# CTF Report

## Full Name: Milind Patel
## Program: HCS - Penetration Testing 1-Month Internship
## Date: 09-03-2025

---

**Category: Web (Lock Web)**

**Description:** A web-based challenge involving a simple numeric lock.

**Challenge Overview:** The challenge presented a numeric keypad interface resembling a basic lock mechanism. The objective was to find the correct passcode to unlock it and retrieve the flag.

**Steps for Finding the Flag:**

1. **Initial Reconnaissance:** Examined the web application and its structure. The interface contained a numeric lock with no immediate clues. Checked for hidden directories or sensitive files.
2. **Directory Enumeration:** Appended /robots.txt to the URL to check for restricted paths. This led to discovering a hidden entry containing the passcode.
3. **Exploitation:** Retrieved the passcode (1928) from the robots.txt file and used it to unlock the numeric lock.
4. **Flag Retrieval:** After unlocking the interface, captured the flag from the response.

**Flag: flag{V13w_r0b0t5.txt_c4n_b3_u53ful!!!}**

**Category: Web (The World)**

**Description:** A web-based challenge that encouraged exploration to uncover hidden resources. The goal was to find a way past the initial welcome page and retrieve the flag.

**Challenge Overview:** The page displayed a simple welcome message with hints about exploring further. By analyzing the page source and checking hidden files, I was able to extract the flag.

**Steps for Finding the Flag:**

1. **Initial Reconnaissance:** Opened the webpage, which contained a welcome message and general instructions. There were no visible input fields or interactive elements to engage with.
2. **Page Source Analysis:** Viewed the page source to check for hidden elements. clicking.
3. **Directory Enumeration:** Tried accessing different files and directories manually. Appended /secret.txt to the URL, which revealed a base64-encoded string.
4. **Exploitation:** The encoded text , Decoded it using a base64 decoder, which converted it into the flag.

   " RkxBR3tZMHVfaGF2M180eHBsMHJlRF90aDNfVzByTGQhfQ== "

5. **Flag Retrieval:** Successfully retrieved and documented the flag for submission. This challenge highlighted the importance of checking hidden files and encoded data.

**Flag:** **FLAG{Y0u_hav3_4xpl0reD_th3_W0rLd!}**

**Category: Network Forensics (Corrupted)**

**Description:** A forensics-based challenge where a corrupted PNG file was provided. The objective was to recover the image and extract the flag hidden within it.

**Challenge Overview:** The given PNG file was corrupted and wouldn't open normally. By analyzing and fixing its header using a hex editor, I successfully restored the image and retrieved the flag.

**Steps for Finding the Flag:**

1. **Initial Reconnaissance:** Tried opening the provided PNG file, but it was unreadable. Suspected file corruption, particularly in the header section.
2. **Hex Editor Investigation:** Opened the file in an online hex editor to examine its structure. Noticed that the first few bytes of the header were incorrect compared to a valid PNG file.
3. **Header Modification:** Replaced the corrupted header with the correct PNG signature (89 50 4E 47 0D 0A 1A 0A). Used a valid PNG file to copy the header structure.
4. **File Restoration:** Saved the modified file and attempted to open it. The PNG image was now visible, displaying text inside the image.
5. **Flag Retrieval:** Read the text in the restored image, which contained the flag. Successfully documented the flag for submission.

**Flag: flag{m3ss3gd_h3ad3r$}**

**Category: Network Forensics (Shadow web)**

**Description:** Analyzing packet capture data to extract hidden information.

**Challenge Overview:** A .pcapng file was provided, and analyzing it in Wireshark revealed a changing character in each HTTPS request. By extracting and decoding these characters using Base64, the flag was obtained.

**Steps for Finding the Flag:**

1. **Packet Inspection:** Opened the .pcapng file in Wireshark to analyze network traffic.
2. **Pattern Identification:** Noted that one character was changing in each HTTPS request.
3. **Character Extraction:** Manually recorded each changing character.
4. **Decoding:** Used Base64 decoding to reveal the flag.
5. **Flag Retrieval:** Successfully extracted and documented the flag.

**Flag: flag{mult1pl3p4rtsc0nfus3s}**

**Category: Reverse Engineering (Lost in the Past)**

**Description:** A reverse engineering challenge where an .aia file was provided. The objective was to analyze its contents and extract the flag.

**Challenge Overview:** The given .aia file was actually a compressed archive. By converting it into a .zip file and inspecting its contents, I found encoded text in a .bky file, which led to the flag after decoding.

**Steps for Finding the Flag:**

1. **File Analysis:** The challenge provided an .aia file, typically associated with MIT App Inventor projects. Suspected it contained hidden information.
2. **File Conversion:** Renamed the .aia file to .zip and extracted its contents. Found multiple files inside the extracted folder.
3. **Identifying Key Files:** Among the extracted files, a .bky file caught my attention. This file is used in block-based programming and could contain hidden text.
4. **Decoding Hidden Data:** Opened the .bky file and found a suspicious base64-encoded string. Decoded it using an online base64 decoder.
5. **Flag Retrieval:** The decoded text revealed the flag, which I successfully documented for submission.

**Flag: flag{t00_much_rev3rs1ng}**

**Category: Reverse Engineering (Decrypt Quest)**

**Description:** A multi-layered decryption challenge that involved base64 encoding, code analysis, and cryptographic puzzles.

**Challenge Overview:** he challenge involved decoding base64 text, analyzing hidden clues in code, and modifying the logic to reveal the flag.

**Steps for Finding the Flag:**

1. **Extracting the ZIP File:** The challenge provided a ZIP file, which I extracted to find a text file inside.
2. **Analyzing the Text File:** The file contained base64-encoded text, Java code, and misleading hints meant to confuse participants.
3. **Decoding Base64:** After decoding the first base64 text, I found another encoded string but still no flag. The Java code also contained a commented-out Google Drive link.
4. **Finding the Brainfuck Code:** The Google Drive document contained Brainfuck-encoded text. I decoded it to reveal a clue about the UNIX epoch year.
5. **Identifying the Epoch Year:** I Googled the UNIX epoch year and found it to be 1970.
6. **Bypassing Input Validation:** Entering 1970 resulted in an invalid input error. To bypass this, I modified the Java code to reveal all possible outputs.
7. **Flag Retrieval:** Among the outputs, I found one containing 1970, which was the flag.

**Flag: flag{hjwilj111970djs}**

**Category: OSINT (Time Machine)**

**Description:** An OSINT challenge that required online investigation. The goal was to track down a hidden file containing confidential information.

**Challenge Overview:** The challenge hinted at a character named "Mr. TrojanHunt" with time-traveling abilities. By conducting a targeted online search, I was able to locate and retrieve the flag.

**Steps for Finding the Flag:**

1. **Initial Reconnaissance:** Read the challenge description, which suggested a connection between "Mr. TrojanHunt" and time travel. The mention of confidential files hinted at an online footprint.

2. **Targeted Search:** Conducted a Google search for "Mr. TrojanHunt time traveller". The first website in the search results seemed relevant, so I visited it.

3. **File Discovery:** Browsed the website and found a file named secret.txt. The presence of this file strongly indicated it contained valuable information.

4. **Exploitation:** Clicked on secret.txt, which revealed the flag inside. No further decryption or analysis was required.

5. **Flag Retrieval:** Successfully documented the flag for submission. This challenge reinforced the importance of OSINT techniques and online search strategies.

**Flag: flag{Tr0j3nHunt_t1m3_tr4v3l}**

**Category: Crypto (Success Recipe)**

**Description:** Deciphering an encoded message hidden within a recipe-style text using an esoteric programming language.

**Challenge Overview:** A .txt file containing a recipe-style text was provided. Upon analyzing the pattern, it was identified as Chef, an esoteric programming language. Using an online tool (Esolang Park), the text was converted into Brainfuck code. After decoding the Brainfuck code, the flag was retrieved.

**Steps for Finding the Flag:**

1. **Text Analysis:** Observed that the given text followed a structured pattern resembling an esoteric language.
2. **Esolang Identification:** Recognized the text as code written in Chef, an esoteric programming language.
3. **Decoding Attempt:** Used the Esolang Park online tool to convert the Chef code into Brainfuck.
4. **Brainfuck Decoding:** Deciphered the Brainfuck code to retrieve the hidden message.
5. **Flag Retrieval:** Successfully extracted and documented the flag.

**Flag: flag{y0u_40+_s3rv3d!}**

**Category: Crypto (Wh@t7he####)**

**Description:** Deciphering an encoded message hidden within an esoteric programming language script.

**Challenge Overview:** A .file was provided containing encoded text. Upon analyzing the content, it was identified as ReverseFuck, an esoteric programming language. By decoding the ReverseFuck script, the flag was successfully retrieved.

**Steps for Finding the Flag:**

1. **File Inspection:** Opened the provided .file and identified the content as code.
2. **Esolang Identification:** Recognized the code as ReverseFuck, a variation of the Brainfuck language.
3. **Decoding Attempt:** Used an online ReverseFuck decoder to process the code.
4. **Flag Extraction:** Successfully retrieved the decoded message containing the flag.

**Flag: flag{R3vers3ddd_70_g3t_m3}**