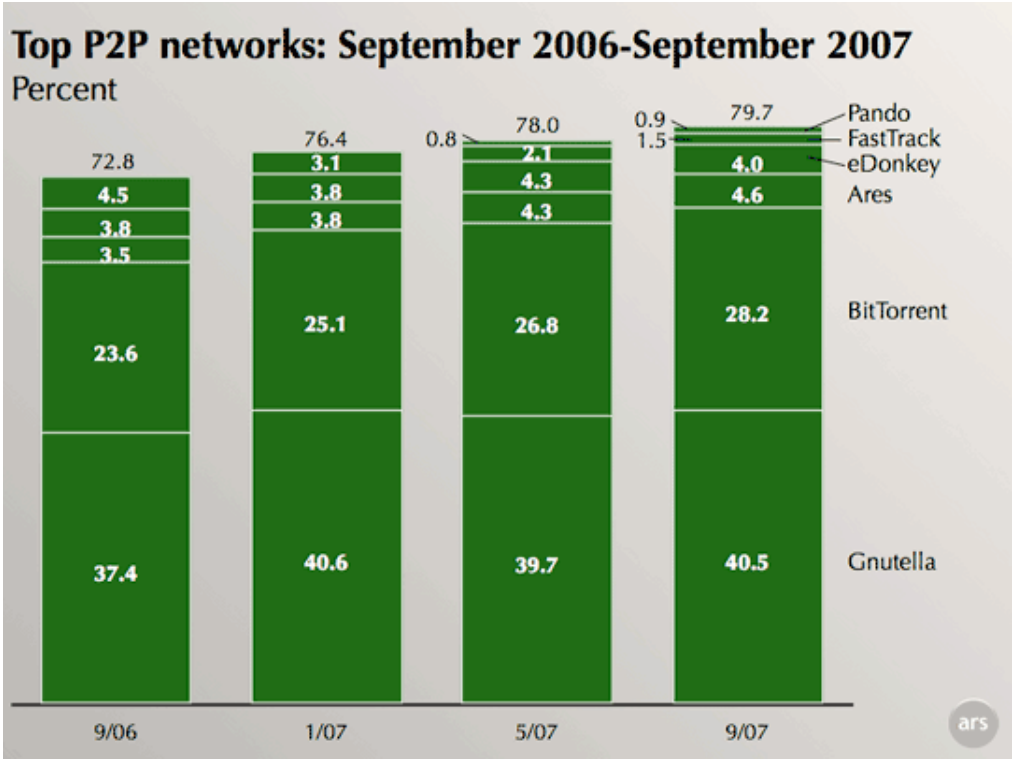


Figure 2.5: Top P2P networks: September 2006-September 2007 [6].

There are several reasons why malware is so widespread in file sharing networks, the most obvious is that there is no or very limited control of the files that are shared on such networks, compared to for instance web sites like download.com. There are also few possibilities of checking the legitimacy and origin of the files. Files are not necessarily what they claim to be, and file names can be misleading. For example a user can be tricked into downloading a file that claims to be a piece of music or video, but actually is an

[1]The Norwegian home page of MSN: http://www.msn.no

executable: A malicious person can share a piece of malware and call it something like *bob-dylan-blowing-in-the-wind.mp3.exe*, a simple trick like this can be enough to fool a novice user into downloading and executing the file. Another trick is to name the file as a popular piece of software, and in that way fool the user to download and execute the file. Files are also often packed in archives to avoid immediate detection by anti-virus software.

When a user gets infected with this type of malware, it can spread further in the network, the primary reason for this is that files downloaded are often automatically shared on the network. The malware can also duplicate in the *Shared folder* and rename itself to match popular search strings. This behavior can be categorized as a hybrid between a virus and a worm. In addition the file will also most likely contain some other form of malicious payload like a trojan or rootkit.

Another attack vector can be the client software itself, it is quite possible that there exist vulnerabilities that can be exploited by an attacker. Especially since this software accepts multiple in- and outgoing connections to other hosts on the Internet. Also the client can contain malware, mostly ad- and spyware, that is intentionally put there by the developers. The client Kazaa is especially infamous for this, as it contains seven different malware installations [25] even though it claims to have *"NO Spyware" [44].*

This section is a somewhat detailed study of file-sharing networks, with special focus on the Gnutella network. The reason for this thoroughness will be eminent when the results of our experiment are presented and discussed in Section 6.1.2 and 7.6 respectively.

2.3.2.1 Limewire and the Gnutella network

Limewire is a Java-based file-sharing application which is available for several platforms [38]. It is the most popular software for use with the Gnutella peer-to-peer network. Gnutella is based on the protocol with the same name, and is according to the Gnutella protocol specification *"a protocol for distributed search and digital distribution"* [37]. The Gnutella protocol also supports a more traditional client-server model, but the model used by the Gnutella network is a peer-to-peer, decentralized model. In such a model all participating nodes acts as both client and server, so called *servents*. The network servents can take one of two roles in the network; leaf node or ultrapeer node. These differ in the number of connections they have to other peers in the network.

Note that the Gnutella protocol specification [37] is Wiki-based, and is lacking or indecisive on some points. There also exsists a draft specification [35] which the mentioned protocol specification is based on, but this is from June 2002 and since then there has been many changes and additions to the protocol. This fact makes it difficult to get a comprehensive view of how the network functions.

When a servent wants to connect to the network it has to locate other servents in the network, this is called bootstrapping. This is can be done in several different ways, the most common are:

- Trying a predefined set of adresses that shipped with the software.
- Querying by using GWebCache, which is a part of the Gnutella protocol that allows servents to get addresses of other servents by sending a HTTP request to a

GWebCache server.

- Using User Datagram Protocol (UDP) host caches. The use of UDP-based host caching is the newest and preferred method of doing host-discovery and is based on Gnutella ping/pongs over UDP. Pings and Pongs in Gnutella works similarly as in Internet Control Message Protocol (ICMP), but with some differences. A Ping in Gnutella is sent between ultrapeers to send information about itself and to request a Pong message. A Pong message contains information about the ultrapeer and other ultrapeers it is connected to. The Pong message is forwarded to the leaf node so that the leaves has a list of addresses of other ultrapeers it can connect to.

Once a connection is made to one other servent, addresses of other servents will be sent over the network. When connecting to another servent, a proper protocol handshake has to be performed. This handshake is described in the Gnutella Protocol Specification [37].

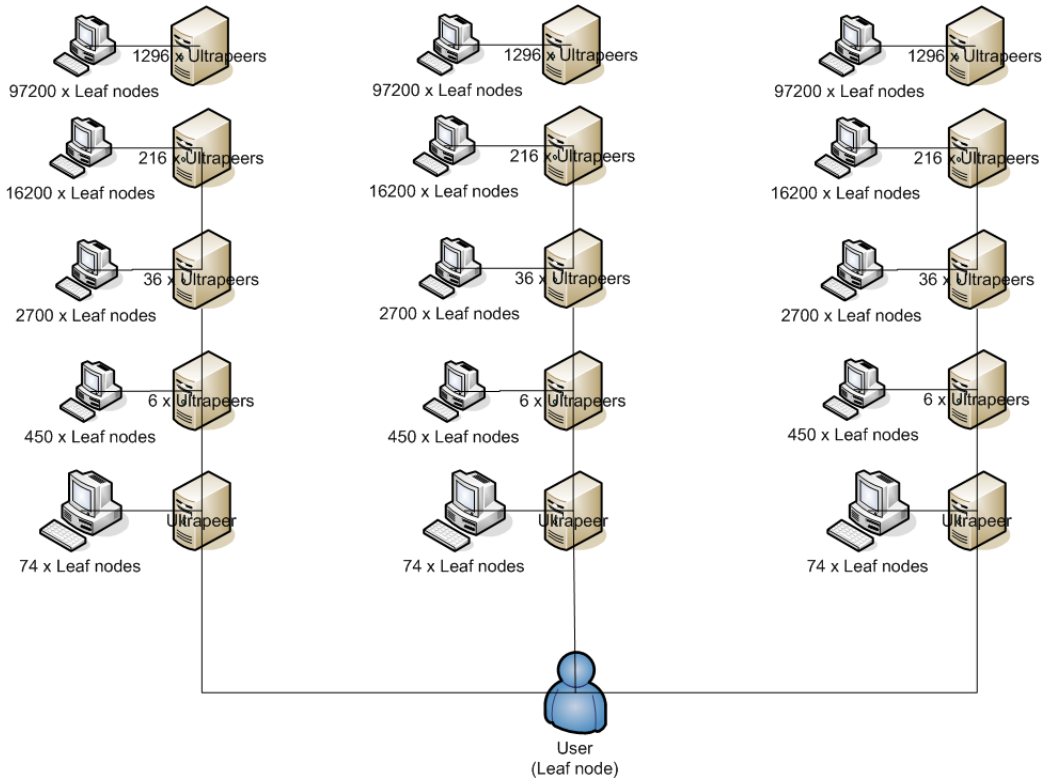


Figure 2.6: A sketch of the topology of the first five hops in the Gnutella network as seen from a leaf node.

A leaf node typically connects to three ultrapeers and an ultrapeer typically connects to six other ultrapeers and 75 leaf nodes [36]. This means that a servent can connect to quite a large portion of the network in only a few hops. The maximum number of hops is restricted to ten, but it is not recommended that the number exceeds seven. The number of hops is set in a Time To Live (TTL) field in the header of the messages sent, this number is decreased by one for each hop and the message is discarded when this field reaches zero. See Figure 2.6 for a sketch of the network, note that both a leaf node and an ultrapeer can exist several places in this figure because of the unsymmetrical nature of the connections, in other words loops exist in the network. Note also that the number of connections

per leaf node and ultrapeer can vary, so the numbers in the figure are just examples to illustrate the growing number of peers as queries go deeper into the network. This means that the same network traffic can arrive at the same node from different connections, such messages are discarded and not forwarded. This also means that the network is in fact smaller than what the figure suggests. Connecting as a leaf node allow users with low bandwidth connections, typically dial-up users, to connect to the network without using a large share of their available bandwidth to relay other network traffic like searches. It is up to each node to decide if it want to act as an ultrapeer or not, this is decided based on available bandwidth, uptime, not being behind a firewall and other criteria. If these are met then the node can become an ultrapeer if needed in the network.

When a user initiates a search for a file from his servent, e.g. Limewire, a Query message is sent to all its directly connected servents. A servent will then forward the Query to all its direct connections, except to the originator. If the servent is an ultrapeer the Query is only forwarded to the leaf nodes which most likely has a match. This decision is taken based on hash tables generated by the Query Routing Protocol [37, Chapter 3.8]. The Query continues to travel through the network until the TTL field in the header reaches zero. When a servent receives a Query it also checks if it matches any of the files that it shares, if it does it responds with a Query Hit message back through the same network path. The file transfer is then initiated directly between the servents using standard HTTP 1.1 file transfer.

A servent will not be able to reach the entire network with its Query messages, because of the TTL limitation. Because of this, the search results you get depend on what peers you are connected to. A result of this is that two different computers running the same software at the same time and searching for the same keywords can get very different search results. The Gnutella network had a total population of 1.81 million computers as of June 2005 [Bha08], and is most likely much larger now [6]. The large size of the network makes it infeasible that one single participant in the network is within seven (or ten) hops of the entire network, this limit is called the Horizon of the servent [36].

2.3.2.2 BitTorrent

BitTorrent is a protocol designed to distribute large amounts of data, where the original distributor, the *seed*, sends pieces of the the total file to downloaders, called *peers*. When the peers have obtained a piece they start uploading the piece to other peers, creating a peer-to-peer based distribution network or *swarm* [10]. The sharing of pieces is done using a *rarest-first* or random algorithm to increase the chances of peers downloading different pieces, and hence be able to share the pieces between them. Compared to traditional server based distribution this reduces the resources required significantly and could also increase the download speed for the end-users. BitTorrent has seen significant increase in popularity in the recent years, and was the second most popular file-sharing technology on the Internet as of September 2007 [6], second only to the Gnutella network.

BitTorrent combines traditional web servers to host a *torrent* (A metainfo file containing information about the file and the tracker, typically named *filename*.torrent) and peer-to-peer based applications to download the wanted file (or files). To share a file using BitTorrent, several things are needed [10]:

- One or many *tracker(s)*, which is a server that holds information about the torrent and connected peers. It is also possible to have tracker-less torrents using distributed hash tables, but this is less common, and will not be discussed here.
- A web server to host the .torrent file. This is typically a torrent search engine, called *BitTorrent index*, like for instance *thepiratebay.org* or *btjunkie.org*
- A downloader application which already has the complete file, the seeder.

When a user wants to download a file he will obtain the torrent file from a web server. Then the user must open this file in his preferred BitTorrent client and the downloading (and eventually uploading) will begin. When the file is complete the file will usually, depending on the client and settings, continue to upload the file to other peers until the torrent file is removed from the client or the client is exited. This means that the peers become seeds when the download has completed.

BitTorrent differs in several ways to traditional file-sharing networks, like Gnutella or Napster. The largest difference is that there is no *BitTorrent network* in which all files are shared. In stead a network or swarm is created for each torrent file being downloaded. This also means that there is no way to search the *network* for shared files, in stead the user has to obtain the .torrent file from a separate source, like the web. Another difference is that since torrent files does not contain the actual file, trackers or torrent indexers do not host any potentially illegal or copyrighted material. This means that these types of servers are legal in most countries, even if they host torrents that are being used to illegally download files.

2.3.3 Spam

E-mail is the most common infection source for malware, accounting for 27% of malware infections overall [Lev08], and over 90% of this malware is coming from known spam sources, mostly botnets. In other words there is a strong correlation between spam and malware. Spam-based malware is based either on attachments, like executables or infected documents, or links to malicious web sites.

Links to malicious web sites can be made in such a way that they look like legitimate web sites, this is called *Link Manipulation*. Many techniques to do this exist, ranging from simply changing the link's anchor text, to using vulnerabilities in the e-mail client or web browser making it display another URL than the one the user actually connects to [39].

It was originally planned to use spam as an infection source, but due to time constraints this was not done, see Section 1.4 for further details.

2.4 Malware Detection Techniques

This section is a collection of detection techniques used in traditional anti-virus software. Because of the various techniques, not all anti-virus engines will search for the same elements, which in return can lead to different search results. This part serves as a background study for the discussion of problems with current anti-malware software, which is conducted in Section 7.5.

There are relatively few sources that describe exactly how commercial anti-virus solutions work. One reason for this is the highly competitive industry among anti-virus vendors. Another reason is the battle between anti-virus software developers and malware creators. These people will use any knowledge of anti-virus techniques to defeat it. Although the described techniques have been around for a long time, they are still regarded as the core of anti-virus software. Most of the work in this section is based on [Szo05], a thorough presentation of how anti-virus engines work.

2.4.1 First-Generation Scanners

One of the most basic malware detection techniques is known as signature scanning. A signature is a characteristic sequence of bytes that is part of a certain type of malware, but not likely to be found in clean programs. Since such a byte pattern possibly can occur in ordinary programs, false positives are a challenge. One technique for speeding up the string matching process is to only scan the beginning and the end of files, but it should not come at the sacrifice of detection accuracy. Another speeding technique is hashing.

2.4.2 Second-Generation Scanners

Smart scanning skips instructions like *NOP* (no operation) or other junk instructions inserted in the source files of a program. Such instructions can cause a virus to look very different compared to its original. By omitting these values from the virus signature, this problem is reduced. The same technique can also be used with malware appearing in textual formats, such as script and macro viruses. Characters like extra white spaces etc. can be dropped in order to enhance the scanner's detection capabilities.

There are several techniques that takes this approach one step further. *Skeleton detection* are especially useful in detecting macro viruses. The scanner drops all non-essential statements, resulting in a macro body with only essential code to use in detecting viruses. *Nearly exact identification* typically uses double-string detection, instead of one string, in order to detect malware more accurately. *Exact identification* is usually combined with first-generation techniques to guarantee precise malware identifications. It uses a checksum of all constant bits of the malware body, which then are covering a longer area of the malware body than the previous techniques.

2.4.3 Algorithmic Scanning Methods

Algorithmic scanning, also called *virus-specific detection algorithm*, uses a scan definition language to define detection routines for individual malware. These routines are then executed on the objects opened by the scanner. Another technique for more efficient searches, is filtering. The idea is that malware typically infect only a subset of known object types. A boot virus signature can typically be limited to boot sectors, EXE signatures to their types, etc. By setting an extra flag in the detection routine, it can be indicated whether or not the signature in question is expected to appear in the object being scanned.

2.4.4 Behavioral Analysis

Behavioral analysis is an approach to detect unknown malware, this scheme relies on detecting malicious behavior of the malware when it is executed. This approach requires

that the file is opened or executed, and then monitor the behavior and block potential malicious actions performed. To avoid infecting and damaging the computer, the file is run in a protected sandbox or on a virtual machine which is separated from the normal operating system.

2.4.4.1 Sandboxing

Sandboxing is a security mechanism for safely running programs that are not trusted, for instance files downloaded from the Internet. Instead of being exectuted on the actual running system, cages or virtual subsystems are introduced. The idea is to let untrusted programs run on a virtual machine that has access to the same information as a user on a local machine, or in a cage where access to vital system files and settings are restricted. The malware can be made to believe it has access to a normal system, but no modifications are actually performed on the running system. Changes in system files and in the registry will only be applied within the cage or virtual machine, and the running system is therefore left unharmed. Malicious-looking actions can than be monitored and logged, the anti-virus software can then label the file as malware and remove it from the system.

2.4.4.2 Code Emulation

Code emulation is kind of a minimal sandbox environment, where a virtual machine is implemented to simulate a CPU and memory system. By executing code in this environment for a short period of time, any encrypted code will be decrypted and written to memory or disk. It can then be scanned using an ordinary signature scanner. The biggest problem with this approach is performance. Executing only one file in this emulation environment works great, but performing a full system scan will be too time-consuming.

2.4.5 Metamorphic Virus Detection

As metamorphic viruses can change both behavior and appearance, it is not easy to base detection on signatures. Other techniques must be used, such as examination of the file structure or the code stream, or analysis of the code's behavior. Many techniques exist to detect metamorphic malware, a good example is heuristics, presented in the next section.

2.4.6 Heuristics

In the anti-virus industry, the term *heuristics* is used to describe a specific type of virus detection which analyzes the program's structure, its behavior, and other attributes instead of looking for signatures [Cor02]. In order to analyze behavior, heuristic scanners need to have access to the code of the malware. To achieve this, heuristic engines need to be combined with other techniques that can unpack packed code, and decrypt if needed. Another possibility is to let the heuristic engine monitor executing threads and track their operations on the system.

Heuristic scanning can be very effective, especially against new viruses where no signature exist, but the risk of false positives is high. Reducing the capability of heuristic analyzers to a level where the number of false positives is low, while still able to catch a reasonable number of viruses is a difficult task.

2.4.7 Memory Scanning

As described in Section 2.1.1, viruses that load in main memory and remain active even after an application has terminated, is called *Terminate and Stay Resident (TSR)*. Such viruses has the potential to hide itself from scanners by using stealth techniques. Removal of such a virus is hard, as it can infect previously disinfected objects again and again. In order to detect and remove such malware, memory scanners are required. TSR malware can run as kernel modules or device drivers and so should the memory scanner, to be able to access all relevant memory pages.

Chapter 3

Laboratory Environment

The laboratory equipment was installed in the SISLab offices of SINTEF ICT in Trondheim. The experiment consists of six computers, plus an additional machine, *Jerry*, which was used for testing and other work which would be unfortunate to do on the actual experiment computers. An overview of the hardware and software of the computers can be found in respectively Table 3.1 and Table 3.2. To save space the computers were set up in pairs with a KVM-switch, this meant we only needed one keyboard, monitor and mouse per pair of computers. A picture of the lab can be seen in Figure 3.1.

Figure 3.1: The LAB environment at SINTEF ICT.

3.1 System and Network Overview

The computers were connected to a separate Virtual Local Area Network (VLAN), which is not behind any form of firewall. This put our machines directly connected to the Internet, and was done to further increase the chances of infections. Another reason for this separation was to avoid having the computers on the same network as other machines at SINTEF. The machines in our experiment would most likely be infected with various forms of malware and could therefore be a security risk for the rest of the SINTEF network. The network topology in our laboratory is illustrated in Figure 3.2.

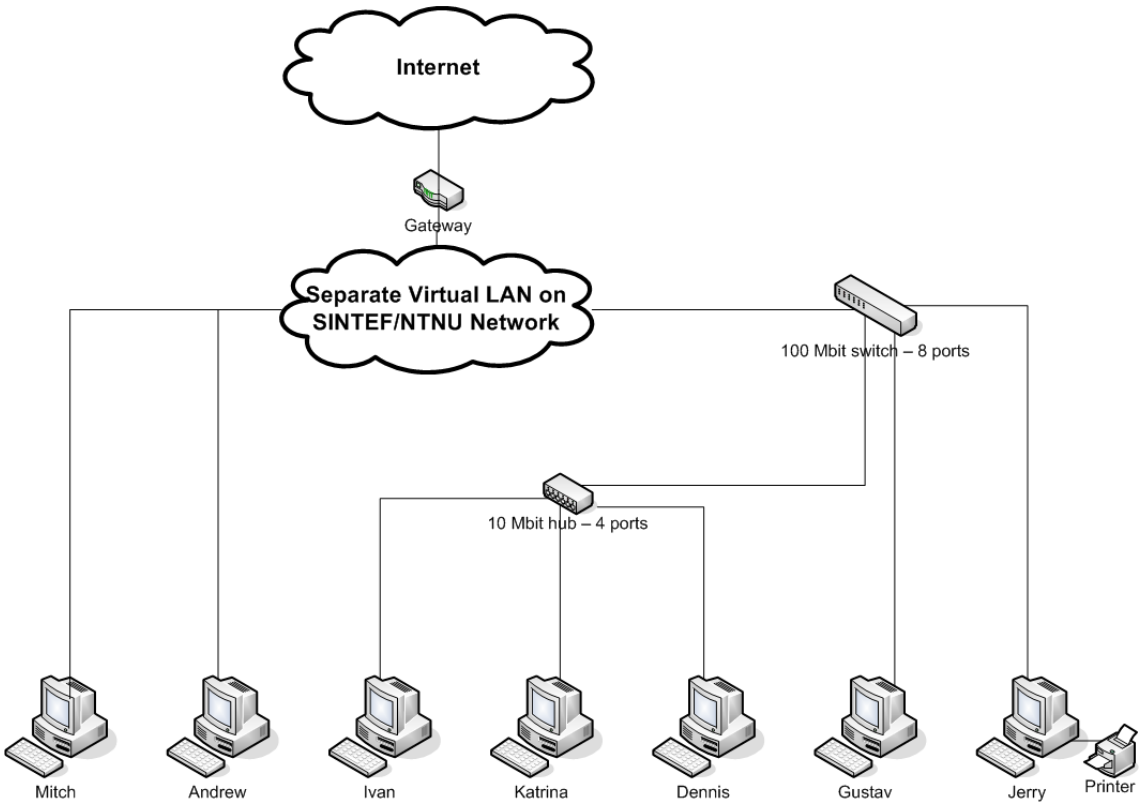


Figure 3.2: An overview of the network in the laboratory.

3.2 Hardware

The lab consisted of seven computers with relatively old hardware, the hardware configuration is summarized in Table 3.1. The most powerful computer, *Dennis*, was chosen to have Windows Vista installed as this operating system has higher hardware requirements than Windows XP. *Jerry* was the machine with the most memory installed, and was therefore used as a test machine outside the experiment because it would run several tasks simultaneously, and to make the other machines as equal as possible.

Computer name	**System type**	**Processor**	**RAM**
Dennis	Dell Dimension 4550	Pentium 4 2.53 GHz	1 GB
Gustav	Dell OptiPlex GX260	Pentium 4 2.00 GHz	512 MB
Ivan	Custom built	Athlon XP 1.47 MHz	512 MB
Katrina	Custom built	Athlon 1.4 GHz	512 MB
Mitch	Dell OptiPlex GX260	Pentium 4 2.00 GHz	512 MB
Andrew	Dell OptiPlex GX260	Pentium 4 2.00 GHz	512 MB
Jerry	*Custom built*	*Athlon 1.4 GHz*	*1.25 GB*

Table 3.1: Computer hardware specifications.

3.3 Software

In addition to different anti-virus software, we also installed anti-spyware software and file-sharing programs. An overview of the anti-virus software installed on the computers is listed in Table 3.2.

Computer name	**Operating system**	**Anti-virus**	**Anti-spyware**
Dennis	Windows Vista	F-Secure	Spybot Search & Destroy
Gustav	Windows XP	Norman	Spybot Search & Destroy
Ivan	Windows XP	Norton	Spybot Search & Destroy
Katrina	Windows XP	F-Secure	Spybot Search & Destroy
Mitch	Windows XP	Avast!	Spybot Search & Destroy
Andrew	Windows XP	AVG	Spybot Search & Destroy
Jerry	*Windows XP*	*F-Secure*	*Spybot Search & Destroy*

Table 3.2: Computer software overview.

3.3.1 Operating Systems

The computers in the lab were installed with Windows XP except for one machine which was installed with the newer operating system Windows Vista. The computers were installed with the latest service packs, SP3 for Windows XP and SP1 for Windows Vista, from CD. The computers were not connected to the Internet until proper anti-virus software was installed. When the machines were connected to the Internet the first thing we did was to install the latest updates from Windows Update.

Windows Firewall was disabled to increase the chances of infection from network worms and other network-based threats.

3.3.2 Anti-virus

Three different types of anti-virus products were obtained for use in the project. Desktop anti-virus software was continuously running on all the machines in the laboratory, while the two offline products Avast! BART and F-PROT were only used at the control, baseline and final scans in the experiment. Finally, the online anti-virus service VirusTotal was used when analyzing malware in the end of the project.

3.3.2.1 Desktop Anti-virus Software

Each machine in the laboratory was set up with different anti-virus software, except for the machine running Windows Vista which had the same software as one of the Windows XP machines: F-Secure Anti-Virus [22]. All the software was chosen so that we should not have to pay to obtain it. SINTEF provided us with a licensed copy of Norman Security Suite [4] and from NTNU we have access to licenses to F-Secure Anti-Virus. We also chose the two most popular free anti-virus packages: AVG Free Edition [56] and Avast! Home Edition [2]. For the last anti-virus program we contacted Symantec [13] and asked for them to provide us with a free license to use in the project. We got a copy of Norton Internet Security 2008 free of charge. These five software packages covers both popular free software and well known commercial software, and offer a wide protection and a good

possibility of preventing the machines from being infected by ordinary malware.

All the anti-virus software was installed from CD, and the computers were not connected to the Internet until they all had anti-virus software installed. This was done to avoid infection from network worms before the actual experiment was started. When the anti-virus software detected an Internet connection they all automatically updated themselves with the latest virus definitions. All the software was installed with default settings, but some changes were made to make the different anti-virus software as similar as possible. Norton Internet Security 2008 comes with a built-in firewall, but this was disabled. All the anti-virus software was set to quarantine infected files if possible.

3.3.2.2 Offline Anti-virus Software

In addition to desktop anti-virus software, we obtained two offline anti-virus programs which could be run from a CD. Avast! has a product called BART (Bootable Antivirus & Recovery Tool) which can be obtained from the the homepage of Avast! [1]. By filling out a request form, you get a 14 day trial license.

The second offline tool was F-PROT [31], a popular and free anti-virus package available for Linux and other systems. To run the program we used Knoppix version 5.1.1, a Live CD Linux distribution, but other distributions could also have been used. By using these two offline anti-virus products, we were able to scan all the computers in the laboratory without booting the host operating system. In this way, malware that is using various hiding techniques could also be detected. It was these two programs which created the basis for detecting zero-day malware, when comparing the results from the baseline scan and the final scan in our project.

Instructions for obtaining and using both Avast! BART and F-PROT is given in Appendix B.

3.3.2.3 VirusTotal

"VirusTotal is a service that analyzes suspicious files and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines" [62].

VirusTotal is a free online service, developed by an independent IT Security laboratory called Hispasec Sistemas, where you can upload your files and search them for malware by the use of 36 different anti-virus engines. Well-known anti-virus software like F-Secure, Symantec, Norman, Norton, AVG, Avast! and F-PROT are included. The different anti-virus engines are updated regularly with official signature files that are published by their respective developers. When uploading and scanning files using VirusTotal, specific results from each of the anti-virus engines will be presented. VirusTotal also presents real-time global statistics based on their malware findings [62].

3.3.3 File-sharing Programs

File-sharing networks are known to be a significant source of malware [SJB06], and therefore it is important to expose the experiment to these networks. File-sharing networks are further explained in Section 2.3.2.

For the Gnutella network we chose the client Limewire [38], which is the most popular client for this network [6]. Limewire is free and open source and is written in the Java programming language. Limewire was installed with default settings, but sharing of files was disabled to avoid any legal issues and excessive network traffic.

The most popular client for BitTorrent is μTorrent [60] and is well suited for our laboratory as it is light-weight and easy to use.

3.3.4 Anti-spyware

We installed additional software to protect against spy- and adware. We decided to install Spybot Search & Destroy [40] on all machines. Spybot Search & Destroy is a popular and free ad- and spyware detection and removal tool. While we did not originally intend to have this kind of software installed, it became apparent that it was needed to avoid the machines to become so cramped up with spy- and adware that the machines would be practically unusable for the intended activity. The decision of installing this kind of program is further described in Section 4.

3.3.5 Hex Comparison

We installed a 15-day trial version of a hex editor called HexCmp2 [23], version 2.34, to compare downloaded files to see if they were the same. Many of the downloaded files had different names but the same size. We refer to Section 6.3.1 for a more detailed description on how we used HexCmp2.

Chapter 4

Pre-study

This chapter describes some experiences we did early in the project, which made us change some aspects of the initial plan. A few things were in fact realized during the experiment, but they are emphasized here, so the reader is aware of those choices when reading the next chapter: Experiment Procedure.

4.1 Kazaa

Initially we were planning to use the file sharing applications LimeWire and Kazaa. These choices were based on studies about malware in peer-to-peer networks [KAG06, SJB06]. We installed both applications on our test computer *Jerry* in order to get some experience before using them in the real experiment. We were aware that Kazaa might come with some additional spyware, but it turned out that seven programs were bundled with the application [25]. This caused our computer to use a lot of memory and normal operations took forever. While doing some more research about file sharing networks, we discovered that Kazaa and its network, FastTrack, are in fact not among the most popular [6]. We did not see any reason to keep Kazaa, and decided to replace it with a popular file sharing network, BitTorrent. It is likely that zero-day malware will turn up faster in a popular network than in a smaller and decreasing network.

4.2 Spybot Search & Destroy

While surfing the web and visiting malicious sites, as described in Section 5.2.3, we encountered malicious software known as Spyware/Adware. The very first choice was a web site found in the search engine Yahoo! and had a warning saying *Dangerous Downloads*. A javascript exploit executed some pop-ups and after enabling ActiveX most of our computers suddenly had installed a fake anti-virus program (AntiVirus Lab 2009). This resulted in a massive wave of pop-ups and warnings, and we were encouraged to pay for anti-virus protection. It was impossible to visit more web sites and we had to clean the computers in order to continue.

After this incident, we realized that additional software to protect against such spy- and adware threats was necessary. Spybot Search & Destroy [40] was installed and was run when needed.

4.3 Collection of Files in LimeWire

When searching for files in LimeWire, you do not always get the same results, even though the search keyword is the same. This phenomenon occurs due to the structure of the Gnutella network and is described in Section 2.3.2.1. This was also experienced when we executed the same queries on each of the computers in our laboratory, but received very different list of files to download. The primary reason for this is that the clients are connected to different nodes in the network, and thus their queries travel different paths.

Despite this fact, we still believed that when comparing the downloaded files, we would surely find some popular files that existed on every computer. Unfortunately, we were wrong. In most cases, there were always a few computers missing the file. Because of this, we decided to collect all of the downloaded files from LimeWire on each of the computers, into one folder. We did this by sharing the download folder on each of the six computers and used the seventh PC, *Jerry*, to assemble all of the files from the six computers. It is worth mentioning that we disabled the anti-virus program on Jerry while doing this, in order to prevent it from removing any of the files. This big pool of files were then shared, and every one of the six computers in the project could now copy this folder back. Now the computers had the same files and it was an easy task to choose some selection of files to install.

4.4 Risks

When working with computers, and especially since we aim to be infected with malware, there are some risks attached. One obvious risk is the possibility that a computer gets so seriously infected, that it could stop working. Computers can also fail due to power failure or other hardware problems, but with six computers in our lab, we did not expect this to be a major problem.

Another important risk is that of unintentionally breaking the IT regulations at NTNU[1] or even committing an action that is illegal according to Norwegian law. In order to download popular files, it is necessary to perform actions that could result in abuse messages, but we have tried to minimize this risk. In LimeWire we disabled the sharing functionality, and in μTorrent we adjusted the upload speed and number of connection slots to a minimum. We realize this is not an optimal or desirable situation, but in order to carry out our experiment, it is hard to avoid.

[1]IT Regulations at NTNU: https://sikkerhet.ntnu.no/doc/NTNU-IT-regulations.html

Chapter 5

Experiment Procedure

In this chapter the test procedure for carrying out the experiment is described. First an outline of the procedure is given, before going into details regarding Gnutella, BitTorrent and the web. Finally, Section 5.3 describes how the anti-virus software was used during the experiment. This includes the desktop anti-virus software, the offline anti-virus CDs and the methodology used to determine zero-day malware from all the different anti-virus logs.

As the time schedule in Table 5.1 shows, we actively exposed the computers to suspicious web sites and file-sharing networks during a period of two weeks. The computers were then shut down for about a month, before they were turned on in the beginning of November to perform anti-virus scans and analyses.

September 10th:	All computers in the laboratory were connected to the Internet.
September 15th:	Control malware scans and experiment start-up.
October 1th:	Baseline malware scans before computers were shut down for a month.
November 3th:	Final malware scans.

Table 5.1: Time schedule for the experiment procedure.

5.1 Preparations

The laboratory was set up as described in Chapter 3. September 10th we plugged in the Internet cables and all the computers went online. Nothing was downloaded or executed in order to expose the computers to malware, instead the computers were put online for the following reasons:

- Make sure all Windows installations had a working license.
- Update Windows with all available updates and patches through Windows Update.
- Update the anti-virus software with the latest virus definitions.
- Perform an initial test on September 15th to see if the computers were infected with malware just by the fact that they had been connected to the Internet.

5.2 System Exposure

Monday September 15th we actively started to expose the computers to web sites, file-sharing networks, etc. We had prepared an initial list of suspicious web sites containing warez, screensavers, codecs, mp3s and other free downloads. The list can be found in Appendix A, but the actual number of visited web sites is much larger, as we clicked on many advertisements and visited partner sites. Since web sites from Romania, Hong Kong and Russia were considered most risky by the computer security company McAfee, Inc. [46] [Kea08], we tried to include some sites from those countries as well.

As seen in Table 5.2 we had also come up with a list of search keywords, which we applied when using file-sharing programs. The list was compiled from the names of the 50 most popular Windows downloads at Download.com [9] on September 15th. Download.com offers reviews of software and includes freeware, shareware and try-first downloads. The site attracts a large number of visitors and the most popular files exceeds a million downloads per week. Inspired by a study of malware in the peer-to-peer network Kazaa [SJB06] we used file names from Download.com as proxies for popular files.

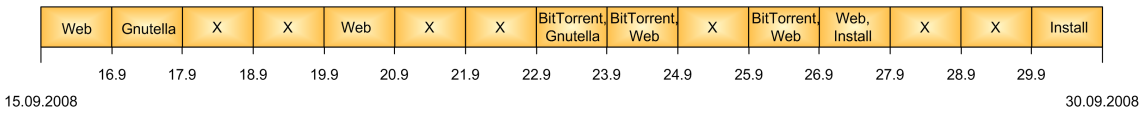


Figure 5.1: System exposure timeline.

As the timeline in Figure 5.1 indicates, the system was exposed to web sites and file-sharing networks over a two week period. Some days were spent on one source only, while other days consisted in the use of several sources. *Install* indicates that the same downloaded files were installed on all the computers. The time slots containing *X* means nothing was actively done to expose the computers, but they were powered on and still connected to the Internet and both Internet Explorer, μTorrent and LimeWire were running.

It is important to point out that the same actions were performed on the six computers in the laboratory almost simultaneously. With three persons and six computers, each person was responsible for two computers, and the actions and downloads were coordinated accordingly. The following sections describe the procedure and actions that were taken in Limewire, μTorrent, Internet Explorer and the anti-virus software, respectively. Problems and important experiences are covered in Section 6.3 and 6.4.

5.2.1 LimeWire (Gnutella)

As already described the file-sharing client LimeWire was used to download files from the Gnutella network. We searched for and downloaded files based on a prepared list of keywords, as shown in Table 5.2. Since we basically were searching for software files, we chose to download what LimeWire considers to be *Programs*[1]. An apparently new function in LimeWire is called *What's New?* and is explained to search for files that have been recently added to the network. This was also applied.

[1]LimeWire considers the following extensions to be program files: 7z, ace, arj, bin, bz2, cue, deb, dmg, exe, gz, gzip, hqx, iso, jar, pl, rar, rpm, sit, tar, taz, tgz, z, zip

The following steps were performed:

- In Limewire we chose *Programs* on the left hand menu.
- The keywords listed in Table 5.2 were applied, one by one.
- A *What's New* search was also performed.
- From the results, we chose to download all files that were less than 10 MB in size.
- If LimeWire reported duplicate file names, we chose to automatically rename them. Equal names are not necessarily equivalent with equal content.

An extract of the downloaded files is shown in Figure 5.2.

avg, antivirus, adaware, limewire, frostwire, winrar, winzip, mirc, irc, player, real, media player, zip, free edition, youtube, downloader, irfanview, google, chrome, adobe, firefox, virtualdj, vlc, iso, cleaner, msn, live, nero, divx, spyware, torrent, activex, flash, trillian, norton, mp3, 2008

Table 5.2: List of keywords used for data collection.

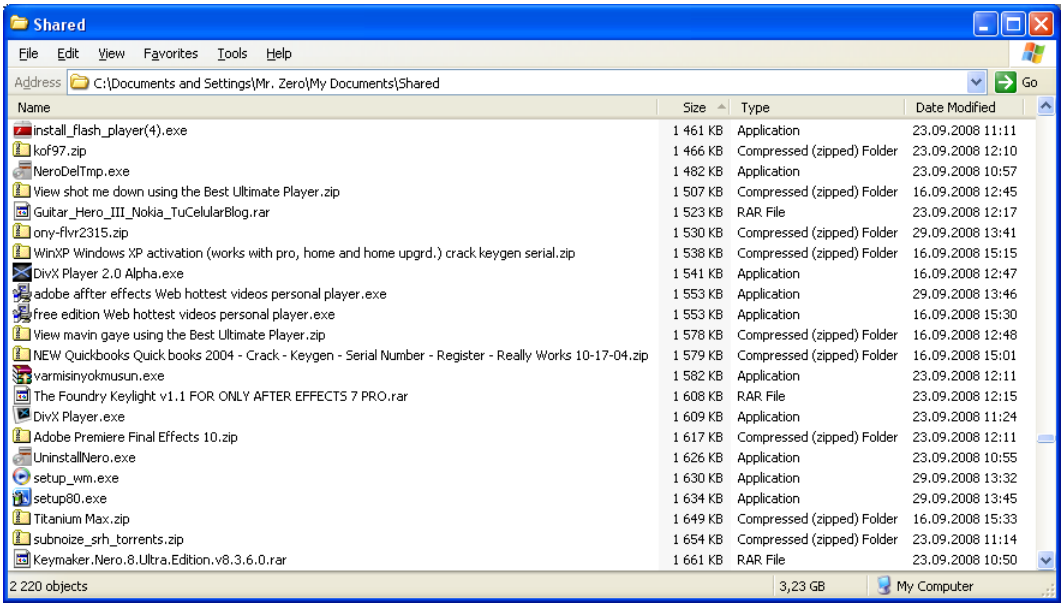


Figure 5.2: An extract of the downloaded files using Limewire.

Because of the huge amount of downloaded files and the limited time available, we could not install all of these programs, but had to make a selection. Unfortunately, we experienced that not many identical files existed on all of the computers. As described in Section 4.3, we collected the downloaded files from the different computers to a common pool, then distributing them back to the laboratory computers. Now every computer had the exact same files.

Table 5.3 lists the files that were installed on all computers. It is a selection among the total of 2220 files, where we tried to install both archive files and ordinary executables, and of different size.

File name	Size
mplayer.exe	5 KB
Nero 6_Keygen.exe	46 KB
bittorrent_downloader.zip	131 KB
irc.jar	200 KB
Wyzo Setup.exe	656 KB
NERO 8 Ultra Edition (ENHANCED) + Serials!!(REAL) for Windows Vista+++.zip Neroremovaltool.exe URL files	3297 KB 966 KB
antivirus Crack All Version.zip	2 KB
football manager 2008_v8_keygenerator.rar	328 KB
irfanview_plugins_410_setup.exe	6353 KB
Jnes.exe	72 KB
Norton Antivirus v2007.exe	4422 KB
RhapDrmClean.exe	188 KB
Sims.exe	2116 KB
TrillianPro-3.1.0.121.exe	6337 KB
VirtualDJ Pro v4.3.R12.exe	5469 KB
VLC Media Player 0.8.6d (Latest Version) -Legal-Ups.exe	5573 KB

Table 5.3: List of installed files downloaded from the Gnutella network.

Although these files had been scanned by the desktop anti-virus programs, some of them still contained malware. Because our computers were running different anti-virus software, the behavior was not the same on all machines. Some anti-virus programs refused to install a few files, while others did not complain at all.

5.2.2 μTorrent (BitTorrent)

Downloading of files from the BitTorrent network have become very popular in the file-sharing community. There are many applications that can be used to download files from this network, but we decided to use a very popular client called μTorrent. For a description of how the BitTorrent network is structured, refer to section 2.3.2.2.

The search procedure in BitTorrent differs from the one in LimeWire/Gnutella. While LimeWire offered an integrated search functionality, μTorrent and similar BitTorrent clients do not have this feature. In this network you rely on search engines on the web, which search through torrent files on a BitTorrent tracker. We decided to use BTJunkie [7], an advanced BitTorrent search engine which uses a web crawler to search for torrent files from other BitTorrent trackers. With over 1.5 million active torrents and from 5,000-25,000 torrents added daily, it is considered one of the largest BitTorrent search engines.

When searching for torrents at BTJunkie, we used the same keywords as in the Gnutella case. See Table 5.2 for a complete listing.

In order to download the most recent files, we performed the following strategy:

- The option *From Newest to Oldest* was chosen from the *Advanced* search.

- The keywords listed in Table 5.2 were applied, one by one.
- From the results, we chose to download the ten newest files published in September with a size up to 100 MB.
- If there was not as much as ten files added in September, we downloaded less than ten files.
- We also downloaded files with red warning marks, regardless of whether they were among the ten newest in September or not. The red mark indicate that the torrent is given negative feedback and it could possibly contain malware.

A typical search result in BTJunkie can be seen in Figure 5.3.

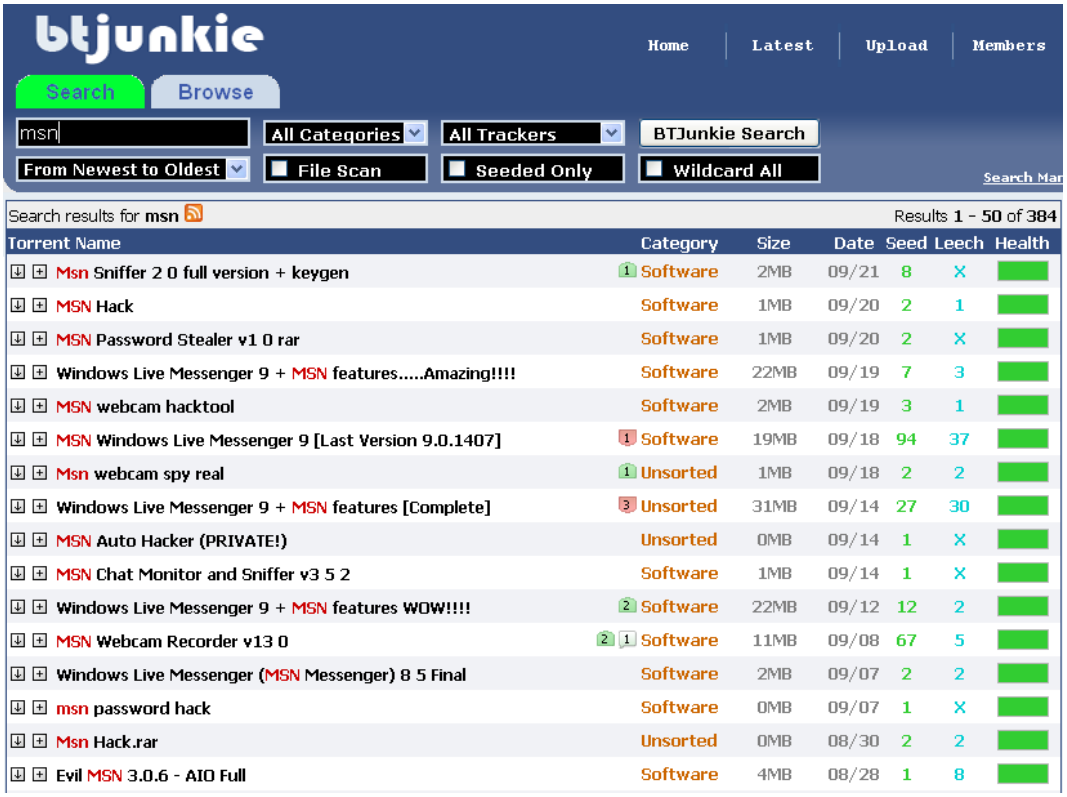


Figure 5.3: Search results for *msn* in BTJunkie, sorted from newest to oldest.

Again we made a selection of files to install, based on file names and file sizes. If the archive files contained several files, all of them were executed. The list of installed files is shown in Table 5.4.

As mentioned previously, the desktop anti-virus programs were constantly running on the different laboratory computers. The idea was that all known malware would be removed immediately, as our goal merely was to get zero-day malware. On the basis of this fact, it was kind of surprising that most archives contained files with malware. It seems like a lot of the anti-virus software have lower detection rates when malware are compressed into archive files. Also, a lot of malware are detected when a file is executed, not in advance. This could indicate the presence of anti-virus technologies like behavioral analysis and is described in Section 2.4.4.

File name	Size
(4 in 1) DVDFab Platinum 5.0.9.0 Final+++.zip	29,4 MB
chrome_installer.exe	7,34 MB
DVDFab5085.exe	6988 KB
thebat_pro_4-0-34	15664 KB
Everest KEYGEN-FFF	163 KB
everestultimate460.exe	8,12 MB
All Codecs For Windows Media Player.zip	5,97 MB
GordianKnot 0.21 Setup.exe	6116 KB
Driver_Genius_Pro_2008_v8.0.316 PLus Keygen.zip	9,87 MB
setup.exe	9722 KB
keygen.exe	322 KB
Iwin Games - Pageant Princess + Adnan boy 2008 + precracked.zip	24,5 MB
Paegant Princess.exe	24996 KB
Mirc.v6.34.Incl.Keygen.Server.Patch.and.Blowfish-IITC.zip	2,64 MB
setup.exe	2,19 MB
server-patch.exe	70,5 KB
keygen.exe	155 KB
mirc.6.34.fish.no.online.check.patched-patch.exe	122 KB
Msn Sniffer 2.0 full version + keygen.zip	1,33 MB
MSN.Sniffer.exe	1,32 MB
keygen.exe	311 KB
PURECODEC2008 (#1 codec pack ALL FORMATS PLAYER).zip	60,2 MB
Purecodec2008.exe	60,2 MB

Table 5.4: List of installed files downloaded from the BitTorrent network.

5.2.3 Surfing the Web

More and more people get access to the Internet, but it can not be regarded safe, even though you try to avoid obviously suspicious web sites. According to a technical report by Google [PMRM08] approximately 1.3% of the incoming search queries to Google's search engine return URLs labeled as malicious. In order to expose the laboratory computers to a wide range of threats, it was then natural to visit some potentially risky web sites.

A great deal of all web sites contain adware, viruses and other threats. McAfee has published two reports that show which domains that are most risky [46] [Kea08]. These are domains from Romania, Hong Kong and Russia and therefore we tried to include some sites from those countries. Based on these reports and the use of search engines like Google and Yahoo! together with popular search phrases, we came up with a list of possibly malicious sites.

To obtain search results from a specific domain/country, we used the *site:* function in Google. For instance if we wanted a Russian mp3 site we would use the query: *mp3 site:*.ru*. We also included sites which were labeled as potentially malicious by the search engine. The complete list can be found in Appendix A. The list contains a mixture of popular ordinary web sites and sites which claim to serve downloadable items, such as warez, screensavers and mp3s, obtained through the methods described above.

When visiting web sites, the integrated browser Internet Explorer 7 was a natural choice. If plugins like Flash etc. were missing, they were installed on demand.

The following strategy was followed:

- We basically started at the top of our list and visited the web sites one by one.
- Since our goal was to be exposed to as much malware as possible, we acted like a novice user, uncritically clicking *OK* to everything that popped up.
- If the particular web site had partner sites or other tempting links, we paid them a visit too.
- When visiting warez sites, where it was possible to download for instance software, we typically chose a few of the most popular items and saved them to a directory on the computer for later analysis.

The same procedure was performed on all our computers almost simultaneously. Still we experienced that different pop-up windows showed up on different computers, so minor dissimilarities occurred.

Table 5.5 lists the files that were installed. As with the installed files from Gnutella and BitTorrent, a few of the files from the web also turned out to contain malware, although they had been scanned by anti-virus software previously.

File name	**Size**
keygencrackv1.exe	29 KB
Download_ad_Savannah_camp_trial	126 KB
Adriana_Lima3.exe	3731 KB
PartypokerSetup.exe	360 KB
mp4playersetup.exe	809 KB
youtubedownloader.exe	1042 KB
babefest2008.exe	4315 KB
cool_dvd_player_standard_setup_7.0.exe	28499 KB
odc-5.32.exe	1903 KB
Install_MSN_Messenger	9189 KB
DIVXInstaller.exe	20214 KB
kitd.exe	8088 KB
xprepairpro.exe	6088 KB
ramaradio.v.2.22.Multilingual.incl.keymaker.and.patch-CORE.zip keygen.exe	 160 KB
azureus-ez-booster.zip patch.exe	 70 KB

Table 5.5: List of installed files downloaded from the web.

5.3 Malware Search

As described in section 3.3.2, different desktop anti-virus software was installed on the laboratory computers together with an anti ad- and spyware program, Spybot - Search & Destroy. In addition we obtained two offline anti-virus programs, F-PROT and Avast! BART CD. By offline we mean that the host operating system is not booted, but the anti-virus software is booted from CDs as described in Appendix B.

5.3.1 Desktop Malware Search

The desktop anti-virus programs[2] were constantly running on all the laboratory computers. The idea was that all known malware should be removed from the systems, and we had configured all programs to quarantine infected files if possible. In addition, every anti-virus software was scheduled a nightly scan at 4:00 a.m. to make sure the complete system was checked for malware. In the beginning this was a trivial task, but soon we realized that surfing the web and downloading and installing applications uncritically, resulted in serious malware infections. At the end of our two weeks of exposure, three of the six computers were so infected with different malware that they were unable to perform a decent full computer scan.

Another problem, which occurred in all anti-virus packages to various extent, was related to files compressed in the RAR format. Some programs had very limited scanning capability for this type of file, and would therefore require to extract the compressed files manually. Of those programs that actually found viruses in RAR files, most did not manage to delete them. Thus, after each nightly scan, we had to go through the logs and manually delete the files (i.e. RAR files) that the anti-virus program reported as infected, but had not deleted.

To remove ad- and spyware infections not detected by the standard anti-virus software we used Spybot - Search & Destroy. This program has limited real-time protection and is most effective when used as a full system scan. When it was needed, i.e. when the system had become too infected to perform the tasks at hand, we performed a full system scan. This scan sometimes requires a reboot of the machine and starts very early in the boot process, thus having some of the same properties as an offline scan. This type of scan proved effective against most of the ad- and spyware threats encountered, but some infections could still not be removed or even detected.

5.3.2 Offline Malware Search

As noted in the time schedule at the beginning of Chapter 5, offline scans were performed three times during this project.

1. A control scan was performed before starting the exposure phase, to make sure or check that the laboratory computers were clean.

2. A baseline scan was performed after the two weeks of exposure.

[2]The desktop anti-virus programs consisted of F-Secure, Norton Internet Security 2008, Norman Security Suite, AVG Free Edition and Avast! Home Edition.

3. The final scan was performed one month after the baseline scan, in order to compare the results.

5.3.2.1 Control Scan

When performing the first control virus-scan, the laboratory computers had been connected to the Internet for five days and we wanted to check whether they were clean or not, before starting the actual experiment. We chose one of our two offline anti-virus programs, F-PROT, and performed a full system scan according to the instructions described in Appendix B.2. Nothing was found on any of the computers, but there is a slight chance that they were already infected by zero-day malware.

5.3.2.2 Baseline Scan

After two weeks with exposing the computers to possible threats, it was time to disconnect them from the Internet and turn them off. In addition to the full system desktop scans we did every day, we now scanned all computers with both F-PROT and Avast! BART CD. The anti-virus definitions were updated the same day as we carried out the malware search.

First all machines were scanned with Avast! BART, and where possible we instructed the program to delete all the infected files. This was done because some machines had nearly one thousand infections, and the comparison between the logs would be a time-consuming task with this amount of information. Our goal was to find zero-day malware, so infections found in this baseline scan were not of particular interest. It seems dramatic that some computers still had this big number of infected files, although they had been scanned by the desktop anti-virus programs. One reason is of course that some of the machines at the end were so infected with malware, that they did not manage to perform a full system scan. For further information about the different behavior and results during the exposure period, refer to Section 7.3.

The machines were then scanned with the second offline anti-virus program, F-PROT. In this case no modifications were done to the hard drives of the host computers. We encountered some memory issues when scanning with F-PROT, but these were resolved and a solution can be found in Appendix B.2.2.

5.3.2.3 Final Scan

One month after the baseline scan, the laboratory computers were again scanned with the offline anti-virus programs, Avast! BART and F-PROT. During this one-month period all the six computers had been shut down and nothing was done in the laboratory. Since F-PROT was applied lastly in the baseline scan, we chose this to scan first in the final malware-scan. Subsequently Avast! BART was applied and the practical part of the experiment had reached an end.

Both the baseline scan and the final scan were thoroughly logged and the results of these scans are described in Section 6. With six different machines and a lot of scan logs, an in-depth examination of the logs were required. The methodology for comparing and merging all the different anti-virus logs are explained next.

5.4 Consolidation Methodology

This section describes how we used the anti-virus scan logs from F-PROT and Avast! BART to compile a list of zero-day malware candidates to be further investigated. We first refer to these files as zero-day malware *candidates*, because there might be false positives and duplicates among them.

5.4.1 Comparison of Anti-virus Logs

The first step was to compare the F-PROT logs from the baseline scan with those from the last scan, and for each of the six machines we compiled a list of the malware detected only in the final scan.

The second step was to compile a list of zero-day malware from the Avast! BART scans. The first action done in this step was to eliminate all the malware that was deleted in the baseline scan by going through the logs from the baseline scan. Then we compared the remaining lists with the results from the last scan. A list was then made for each machine consisting of the malware only detected in the final scan.

All this information was then merged into a large spreadsheet, with the filenames of all zero-day candidates and on which machine(s) they were detected, together with the name of the malware given by Avast! BART and/or F-PROT. All the files in these lists can be said to be zero-day malware (as of the time of the baseline scan) with respect to either F-PROT and/or Avast! BART, excluding potential false positives. This large spreadsheet can be found on the attached CD, further described in Appendix E.

5.4.2 Eliminating Duplicates

The resulting spreadsheet consisted of 411 files, and all these files were copied to a separate folder on *Jerry*. By further inspection, we noticed that several of these files seemed equal. For instance the file name could be different, but the file sizes were equal. Some files also seemed to be equal, with equal name and/or size, but were not. By comparing the hash value of all apparently equal files, we were able to eliminate about 3/4 and now we had 124 zero-day malware candidates as a basis for further analysis. Note that a very large amount of this reduction was due to 209 packed media files which proved to actually be only 3 different files. Also note that some of the malware candidates are within compressed archives (mostly .rar and .zip) and may contain several malware candidates, this means that the same archive can be listed several places in the final table.

5.4.3 Uploading to VirusTotal

All the 124 files were then uploaded to the online anti-virus scanner VirusTotal, this service is described in Section 3.3.2.3. The number of malware detections for each file was noted, and in addition the full results were saved in HTML files. These files can be found on the attached CD, see Appendix E.

The number of VirusTotal detections was added to the table of zero-day malware candidates, along with the date of when the specific file was first seen by VirusTotal if it had

been uploaded previously by others. The detections made by Symantec and F-Secure via VirusTotal was also added to this table.

5.4.4 False Positives

False positives (or false alarms) occur when an anti-virus program mistakenly flags an innocent file as being infected. Sometimes the reported malware name can give an indication of whether a file is correctly identified as malware or not. If it ends in .gen (generic) or if it is identified as suspicious, one should take a closer look. In addition to analyzing the malware names reported by the different anti-virus programs, we sorted our list of zero-day malware candidates by how many of the anti-virus engines at VirusTotal who reported the file as infected. Every file reported by 12 or less anti-virus engines was uploaded and analyzed at VirusTotal a second time, one week after the first analysis. Since every anti-virus vendor represented at VirusTotal gets feedback about the results achieved for the submitted files, the number of detections at VirusTotal is likely to increase rapidly. If the number of infections over this one week period had increased, it was likely that the file contained real malware. On the other hand, if the number of infections were the same or lower than the first time, it was a possibility of a false positive and we analyzed the specific infection more thoroughly.

The number of files with 12 or less infections at VirusTotal were 40. After the re-scan about half of them had increased detection rates, and the other half still had the same number. We analyzed these cases of doubt further, by searching for additional info about the reported infection names. Still we found it hard to conclude with certainty that detections were false positives. Some files are for instance reported as *Win32:FraudTool-JP [Tool]*, *Hoax.Win32.Agent.s* and *Win32:Hacktool-AU [Tool]* and are probably not malware according to a strict definition, but they are not ordinary innocent files either.

Out of the 124 zero-day malware candidates, none of them are removed from the list for being a false positive. We expect the number of such false alarms to be relatively small, but when examining the list of zero-day malware, be aware that some of the files with a low number of detections at VirusTotal may not necessarily be harmful malware. The remaining 124 files are thus all zero-day malware files with respect to F-PROT and/or Avast! BART, and no longer just candidates.

5.4.5 Non Zero-day Malware

It is important to note that these 124 files are not necessarily zero-day malware to both Avast! BART and F-PROT, it is possible that for instance if a file was detected as zero-day by Avast! BART it could have been detected in both the baseline and the final scan by F-PROT. This would mean that this file is *not* considered zero-day with respect to F-PROT, but it is still zero-day with respect to Avast! BART.

To check if some files had this property we examined the VirusTotal log files for all the 124 zero-day files, to see if they were in fact detected by the anti-virus engine that had not labeled them as zero-day malware previously. If the file was detected in VirusTotal this would indicate that the file had been detected in the baseline scan, and therefore the log files from the baseline and final scan were examined to see if this was true. If this was the case, the files would be added to its respective column in the table, but labeled as

Not-Zero: . Out of the 124 zero-day files, 9 were detected as *Not-Zero* by F-PROT and 3 by Avast! BART. The final zero-day malware files are listed in Table C.1.

Chapter 6

Results

This chapter presents the results obtained in our experiment (please note that these results are discussed further in Chapter 7). The main result is the 124 zero-day malware instances that were obtained in our experiment. A selection of these files is listed in Section 6.1, along with results from scanning these files with the online virus scanner VirusTotal [61]. Further we will present various statistics based on our results, for example from where the malware originated and an overview of the various malware types.

In Section 6.2 we give some simple analyses of the most interesting zero-day malware instances found. Section 6.3 refers to some of the experiences we had with the sources used to obtain potential malware: Gnutella, BitTorrent and the Web. We continue in Section 6.4 by describing our experiences with the various anti-virus software.

6.1 Zero-day Malware Results

Table 6.1 shows an illustrative selection of all the 124 zero-day malware listed in Table C.1 in Appendix C. According to the offline anti-virus programs Avast! BART and/or F-PROT, they are all zero-day malware. In addition, all files detected by Avast! BART and F-PROT have been uploaded and scanned at VirusTotal [61] where 36 different updated anti-virus engines are present. This is indicated in the last column named *VT* and shows how many of those anti-virus engines that detected the file to be malicious.

File name	**Avast! BART**	**F-PROT**	**VT**
Easy Video Downloader 1.1 2008 fxg.rar	Win32:Trojan-gen {Other}		5/36
The Cleaner S01E02 HDTV XviD-2HD.zip	WMA:Wimad [Drp]		10/36
A0006004.dll	Win32:Adware-gen [Adw]		14/36
HirensBootCD.9.5.rar/DriverBackup.exe	Win32:Trojan-gen {Other}		15/36
WinRAR 3.71 Corporate.EXE		W32\Backdoor2.CXGS	17/36
urqPhgff.dll	Win32:Trojan-gen {Other}	*Not-Zero: W32/Virtumonde-.AC-.gen!Eldorado*	18/36
mIRC 634.zip	Win32:Trojan-gen {Other}		21/36
allok_rm2mp31.exe	Win32:Trojan-gen {Other}		24/36
MicroAV.cpl	Win32:Neptunia-AGB [Trj]		28/36
tdssadw.dll	Win32:Rootkit-gen [Rtk]	W32\FakeAlert.3!Generic	30/36
44Lty0BQ.exe	Win32:Trojan-gen {Other}	W32\Downldr2.ERTR	33/36

Table 6.1: An illustrative selection of zero-day malware found by Avast! BART and F-PROT.

A few of the malware instances were zero-day according to only one of the two anti-virus

engines. Such occurrences are marked with the prefix *Not-Zero* and is also written in italic. As an example, the file *urqPhgff.dll* were detected by F-PROT at both the baseline and final scan, and is thus only zero-day with respect to Avast! BART. The blank cells indicate that the anti-virus program did not detect the specific file to be malware. Take the first row as an example: The file *Easy Video Downloader 1.1 2008 fxg.rar* is detected by Avast! BART as *Win32:Trojan-gen {Other}*, while F-PROT did not detect it at all. However, F-PROT is not the only anti-virus engine that lacks a signature for this file. As shown in the last column, only 5 out of 36 engines detected this file to be malware. That means 31 engines, F-PROT included, were lacking a signature at the time of the last malware-scan.

From the VirusTotal results, we were able to check whether other anti-virus engines detected the same files as Avast! BART and F-PROT. F-Secure and Symantec were chosen because they are big vendors of anti-virus solutions and they also had good descriptions of the different malware types, as apposed to F-PROT who did not offer any information about the malware types on their web site. It was also impossible to obtain information about when Avast! and F-PROT had incorporated specific signatures to their databases. The results from F-Secure and Symantec are presented in Table C.2 in Appendix C, while a small selection is shown below in Table 6.2. The second column under F-Secure specifies when the malware signature was added to F-Secure's database as this vendor was one of the few that offered specific detection names and a date indicating when the signature was added.

File name	**F-Secure**		**Symantec**	**VT**
Easy Video Downloader 1.1 2008 fxg.rar	Backdoor.Win32.Reload.cg	07.07.2008		5/36
The Cleaner S01E02 HDTV XviD-2HD.zip			Trojan.Wimad	10/36
A0006004.dll			Downloader.Zlob!gen.3	14/36
HirensBootCD.9.5.rar/DriverBackup.exe	Trojan-Downloader.Win32.Zlob.aakn	16.10.2008		15/36
WinRAR 3.71 Corporate.EXE	Backdoor.Win32.Agent.sly	30.09.2008		17/36
urqPhgff.dll			Packed.Generic.180	18/36
mIRC 634.zip	Trojan.Win32.Buzus.aaqz	08.10.2008		21/36
allok_rm2mp31.exe	Trojan-Downloader.Win32.Zlob.zoe	24.09.2008	Downloader.Zlob!gen.3	24/36
MicroAV.cpl	Rogue:W32/XPAntivirus.GGW	08.10.2008	AntiVirus2009	28/36
tdssadw.dll	Rootkit.Win32.Clbd.kr	01.10.2008	Trojan.Adclicker	30/36
44Lty0BQ.exe	Trojan-Downloader.Win32.Agent.aidr	30.09.2008		33/36

Table 6.2: An illustrative selection of zero-day malware with detection names given by F-Secure and Symantec.

As Table 6.2 indicates, a lot of the malware we found is also considered zero-day according to both F-Secure and Symantec. If F-Secure reports to have added the signature at some date in October, it means they did not detect the malware at the time of the baseline scan (which was done on October 1th). As indicated by the empty cells, some of the files were not even detected at all, and we conclude that these are zero-day malware with reference to the specific anti-virus software. All files in these lists were gathered during September, which mean they have been around for over one month. It is disquieting that a large number of anti-virus engines do not detect these files to be malicious, although being in the wild for such a long time.

The results in Figure 6.1 are categorized according to when F-Secure claimed to have the given malware in its definitions database, if it was detected at all, detected by heuristics, or if F-Secure detected the malware but does not give a date of the virus definition update. The combination of the files not detected at all and the files which were detected by a

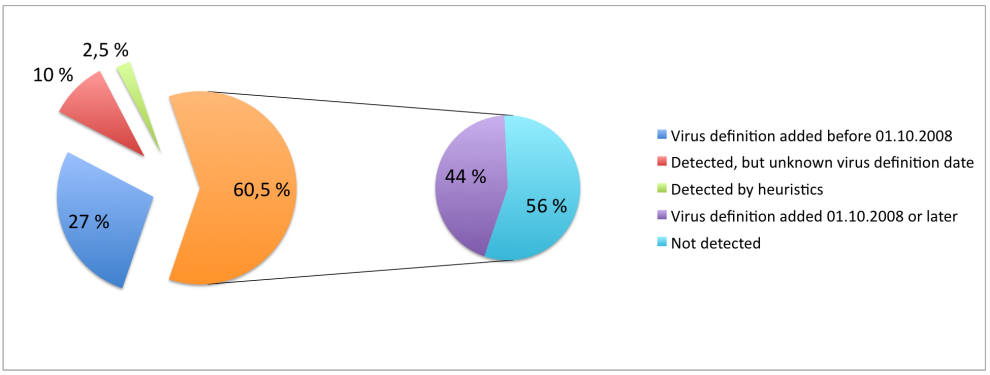


Figure 6.1: An overview of F-Secure's virus definition updates and detection rate with respect to the 124 zero-day malware candidates.

definition update after September 30th, can be said to be zero-day malware with respect to F-Secure on this date. This amounts to 60,5% of the zero-day malware candidates. The results are based on scans with VirusTotal on November 6th and information from F-Secure's virus definition updates. It is possible that both the malware detected by heuristics and the malware with unknown definition date are zero-day malware, but it is not possible to know this for sure. Another possibility is that the claimed definition update was incorrect and has been corrected at a later time, and thus the malware would not be detected at the time of the baseline scan even though this is claimed through the definition updates.

6.1.1 VirusTotal Detection

As mentioned in Section 3.3.2.3, VirusTotal consists of 36 different anti-virus engines. All of the 124 zero-day malware files were uploaded and scanned here, in order to compare the result with other anti-virus software. Figure 6.2 shows that a lot of files were only detected by a fraction of the total number of anti-virus engines. This is logical, since all of the files are zero-day and relatively new malware. It is likely that, if all files were to be scanned at VirusTotal again in a month, most of the pillars would have moved to the right in the figure, approaching 36 out of 36 anti-virus engines.

6.1.2 Zero-day Malware Sources

As we can see from Figure 6.3, which is based on the 124 zero-day malware infected files, most of the zero-day malware comes from the use of BitTorrent. While zero-day malware from the use of BitTorrent is estimated to 47%, zero-day malware from the Gnutella network on the other hand constitutes only 7%. The reason for this big difference can be explained by the difference in their search mechanisms, which is discussed in Section 7.6.

The part labeled *Other* in Figure 6.3 constitutes 42% of the diagram. *Other* includes files that are not located in the limewire, torrent or web download folder, but rather on the *Desktop*, in the *Temp* folder, the *Temporary Internet Files* folder, the *Program files* folder or different system files folders, to mention some locations. These files are difficult to indicate the origin of with certainty. Some of them may come from surfing the web, by clicking on different pop-ups and ads, and some may have been created when we installed

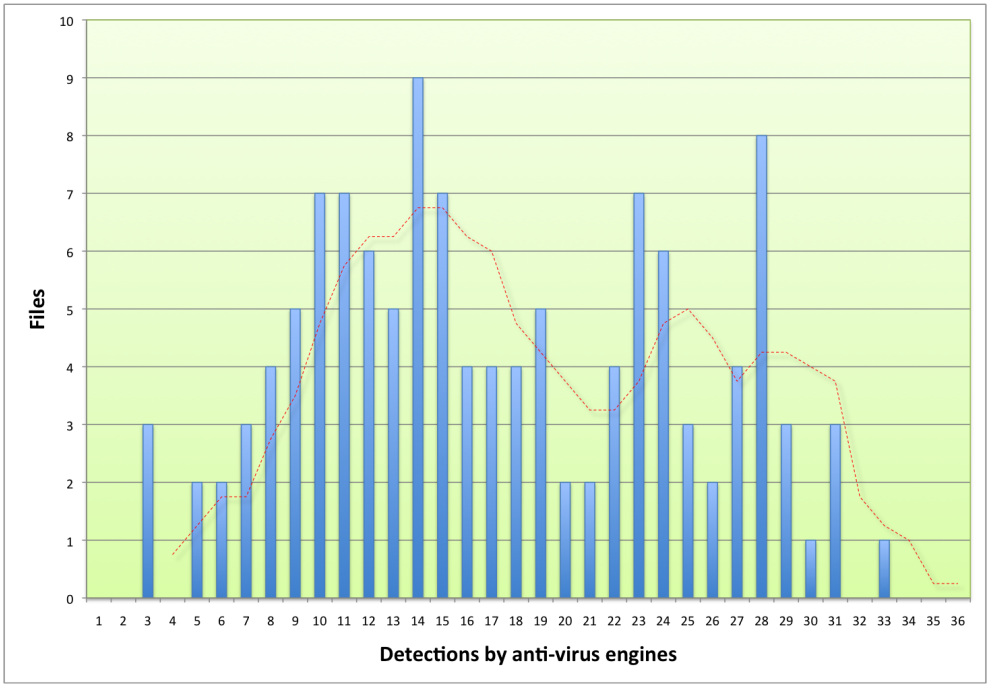


Figure 6.2: Distribution of how many engines at VirusTotal that detected the zero-day instances as malware. The dotted red line is a four-point moving average of the same distribution.

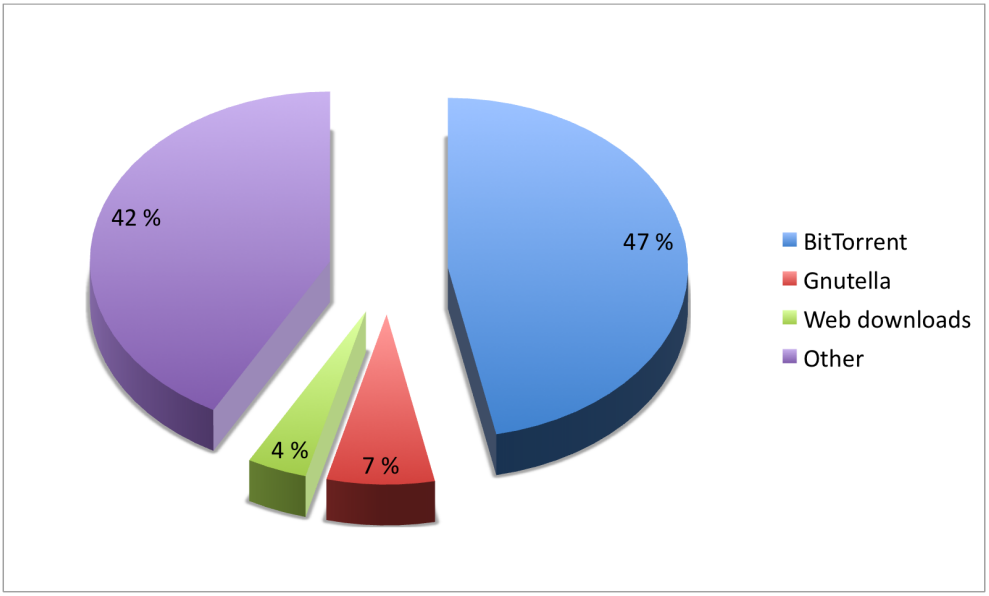


Figure 6.3: Sources for the 124 zero-day malware infected files.

some of the downloaded files. The part in the figure labeled *Web downloads* indicates that 4% of the zero-day malware comes from files downloaded when surfing the web, which is described in Section 5.2.3.

Table 6.3 is an attempt to estimate how many percent of the downloaded files in Bit-

Source	Downloaded files	Zero-day malware	Percentage
BitTorrent	∼400	58	14,5%
Gnutella	∼6000	9	0,15%
Web downloads	∼80	5	6,25%
Unknown source	?	52	?

Table 6.3: Approximately percentage of zero-day malware from different sources, based on all downloaded files.

Torrent, Gnutella and from the web, respectively, that contained zero-day malware. It is important to notice the difference between Table 6.3, which is based on all downloaded files, and Figure 6.3, which is only based on the 124 zero-day malware infected files. Due to our procedure, where desktop anti-virus software removed malware as they were detected, we do not have the exact number of downloaded files, and the percentages should only be used as a rough estimate. Through BitTorrent the number of download files were actually about 400. We had 40 keywords which on average resulted in 10 downloads apiece. The Gnutella number is more difficult to estimate, due to the procedure described in Section 4.3, where files from the different computers were gathered to a common pool. Each machine had a smaller number of downloaded files, but when merging the content of all six computers, the total number of files is estimated to approximately 6000. The number is not exact, but the conclusion is unchanged. The percentage of zero-day malware was significantly higher in the BitTorrent network than in the Gnutella network. Again, this can be partly explained by the difference in the search mechanisms, which is further discussed in Section 7.6.

It is worth mentioning that the files downloaded from the web are typically from the suspicious sites listed in Appendix A. It will be wrong to think that 6,25% of all files from the world wide web contain zero-day malware, but what the table indicates is that zero-day malware exists in all of these areas.

6.1.3 Detection by Anti-Virus Software

Figure 6.4 shows the detection rates of zero-day malware by the two offline anti-virus programs in the experiment. Avast! BART, represented by the blue pie slice, clearly outperformed F-PROT. 62% of the zero-day malware were only detected by Avast! BART, while the corresponding number for F-PROT was 23%. In addition 15% of the zero-day malware was detected by both anti-virus vendors, which is a quite small share.
A possible reason for this big difference is discussed in Section 7.7. It should be noted that 9 (7.3%) of the 124 malware was detected by F-PROT in both the baseline and the final scan, these instances are thus *not* zero-day malware with respect to F-PROT, but they are still considered zero-day by Avast! BART. 3 (2.4%) of the 124 zero-day malware instances were detected in both scans by Avast! BART, and are therefore only zero-day malware with respect to F-PROT.

The final anti-malware scan performed with Avast! BART and F-PROT was done on November 3th, while all files were uploaded to VirusTotal on November 6th, three days later. Because of this delay, it is not possible to directly compare the detection rates between Figure 6.4 and Figure 6.5. F-Secure, Symantec, Norman and AVG could have

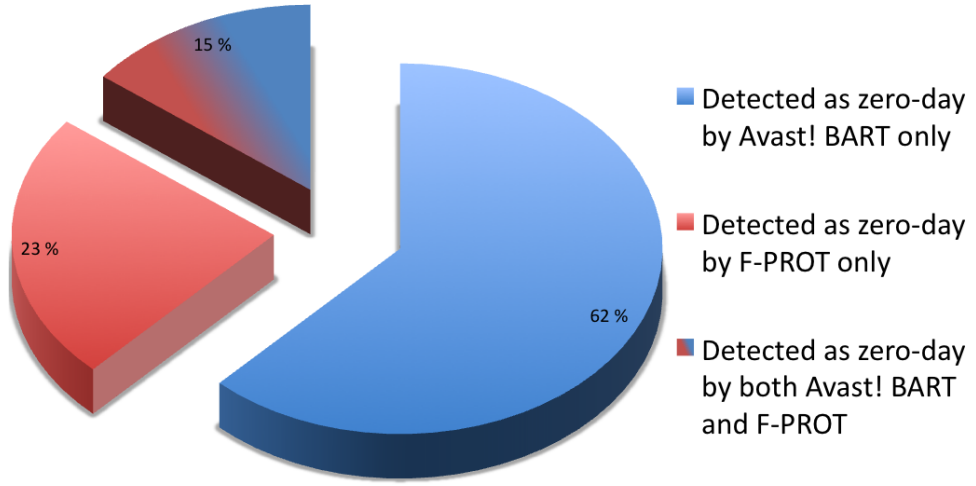


Figure 6.4: Percentage share of the 124 zero-day malware instances detected as such by Avast! BART, F-PROT or both.

updated their signatures in that space of time. Figure 6.5 illustrates the share of zero-day malware that is detected by F-Secure, Symantec, Norman and AVG when we uploaded the 124 zero-day malware files to VirusTotal [61]. The reason why we chose these programs is because they are the same programs that we used for the desktop malware scans, if we do not count Avast! which we also used for the offline scans together with F-PROT.

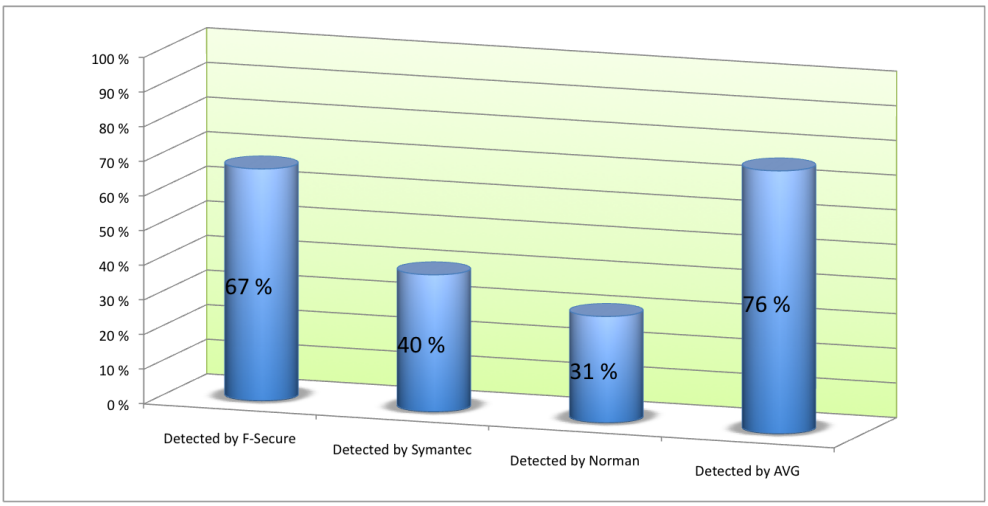


Figure 6.5: Share of zero-day malware detected by F-Secure, Symantec, Norman and AVG according to VirusTotal.

Still Figure 6.5 illustrates the big differences between anti-virus software when it comes to new malware. F-Secure and AVG detected more than twice as much zero-day malware compared to Norman. Symantec, with its 40%, did not impress much either.

6.1.4 Malware Types

Figure 6.6 shows the distribution of malware types. And as we see do trojans constitute most of the malware types, with its 74%. All other types constitute a very small percentage, with the second largest share being *other* malware at 10%, and ad- and spyware as a close third with 9%. It should be noted that no *viruses* were discovered in our experiment.

The categorization is based on the names given by four different anti-virus engines: F-Secure, Symantec, Avast! and F-PROT. The method followed was this: First we checked to see if F-Secure had a specific name of the malware, if it did and the majority of the others did not categorize it differently, this was the type used. If F-Secure did not have a specific name or no detection at all, Symantec was checked using the same method, this continued with Avast! and F-PROT. If no definite type could be found by using this method, the logs from VirusTotal were checked to see if a specific type was named by the majority of the anti-virus engines there, if not the malware was put in the *other* category. The *other* category also contains malware such as *hoaxes* and *hacktools*.

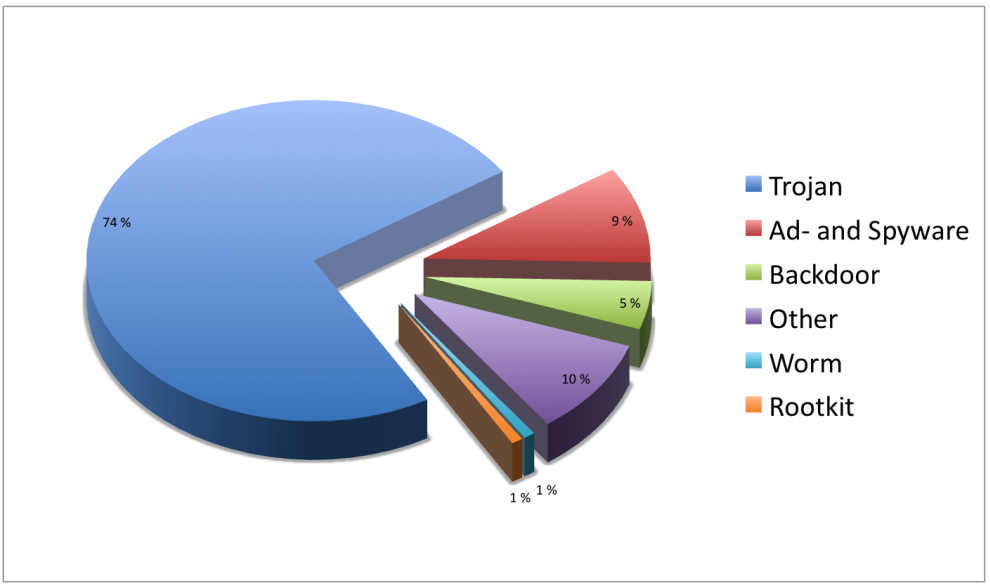


Figure 6.6: The distribution of malware types among the 124 zero-day malware instances.

6.2 Analysis of Malware

A thorough analysis of the 124 zero-day malware instances found are too time-consuming and out of the scope of this project. Still we have performed a quick web search of the malware based on the names given by F-Secure and Symantec. The information from these vendors' web sites are limited and it is hard to say if the malware is exploiting zero-day vulnerabilities or not. We suspect most of the malware to be just new versions of already known threats, but since a lot of the malware are lacking a proper description, it is impossible to know for sure.

Out of the 124 zero-day malware instances, the most prevalent are different versions of *Trojan-Downloader.Win32.Zlob*, *Trojan-Downloader.Win32.Agent* and the trojan *Virtu-*

monde. In addition many different kinds of fake anti-virus programs are detected. This section provides a short description of the most prevalent malware instances.

6.2.1 Zlob Trojan

Out of the 124 zero-day malware instances, 19 were infected by some variant of the Zlob trojan. The trojan allows the remote attacker to perform various malicious actions on the compromised computer. Typically it attempts to hiddenly download and run files from remote web sites and shows fake error messages. It will often download software considered as rogue anti-spyware. The first detection was done in late 2005, however, new variants continuously seem to appear. As an example F-Secure added 68 different versions of Zlob on November 12, 2008 [5].

Some variants of the Zlob family adds rogue DNS servers to the registry of Windows, an incident that we experiences on several of the machines in our lab. When for instance Norton tried to update its virus definitions, the request was redirected and nothing happened. Another "feature" is the way Zlob can infect a computer, by tricking the user into downloading a fake codec or protection system [16].

6.2.2 Trojan Downloaders and Droppers

There are a lot of trojan *downloaders* among the 124 zero-day malware instances. Even the Zlob trojan is actually in this category. A trojan downloader is typically installed through an exploit or some other deceptive means and facilitates the download and installation of other malware onto the computer [18]. A trojan *dropper* is somewhat similar, but instead of fetching malware from remote web or ftp sites, the dropper itself contains a few other files compressed inside its body. When a trojan dropper is run, the files are extracted and run simultaneously [19].

6.2.3 Virtumonde

Both *Trojan.Monder* and *Trojan.Vundo* are aliases for Virtumonde, a component of an adware program that downloads and displays pop-up advertisements [17]. It actively prevents removal by using several different techniques. Virtumonde installs itself as a Winlogon notification package and locks its own module. Since it is attached to Winlogon, it must be stopped before trying to remove it. Without Winlogon, there is no way to reboot the computer, so a forced reboot is needed and when Winlogon re-starts, the malicious files are re-created.

6.2.4 Fake Anti-virus Software

The list of zero-day malware shows that the machines in the lab were infected by various types of rogue and fraudulent anti-virus software. This was also inescapable to notice during the experiment as the computer screen was filled with pop-ups that offered to sell anti-virus products which would fix all problems. Panda Security claims that more than 7,000 variants of this type of malware are distributed during the last year and that about 30 million computers are infected [47]. To mention the products we faced during the exposure phase:

- Smart Antivirus 2009

- VirusRemover 2008
- MicroAV
- AntiVirus Lab 2009
- XPAntivirus also known as AntiVirus2009

We experienced that some of the fake products looked very authentic and the fact that 30 million users may have fallen victim to such scams is worrying. According to a recent study by the Psychology Department of North Carolina State University [SSW08] most users do not behave in a cautious manner when presented with fake dialog boxes and pop-ups in Windows.

6.3 Gnutella, BitTorrent and Web Experiences

During the experiment, we experienced problems and other events while using both LimeWire, μTorrent and Internet Explorer. The next sections describe some of the most important observations.

6.3.1 Gnutella

By using LimeWire we experienced that a lot of the downloaded files from the Gnutella network contained malware. This was confirmed by comparing the *download* folder and *quarantine* folder. About 70-80% of the files from LimeWire contained malware and was reported and moved to quarantine by the desktop anti-virus programs. This is not an exact number, but corresponds with the results achieved by Kalafut et al. [KAG06]. Another observation was that a lot of the files in LimeWire were of equal small size. Typically a search would result in about 20 files with a size of 113 KB, but with a lot of different file names. We suspected these files to actually be the same file even though they had different names and thus contain the same malware. To see if our theory was correct the following was done:

- Made a search in LimeWire using the keyword *free*, choosing *Programs* on the left hand menu.
- Downloaded the two files seen in Table 6.4
- Unpacked the two zip files and saw that both zip files contained a file called *setup.exe*. The size of these files were 276 KB.
- Scanned the two setup.exe files by uploading them to VirusTotal [61] to see if the two files contained the same type of malware.
- Compared the two setup.exe files by inspected both codes in a hex editor called HexCmp2, to see if the two files actually were the same file. A snapshot of the beginning of each file is shown in Figure 6.8.

From this evaluation we saw that the two apparently different zip files, with the same size, contained the same malware infected file called *setup.exe*. From Figure 6.7 we can see that 34 out of 35 anti-virus programs at VirusTotal [61] detected some kind of malware

File name	File size	Extracted content
Pos Free Photo Editor 1.05.zip	114KB	setup.exe
A Sense of Freedom (1979).zip	114KB	setup.exe

Table 6.4: The two files used for comparison. The *Extracted content* column indicates the actual files being compared.

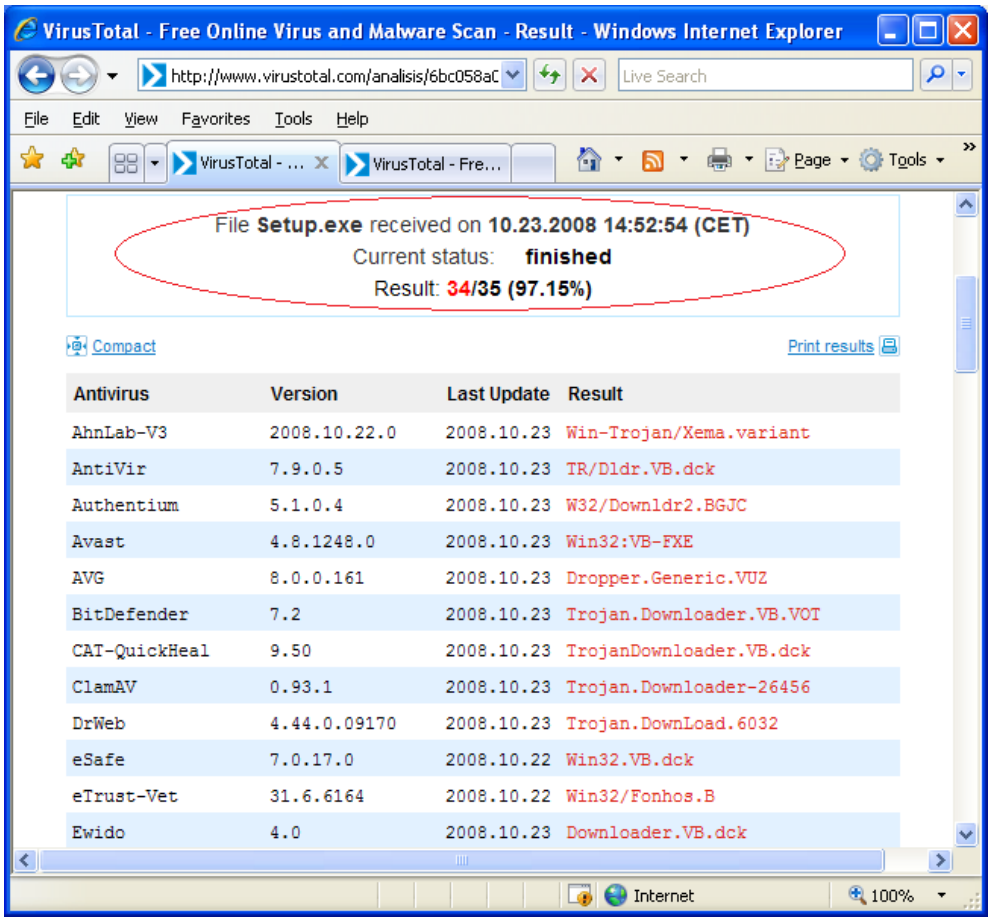


Figure 6.7: Result of virus and malware scan of setup.exe at VirusTotal.

in *setup.exe*. We got the exact same result for the other setup file showing that they contained the same malware. To be sure that these two files actually were the same we had to compare them. And as we can see from Figure 6.8 this was the case. Based on these results we think that many of the files in LimeWire with the same small size and different file names most likely are the same malware infected file.

6.3.2 BitTorrent

By using μTorrent and BTJunkie to download files from BitTorrent, we also experienced the prevalence of malware. About 25% of the total of 400 downloaded items contained malware and was quarantined/deleted by the desktop anti-virus programs. This number is somewhat similar to the one found by Andrew Berns at The University of Iowa [Ber08].

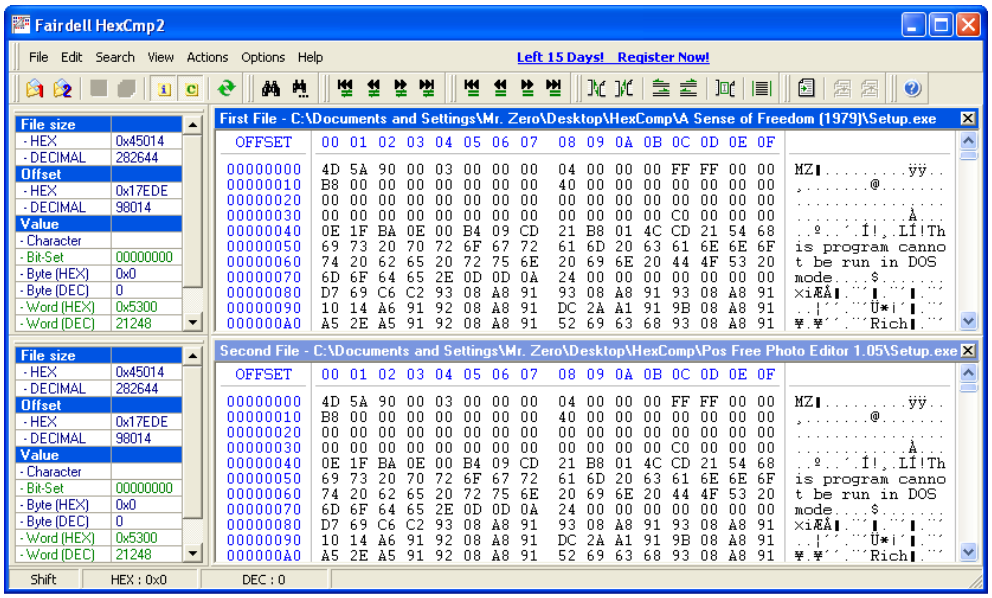


Figure 6.8: Hex comparison between two files with different file names, but with equal size.

The BitTorrent network felt more serious than the Gnutella network, mostly because the search results did not contain all these small files as experienced in LimeWire. A more detailed comparison between the search mechanisms in Gnutella and BitTorrent is given in Section 7.6.

6.3.3 The Web

In the exposure phase of the project we managed to visit nearly 300 different web sites. This is a limited number, but with six computers that had to be manually controlled, it was a time consuming task, especially since a lot of the sites infected our computers with annoying spyware, adware and other types of malware. With an increasing number of web sites, the probability of getting infected by some zero-day malware would of course have been higher. In order to cover a considerable amount of web sites, a web crawler might have been a better solution, but then again, you would not manage to expose the different computers to the exact same web sites. The possibility of a web crawler is further described in Section 7.10.

6.4 Anti-malware Experiences

In this section we share some of our experiences with the desktop anti-virus programs that we used for our experiment. A more general discussion about anti-virus software can be found in Section 7.3.

6.4.1 F-Secure Anti-Virus

F-Secure was installed on both a Windows XP and a Windows Vista machine. The general impression was very good and one of the better anti-virus packages in our lab. It seemed to find quite a lot of malware and managed to remove most of it. Malware that

automatically infected many of our computers, were stopped by F-Secure. The Windows XP machine performed very well until the last day of exposure. Then we tried to install some of the programs downloaded from BitTorent. F-Secure did not manage to stop one of the malware infected files before it was too late and it took control over the operating systems, for one thing it disabled the task manager. We initiated a full system scan and hoped that this would fix the computer, but the infection was so bad, that the virus scan never finished. During the nightly scan, F-Secure only managed to complete a few percent of the scheduled task.

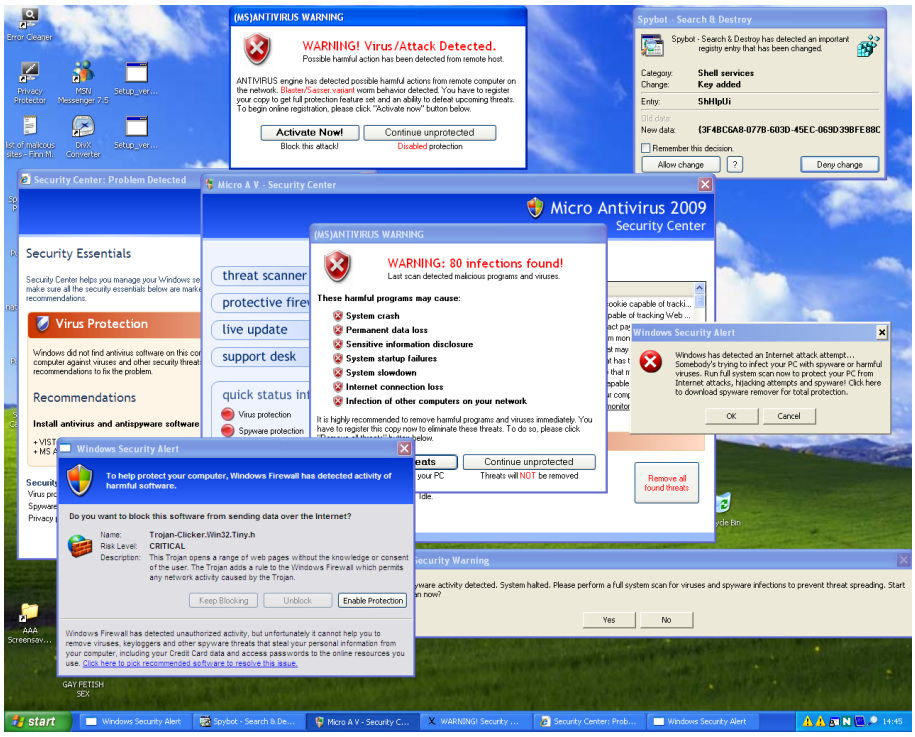


Figure 6.9: A screenshot from the Norman machine which got infected with a fake anti-virus program.

The Windows Vista machine had no user-noticeable infections at the end of the experiment. This can be a result of the more secure user-rights handling in this operating system, and other new security features compared to Windows XP. Another reason may be that there is more malware designed specifically for Windows XP. Windows Vista also has a built-in spyware protection, called Windows Defender, which stopped several spyware-infections. F-secure functioned well on this operating system as well, and provided good protection, a good user interface and a nice log feature with easy to read html-output.

6.4.2 Norman Security Suite

Norman Security Suite did not detect as much malware as for instance F-secure, but provided decent protection and detected most malware. One interesting function of this software was the detection of malicious Javascript/popup-exploits which contributed to a safer web-experience. Even though Norman stopped and detected several malware types, it became infected with so much malware in the end that the system was unusable in practice. For instance the last full system scan took over seven hours, which is many times

more than what is normal. Also the scan logs provided by Norman were a bit difficult to follow.

Figure 6.9 is a screenshot taken on the Norman machine after leaving it unattended for some time. As you can see a lot of dialog boxes from fake anti-virus software, a form of adware, are claiming that the machine is severely infected and suggests that the user purchases the full version of the software. Note that the fake anti-virus is very good at imitating the built in Windows dialog boxes, and the messages are very similar to original Windows security messages. In the upper right corner you can see that Spybot Search & Destroy has detected a change in the windows registry.

6.4.3 Norton Anti-Virus 2008

Norton Anti-virus 2008 performed well and managed for instance to stop the installation of the fake anti-virus program AntiVirus Lab 2009 one of the first days surfing the web. Unfortunately it was infected with a lot of other malware through the two weeks and we did not manage to remove all of it. At the end of this period, Norton actually stopped working. The subscription seemed to be expired and instead of performing a full system scan, Norton only managed to scan about 4800 files. A few days earlier a full system scan resulted in over 200 000 scanned files. We managed to fix the subscription issue, but Norton seemed to be infected by some form of malware.

6.4.4 AVG Free Edition

AVG was one of the two anti-virus software used in this experiment that were possible to download from the Internet for free. Both AVG and Avast! are so called lightweight programs, meaning that they do not take up much of the computer's resources while running. This is probably one reason why these kind of programs have become very popular. We have seen, through this project, that AVG performed quite well. AVG managed to remove or quarantine a lot of the infected files, even though there also were many files that it was not able to remove. It had trouble removing files inside .zip or .rar folders and therefore we had to remove some of them manually. It is worth mentioning that AVG did not get any serious problems from an infection, for example problems completing the scanning process, like F-Secure did. A very interesting function that AVG offers is that it gives a warning when the user attempts to enter a site reported as suspicious. This may stop you from entering the site in the first place and therefore works in a preventative way. AVG had an intuitive user interface and the scan logs were easy to read.

6.4.5 Avast! Home Edition

Avast! was the other anti-virus software we used that was freely available on the Internet. Even though Avast! detected quite a few of the infected files it had some problems taking care of them. Avast! had also, like AVG, problems with removing files inside .zip or .rar folders and we had to delete some of these manually. Avast! reported also that the file was too large for its quarantine folder. At first we did not realize what caused this problem, but to solve it we had to increase the size of the quarantine folder. This seems like an unnecessary user interaction requirement. Avast! should be able to take care of this by itself. From our experiences Avast! did not detect and remove as many infected files as for example AVG. Mitch, which had Avast! installed, got some bad infections during the

exposure period, but was operative during the whole experiment. Another problem with Avast! is that when you are performing a scan, the scan stops for each infected file and expect you to choose a way to take care of this file, for example sending it to quarantine. When it comes to the user interface we consider it as bad. It was not easy to browse the menus to find what you were looking for.

6.4.6 Spybot Search & Destroy

By installing the additional spyware program, Spybot Search & Destroy, the anti-virus software on the different computers were presented to more similar conditions. However, it turned out that Spybot did not offer sufficient real-time protection, but it was a nice tool for cleaning the computers when they had become severely infected by adware and spyware. An additional feature was reporting of attempts to make changes in the registry, and the user was requested to allow or deny the change. Unfortunately, it was hard to determine if the change was harmful or just a normal event.

Chapter 7

Discussion

In this section we will do a more comprehensive discussion based on our findings and experiences presented in *Results*, Section 6.

The main result of this project is that we were able to identify a total of 124 unique files in the category of zero-day malware. Actually we feared that nothing would be found and that desktop anti-virus software would remove nearly all of the malware. We kind of hoped for at least one zero-day malware, but 124 were far more than expected.

7.1 Zero-day Malware Prevalence

The term zero-day malware is widely used and everybody "knows" that it exists, but very few can refer to some actual numbers. So how prevalent is zero-day malware? During the two-week exposure phase ∼400 items were downloaded from BitTorrent, ∼6000 files from Gnutella and ∼80 files through the web. The number of visited web sites was nearly 300. From this amount of files and web sites, 124 unique files were identified to be infected with zero-day malware. As described in Section 6.1.2, BitTorrent generally contained less malware than Gnutella, but the amount of *zero-day* malware was in fact bigger. As pointed out in Section 7.6, this is probably related to the different search mechanisms in the two P2P networks. The task of downloading newly added files were much easier in BitTorrent, and the aspect of *new* malware is one of the most important matters of zero-day malware. If malware creators manage to distribute a new malware instance, which the anti-virus vendors do not currently detect, the possibility of successfully infecting a large number of hosts is a lot bigger.

Since all actions were done manually in the laboratory, the number of files mentioned above are not extremely high. Still we achieved to detect 124 unique files infected with zero-day malware. The procedure focused on exposing the computers in the laboratory to a broad range of suspicious material and generally acting as novice users. We did not hesitate to install programs, visiting ads and clicking *OK* to everything that popped up. Of course, a normal user would probably not manage to expose his or her computer to as much suspicious material as in this experiment, but the risk of getting infected by zero-day malware is still alarmingly high.

A deep analysis of the infected files is time-consuming and out of the scope of this project, but by briefly looking at the malware descriptions at F-Secure's web site, it does not seem

like the malware are exploiting zero-day vulnerabilities. Most of the infections are just new malware or new versions of already known types. Still they fall under our definition of zero-day malware and would not be detected by systems based on purely signature-based solutions.

7.2 Implications of Zero-day Malware

A naive, but well-known statement is the following: "*I have never had a problem with a virus, which is quite strange really, as I neither have anti-virus nor firewalls running on my computer.*" Such statements usually come from unexperienced and ignorant users who do not understand the risk of using a computer. The problem, which have been revealed through this project, is that even with an updated version of anti-virus software one is not safe. New malware, which the anti-virus engines do not have a signature for, is likely to penetrate a simple desktop anti-virus solution without being detected. Proper behavior on the Internet can only protect users to a certain extent, because if they visit the wrong web site or download a file with a zero-day malware, nothing can protect them from being infected.

Several industry analysts are proclaiming that traditional anti-virus systems based on signatures are dead [Blo06, Jaq07]. One of the arguments have been the poor detection of new malicious code. According to AusCERT, Australia's Computer Emergency Response Team, the top-selling AV solutions let in 80% of new malware. With a view to the increasingly number of malware that show up every year, the anti-virus vendors have problems keeping the pace. In its threat summary for the second half of 2008 [20], F-Secure report an additional one million database signatures added during the year, a number which have never been so high. As indicated in Figure 7.1 the growth is exponential and will most likely continue into 2009.

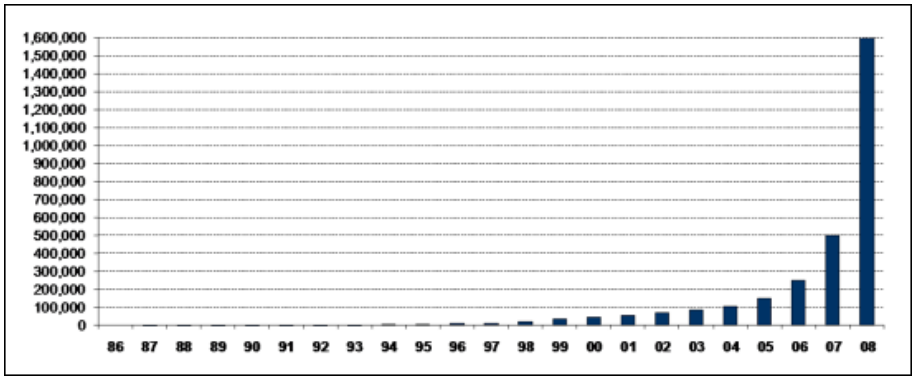


Figure 7.1: Accumulated number of malware signatures in F-Secure's database from 1986 to 2008 [20].

The acceleration of new malware instances can partly be explained due to techniques as polymorphism and metamorphism. These have been demonstrated as successful in evading commercial virus scanners [MKK07] by using various methods of obfuscation. When malware instances replicate, the code can be changed in a variety of ways. In a study by Christodorescu and Jha [CJ04] they tried to answer the question of how resistant a malware detector is to obfuscations or variants of known malware. They applied known

transformations to already existing malware and their testing showed that commercial virus scanners are not resilient to common obfuscations transformations. This project's list of zero-day malware supports these results. A lot of the infections are variants of the same malware, for instance the Zlob trojan. The observation of the different Zlob variants are further described in Section 7.5 while in Section 7.5.1 we try for ourselves to change the code of the Wimad trojan.

7.3 Issues With Desktop Anti-virus Software

As mentioned above there have been a vast change in the threat landscape over the last years [Per08]. All the different malware and unique patterns are being delivered so frequently that an anti-virus senior manager have referred to this as "*a denial of service [attack] against our labs*". Andrew Jaquith, a security analyst at Yankee Group have published a research paper entitled *Anti-Virus is Dead; Long Live Anti-Malware* and points out three main reasons why detection of new malware is so low [Jaq07]:

- *Limitations in static analysis*: Most anti-virus programs detect malware by recognizing data sequences that are unique to each variant. Such static analysis techniques fail when malware compresses (packs) or encrypts their payloads.

- *Signature bloat*: As malware instances increase, so are the corresponding signature file sizes. F-Secure recently reported that their total number of signature based detections is approximately 1.5 million. As a consequence, a full system scan on a typical PC will be more time-consuming.

- *Low and slow variants under the radar*: The malware instances that are most prevalent are typically prioritized by anti-virus labs. This mean malware creators can escape the system by creating thousands of variants. By making the malware mutate rapidly, individual variants can possibly avoid the radar.

As an example of the limitations in static analysis, refer to Figure 7.2.

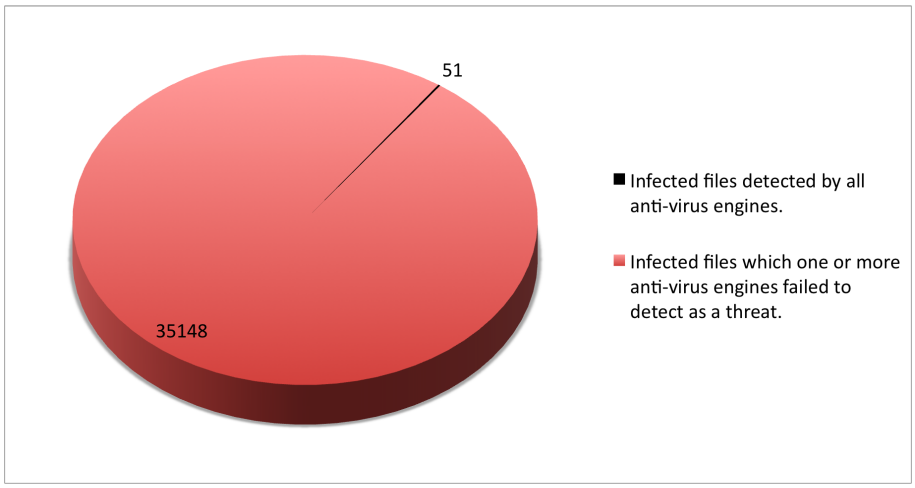


Figure 7.2: The share of malware files detected by all anti-virus engines in VirusTotal, statistics based on last 24 hours, November 27, 2008 [61].

Out of 35,199 infected files uploaded to VirusTotal, only 51 of them were detected by all of the 36 different anti-virus engines. Probably these 51 files are infected by relatively old or known malware, while the rest of the pie needs more days in order to update all the different signature databases.

It is obvious that the signature lag for anti-malware vendors is substantial. By looking at the list of zero-day malware in Appendix C.2 one is able to see when F-Secure added signatures for the different malware. The dates are widely distributed and range from July to October. The biggest part however, is added in the end of September or during October, while a considerable amount of the 124 zero-day malware instances are still missing signatures by F-Secure. The date when F-Secure added signatures and its protection percentage for the 124 zero-malware files is plotted in Figure 7.3. As seen in this figure the last signature was added about three weeks into the dormant phase, this implies that the signature lag for this specific malware was at least three weeks. As the exposure phase of our experiment was carried out the last two weeks of September, this indicates that the signature lag of anti-malware vendors vary from days to months. In addition there are big differences between the various products, as shown in Section 6.1.3. By uploading all the files to VirusTotal and looking at how many of the 36 different anti-virus engines that detected the files as malicious, the same spreading was revealed. A large share of the files were only detected by between 5 and 20 engines, while a smaller share was detected by 20 or more.

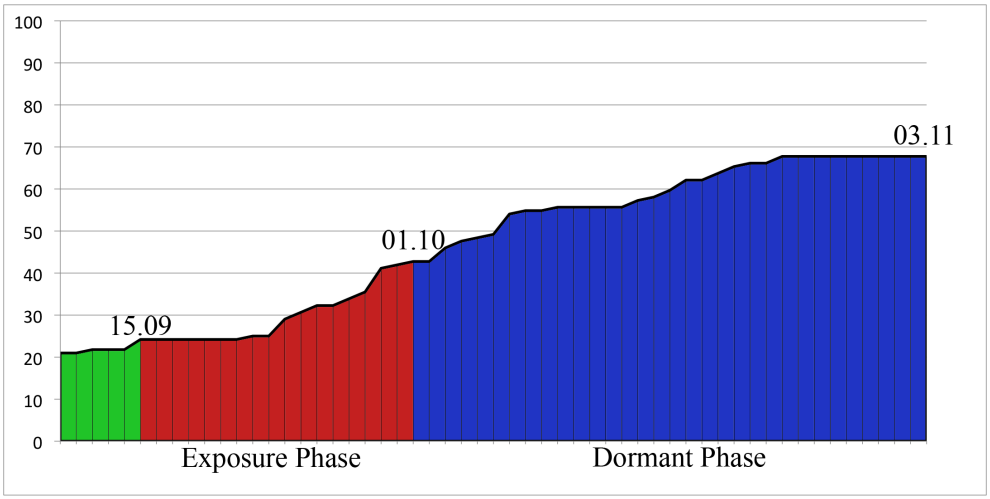


Figure 7.3: The protection percentage for the 124 zero-day malware candidates by F-Secure during the experiment, based on dates for signature updates.

Trying to estimate the window from a malware is in the wild until it is incorporated into a signature is not easy, because of the uncertainty of when a piece of malware surfaced on the Internet. Per definition the vulnerability window is larger than the signature lag, as this also includes the time before the anti-virus vendor is aware of the malware's existence. The vulnerability window is thus also in the range of days to months, but as the following example shows it can be substantially longer. As already described in Section 2.2.3.2 the *Rustock.C* rootkit was unknown to the anti-virus vendors for about 5 months. In addition it had a signature lag of over one month, because of sophisticated obfuscation techniques. The main reason for this large vulnerability window is that this rootkit was not very

widespread. Malware that spreads more rapidly will probably be detected and receive a signature in a much shorter time.

7.4 Web Virus Scanners

A web virus scanner, or online virus scanner, is in this context a web service where a user can upload a file and have it scanned for malware. The online virus scanner VirusTotal has been used throughout this project, and while this has been a powerful tool for us, some aspects of such online scanners have to be discussed. VirusTotal is a recognized tool for malware detection, and its makers are working together with the anti-virus vendors to provide a reliable and up-to-date service [62]. But such a powerful tool could easily be abused by malware-creators as it is easy for them to upload their newly created malware to VirusTotal to see if it is detected by the anti-virus engines. By submitting their malware to VirusTotal they are able to see if and how many anti-virus engines that detect their creations, they then have the possibility to modify and resubmit the malware until no engines are able to detect the file as malware. It is claimed that this method of malware creation renders the heuristic capabilities of the anti-virus engines helpless against this malware, as a malware-creator can test various tricks to avoid detection, before the malware is released to the wild [26].

The problem for the malware creators is that VirusTotal will send all non-detected samples to the vendors of the anti-virus engines that did not detect the malware, so they can create a signature for that specific malware. Up until January 2008 VirusTotal had an option called *Do not distribute the sample*, this was removed because of some questions raised by, amongst others, Kaspersky Lab [26]. If this option was checked, VirusTotal did not distribute the sample to anti-virus vendors. Originally created to protect sensitive information contained in the files submitted, this option made it *safer* to abuse the service by malware-creators. This option relied on trusting that VirusTotal did not distribute the sample to the anti-virus vendors, and some rumors claimed that the privacy of the submissions was not always maintained. Because of this an alternative to VirusTotal: *AvCheck.ru*, which claims total privacy for a fee of $1 per submission, can be used by people who do not want their submissions in the hands of anti-virus creators. VirusTotal also mention in their blog that they are aware of underground tools, containing pirated or free anti-virus engines, that are able to provide a similar functionality to that of their own service. The difference is that these tools can be installed and run on a desktop computer [55].

Another aspect that should be mentioned is that many anti-virus engines have additional behavior blocking functionality, which is not able to function in services like VirusTotal. Such detection rely on actually executing the file, either in some kind of protected sandbox or in a virtual machine, and observe if the file performs actions labeled as malicious (See Section 2.4.4). For this functionality to work, an actual copy of the anti-virus solution has to be installed and activated on a host machine. The makers of VirusTotal argue that their service is unsuitable for malware creators for this reason, and that professional malware creators have their own test labs consisting of virtual machines with different anti-virus solutions installed [55].

7.5 Problems With Signature Based Detection

The results obtained in our experiment show that a lot of the zero-day malware found are new variants of relatively old malware. For instance does variants of the Zlob trojan account for a significant share of the zero-day malware found. Other malware names that occur in several different variants in our results include Vundo, Monder and Wimad. It is common for anti-virus vendors to give names to malware, and the last part of this name is most often the *generation* or *variant* of the malware.

An example can be *Trojan-Downloader.Win32.Zlob.aamn*, which is a name given by F-Secure. The last part: .aamn, indicates that this is only one of many, probably thousands, versions of the Zlob trojan. Even though we could not find any good explanation of F-Secure's naming conventions, it seems that they are just incrementing the letters a-z and adding an additional letter when needed. This would mean that the mentioned Zlob variant is number 17954 in F-Secure's malware database.

Why cannot the anti-virus applications detect the new variants of such malware? Should it not be able to see that this is just a variant of a known threat? Almost all anti-virus software relies on signature-based detection, this is explained further in Section 2.4. In short this means that the anti-virus engine must have a more or less exact signature for every variant of a single type of malware, this is the reason why F-Secure has such an enormous number of signatures. As an example, F-Secure added 68 new variants of Zlob to their database November 12, 2008 [5].

Other engines like for instance Avast! and Symantec do not give a new name to every variant. Avast! gives the name *Win32:Trojan-gen (Other)* to almost any type of trojan, Symantec is a bit more specific with names like *Downloader.Zlob!gen.3*. Even though Avast! gives this general name to the Zlob trojan and others, it does not mean that they can detect unknown malware, they too need to update their signature files for every new variant. This statement is supported by our experiment where a large portion of the zero-day malware detected by Avast! is given this name.

7.5.1 Example of Weaknesses of Signature Based Detection

Some of the zero-day malware found in our experiment was based on a known exploit in Windows Media files, the so called *Wimad* exploit. This exploit uses a to open a URL and download a potentially malicious file or web site. The user can be told that the downloaded file is needed to play the media file, and in that way is tricked into installing the file, which can contain harmful malware. In other words the exploit is used as a trojan horse, as shown by the name given to it by F-Secure: *Trojan-Downloader.WMA.GetCodec*. After some investigation we found that this exploit has been known for some time, Symantec reports its first known use as early as January 2005 [28], and we were curious as to why this was detected as zero-day malware in our experiment. Some sources, including BitDefender [8], claim that this is actually a feature of the file format that is exploited, but a "feature" of this kind is so easily exploited that we would definitely consider it a vulnerability.

To investigate further we opened one of the malicious files, *The Cleaner S01E02 HDTV XviD-2HD.avi*, in a HEX-editor, and searched for the URL that was opened when we

tried to play the video. The URL was in clear-text in the file and we wanted to see if a change in the URL would be enough to avoid detection by anti-virus engines. So we removed the original URL: *http://www.lwstats.com/11/PLAY-MOVIE.exe* (Caution: This URL contains malware!), and replaced it with our own URL of the same length: *http://folk.ntnu.no/finnmich/testdow.exe*. The .exe file in the first link is a known malicious software, the one in our own link is just a file containing a small text string: *"heihei"*.

We uploaded both the original media file and our own edited file to VirusTotal and to our surprise the results differed by 50%. The original file was detected by 12 out of 36 anti-virus engines, while our new version was only detected by 6. Logs from VirusTotal for these two files can be found in Appendix D. By inspecting the logs from VirusTotal we could see that our new file was correctly detected as a variant of the Wimad exploit, these six anti-virus engines must therefore use a more general signature or some other means of detection. The six engines that did not detect the modified version of the file, but the original, probably use very specific signatures, at least for this type of malware. While our URL did not contain any malware, it could very well have been if it was made by a malware creator. This shows how easy it is to create zero-day malware by modifying existing malware, and how signature-based anti-virus software is a non-optimal approach to malware detection.

We know that VirusTotal sends the malware received to the anti-virus vendors if it is not detected by their engine [61]. We thought it would be interesting to see if any vendors had made and added a signature for our modified file in their database, so we uploaded the file again two days later. To our surprise the file was now detected by 7 engines , Microsoft's Live OneCare had added a signature for our file: *TrojanDownloader:ASX/Wimad.AB* [42]. This confirms that some anti-virus engines use very specific signatures for detection. Why Microsoft was the only one that updated their definitions is unknown, but one reason may be that the other vendors noticed that the .exe file downloaded with our variant was harmless, or simply that they have ignored the sample received from VirusTotal. The last log from VirusTotal can also be found in Appendix D.

7.5.2 Alternatives to Signatures

As shown, signature-based malware detection has severe weaknesses, and some people are now claiming that traditional anti-virus software based on signatures is becoming obsolete. The enormous increase in malware variants force the anti-virus vendors to update their signatures more and more often. Typically new signatures are being made available several times a day, but some claim this is not enough, and for vendors to keep up updates should be made every ten minutes. This is of course impossible and some have described the increase in malware variants as a DoS attack on the anti-virus vendors [41].

Other means of detection have been proposed, including the use of heuristic detection as described in Section 2.4. Another technique suggested is white-listing, this approach is attacking the problem from the opposite end compared to the signature-based approach. This scheme suggest that a list of signatures of all *non-malicious* programs is maintained in a database, this would in practice mean that a user is unable to execute or install files not present in this list. Several smaller and not so known anti-malware vendors have been using this approach for some time, but also large, traditional vendors are starting to use this approach, including CA and Symantec [41, 59]. The problem with this ap-

proach is how to populate this list. For a corporate environment this should be a doable administrative task as employees should be confined to a defined set of applications. The complexity increases substantially for desktop computers as the applications that are run on these machines vary. In desktop environments both type, version and other elements such as language of the applications can vary in almost any way, but still the supporters of white-listing claim that a list of these are easier maintained than a list of malware.

Behavioral analysis is another approach to detect unknown malware, this scheme relies on detecting malicious behavior of the malware when it is executed. This approach is further discussed in Section 2.4.4. One example of this type of protection is Norman Sandbox [4], which analyzes the malware in a sandbox environment. Norman also provides an online service for their sandbox product, where it is possible to upload files and receive an analysis performed by Norman Sandbox via e-mail [3]. A limitation of ordinary sandboxing environments is that they do not allow software to communicate with the Internet. However, a trend among malware authors is to use Trojan applications which do not contain any viral code themselves, but download malicious files from the Internet when they are run. An approach demonstrated by Jostein Jensen from SINTEF ICT uses the ideas from sandboxing, but also allows controlled connections to the Internet [Jen08].

As seen in our small experiment with the Wimad exploit, some vendors did manage to detect our modified version, and as mentioned we think the reason for this is that they use a more general type of signature, maybe combined with some form of heuristics. Such an approach will in general have a higher detection rate, but the downside of this is an increased chance of false positives. This is a trade-off that must be considered, this is also to some extent a problem with heuristic approaches.

The big advantage of pure signature based detection is that false positives are almost non existent, it is also much faster than the analysis required in a heuristic analysis. Another aspect mentioned by the supporters of signature based detection is that, even though malware can be detected by heuristics, a signature is needed for cleaning and removal of the malware.

7.6 Search Mechanisms in P2P Networks

As described in Chapter 6 most of the zero-day malware originated from the BitTorrent network. According to Figure 6.3 47% of the zero-day malware came from BitTorrent, while only 7% was from the Gnutella network. The number of files downloaded from Gnutella (about 6000) were in fact much higher than the number downloaded from BitTorrent (about 400), which indicates an even bigger difference between the two networks. We do not believe these numbers reflect the content of the BitTorrent and Gnutella networks correctly, but suspect different search functionalities to affect the result.

When searching for files in the Gnutella network, we used the software client LimeWire. All of our predetermined search keywords were applied and we started downloading all files below a certain file size. Through this process, two limitations were revealed:

- No possibilities to search for the newest files.
- The search stops when enough hits are found.

Actually, there was a new function in LimeWire called *What's New?* which said to search for new content, but there was no way to specify keywords. In the experiment procedure, we applied this function, and downloaded apparently new random programs. However, we could not see any time stamps and the search stopped after a limited number of hits in this search too. The idea is that the newer the file is the bigger is the chance that they contain malware that the anti-virus software does not detect.

The reason why the search query stops is explained in Section 2.3.2.1. The Gnutella protocol has technologies like Query Routing Protocol (QRP) which make searches lean and efficient. A search query is only forwarded to the nodes most likely to have a match and will terminate if enough hits are made. Continuing the search after sufficient number hits are reached will only result in unnecessary load on the network. The search messages also carry a time-to-live parameter (TTL), which place a limitation on how far a query can travel. These are smart functions, but inconvenient in our case when searching for newly added files.

In BitTorrent, on the other hand, we did not experience these problems. When using the search engine BTJunkie [7], we were able to specify our interest for new torrents and all the results specified on which date the torrent was submitted to the BitTorrent network. We could then be certain to have downloaded newly added files, which is not the case in the Gnutella network.

Another aspect of BitTorrent is the possibility to rate or give feedback on the torrents. In this way one is able to warn other people in the file-sharing community about suspicious or harmful files. But when downloading completely new torrents, as we did in the experiment, very few people have the complete file and this feedback is probably missing. One suggestion for avoiding potentially harmful files could then be to never download torrents with low seed counts, but unfortunately these seed and leech counts are possible to fake. At least in BitTorrent one has the opportunity to check how other people rate or comment on a torrent, while Gnutella lacks this functionality.

It is not easy to conclude which of the two networks that contain the most zero-day malware. If the search mechanisms had been similar, perhaps the results would not have been so different. In any case, this project has shown that zero-day malware exists in both networks.

7.7 Avast! BART versus F-PROT

As can be seen from Figure 6.4 in Section 6.1.3 Avast! BART outperformed F-PROT when it comes to zero-day malware detection. If the percent detected by for example Avast! BART only, Avast! BART and F-PROT together and the non-zero-day instances are added together, the result is that Avast! BART detected 79,4% of the 124 zero-day malware instances while F-PROT detected only 45,3%.

These numbers are in favor of Avast! BART and a possible reason why Avast! BART detected the biggest portion may be related to different detection techniques. By looking at the detection names in the anti-virus logs, Avast! BART has many similar and generic names. A great deal of the detections are for instance called *Win32:Trojan-gen* {*Other*}

which does not specify more than the main category for this malware, namely a trojan. It is hard to point out exactly what the differences between the detection techniques for Avast! BART and F-PROT are, because the vendors are of course not willing to reveal how the scans are performed, but a possibility is that Avast! BART uses a more heuristic or behavior-based approach and therefore is able to detect more. For F-PROT we used the default setting for the scan level, which was scan level two, but it is also possible to set this to level four, which is the highest level possible. The scan level decides how deeply it scans the files. This could have resulted in more detections by F-PROT. Another point is that F-PROT do not include adware search, but Avast! BART uses this by default. This can lead to a few more detections for Avast! BART.

7.8 Windows Vista

One of the computers in our experiment was installed with Windows Vista, as opposed to the rest of our computers which had Windows XP. During the experiment procedure this computer showed no signs of malware infection, and our results show that no malware was detected on this computer except from in the download folders.

Microsoft had a goal of making Windows Vista intrinsically more secure than previous versions of Windows, by implementing security features such as User Access Control and Windows Defender. User Access Control, or UAC, prevents the user from running program with Administrator privileges, which is considered to be a serious security risk. If an application tries to perform an action that requires Administrator privileges, UAC will block the action and prompt the user for consent [RS07]. Windows Defender is a built in ad- and spyware protector, which is enabled by default. During our experiment Windows Defender blocked several attempts to install infected programs, and it seemed to provide an efficient means of protection against this type of malware. It would seem that the goal of increased security in Windows Vista has been achieved, but is it more secure, or are malware creators simply not targeting Windows Vista to the same extent as Windows XP?

In an experiment conducted in early 2007, Symantec tested 2000 different malware files on Windows Vista, and found that in fact 70% would run on this operating system, but only 6% were able to infect the system, and only 4% would survive a reboot [tag07]. The underlying security enhancements of Windows Vista, especially UAC, blocked most of the malware. But the experiment also concluded that there exist attack vectors that would circumvent some of these features, this could indicate that (in 2007), most malware was still targeting Windows XP.

It should be noted that only one of the computers in our experiment had this operating system installed, and therefore our findings just give an indication of the security of Vista compared to XP. The machine was also installed with F-Secure anti-virus, which offered one of the better protections of the anti-malware software used in our experiment. Maybe the machine would have been infected had another anti-virus software been installed. It is even a possibility that the computer was infected, but no detection was made.

Our experiment is as mentioned too small to give any conclusive results on Windows Vista, but our results gives a clear indication that there exist far more threats against Windows XP, this is also supported by results obtained by others, for instance Symantec

[tag07]. Whether malware creators target Windows XP because of the increased security features of Windows Vista or because of the still widespread use of Windows XP remains a question. There is also a trend of malware creators increasingly targeting applications, like Adobe Flash or web browsers, instead of operating systems [43]. This could also suggest that the introduction of Windows Vista, as a more secure operating system, has forced this change of attack vector for the malware creators.

7.9 Lessons Learned

Here we point out some lessons learned and some key aspects that might have been carried out in a different manner. We also present an alternative method with another approach to the detection of zero-day malware.

7.9.1 Laboratory Equipment

It became evident that the computers used in our laboratory environment were not as powerful as we might have wanted. The time to complete malware scans increased substantially throughout the experiment, and would not complete on some computers at all. This was most likely due to malware infections, but more powerful hardware would probably compensate for this. Also more powerful computers are needed if a similar experiment is carried out with the use of virtual machines.

Unfortunately we were provided with a 10 Mbit hub as a part of our networking equipment. When downloading files from file-sharing networks this hub would cause a big bottleneck as several of the computers were connected to it, thus the network throughput was reduced significantly. This led to increased download times in the experiment.

7.9.2 Installation of Files

In the last part of the exposure phase, we installed files from both the file-sharing networks and the web. This was done to increase the real world aspect of the experiment, but this also led to some bad malware infections that interfered with the functionality of the computers. In retrospect this could have been done in an other way, as described in the next section.

7.9.3 Alternative Methods

There are several alternative methods that could have been used in this project. Our method focused on making the experiment as close to a real-world scenario as possible, this resulted in some unfortunate situations, where the most severe was that some of the machines got so infected that desktop scanning became impossible. An alternative method is presented in Figure 7.4, this method uses one machine for the actual downloading of files and another machine for installation of files. The harddrive is then removed from the computer and installed in for instance a USB-cabinet. The harddrive is then scanned by other machines with different anti-virus software installed. This scan would be a “desktop offline-scan”, because the anti-virus software is running on an operating system which is booted, but the files that are actually scanned are residing on an offline system. It is possible to use virtual machines for this method, thus reducing the complexity and hardware requirements considerably. Another advantage could be the use of snapshots in virtual

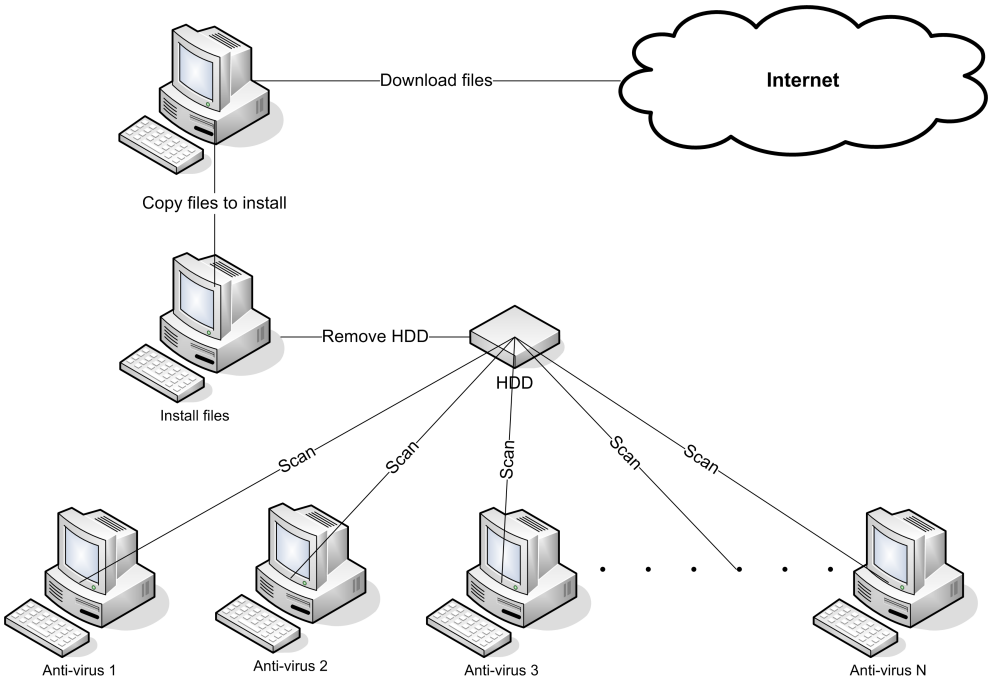


Figure 7.4: An alternative method for downloading, installing and scanning potential malware.

machines, if an executed file causes too much problems it is easy to revert to a safe state by using this functionality.

While we originally had planned to use desktop anti-virus software as a basis for our results, this proved difficult because of the machines getting infected. The alternative method presented would eliminate the risk of not being able to use the desktop anti-virus software, because of the "desktop offline-scanning". It would also be easier to control what files are downloaded, as this would only be done on one machine. A disadvantage is the loss of the real-world aspect, this is especially important when surfing the web as many infections can originate from drive-by-downloads or by using exploits in the web browser.

7.10 Further Work

In this section we suggest some topics that can be further investigated. These topics are either out of scope of our project or omitted due to time constraints.

7.10.1 Zero-day Exploits

As this project is focusing on zero-day *malware*, another possibility is to perform a similar experiment with the focus on zero-day *exploits*.

Trading of zero-day exploits is another possible topic. To be able to investigate this an infiltration to the black hat community should be done. There are malware creators who want to make money by trading their malware.

7.10.2 Crawler

With a web crawler, we would have covered a greater amount of web sites. This could have eased the process of surfing the web and also possibly increased the amount of malware detections. On the other hand this would reduce the real world aspect.

Another type of crawler can automate the process of downloading files from file-sharing networks. On the other hand this leads to less control with the files to be downloaded. We refer to [SJB06] and [KAG06] for a more detailed description of such crawlers.

7.10.3 Other Systems and Applications

Although we have chosen to only use Windows XP and Windows Vista in our project, it is possible to perform a similar experiment using other operating systems like Mac OS X, Linux and the new Windows 7. As we only included one Windows Vista machine, this experiment could also be carried out using Windows Vista machines only.

Other applications, like for example Adobe and Microsoft Office, could also have been installed in order to try to get infected with zero-day malware. As already mentioned, the trend reports show that malware creators are about to move their focus towards applications instead of exploiting the operating system.

7.10.4 Spam

As already mentioned we created an email account to hopefully receive emails containing spam. We received over 1600 mails in our account, but due to time constraints we did not investigate these any further. The reason why we created this account was to follow the links given in the emails and hopefully get infected with zero-day malware. As this was not performed, this is left to further work.

7.10.5 Deeper Analysis of Zero-day Malware

In this project we have only given a short description of some of the malware instances encountered during the experiment. A more thorough investigation of specific malware instances could be carried out, like map the behavior of the malware in a controlled environment to see explicitly what it does to the computer system.

Chapter 8

Conclusion

In this project a method for discovering zero-day malware and mapping the prevalence of such malware has been presented and put into practice. The technique has proven to be a feasible and valid approach for this purpose, and has been carried out manually in a laboratory consisting of six computers. This approach gave the project a real-life aspect that would not have been achieved using an automatic approach utilizing tools like crawlers and virtual machines.

This work started with a theoretical study of malicious software with emphasize on zero-day malware. The architecture and the role of file-sharing networks as malware sources have also been investigated thoroughly. This led to the development of a testbed where the computers were exposed to potentially malicious sources on the Internet, including file-sharing networks and the world wide web. After this exposure phase a baseline scan with offline anti-virus software was performed, before all the computers were shut down for a month. A final offline scan was then carried out and compared to the baseline scan. Malware that was only detected at the final scan was, according to our definition, labeled as zero-day malware.

The comparison showed that the computers in our experiment were infected by zero-day malware to an extent far greater than expected. A total of 124 zero-day malware instances were identified. Out of these, 47% originated from BitTorrent while the percentage from Gnutella constituted only 7%, even though the number of downloaded files was far higher in Gnutella. We believe that this major difference is due to the fact that retrieval of *new* files is much easier in BitTorrent than in Gnutella. However, the amount of non-zero-day malware in Gnutella was disturbingly high, estimated to 70-80%. The same number for BitTorrent was approximately 25%. A significant amount of zero-day malware was also detected in files downloaded from the web, but the number of obtained files from this source was much less than from the file-sharing networks. In any case, these results have shown that zero-day malware exist in both file-sharing networks and on the web.

The prevalence of zero-day malware implies that anti-virus software, which mainly relies on signatures, does not provide sufficient protection. Coupled with the exponential growth of new malware variants, the anti-virus vendors have big problems keeping the signature lag as short as possible. An anti-virus senior manager have referred to this as *"a denial of service [attack] against our labs"* [Jaq07], and the trend is predicted to continue into 2009.

Examining the time of signature updates by F-Secure, with respect to the zero-day malware found in our experiment, gave us an indication that the signature lag varies from a few days to over one month. It is not unreasonable to believe that other anti-virus vendors have comparable characteristics. Trying to estimate the size of the vulnerability window is not an easy task, because of the uncertainty of when a piece of malware surfaced on the Internet. Per definition the vulnerability window is larger than the signature lag, thus it is not unlikely that for a significant amount of malware the vulnerability window of some systems is in the order of weeks to months. One must assume that anti-virus vendors have some sort of priority, and that malware which spreads very rapidly and/or has particularly malicious payload will receive a signature before more harmless variants. This means that for such malware the vulnerability window will probably be significantly smaller.

One of the computers in our experiment was installed with the new operating system Windows Vista. This system did not exhibit any signs of malware infection, as opposed to the rest of our computers which were severely infected with a lot of different malware. This would suggest that Windows Vista is a more secure operating system than Windows XP, most likely because of the new and improved security features such as User Access Control. Another reason for this difference could be that malware creators are still targeting Windows XP to a larger extent.

The results of our experiment have shown that zero-day malware is a real threat and is more prevalent than first assumed. A consequence of this is that even with updated anti-virus software, a user is not immune to infections. Cautious use of the Internet reduces the risk of malware infections, but even the most prudent user is not safe. The signature-based approach to malware protection will always suffer from a signature lag, this has been more evident with the recent exponential growth of new malware. Even though most modern anti-virus solutions incorporate some forms of proactive detection mechanisms, like for instance heuristics, they are still dependent on signatures. No current solution is able to protect against all new threats, and a viable and effective alternative to current techniques is seriously needed.

Bibliography

[AFM00] William A. Arbaugh, William L. Fithen, and John McHugh. *Windows Of Vulnerability: A Case Study Analysis. Computer*, pages 52–55, 2000.

[Ber08] Andrew Berns. *Searching for Malware in BitTorrent. Computer Security Presentation at The University of Iowa*, April 29, 2008. http://www.cs.uiowa.edu/~ejjung/courses/169/project/publish/AndrewBerns_presentation.pdf.

[Bha08] Rijubrata Bhaumik. *Peer to Peer content delivery in the Internet (history , current systems). Helsinki University of Technology*, Last accessed October 16, 2008. www.tml.tkk.fi/Opinnot/T-110.6120/2007/fall/p2p.pdf.

[Blo06] Robin Bloor. *Anti-Virus Is Dead: The Advent of the Graylist Approach to Computer Protection. Hurwitz & Associates*, 2006. http://www.bit9.com/files/wp-2006-Bit9-Anti-Virus-is-Dead.pdf.

[CJ04] Mihai Christodorescu and Somesh Jha. Testing malware detectors. *SIGSOFT Softw. Eng. Notes*, 29(4):34–44, 2004.

[CJS+05] Mihai Christodorescu, Somesh Jha, Sanjit A. Seshia, Dawn Song, and Randal E. Bryant. Semantics-aware malware detection. *Security and Privacy, IEEE Symposium on*, 0:32–46, 2005.

[Cle08] Andreas Clementi. *Anti-Virus Comparative No.18: Proactive/retrospective test. AV-Comparative*, May 31, 2008.

[Coh87] Fred Cohen. Computer viruses : Theory and experiments. *Computers & Security*, 6(1):22 – 35, 1987.

[Cor02] Symantec Corporation. *Understanding Heuristics: Symantec's Bloodhound Technology. Symantec White Paper Series Vol. XXXIV)*, March 15, 2002.

[HB06] Greg Hoglund and James Butler. In *Rootkits: Subverting the Windows kernel*, page 4, USA, 2006. Pearson Education, Inc.

[HM04] Greg Hoglund and Gary McGraw. *Exploiting Software: How to break code.* Addison-Wesley, 2004.

[Ins07] SANS Institute. *SANS Top-20 2007 Security Risks. SANS Institute - Network, Security, Computer, Audit Infomation & Training*, November 28, 2007.

[Jaq07] Andrew Jaquith. *Anti-Virus Is Dead; Long Live Anti-Malware. Yankee Group*, January 17, 2007.
http://www.yankeegroup.com/ResearchDocument.do?id=15059.

[Jen08] Jostein Jensen. A novel testbed for detection of malicious software functionality. In *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, pages 292–301, Washington, DC, USA, 2008. IEEE Computer Society.

[KAG06] Andrew Kalafut, Abhinav Acharya, and Minaxi Gupta. A study of malware in peer-to-peer networks. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 327–332, New York, NY, USA, 2006. ACM.

[KE03] Darrell M. Kienzle and Matthew C. Elder. Recent worms: a survey and trends. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 1–10, New York, NY, USA, 2003. ACM.

[Kea08] Shane Keats. *Mapping the Mal Web Revisited. McAfee SiteAdvisor*, June 4, 2008.
http://www.siteadvisor.com/studies/map_malweb_jun2008.pdf.

[Lev08] Amir Lev. *The Marriage Of Spam And Malware: Implication For SMTP Malware Defence. commtouch.co.kr*, Last accessed October 16, 2008.
http://www.commtouch.co.kr/downloads/MarriageofMalwareandSpamVB2007.pdf.

[MKK07] A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 421–430, Dec. 2007.

[Nat01] Kurt Natvig. Sandbox technology inside av scanners. In *Proceedings of the 2001 Virus Bulletin Conference*, pages 1–18. Virus Bulletin, September 2001.

[Per08] David M. Perry. *The (Life and) Death of the Pattern File. Trend Micro, Virus Bulletin Conference*, October, 2008.
http://us.trendmicro.com/imperia/md/content/us/pdf/threats/securitylibrary/perry-vb2008.pdf.

[PMRM08] Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose. All Your iFRAMEs Point to Us. In *Proceedings of the 17th USENIX Security Symposium*, 2008.

[RS07] Edward Ray and E. Eugene Schultz. An early look at windows vista security. *Computer Fraud & Security*, 2007(1):4 – 7, 2007.

[Rus08] Vyacheslav Rusakoff. *Win32.Ntldrbot (aka Rustock.C) no longer a myth, no longer a threat. Doctor Web, Ltd.*, May, 2008.
http://www.drweb.com/upload/6c5e138f917290cb99224a8f8226354f_1210062403_DDOCUMENTSArticales_PRDrWEB_RustockC_eng.pdf.

[SJB06] Seungwon Shin, Jaeyeon Jung, and Hari Balakrishnan. Malware prevalence in the kazaa file-sharing network. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 333–338, New York, NY, USA, 2006. ACM.

[SSW08] David Sharek, Cameron Swofford, and Michael Wogalter. Failure to recognize fake internet popup warning messages. In *Proceedings of the Human Factors and Ergonomics Society 52nd Annual Meeting-2008*, pages 557–560, Raleigh, NC 27695-7650 USA, 2008.

[Sta06] William Stallings. *Cryptography and Network Security: Principles and Practices*. Prentice Hall, 4th edition, 2006.

[SZ04] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall, 2004.

[Szo05] Peter Szor. *The Art of Computer Virus Research and Defense*. Addison-Wesley, 2005.

[tag07] Vista security verdicts roll in. *Network Security*, 2007(3):1 – 2, 2007.

[WBMW04] Cynthia Wong, Stan Bielski, Jonathan M. McCune, and Chenxi Wang. A study of mass-mailing worms. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 1–10, New York, NY, USA, 2004. ACM.

Web References

[1] ALWIL Software a.s. *Avast! BART CD Download. avast.com*, Last accessed December 9, 2008.
http://www.avast.com/eng/download-avast-bart-cd.html.

[2] ALWIL Software a.s. *Avast! Home Edition. avast.com*, Last accessed October 14, 2008.
http://www.avast.com/eng/avast_4_home.html.

[3] Norman ASA. *Submit file for SandBox analysis. Norman SandBox Information Center*, Last accessed November 20, 2008.
http://www.norman.com/microsites/nsic/Submit/en.

[4] Norman ASA. *Norman Security Suite. norman.com*, Last accessed October 14, 2008.
http://www.norman.com.

[5] F-Secure AVUpdate. *Virus definition updates. F-Secure Forum*, Last accessed November 13, 2008.
http://forum.f-secure.com/forum.asp?FORUM_ID=12.

[6] Eric Bangeman. *Study: BitTorrent sees big growth, LimeWire still #1 P2P app. ars technica*, April 21 2008.
http://preview.tinyurl.com/3recfp.

[7] BTJunkie. *BTJunkie - the largest bittorrent search engine. BTJunkie Home Page*, Last accessed November 17, 2008.
http://btjunkie.org.

[8] Daniel Chipiristeanu. *Trojan.Downloader.WMA.Wimad. Bitdefender.com*, Last accessed November 12, 2008.
http://www.bitdefender.com/VIRUS-1000317-en--Trojan.Downloader.WMA.Wimad.html.

[9] Cnet. *Most popular windows downloads. Download.com*, Last accessed September 20, 2008.
http://www.download.com/3101-2001_4-0.html?tag=mncol;sort.

[10] Bram Cohen. *The BitTorrent Protocol Specification. Bittorrent.org*, Last accessed October 13, 2008.
http://www.bittorrent.org/beps/bep_0003.html.

[11] F-Secure Corporation. *F-Secure Spyware Information Pages: Adware. f-secure.com*, Last accessed November 2, 2008.
http://www.f-secure.com/sw-desc/adware.shtml.

[12] Symantec Corporation. *Symantec.com Security Response AntiVirusXP2008. symantec.com*, Last accessed November 2, 2008.
http://www.symantec.com/security_response/writeup.jsp?docid=2008-071613-4343-99&tabid=2.

[13] Symantec Corporation. *Symantec Home Page. symantec.com*, Last accessed October 14, 2008.
http://www.symantec.com/index.jsp.

[14] The Tech Terms Computer Dictionary. *Zero Day Exploit. techterms.com*, Last accessed October 22, 2008.
http://www.techterms.com/definition/zerodayexploit.

[15] Ann Kristin Bentzen Ernes. *Virus rammer fortsatt Norges største bank (In Norwegian). Digi.no*, March 12, 2007.
http://www.digi.no/php/art.php?id=372788.

[16] F-Secure. *F-Secure Malware Information Pages: Zlob. F-Secure.com*, Last accessed December 09, 2008.
http://www.f-secure.com/v-descs/zlob.shtml.

[17] F-Secure. *F-Secure Spyware Information Pages: Virtumonde. F-Secure.com*, Last accessed December 09, 2008.
http://www.f-secure.com/sw-desc/virtumonde.shtml.

[18] F-Secure. *F-Secure Virus Descriptions : Trojan-Downloader. F-Secure.com*, Last accessed December 09, 2008.
http://www.f-secure.com/v-descs/trojdown.shtml.

[19] F-Secure. *F-Secure Virus Descriptions : Trojan-Dropper. F-Secure.com*, Last accessed December 09, 2008.
http://www.f-secure.com/v-descs/trojdrop.shtml.

[20] F-Secure. *F-Secure IT Security Threat Summary for the Second Half of 2008. F-Secure Data Security Wrap-up*, Last accessed December 5, 2008.
http://www.f-secure.com/2008/2/index.html.

[21] F-Secure. *F-Secure Virus Descriptions : Brain. F-Secure Home Page*, Last accessed October 08, 2008.
http://www.f-secure.com/v-descs/brain.shtml.

[22] F-Secure. *Antivirus Software and Internet Security Tools - F-Secure. F-Secure*, Last accessed September 20, 2008.
http://www.fsecure.com.

[23] Fairdell. *Downloads Free. Fairdell.com*, Last accessed December 10, 2008.
http://www.fairdell.com/downloads/.

[24] Paul Festa. *See you later, anti-Gators? CNET News*, Last accessed November 2, 2008.
http://news.cnet.com/2100-1032_3-5095051.html.

[25] Harvard Law School Berkman Center for Internet and Society. *StopBadware.org - Kazaa. StopBadware.org*, Last accessed October 17, 2008.
http://www.stopbadware.org/reports/reportdisplay?reportname=kazaa.

[26] Aleks Gostev. *The darker side of online virus scanners. Analyst's Diary*, Last accessed November 19, 2008.
http://www.viruslist.com/en/weblog?weblogid=208187473.

[27] Alexander Gostev. *Rustock and All That. Viruslist.com*, July 15, 2008.
http://www.viruslist.com/en/analysis?pubid=204792011.

[28] Kaoru Hayashi. *Trojan.Wimad. Symantec.com*, Last accessed November 12, 2008.
http://www.symantec.com/security_response/writeup.jsp?docid=2005-011213-2709-99.

[29] iaria.org. *Call for Papers. ICIMP 2009*, Last accessed December 14, 2008.
http://www.iaria.org/conferences2009/CfPICIMP09.html.

[30] Trend Micro Incorporated. *Adware. Trend Micro Europe, Middle East and Africa*, Last accessed November 2, 2008.
http://emea.trendmicro.com/emea/threats/enterprise/glossary/a/adware/index.php.

[31] FRISK Software International. *F-PROT Antivirus. f-prot.com*, Last accessed December 9, 2008.
http://www.f-prot.com.

[32] What is what.com. *What is a Zero-Day Exploit? what-is-what.com*, Last accessed October 22, 2008.
http://what-is-what.com/what_is/zero_day_exploit.html.

[33] Neil Johnson. *Virus scan Windows using a Linux live CD. Neil's Open Source & Linux Blog*, Last accessed September 15, 2008.
http://njlinux.blogspot.com/2008/01/virus-scan-windows-using-linux-live-cd.html.

[34] Leif Martin Kirknes. *Trojaner p MSN.no. Computerworld (In Norwegian)*, Last accessed October 31, 2008.
http://www.idg.no/computerworld/article104791.ece.

[35] Tor Klingberg and Raphael Manfredi. *Gnutella Protocol Development, Gnutella 0.6. Network Working Group*, Last accessed October 08, 2008.
http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[36] Lime Wire LLC. *Limewire Glossary. Limewire Homepage*, Last accessed October 08, 2008.
http://www.limewire.com/about/glossary.php.

[37] LimeWire LLC. *Gnutella Protocol Specification. LimeWire.org*, Last accessed October 13, 2008.
http://wiki.limewire.org/index.php?title=GDF.

[38] LimeWire LLC. *Limewire home page*. *LimeWire.com*, Last accessed October 13, 2008.
http://www.limewire.com.

[39] FraudWatch International Pty Ltd. *Phishing Web Site Methods* . *FraudWatch International*, Last accessed October 31, 2008.
http://www.fraudwatchinternational.com/phishing-fraud/phishing-web-site-methods/.

[40] Safer Networking Ltd. *Home of Spybot-S&D*. *safer-networking.org*, Last accessed October 14, 2008.
http://www.safer-networking.org/en/home/index.html.

[41] Ellen Messmer. *Has the end arrived for desktop antivirus?* *Network World*, Last accessed November 17, 2008.
http://www.networkworld.com/news/2007/040507-desktop-antivirus-dead.html?page=1.

[42] Microsoft. *Virus Encyclopedia: TrojanDownloader:ASX/Wimad.AB*. *Windows Live OneCare - Virus Encyclopedia*, Last accessed November 19, 2008.
http://onecare.live.com/standard/en-us/virusenc/VirusEncInfo.htm?VirusName=TrojanDownloader:ASX/Wimad.AB.

[43] Mircosoft. *Microsoft Security Intelligence Report volume 5*. *Microsoft.com*, November 14, 2008.
http://www.microsoft.com/downloads/details.aspx?FamilyId=B2984562-47A2-48FF-890C-EDBEB8A0764C&displaylang=en.

[44] Sharman Networks. *KaZaA homepage*. *kazaa.com*, Last accessed October 17, 2008.
http://kazaa.com/us/index.htm.

[45] Symantec Norton. *Glossary*. *Norton, Symantec*, Last accessed October 08, 2008.
http://www.symantec.com/norton/security_response/glossary.jsp#e.

[46] Dan Nunes and Shane Keats. *Mapping the Mal Web*. *McAfee SiteAdvisor*, March 12, 2007.
http://www.siteadvisor.com/studies/map_malweb_mar2007.html.

[47] PandaLabs. *30 million computers are infected by fake antivirus programs generating profits for cyber-crooks of more than €10 million every month*. *Panda Security*, October 15, 2008.
http://www.pandasecurity.com/homeusers/media/press-releases/viewnews?noticia=9394.

[48] Secunia. *Microsoft Windows WMF "SETABORTPROC" Arbitrary Code Execution*. *Secunia Advisories*, February 28, 2006.
http://secunia.com/advisories/18255.

[49] About.com: Internet/Network Security. *What Is A Rootkit?* *netsecurity.about.com*, Last accessed November 12, 2008.
http://netsecurity.about.com/od/frequentlyaskedquestions/f/faq_rootkit.htm.

[50] About.com: Internet/Network Security. *What Is A Bot? netsecurity.about.com*, Last accessed October 20, 2008.
http://netsecurity.about.com/od/frequentlyaskedquestions/qt/pr_bot.htm.

[51] Shadowserver. *Malware. shadowserver.org*, Last accessed November 12, 2008.
http://www.shadowserver.org/wiki/pmwiki.php?n=Information.Malware.

[52] Shadowserver. *Botnets. shadowserver.org*, Last accessed October 21, 2008.
http://www.shadowserver.org/wiki/pmwiki.php?n=Information.Botnets.

[53] Shadowserver. *Bot Count Yearly. shadowserver.org*, Last accessed October 31, 2008.
http://www.shadowserver.org/wiki/pmwiki.php?n=Stats.BotCountYearly.

[54] Shadowserver. *Shadowserver homepage. shadowserver.org*, Last accessed October 31, 2008.
http://www.shadowserver.org.

[55] Hispasec Sistemas. *Deleting the option "Do not distribute the sample". VirusTotal Blog*, Last accessed November 19, 2008.
http://blog.hispasec.com/virustotal/28.

[56] AVG Technologies. *AVG Anti-Virus. avg.com*, Last accessed October 14, 2008.
http://www.avg.com.

[57] Katrin Tocheva, Mikko Hypponen, and Sami Rautiainen. *F-Secure Virus Descriptions : Melissa. F-Secure*, Last accessed October 23, 2008.
http://www.f-secure.com/v-descs/melissa.shtml.

[58] Mika Tolvanen. *New WMF 0-day exploit. F-Secure Weblog*, December 28, 2005.
http://www.f-secure.com/weblog/archives/archive-122005.html#00000752.

[59] Liam Tung. *Norton 2009 tackles whitelisting. ZDNet.com.au*, Last accessed November 17, 2008.
http://www.zdnet.com.au/news/security/soa/Norton-2009-tackles-whitelisting/0,130061744,339291814,00.htm?feed=pt_norton.

[60] uTorrent. *uTorrent Home Page. uTorrent.com*, Last accessed October 14, 2008.
http://utorrent.com.

[61] VirusTotal. *VirusTotal - Free Online Virus and Malware Scan. VirusTotal.com*, Last accessed November 10, 2008.
http://www.virustotal.com.

[62] VirusTotal. *About VirusTotal. VirusTotal.com*, Last accessed November 12, 2008.
http://www.virustotal.com/sobre.html.

Appendix A

Suspicious Sites

This was our initial list of suspicious web sites. It is a mixture of sites which contain warez, screensavers, codecs, mp3s and other free downloads. Since web sites from Romania, Hong Kong and Russia are considered most risky [46] [Kea08], we tried to include some sites from those countries.

It is worth mentioning that from each of these web sites, we clicked on a lot of ads, links and partner sites. By counting the different URLs in the history file of Internet Explorer, the total number of visited web sites are estimated to nearly 300. The complete list can be found on the CD described in Appendix E.

http://www.rocketdownload.com/software/warez.html
http://www.fullforum.com
http://flmsdown.net/category/software
http://crackfind.com/top/index.php?a_m=1
http://www.onlyscreensavers.com/Download_657.html
http://www.download3000.com
http://www.toggle.com/lv/group/view/kl34320/Passware_Kit.htm
http://www.winpcware.com/system25.htm
http://www.astalavista.com
http://www.astalavista.box.sk
http://www.andr.net
http://www.crackz.ws
http://www.cracks.cd
http://www.bittorrent.am
http://www.astalavista.am
http://www.crackhell.com
http://www.fulldownloads.us
http://www.wareznova.com
http://www.downloadwarez.org
http://www.xdownloads.biz
http://www.xxx.ru
http://allofmp3.pp.ru
http://musicmp3.ru
http://www.freealtonia.ru
http://www.malicious.biz

http://www.downloadporn.biz
http://www.worldbbs.info
http://xaboom.info
http://www.fuckzone.info
http://www.freeze.com
http://www.scenicreflections.com/cat_id=3
http://www.8screensavers.com
http://www.acez.com
http://free-patriotic-screensavers.com
http://www.screensavers.com
http://www.jookz.com
http://www.aaascreensavers.com
http://www.screensaverfree.com
http://www.free-screen-savers.to
http://thanksgiving-screensavers.info
http://www.fast-download.info
http://download-free-screensavers.info
http://www.divxmovies.com/codec
http://www.downloadjunction.com/product/freeware/2245/index.html
http://www.avicodecpack.com
http://www.anunturirapide.ro/cauta2.php?q=free%20download%20fifa2008&page=2
http://www.kujuso.ro/download.htm
http://www.indexsite.ro/Underground/Download
http://www.frekeno.0x.ro
http://www.studioline.ro
http://boyfriend.com.hk
http://3mp3.ru/eng
http://tagoo.ru/en
http://tabyk.ykt.ru
http://www.asfo.ru/forum/viewtopic.php?p=13707&sid=9bebd3fcfd67ed5c52bf6270e7854e1d
http://www.10ue.ru
http://conference.apex.auto.ru/dialog/wwwboard/30830.shtml
http://www.windowsxpprofessional.windowsreinstall.com
http://www.taniewizytowki.com/js/2/blog.php?p=21767
http://free-key-gen-archive.net
http://upgrade-your-software.com
http://quicksoftupdate.com

Appendix B

Offline Anti-virus CDs

Here are step-by-step descriptions for obtaining and using Avast! BART CD and Knoppix Live Linux CD/F-PROT.

B.1 Avast! BART CD

The Avast! Bootable Antivirus & Recovery Tool (BART) CD is a bootable CD, giving you the capability to detect and remove virus infections without the risk of spreading the infection any further.

B.1.1 How To Get The CD

Avast! BART CD can be obtained from the homepage of Avast! [1]. By filling out a form, you will get a 14 day trial license.

As mentioned in the email you will receive, the following steps are necessary to create an updated version of the boot CD.

1. Run the downloaded `BART_FULL.EXE` program and when asked, use the license file attached to the received email (unzip it first).
2. After finishing the installation, start Avast! BART CD Manager and choose *Update and Generate ISO* to create the ISO image.
3. Burn the image to a CD with your favorite CD burning software.

B.1.2 Offline Scan Instructions

1. Boot the computer with updated BART CD
2. Select Avast! Anti-virus from the Security Tools menu, click *Run Now*
3. In the *Select areas to scan*, choose *Local hard disks* where your OS is installed
4. Set scan level to *Thorough* and use the default virus database
5. Mark all archive types and choose a folder for temporary files, for example `C:\WINDOWS\Temp\`

6. Select yes when asked about creating a report file for the current scan

7. Select every report parameter except *OK files* and choose a location for the file, for example on a floppy disk or a memory stick

B.2 Knoppix and F-PROT

Knoppix is a Live CD Linux distribution which can be booted from a CD, without using or modifying the harddrives of a computer. This makes Knoppix a good choice for offline virus scanning. F-PROT is a popular and free anti-virus package which is available for Linux and other systems. We used Knoppix version 5.1.1, but any version of Knoppix and any other Live CD Linux distribution should be usable. For F-PROT we used the latest version available at the time of our project, 6.0.2. The instructions below are based on a guide [33] found on the web and our own knowledge of Linux based systems.

B.2.1 F-PROT AntiVirus Offline Scan Instructions

1. Boot Knoppix Live-CD Linux

   - Knoppix can be downloaded from `www.knoppix.net`.

2. Configure the network adapter

   - Start-menu - Knoppix - Network/Internet - Network card configuration
   - Follow the on-screen instructions to set up your network adapter

3. Download F-PROT (GZIP-ed TAR-file) from `http://www.f-prot.com/download/home_user/download_fplinux.html`

4. Unpack the file, and start the install script

   - `cd f-prot`
   - `sudo ./install-f-prot.pl`
   - Choose the default options and wait for F-PROT to download the latest virus definitions

5. Mount the windows drive by clicking on it on the desktop

6. Run the command:

   - `fpscan -s 4 /media/name-of-disk -o /home/knoppix/Desktop/name-of-logfile.log`
   - `name-of-disk` is your windows disk, i.e hda1, and `name_of_logfile.log` is your log file. Also remember to save the log file to persistent storage before exiting Knoppix.

B.2.2 Troubleshooting

During our experience with Knoppix and F-PROT, we encountered some problems. On some of the computers the `fpscan` command would exit and display the message `Killed` in the console. The log file did not give us any good information about the issue, as it just ended in the middle of a file-name. After trying to delete the files in question with no improvement, we concluded that it had to be a memory-size issue. As one of the computers that finished without problems had 1 GB of memory and most of the others only had 512 MB.

The easiest way to free up memory in a Live CD Linux environment is to exit the graphical user interface, X Windows system, and use the machine from console only. The most practical way to do this is to first complete steps 1 through 5 and then type the command `sudo init 3` in the console to exit the graphical user interface. Then continue with step 6 to complete the anti-virus scan, when the scan finishes type `init 5` to restart X Windows.

Appendix C

List of Zero-day Malware

The first table lists all the 124 zero-day malware found in the project according to Avast! BART and/or F-PROT. The blank cells indicate that the anti-virus program did not detect the specific file to be malware, while the few cells marked with the prefix *Not-Zero* and written in italic, are instances that were only zero-day according to one of the two anti-virus engines.

Next we present a list of the same files, but with detection names given by F-Secure and Symantec. This was achieved by uploading all files to VirusTotal. F-Secure was the only vendor that offered decent information about when they added the signatures, thus we have included a column that shows the date F-Secure first was able to detect the malware.

C.1 Avast! BART and F-PROT Zero-day Malware

File name	**Avast! BART**	**F-PROT**	**VT**
Download_ad_Savannah_Camp-_TRIAL.exe	Win32:FraudTool-JP [Tool]		03/36
Download_AD_Savannah_Safari-_TRIAL.exe	Win32:FraudTool-JP [Tool]		03/36
Download_ss_Savannah_Camp-_TRIAL.exe	Win32:FraudTool-JP [Tool]		03/36
Autodesk.Revit.Architecture.2008-.Full.Version.with.Keygen.zip	Win32:Trojan-gen {Other}		05/36
NodLogin9.4_64bits.rar	Win32:Trojan-gen {Other}		05/36
Easy Video Downloader 1.1 2008 fxg.rar	Win32:Trojan-gen {Other}		06/36
MP3_Cutter_Joiner_v2.20.zip	Win32:Trojan-gen {Other}		06/36
zopt.exe	Win32:Adware-gen [Adw]		07/36
Microsoft Streets & Trips 2008 Activated.exe	Win32:Trojan-gen {Other}		07/36
WordPress Theme – Revolution Magazine.iso	Win32:VB-KJE [Trj]		07/36
Firefox 3.exe	Win32:Trojan-gen {Other}		08/36
MagicISO Maker v5.5.0261 - inAcrysis.rar		W32\Backdoor2.DASK	08/36

continued on next page

continued from previous page

File name	Avast! BART	F-PROT	VT
Nero 8.2.8.0 Lite.Inc.Key.rar/keygen.exe	Win32:Hacktool-AU [Tool]		08/36
SoundBooth CS3.exe	Win32:Trojan-gen {Other}		08/36
Trillian Astra Alpha 4.0.0.83 [Released July 18 2008]LATEST.rar		W32\Downldr2.EDDI	09/36
WinZip_12_Keygen.exe	Win32:Trojan-gen {Other}		09/36
All.Codecs.For.Windows.Media-.Player.rar		W32\Dropper.ACTG	09/36
keygen.exe	Win32:Trojan-gen {Other}		09/36
HirensBootCD.9.5.rar/Shredder.exe	Win32:Trojan-gen {Other}		09/36
_freescan[1].htm		JS\FakeAV.A	10/36
A0005385.dll	Win32:Trojan-gen {Other}		10/36
slphseeccn.dll	Win32:Trojan-gen {Other}		10/36
Flash.Player.Pro.v3.7.Cracked.rar		W32\Dropper.ACTG	10/36
Jookz.com_Bangin_Babes-_Screensaver.exe		W32\Malware!0409	10/36
Magic ISO Maker 5.4 Build 239.exe		W32\Trojan2.EKDB	10/36
The Cleaner S01E02 HDTV XviD-2HD.zip	WMA:Wimad [Drp]		10/36
Trillian Pro 3.1.10.0 [Skin + Plugin Packs].rar		W32\Downldr2.EDDI	11/36
Eset NOD32 Antivirus + Lifetime Updates.rar	Win32:Trojan-gen {Other}		11/36
DivXInstaller_1.exe		W32\Downldr2.EMMK	11/36
DivXInstaller.exe		W32\Trojan2.EFWE	11/36
10[1].exe	Win32:Trojan-gen {Other}		11/36
13_3.ex_	Win32:Trojan-gen {Other}		11/36
HirensBootCD.9.5.rar-/WinKeyFinder.exe	Win32:Trojan-gen {Other}		11/36
YoutubeGet 4.7 [Youtube Video Downloader].rar		W32\Downldr2.EDDI	12/36
DVDFab5085.exe		W32\Trojan2.EJVD	12/36
VW81DE2V.dll	Win32:Trojan-gen {Other}		12/36
46[1].exe	Win32:Trojan-gen {Other}		12/36
HirensBootCD.9.5.rar-/StartupMonitor.exe	Win32:Trojan-gen {Other}		12/36
HirensBootCD.9.5.rar/uharcd.exe	Win32:Trojan-gen {Other}		12/36
Keymaker.exe		W32\Downldr2.EMMK	13/36
13_1.ex_	Win32:Trojan-gen {Other}		13/36
13.ex_	Win32:Trojan-gen {Other}		13/36
HirensBootCD.9.5.rar/KillBox.exe	Win32:Trojan-gen {Other}		13/36
Keymaker.Nero.8.Ultra-.Edition.v8.3.6.0.rar	Win32:Hacktool-AU [Tool]		13/36
HirensBootCD.9.5.rar/VDefs.exe	Win32:Trojan-gen {Other}		14/36
HirensBootCD.9.5.rar-/DefragNT.exe	Win32:Trojan-gen {Other}		14/36
vlc sexy girl has shaking orgasm during sex.mpg	WMA:Wimad [Drp]		14/36
Justin Timberlake-Recrimination-2CD-2008 [www TorrentMas com].zip	WMA:Wimad [Drp]		14/36

continued on next page

continued from previous page

File name	Avast! BART	F-PROT	VT
WinRAR 3.7 Full Corporate Edition.rar		W32\Dropper.ACTG	14/36
PC Tank Assault _Attack 3.9.rar		W32\Trojan2.EFWE	14/36
Space game-Galaxy Invaders.rar		W32\Trojan2.EFWE	14/36
A0006004.dll	Win32:Adware-gen [Adw]		14/36
A0007323.dll	Win32:Adware-gen [Adw]		14/36
Nero 8.3 Ultra Keygen.rar		W32\Backdoor2.CXHP	15/36
13_4.ex_	Win32:Trojan-gen {Other}		15/36
HirensBootCD.9.5.rar-/DriverBackup.exe	Win32:Trojan-gen {Other}		15/36
HirensBootCD.9.5.rar-/ShExView.exe	Win32:Trojan-gen {Other}		15/36
HirensBootCD.9.5.rar/xp-AntiSpy.exe	Win32:Trojan-gen {Other}		15/36
HirensBootCD.9.5.rar-/IBProcMan.exe	Win32:Trojan-gen {Other}		15/36
WinRAR Corporate Special Edition 3.7.1 [Zipped Final Edition].rar		W32\Backdoor2.DAUG	15/36
HirensBootCD.9.5.rar/Splitter.exe	Win32:Trojan-gen {Other}		16/36
Magic ISO Maker 5_1.4 Build 239.exe	Win32:Fabot [Trj]		16/36
file[1].exe		W32\FakeAlert.3!Generic	16/36
msxml71.dll	Win32:Trojan-gen {Other}		16/36
WinRAR 3.71 Corporate.EXE		W32\Backdoor2.CXGS	17/36
NeuView Media Player Professional v6.0.8.0253 [Patch].rar		W32\Downldr2.EDDI	17/36
HirensBootCD.9.5.rar/Autorun.exe	Win32:Trojan-gen {Other}	W32\Downldr2.ESSB	17/36
cntr[1]	Win32:Trojan-gen {Other}		17/36
CleanerInstaller_no.exe		W32\FakeAV2008.DR	18/36
Adult Sexy BABY TOY Game-PC.rar		W32\Trojan2.EFWE	18/36
smchk.exe	Win32:Trojan-gen {Other}		18/36
urqPhgff.dll	Win32:Trojan-gen {Other}	*Not-Zero: W32/Virtumonde.AC-.gen!Eldorado*	18/36
Flash.Player.Pro.v3.7.WinALL-.Cracked-CzW.rar	*Not-Zero: Win32:Trojan-gen {Other}*	W32\Downldr2.ECWW	19/36
HiHiSoft.Youtube.Download-.v4.8.5.Regged-F4CG.rar	*Not-Zero: Win32:Trojan-gen {Other}*	W32\Downldr2.ECWW	19/36
byXOgfFV.dll	Win32:Trojan-gen {Other}	*Not-Zero: W32/Virtumonde.AC-.gen!Eldorado*	19/36
file[1]_1.exe	Win32:Trojan-gen {Other}		19/36
qoMccYqq.dll	Win32:Trojan-gen {Other}	*Not-Zero: W32/Virtumonde.AC-.gen!Eldorado*	19/36
De31.exe/is159287.exe	Win32:Trojan-gen {Other}		20/36
YUR1.exe	Win32:Trojan-gen {Other}		20/36
mIRC 634.zip	Win32:Trojan-gen {Other}		21/36
only date ballers.KeyGen.All_Version.zip	Win32:Downloader-BVY [Trj]	*Not-Zero: W32/Downloader-Sml-based!Maximus*	21/36

continued on next page

continued from previous page

File name	Avast! BART	F-PROT	VT
Setup_ver1.1482.0.exe	Win32:Trojan-gen {Other}		22/36
efcAQKEt.dll	Win32:Trojan-gen {Other}	*Not-Zero: W32/Virtumonde.AC-.gen!Eldorado*	22/36
dfmlxbpkwqd.dll	Win32:Adware-gen [Adw]		22/36
HDVideoCodec_ver1.6050.0.exe	Win32:Trojan-gen {Other}	W32\Downldr2.ERWI	22/36
tdssserv.sys	Win32:Rootkit-gen [Rtk]		23/36
vlc-0.8.6d-win32-.exe	Win32:Trojan-gen {Other}	W32\Downldr2.EOKQ	23/36
vlc-0.8.6d-win32.exe	Win32:Trojan-gen {Other}	W32\Downldr2.ERXM	23/36
A0006003.dll	Win32:Trojan-gen {Other}	*Not-Zero: W32/Virtumonde.AC-.gen!Eldorado*	23/36
hgGwWOFU.dll	Win32:Trojan-gen {Other}	*Not-Zero: W32/Virtumonde.AC-.gen!Eldorado*	23/36
gta rock star games windows KeyGen All Version.zip	Win32:Downloader-BWA [Trj]	*Not-Zero: W32/Downloader-Sml-based!Maximus*	23/36
HDVideoCodec_ver1.605060506050-6050605060506050.0.exe	Win32:Trojan-gen {Other}	W32\Downldr2.EPRY	23/36
allok_rm2mp31.exe	Win32:Trojan-gen {Other}		24/36
avgas-setup 7.5.1.43.exe	Win32:Trojan-gen {Other}	W32\Downldr2.EOKQ	24/36
WinRAR v3.82 Corp.exe	Win32:Trojan-gen {Other}	W32\Downldr2.ERXM	24/36
HDVideoCodec_ver1.605060506050-.0.exe	Win32:Trojan-gen {Other}	W32\Downldr2.EMDU	24/36
HDVideoCodec_ver1.605060506050-6050605060506050605060506050-.0.exe	Win32:Trojan-gen {Other}	W32\Backdoor2.DAVU	24/36
A0006231.exe	Win32:Trojan-gen {Other}		24/36
mirc63.exe	Win32:Trojan-gen {Other}	W32\Downldr2.EOKQ	25/36
HDVideoCodec_ver1.605060506050-6050605060506050605060506050605060-50605060506050.0.exe	Win32:Trojan-gen {Other}	W32\Downldr2.EMVD	25/36
svhost.exe	Win32:Delf-LLM [Drp]	W32\Backdoor2.CVKP	25/36
Uninstaller[1].exe	Win32:Trojan-gen {Other}		26/36
AVG.Pro-8.0.138.complete.serial.rar-/is165807.exe	*Not-Zero: Win32:Trojan-gen {Other}*	W32\Backdoor2.DAUC	26/36
A0006226.exe	Win32:Trojan-gen {Other}		27/36
A0006233.exe	Win32:Adware-gen [Adw]		27/36
rqjvopyg.dll	Win32:Rootkit-gen [Rtk]		27/36
AS_AIO_KP_[RH].rar	Win32:Trojan-gen {Other}	*Not-Zero: W32/Downldr2.ZGN*	27/36
MicroAV.cpl	Win32:Neptunia-AGB [Trj]		28/36
A0007403.exe	Win32:Neptunia-AGB [Trj]		28/36
BS.Player PRO 2.28 Built 964 [FINAL VERSION].rar/Setup1.exe		W32\Downldr2.EDDI	28/36
avg_avwt_stf_all_8_164a135.exe-/Setup_ver1.1813.5.exe	Win32:Trojan-gen {Other}	W32\Downldr2.EOKQ	28/36
A0005985.exe	Win32:Trojan-gen {Other}		28/36
A0006229.exe	Win32:Trojan-gen {Other}		28/36
upd105320[1]	Win32:Rootkit-gen [Rtk]		28/36
eohgvrpa.dll	Win32:Trojan-gen {Other}		28/36

continued on next page

continued from previous page

File name	Avast! BART	F-PROT	VT
tdssmain.dll	Win32:Trojan-gen {Other}		29/36
Google Earth Pro 4.2 + Sky and Crack.rar/setup.exe		W32\Downldr2.ENTH	29/36
x	Win32:Spyware-gen [Trj]		29/36
tdssadw.dll	Win32:Rootkit-gen [Rtk]	W32\FakeAlert.3!Generic	30/36
A0007404.exe	Win32:Trojan-gen {Other}	W32\Backdoor2.CVOT	31/36
Setup_ver1.1482.02.exe	Win32:Trojan-gen {Other}	W32\Downldr2.ENTE	31/36
lwpwer.exe	Win32:Neptunia-AGB [Trj]	W32\Trojan2.DZLA	31/36
44Lty0BQ.exe	Win32:Trojan-gen {Other}	W32\Downldr2.ERTR	33/36
VLC_0.9.2_win32.rar		W32\Trojan2.EAZM	Too big to upload

Table C.1: Zero-day malware according to Avast! BART and/or F-PROT.

C.2 Results by F-Secure and Symantec

File name	F-Secure		Symantec	VT
Download_ad_Savannah_Camp-_TRIAL.exe				03/36
Download_AD_Savannah_Safari-_TRIAL.exe				03/36
Download_ss_Savannah_Camp-_TRIAL.exe				03/36
Autodesk.Revit.Architecture.2008.Full.Version.with.Keygen.zip	Hoax.Win32.Agent.s	11.10.2008		05/36
NodLogin9.4_64bits.rar				05/36
Easy Video Downloader 1.1 2008 fxg.rar	Backdoor.Win32.Reload.cg	07.07.2008		06/36
MP3_Cutter_Joiner_v2.20.zip				06/36
zopt.exe	AdWare.Win32.OneStep.x			07/36
Microsoft Streets & Trips 2008 Activated.exe	Suspicious:W32/Malware-!Gemini	Heuristic		07/36
WordPress Theme – Revolution Magazine.iso	Trojan-Dropper.Win32.VB.bix	18.07.2008		07/36
Firefox 3.exe	Suspicious:W32/Malware-!Gemini	Heuristic		08/36
Nero 8.2.8.0 Lite.Inc.Key.rar/keygen.exe				08/36
SoundBooth CS3.exe				08/36
MagicISO Maker v5.5.0261 - in-Acrysis.rar				08/36
Trillian Astra Alpha 4.0.0.83 [Released July 18 2008]LATEST.rar	Trojan-Downloader.Win32.Agent.aeog	01.09.2008		09/36
WinZip_12_Keygen.exe	Trojan-Downloader.Win32.Zlob.aauo	22.10.2008		09/36
All.Codecs.For.Windows-.Media.Player.rar	Trojan-Dropper.Win32.Agent.xnw	04.10.2008		09/36
keygen.exe	W32/Packed_Mew.C.dropper	?		09/36
HirensBootCD.9.5.rar-/Shredder.exe				09/36
_freescan[1].htm	Trojan-Downloader.JS.Agent.cru	29.09.2008		10/36

continued on next page

continued from previous page

File name	F-Secure		Symantec	VT
A0005385.dll	Trojan-Downloader.Win32.Zlob.abax	25.10.2008		10/36
slphseeccn.dll	Trojan-Downloader.Win32.Zlob.abaz	25.10.2008		10/36
Flash.Player.Pro.v3.7-.Cracked.rar	Trojan-Dropper.Win32.Agent.xnw	04.10.2008		10/36
The Cleaner S01E02 HDTV XviD-2HD.zip			Trojan.Wimad	10/36
Jookz.com_Bangin_Babes-_Screensaver.exe				10/36
Magic ISO Maker 5.4 Build 239.exe				10/36
Trillian Pro 3.1.10.0 [Skin + Plugin Packs].rar	Trojan-Downloader.Win32.Agent.aeog	01.09.2008		11/36
Eset NOD32 Antivirus + Lifetime Updates.rar	Trojan-Downloader.Win32.Zlob.aaok	21.10.2008		11/36
DivXInstaller_1.exe	Trojan-Downloader.Win32.Zlob.zmt	26.09.2008		11/36
DivXInstaller.exe	W32/Vundo.ESD	?		11/36
10[1].exe				11/36
13_3.ex_				11/36
HirensBootCD.9.5.rar-/WinKeyFinder.exe				11/36
YoutubeGet 4.7 [Youtube Video Downloader].rar	Trojan-Downloader.Win32.Agent.aeog	01.09.2008		12/36
DVDFab5085.exe	Trojan.Win32.Monder.vhf	23.10.2008		12/36
VW81DE2V.dll			Trojan Horse	12/36
46[1].exe				12/36
HirensBootCD.9.5.rar-/StartupMonitor.exe				12/36
HirensBootCD.9.5.rar-/uharcd.exe				12/36
Keymaker.exe	Trojan-Downloader.Win32.Zlob.zmt	26.09.2008		13/36
Keymaker.Nero.8.Ultra.Edition-.v8.3.6.0.rar				13/36
13_1.ex_				13/36
13.ex_				13/36
HirensBootCD.9.5.rar-/KillBox.exe				13/36
HirensBootCD.9.5.rar/VDefs.exe	Trojan-Downloader.Win32.Agent.ajvb	17.10.2008		14/36
HirensBootCD.9.5.rar-/DefragNT.exe	Trojan-Downloader.Win32.Zlob.aamg	18.10.2008		14/36
vlc sexy girl has shaking orgasm during sex.mpg	Trojan-Downloader.WMA.GetCodec.g	19.08.2008	Trojan.Wimad	14/36
Justin Timberlake-Recrimination-2CD-2008 [www TorrentMas com].zip	Trojan-Downloader.WMA.GetCodec.l	21.10.2008		14/36
WinRAR 3.7 Full Corporate Edition.rar	Trojan-Dropper.Win32.Agent.xnw	04.10.2008		14/36
PC Tank Assault _Attack 3.9.rar	W32/Vundo.ESD	?		14/36
Space game-Galaxy Invaders.rar	W32/Vundo.ESD	?		14/36
A0006004.dll			Downloader.Zlob-!gen.3	14/36
A0007323.dll			Downloader.Zlob-!gen.3	14/36
Nero 8.3 Ultra Keygen.rar	Backdoor.Win32.Rbot.vnc	22.10.2008		15/36
13_4.ex_	Trojan-Clicker.Win32.Agent.dod	29.09.2008		15/36

continued on next page

continued from previous page

File name	F-Secure		Symantec	VT
HirensBootCD.9.5.rar-/DriverBackup.exe	Trojan-Downloader.Win32.Zlob.aakn	16.10.2008		15/36
HirensBootCD.9.5.rar-/ShExView.exe	Trojan-Downloader.Win32.Zlob.aamn	19.10.2008		15/36
HirensBootCD.9.5.rar/xp-AntiSpy.exe	Trojan-Downloader.Win32.Zlob.aamp	19.10.2008		15/36
HirensBootCD.9.5.rar-/IBProcMan.exe	Trojan-Dropper.Win32.KGen.gic	19.10.2008		15/36
WinRAR Corporate Special Edition 3.7.1 [Zipped Final Edition].rar	Trojan.Win32.Monderb.sgr	05.10.2008		15/36
HirensBootCD.9.5.rar-/Splitter.exe	Trojan-Downloader.Win32.Agent.ajzr	18.10.2008		16/36
Magic ISO Maker 5_1.4 Build 239.exe	Trojan.Win32.Monderb.sht	06.10.2008		16/36
msxml71.dll			Packed.Generic.187	16/36
file[1].exe			Backdoor.Tidserv	16/36
WinRAR 3.71 Corporate.EXE	Backdoor.Win32.Agent.sly	30.09.2008		17/36
NeuView Media Player Professional v6.0.8.0253 [Patch].rar	Trojan-Downloader.Win32.Agent.aeog	01.09.2008		17/36
HirensBootCD.9.5.rar-/Autorun.exe	Trojan-Downloader.Win32.Agent.ajuj	16.10.2008		17/36
cntr[1]			Packed.Generic.180	17/36
CleanerInstaller_no.exe	FraudTool.Win32-.AntiSpywareSolutionPro.e	?		18/36
Adult Sexy BABY TOY Game-PC.rar	W32/Vundo.ESD	?		18/36
smchk.exe			Packed.Generic.180	18/36
urqPhgff.dll			Packed.Generic.180	18/36
Flash.Player.Pro.v3.7.WinALL-.Cracked-CzW.rar	Trojan-Downloader.Win32.Agent.agld	15.09.2008		19/36
HiHiSoft.Youtube.Download-.v4.8.5.Regged-F4CG.rar	Trojan-Downloader.Win32.Agent.agld	15.09.2008		19/36
byXOgfFV.dll			Packed.Generic.180	19/36
file[1]_1.exe			Packed.Generic.180	19/36
qoMccYqq.dll			Packed.Generic.180	19/36
De31.exe/is159287.exe	Suspicious:W32/Malware-!Gemini	Heuristic	Trojan.Vundo	20/36
YUR1.exe				20/36
mIRC 634.zip	Trojan.Win32.Buzus.aaqz	08.10.2008		21/36
only date ballers.KeyGen.All_Version.zip	W32/Downloader	28.09.2007	Downloader	21/36
Setup_ver1.1482.0.exe	Net-Worm.Win32.Kolab.asz	12.09.2008		22/36
efcAQKEt.dll	Vundo.gen257	?	Packed.Generic.180	22/36
dfmlxbpkwqd.dll			Downloader.Zlob-!gen.3	22/36
HDVideoCodec_ver1.6050.0.exe			Trojan.Zlob	22/36
tdssserv.sys	Backdoor.Win32.TDSS.zj	08.10.2008	Backdoor.Tidserv	23/36
vlc-0.8.6d-win32-.exe	Trojan-Downloader.Win32.Zlob.zoe	24.09.2008	Downloader.Zlob-!gen.3	23/36
vlc-0.8.6d-win32.exe	Trojan-Downloader.Win32.Zlob.zpe	25.09.2008	Trojan.Zlob	23/36
A0006003.dll	Vundo.gen257	?	Packed.Generic.180	23/36
hgGwWOFU.dll	Vundo.gen257	?	Packed.Generic.180	23/36
gta rock star games windows KeyGen All Version.zip	W32/Downloader	28.09.2007	Downloader	23/36
HDVideoCodec_ver1.60506050-60506050605060506050.0.exe			Trojan.Zlob	23/36
avgas-setup 7.5.1.43.exe	Trojan-Downloader.Win32.Zlob.zoe	24.09.2008	Downloader.Zlob-!gen.3	24/36

continued on next page

continued from previous page

File name	F-Secure		Symantec	VT
allok_rm2mp31.exe	Trojan-Downloader.Win32.Zlob.zoe	24.09.2008	Downloader.Zlob-!gen.3	24/36
WinRAR v3.82 Corp.exe	Trojan-Downloader.Win32.Zlob.zpe	25.09.2008	Trojan.Zlob	24/36
HDVideoCodec_ver1.60506050-6050605060506050605060506050-6050.0.exe			Trojan.Zlob	24/36
HDVideoCodec_ver1.60506050-6050.0.exe			Trojan.Zlob	24/36
A0006231.exe				24/36
mirc63.exe	Trojan-Downloader.Win32.Zlob.zoe	24.09.2008	Downloader.Zlob-!gen.3	25/36
HDVideoCodec_ver1.60506050-6050605060506050605060506050-506050605060506050605060.0.exe	Trojan-Downloader.Win32.Zlob.zux	30.09.2008	Trojan.Zlob	25/36
svhost.exe	Trojan.Win32.Buzus.aaup	09.10.2008		25/36
Uninstaller[1].exe	Trojan.Win32.Agent.afdz	30.09.2008	Trojan.Fakeavalert	26/36
AVG.Pro-8.0.138.complete.serial.rar-/is165807.exe	W32/Vundo.ESD	?	Trojan.Vundo	26/36
A0006226.exe	FraudTool.Win32.Agent.cq	?		27/36
A0006233.exe	Rogue:W32/XPAntivirus.GGY	08.10.2008	AntiVirus2009	27/36
rqjvopyg.dll	Trojan.Win32.Pakes.kva	07.10.2008	Packed.Generic.180	27/36
AS_AIO_KP_[RH].rar	W32/Packed_FSG.D	?	W32.DSS.Trojan	27/36
MicroAV.cpl	Rogue:W32/XPAntivirus.GGW	08.10.2008	AntiVirus2009	28/36
A0007403.exe	Rogue:W32/XPAntivirus.GGW	08.10.2008	Trojan.Fakeavalert	28/36
BS.Player PRO 2.28 Built 964 [FINAL VERSION].rar/Setup1.exe	Trojan-Downloader.Win32.Agent.aeog	01.09.2008	Downloader	28/36
avg_avwt_stf_all_8_164a135.exe-/Setup_ver1.1813.5.exe	Trojan-Downloader.Win32.Zlob.zoe	24.09.2008	Downloader.Zlob-!gen.3	28/36
A0005985.exe	Trojan.Win32.Agent.afee	30.09.2008	Trojan.Fakeavalert	28/36
A0006229.exe	Trojan.Win32.Agent.afkq	02.10.2008		28/36
upd105320[1]	Trojan.Win32.Monder.qzu	04.10.2008	Trojan.Vundo	28/36
eohgvrpa.dll	Trojan.Win32.Monder.rhn	05.10.2008	Trojan.Vundo	28/36
tdssmain.dll	Backdoor.Win32.Agent.tcb	08.10.2008	Packed.Generic.188	29/36
Google Earth Pro 4.2 + Sky and Crack.rar/setup.exe	Trojan-Downloader.Win32.Agent.agnb	15.09.2008		29/36
x	Trojan.Win32.Agent.acrp	07.09.2008		29/36
tdssadw.dll	Rootkit.Win32.Clbd.kr	01.10.2008	Trojan.Adclicker	30/36
A0007404.exe	Backdoor.Win32.Frauder.ik	30.09.2008	Trojan.Fakeavalert	31/36
Setup_ver1.1482.02.exe	Trojan-Downloader.Win32.Zlob.zme	22.09.2008	Trojan.Zlob	31/36
lwpwer.exe	Trojan.Win32.Agent.afgq	30.09.2008	Trojan.Fakeavalert	31/36
44Lty0BQ.exe	Trojan-Downloader.Win32.Agent.aidr	30.09.2008		33/36
VLC_0.9.2_win32.rar	?		?	Too big to upload

Table C.2: Detection of malware according to F-Secure and Symantec.

Appendix D

Logs From VirusTotal For The Wimad Example

The log files from VirusTotal for the experiment performed with the Wimad exploit in Chapter 7.5.1 are presented here. Figure D.1 shows the results for the original file, while Figure D.2 shows the results from the modified file.

File The_Cleaner_S01E02_HDTV_XviD-2HD. received on 11.12.2008 16:42:59 (CET)			
Antivirus	Version	Last Update	Result
AntiVir	7.9.0.31	2008.11.12	EXP/ASF.GetCodec.Gen
Avast	4.8.1248.0	2008.11.11	WMA:Wimad
AVG	8.0.0.199	2008.11.12	Downloader.Generic_c.AGE
BitDefender	7.2	2008.11.12	Trojan.Downloader.Wimad.B
eTrust-Vet	31.6.6203	2008.11.11	ASF/Wimad!generic
GData	19	2008.11.12	Trojan.Downloader.Wimad.B
Microsoft	1.4104	2008.11.12	TrojanDownloader:ASX/Wimad.AA
NOD32	3606	2008.11.12	a variant of WMA/TrojanDownloader.GetCodec.gen
Rising	21.03.22.00	2008.11.12	Trojan.DL.Win32.GetCodec.b
SecureWeb-Gateway	6.7.6	2008.11.12	Exploit.ASF.GetCodec.Gen
Sophos	4.35.0	2008.11.12	Troj/Wimad-F
Symantec	10	2008.11.12	Trojan.Wimad
Additional information			
File size: 3545425 bytes			
MD5...: 483bc61b01ee33fccce1b31ea65a29d1			
TrID..: File type identification Windows Media Video (100.0%)			

Figure D.1: Results from scanning a file using the Wimad exploit, 12 out of 36 anti-virus engines detected malware in the file.

File The_best_zero-day_video_ever.avi received on 11.12.2008 16:42:25 (CET)			
Antivirus	Version	Last Update	Result
AntiVir	7.9.0.31	2008.11.12	EXP/ASF.GetCodec.Gen
AVG	8.0.0.199	2008.11.12	Downloader.Generic_c.AGE
eTrust-Vet	31.6.6203	2008.11.11	ASF/Wimad!generic
NOD32	3606	2008.11.12	a variant of WMA/TrojanDownloader.GetCodec.gen
Rising	21.03.22.00	2008.11.12	Trojan.DL.Win32.GetCodec.b
SecureWeb-Gateway	6.7.6	2008.11.12	Exploit.ASF.GetCodec.Gen
Additional information			
File size: 3545425 bytes			
MD5...: fb820d765a44962d25c511624f02e4b8			
TrID..: File type identification Windows Media Video (100.0%)			

Figure D.2: Results from scanning a modified file using the Wimad exploit, only 6 out of 36 anti-virus engines detected malware in the file.

File The_best_zero-day_video_ever.avi received on 11.14.2008 13:15:55 (CET)			
Antivirus	Version	Last Update	Result
AntiVir	7.9.0.31	2008.11.14	EXP/ASF.GetCodec.Gen
AVG	8.0.0.199	2008.11.14	Downloader.Generic_c.AGE
eTrust-Vet	31.6.6208	2008.11.13	ASF/Wimad!generic
Microsoft	1.4104	2008.11.14	TrojanDownloader:ASX/Wimad.AB
NOD32	3613	2008.11.14	a variant of WMA/TrojanDownloader.GetCodec.gen
Rising	21.03.42.00	2008.11.14	Trojan.DL.Win32.GetCodec.b
SecureWeb-Gateway	6.7.6	2008.11.14	Exploit.ASF.GetCodec.Gen
Additional information			
File size: 3545425 bytes			
MD5...: fb820d765a44962d25c511624f02e4b8			
TrID..: File type identification Windows Media Video (100.0%)			

Figure D.3: Results from scanning a modified file using the Wimad exploit two days later, now 7 out of 36 anti-virus engines detected malware in the file.

Appendix E

Attached CD

Enclosed with this project report is a CD. It contains all the scan logs acquired during the experiment, along with Microsoft Excel spreadsheets showing all the zero-day malware in a more well arranged structure. The content is as follows:

File: ***list_of_zeroday_malware.xls*** is perhaps the most interesting file as it contains the 124 zero-day malware together with detection names by Avast! BART, F-PROT, F-Secure and Symantec. The share of anti-virus engines from VirusTotal is also included and makes this a complete overview of our main results.

File: ***zeroday_giant_spreadsheet.xls*** lists the path of all zero-day infections and on which machine(s) they were detected. This spreadsheet was used to merge the results from our six computers into a joint list.

File: ***internet_explorer_history.xls*** shows the history of Internet Explorer and includes nearly 300 URLs.

Directory: ***Scan logs*** contains all log files acquired from the six different computers. The logs from the control, baseline and final scan are written in the format *yyyymmdd_antivirusprogram_computername*. As an example *20081103_fprot_andrew* is the log from the final scan, performed November 3th with F-PROT on the machine named Andrew.

Directory: ***VirusTotal logs*** contains all the results from uploading the files to VirusTotal. Each scan result is located in a .htm file. As an example *allok_rm2mp31.exe.htm* is the result for the file *allok_rm2mp31.exe*.

Where Only Fools Dare to Tread: An Empirical Study on the Prevalence of Zero-day Malware

Håvard Vegge, Finn Michael Halvorsen and Rune Walsø Nergård
Norwegian Univeristy of Science and Technology (NTNU)
NO-7491 Trondheim, Norway
{havardv, finnmich, runewals}@stud.ntnu.no

Martin Gilje Jaatun* and Jostein Jensen
SINTEF ICT
NO-7465 Trondheim, Norway
{Martin.G.Jaatun, Jostein.Jensen}@sintef.no

Abstract

Zero-day malware is malware that is based on zero-day exploits and/or malware that is otherwise so new that it is not detected by any antivirus or anti-malware scanners. This paper presents an empirical study that exposed updated Micsosoft Windows XP PCs with updated antivirus software to a number of unsavoury internet software repositories. A total of 124 zero-day malware instances were detected in our experiment. Our conclusion is that if a user is sufficiently adventurous (or foolish), no antivirus protection can prevent a zeroday malware infection.

1 Introduction

IT administrators are constantly fighting to keep their systems patched and updated, while malware authors keep churning out more and more malware every day. The time from when a vulnerability is detected by "the good guys" until exploit code is available keeps shrinking, but the deluge of new malware that do *not* rely on new exploits or other fancy mechanisms ensure that there is also a growing lag between the discovery of a new malware specimen and the time of generally updated virus signatures for this malware.

Zero-day is a broad term and can be applied to various areas of information security. Often people associate the zero-day with software vulnerabilities which are not known to the public, and the creation of zero-day exploits. This paper is focused on zero-day malware, that is, malicious software which is not detected by anti-virus programs due to lack of existing virus signatures or other malware detection techniques. Zero-day malware can also – but does not necessarily have to – be based on zero-day exploits.

Although the concept of zero-day exploits (and malware) has been around for years, no major studies or scientific articles seem to have been published on this topic. Most of the related literature available consists of loose web articles with limited details.

1.1 Background

There is no doubt that files from file-sharing networks represent a great risk. According to a study of malware prevalence in Kazaa by Shin et al. [1], 15% of 500,000 downloaded files were infected by malware. Kalafut et al. [2] found that in over a month of data, 68% of all downloadable responses in LimeWire/Gnutella contained malware. In a study by Andrew Berns [3], 70 out of 379 downloads from the BitTorrent network had malware (18.5%).

Many companies or web sites test different anti-virus software on a regular basis. Two of the biggest actors in this area are AV-Comparatives.org and AV-Test.org. As such companies are comparing anti-virus vendors, their methodology is not the same as in this paper where we are searching for zero-day malware. Still, a proactive/retrospective test performed by AV-Comparatives [4], can give indications of what results to expect. A retrospective test is used to test the proactive detection capabilites of scanners. It gives an idea how much new malware a scanner can detect (for example by heuristic/generic detection), before a signature is provided for the malware.

According to SANS Institute [5], all operating systems and all software applications are vulnerable to zero-day vulnerability discovery and exploitation. This paper is limited

*Corresponding Author

to detection of zero-day malware threatening Windows XP and Internet Explorer.

1.2 Paper outline

The rest of this paper is structured as follows: In Section 2 we describe the method and preparations for our experiment, and in Section 3 we describe how the experiment was carried out. We present our results in Section 4 and discuss them in Section 5. Section 6 concludes the paper.

2 Method

An outline of the method can be seen in Figure 1. The preparations before the exposure phase included setting up a lab environment directly connected to the internet, and installing operating systems and anti-malware packages on the laboratory computers.

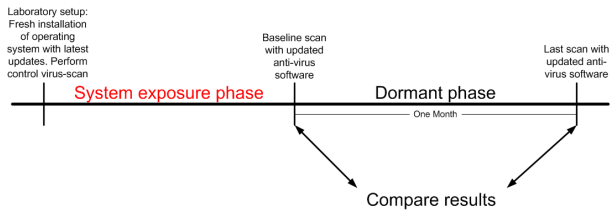


Figure 1. Method to find zero-day malware

The computers were installed with Windows XP, and the latest service pack (SP3) was applied from CD. The computers were not connected to the Internet until proper anti-virus software was installed, and then the first thing we did was to install the latest updates from Windows Update.

Each machine in the laboratory was set up with different anti-virus software (see Table 1), which all automatically updated themselves with the latest virus definitions when they detected an Internet connection. The main purpose of the desktop anti-virus programs was to avoid having our computers infected with already known malware. All the software was installed with default settings, but some changes were made to make the different anti-virus software as similar as possible. Norton Internet Security 2008 comes with a built-in firewall, but this was disabled. All the anti-virus software was set to quarantine infected files if possible.

We installed Spybot Search & Destroy[1] on all machines to protect against spy- and adware. While we did not originally intend to do this, it became apparent that it was necessary to avoid the machines becoming so cramped up with spy- and adware that they would be practically unusable for the intended activity.

[1] http://www.safer-networking.org

Computer name	Anti-virus
Gustav	Norman
Ivan	Norton
Katrina	F-Secure
Mitch	Avast!
Andrew	AVG

Table 1. Computer overview.

3 Procedure

As the time schedule below shows, we actively exposed the computers to suspicious web sites and file-sharing networks during a period of two weeks. The computers were then shut down for about a month, before they were turned on in the beginning of November 2008 to perform anti-virus scans and analyses.

September 10th:	All computers in the laboratory were connected to the Internet
September 15th:	Offline virus scans and experiment start-up.
October 1th:	Offline virus scans before computers were shut down for a month.
November 3th:	Offline virus scans.

3.1 System Exposure

Monday September 15th 2008 we actively started to expose the computers to web sites, file-sharing systems, etc. We had prepared an initial list of suspicious web sites containing warez, screensavers, codecs, mp3s and other free downloads. The actual number of visited web sites ended up being much larger, as we clicked on many advertisements and visited partner sites. Since web sites from Romania, Hong Kong and Russia were considered most risky by the computer security company McAfee, Inc. [6, 7], we tried to include some sites from those countries as well.

As seen in Table 2 we had also come up with a list of search keywords, which we applied when using file-sharing programs. The list was compiled from the names of the 50 most popular Windows downloads at Download.com [8] on September 15th.

As the timeline in Figure 2 indicates, the system was exposed to web sites and file-sharing networks over a two week period. Some days were spent on one source only, while other days consisted in the use of several sources. *Install* indicates that the same downloaded files were installed on all the computers. The time slots containing *X* means nothing was actively done to expose the computers, but they were still connected to the Internet and both Internet Explorer, μTorrent and LimeWire were running.

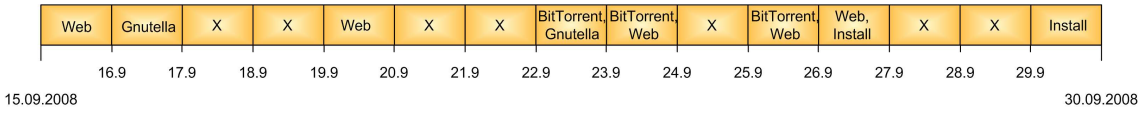


Figure 2. System exposure timeline.

The same actions were performed on the five computers in the laboratory almost simultaneously, in order to facilitate as fair as possible comparisons. The following sections describe the procedure and actions that were taken in Limewire, μTorrent, Internet Explorer and the anti-virus software, respectively.

3.1.1 File-sharing networks

File-sharing networks are known to be a significant source of malware [1], and therefore it was important to expose the experiment to these networks. We used the keywords in Table 2 to search for candidate files.

For the Gnutella network we chose the client Limewire[2], which is the most popular client for this network [9]. Limewire was installed with default settings, but sharing of files was disabled to avoid any legal issues and excessive network traffic. The most poular client for BitTorrent is uTorrent[3] and is well suited for our laboratory as it is light-weight and easy to use.

avg, antivirus, adaware, limewire, frostwire, winrar, winzip, mirc, irc, player, real, media player, zip, free edition, youtube, downloader, irfanview, google, chrome, adobe, firefox, virtualdj, vlc, iso, cleaner, msn, live, nero, divx, spyware, torrent, activex, flash, trillian, norton, mp3, 2008

Table 2. Keywords used for data collection.

3.1.2 Surfing the Web

More and more people get access to the Internet, but it can not be regarded safe, even though you try to avoid obviously suspicious web sites. According to a technical report by Google [10] approximately 1.3% of the incoming search queries to Google's search engine return URLs labeled as malicious. In order to expose the laboratory computers to a wide range of threats, it was then natural to visit some potentially risky web sites.

A great deal of all web sites contain ad-ware, viruses and other threats. McAfee has published two reports that show which domains that are most risky [6, 7]. Based on these reports and the use of search engines like Google and Yahoo! together with popular search phrases, we came up with a list of possibly malicious sites. It is a mixture of popular ordinary web sites and sites which claim to serve downloadable items, such as warez, screensavers and mp3s.

When visiting web sites, the integrated browser Internet Explorer 7 was a natural choice. If plugins like Flash etc. were missing, they were installed on demand.

The following strategy was followed:

- We basically started at the top of our list and visited the web sites one by one.
- Since our goal was to be exposed to as much malware as possible, we acted like a foolish person, uncritically clicking *OK* to everything that popped up.
- If the particular web site had partner sites or other tempting links, we paid them a visit too.
- When visiting warez sites, where it was possible to download for instance software, we typically chose a few of the most popular items and saved them to a directory on the computer for later analysis.

The same procedure was performed on all our computers almost simultaneously. Still we experienced that different pop-up windows showed up on different computers, so minor dissimilarities occurred.

As with the installed files from Gnutella and BitTorrent, a few of the files from the web also turned out to contain malware, although they had been scanned by anti-virus software previously.

3.2 Offline Search

In addition to the installed anti-virus packages, we obtained two offline anti-virus programs, F-PROT and avast! BART CD. By offline we mean that the host operating system is not booted, the anti-virus software is run from a live CD.

As noted in the time schedule at the beginning of Section 3, offline scans were performed three times during this project.

1. A control virus-scan was performed before starting the exposure phase, to verify that the laboratory computers were clean.

[2] http://www.limewire.com
[3] http://utorrent.com

2. A baseline scan was performed after the two weeks of exposure.

3. The last scan was performed one month after the baseline scan, in order to compare the results.

4 Results

Our experiment resulted in 124 zero-day malware candidates, i.e. malware that was not found by F-prot or avast! BART immediately after the exposure period, but was detected after a new scan with updated signatures after the 1-month dormant period.

4.1 Zero-day Malware Results

According to the offline anti-virus programs avast! BART and/or F-PROT, the 124 files are all zero-day malware, and none of the files were reported malicious at the baseline scan. In addition, all files detected by avast! BART and F-PROT have been uploaded and scanned at VirusTotal [4] where 36 different updated anti-virus engines are present.

As an example, the file *Easy Video Downloader 1.1 2008 fxg.rar* is detected by avast! BART as *Win32:Trojan-gen {Other}*, while F-PROT did not detect it at all. However, F-PROT is not the only anti-virus engine that lacks a signature for this file. Only 5 out of 36 engines at VirusTotal detected this file to be malware. That means 31 engines, F-PROT included, were lacking a signature at the time of the last virus-scan.

From the VirusTotal results, we were able to check whether other anti-virus engines detected the same files as avast! BART and F-PROT. F-Secure and Symantec were chosen because they are big vendors of anti-virus solutions and they also had good descriptions of the different malware types, as apposed to F-PROT who did not offer any information about the malware types on their web site. It was also impossible to obtain information about when avast! had incorporated specific signatures to their database.

A lot of the malware is also considered zero-day according to both F-Secure and Symantec. If F-Secure reports to have added the signature at some date in October, it means they did not detect the malware at the time of the baseline scan (which was done on October 1st). Some of the files were not even detected at all, and we conclude that these are zero-day malware with reference to the specific anti-virus software. All files in these lists were gathered during September, which means they have been around for over one month. It is disquieting that a large number of anti-virus engines do not detect these files to be malicious, even though they have been in the wild for such a long time.

[4] http://www.virustotal.com

4.2 Zero-day Malware Sources

As we can see from Figure 3, which is based on the 124 zero-day malware infected files, most of the zero-day malware comes from the use of BitTorrent. While zero-day malware from the use of BitTorrent is estimated to 47% zero-day malware from the Gnutella network on the other hand constitute only 7%. The reason for this big difference can be explained by the difference in their search mechanisms, as discused further in Section 5.

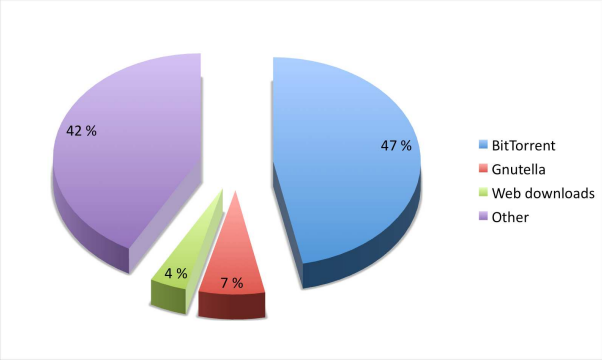


Figure 3. Zero-day malware sources based on the 124 zero-day malware infected files.

The part labeled *Other* in Figure 3 constitutes 42% of the diagram; this includes files that have not been installed in the limewire, torrent or web download folder but rather on the desktop, in the Temp folder, the Temporary Internet Files folder, the Program files folder or different system files folders, to mention some locations. These are files whose source cannot be determined for certain. Some of them may come from surfing the web, by clicking on different pop-ups and ads, and some may be created when we installed some of the downloaded files. The part in the figure labeled *Web downloads* indicates that 4% of the zero-day malware comes from files downloaded during surfing the web.

Source	Downloaded files	Zero-day malware	%
BitTorrent	∼ 400	58	14,5 %
Gnutella	∼ 6000	9	0,15 %
Web downloads	∼ 80	5	6,25 %
Unknown source	?	52	?

Table 3. Approximate percentage of zero-day malware from different sources, based on all downloaded files.

Table 3 is an attempt to estimate what percentage of the downloaded files in BitTorrent, Gnutella and from the web,

respectively, that contained zero-day malware. It is important to notice the difference between the percentages in Table 3, which is based on all downloaded files, and the percentages in Figure 3, which is only based on the 124 zero-day malware infected files. Due to our procedure, where online anti-virus software removed malware as they were detected, we do not have the exact number of downloadeded files, and the percentages should only be used as a rough estimate. Trough BitTorrent the number of download files were actually about 400. We had 40 keywords which on average resulted in 10 downloads apiece. The Gnutella number is more difficult to estimate, since files from the different computers were gathered to a common pool. The actual number of download files in LimeWire/Gnutella are probably much higher, maybe twice as much as indicated in the table, but the conclusion is unchanged. The percentage of zero-day malware was significantly higher in the BitTorrent network than in the Gnutella network. Again, this can be partly explained by the difference in the search mechanisms.

It is worth mentioning that the files downloaded from the web are typically from the suspicious sites identified partly by the Google and Yahoo search engines. It will be very wrong to think that 9% of files from the world wide web contain zero-day malware, but what the table indicates is that zero-day malware exist in all of these areas. Also note that a significant number of files with malware was found elsewhere on the computers; we can only assume that these files were downloaded by spyware contracted during the experiment.

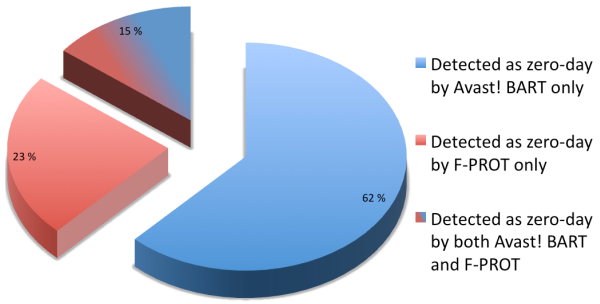


Figure 4. Percentage share of zero-day malware detected by avast! BART, F-PROT or both.

5 Discussion

The term zero-day malware is widely used and the existence of them is well known. However, very few can refer to some actual numbers of their prevalence. 124 unique files were identified to be infected with zero-day malware. The procedure focused on exposing the computers in the laboratory to a broad range of suspicious material and generally acting as an ignorant Internet user. Installing programs, visiting ads and clicking OK to everything that popped up was part of the exercise. Although a normal user would probably not manage to expose his or her computer to the same amount of suspicious material in the short timeframe used in this experiment, a normal user has a much longer exposure period (i.e. countinuous and never ending). This illustrates that the risk of getting infected by malware that is not detected by anti-virus protection is alarmingly high.

New malware that the anti-virus engines do not have a signature for is likely to escape detection by a desktop anti-virus solution. Proper behavior on the Internet can only protect users to a certain extent. If they visit the wrong web site or download a file with a zero-day malware, however, they will probably *not* be protected from infection.

In a threat summary for the second half of 2008 [11], F-Secure reports that one million detection signatures were added during the year - a number that has never before been so high. The acceleration of new malware instances can likely be explained due to obfuscation techniques such as polymorphism and metamorphism. Such techniques have successfully been demonstrated to aid malware in evading detection by commercial virus scanners [12].

A deep analysis of the infected files is time-consuming and considered out of the scope for our experiment. Thus, we did not attempt to determine whether any of the 124 zero-day samples were just different obfuscated instances of the same origin. An indication is however given by looking at the malware descriptions on F-Secure's web site. It seems like many of the infections seem to be just new types of malware or new instances of already known types. Still they fall under our definition of zero-day malware.

Concerning prevalence of zero-day malware in the different infection sources we based our experiment on, BitTorrent generally seems to contain less malware than Gnutella, although the amount of zero-day malware were in fact higher. This may be related to the different search mechanisms in the two P2P networks. Only BitTorrent provides the ability to search for newly uploaded files. As such, downloading newly added files are easier in BitTorrent, and the aspect of new malware is one of the most important matters of zero-day malware. If malware creators manage to distribute a new malware instance that the anti-virus vendors do not currently detect, the possibility of successfully infecting large number of hosts is a lot higher.

In this experiment we have demonstrated that our method is adequate to perform a retrospective measurement of the prevalence of zero-day malware. Although our approach was primarily based on using two offline anti-virus scanners to perform the baseline scan and the final scan, the use of VirusTotal illustrated the benefits of including even

more anti-virus tools in the process to get more accurate results.

Our procedure for exposing the laboratory machines to potentially malicious content was focused on making the experiment as close to a real world scenario as possible, and that implies combining web surfing with file downloading and file sharing activities. In our results the prevalence of zero-day malware for the different infection sources are indicated. However, because of our procedure we cannot state the origin of each detected zero-day malware instance with exact certainty. A stricter and more firmly defined procedure would have to be defined and followed if the goal were to get more accurate measures.

6 Conclusion

We have presented an empirical study where we have exposed updated Microsoft Window XP PCs with different up-to-date antivirus packages to numerous locations we suspected of containing zero-day malware. After a two-week exposure period, our computers had contracted a minimum of 124 malware specimens that were not detected by our anti-virus packages during (or at the end of) the period.

The prevalence of zero-day malware implies that anti-virus software, which primarily relies on signatures, does not provide sufficient protection. Coupled with the exponential growth of new malware variants, our findings indicate that the anti-virus vendors already have major problems with keeping the signature lag within acceptable limits.

7 Further work

Due to time constraints, we have not gained any further knowledge on the prevalence of zeroday *exploits*. We suspect that this would require a more extensive lab setup, and a longer dormant phase.

It is possible that more complete results could be obtained by automating the exposure process, e.g. by using web crawler technology.

Acknowledgments

This paper is based on the results of a minor thesis at the Norwegian University of Science and Technology (NTNU).

References

[1] S. Shin, J. Jung, and H. Balakrishnan, "Malware prevalence in the kazaa file-sharing network," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 333–338.

[2] A. Kalafut, A. Acharya, and M. Gupta, "A study of malware in peer-to-peer networks," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 327–332.

[3] A. Berns, "Searching for malware in bittorrent," *Computer Security Presentation at The University of Iowa*, April 29, 2008, http://www.cs.uiowa.edu/~ejjung/courses/169/project/publish/AndrewBerns_presentation.pdf.

[4] A. Clementi. (May 31, 2008) Anti-Virus Comparative No.18: Proactive/retrospective test. AV-Comparatives. [Online]. Available: http://www.av-comparatives.org/seiten/ergebnisse/report18.pdf

[5] SANS Top-20 2007 Security Risks. SANS Institute. [Online]. Available: http://www.sans.org/top20/

[6] D. Nunes and S. Keats, "*Mapping the Mal Web*," *McAfee SiteAdvisor*, March 12, 2007, http://www.siteadvisor.com/studies/map_malweb_mar2007.html.

[7] S. Keats, "*Mapping the Mal Web Revisited*," *McAfee SiteAdvisor*, June 4, 2008, http://www.siteadvisor.com/studies/map_malweb_jun2008.pdf.

[8] Most popular Windows downloads. Download.com. [Online]. Available: http://www.download.com/3101-2001_4-0.html?tag=mncol;sort

[9] E. Bangeman, "Study: Bittorrent sees big growth, limewire still #1 p2p app," *ars technica*, April 21 2008. [Online]. Available: http://preview.tinyurl.com/3recfp

[10] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, "All Your iFRAMEs Point to Us," in *Proceedings of the 17th USENIX Security Symposium*, 2008. [Online]. Available: http://www.usenix.org/events/sec08/tech/full_papers/provos/provos.pdf

[11] F-Secure IT Security Threat Summary for the Second Half of 2008. F-Secure. [Online]. Available: http://www.f-secure.com/2008/2/index.html

[12] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pp. 421–430, Dec. 2007.