picoCTF2025-Writeup

kindun

2025 年 3 月 21 日

1 初めに

本記事の内容は筆者の理解に基づいており, 誤りが含まれる可能性があります.

2 Cookie Monster Secret Recipe[Web Exploitation]

2.1 問題分

Cookie Monster has hidden his top-secret cookie recipe somewhere on his website. As an aspiring cookie detective, your mission is to uncover this delectable secret. Can you outsmart Cookie Monster and find the hidden recipe? You can access the Cookie Monster here and good luck

2.2 解法

2.2.1 cookie に着目

1. **ctrl+shift+i で開発者ツール (呼び名色々) を開く**
2. **上タブの Storage をクリック**
3. **左タブの Cokkies をクリック → すぐ下の URL をクリック**
   右側になにか出ていたら次の節に移動
4. **開発者ツールは閉じずに適当に Username と Password を入力し Login する**

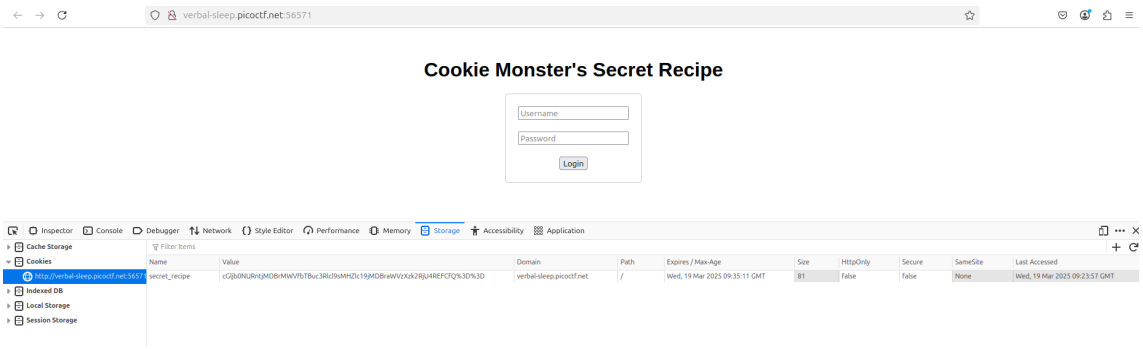

図 1 cookie

2.2.2 URL をデコード by CyberChef

1. **文字列を確認**

   Name が secret __ recipe とあることから flag だと分かる.

   問題文的にも cookie になにかあることが推測できる.

   しかし, このままではいけなさそう.picoCTF の形式ではない

2. **デコード**

   文字列の中に%があるので URL でエンコードされている可能性がある.

   とりあえず CyberChef の Magic を使ってみる.

   だめだったので次に URL Decode を使ってみる.

   デコードができた. やはり URL でエンコードされていた.

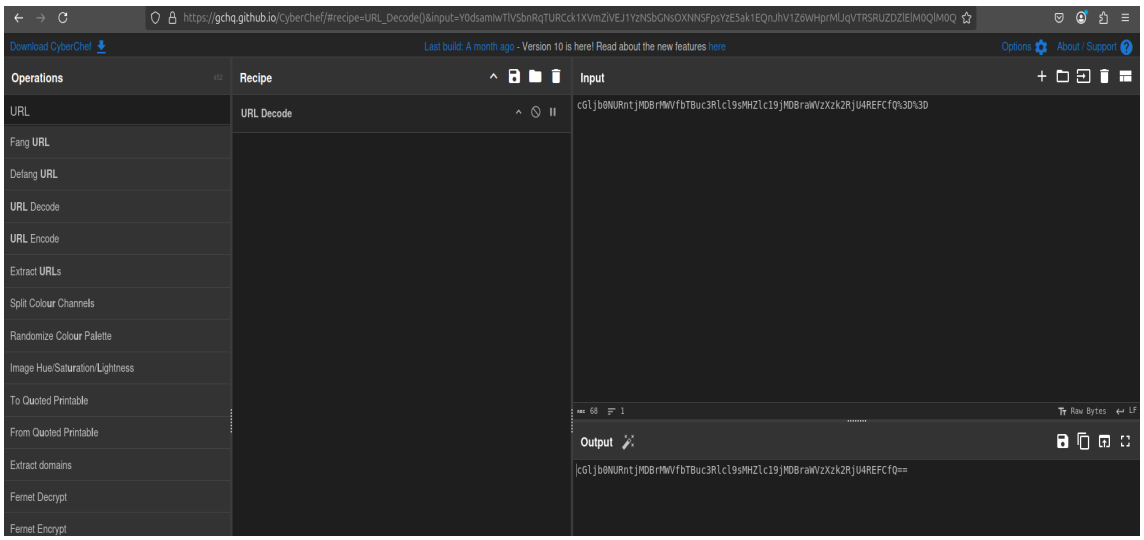

図 2　URL_decode

2.2.3 base64 をデコード by CyberChef

1. **とりあえず,Cyberchef の Magic を使う.**

   base64 でエンコードされていたことが分かる.

   デコードされた結果が出ていて, それが flag だと分かる.

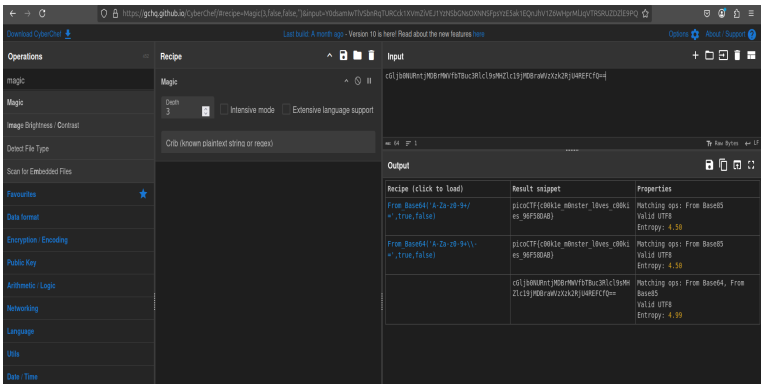

図 3　base64_decode

3　PIE TIME[Binary Exploitation]

3.1　問題文

Can you try to get the flag? Beware we have PIE! Additional details will be available after launching your challenge instance.

3.2　解法

3.2.1　問題の全体図を捉える

1. **Launch Instance をクリックし,nc コマンドで実行**

```
$nc rescued-float.picoctf.net ○○
Address of main: 0x55f9a559433d
Enter the address to jump to, ex => 0x12345:
```

main のアドレスが書いてあり, 入力するとアドレスがジャンプするらしい

2. **とりあえず, プログラムをダウンロードし, 見てみる (一応 binary ファイルも)** 中身はこんな感じ

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <signal.h>
4  #include <unistd.h>
5
6  void segfault_handler() {
7    printf("Segfault␣Occurred,␣incorrect␣address.\n");
8    exit(0);
9  }
10
11 int win() {
12   FILE *fptr;
13   char c;
14
```

```
15    printf("You won!\n");
16    // Open file
17    fptr = fopen("flag.txt", "r");
18    if (fptr == NULL)
19    {
20        printf("Cannot open file.\n");
21        exit(0);
22    }
23
24    // Read contents from file
25    c = fgetc(fptr);
26    while (c != EOF)
27    {
28        printf ("%c", c);
29        c = fgetc(fptr);
30    }
31
32    printf("\n");
33    fclose(fptr);
34 }
35
36 int main() {
37    signal(SIGSEGV, segfault_handler);
38    setvbuf(stdout, NULL, _IONBF, 0); // _IONBF = Unbuffered
39
40    printf("Address of main: %p\n", &main);
41
42    unsigned long val;
43    printf("Enter the address to jump to, ex => 0x12345: ");
44    scanf("%lx", &val);
45    printf("Your input: %lx\n", val);
46
47    void (*foo)(void) = (void (*)())val;
48    foo();
49 }
```

見てみると.win 関数を実行できれば flag.txt が見れることが分かる.

3. **44,47,48 行のプログラムに注目**

```
1        void (*foo)(void) = (void (*)())val;
```

void 型 `foo` という関数ポインタに入力値 `val` を代入している.

```
1        foo();
```

よって, 入力したアドレスの関数が実行されるプログラムであるとわかった.

4. main 関数と win 関数のアドレスを知りたいダウンロードした ELF ファイル (vuln) にいろいろ書いてあるのでコマンドを使って見てみる.

```
$ vuln | grep -w -e "win" -w -e "main"
000000000000133d T main
00000000000012a7 T win
```

`-w` は完全一致か,`-e` は複数検索のときに使う.

5. `I want to jump`
   main 関数のアドレスが毎回変わってしまうが main 関数と win 関数のアドレスの差は変化していないので (何回も nc 接続したらわかる), アドレス差さえわかれば win 関数のアドレスがわかるということだ.
6. **差を求める**
   さまざまなツールがあるが, とりあえず,10 進数に直して計算すると差が 150(10),0x96(16) だと分かった.
7. **入力**
   nc 接続をして main のアドレスが表示されるので, そこから-0x96(16) のアドレスを求め, 入力すると flag が出てくる

4 hashcrack[Cryptography]

4.1 問題文

A company stored a secret message on a server which got breached due to the admin using weakly hashed passwords. Can you gain access to the secret stored within the server? Access the server using nc verbal-sleep.picoctf.net 57356

4.2 解法

1. `hash` **をとりあえず検索**

```
$ nc verbal-sleep.picoctf.net 57356
Welcome!! Looking For the Secret?

We have identified a hash: 482c811da5d5b4bc6d497ffa98491e38
Enter the password for identified hash:
```

hash をとりあえず検索してみると

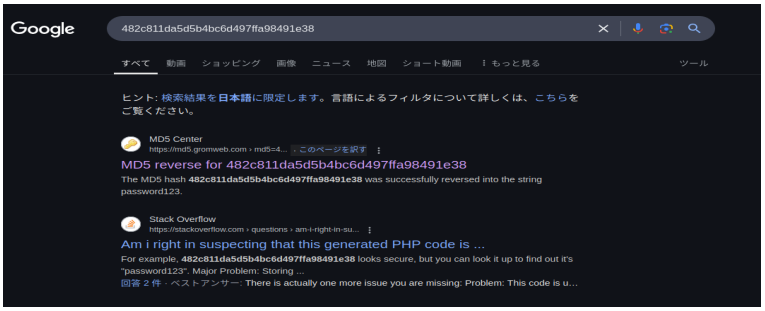

図 4　hash_serch

このように MD5 と書かれており, ついでに decode もできそうなのでやって, それを入力するとクリア

2. **次も同じように**
3. SHA256
   検索しても出ない人が多いかもしれません. しかし,SHA256 **デコード**と調べると変換ツールが出ると思うので, そちらで行いましょう

5 ph4nt0m 1ntrud3r[Forensics]

5.1 問題文

A digital ghost has breached my defenses, and my sensitive data has been stolen! Your mission is to uncover how this phantom intruder infiltrated my system and retrieve the hidden flag. To solve this challenge, you'll need to analyze the provided PCAP file and track down the attack method. The attacker has cleverly concealed his moves in well timely manner. Dive into the network traffic, apply the right filters and show off your forensic prowess and unmask the digital intruder! Find the PCAP file here Network Traffic PCAP file and try to get the flag.

5.2 解法

1. tshark **を使ってみる.(**wireshark **のコマンドライン版)**
   とりあえず, ファイルの中身を見てみる.

```
$ tshark -r myNetworkTraffic.pcap
1   0.000000  192.168.0.2 → 192.168.1.2  TCP 48 20 →
     80 [SYN] Seq=0 Win=8192 Len=8 [TCP segment of a reassembled PDU]
2   0.001763  192.168.0.2 →
     192.168.1.2  TCP 52 [TCP Out-Of-Order] [Illegal Segments]
3   0.002804  192.168.0.2 → 192.168.1.2  TCP 44 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=4
4  -0.000425  192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=8
5   0.002560  192.168.0.2 → 192.168.1.2  TCP 52 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=12
6  -0.001852  192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=8
7  -0.001604  192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=8
8   0.000751  192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=8
9   0.002115  192.168.0.2 → 192.168.1.2  TCP 52 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=12
10   0.001522  192.168.0.2 → 192.168.1.2  TCP 52 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=12
11   0.001001  192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=8
12   0.001229  192.168.0.2 → 192.168.1.2  TCP 52 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=12
13  -0.000655  192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
     80 [SYN] Seq=0 Win=8192 Len=8
```

```
14    0.000289   192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
      80 [SYN] Seq=0 Win=8192 Len=8
15   -0.003030   192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
      80 [SYN] Seq=0 Win=8192 Len=8
16   -0.002336   192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
      80 [SYN] Seq=0 Win=8192 Len=8
17   -0.000931   192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
      80 [SYN] Seq=0 Win=8192 Len=8
18   -0.002580   192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
      80 [SYN] Seq=0 Win=8192 Len=8
19   -0.002085   192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
      80 [SYN] Seq=0 Win=8192 Len=8
20   -0.001378   192.168.0.2 → 192.168.1.2  TCP 48 [TCP Retransmission] 20 →
      80 [SYN] Seq=0 Win=8192 Len=8
21    0.002336   192.168.0.2 → 192.168.1.2  TCP 52 [TCP Retransmission] 20 →
      80 [SYN] Seq=0 Win=8192 Len=12
```

2. **時間に注目**

   問題文より, 時間が重要そうなので frame.time, そして Len=で payload になにかあることが分かるので, それも表示してみる.

   ついでに, 時間について sort を行う.

   tcp.segment_data は tcp.payload と一緒

```
$ tshark -r ../myNetworkTraffic.pcap -T fields -e frame.time -e tcp.segment_data |
    sort -k4
Mar  6, 2025 12:32:17.183957000 JST     6f6b4b544272383d
Mar  6, 2025 12:32:17.184407000 JST     7938382f7064413d
Mar  6, 2025 12:32:17.184651000 JST     5a44516b5033553d
Mar  6, 2025 12:32:17.184902000 JST     4d77624d5971513d
Mar  6, 2025 12:32:17.185135000 JST     4b695a502b75413d
Mar  6, 2025 12:32:17.185383000 JST     6d554b6c3471343d
Mar  6, 2025 12:32:17.185609000 JST     2b5256384e56593d
Mar  6, 2025 12:32:17.186056000 JST     5542635a5079593d
Mar  6, 2025 12:32:17.186332000 JST     554a64644e6a343d
Mar  6, 2025 12:32:17.186562000 JST     2b5448325269413d
Mar  6, 2025 12:32:17.186987000 JST     3677446f5438383d
Mar  6, 2025 12:32:17.187276000 JST     75316e376157673d
Mar  6, 2025 12:32:17.187506000 JST     6c7746703577303d
Mar  6, 2025 12:32:17.187738000 JST     42627a516731303d
Mar  6, 2025 12:32:17.187988000 JST     516a3961744d593d
Mar  6, 2025 12:32:17.188216000 JST     63476c6a62304e5552673d3d
Mar  6, 2025 12:32:17.188509000 JST     657a463058336330637 73d3d
Mar  6, 2025 12:32:17.188750000 JST     626e526664476730644 13d3d
Mar  6, 2025 12:32:17.189102000 JST     587a4d3063336c6664413d3d
Mar  6, 2025 12:32:17.189323000 JST     596d68664e484a664d673d3d
Mar  6, 2025 12:32:17.189547000 JST     5a54466d5a6a41324d773d3d
Mar  6, 2025 12:32:17.189791000 JST     66513d3d
```

3. payload を 16 進数にする.

   まず,plan テキストに戻してから,16 進数に直す. $6 は 6 列目という意味

```
tshark -r ../myNetworkTraffic.pcap -T fields -e frame.time -e tcp.segment_data |
    sort -k4 | awk '{printf($6)}' | xxd -p -r
```

```
okKTBr8=y88/pdA=ZDQkP3U=MwbMYqQ=KiZP+uA=mUK14q4=+RV8NVY=UBcZPyY=UJddNj4=+TH2RiA=6
    wDoT88=u1n7aWg=lwFp5w0=BbzQg10=Qj9atMY=cGljb0NURg==ezF0X3c0cw==bnRfdGg0dA==
    XzM0c3lfdA==YmhfNHJfMg==ZTFmZjA2Mw==fQ==
```

4. base64 じゃねえか
   でてきた文字列が base64 っぽいので, デコードする.

```
 tshark -r ../myNetworkTraffic.pcap -T fields -e frame.time -e tcp.segment_data
| sort -k4 | awk '{printf($6)}' | xxd -p -r | base64 -d
 ^^ef^^bf^^bd^^d0^^83]B?Z^^ef^^bf^^bd^^ef^^bf^^bdpicoCTF{1t_w4snt_th4t_34sy_tbh
_4r_2e1ff063}^^ef^^bf^^bdi^^ef^^bf^^bd
```

   無事に flag が出ました！！！！！！！

5. 補足

cmd	オプション	説明
tshark	-r	ファイルの読み込み
〃	-T	format の指定
〃	-e	field の選択
sort	-kN	N 列目を基準に sort(標準は昇順)
xxd	-p	plain テキストに変換
〃	-r	16 進数に変換（戻す）
base64	-d	decode

6 おわりに

ご覧いただきありがとうございました.