CTF Report

Full Name: Isha Sangpal
Program: HCS - Penetration Testing 1-Month Internship
Date: 08-03-2025

Category: Web (Lock Web)

Description: The Web category in CTF challenges involves exploiting vulnerabilities in web applications, such as authentication flaws, API weaknesses, and brute-force attacks. It tests skills in analyzing HTTP requests, bypassing security mechanisms, and extracting sensitive data through methods like enumeration, injection attacks, and automation.

Challenge Overview: The challenge presents a web-based PIN authentication system where a 4-digit PIN is required to unlock a secret. The goal is to analyze the authentication mechanism, identify a weakness, and use an automated brute-force attack to retrieve the correct PIN and capture the flag.

Steps for Finding the Flag (Detailed Explanation)

1. Initial Reconnaissance

- The web application was explored to understand its structure and authentication mechanism.
- Using the **Developer Tools (F12 → Network Tab)**, the PIN submission request was identified, revealing an API call to `/check-pin?pin=XXXX`.
- JavaScript source code was reviewed to confirm that no additional obfuscation or client-side security measures were in place.
- The request behavior was analyzed for security measures such as rate limiting, CAPTCHA, or account lockout, none of which were present.
- It was determined that the PIN verification relied solely on the backend's response, making it vulnerable to brute-force attacks.

---

2. Brute-Force Attack on PIN Authentication

- A sample PIN entry request was captured using **Burp Suite Professional's Proxy** and forwarded to **Intruder** for automated attack configuration.
- The PIN parameter within the request URL was marked as a payload position to allow dynamic substitution.

- **Burp Suite Intruder** was configured to use the **"Numbers" payload type**, generating sequential PINs from `0000` to `9999` with a step size of `1`.
- The attack was executed, sending thousands of requests to systematically test each possible PIN.

---

3. Response Analysis

- Once the brute-force attack was completed, the results were analyzed in **Burp Suite Professional's Intruder results tab**.
- A pattern in the responses was observed:
  - **Most incorrect PIN attempts returned responses of uniform length (~881 bytes).**
  - **One response was significantly different in length (~289 bytes), indicating a successful authentication attempt.**
  - Sorting the results by **response length** helped quickly identify the anomaly.
- Further examination of the response confirmed that this PIN was the correct one, as it led to a different outcome in the web application.

---

4. Successful Exploitation

- The identified PIN was manually entered into the **PIN input field** on the web application.
- Upon submission, the application validated the PIN and provided access, confirming successful authentication.
- A pop-up message appeared, displaying the retrieved flag.
- To verify consistency, the same request was replayed in **Burp Suite's Repeater tool**, and it consistently returned the success response, confirming the exploit's validity.

---

5. Flag Retrieval

- The flag was displayed as a plaintext message within the authentication success dialog.
- The flag was carefully documented and cross-verified in the **Burp Suite Response Inspector** to ensure accuracy.
- The retrieved flag was then submitted to the CTF platform to complete the challenge.

Flag: flag{V13w_r0b0t5.txt_c4n_b3_u53ful!!!}

Conclusion:

This challenge demonstrated the effectiveness of brute-force attacks against weak authentication mechanisms when proper security controls such as rate limiting or account lockouts are not implemented. **Burp Suite Professional** played a crucial role in automating and analyzing the attack, allowing for rapid identification of the correct PIN. By leveraging response length discrepancies, the correct PIN was extracted efficiently, leading to successful authentication and flag retrieval.

Category: Crypto (Success Recipe)

Description: This challenge falls under the **Esoteric Programming** category, which involves solving problems using obscure or unconventional programming languages. These languages are often intentionally difficult to use, requiring knowledge of syntax, debugging, and interpretation.

Challenge Overview: The challenge presented a **recipe-style script**, which hinted at the **Chef programming language**. Upon execution, it generated **Brainfuck code** that needed to be further decoded. The task was to debug and execute the Chef script, extract the Brainfuck code, and finally interpret it to retrieve the flag.**Steps for Finding the Flag (Detailed Explanation)**

1. Initial Reconnaissance

- Observed the provided **recipe text file** and identified its **structured format**, resembling the **Chef esoteric programming language**.
- Analyzed the challenge description, which hinted at an **unusual or "weird" language**, confirming the need to execute the script in a Chef interpreter.

---

2. Identifying & Fixing Errors in the Chef Script

Used an online Chef interpreter at:
https://esolangpark.vercel.app/ide/chef

Encountered errors due to incorrect syntax, particularly unnecessary instructions like "until crack", "until juice", etc.

Manually fixed the errors, making the script valid for execution.

Successfully executed the Chef script, which produced Brainfuck code as output:

```
++++++++++[>+>+++>+++++++>++++++++++<<<<-
]>>>>+++++++++++++++++++++.<<++++++++++++++++++.>>----.----------------
```

---

3. Decoding the Brainfuck Output

Recognized that the generated text was in Brainfuck, another esoteric programming language.

Used an online Brainfuck interpreter:

https://www.dcode.fr/brainfuck-language
Successfully decoded the Brainfuck script, revealing the output:
y0u_40+_s3rv3d!

---

4. Flag Retrieval

Wrapped the extracted string in the required flag format:
flag{y0u_40+_s3rv3d!}
Successfully submitted the flag, completing the challenge.

---

5. Flag Retrieval

- The flag was displayed as a plaintext message within the authentication success dialog.
- The flag was carefully documented and cross-verified in the **Burp Suite Response Inspector** to ensure accuracy.
- The retrieved flag was then submitted to the CTF platform to complete the challenge.

Flag: flag{y0u_40+_s3rv3d!}

Conclusion:

This challenge tested the ability to recognize and work with **esoteric programming languages**, requiring multi-step decoding:

- **Chef programming language** was used to generate **Brainfuck code**.

- **Brainfuck code was executed** to extract the final message.

- **Debugging, interpretation, and execution were key skills needed to solve this challenge.**

Category: Reverse Engineering (Lost in the Past)

Description: The OSINT category involves gathering information from publicly available sources to find hidden data or clues. These challenges test the ability to extract and analyze data from various platforms to identify useful leads.

Challenge Overview: The challenge presented an encrypted message hidden in an App Inventor project file. The objective was to extract the hidden text by analyzing the project files and decoding the ROT47 encryption used in the app's components.

Steps for Finding the Flag (Detailed Explanation)

1. Initial Reconnaissance

The challenge file was inspected, and it was identified as an MIT App Inventor project file (.aia). The structure was examined, and the project.properties, Scrum.bky, and Screen1.scm files were extracted for further analysis.

---

2. Component Analysis:

After extracting the project files, attention was given to the **TextBox1** component, which contained an encoded message using ROT47. This was identified in the Scrum.bky file.

---

3. Encryption Identification:

The encrypted message in the **TextBox1** field was identified as ROT47 encoded text:
7=28LE__0>F490C6GbCD`?8N

---

4. Decryption:

flag{t00_much_rev3rs1ng}

---

5. Flag Retrieval

The decoded flag, flag{t00_much_rev3rs1ng}, was retrieved and submitted.

Flag: flag{t00_much_rev3rs1ng}

Category: Web (The World)

Description: The OSINT category challenges participants to gather and analyze publicly available information to uncover hidden data or clues. This particular challenge tested the ability to discover concealed elements on a webpage and decode an encoded message to retrieve the flag.

Challenge Overview: This challenge involved investigating a seemingly simple static webpage. By digging deeper, participants uncovered a hidden file containing a Base64-encoded string. The objective was to decode the string and extract the hidden flag.

Steps for Finding the Flag (Detailed Explanation)

1. Initial Reconnaissance

The webpage at the provided URL appeared to be a basic static website. No obvious clues or flags were present directly on the page.
The mission was to dig deeper, suggesting that there might be hidden elements within the website or its resources.

---

2. Finding the Hidden Path:

Upon further investigation, a hidden file was discovered at: https://the-world-web.hackatronics.com/secret.txt.
This file contained an encoded string:
RkxBR3tZMHVfaGF2M180eHBsMHJlRF90aDNfVzByTGQhfQ==

---

3. Decoding the Hidden Message:

The encoded string appeared to be in Base64 format. To decode it, an online Base64 decoder was used at https://www.base64decode.org/.
The decoded string revealed the flag:

FLAG{Y0u_hav3_4xpl0reD_th3_W0rLd!}

---

4. Flag Retrieval

The decoded flag, flag{Y0u_hav3_4xpl0reD_th3_W0rLd!}, was retrieved and submitted.

Flag: flag{Y0u_hav3_4xpl0reD_th3_W0rLd!}

Conclusion:

The challenge required a methodical approach to explore the website beyond the initial simple appearance. By locating the hidden file, decoding the Base64-encoded message, the flag was successfully uncovered.

Category: OSINT (Time Machine)

Description: The OSINT category focuses on gathering publicly available information to uncover hidden data or clues. This challenge required investigation of a publicly available archive file to extract a secret.

Challenge Overview: The challenge involved finding a hidden file on the Internet Archive. By navigating to the archive and downloading the relevant text file, participants could uncover the hidden flag.

1. Initial Reconnaissance

The first step involved analyzing the publicly available website at https://archive.org/details/secret_202103. This site contained the clue, and by clicking the "TEXT" download option, a new link was revealed that pointed to a text file containing the flag.

---

2. Flag Retrieval:

By opening the downloaded file at https://ia601807.us.archive.org/10/items/secret_202103/secret.txt, the flag was displayed directly in the text: flag{Tr0j3nHunt_t1m3_tr4v3l}.

Flag: flag{Tr0j3nHunt_t1m3_tr4v3l}

Category: Network Forensics (Corrupted)

Description: Forensics challenges involve analyzing files and data to uncover hidden information. This challenge tested the ability to repair and extract data from a corrupted file.

Challenge Overview: The challenge provided a corrupted PNG file. The task was to use a hex editor to manipulate the file's bytes, repair the image, and reveal the hidden flag within.

Steps for Finding the Flag (Detailed Explanation)

1. Initial Reconnaissance

The provided PNG file was corrupted and could not be opened normally. The file's structure was analyzed using a hex editor at https://hexed.it/.

---

2. Byte Modification:

By inspecting the hexadecimal data of the file, certain bytes were identified as potentially corrupted. These bytes were altered to restore the file to a viewable state.

---

3. Flag Retrieval

After modifying the necessary bytes and saving the file, the image was successfully opened, revealing the flag.

Flag: flag{m3ss3d_h3ad3r$}

Conclusion:

The challenge required a methodical approach to explore the website beyond the initial simple appearance. By locating the hidden file, decoding the Base64-encoded message, the flag was successfully uncovered.