



SawCTF

Created	@October 15, 2025 12:43 PM
Tags	
Site	

The screenshot shows the challenge page for 'SawCTF' on TryHackMe. It features the OFFSEC logo, the challenge title 'SawCTF', a description 'CTF inspirado em Jogos Mortais, criado pelo OFFSEC para a SATECH/UFSC', and metrics indicating it takes 60 minutes, has 23 users, and is marked as medium ('M').

Link: <https://tryhackme.com/room/sawctf>

Task 1

Para completar a Task 1 bastava realizar um enumeração de portas no endereço IP e descobrir quais serviços a máquina estava rodando. Isso foi feito pelo comando abaixo.

```
nmap -sV <endereço_ip>
```

Resultado do comando:

```
(kvothe@Viper)-[~]
$ nmap -sV 10.10.14.218
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-15 09:19 -03
Nmap scan report for 10.10.14.218
Host is up (0.22s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.57 ((Debian))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

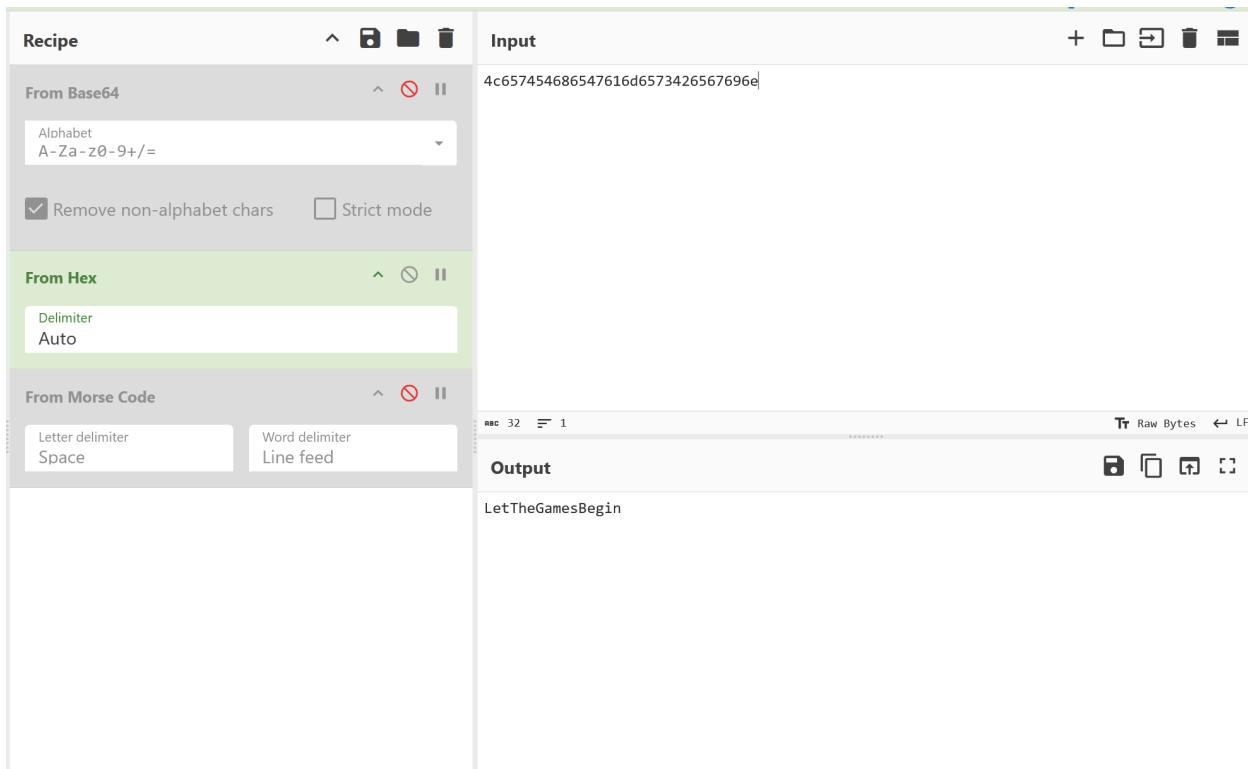
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.75 seconds
```

Task 2

Na Task 2, precisamos encontrar uma web flag, decodificá-la e encontrar as credenciais de um usuário. Para isso, como a máquina estava com a porta 80 aberta, colocamos o IP no navegador e acessamos uma página Web. Ao inspecionar a página, encontramos a web flag `4c657454686547616d6573426567696e` nos comentários do código da página.

The screenshot shows a browser window titled "SawCTF" displaying a dark-themed page. The page features a large image of the Jigsaw mask from the movie "The Saw" series. At the top, there is a banner with the word "OFFSEC". Below the banner, a message reads: "Que os jogos começem! Você deve passar pela armadilha para encontrar a chave. Faça a sua escolha." In the bottom right corner of the page, there is a small text box containing the web flag: "O criador das provas não fala, apenas observa... mas deixa uma mensagem clara: todos são convidados... web.flag: 4c657454686547616d6573426567696e". The browser's developer tools are open, showing the HTML structure and the CSS inspector panel. The CSS inspector highlights the element containing the flag with a purple border and provides a detailed box model breakdown.

Para de codificar a web flag, primeiramente tentamos usar o hashid e o hashcat. Entretanto, a flag não estava codificada em hash e sim em hexadecimal. Assim que percebemos isso, usamos o CyberChef para decodificar o hex, obtendo a string `LetTheGamesBegin`.



Em seguida, usamos o gobuster para fazer o mapeamento dos diretórios da aplicação web e descobrimos o diretório `/trap`, o qual nos deu acesso a uma página de login. Ao inspecionar a página, percebemos que o login da página estava embutido na própria página:

```

<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <div class="login-card">
      <label for="user">User</label>
      <input id="user" type="text" name="user">
      <label for="pass">Password</label>
      <input id="pass" type="password" name="pass">
      <button onclick="submitLogin()> Logar</button>
    </div>
    <div class="warning-box">
      Lembre-se que o tempo é o seu inimigo... você precisa chegar no quebra-cabeça para encontrar a chave
    </div>
    <div class="button-box">Se você falhar, todos nós falhamos</div>
    <script src="https://cdns.cloudflare.com/ajax/libs/blueimo-md5/2.19.0/js/md5.min.js"></script>
    <script>
      function hoffman(input) {
        let str = '';
        for (let i = 0; i < input.length; i += 2) {
          str += String.fromCharCode(parseInt(input.substr(i, 2), 16));
        }
        return str.split('').reverse().join('');
      }

      function submitLogin() {
        const userInput = document.getElementById("user").value.trim();
        const passInput = document.getElementById("pass").value.trim();
        const validUser = "6209804952225ab3d14348307b5a4a27"; //MD5
        const validPass = "c145210d2ed44495b83442d2276f19bc"; //MD5
        if (userInput === validUser && passInput === validPass) {
          const tra = "2772656d6172686e86f6a277081727427"; //MD5
          window.location.href = hoffman(tra);
        } else {
          alert("Acesso negado! Usuário ou senha inválidos.");
        }
      }
    </script>
  </body>
</html>
<!-- Go /trap/jigsaw-->

```

Inspecionando o código de login, encontramos dois códigos:

`6209804952225ab3d14348307b5a4a27` e `b145210d2ed44495b83442d2276f19bc`, referentes a um usuário e à senha do usuário, respectivamente. O comentário ao lado dos códigos nos induz a acreditar que ambos os códigos são os hashes do nome do usuário e da senha do usuário, respectivamente. Então, tentamos quebrá-los com um ataque de dicionário usando a wordlist `rockyou`, o ruleset `best64` e o algoritmo MD5. Entretanto, só conseguimos descobrir o nome do usuário:

```
hashcat -a 0 -m 0 hash.txt /usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/rules/best64.rule
```

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 4 MB

Dictionary cache hit:

- * **Filename...**: /usr/share/wordlists/rockyou.txt
- * **Passwords...**: 14344385
- * **Bytes.....**: 139921507
- * **Keyspace...**: 1104517645

6209804952225ab3d14348307b5a4a27:amanda

Assim, notamos o comentário no final do código da página: `<!-- Go /trap/jigsaw -- >`. Então, acessamos o diretório `/trap/jigsaw` e inspecionamos o código da página, achando o verdadeiro código referente à senha do usuário `amanda`:

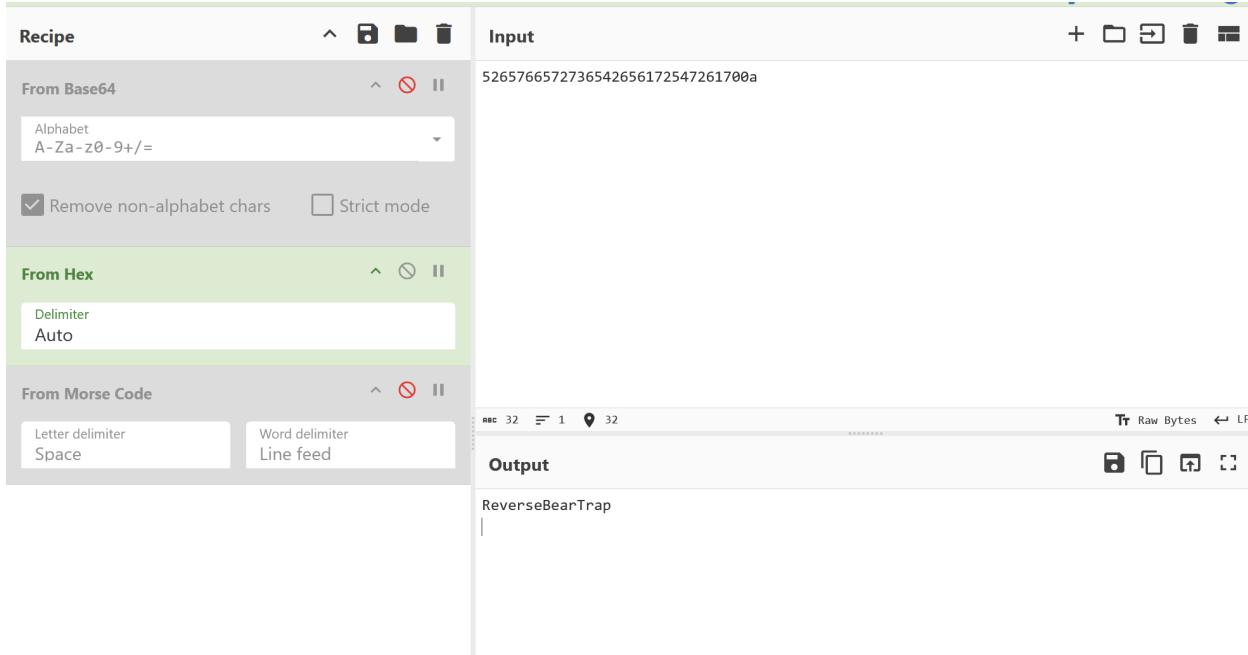
The screenshot shows a browser window with multiple tabs all titled "SawCTF". The active tab is displaying a page with a woman's face in profile, looking over her shoulder. A yellow banner at the top says "Você passou pela armadilha... agora pegue a chave e comece o seu jogo". Below the banner, there is some text in Portuguese: "Se tiver sucesso, todos teremos". The browser's developer tools are open, specifically the "Inspector" and "Computed" panels. The "Computed" panel shows the CSS properties for the "body" element, including a box model diagram and detailed styling information.

```
html > body { margin: 0; border: 0; padding: 0; width: 100%; height: 100%; }
```

Computed Properties (for body):

- width: 100%
- height: 100%
- margin: 0
- border: 0
- padding: 0
- display: block
- float: none
- line-height: normal
- position: static
- z-index: auto

Colocamos o código `5265766572736542656172547261700a` hex no CyberChef para extrair a senha `ReverseBearTrap`.



Task 3

Na Task 3, precisamos descobrir as credenciais de um segundo usuário e uma flag com o usuário achado na task anterior. Assim, usamos o usuário `amanda` e a senha `ReverseBearTrap` para acessar a máquina via SSH e encontramos a flag

`RG9udFRydXNOVGhLT25IV2hvU2F2ZXNZb3U=`.

```

(kvothe@Viper) [~]
$ sudo ssh amanda@10.10.14.218
[sudo] password for kvothe:
The authenticity of host '10.10.14.218 (10.10.14.218)' can't be established.
ED25519 key fingerprint is SHA256:LsWOF402aDb/w6V7Z5VEAcjNfkxMmPOzyEIC7HMr91o.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '10.10.14.218' (ED25519) to the list of known hosts.
amanda@10.10.14.218's password:
Linux sawctf 6.1.0-11-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.38-4 (2023-08-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
amanda@sawctf:~$ ls
user.txt
amanda@sawctf:~$ cat user.txt
Flag: RG9udFRydXN0VGhlT25lV2hvU2F2ZXNZb3U=

```

Percebemos que a flag está codificada em base64, então usamos o CyberChef para obter a flag original [DontTrustTheOneWhoSavesYou](#).

The screenshot shows the CyberChef interface with the following configuration:

- Recipe:** From Base64
- Input:** RG9udFRydXN0VGhlT25lV2hvU2F2ZXNZb3U=
- Show Base64 offsets:** Enabled
- Input format:** Raw
- Output:** DontTrustTheOneWhoSavesYou

Agora, precisamos encontrar o segundo usuário do sistema. Ao realizar o login com as credenciais do usuário `amanda` na página `/trap`, somos redirecionados a outra página. Ao inspecioná-la, encontramos outros dois códigos:

`4957616e74546f506c61794147616d650a` e `....-.-. . .`.

The screenshot shows a browser window with multiple tabs labeled "SawCTF". The active tab displays a dark-themed page featuring a close-up of a human face with a Jigsaw mask. The text on the page reads:
"Se você for bom em antecipar a mente humana, nada fica ao acaso"
"Suas emoções também são suas fraquezas"
Below the page content, the browser's developer tools are open, specifically the "Inspector" panel. The HTML structure is visible, showing the following code snippet:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <div class="message">
      Se você for bom em antecipar a mente humana, nada fica ao acaso
      <!--4957616e74546f506c61794147616d650a-->
    </div>
    <div class="bottom-message">
      Suas emoções também são suas fraquezas
      ....-.-. . .-.
    </div>
  </body>
</html>
```

The "Box Model" section of the developer tools shows a detailed breakdown of the element's dimensions and padding. The overall width is 2560x72, with padding of 0 on all sides. The "Box Model Properties" section shows the following settings:

content-box
block
none
normal
static
auto

O primeiro está codificado em hexadecimal. Ao usar o CyberChef para decodificá-lo, obtemos a senha `IWantToPlayAGame`.

The screenshot shows the CyberChef interface with a recipe for decoding. The 'Input' field contains the Base64 string: 4957616e74546f506c61794147616d650a. The 'From Base64' section is active, with the 'Alphabet' dropdown set to 'A-Za-z0-9+/=' and the 'Remove non-alphabet chars' checkbox checked. The 'Output' field displays the decoded text: IWantToPlayAGame.

O segundo código está codificado em código Morse. Ao usar o CyberChef, obtemos o usuário [john](#).

The screenshot shows the CyberChef interface with a recipe for decoding. The 'Input' field contains the Morse code: -.- - - - .-.|. The 'From Morse Code' section is active, with the 'Letter delimiter' set to 'Space' and the 'Word delimiter' set to 'Line feed'. The 'Output' field displays the decoded text: JOHN.

Task 4

Na Task 4, precisamos descobrir a senha do root e a root flag. Para isso, usamos o SSH para nos conectarmos à máquina alvo com as credenciais do usuário `john` e obtivemos o código `545746725a566c7664584a44614739705932553d` referente à senha do root.

```
(kvothe@Viper) [~]
$ sudo ssh john@10.10.21.97
[sudo] password for kvothe:
john@10.10.21.97's password:
Linux sawctf 6.1.0-11-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.38-4 (2023-08-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
john@sawctf:~$ ls
root.txt
john@sawctf:~$ cat root.txt
root pass: 545746725a566c7664584a44614739705932553d
```

Para decodificar o código, usamos o CyberChef, obtendo a senha `MakeYourChoice`:

The screenshot shows the CyberChef interface with the following configuration:

- Recipe:** From Hex
- Input:** 545746725a566c7664584a44614739705932553d
- From Base64** section is collapsed.
- Show Base64 offsets** checkbox is checked.
- Output:** MakeYourChoice

Para acessar o usuário root, percebemos por meio do comando abaixo que o programa `su` estava com SUID setado.

```
find / -perm -4000 -type f 2>/dev/null
```

Assim, usamos a senha `MakeYourChoice` ao executar o programa `su` para obter privilégios de root na máquina.

```
john@sawctf:~$ find / -perm -4000 -type f 2>/dev/null
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/su
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/umount
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
john@sawctf:~$ sudo su
[sudo] password for john:
john is not in the sudoers file.
john@sawctf:~$ su
Password:
root@sawctf:/home/john# ls
```

Navegamos até o diretório `/root` e encontramos o código

```
53556468646d565a6233567954476c6d5a55465164584a7762334e6c .
```

```
root@sawctf:/# cd root
root@sawctf:~# ls
next_game.txt  root.txt
root@sawctf:~# cat root.txt
flag: 53556468646d565a6233567954476c6d5a55465164584a7762334e6c
```

Usamos, novamente, o CyberChef para decodificá-lo e encontrar a flag

`IGaveYourLifeAPurpose`.

The screenshot shows the CyberChef interface with the following configuration:

- Recipe:** From Hex
- Input:** 53556468646d565a6233567954476c6d5a55465164584a7762334e6c
- From Base64:** Alphabet: A-Za-z0-9+=, Remove non-alphabet chars (checked), Strict mode (unchecked)
- Show Base64 offsets:** RBC 56, 1
- Output:** IGiveYourLifeAPurpose

Task 5

Para concluir a Task 5, bastava encontrar o link para a próxima sala, o qual pode ser encontrado no arquivo `next_game.txt` no diretório `/root`. Link:

<https://tryhackme.com/jr/pr0m3th3us>.

```
root@sawctf:~# cat next_game.txt
```



Next game in: <https://tryhackme.com/jr/pr0m3th3us>
Make Your Choice!