CAPTURE THE FLAGS

A CTF, or Capture the Flag, is a cybersecurity competition and training exercise where participants solve challenges to find hidden digital "flags", a secret string of text.

By Aditya Singh Rathaur

Contents

Challenge 1: Forbidden Path

Description:

A **path traversal vulnerability** is a security flaw that allows an attacker to access files and directories stored outside the normal web root folder by manipulating file paths in user input. Attackers exploit this by inserting sequences like ../ (dot-dot-slash) to navigate up the directory and read sensitive files, such as configuration files or passwords, that should not be accessible via the application.

Impact:

The impact of this is that the attacker can look at secret or important files like source, code passwords, personal info, database credentials or system settings. The attacker can gain access to data that compromises the confidentiality, integrity, or availability of the application and server. Sometimes, they can even change these files or put harmful software on the server, leading to more damage.

Remediation:

- Validate all user input strictly: Only allow known safe (whitelisted) filenames or patterns, never rely on blacklisting or attempting to remove ../ (dot-dot-slash).
- Normalize file paths: Convert any user-provided path to an absolute one and check it remains inside a predetermined (safe) directory.
- Limit permissions: Ensure the application and its users cannot access critical system files, even if a vulnerability exists.

References:

- https://portswigger.net/web-security/learning-paths/path-traversal
- https://owasp.org/www-community/attacks/Path_Traversal

Proof of Concept:

Step 1: Navigate the target URL of the challenge: https://play.picoctf.org/practice/challenge/270 and click on Launch Instance.

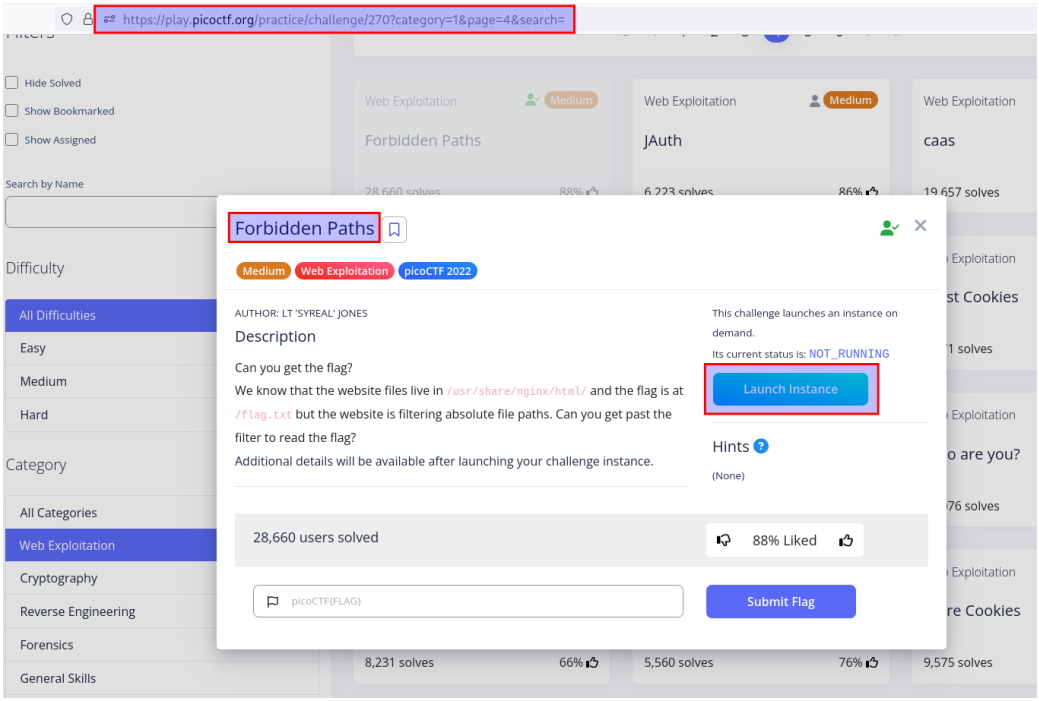

Figure 1

Step 2: Once we click on launch instance, we get a hyperlink for the website containing the flag. Click on it to open the website.

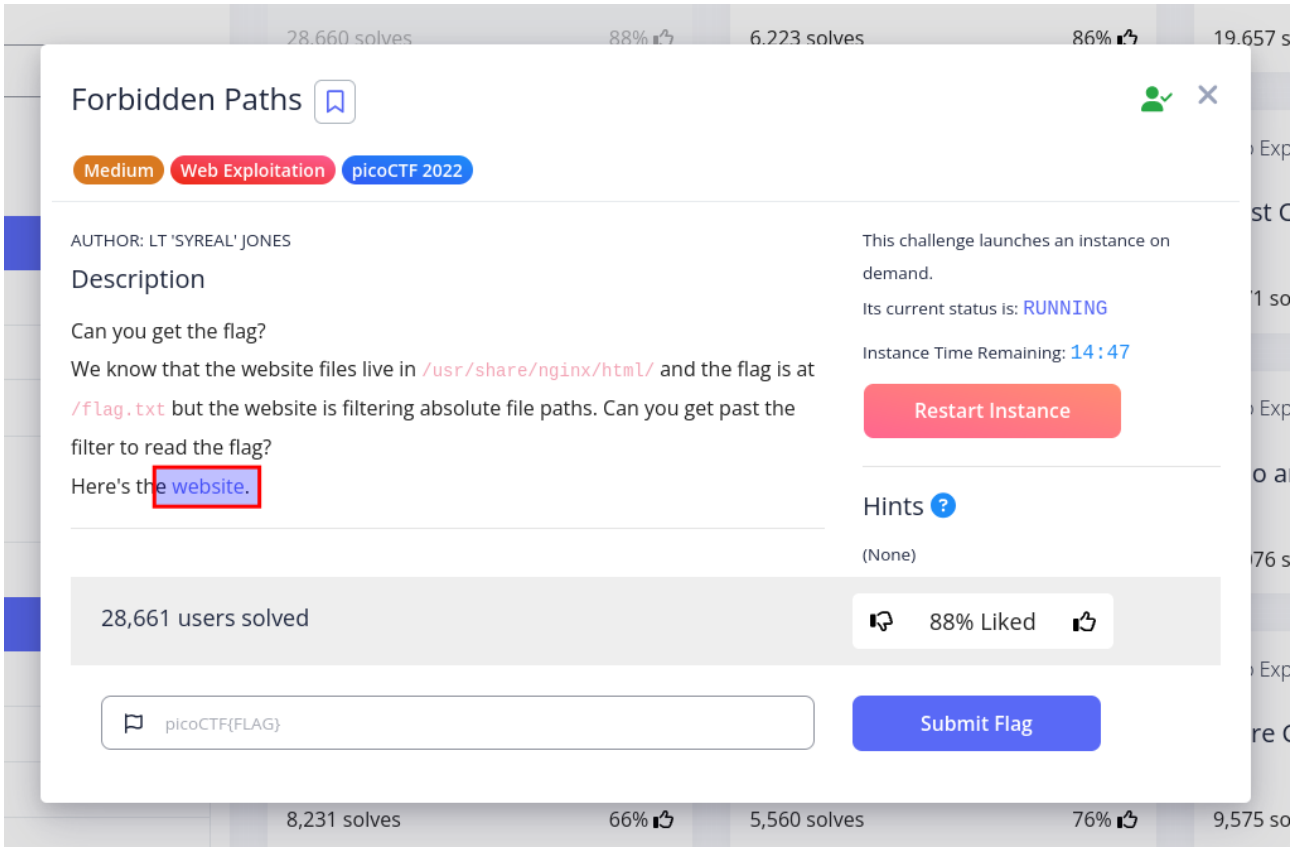

Figure 2

Step 3: On the website, we can find three .txt files, namely **divine-comedy.txt**, **oliver-twist.txt** and **the-happy-prince.txt**. Enter the first name i.e. **divine-comedy.txt** and click on read. We are navigated to /read.php.

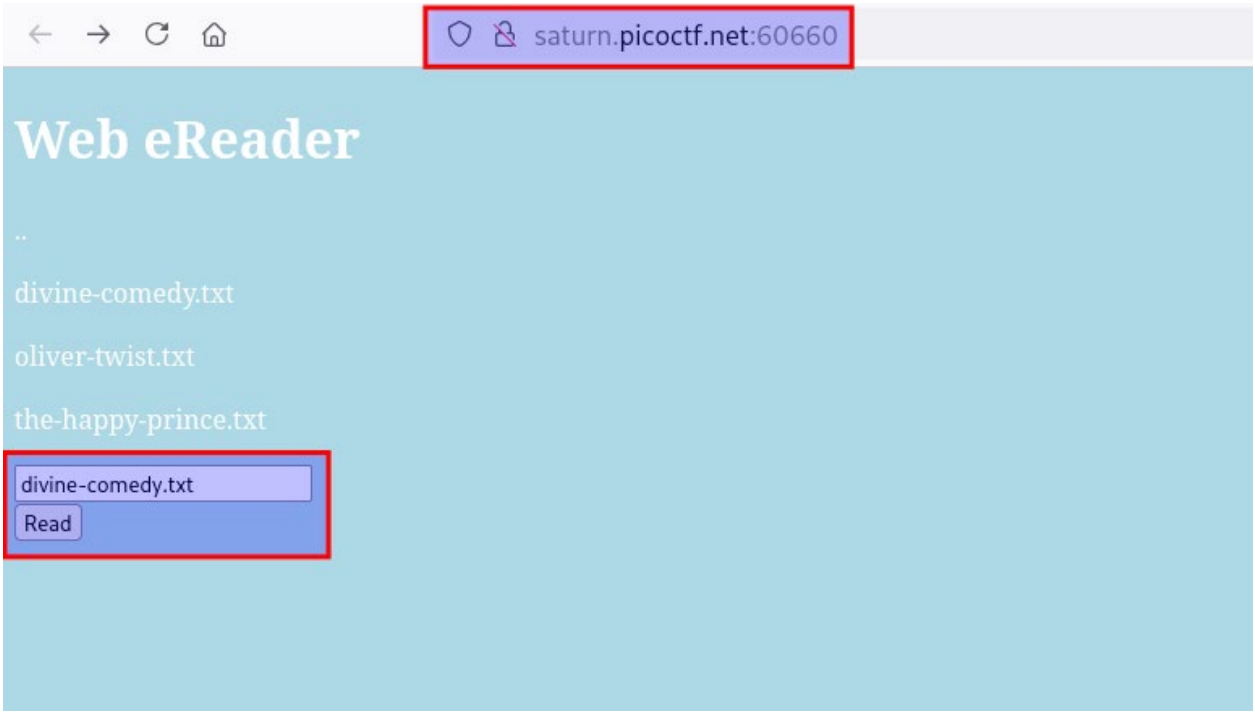

Figure 3

Step 4: Go back to the home page of the website and then repeat the last step with **oliver-twist.txt** and **the-happy-prince.txt**.

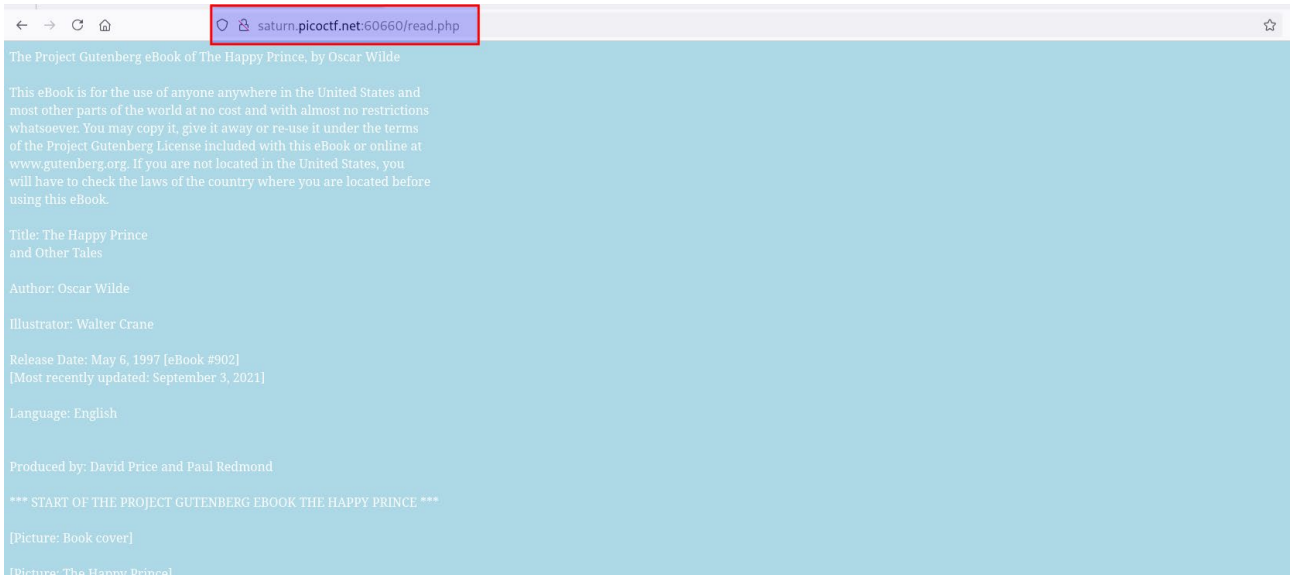

Figure 4

Step 5: Go back to the home page of the website and enter **flag.txt** and click on read. We get the result that the file does not exist in this directory.

Figure 5

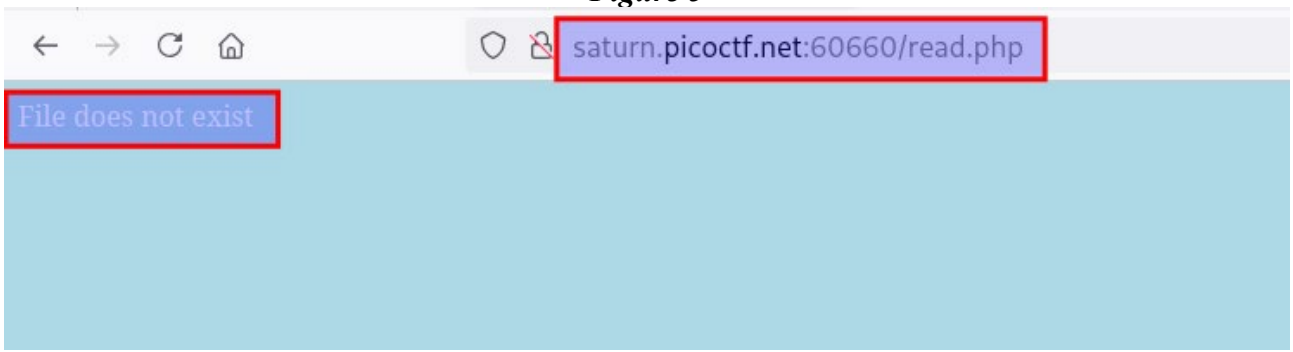

Figure 6

Step 6: As given in the challenge description in the *Figure 1*, the files live in **/usr/share/nginx/html/** and the flag is at **/flag.txt**. It is also mentioned that the website is filtering absolute file paths which means that we need to navigate using relative paths.

Figure 7

Step 7: As both **/flag.txt** and **/usr/share/nginx/html/** are absolute paths, we can guess that we need to go back four folders (html, nginx,share,usr) to go the root folder and then to **flag.txt** file. In the input box, enter **../../../../flag.txt** in which **../../../../** takes us to the root folder and **/flag.txt** takes us to our desired file. Click on read and we can find the flag as shown in *Figure 8*.

Figure 8

Challenge 2: Logon

Description:

Authentication bypass happens when an attacker can get into a system, website, or app without logging in the correct way. Instead of using a username and password, they find a weakness in the system that lets them skip or avoid the login checks entirely. This can happen due to programming mistakes or flaws in how the application handles user access.

Impact:

When an attacker bypasses authentication, they can access private or protected information without permission. They might steal sensitive data, delete or change important files, or even take full control of the system, including administrator privileges. This can lead to serious breaches, data loss, or further attacks on the system.

Remediation:

To fix authentication bypass, ensure the system checks each user carefully before giving access. Use strong and updated authentication methods like multi-factor authentication (MFA). Always validate and sanitize user input to prevent tricks or manipulations. Keep software and security patches up to date and encrypt session information like cookies. Also, avoid exposing any alternate ways to enter the system without authentication.

Reference:

- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- https://portswigger.net/web-security/authentication

Proof on Concept:

Step 1: Navigate the target URL of the challenge: https://play.picoctf.org/practice/challenge/46 and click on the hyperlink to the website where the flag exists.

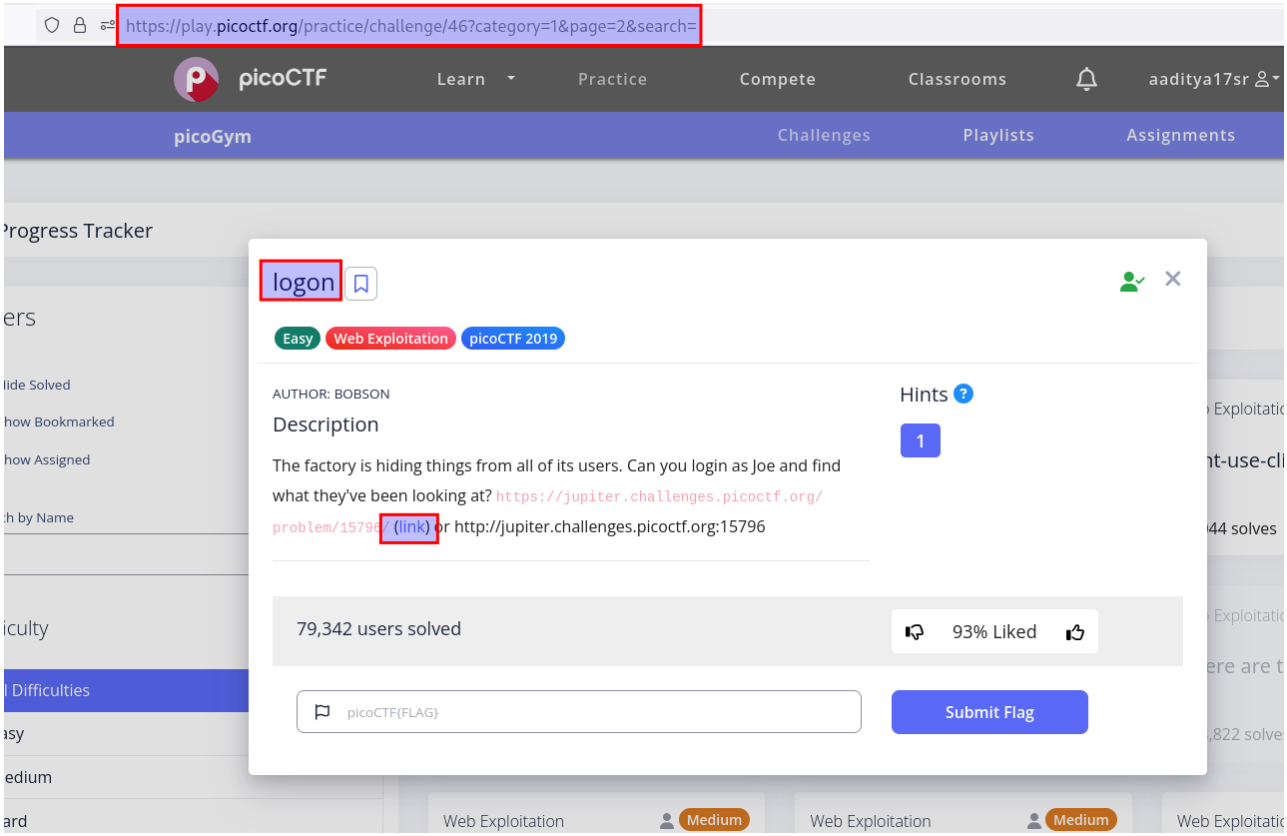

Figure 9

Step 2: As we can see *Figure 10*, we have a factory login page having input box to enter username and password. Enter username as **Joe** and a random password, for example we have taken **1234**.

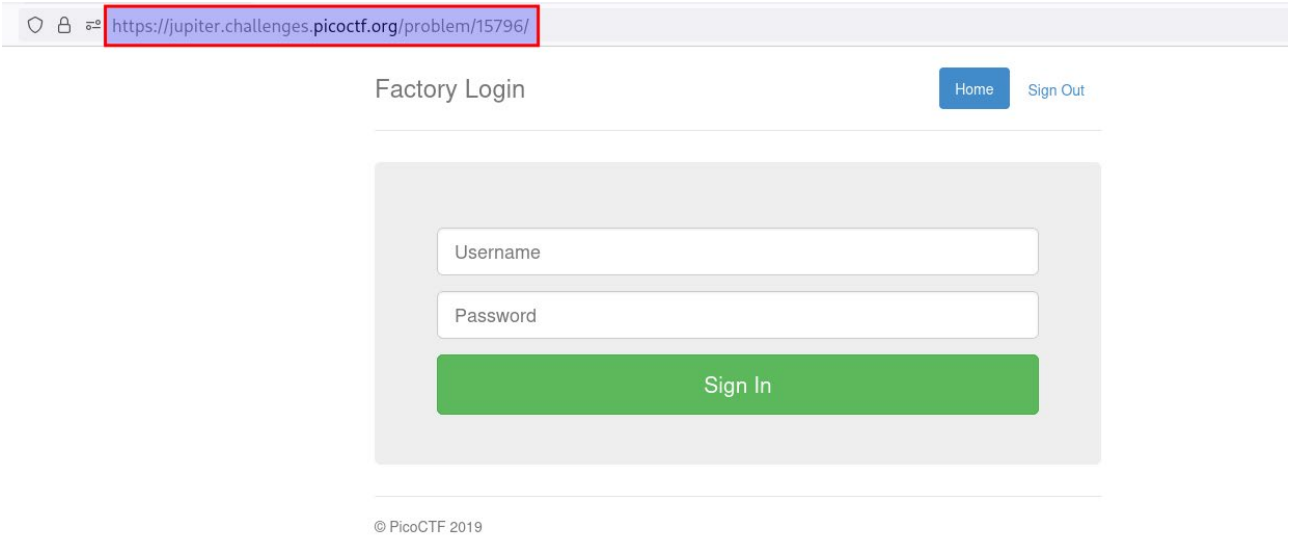

Figure 10

Step 3: As we can see the login is successful, but it says that we won't be able to see the flag.

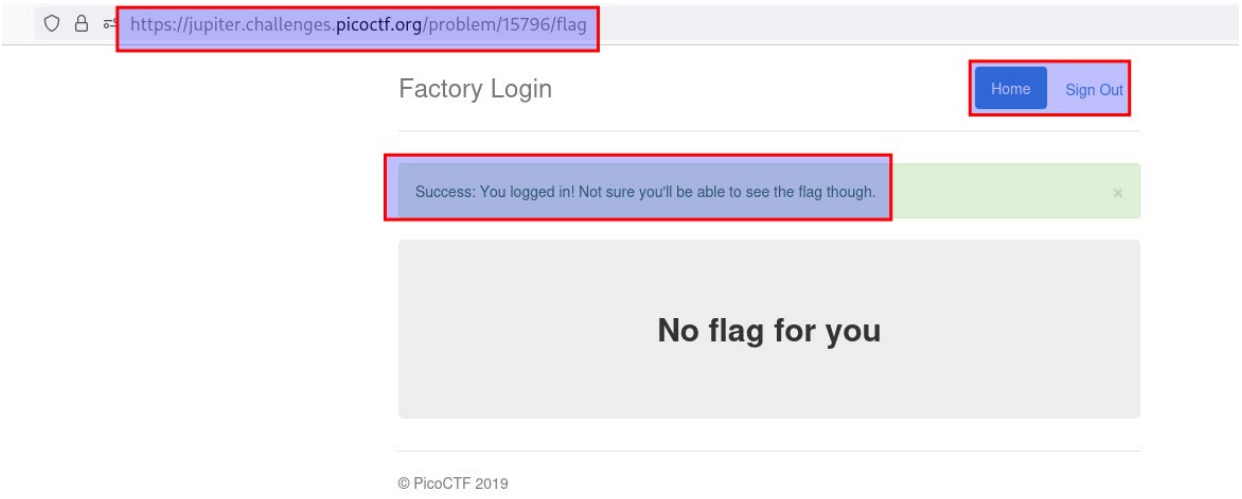

Figure 11

Step 4: Click on the **Home** or **Sign Out** button. We go back to the login page. **Burp Suite** is a popular tool used for web application security testing. It intercepts and analyses web traffic to find vulnerabilities, helping testers identify security issues like SQL injection and cross-site scripting. It offers features for manual testing, automated scanning, and attack simulation. Open **Burpsuite** and go to **Proxy>HTTP history** and click on **Open Browser**.

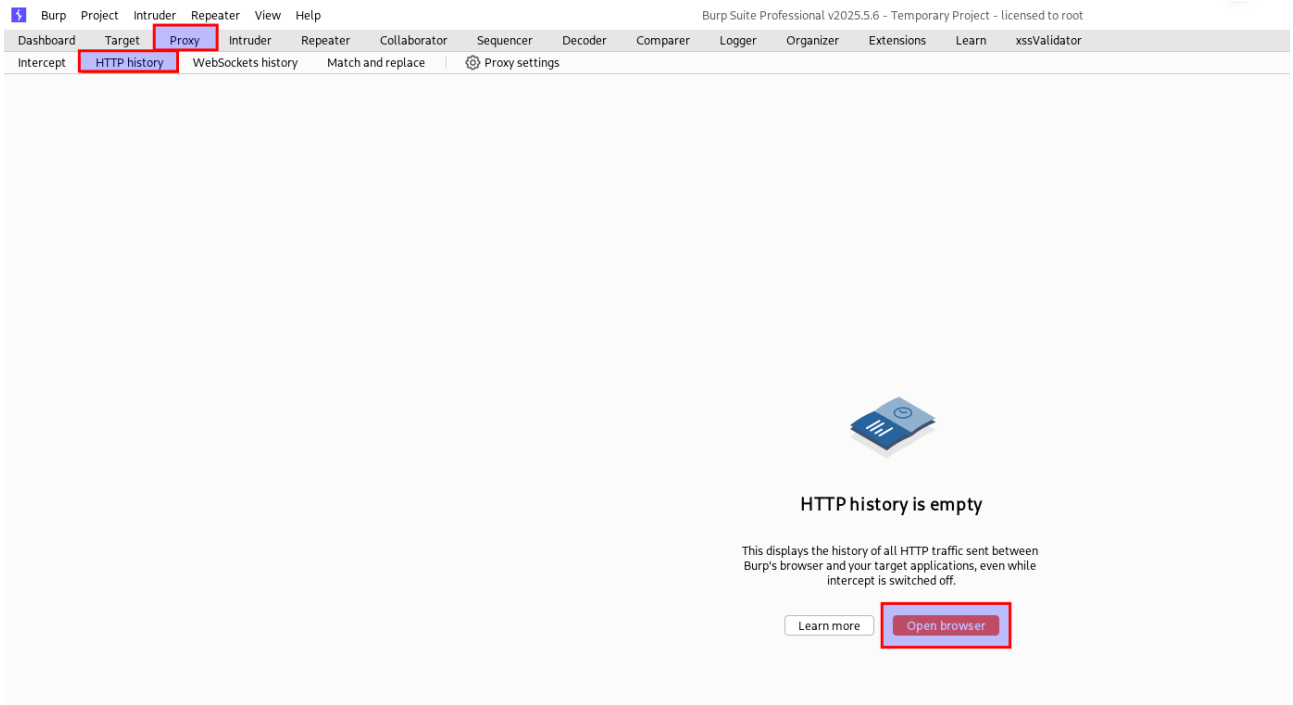

Figure 12

Step 5: Copy the link of the login page from the browser and paste it in the new Chromium Browser window opened using Burp suite. Give random username and password. For example, we gave both username and password as **test** as shown *Figure 13* and Sign in.

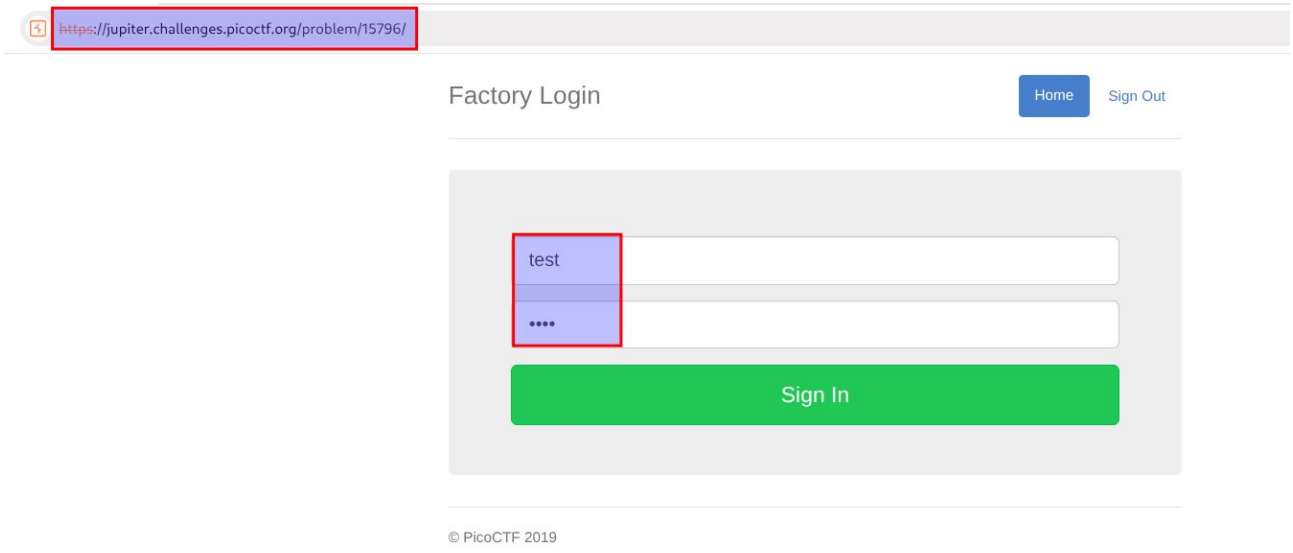

Figure 13

Step 6: In **Burp suite>Proxy>HTTP history** we can see multiple **POST** and **GET** requests. Click on the request having the URL **/problem/15796/flag**. We can observe the Request and the Response in *Figure 14*. In the **Request** we can observe that **admin=False**. To send a modified request, right click on that particular HTTP GET request and click on **Send to Repeater**.

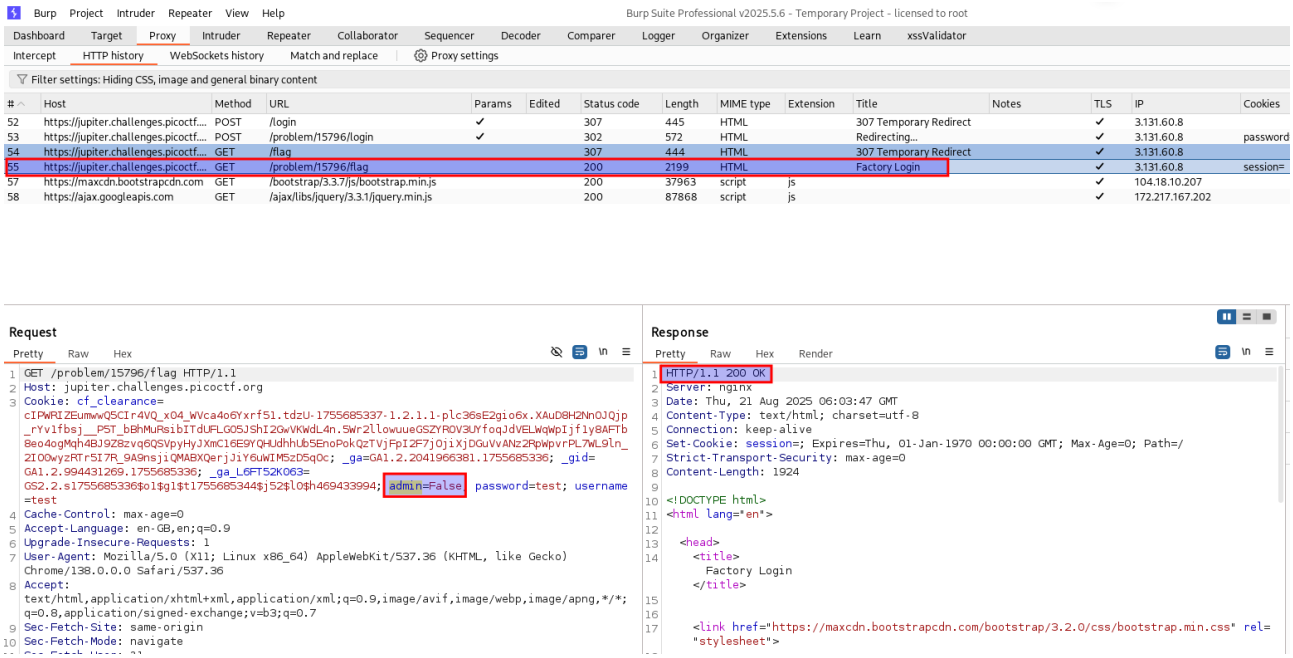

Figure 14

Step 7: Go to **Repeater** tab and click and modify the request and make it **admin=True** as shown in *Figure 15* and click on **Send**.

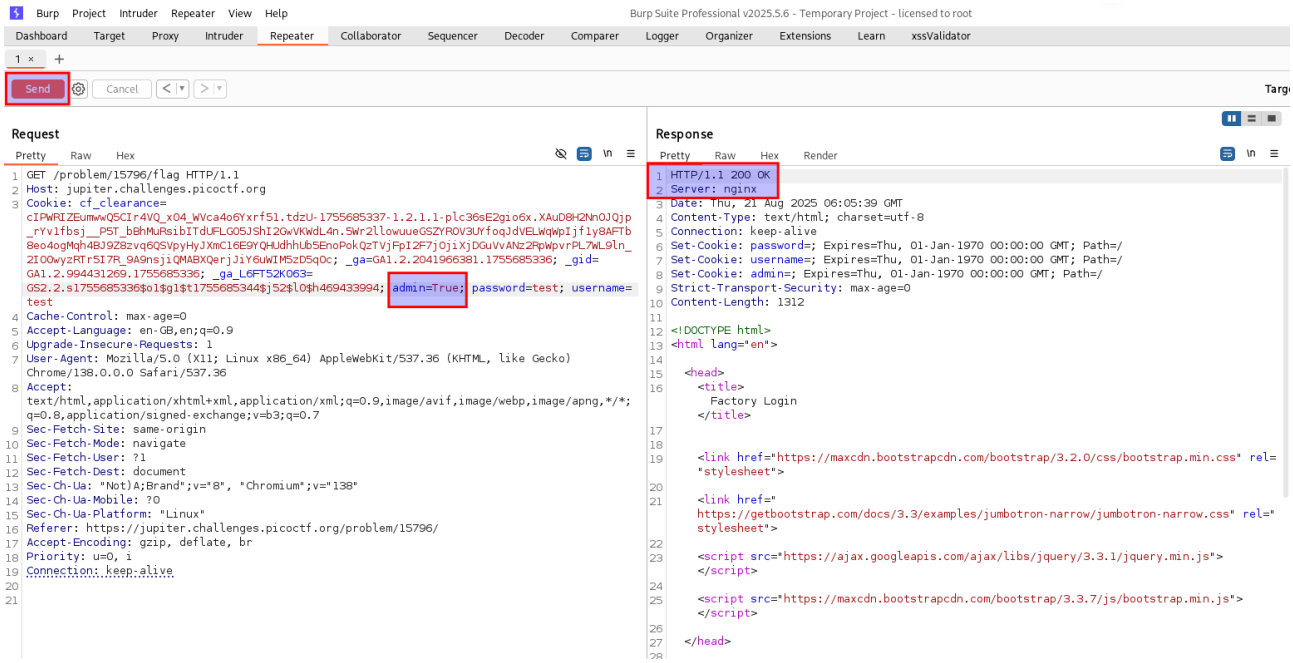

Figure 15

Step 7: We can observe that in *Figure 15* that we get a **200 OK** response. Scroll down in the response and we can find the hidden flag in the response as shown in *Figure 16*.

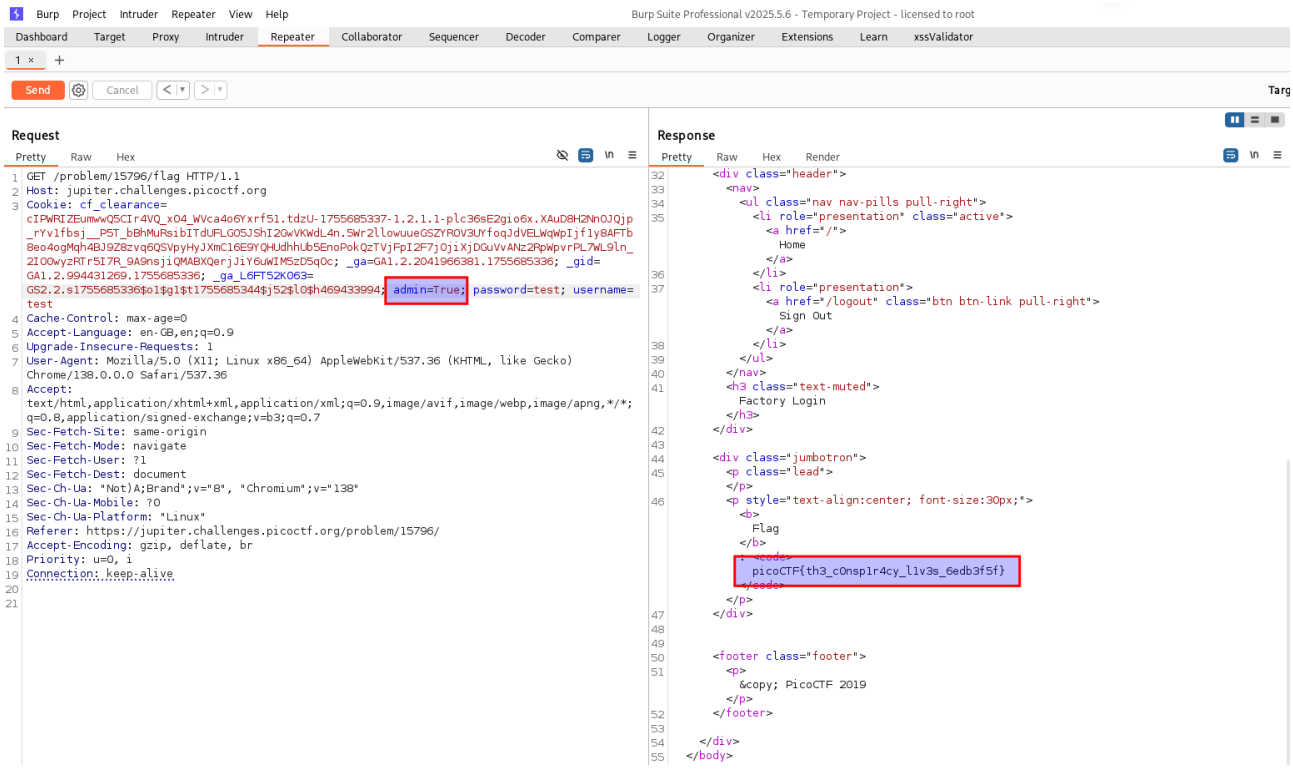

Figure 16

Challenge 3: Can you find the robots?

Description:

Reconnaissance in cybersecurity means gathering information about a target system, network, or organization before launching an attack. It is the first step attackers use to learn about vulnerabilities, technologies, people, and infrastructure without immediately being noticed. Reconnaissance can be of two types: passive, where information is collected from public sources without touching the target, and active, where the attacker directly interacts with the target system to gather more detailed data.

Impact:

Reconnaissance helps attackers plan their attacks better by revealing weak points and sensitive information about the target. If attackers effectively gather this information, they can launch more successful and damaging attacks such as hacking into systems, stealing data, or causing disruptions. Even if no attack follows, reconnaissance may expose confidential details that could harm the organization's reputation or security.

Remediation:

To reduce the risk from reconnaissance, organizations should limit the exposure of sensitive information online and use strong network defences like firewalls and intrusion detection systems to detect and block scanning activities. Regularly monitor systems for unusual activity, secure publicly available information, and educate staff about social engineering risks. Keeping software updated and minimizing publicly accessible details also help protect against attackers gathering useful reconnaissance data.

Reference:

- https://www.sentinelone.com/cybersecurity-101/threat-intelligence/what-is-cyber-reconnaissance/
- https://www.imperva.com/learn/data-security/cybersecurity-reconnaissance/

Proof on Concept:

Step 1: Navigate to the target URL of the challenge: https://play.picoctf.org/practice/challenge/4 and click on the hyperlink to the website where the flag exists.

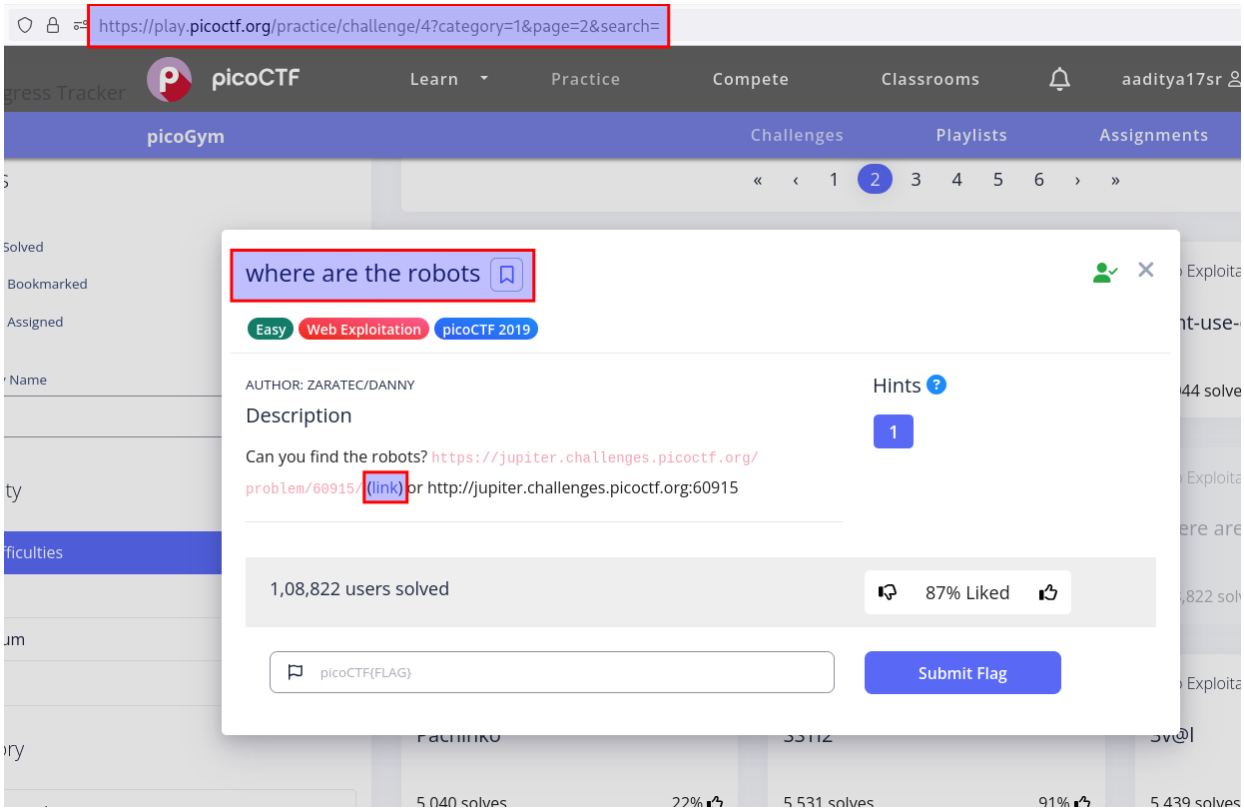

Figure 17

Step 2: We get a black screen with a **Welcome** message saying **Where are the robots?**

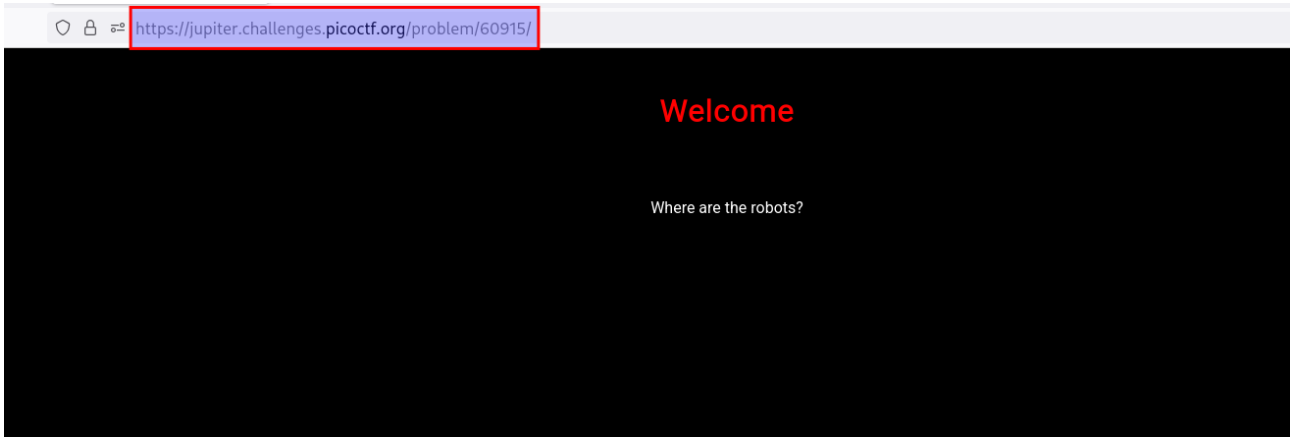

Figure 18

Step 3: Using the URL, go to **robots.txt**. W can see a blank white screen. Right click on the page and click on **Inspect**.

Figure 19

Step 4: In the **Inspector** tab, we can fine a message saying **Disallow: /8028f.html**.

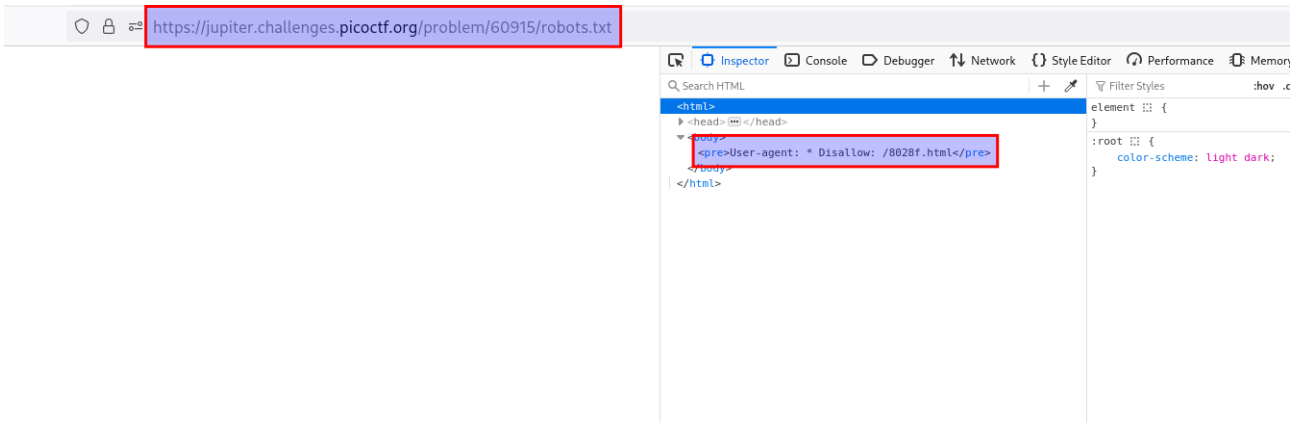

Figure 20

Step 5: Go to the home page of the website and then go to **/8028f.html** and we can find the flag.

Figure 21

Challenge 4: GET aHEAD

Description:

HTTP method manipulation occurs when attackers exploit or abuse the HTTP request methods (such as GET, POST, HEAD, PUT, DELETE, OPTIONS, TRACE, etc.) that a web application supports. If a server or web application improperly allows unnecessary methods or lacks adequate restrictions, attackers may utilize these methods to perform unauthorized actions, such as uploading malicious files, deleting content, or bypassing intended security controls.

Impact:

The impact of HTTP method manipulation can be severe for a web application and its users. Attackers could exploit insecure HTTP methods to manipulate, delete, or disclose sensitive data. For example, an attacker sending a PUT request could modify an important resource, or a DELETE request could irretrievably remove data. Beyond data loss, attackers might gain unauthorized access to resources or functions or use methods like TRACE for information disclosure or Cross-Site Tracing attacks. These weaknesses can lead to a loss of data integrity, breach of confidentiality, and even total compromise of the application's security.

Remediation:

To remediate HTTP method manipulation issues, it is recommended to disable or restrict any HTTP methods not strictly required for the application to function. Only allow essential methods, and configure web servers, application firewalls, and middleware to enforce these restrictions. Sensitive or potentially destructive methods should be protected through strict access controls. Additionally, always validate and sanitize inputs for every allowed method, and regularly audit which HTTP methods are enabled. Avoid workarounds that bypass security measures, and monitor for any unusual usage of HTTP methods to detect potential attacks

Reference:

- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- https://portswigger.net/web-security/authentication

Proof on Concept:

Step 1: Navigate to the target URL of the challenge: https://play.picoctf.org/practice/challenge/132 and click on the hyperlink to the website where the flag exists.

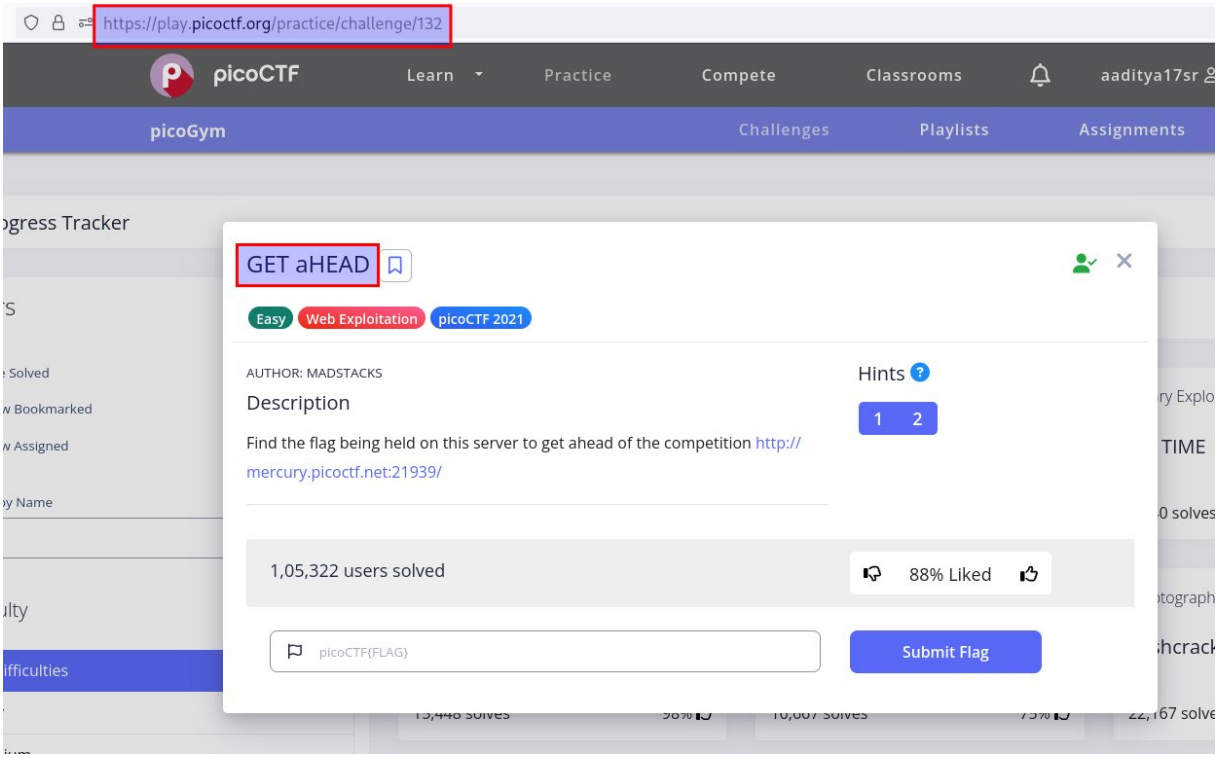

Figure 22

Step 2: As we can see in *Figure 23* and *Figure 24*, when we click on **Choose Red**, the background turns red and the background turn blue when we click on **Choose Blue**. Copy the link of the website.

Figure 23

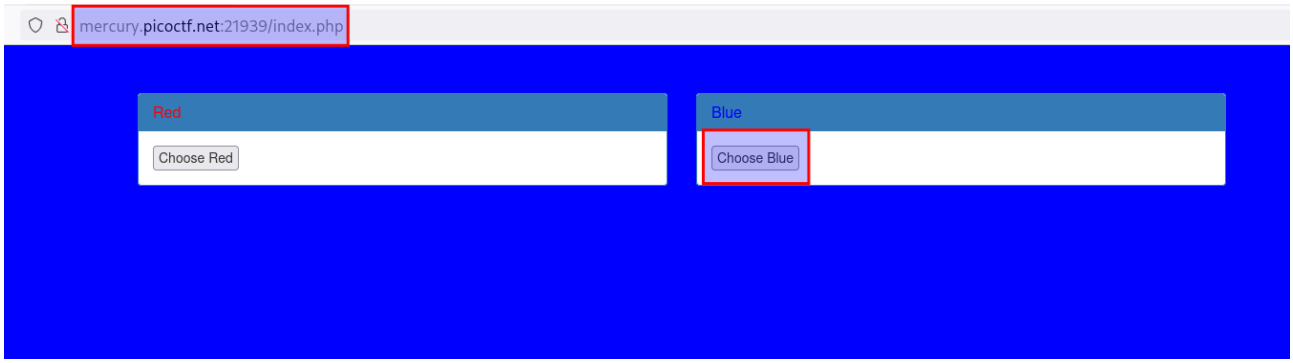

Figure 24

Step 3: Open Burp suite. Go to **Proxy>HTTP history** and click on **Open Browser**. Paste the copied link of the website.

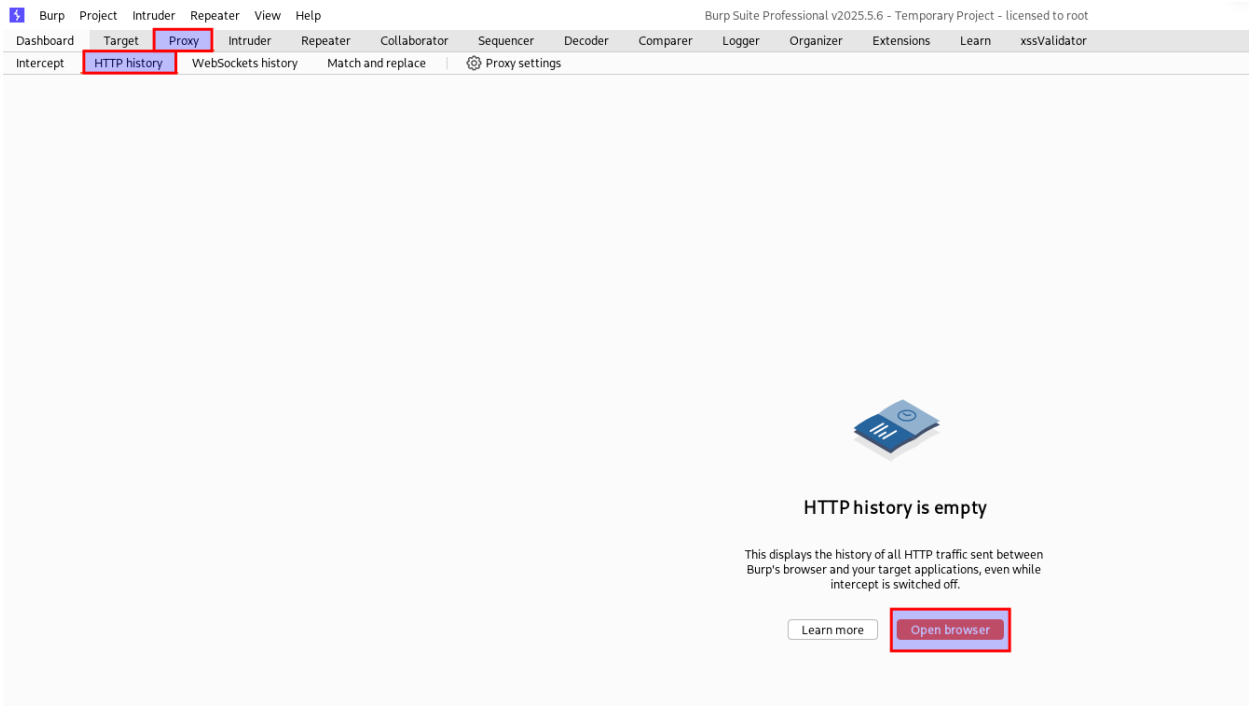

Figure 25

Step 4: In the **HTTP history**, we can see multiple **GET** requests. Click on **/index.php** and we can see the Request and Response tab. Right-click on the request and click on **Repeater**.

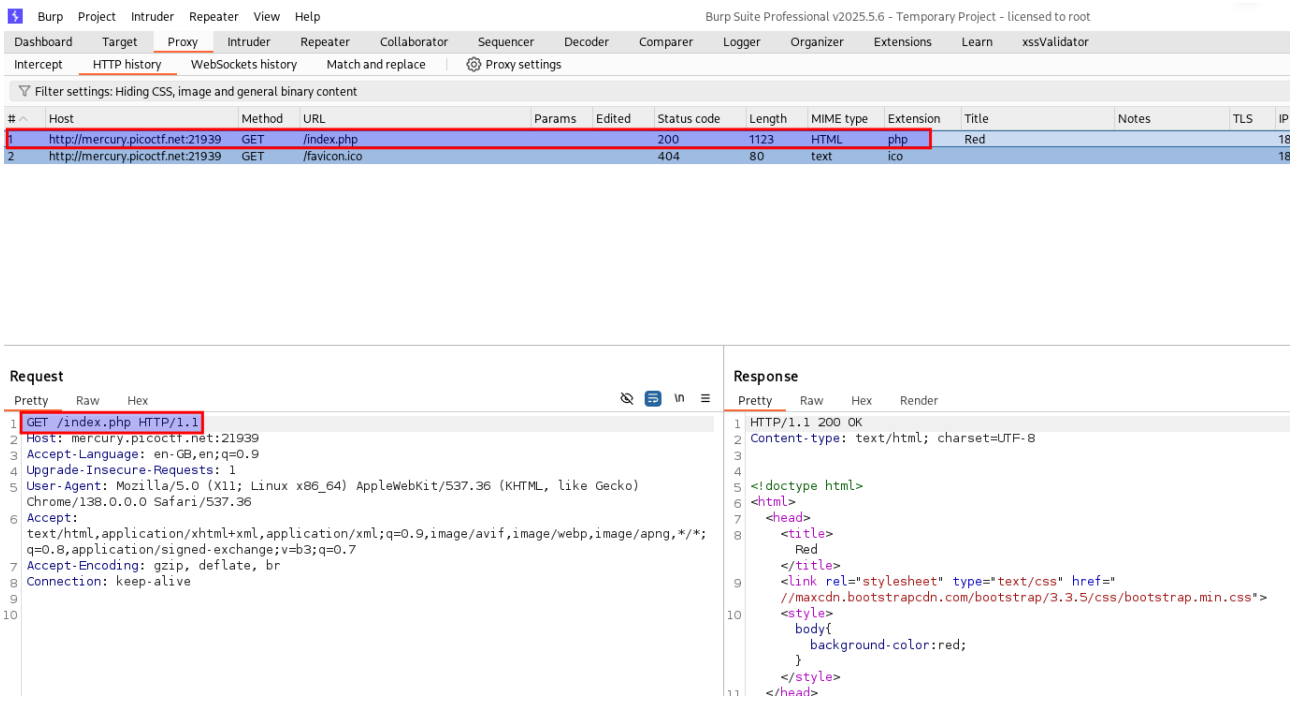

Figure 26

Step 5: As we can see in *Figure 27*, it is a **GET /index.php** request and its response is **200 OK**.

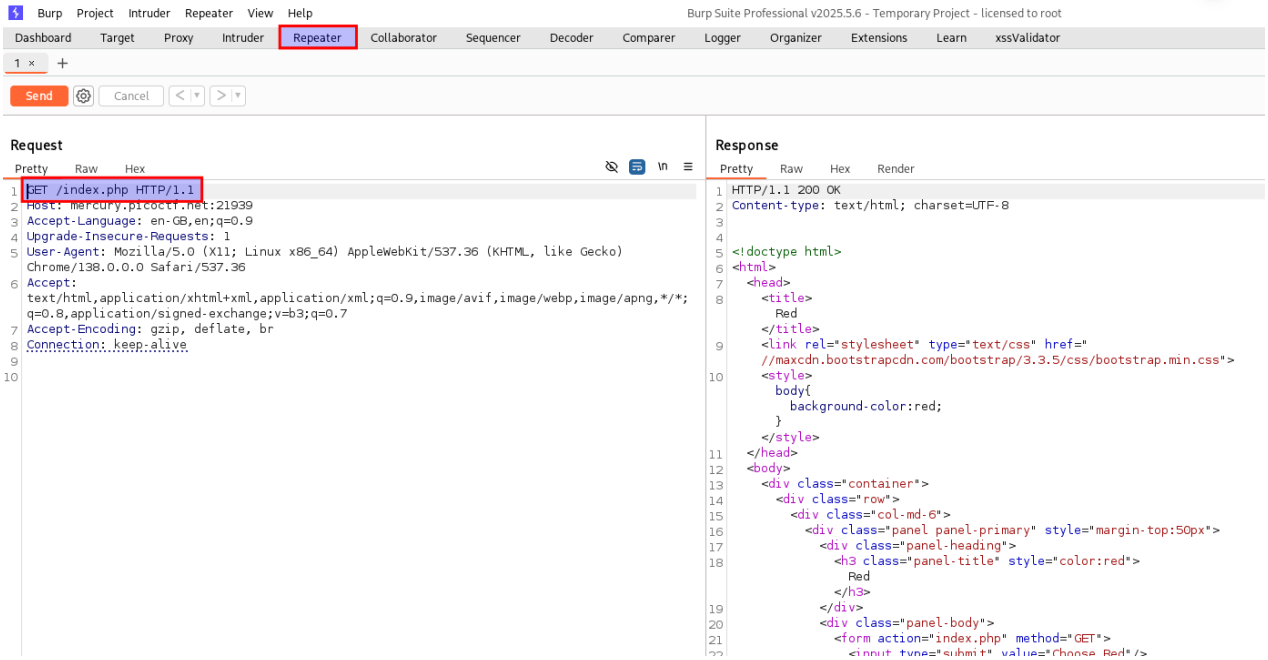

Figure 27

Step 6: Try modifying the request to **HEAD /index.php** and click on **Send**. In the response, be get **200 OK** and also the flag in it.

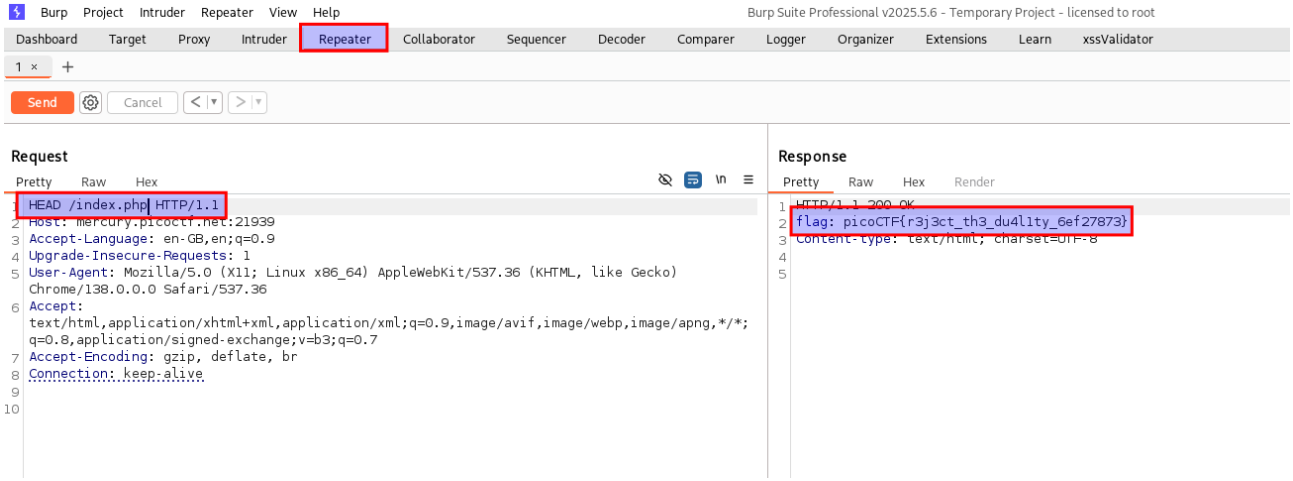

Figure 28