Matthias Deeg, Sebastian Nerz, Daniel Sauder

Outsmarted – Why Malware Works in face of Antivirus Software

For many years, different types of malware rank among the biggest IT security threats both in the business and the private domain. In order to protect oneself from the dangers of malware, numerous software manufacturers offer IT security products like antivirus and endpoint protection software. But these products alone offer no sufficient protection from malware that knows some tricks, as the results of our recent research with the topic antivirus evasion show.

In the recent past, there were several computer-based attacks against IT networks that became public and raised a lot of media attention. Especially the attacks against the New York Times [1] and the Washington Post [2] at the beginning of 2013 had a world-wide media coverage and also heated the debate about such *cyber* threats with manufacturers of IT security products like antivirus and endpoint protection software. In both mentioned cases, attackers were able to install malware on computer systems of employees in order to literally spy on the affected companies – and this probably undetected for several months.
Once more, incidences like these have pointed out that in spite of the use of IT security products like antivirus software or host intrusion detection/prevention software (HIDS/HIPS) such attacks cannot be entirely prevented. This kind of threat illustrates that enterprises and also government agencies require a master plan with a working information security management and security awareness of all employees.

This paper discusses how developers of malware like trojan horses (in short trojans), viruses, and worms proceed to hide their malicious intentions from antivirus software. Thereby, current results of our recent research are presented and recommendations are given for dealing with threats and security risks caused by malware.

How Antivirus Software Works
Current antivirus software, no matter if a standalone software product or a component of a software suite (host intrusion detection/prevention software, endpoint protection software, etc.), uses different methods to detect known and unknown threats by means of malware.
In general, these methods used for protecting computer systems from unwanted, malicious software can be assigned to the following two strategies:

1. Blacklisting
2. Whitelisting

In the context of antivirus software, the two terms blacklisting and whitelisting simply mean that the execution of a program is either explicitly forbidden (being on a *black list*) or explicitly allowed (being on a *white list*). Thus, by following the blacklisting approach antivirus software will prevent the execution of programs that are

Exploiting
Antivirus software does not offer any protection against attacks of vulnerable network services, for example, an outdated web server. Because in such attacks, malicious code, so-called shellcode, is directly loaded into the main memory of the affected system and executed there, for example by exploiting a buffer overflow vulnerability. Thus, there is no file containing the malicious code within the file system of the target system that could be found by the usual malware detection mechanisms of antivirus software.

not found on a *black list*. In contrast, using the whitelisting approach antivirus software will only allow the execution of programs found on a *white list*. These two strategies help to pursue the goal allowing only desired, expected, and benign behavior and preventing unwanted, unexpected, and malicious behavior.
The adminstration of black and white lists can either be the responsibility of the antivirus software, which is the default case for the blacklisting strategy, or it can be the responsibility of the end users or administrators, which is primarily the case for the whitelisting strategy.
The majority of used antivirus software only follows the blacklisting strategy and thus tries to detect whether software is malicious and should therefore not be executed.
For malware detection there are generally the following two methods which are described in more detail in the following sections:

1. Signature-based
2. Behavior-based

Signature-Based Malware Detection
By using signature-based malware detection, antivirus software is searching for known patterns in the form of byte sequences (also called signatures) in files that allow to identify specific malware. A disadvantage of this methodology is that only known patterns can be searched for. These patterns are created by the manufacturers of antivirus software based on analyses of malicious software and are added to their signature database. Hence, with this method it is only possible to detect malware for which at least one corresponding signature exists.
The manner in which manufacturers of antivirus software create signatures for malware and search for the specific patterns of course influences the error rate of signature-based malware detection (error type 1 [false positive], error type 2 [false negative]).

Malware authors and users exploit this situation by developing and modifying malware in such a way that it does not contain any known signatures that are sufficient for classifying it as malware. Thus, this kind of malware cannot be detected by solely signature-based malware detection mechanisms. For bypassing some signature-based malware detections it is sufficient to compile the malware with different compiler settings or with a different compiler (malware source code is available), or to compress and/or encrypt the executable file (malware source code is not available) using so-called EXE packers or EXE cryptors. In this way, the original functionality of known malware can be preserved but its form has changed.
Thus, polymorphism is a big challenge for signature-based detection mechanisms and is used by malware for a long time.

Behavior-Based Malware Detection
Behavior-based malware detection tries to determine the behavior of software and to classify it according to defined criteria as benign or malicious. For this, usually rule-based techniques are used in combination with a scoring system and specified thresholds for calculated scores. This detection method, also known as heuristic procedure, is capable of detecting unknown malware due to its behavior. The used scoring system and threshold values have a major influence on the error rate of this generic method (error type I [false positive], error type II [false negative]). The analysis of a software program can be performed only statically at which the code of the executable file is analyzed in a so-called static code analysis and corresponding software behavior is determined by means of the presence of specific fea-

tures. However, a disadvantage of this approach is that only program code can be analyzed that is directly accessible to the antivirus software. Thus, program code that is only available during runtime respectively detectable as code, for example due to compression, encryption or self-modifying code, is not considered for the static code analysis.

For this reason, most of the antivirus software uses a so-called sandbox environment besides a static code analysis. In the context of antivirus software, a sandbox means a secure and controllable execution environment in which the behavior of an executable file can be analyzed during runtime and be classified as benign or malicious due to defined criteria. In this way, the effectiveness of some malware obfuscation methods, for instance based on compression or encryption, can be reduced. Furthermore, another goal of antivirus sandbox environments is to hide their presence and not to be detectable by malware. This prevents malware from adjusting its behavior according to the discovered execution environment and thus malicious code is not executed within a sandbox. In general, the emulation of an execution environment is a very tough problem and far from being perfectly implemented considering antivirus software sandboxes. Therefore, there are different ways to detect sandbox environments and to manipulate behavior-based analyses by adjusting the software behavior accordingly.
Moreover, the classification of programs by antivirus software into the two categories benign and malicious is also subject to time constraints. Because it is not acceptable to an end user if the analysis of a program lasts a longer period of time and he or she is thus prevented from carrying out her work, the manufacturers of antivirus software have to consider the needs of users and should accept compromises for all their malware detection methods, no matter whether signature- or behavior-based.
Behavior-based malware detection as it is implemented in current antivirus software using different methods has several weaknesses that can be exploited by malware developers.

Research Project: Antivirus Evasion

In the course of our research project with the topic antivirus evasion, the authors have tested different antivirus evasion techniques with several well-known and widely used antivirus software products that follow the blacklisting strategy.
The applied techniques are not new, they have been used by malware for years and exploit the abovementioned weaknesses in signature- and behavior-based detection methods of antivirus software.

Characteristics of Antivirus Software: Updates and Scan Engines

During the conducted antivirus evasion tests of this research project and during penetration tests, the SySS GmbH noticed on several occasions that the behavior of scan engines of tested antivirus software products varied in different environments. For example, if a malicious file is not detected by the locally installed antivirus software, nevertheless, it can be rated as malicious by the assumed identical scan engine when uploaded to VirusTotal. The reason for this different behavior one should be aware of is that antivirus companies often parametrize their scan engines specifically for VirusTotal (stronger heuristics, cloud interaction, inclusion of beta signatures, etc.) [3].
Furthermore, it has to be considered that besides false-positive results there are also false-negative results now and again, for example in the case of known malware that is not correctly detected although it has been known for several months or even years.
In general, it is recommended to have short update intervals for antivirus signature databases as already a few hours can make a difference regarding malware that spreads via e-mail.

Methodology

The SySS GmbH developed two software tools named *Avet* and *ShCoLo* that could be used to create executable files containing malware for Windows operating systems. Both software tools supported different antivirus evasion techniques

Listing 1: Generating and encoding a Meterpreter shellcode with msfpayload and msfencode

```
$ msfpayload windows/meterpreter/reverse_https LHOST=192.168.23.1 LPORT=443 R |
msfencode -e x86/shikata_ga_nai -t raw > meterpreter_reverse_https.bin
[*] x86/shikata_ga_nai succeeded with size 377 (iteration=1)
```

exploiting weaknesses in signature- and behavior-based detection methods. Using these software tools, it should be possible to successfully execute malware in the face of antivirus software on a target system.
As malware, a well-known *Meterpreter* shellcode (*windows/meterpreter/reverse_https*) of the free open source *Metasploit Framework* [4] was used that was created and encoded with the two *Metasploit* tools *msfpayload* and *msfencode* (Listing 1).

Remote access to computer systems using a *Meterpreter* shell is very popular among penetration testers as well as attackers who operate without an official permission. Therefore, such a *Meterpreter* shell is a desirable objective and an interesting test vector for our research. The *Meterpreter* shell supports functions like extracting sensitive password information, key logging, taking screenshots, and of course remote shell access to an affected target system. The chosen *Meterpreter* shellcode *windows/meterpreter/reverse_https* also uses a secure, encrypted communication channel (HTTPS connection) between the target system and the system of the attacker. This connection is established from the target system to the attacker‘s system and thus is a so-called reverse shell or connect back shell that works with a high probability even if firewall or web proxy systems are in use, provided that network access to the attacker‘s system is given.
In the course of our research, the TCP port 443, that is usually used for HTTPS connections, was selected as the communication endpoint on the attacker‘s system.

The two software tools *Avet* and *ShCoLo* are so-called shellcode loaders with extended functionality concerning antivirus evasion techniques. During our tests, all executable files were created with the developed software tool *ShCoLo* (Listing 2). The used *ShCoLo* software version supported the following antivirus evasion techniques that were used in different combinations:

- Polymorphism
- Encryption
- Sandbox detection
- Process injection

With regard to the German Criminal Code (§ 202c StGB), technical details of all the used antivirus evasion techniques and their implementation are not further discussed in this paper.
The created executable files containing malicious code were copied to the respective target systems with installed antivirus software and then executed. The result of each test case was measured and documented according to the following three possible outcomes:

- Shellcode was not detected and executed successfully (green)
- Shellcode was detected and not executed (red)
- Shellcode was not detected but not executed successfully (blue)

Results

The results of our antivirus evasion tests with twelve antivirus software products are shown in table 1.

Antivirus Evasion in IT Forensics and Incident Response

The described strategies for malware detection also affect the retrospective forensic analysis of infected computer systems. Occasionally, detecting malware is cumbersome. But sometimes malware can be identified by simple persistence vectors because it somehow wants to be restarted after a system reboot. A straightforward way for accomplishing this, for instance, is to use one of the many autorun mechanisms of the Windows operating system. Thus, by analyzing automati-

cally started software, many types of malware can simply be found without much effort.

However, this task becomes more difficult when it comes to malware that hides itself in a better way. For example, there is malware that modifies existing executable files like EXEs or DLLs so that it is also executed when the manipulated software is started. If such a modification affects a device driver or another operating system component, this can be sufficient for achieving an automatic restart of the malware after a system reboot. Modifications of frequently used software like web browsers, e-mail software or office applications can also be promising
to accomplish this task. Such modifications are far more difficult to identify by means of a forensic analysis. If the point in time of a malware infection is known, for example, changes of the file system can be analyzed. However, malware developers are also active and creative in regard to this fact and they use different techniques in order to tamper with timestamps, for instance, concerning file systems.

For this reason, methods for rapidly detecting malware were established that use techniques which are conceptually similar to black- and whitelisting. By using databases of known malware and software tools for automatically scanning files for malicious code with scan engines of different antivirus software products, infections with known malware can be identified. Heuristic signature-based methods are used for finding known EXE packers, known strings, specific anomalies in program code like NOP slides (sequence of NOP opcodes [no operation]), or for finding seemingly random and thus potentially encrypted code sections.

These techniques are also used in a modified form in antivirus software, but regarding forensic analyses, they are allowed to produce many more false-positive results as these are (hopefully) checked afterwards. Nevertheless, these described techniques will only have a little chance of success if malware authors have analyzed them. Therefore, another method for detecting malware is used that is based on databases with known files. For instance, a Windows system mainly contains files that do not differ from files of other Windows installations. Dependent on the software version, patch level, and installed software, there are different numbers of files with the same characteristics. These file characteristics can be used to build databases that serve the purpose of finding all those files that deviate from those characteristics. However, malware infections in uncommon software, macro viruses in office documents or attachments to e-mails cannot be identified in this way. But as mentioned above, malware often wants to persist itself on the target system and most of the malware tries to accomplish this by modifying popular software via autorun mechanisms. And precisely these methods can be detected in the described way with a little bit of luck.
Specifically adjusted or exclusively developed malware for attacking individual systems – so-called targeted malware – will elude this kind of

Metasploit Shellcode Loader
With the use of the two software tools *msfpayload* and *msfencode* of the *Metasploit* framework, executable files with chosen payloads (shellcode) can be created and encoded for different target platforms. Executable files, however, generated in this way are identified as malware by the majority of antivirus software products even if they do not contain any malicious code. This fact can be demonstrated very easily with an executable file for Windows created by *msfpayload* (Listing 3) that is scanned afterwards using the online service VirusTotal [4] (see Figure 1).
In many cases, it is possible to achieve that less antivirus software detects the *malicious* file as malware by simply using a self-made shellcode loader.
A further encoding of the malware using the software tool *msfencode* does not guarantee a lower detection rate as the publicly known encoders are either detected due to signature- or behavior based methods. In order to counteract this fact and to decrease the detection rate, one can build her/his own encoder.

detection. Anyhow, a method for identifying it successfully is a detailed analysis of the file system, log files, and file system changes. Fortunately, such cases are not the norm as they are very time-consuming and labor-intensive.

The last and also frequently used method for finding malware during forensic analyses are behavior-based techniques. Thereby, running processes can be identified, log files can be analyzed for anomalies, or the network can be monitored. In doing so, a good IDS will not only search for known command & control servers (C2 servers), but it will also report other anomalies. For instance, if a server suddenly accesses foreign systems, this could and should cause distrust. Periodically visited web sites, network data traffic on unknown ports, incoming network connections, encrypted connections, or other deviations from the rule could indicate a malware infection.

As stated before, deviations from the rule are suspicious. But what if there is no rule? A network monitoring system that was installed after a malware infection may identify it in some cases as the rule and not as anomaly. Exactly the same problem exists, as mentioned above, by analyzing file systems: What is legitimate and what is not? How can we decide retrospectively what software has been modified? And how can we accomplish this task efficiently and within an acceptable time frame?

IT administrators and other personnel responsible for information security can support IT forensic experts and incident responders significantly. For instance, a network monitoring system that was installed at the right time can be used for baselining purposes. Thus, the rule can be identified by means of network data traffic, number and type of network connections, and so on. Furthermore, managed lists of installed software, frequent comparisons of the current and the target state, and maybe even databases with hash values or other characteristics suitable for identification purposes of files, centralized logging and network monitoring are a first step.

To make it clear: Logging and the identification of the rule may later help to prevent time-consuming analyses or to raise initial suspicion in the first place.

Conclusion

In the course of the research project with the topic antivirus evasion, the SySS GmbH could demonstrate that the malware detection mechanisms of current antivirus software can be bypassed and that malware can be executed in this way. Thereby, the applied antivirus evasion techniques are not new, they have been used by malware for years and exploit described weaknesses in signature- and behavior-based detection methods of antivirus software. The majority of the used antivirus techniques

Listing 2: Creating a Windows executable file with ShCoLo

```
$ python shcolomaker.py -f win32 -e meterpreter_reverse_https.bin
     _/_/_/  _/           _/_/_/             _/
  _/         _/_/_/    _/           _/_/    _/          _/_/
   _/_/    _/    _/  _/         _/    _/  _/        _/    _/
      _/  _/    _/  _/         _/    _/  _/        _/    _/
_/_/_/    _/    _/   _/_/_/    _/_/    _/_/_/_/    _/_/
Shellcode Loader Maker v0.7 by Matthias Deeg <matthias.deeg@syss.de> - SySS GmbH (c) 2013, 2014
[*] Process shellcode (377 bytes)
[*] Encrypt shellcode
[*] Generate source code
[*] Generate Makefile
[*] Build executable ...
make: Entering directory ,/home/matt/playground/antivirus-evasion/shcolo/build‘
nasm -fwin32 shcolo.asm
wine ./tools/link.exe /SUBSYSTEM:WINDOWS /MACHINE:X86 /ENTRY:start /OUT:shcolo.exe shcolo.obj ./lib/
kernel32.lib
fixme:heap:HeapSetInformation (nil) 1 (nil) 0
Microsoft (R) Incremental Linker Version 9.00.30729.207
Copyright (C) Microsoft Corporation.  All rights reserved.

fixme:msvcrt:__clean_type_info_names_internal (0x10044484) stub
make: Leaving directory ,/home/matt/playground/antivirus-evasion/shcolo/build‘
[*] Successfully built the shellcode loader ,./build/shcolo.exe‘
```

Listing 3: Generating an msfencoded executable file without a valid payload (shellcode)

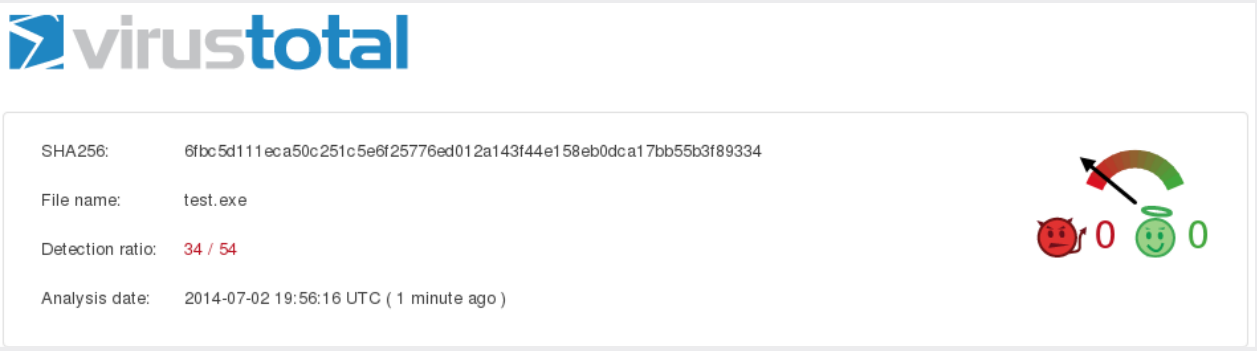
```
$ echo test | msfencode -e genric/none -t exe > test.exe
[*] generic/none succeeded with size 5 (iteration=1)
```

is rather simple and can be leveraged by less skilled attackers and their malware. Furthermore, for many years now there have been numerous antivirus evasion tools or frameworks freely available on the Internet that also support more advanced and more complex antivirus evasion methods that can be used by anyone without expert knowledge. A popular example for such an antivirus evasion framework is Veil [5].

In IT networks, antivirus software is the last line of defense against malware, especially on end user client systems. Because of the demonstrated weaknesses of antivirus software, this security control should not be the only countermeasure against malware threats. A defense-in-depth strategy should be followed as part of an enterprise-wide IT security management system. Thereby, identified risks are not only avoided or mitigated with a single security control but with several security controls that ideally fall into one of the three different categories preventive, detective, and corrective.

In the opinion of the SySS GmbH, the following security measures have been proven effective as part of a defense-in-depth strategy in order to increase the security level of IT networks, particularly concerning malware protection, and to reduce the impact of security incidents:

Figure 1: VirusTotal scan results for a Windows executable file created with msfpayload



- Security awareness training of employees
- Implementation of a working patch management
- Use of current antivirus software with regular updates
- Implementation of the principle of least privilege (least-privileged user accounts that only have the privileges actually needed for fulfilling required tasks)
- Antivirus detection at different locations within the IT network (mail server, file server, proxy server, client systems, etc.)
- Conduct of frequent IT security assessments
- Incident readiness (business continuity and disaster recovery management)
- Baselining of the IT infrastructure

According to estimations of the SySS GmbH, the situation for IT networks regarding malware threats will not ease in the near future – on the contrary – it will get worse. Reasons for this are steadily increasing professionalism and increasing resources of malware developers – a trend that has been observable for several years now. Besides targeted attacks that are usually performed by highly skilled attackers with the use of specifically adjusted or exclusively developed malware, mass computer-based attacks with malware generated by widely-used malware construction or exploit kits are a growing threat for enterprises and government agencies.

Table 1: Results of antivirus evasion tests

Product Name	Version	Date of Signature File	Operating System of the Target System
avast! Endpoint Protection	8.0.1603	2014/07/03	Windows 7 SP 1 (64 Bit)
AVG AntiVirus Free	2014.0.4714	2014/06/30	Windows 7 SP 1 (64 Bit)
Avira Professional Security	14.0.5.450	2014/07/02	Windows 7 SP 1 (64 Bit)
ESET NOD32 Antivirus	7.0.317.4	2014/07/02	Windows 7 SP 1 (64 Bit)
Kaspersky Anti-Virus	14.0.0.4651(g)	2014/07/02	Windows 7 SP 1 (64 Bit)
McAfee VirusScan Enterprise	8.8.5400.1158	2014/07/02	Windows 7 SP 1 (64 Bit)
Microsoft Security Essentials	1.177.1250.0	2014/06/30	Windows 7 SP 1 (64 Bit)
Panda Antivirus Pro 2014	13.01.01	2014/06/30	Windows 7 SP 1 (64 Bit)
Panda Cloud Antivirus	3.0.1	2014/07/03	Windows 7 SP 1 (64 Bit)
Sophos Anti-Virus	10.3.7.527	2014/06/19	Windows 7 SP 1 (64 Bit)
Symantec Endpoint Protection	12.1.4013.4013	2014/07/02	Windows 7 SP 1 (64 Bit)
Trend Micro Titanium Antivirus+	7.0.1255	2014/07/01	Windows 7 SP 1 (64 Bit)

A change of strategy away from blacklisting to whitelisting has been observable for quite some time. In the opinion of the SySS GmbH, this is a step in the right direction. But there is also a need for further clarification, especially in finding the right balance between usability and security and considering cost-benefit ratios, particularly in regard to possibly increased administrative expenses.

References

[1] http://www.heise.de/security/meldung/New-York-Times-Hack-Symantec-wehrt-sich-1795358.html
[2] http://www.heise.de/security/meldung/Auch-Washington-Post-gehackt-1796535.html
[3] https://www.virustotal.com/en/faq/
[4] http://www.rapid7.com/products/metasploit/
[5] https://www.veil-framework.com/