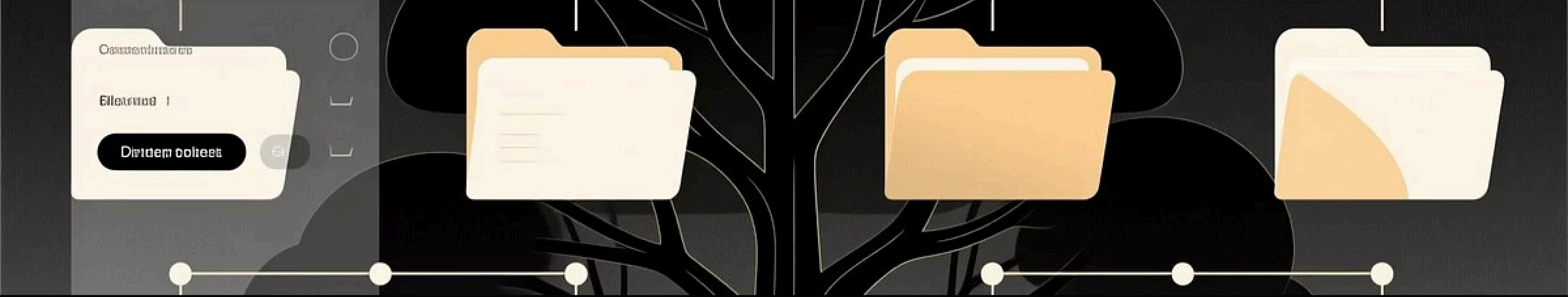
CTF Crash Course: Part 3

Advanced Techniques for Capture the Flag

Web Exploitation Fundamentals

Web applications are battlegrounds where attackers probe for weaknesses. In CTFs, web exploitation challenges test your ability to identify and exploit vulnerable input handling, authentication flaws, and improper access controls.

Why it matters: Web vulnerabilities power real-world breaches. Understanding these techniques prepares you for pentesting and secure development roles.



Common Attack Surfaces

/robots.txt

Directory listing file that often reveals hidden or restricted paths administrators intended to hide from search engines.

/admin Paths

Admin panels frequently lack proper access controls. Discovering these endpoints can grant unauthorized access to sensitive functions.

Hidden Routes

Developers sometimes leave test endpoints, debug panels, or backup files exposed. Directory fuzzing uncovers these quickly.

Tools & Methods for Web Exploitation

01

Inspect Element

Browser DevTools reveal HTML, JavaScript, cookies, and network requests. Start here to understand client-side logic.

02

Burp Suite

Industry-standard proxy for intercepting, modifying, and replaying HTTP requests. Essential for manual testing and payload crafting.

03

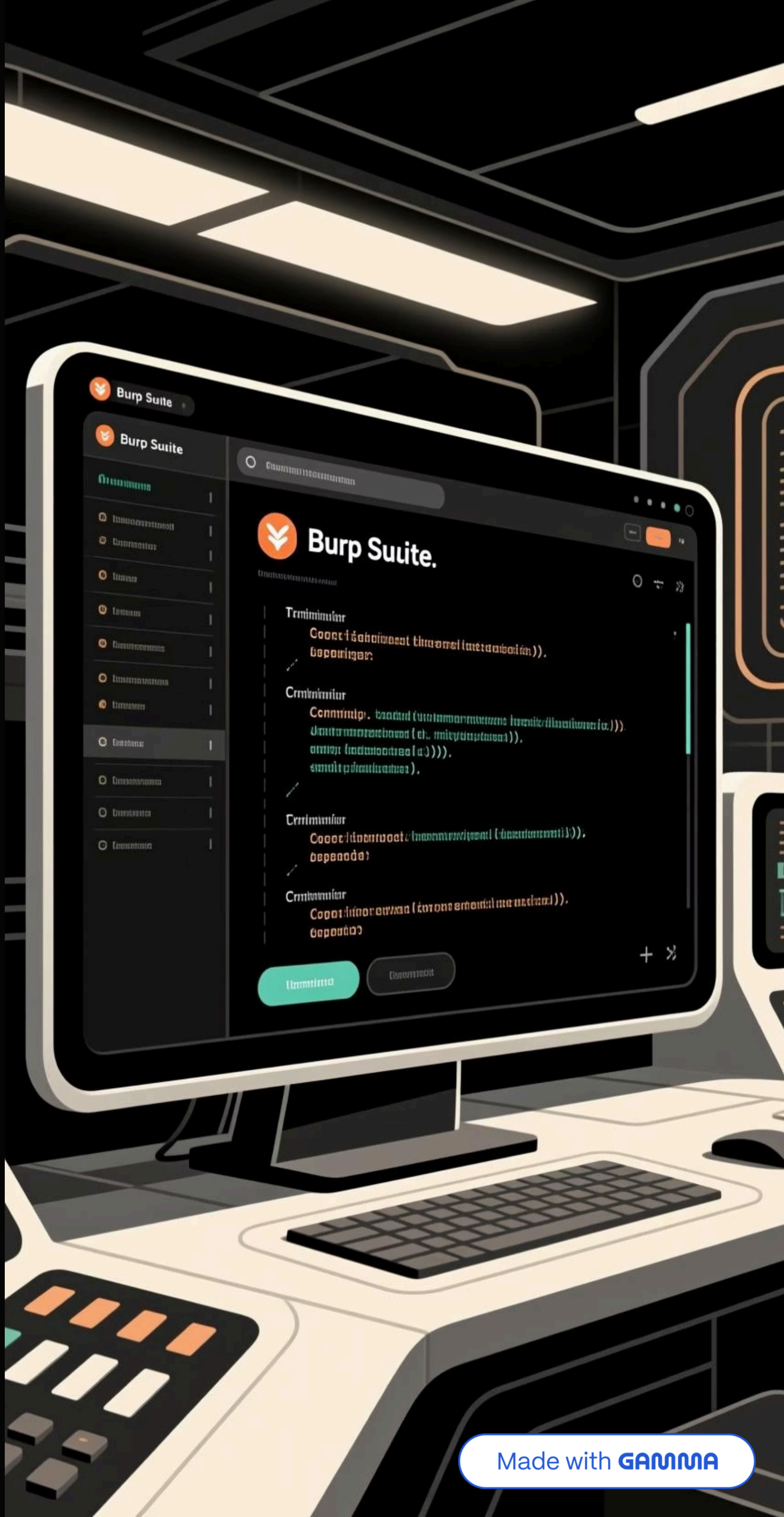
Gobuster

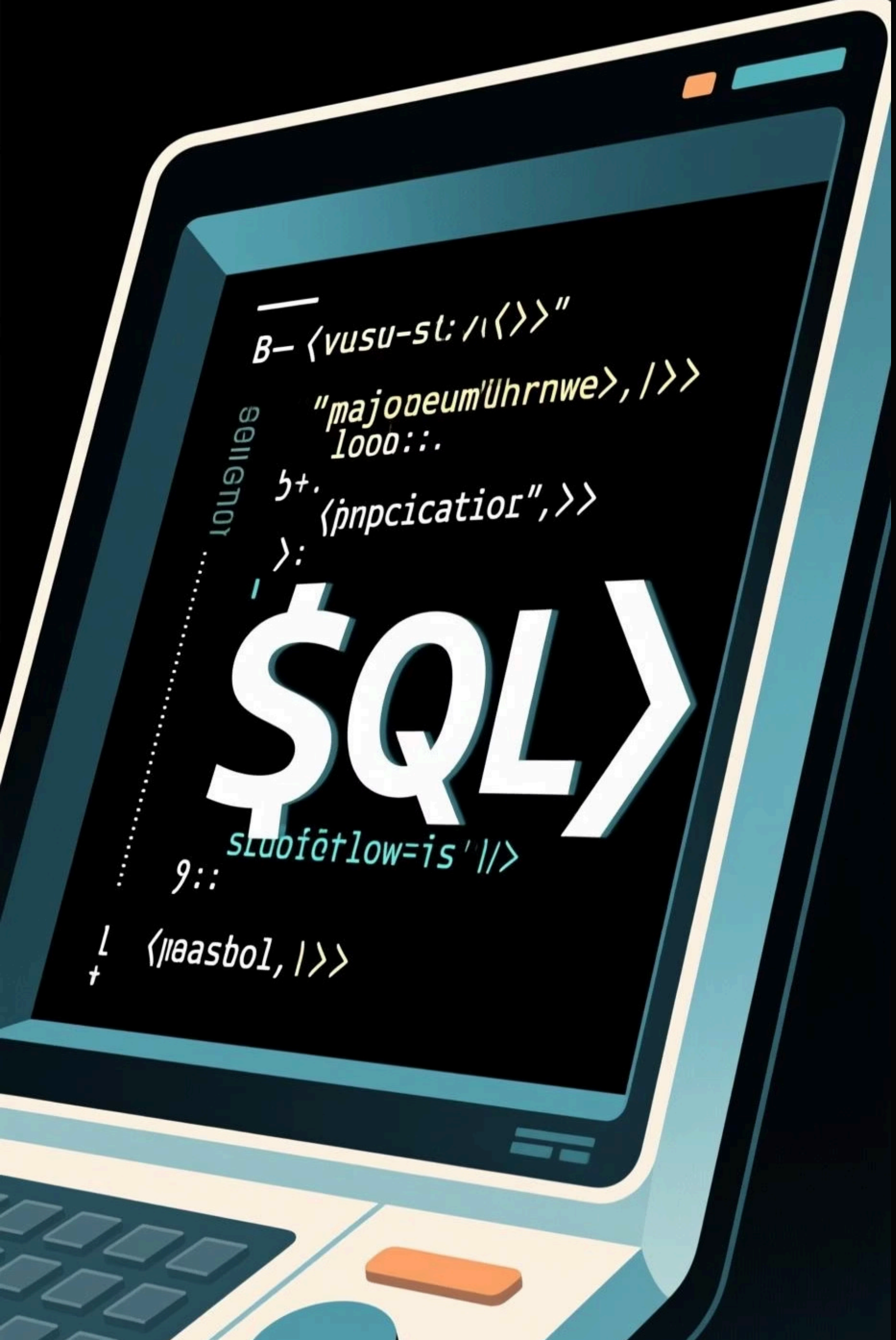
Fast directory and DNS fuzzing tool. Discovers hidden endpoints by brute-forcing common paths and file names.

04

SQL Injection Basics

Inserting malicious SQL into input fields bypasses authentication or leaks data. Always test for quote escaping and comment syntax.





Web Exploitation Challenge Example

Scenario: A login page with weak input validation. Your goal: bypass authentication and extract the flag.

Step 1: Reconnaissance

Use Gobuster to discover `/admin`, `/backup.sql`, and `/config.php`. Check robots.txt for hints.

Step 2: Identify Vulnerability

Inspect the login form. Test for SQL injection: enter `' OR '1'='1` in the username field.

Step 3: Extract Flag

Successful injection grants access. Navigate to the admin panel and retrieve the flag from the database or file system.

Reverse Engineering & Binary Analysis

Reverse engineering transforms compiled code back into human-readable instructions. In CTFs, you analyse binaries to uncover hardcoded flags, bypass protections, or find logic flaws.

Core Skills: Disassembly, control flow analysis, and patch engineering are fundamental to incident response and malware analysis roles.

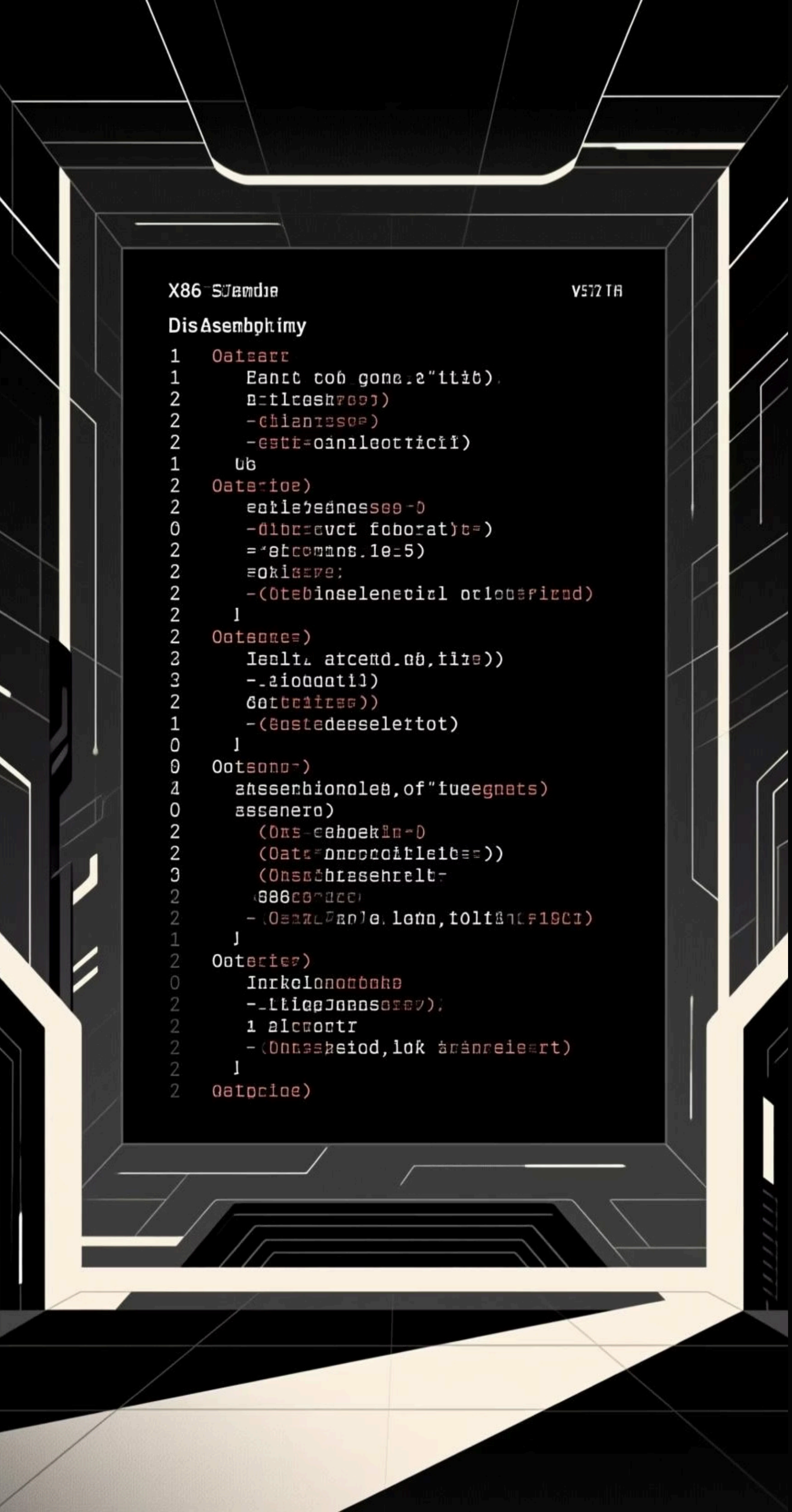
Reverse Engineering Tools & Workflow

Essential Tools

- **Ghidra:** Open-source decompiler with excellent disassembly and decompilation.
- **Radare2:** Lightweight, scriptable framework for forensic analysis.
- **IDA Pro:** Industry-standard with interactive debugging and plugin support.

Analysis Process

- Load binary and identify entry point.
- Disassemble to understand instruction flow.
- Identify functions and control structures.
- Look for strings, hardcoded values, or suspicious patterns.



Binary Challenge: Flag Extraction

Example: Given an ELF binary, extract the hidden flag by reversing the authentication function.

1. Load & Inspect

Run `file challenge` to identify the architecture. Load into Ghidra and examine the symbol table for function names.

2. Locate Check Function

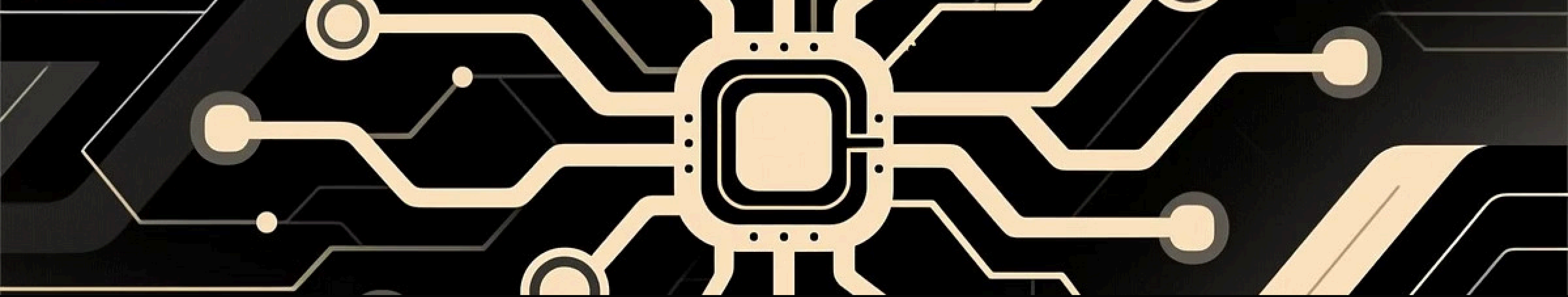
Search for string references (e.g., "flag", "correct", "invalid"). These often appear near authentication logic.

3. Disassemble & Analyze

View the function's assembly. Compare register values or follow conditional jumps to understand the validation algorithm.

4. Extract or Bypass

Either patch the binary to skip validation or brute-force the password using the logic you discovered. Run and capture the flag.



LLM Jailbreaking: The Modern Frontier

As AI models proliferate, CTF challenges now include LLM exploitation. These tasks test your ability to manipulate prompt context, bypass safety guardrails, and understand token processing—a skill increasingly relevant in AI security roles.

Prompt Injection

Embed hidden instructions in user input to override system prompts and reveal sensitive data or alter model behavior.

Token Manipulation

Exploit how models tokenize input. Specially crafted tokens can bypass content filters or trigger unintended outputs.

Ethical Implications

These techniques mirror real-world AI misuse. Understanding defences helps you build safer systems and contributes to responsible AI development.

Data Exfiltration Successful. Valuable Skills Acquired.

You now command web exploitation, reverse engineering, and emerging AI security techniques. These skills transcend CTFs: they form the foundation for careers as penetration testers, threat hunters, malware analysts, and AI security researchers.

Next Steps

Compete in live CTF events. Build a portfolio. Network with the security community.

Career Path

CTF success opens doors to incident response, threat intelligence, and security research roles across Fortune 500 companies and government agencies.

Keep Learning

The threat landscape evolves constantly. Stay curious, participate in hackathons, and contribute to open-source security tools.