



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Enterprise Penetration Testing (Security 560)"
at <http://www.giac.org/registration/gpen>

Bypassing Malware Defenses

GIAC (GPEN) Gold Certification

Author: Morton Christiansen, moc@nsense.dk

Advisor: Dr. Carlos Cid

Accepted: May 7th, 2010

Abstract:

While most other areas of penetration testing are well understood, and their methodologies well documented, little information regarding testing and bypassing malware defenses is available in the public domain. Still, malware incidents remain the most expensive type of incidents caused by outsiders, while also being the most frequent type of incidents occurring to organizations. In addition, since malware payloads are normally executed on internal networks, bypassing most firewall restrictions, they do tend to be the weapon of choice for targeted attacks. Malware is used to perform multiple offensive activities: launching distributed denial of service attacks (DDoS), collecting classified information, etc. Consequently, testing and understanding the efficiency and configurations of malware defense systems is of uttermost importance.

Firstly, the paper describes how host-based anti-virus signatures can be bypassed. The paper illustrates that known malware may sometimes bypass host-based anti-virus systems (AVs), even when the piece of malware has not been modified to do so. Secondly, we show how changing the signatures of known pieces of malware may help the adversary bypass most signature-based AVs and some behavior-based AVs. Techniques such as hex editing, repacking and reverse engineering are demonstrated to this end. Thirdly, we demonstrate how a new piece of malware, which bypasses all of the 42 AV products included in the test set and is able to use the target organization's internal

Domain Name System (DNS) server as a covert channel for command and control (C&C), can be created. Using recursive DNS as a communication channel allows the malware author to control the malware over the Internet even if the infected hosts have no direct Internet connectivity at all. This is an improvement over current malware that does require a direct connection for its C&C interface to function.

Finally, having clearly demonstrated that it is necessary to detect malware before it reaches the hosts, the paper then describes and compares various channels whereby malware may be introduced to the target. The focus here is on bypassing HTTP/SSL and SMTP malware defenses, since they are the communication channels utilized by most organizations. Surprising no previous research appears to have been done into this area. All results in this part of the paper are based on professional penetration tests conducted for some of the largest organizations in Northern Europe.

Outline

1.	Introduction	4
2.	Bypassing Host-based AV	6
2.1.	Methodology	6
2.2.	Known Signatures	7
2.3.	Modifying Signatures	10
2.4.	New Malware for Targeted Attacks	13
3.	Bypassing SMTP & HTTP/SSL Malware Defenses	17
3.1.	Methodology	17
3.2.	Dangerous File Formats	18
3.3.	Renamed Malware	21
3.4.	Compression	23
3.5.	Embedded Malware	26
3.6.	Encryption	28
3.7.	Exploits	29
3.8.	Steganography	30
3.9.	Out of Band Attacks	31
3.10.	Encoding & Encapsulation	32
4.	Conclusion	34
5.	References	35

1. Introduction

Western societies increasingly rely upon information as the foundation for their social, political, financial and military success. Much of this information is transmitted through the Internet, or is handled in intranets using the Internet protocols. Often these internal networks even engage in some sort of (in)direct communication with the Internet itself.

Examples of such mostly internal systems include Supervisory Control and Data Acquisition (SCADA) at times controlling nuclear reactors, civil defense sirens and air traffic control or the electricity/water/oil supply for entire nations. Other examples of sensitive internal systems include databases of large banks, of the police and of the military containing financial or intelligence information.

Malware is interesting in this context, because it is often executed directly on the internal networks, at times giving the adversary complete control over internal systems at the targeted organization using the malware's command and control (C&C) interface. In addition, malware incidents are the most expensive and the most frequent type of incidents occurring to organizations (Richardson, 2008).

Surprising it is then, that very little research has been conducted concerning bypassing the malware defenses that protect such systems and organizations at large. The research that has been done mainly focus on how malware may contain functionality that allow it to change itself in order to avoid detection, i.e. polymorphism (MacManus, Mason, Monroe, & Small, 2009; Song et al., 2007). However, only very little research has been done concerning how to change existing malware to bypass detection (Aharoni, 2009; Christodorescu, Jha, 2004). The difference between the two areas is notable. Polymorphism focus on a mechanism *already* present in the malware. Here the AV vendors have the advantage: once a specimen has been collected they can analyze it to try and find a signature or behavior always present in the malware. On the other hand, trying to change existing malware to bypass AV products instead gives the advantage to the adversary: here he can check known AV products and the signatures or behaviors they are reacting to, then change his malware to try and bypass these defenses. Such malware may then be used for targeted attacks. Finally, to the author's knowledge no previous

research appears to have been done into bypassing SMTP and HTTP/SSL gateways. The primary objective in this paper is to create awareness about the vulnerabilities or weaknesses often present in such defenses.

Generally seen, there are two techniques for host-based AV products to detect malware: Signature-based and behavioral based.

Signature-based techniques are the most common (Aharoni, 2009). These work by the AV having a large database of signatures. A signature in this regard is a byte- or binary pattern, or a hash hereof, known to exist in a specific piece of malware (Christodorescu, Jha, 2004). Since these signatures are mostly unique, false-positives are rare. The disadvantage with this approach is its ineffectiveness in detecting new variants of an old piece of malware. I.e. malware authors often release multiple strains of malware, each with only slight modifications in order to try to bypass AV signatures. AV vendors sometimes try to combat this by utilizing generic signatures. Generic signatures try to identify whole families of malware, e.g. by designing a signature for a code segment shared by multiple pieces of malware.

The more generalized the detection mechanism become, the more likely false-positives are to occur. For this reason *behavior-based* detection techniques are more rarely utilized (Aharoni, 2009). These techniques often work by either inspecting the code before running it, or by running the code in a virtual sandbox environment, looking for suspicious behavior. Examples of such suspicious behavior include key logging, injection into privileged processes, or alteration of critical operating system (OS) or AV files. The advantage of this approach is that, if successful, it can detect completely new malware. That is, these techniques do not need a database of byte- or binary patterns existing in known malware; by definition they simply assess the program behavior instead.

In section 2 the paper describes bypassing host-based AV (AntiVirus) engines. First, we will look at AV efficiency rates for detecting what should be well-known malware in the wild. Then, we will look at how to change signatures of malware, so that we can evade the host-based AV defenses. The paper describes these various techniques for changing malware signatures, and estimates the likelihood that these changes will be

able to evade host-based AV defenses. Finally, the most effective method for bypassing host-based AVs is presented: writing your own malware. It is shown how this simple PoC (Proof of Concept) code evades all 41 AV products tested against. In addition, the PoC will be using a covert channel over recursive DNS for controlling the backdoor in order to evade the vast majority of firewalls in existence. This is an improvement over current malware which do require a direct connection for its C&C interface to function.

Clearly demonstrating that we need to detect malware *before* it reaches the host, section 3 of the paper then describes various channels whereby malware may be introduced at the target. The focus here is on bypassing HTTP/SSL and SMTP malware defenses, since they are the communication channels utilized by most organizations. To the author's knowledge no previous research appears to have been conducted within the area of testing and bypassing such gateways. This section is based on professional penetration tests conducted by the author for some of the largest organizations in Northern Europe.

2. Bypassing Host-based AV

This section illustrates why allowing employees to download executables from unknown sources on for example the Internet can be dangerous. This is illustrated by altering existing malware, or creating new malware, in a way they may bypass host-based AV defenses.

2.1. Methodology

All malware samples mentioned in this section have been uploaded to VirusTotal in order to determine how often they are detected as malware by common AV products. VirusTotal is a service offered free of charge from Hispasec Sistemas (2010). The site scans the uploaded file using a total of 41 different AV engines. All engines are subject to real-time automatic updates of malware signatures, thus indicating the degree to which current AV engines are able to detect the malware samples mentioned in this section. VirusTotal is used by many reputable malware researchers (Zeltser, 2009).

It should be noted that VirusTotal does not actually execute the uploaded file for more comprehensive analyses. Thus, there remains a chance that the AV engines may still detect the malware if they perform behavior or Signature-based analysis during runtime. This potential difference in detection rates for *stored* and *executed* malware respectively results from malicious behavior or signatures that can not always be inferred from stored files alone (Aharoni, 2009; Zeltser, 2009). Malware with a known malicious signature that has had its packer changed is just one such example. When the malware is unpacked during runtime the known malicious signature may, depending on the nature of the packer, re-emerge. This difference in detection rates for stored and executed malware should not be confused with the terms “On Demand” and “On Access,” which simply denote whether the AV scan is initiated by the user (On Demand), or run automatically whenever the file is read, executed or copied (On Access). In addition, VirusTotal does not utilize cloud AV mechanisms. Instead of the AV scan taking place on individual clients, cloud AV mechanisms utilize a network cloud to scan the submitted samples. The scan in the cloud typically involves scanning by multiple AV engines. Consequently, the statistics in this section based on VirusTotal concern the probability of the malware sample being able to be stored on the hard disk of AV protected systems, while there still remains a chance of the malware being detected during runtime.

To overcome these limitations, external studies are referenced and the illustrated pieces of malware are executed on hosts with AV products installed. The AV products tested against are the latest versions available from the AV vendors, including the latest updates to these. In addition, findings are based on penetration tests for some of the largest organizations in Northern Europe, including execution of custom-made malware within the internal networks of these organizations in a controlled manner.

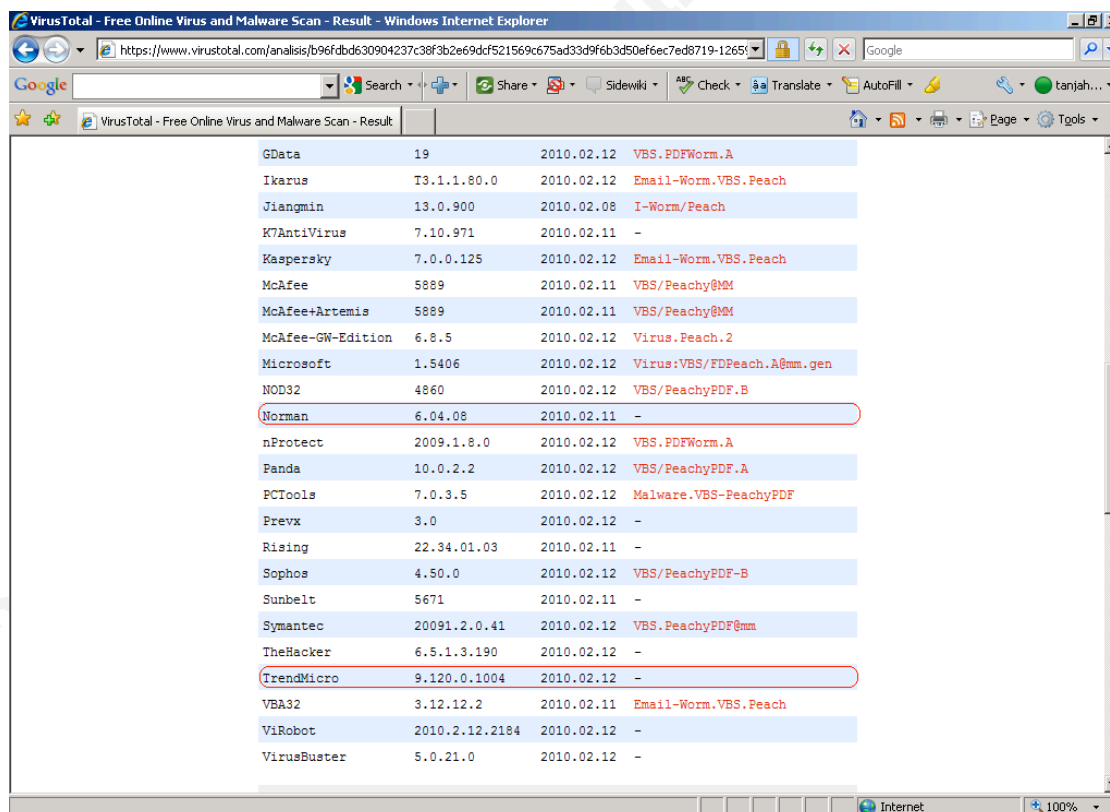
2.2. Known Signatures

From the defense side, most AV solutions are Signature-based (Aharoni, 2009). Simplified, these systems search executables and other documents for strings of characters, known to occur in specific pieces of malware. If a file contains the exact same

string as one of the strings in the AV's database, the file will be detected as malicious; otherwise it will not.

From the offense side, studies have shown that approximately 22.000 new strains of malware appear every day (Panda Security, 2008). For a Signature-based AV to accurately detect all these strains it would require knowledge of every single strain released. In practice, this appears to be an impossible task – surely some malware is bound to be missed. This problem will be illustrated by first testing whether randomly selected pieces of malware will be detected by various AV engines, then by referencing empirical studies concerning detection rates afterwards.

To illustrate this problem in practice, two randomly selected pieces of malware have been used for this paper: a PDF based e-mail worm (Peachy) found in the wild, and a Trojan called Turkojan made publicly available from the hacker group's web site. Figures 1 and 2 illustrate the results we obtain, when submitting these two files to VirusTotal.



Engine	Version	Date	Result
GData	19	2010.02.12	VBS.PDFWorm.A
Ikarus	T3.1.1.80.0	2010.02.12	Email-Worm.VBS.Peach
Jiangmin	13.0.900	2010.02.08	I-Worm/Peach
K7AntiVirus	7.10.971	2010.02.11	-
Kaspersky	7.0.0.125	2010.02.12	Email-Worm.VBS.Peach
McAfee	5889	2010.02.11	VBS/Peachy@MM
McAfee-Artemis	5889	2010.02.11	VBS/Peachy@MM
McAfee-GW-Edition	6.8.5	2010.02.12	Virus.Peach.2
Microsoft	1.5406	2010.02.12	Virus:VBS/FDPeach.A@mm.gen
NOD32	4860	2010.02.12	VBS/PeachyPDF.B
Norman	6.04.08	2010.02.11	-
nProtect	2009.1.8.0	2010.02.12	VBS.PDFWorm.A
Panda	10.0.2.2	2010.02.12	VBS/PeachyPDF.A
PCTools	7.0.3.5	2010.02.12	Malware.VBS-PeachyPDF
Prevx	3.0	2010.02.12	-
Rising	22.34.01.03	2010.02.11	-
Sophos	4.50.0	2010.02.12	VBS/PeachyPDF-B
Sunbelt	5671	2010.02.11	-
Symantec	20091.2.0.41	2010.02.12	VBS.PeachyPDF@mm
TheHacker	6.5.1.3.190	2010.02.12	-
TrendMicro	9.120.0.1004	2010.02.12	-
VBA32	3.12.12.2	2010.02.11	Email-Worm.VBS.Peach
ViRobot	2010.2.12.2184	2010.02.12	-
VirusBuster	5.0.21.0	2010.02.12	-

Figure 1: Peachy PDF e-mail worm bypassing Norman and TrendMicro AV

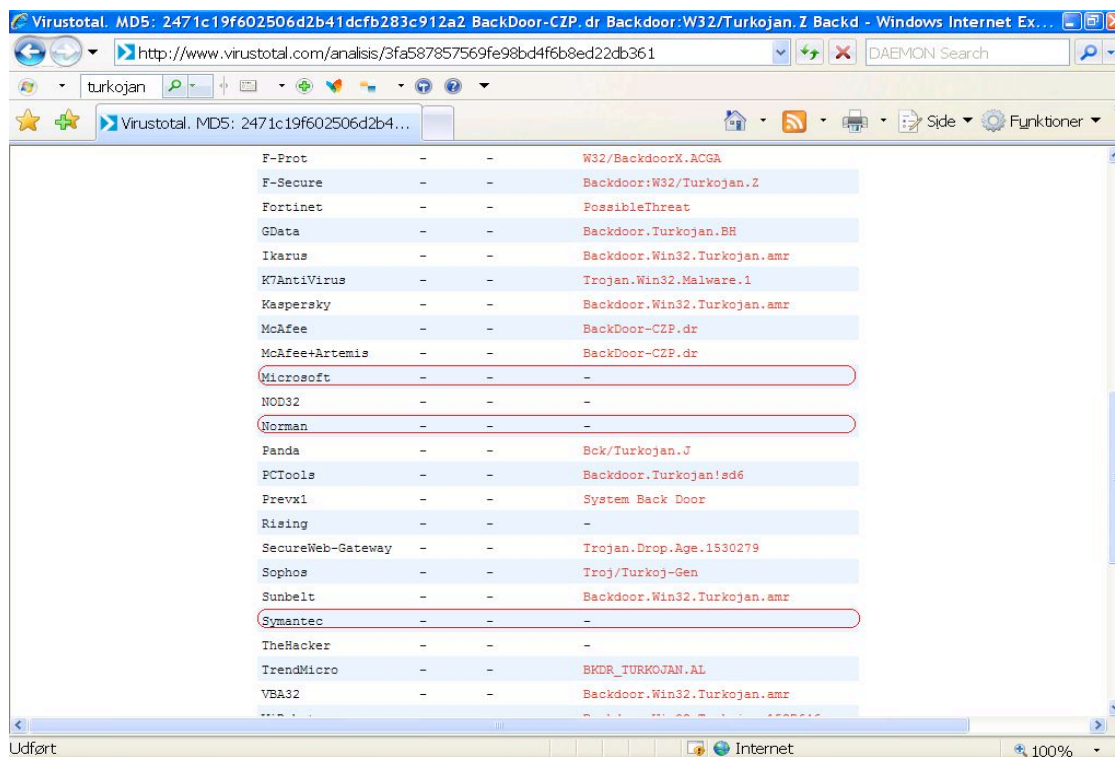


Figure 2: Turkojan Trojan bypassing Microsoft, Norman and Symantec AV

As seen from the figures above, not all AV products detect our two samples. Notably, well reputed products like TrendMicro, Microsoft, Norman and Symantec AV all miss at least one of the two malware samples. Actual executing this malware in a closed lab environment yields the same results. Please, note that AV vendors have been notified about missing the Turkojan Trojan due to the ease of attackers getting this sample from the homepage of the Trojan.

In addition, the samples chosen are in no way benign. Peachy is a PDF based e-mail worm found in the wild; its functionality will be revisited and explained in section 3.8 of this paper. Turkojan is a well known Trojan available for free at its authors' web site (CigiCigi Online, 2010). Turkojan is a backdoor, allowing the adversary to capture key strokes, listen in on audio and webcams, get passwords or password hashes, transfer files, and to get access to remote desktop on the compromised host. Figure 3 illustrates these functionalities.

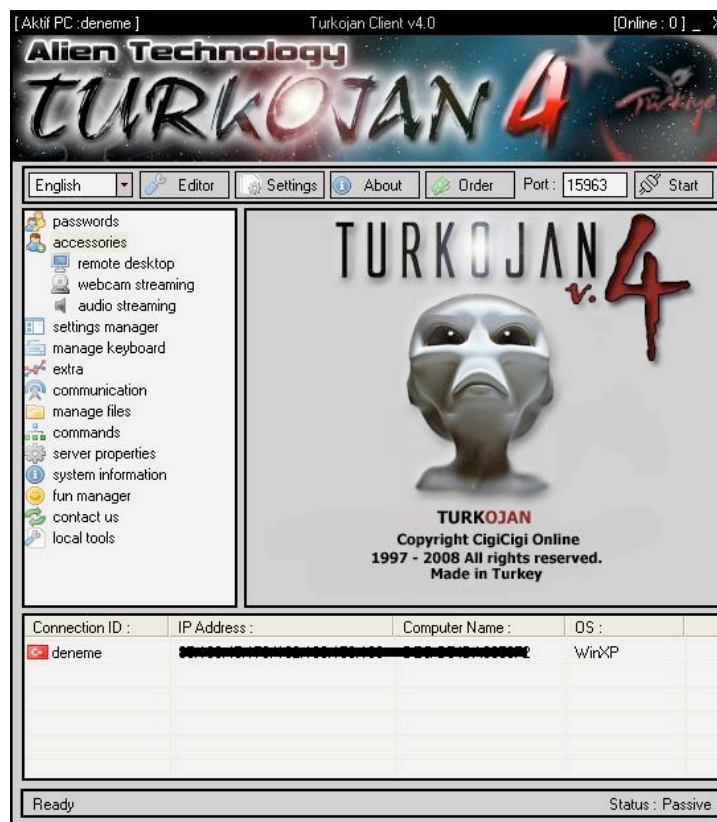


Figure 3: The adversary's interface as provided by Turkojan

AV Comparatives (August, 2009) conduct comparative tests of AV products. Their latest report tested more than 1.5 million malware samples against the 16 leading AV products, configured with their highest protection settings. The report showed the detection rate of these products to be 84,8% - 99,8%. It should be noted that that report suffers from the same limitations as the results from VirusTotal, i.e. they do not actually execute the malware to test whether the malware is detected during execution.

Consequently, even without a single modification to a well known piece of malware, it may sometimes be able to bypass our host-based AV defenses.

2.3. Modifying Signatures

Since host-based malware detectors often work by identifying malicious signatures, modifying these signatures in the malware is one way of bypassing such detectors. First step in this regard is to identify the malicious signature detected by the

AVs targeted for bypassing. The easiest way of doing this is usually to systematically wipe out parts of the malicious file with instructions that do nothing, e.g. No Operation Performed (NOP) instructions, then sending the resulting files to a service like VirusTotal up to the point where the malware is no longer detected. Whether or not the piece of malware works is usually irrelevant at this stage, as long as a valid file header is retained. Consequently, this technique can be used to extract the signature that AV detectors use to detect the malware.

After the signature in the malware has been identified, this must now be changed in order to successfully bypass the malware detector. As demonstrated by Ed Skoudis (2006) the easiest way of doing this, when applicable, is to simply hex edit the malware's signature. For instance many malware detectors still identify the backdoor Tini by a signature including the hardcoded port that the malware listens on for commands. Tini is a well-known backdoor identified by all 41 AV engines provided by VirusTotal prior to alteration. Tini listens with a command shell on port 7777, equal to 1e61 in hex. Consequently, it is possible to search for this part of the malware code, and simply change it to port 443 (01bb hex) instead. Figure 4 illustrates this process.

```

00000430: 00 00 66 c7 05 a8 31 40 00 1e 61 6a 10 68 a6 31 || ..fÇ."1@.aj.h|1
00000440: 40 00 ff 35 a2 31 40 00 e8 85 02 00 00 6a 05 ff || @.ÿ5ç1@.è!...j.ÿ
00000450: 35 a2 31 40 00 e8 7e 02 00 00 32 c0 a2 b6 31 40 || 5ç1@.è~...2Àç1@

00000430: 00 00 66 c7 05 a8 31 40 00 01 bb 6a 10 68 a6 31 || ..fÇ."1@.aj.h|1
00000440: 40 00 ff 35 a2 31 40 00 e8 85 02 00 00 6a 05 ff || @.ÿ5ç1@.è!...j.ÿ

```

Figure 4: Searching for, and changing, the Trojan's hardcoded listening port

Next, we need to test whether Tini still executes successfully. One quick way of doing this is to upload the file to the CWSandbox service offered free of charge from the University of Mannheim (2010). This site will execute the uploaded file in a sandbox environment resulting in a report presented to the user. The report includes information on file and registry changes, network activity and other technical details resulting from the execution of the uploaded file. As illustrated in figure 5, the modified version of Tini has indeed been successfully executed, and now listens on port 443.

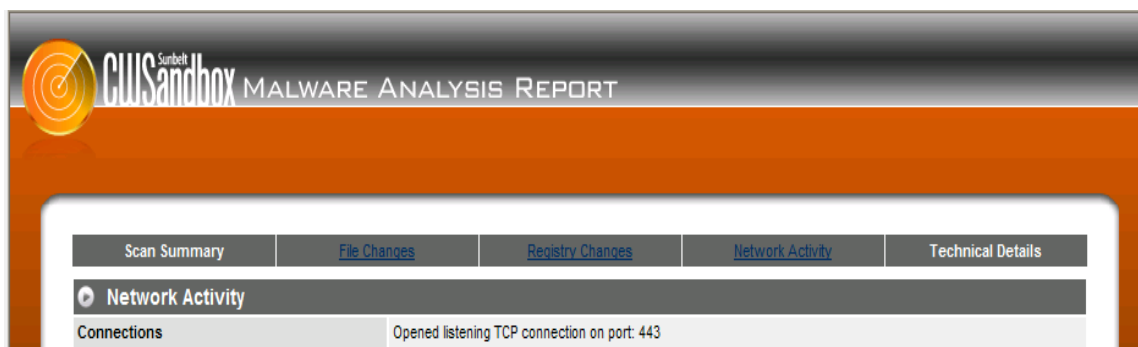


Figure 5: CWSandbox – verifying that the changed functionality works as planned

We have just modified one single signature in Tini through the easiest method available to us, namely hex editing. Uploading the changed version to VirusTotal will give us an indication of the bypassing rate achieved. As illustrated in figure 6, the minor modifications allow us to bypass almost half of all AV engines prior to execution.

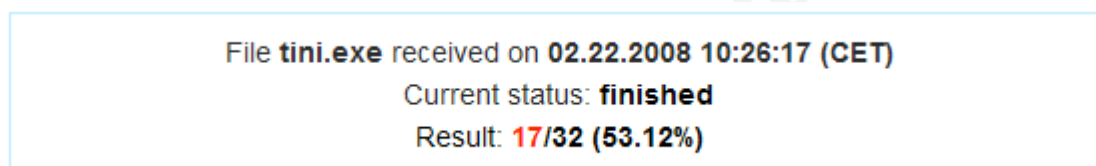


Figure 6: Percentage of host-based AVs detecting Tini after modification

It should be noted that sometimes the signatures used by the AV engines are crucial for the successful execution of the program, i.e. all instructions within the signature must be carried out and cannot be replaced. In these cases altering the signature involves debugging the program, adding several transformations such as inserting NOP-instructions into the signature or performing code transposition (Christodorescu, Jha, 2004; Aharoni, 2009).

An alternative method of bypassing host-based AV engines is to pack the piece of malware with a different packer. (Zeltser, 2009). Packers may be used to pack an executable using techniques ranging from simple XORing of the malware to compression and even encryption hereof. The malware is then unpacked during runtime. Since the packer changes the representation of the malware code, this often results in AVs being unable to detect it. Examples of often-used packers include UPX, ASPack, Petite, Neolite and Themida. Figure 7 illustrates the successful execution of Tini, while at the same time

bypassing McAfee VirusScan Enterprise v8.7.0i after the malware was packed with Themida from Oreans Technologies (2010).

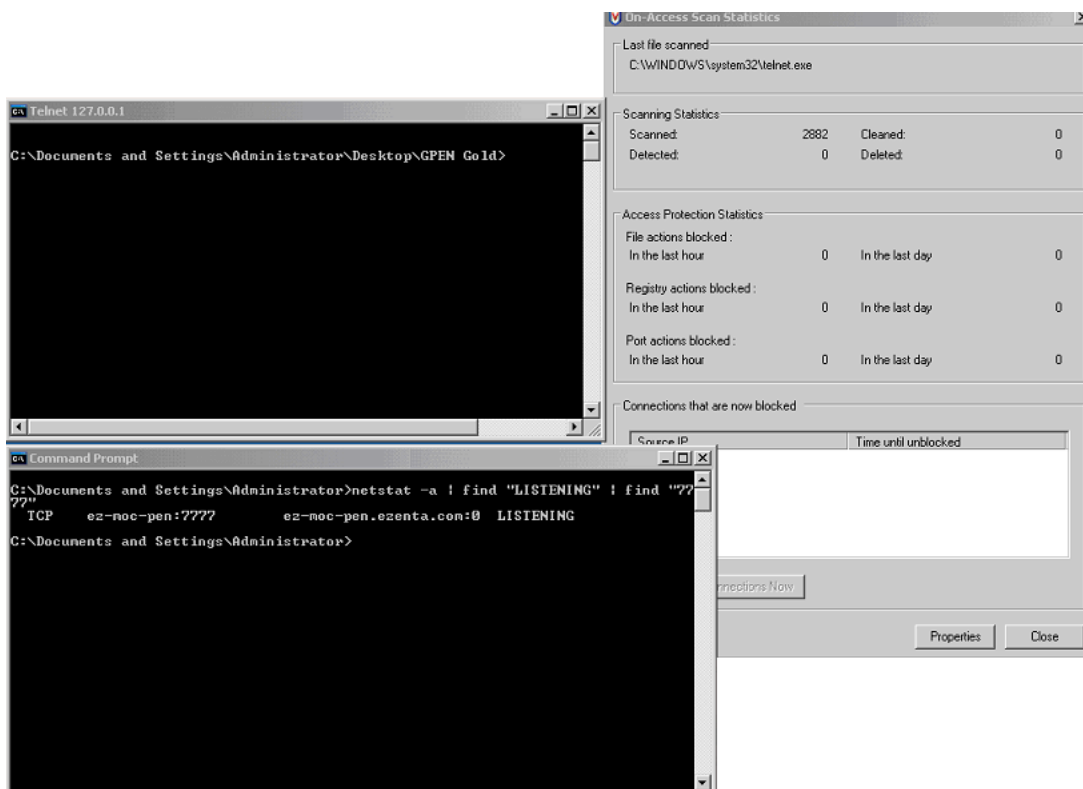


Figure 7: McAfee VirusScan Enterprise v8.7.0i bypassed using the Themida packer

2.4. New Malware for Targeted Attacks

Since host-based AV products are mainly able to detect malware containing known malicious strings, one way of bypassing these systems is naturally for the adversary to write his own piece of malware. Since this new piece of malware contains no strings that are already known to be malicious, usually it cannot be detected. This is especially true if the malware is kept simple by limiting its suspicious behavior to the bare minimum needed. Thus, one such proof-of-concept (PoC) piece of malware was developed by the author. The PoC tested is a backdoor receiving shell commands from the adversary via a recursive DNS covert channel, as explained shortly. Figure 8 illustrates this piece of malware bypassing all 41 host-based AV products tested against.



Figure 8: Bypassing 100% of Signature-based host AVs via custom-made malware

The results from VirusTotal have been confirmed in multiple real world penetration tests against some of the largest organizations in Northern Europe, where the PoC malware was successfully executed without being detected. The host-based AV products used by these organizations comprise Symantec v11.0.4, Cisco Security Agent v6.0 and McAfee VirusScan v13.11 with their latest database updates. In the study regarding the 16 leading AV products and their abilities to detect new malware by AV Comparatives (May, 2009), G-Data Antivirus had one of the highest detection rates at 60% of their test set with 22.685 malware specimens. Consequently, for this paper G-Data Antivirus was installed and the custom-made malware was scanned using this product during both storage and execution. As illustrated in figure 9, the custom-made malware bypassed this AV product as well.

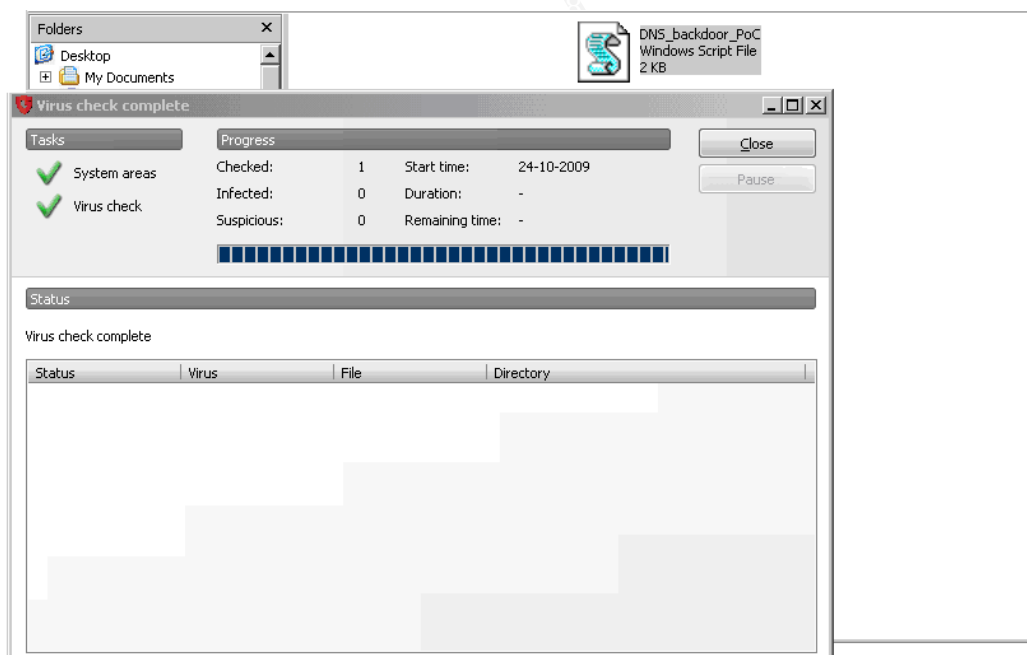


Figure 9: G Data AntiVirus 2010 v20.1.0.50 bypassed using custom-made malware

Going beyond AV avoidance, how can adversaries make sure that they can also communicate with the backdoor after infection has taken place? Clearly, this becomes a matter of bypassing the organization's firewall as well. Several methods are currently used for this by malware. The most effective of these relies on the client making an outbound connection to a server controlled by the adversary (Skoudis, 2009). This method is more effective than the old method of setting the malware to listen on a port on the infected system, since a correctly configured firewall will deny traffic originating from external addresses to these client ports. The methods described do, however, suffer from some notable limitations as they initiate a connection *directly* to external systems. If the infected system is denied web-access by the firewall, malware using these ports will not work. Likewise, if the infected system is not allowed any direct communication towards the Internet at all, the piece of malware fails.

Our goal then is to find a covert channel for our communication with the malware, whereby our chance of getting blocked by the firewall is the least likely. The perfect protocol for this covert channel is arguably DNS. Today, very few of us address systems solely by their IP-address. Thus, if an infected system needs any information about external systems whatsoever, an internal DNS server resolving these queries is available to this system. Consequently, even if the infected system is allowed no direct access to the Internet at all, we can still communicate with this system from the Internet *indirectly* by implementing a covert channel utilizing recursive DNS lookups. Figures 10 and 11 illustrate the author's implementation.

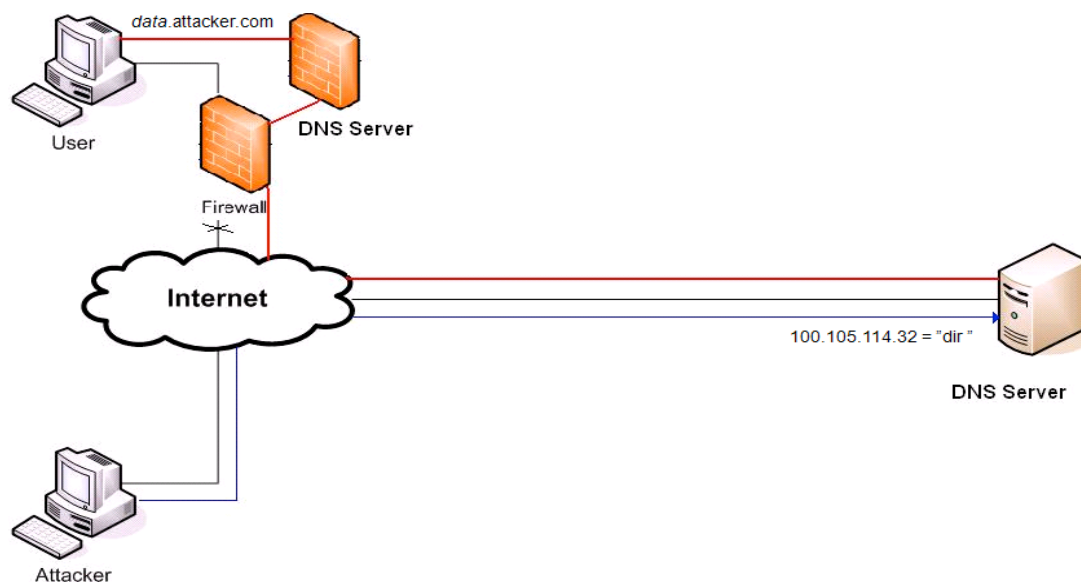


Figure 10: Recursive DNS covert channel design

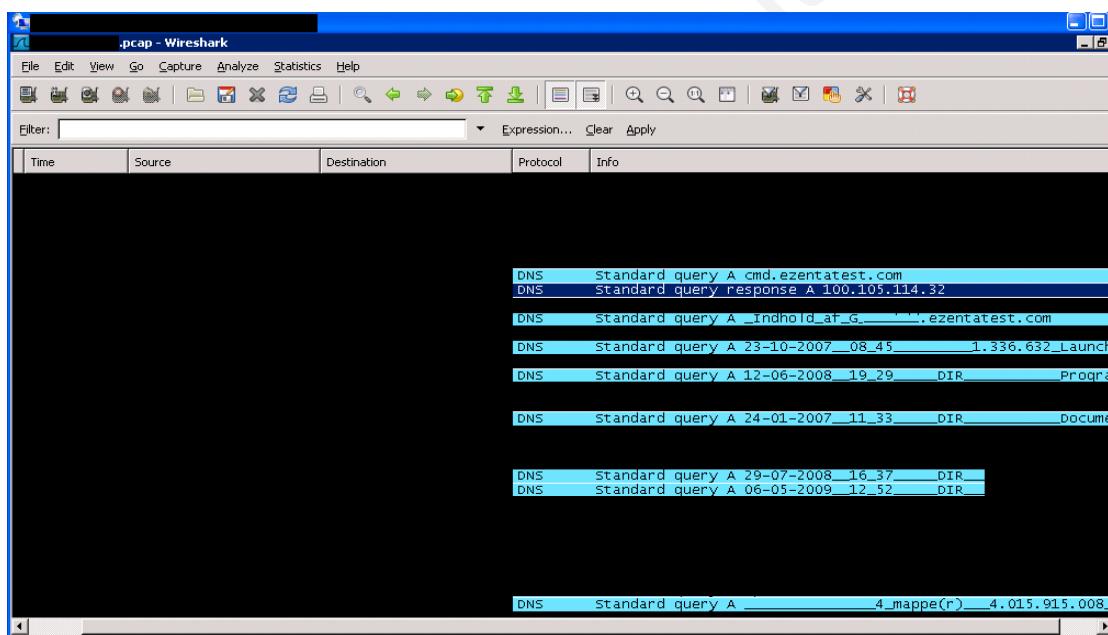


Figure 11: Sending and receiving data to/from the backdoor

As seen in figure 10, the infected system cannot reach the Internet directly due to firewall restrictions. Instead it makes a query for cmd.adversary.com using the internal DNS server available to it. This query ends up on the DNS server for the domain adversary.com, a server under the control of the adversary. The adversary's DNS server answers the result of the query cmd.adversary.com to be 100.105.114.32. The infected system receives this response, converting each number to their ASCII representation, i.e.

100=d, 105=i, 114=r, 32=carriage return – a command to list the contents of the working directory of the piece of malware. Then the infected system proceeds to execute the command received, sending its output to the adversary via recursive DNS queries. Figure 11 illustrates this actual attack from the side of the external adversary.

3. Bypassing SMTP & HTTP/SSL Malware Defenses

We showed in last section that by creating new piece of malware for targeted attacks, or modifying a known piece of malware to pass signature checks, the adversary may be able to bypass host-based AV engines. Once on the host, this piece of malware may bypass firewall restrictions, e.g. by using a covert channel over recursive DNS queries for controlling the malware. Consequently, malware must be detected *before* it reaches the host. Many organizations try to achieve this by implementing SMTP and HTTP/SSL AV gateways. All SMTP and HTTP/SSL traffic originating from the internal network of the organization towards the Internet must then pass through these gateways. The gateways represent an extra layer of security that adversaries need to bypass in order to successfully penetrate the internal clients of the targeted organization. This section deals with methodologies for bypassing such gateways.

3.1. Methodology

All tests mentioned in this section have been conducted as professional penetration tests for some of the largest organizations in Northern Europe. The tests were conducted during the years 2004-2009. Tests and the creation of the test files in this paper have been conducted by the author. Permission to use the data in the anonymous and statistical way displayed in this paper has been granted through the author's work at nSense International.

Importantly, for the malware to be counted as a successful bypass it must actually bypass all the organizations' AV gateways of the type tested. That is, many organizations have more than one AV gateway product including cloud detection mechanisms, especially for SMTP, and in order to reach the clients of the targeted organization the

malware must bypass all these gateway defenses. Consequently, the statistics concern not just bypassing one AV product, but rather bypassing all the AV gateway defenses deployed by the tested organization. Thus, the statistics will give us an indication of the degree to which a specific piece of malware is capable of bypassing the gateway defenses in real-world situations.

3.2. Dangerous File Formats

SMTP and HTTP/SSL AV gateways typically allow filtering by file format. As illustrated in section 2 this functionality is highly called for, as custom-made malware was not detected once on the host. As an example, the malware designed by the author in section 2.5 was not detected by any of the 41 host-based AVs prior to execution. Moreover, writing malware in a little known format like WSF also makes it more likely to successfully bypass SMTP/HTTP AV gateways as well. Diagram 1 (custom EXE malware) and diagram 2 (custom WSF malware) illustrate this point.

As can be seen from the diagrams below, using WSF for malware instead of EXE alone increased the success rate of bypassing real world SMTP AV gateways by notable 28 percent points (from 12% to 40%). This is mainly due to organizations often utilizing black listing instead of white listing: Black listing works by defining a list of file formats that the AV gateway *does not* allow. This is the contrary to white listing, which works by defining file formats that the gateway *does* allow, with all other formats being denied. From a strictly security point of view, the latter option is superior. The problem with black listing is that there is virtually an unlimited number of different file formats in existence. Thus, the changes for successfully blocking all formats that may contain executable code are very bad.

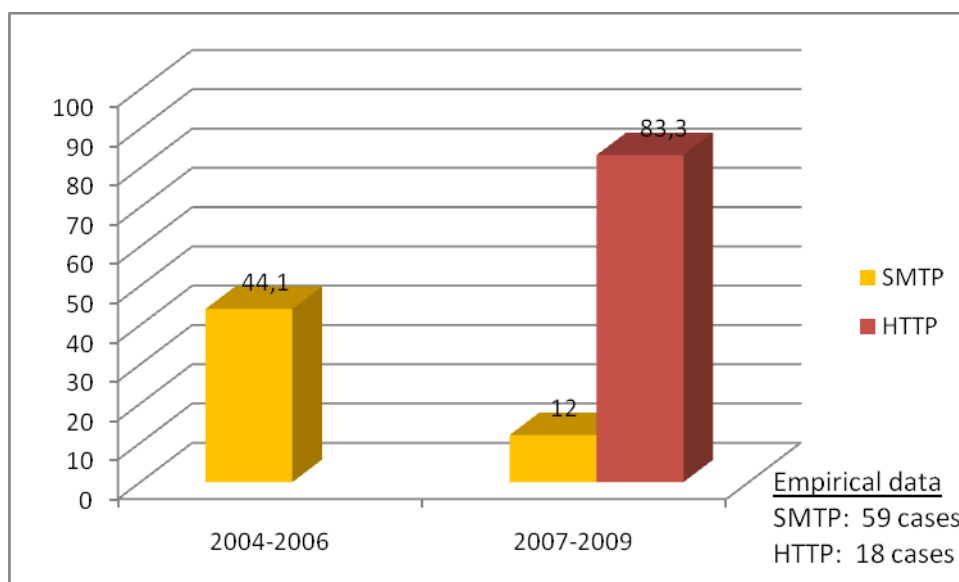


Diagram 1: Percentage of AV gateways bypassed via custom EXE malware

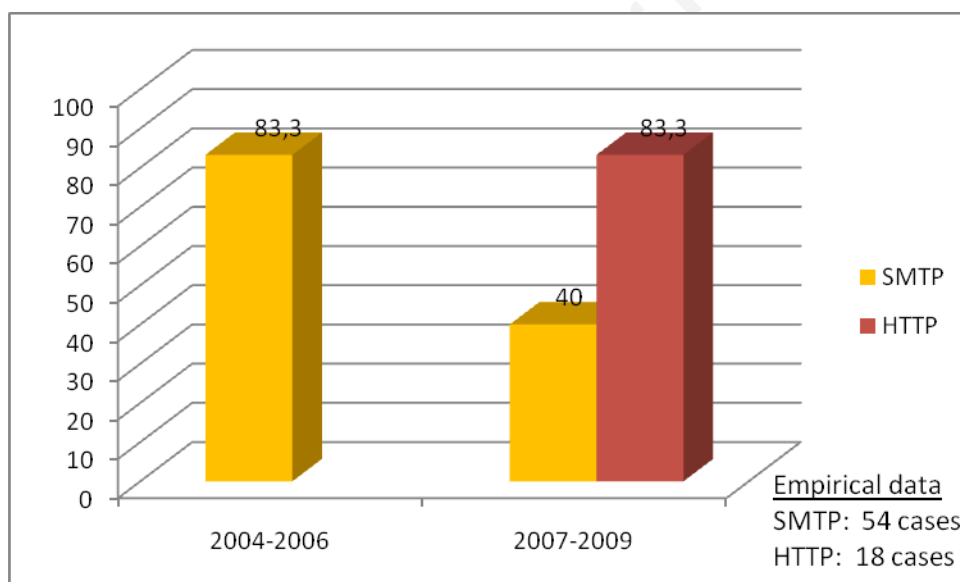


Diagram 2: Percentage of AV gateways bypassed via custom WSF malware

The current success rate of bypassing HTTP/SSL AV proxies via custom malware alone is 83,3%, regardless of the file format used. This is the same rate we saw for WSF malware on SMTP AV proxies back in 2004-2006. And for EXE malware, it is worse than any rate recorded for SMTP AV proxies during the test period from 2004.

The diagrams do, however, also show a good development in organizations' awareness of these problems, as illustrated by a 43,3 percent points' drop in the success

rate for WSF files to bypass AV gateways from 2004-2006 to 2007-2009. WSF is however far from the only format an adversary may create a custom piece of malware in.

Consequently, it is relevant for defenders and adversaries alike to know a list of file formats that may execute malicious code directly, when the victim executes them.

Table 1 contains such a list compiled by the author.

The list was created with its offset being the list of dangerous file types released by Microsoft (2010), which, however, does include a large number of files that do not meet the criteria for our list – namely, that files must be able to execute malicious code simply by double clicking them. Thus, all file formats on Microsoft's list were tested to this end. When a file was found capable of executing malicious code, it was cross referenced with the registry on Windows XP Professional in order to find more directly executable file formats.

File type	Description	Executed by
bat	DOS batch file	shell32.dll
chm	HTML Help Compiled Help File	hh.exe
cmd	Command File	shell32.dll
com	DOS command file	shell32.dll
cpl	Windows Control Panel Extension	rundll32.exe
css	Hypertext Cascading Style Sheet	shell32.dll
exe	Executable file	Directly executed
hlp	Windows Help File	shell32.dll
hta	Hypertext Application	mshta.exe
jar	Java Archive	JRE
js	JavaScript Source Code	WScript.exe
jse	JScript Encoded Script File	WScript.exe
lnk	Windows Shortcut File	rundll32.exe
msc	Microsoft Management Console Snap-in Control File	mmcbase.dll
msi	Windows Installer File	msiexec.exe
msp	Windows Installer Patch	msiexec.exe
pif	Windows Program Information File	Directly executed
reg	Registry Data File	regedit.exe
scr	Windows Screen Saver	rundll32.exe
shs	Shell Scrap Object File	shscrap.dll
vbe	VBScript Encoded Script File	WScript.exe
vbs	VBScript Script File	WScript.exe
wsf	Windows Scripting File	WScript.exe
wsh	Windows Script Host Settings File	WScript.exe

Table 1: Directly executable Windows formats

The list resulting from this work may be divided into four categories.

Executable binaries (com, exe, jar, msi, msp, shs)

Example - jar: Java archive. Not listed as a dangerous file format by Microsoft (2010). With Java being deployed on most clients today this makes jar an interesting format for malicious code.

Executable scripts (bat, cmd, js, jse, vbe, vbs, wsf)

Example – wsf: Windows Scripting File. WSF is interesting because few people know it, and consequently, it is rarely black listed. It may contain scripts such as Perl or VBScript often-used as languages for malware. The script type is defined in the header of the file:

```
<job id="whatever"><script language="VBScript">
```

Execution through built-in OS applications (chm, cpl, css, hlp, hta, msc, reg, scr)

Example – reg: Registry Data File. It directly alters the registry of the client by executing regedit.exe with the data file as the argument.

Executable reference files (lnk, pif, wsh)

Example – lnk: Windows Shortcut File. Simply links to a program. It may contain parameters allowing for the execution of potential malicious code, e.g. by linking to cmd.exe /C <command>. Several commands may be executed at once by using the & operand.

3.3. Renamed Malware

It is often possible to bypass AV gateways by simply renaming the file extension of the malware, e.g. from EXE or VBS to an arbitrary extension such as XYZ. Making this change would however require the victim to rename the file to its original format, before execution can take place. Through social engineering, this is often as simple as asking the user to do it – an act that is not unusual in many organizations, when delivering files through SMTP.

Whether or not this attack will succeed largely depends on how strong the AV gateway's magic byte recognition is. Provided it is implemented at all, magic byte

recognition tries to identify the file format not just by looking at the file extension, but rather by looking at the file's header. Magic byte recognition works by searching for header strings known to be unique to specific formats.

In diagram 3 a custom EXE malware had its extension renamed to XYZ prior to being sent through the organizations' AV gateways. Compared to diagram 1 regarding the same custom-made malware sample a moderate increase in its success rate of bypassing SMTP AV gateways by 16,6 percent points (from 12% to 28,6%) can be seen.

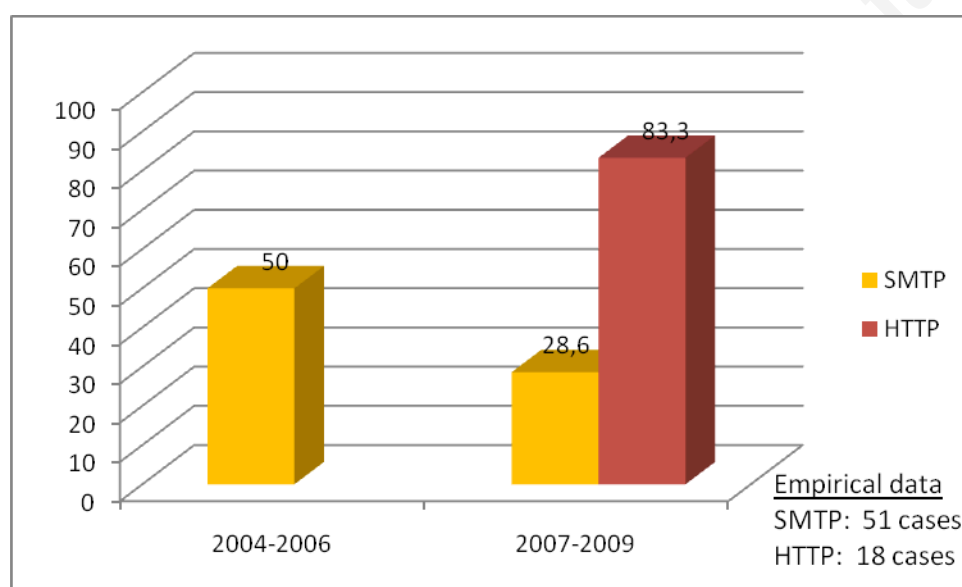


Diagram 3: Percentage of AV gateways bypassed via custom EXE malware with renamed file extension

While magic byte recognition often functions well, some file formats do not put strict requirements on their header contents. VBS, one of the languages often-used for Internet worms, is one such example: simply adding an unusual string like “On Error Resume Next” to the beginning of the custom malware will allow it to bypass most AV gateways. Figure 12 illustrates the manipulated VBS header, while diagram 4 illustrates its success in bypassing AV gateways.

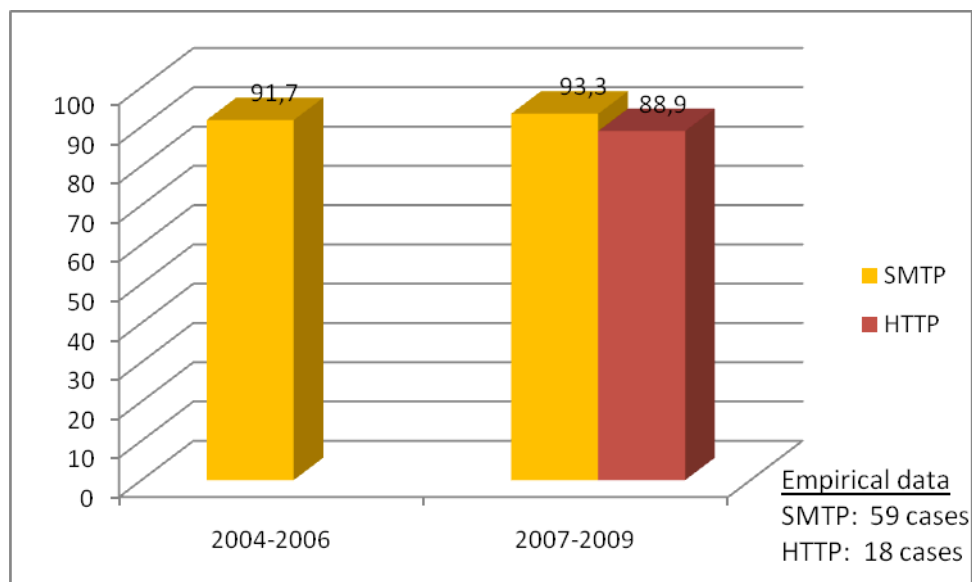


Diagram 4: Percentage of AV gateways bypassed via custom VBS malware with renamed file extension and manipulated header

```
On Error Resume Next
Set WSHShell = CreateObject("wscript.shell")
lcValue1 = WSHShell.RegRead("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkCards\6\ServiceName")
WSHShell.RegWrite "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters\Interfaces\Tcpip_" & lcValue1, 2, "REG_DWORD"
```

Figure 12: Manipulated VBS header

Consequently, renaming file extensions while at the same time manipulating the file header greatly improves the malware's changes of bypassing AV gateways.

3.4. Compression

Compression is yet another method that may be used by adversaries to bypass AV gateways. The testing below has been done with the very well-known Trojan SubSeven. Prior to compression, all AV engines correctly identified the trojan as indicated in figure 13. Diagram 5 shows the result after standard ZIP compression.



Figure 13: 100% success rate in detecting SubSeven Trojan

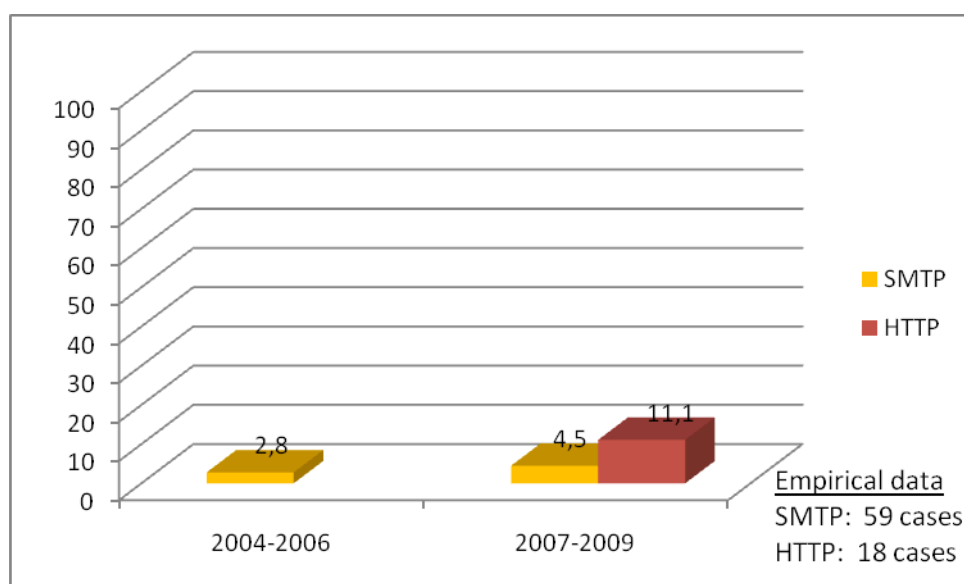


Diagram 5: Percentage of AV gateways bypassed via known malware in ZIP archive

As illustrated above, using even the most common compression type gives the malware an additional chance of bypassing AV gateways of approximately 5-10 percent points. Next, diagram 6 illustrates the success rate of bypassing AV gateways, when the ZIP archive containing the known malware is protected with a password.

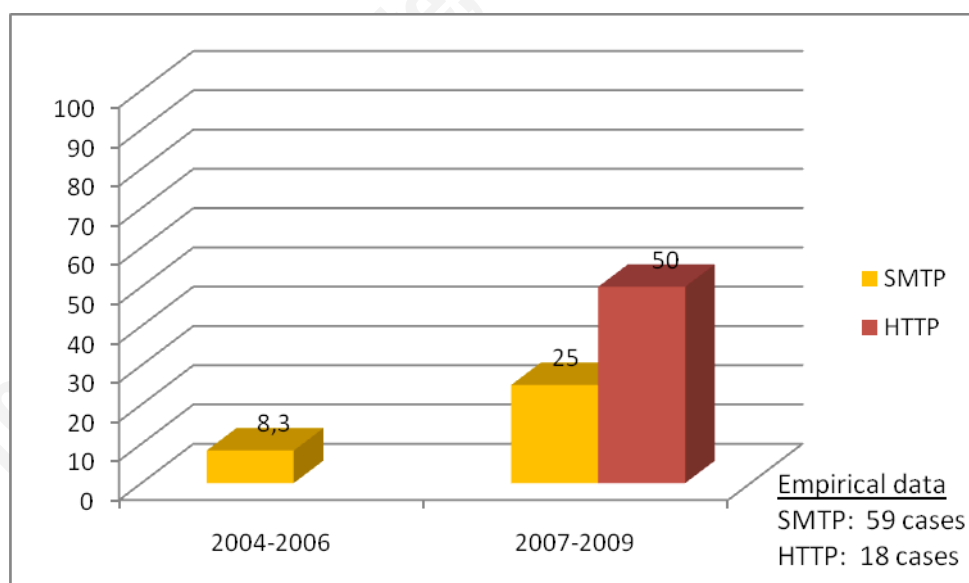


Diagram 6: Percentage of AV gateways bypassed via known malware in password-protected ZIP archive

Thus, password protecting the ZIP archive containing known malware may improve the malware success rate of bypassing AV gateways by 20,5 percent points for SMTP defenses and 38,9 percent points for HTTP defenses. It should be noted that AV gateways have no way of checking the contents of password-protected archives as these are often encrypted (see discussion in section 3.7). Stopping these attacks is therefore a matter of denying password-protected archives. Unfortunately, as organizations have become better at denying various executables, at the same time many have started to allow password protected archives, limiting the gain in security.

Consequently, defenders need to know what types of password-protected archives to scan and deny, and penetration testers need to know these archive formats in order to have a full understanding of the attack surface available to them. Thus, table 2 contains a list of commonly supported compression formats compiled by the author.

File type	Description
7z	7-Zip Compressed File
ace	Ace Compressed File
arj	ARJ Compressed Archive
bz	Bzip UNIX Compressed File
bz2	Bzip 2 UNIX Compressed File
cab	Cabinet File
gz	Gzip Compressed Archive
img	Disk Image
iso	ISO-9660 CD Disc Image
lha	LHA Compressed Archive File
lzh	LZH Compressed Archive File
r00-r29	RAR Split Compressed Archive
rar	RAR Compressed Archive
rev	RAR Recovery Volume File
tar	Tape Archive File
taz	.TAR.Z Compressed File
tbz	BZIP2 Compressed TAR
tbz2	BZIP2 Compressed TAR
tgz	UNIX Tar File Gzipped
uu	Uuencoded File
uue	Uuencoded File
xxe	Xxencoded File
z	UNIX Compressed Archive File
z00-	ZIP Split Compressed Archive
zip	ZIP Compressed Archive

Table 2: Compression formats commonly supported

3.5. Embedded Malware

Embedding the malware into other types of documents is yet another possible method for adversaries to bypass AV gateways. Due to their widely adopted use, Microsoft Office PowerPoint, Excel and Word documents will be used as examples in this section of the paper. For all these formats any type of executable is allowed to be copied and pasted as “objects” into the documents. Getting the victim to execute the malware would then require the user accidentally opening the document and double clicking on the embedded object, or being convinced to do so. Diagram 7 shows the success rate when embedding known malware (the SubSeven Trojan) into a normal Word document. SubSeven was detected by all 41 AV engines at VirusTotal prior to embedding it in Microsoft Word.

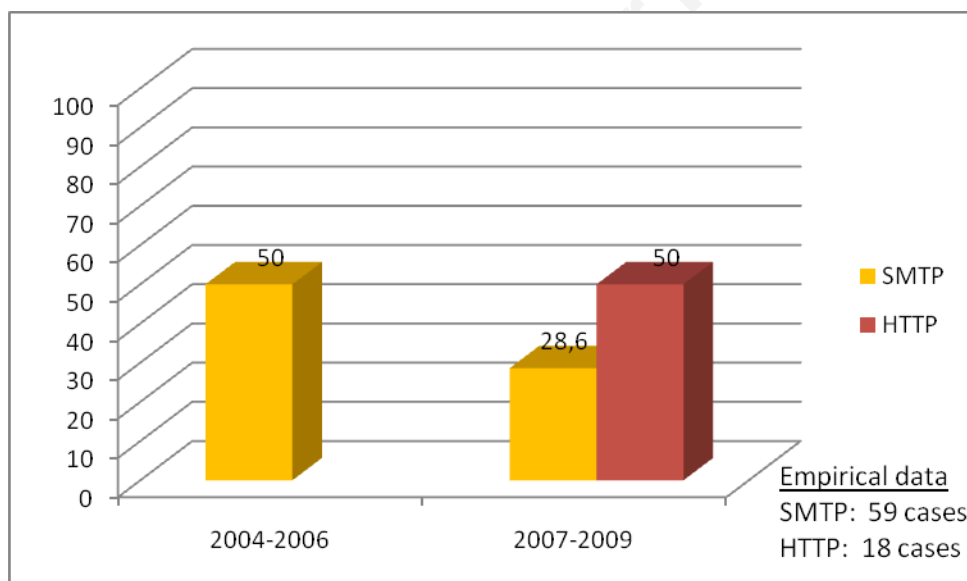


Diagram 7: Percentage of AV gateways bypassed via known malware in MS Word document

As illustrated, embedding known malware into a MS Word document increased the malware success rate of bypassing AV gateway defenses by 28,6 percent points for SMTP defenses and 50 percent points for HTTP defenses. Again, we see organizations improving on this issue with 50% being vulnerable in the 2004-2006 period compared to 28,6% being vulnerable over SMTP today. HTTP defenses reflect a relatively lower level of security comparable to the 2004-2006 state of SMTP malware defenses.

These results are notable, especially because the tests were carried out using the most common of MS Office formats, namely the MS Word .doc format. This is far from the only MS Office format allowing executables or macros to be embedded. Researching the registry entries concerning file formats that will open either PowerPoint, Word or Excel when double clicked, while at the same time being able to contain embedded malicious executables or macros, resulted in the list illustrated in table 3. The list is from a standard MS Office 2007 installation. Noteworthy is that its length exceeds the list of directly executable formats in the Windows OS. All these formats should be either denied or scanned for malware by gateway defenses. Consequently, the amount of file formats AV gateways and defenders need to keep track of just to support MS Office documents is very large.

Filetype	Description
csv	Excel comma-separated variables
doc	Word document
docm	Word macro document
docx	Word document
dot	Word template
dotm	Word macro template
dotx	Word template
iqy	Excel Web Query
odc	Excel data connector
odp	Powerpoint OpenDocument (Ver 2) Presentation
pot	Powerpoint template
potm	Powerpoint macro template
potx	Powerpoint template
ppa	Powerpoint add-in
ppam	Powerpoint macro add-in
pps	Powerpoint slideshow file
ppsm	Powerpoint slideshow macro file
ppsx	Powerpoint slideshow file
ppt	Powerpoint presentation file
pptm	Powerpoint macro presentation file
pptx	Powerpoint presentation file
pwz	Powerpoint wizard
rtf	Word Rich Text Format
sldm	Powerpoint slideshow macro file
sldx	Powerpoint slideshow file
slk	Excel data exchange file
wbk	Word back-up file

wiz	Word wizard
xla	Excel add-in
xlam	Excel macro add-In
xlk	Excel backup
xll	Excel add-in
xlm	Excel macro document
xls	Excel document
xlsb	Excel binary project
xlsm	Excel macro document
xlsx	Excel document
xlt	Excel template
xltm	Excel macro template
xltx	Excel template
xlw	Excel workspace

Table 3: MS Excel, PowerPoint and Word formats capable of hiding malware

3.6. Encryption

Encryption is another optional means for adversaries to bypass AV gateways. This is due to AV gateways generally not being able to scan the contents of files protected by encryption. Basically, these attacks can be distributed into 3 categories:

Self-decrypting encrypted executables: Executables that contain the encrypted pieces of malware and prompt the user for a password when executed. Generally, this type is not very successful in bypassing malware defenses, since plain executables are often blocked, with identifiably encrypted content at times being blocked as well.

Encrypted data files: Data files requiring a specific application and password to be decrypted. This type is often successful in bypassing AV gateways, unless the gateways identify and disallow encrypted content. Generally, not the most efficient attack overall, since the adversary needs to know what application the target is using for decryption, or to convince the target into downloading and installing the required application.

Encryption inside applications: Utilizing encryption offered inside widely used applications. Again, MS Office documents support encryption as well as Information Rights Management (IRM). We note that the author's research concerning IRM seems promising, with AV gateways being split on exactly how to treat these files. Some drop

them, others put them in quarantine, with yet others sending them through without scanning for malware.

Consequently, using encryption or protection mechanisms inside widely used applications seem to be the best way of hiding malware via encryption.

3.7. Exploits

An alternative to hiding executables through various means is to use data files as exploits. This way, the format of the malware can be one allowed by most organizations, such as Excel or PDF documents. One important prerequisite for these attacks to succeed, however, is that an application on the victim's system is vulnerable to the exploit tried. Meaning, the victim organization must have a weak patch management policy, or the exploit must abuse undisclosed vulnerabilities in the target software.

As indicated in section 2, AV engines often fail to recognize exploits. An Excel document exploiting a known vulnerability in that application was sent through various organizations' AV gateways. The results are shown in diagram 8.

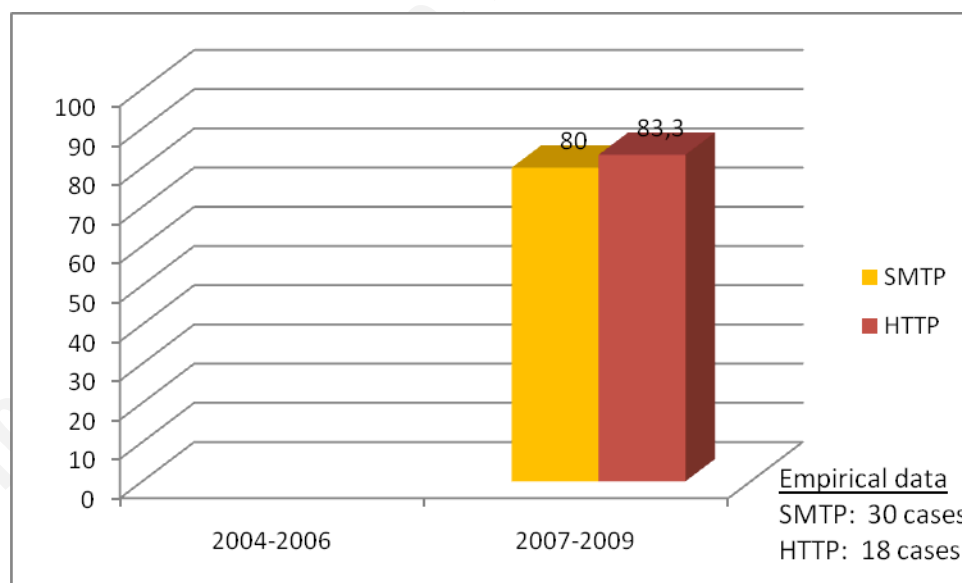


Diagram 8: Percentage of AV gateways bypassed via known Excel exploit

As illustrated above, the exploit, based on a known vulnerability in Excel, bypasses 80-83,3% of AV gateways. When the exploit file is intercepted, the reason is

either that the gateway does not allow Excel documents, or that it correctly identifies the sent file to contain a known exploit against Excel.

Consequently, malware using exploits based on data files is yet another important area for defenders and adversaries to take into consideration. Current malware is actively using this technique to bypass unsuspecting users and AV gateways. As an example, the PDF e-mail worm called Peachy, also mentioned in section 2, appears to be a standard PDF document on the outside. When loaded into a vulnerable PDF reader, however, this data file creates several malicious files on the victim's system, makes registry changes to make sure it is reloaded upon reboot, and even makes an outbound connection to a web server under the control of the adversary. Figure 14 illustrates this.

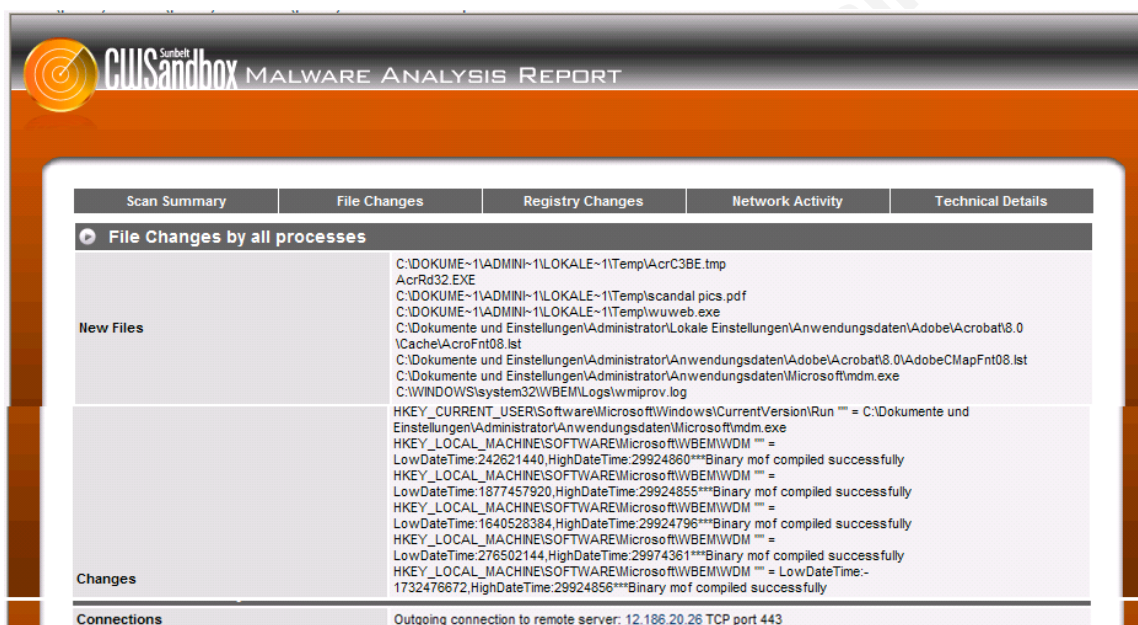


Figure 14: Activities of the Peachy PDF e-mail worm

3.8. Steganography

Steganography is defined as “the art or practice of concealing a message, image, or file within another message, image, or file” (Merriam-Webster Incorporated, 2010). In this context, that means hiding malware inside pictures, movies, music, or such. This can be achieved by making subtle changes to the original files, too small to be noticeable by the human mind. Minimal changes to the pixels a picture consist of, is an example of this.

Diagram 9 shows the success rate of getting a BMP picture with the known SubSeven Trojan inside through various AV gateways.

As indicated below, we are looking at a 100% success rate. Based on 77 cases, this attack was never stopped. This is not as bad as it sounds, though. The AV gateway has no way of detecting the malware, since to them, they are just looking at a normal picture, corresponding to the legit format of a BMP file. In addition, the actions required by the victim in order to execute the hidden malware is substantial. That is, the victim must download the same steganography program as used by the adversary, and then extract the hidden malware.

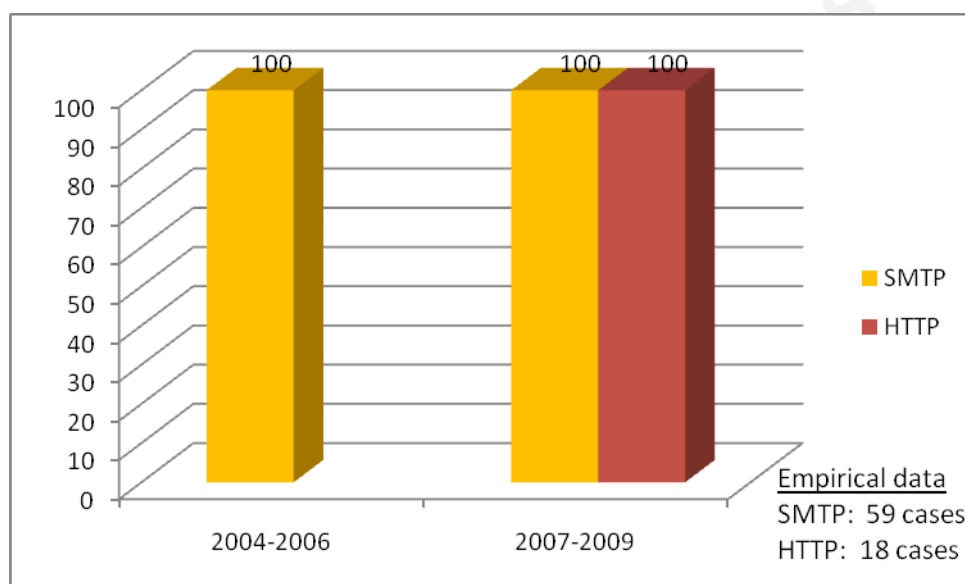


Diagram 9: Percentage of AV gateways bypassed via known hidden malware in BMP picture

Still, this is an interesting proof of concept (PoC), showing that SMTP and HTTP most often can be used to transfer even known malware. In the case of insiders, or an external adversary having compromised an internal host, this may be an interesting method to securely transfer malware.

3.9. Out of Band Attacks

The term “out of band” refers to using a different communication channel, than the one from which the communication originated. In the context of this paper, an example of this could be changing communication from SMTP to HTTP in order to

improve the malware's changes of bypassing the AV gateways. Table 4 illustrates the difference between the vulnerability to bypassing attacks of SMTP vs. HTTP gateways based on the former findings of this paper.

As indicated below, depending on the type of malware used, changing the communication channel used from SMTP to HTTP could allow for an increased success rate of up to 71,3 percent points. The prime example of carrying out this attack is as simple as sending the victim an e-mail, asking him to download the malware from a website. Social engineering techniques may be utilized, e.g. making the e-mail appear to come from an inside web-master, using cross site scripting to create an URL for the malware belonging to the target organization, etc.

Type	SMTP	HTTP	Gain
Dangerous File Formats (EXE)	12,0%	83,3%	+71,3 pp.
Dangerous File Formats (WSF)	40,0%	83,3%	+43,3 pp.
Renamed Malware (EXE)	28,6%	83,3%	+54,7 pp.
Renamed Malware (VBS)	93,3%	88,9%	-4,4 pp.
Compressed Malware (no password)	4,5%	11,1%	+6,6 pp.
Compressed Malware (password)	25,0%	50,0%	+25,0 pp.
Embedded Malware	28,6%	50,0%	+21,4 pp.
Exploits	80,0%	83,3%	+3,3 pp.
Steganography	100%	100%	0 pp.

Table 4: Gained success rate of bypassing AV gateways by changing communication, SMTP to HTTP

3.10. Encoding & Encapsulation

Throughout this paper we have seen the security-wise bad performance of HTTP AV gateways. In multiple scenarios, the failure rate of these systems have been proven to be as high as 83,3%. Making matters worse, these numbers are based on clear text HTTP without any additional encoding. Adding the additional encoding described in this section to the attack, will normally increase the malware's success rate. Overall, three different types of encoding or encapsulation will be described: SSL, MSN and file encoding.

SSL encapsulation is just as easy as it sounds. Most organizations tested simply do not have HTTP SSL functionality on their AV gateways. Meaning, that if a user downloads anything over SSL, the AV gateway will never be able to deny this malware, even if it is malware with known malicious signatures.

MSN encapsulation refers to an attack, where the piece of malware is sent using MSN. Some HTTP AV gateways support MSN over HTTP. However, the format for MSN over HTTP differs slightly from the plain HTTP format. This results in some HTTP gateways missing known malware. Figure 15 illustrates this, with the Love Letter virus being successfully transferred from the HTTP AV gateway to the victim.

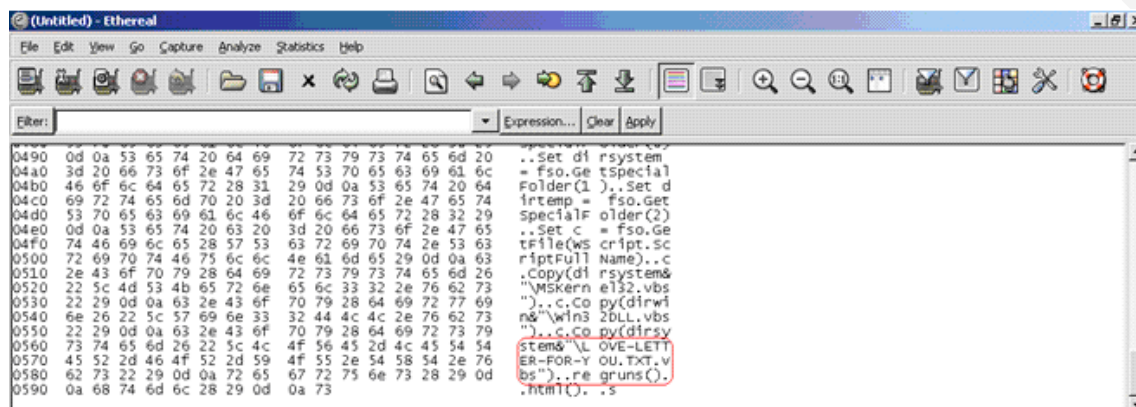


Figure 15: Love Letter virus bypassing AV HTTP gateway via MSN

Finally, even if SSL functionality is implemented, it will often fail to handle various MIME (Multipurpose Internet Mail Extensions) encodings. These encodings are often-used by web-mails to store and transfer mail attachments. Figure 16 illustrates a number of file type encodings, which HTTP/SSL AV gateways often fail to handle correctly according to tests. Usually, this results in the file being delivered to the victim, even if it contains known malicious strings or unwanted dangerous file types.

test14.asx	88 k	[video/x-ms-asf]
test18.vbs	0.4 k	[application/octet-stream]
test25.cmd	0 k	[application/octet-stream]
test26.reg	0 k	[application/octet-stream]
test41.vbs.jpg	1.8 k	[text/plain]
test43.xls	6.7 k	[application/vnd.ms-excel]
test44.wsf	0.3 k	[application/octet-stream]

Figure16: Various malware bypassing AV HTTP/SSL gateway via file encoding

4. Conclusion

By creating new malware for targeted attacks, or modifying the code that matches signatures known to the targeted AV product, the adversary may be able to bypass host-based AVs. Once on the host, this piece of malware may bypass firewall restrictions, e.g. by using a covert channel over recursive DNS for controlling the malware. Consequently, preventing foreign executables from reaching the clients in the first place is of uttermost importance. This involves scanning the external network traffic to the clients, exemplified in this paper by SMTP and HTTP/SSL gateways.

Yet, there is virtually no information available in the public domain as to penetration testing or bypassing AV gateways. The paper demonstrated the need to start conducting such penetration tests, proving that many of these defense systems may be bypassed. Several methods such as using rare (but dangerous) file formats, renamed file extensions, manipulated file headers, compression, embedded malware into other formats such as MS Office documents, encryption, steganography, out of band attacks, and exploits are available to this end. This is particularly true if the malware has been newly created by the adversary himself for a specific target (i.e. targeted attacks), or if the signature in the malware has been changed by the adversary.

HTTP AV gateways proved to be especially vulnerable to bypass attacks with success rates often higher than 80% for custom-made malware. In addition, most organizations do not have SSL gateways and the few that do are often prone to encoding or encapsulation attacks. Consequently, if the victim downloads a malicious file over SSL, with the adversary having encoded this file, the chances of the HTTP/SSL AV gateway detecting it are very poor.

5. References

Aharoni, Mati. *Cracking the Perimeter v.1.1*. Offensive Security LLC: 2009

AV Comparatives. *Anti-Virus Comparative No. 23, August 2009*. Retrieved October 22, 2009, from AV Comparatives Web site:

http://www.av-comparatives.org/images/stories/test/ondret/avc_report23.pdf

AV Comparatives. *Proactive/retrospective test, May 2009*. Retrieved December 6, 2009, from AV Comparatives Web site:

http://www.av-comparatives.org/images/stories/test/ondret/avc_report22.pdf

Christodorescu, Mihai & Jha, Somesh. *Testing Malware Detectors*. Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'04): July 11-14, 2004

CigiCigi Online (2010). Turkojan. Retrieved February 12, 2010, from Turkojan Web site:

<http://www.turkojan.com/eng/>

Hispasec Sistemas (2010). VirusTotal. Retrieved February 12, 2010, from VirusTotal

Web site: <http://www.virustotal.com/>

Merriam-Webster Incorporated (2010). Dictionary and Thesaurus. Retrieved January 29, 2010, from Merriam-Webster Web site:

<http://www.merriam-webster.com/dictionary/steganography>

Microsoft (2010). An overview of unsafe file types in Microsoft products. Retrieved January 2, 2010, from Microsoft Web site:

<http://support.microsoft.com/kb/925330/en-us>

MacManus, G., Mason, J., Monroe, F. & Small, S. *English Shellcode*. Proceedings of the ACM Computer and Communications Security Conference (CCS 2009): November 9-13, 2009

Oreans Technologies (2010). Themia – Advanced Windows Software Protection System. Retrieved January 16, 2010, from Oreans Technologies Web site:
<http://www.oreans.com/themida.php>

Panda Security (2008). Annual Report Pandalabs 2008. Retrieved July 29, 2009, from Panda Security Web site:
http://pandalabs.pandasecurity.com/blogs/images/PandaLabs/2008/12/31/Annual_Report_Pandalabs_2008_ENG.pdf

Richardson, Robert (2008). CSI Computer Crime & Security Survey. Retrieved April 18, 2009, from Computer Security Institute Web site:
<http://i.cmpnet.com/v2.gocsi.com/pdf/CSIsurvey2008.pdf>

Skoudis, Ed. *Security 504: Hacker Techniques, Exploits & Incident Handling*. SANS Institute: 2006

Skoudis, Ed. *Security 560: Network Penetration Testing & Ethical Hacking*. SANS Institute: 2009

Song, Y., Locasto, M.E., Stavrou, A., Keromytis, A.D., Stolfo, S.J. *On the Infeasibility of Modeling Polymorphic Shellcode*. Proceedings of the ACM Computer and Communications Security Conference (CCS 2007): October 29-November 2, 2007

University of Mannheim (2010). CWSandbox – Automated Malware Analysis. Retrieved February 11, 2010, from CWSandbox Web site: <http://mwanalysis.org/>

Zeltser, Lenny. *Security 601: Reverse-Engineering Malware: The Essentials of Malware Analysis*. SANS Institute: 2009

© 2010 SANS Institute, Author retains full rights.