第 3 章

CTF 超入門

75 回生 sashiming

3.1 はじめに

sashiming です。気づけば高 2 で、老いを感じている次第です。

普段は競技プログラミング (略称:競プロ) をたしなんでいるのですが、最近はモチベが上がらなくなってきたので、「CTF」という競技を始めてみました。CTF は競プロよりも敷居が高いので、はじめに何をするべきか分からない人が多いと思います。そこで、この記事では CTF を最低限楽しむための基礎知識を書いていこうと思います。

3.2 注意

この部誌では、OS に Windows10 を用いています。他の OS を使用しているかたは、適宜読み替えるなどしてください。

3.3 What's CTF

CTF は "Capture the Flag" の略で、サイバーセキュリティなどに関する技術や知識を活用して情報を抜き取る競技です。ハッカーのようなことをすると言っても差し支えはないでしょう。

CTF には主に 2 つの形式があります。

- **クイズ形式 (Jeopardy style)**[*1]：独立した問題がたくさん与えられるので、与えられたプログラム・Web サイト・暗号・画像などから "flag" と呼ばれる文字列を探す
- **攻防戦形式 (Attack/Defense style)**：各チームにサーバが与えられ、他のチームのサーバを攻撃して flag を奪取しながら自分のチームのサーバを防衛する

基本的に、オンラインで開かれる CTF コンテストはクイズ形式です。この記事では、クイズ形式のコンテストで出される問題について書きます。

クイズ形式で出される問題にはいくつかジャンルがあります。

- **Reversing**：与えられたバイナリを解析して flag を探す
- **Pwn**：与えられたプログラムの脆弱性を突いて flag を奪う
- **Web**：web サイトの脆弱性を突いて flag を奪う
- **Crypto**：暗号を解いて flag を見つける
- **Forensics/Steganography**：ディスクイメージや画像・音声などを解析して flag を探す
- **Misc**：その他・雑学

[*1] 「Jeopardy!」というアメリカのテレビ番組が由来です

3.4 例題

ここで、簡単な問題を紹介します。ジャンルは Crypto です。

暗号文：`qdgd_vfkrro_ihvwlydo_nqrfn_dqg_glyh`

わけがわからない文字列ですが、勘の良いかたは何となく見当がつくかもしれません。

答えを言ってしまうと、これは**シーザー暗号**と呼ばれる、最も有名かつ古典的な暗号です。仕組みはごく単純で、それぞれのアルファベットを 3 文字ぶん後にずらしているだけです。例えば、a は d に、e は h に、y は b に変換されます。

平文 (もとの文) を 3 文字ぶん後にずらして暗号化しているので、復号するには逆に暗号文を 3 文字ぶん前にずらすだけで良いです。先ほどの暗号文を復号すると、次のようになります。

復号後：`nada_school_festival_knock_and_dive`

ちゃんと意味のわかる文字列になりましたね！

3.5 CTF の学習方法

CTF のコンテストが開かれる Web サイトは、コンテスト終了後閉鎖されてしまうものが多いですが、中には常に問題が公開されている「常設 CTF」があります。

以下に入門レベルのかたでも取り組みやすい常設 CTF を紹介しておきます。

- CpawCTF (https://ctf.cpaw.site/)
  - CpawCTF2 (https://ctf2.cpaw.site/)
- picoCTF (https://picoctf.org/)

常設 CTF はたくさんネット上に転がっているので (Pwn 系は特に多いです)、他にも見てみたい人はネットで調べてみると良いでしょう。

問題を解いていく中で、戦略に行き詰まることが必ずあるはずです。そんなときは迷わず解説を見ましょう。名が知られているコンテストでは、公式に解説がなくてもたいてい有志のかたが解法をブログなどに公開してくれています。この解説のことを **Writeup** といいます。Writeup を読むことで他の人がどのように解いたのか参考にできるので、自力で解けた問題でも Writeup を見てみるのも良いでしょう。

それでは、次の章から本編です。初心者がとっつきにくい Pwn を除いた Crypto・Reversing・Web・Forensics について、例題を見ていきましょう。

3.6 Crypto 編

Crypto の問題では与えられた暗号を解いて flag を取得するのですが、この類の問題はセキュリティというよりは、数学や謎解きの性格が強いです。

早速ですが、例題を見てみましょう。

picoCTF 2021 - Mod 26

Cryptography can be easy, do you know what ROT13 is?

```
cvpbPGS{arkg_gvzr_V'yy_gel_2_ebhaqf_bs_ebg13_GYpXOHqX}
```

問題文に「ROT13 を知っていますか？ 僕は知っています」的なことが書いてあるので、適当に ROT13 でググってみましょう。CTF ではググる技術も問われます。

ROT13 は非常に有名な暗号 (?) で、平文の各アルファベットを後ろに 13 文字ぶん後ろにずらす、というものです。先ほど紹介したシーザー暗号のずらす個数を変えただけですね。

検索すると ROT13 を復号してくれるサイトが無限に出てくるので、適当なサイトで復号してみましょう。次の文字列になるはずです。

```
picoCTF{next_time_I'll_try_2_rounds_of_rot13_TLcKBUdK}
```

次の問題は、RSA 暗号という有名な暗号を用いた問題です。RSA 暗号については Chito 君が記事を書いてくれているようなので、彼に丸投げすることにします。興味のあるかたはご覧ください。

picoCTF 2021 - Mini RSA

What happens if you have a small exponent? There is a twist though, we padded the plaintext so that (M ** e) is just barely larger than N. Let's decrypt this: ciphertext

RSA 暗号の体裁をとっていても、その値が脆弱では意味がありません。今回の場合、$e = 3$ と小さいので、3 乗根をとれば平文が手に入ってしまいます。この攻撃を Low Public-Exponent Attack といったりします。

```python
import gmpy2
from Crypto.Util.number import *

N = int(input('N: '))
e = int(input('e: '))
c = int(input('c: '))

while True:
    m, exact = gmpy2.iroot(c, e)
    if exact:
        print(long_to_bytes(m))
        break
    c += N
```

c の 3 乗根が整数でとれるまで、c に N を足し続けています。RSA 暗号の定義から $c = m^e \bmod N$ なので、この 3 乗根が平文となります。これをバイト列と解釈して文字列に変換すると、

```
picoCTF{e_sh0u1d_b3_lArg3r_a166c1e3}
```

が手に入ります (運営側によって `lArg3r` の後の文字列はちょくちょく変更されます)。

他にも、N が等しく e が異なる 2 つの公開鍵 (e, N) が与えられれば、Common Modulus Attack で平文を入手することもできます。

3.7 Forensics 編

Forensics では、与えられたファイル (画像、ディスクイメージ、etc.) に隠された flag を見つけます。結構楽しいです。

簡単めな問題を見てみましょう。

picoCTF 2019 - extensions

This is a really weird text file TXT? Can you find the flag?

CTF では問題名がヒントになっていることが多々あります。今回は extensions、つまり拡張子なので、拡張子をいじる問題だと推測できます。

では、このファイルの正しい形式は何でしょうか。実はそれが分かるコマンドがあります。

Linux には `file` コマンドがあり、それを使うとファイル形式が表示されます。Windows10 では、WSL(Windows Subsystem for Linux) を使えば Windows 上で Linux を使うことができます。

```
$ file ./flag.txt
flag.txt: PNG image data, 1697 x 608, 8-bit/color RGB, non-interlaced
```

これでファイルが png 形式だと分かりました。あとは拡張子を.png に変更すると...

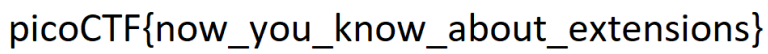

図 3.1: 答え

この問題で使った `file` コマンドと、ファイル中の表示可能な文字列を出力する `strings` コマンドは Forensics では非常によく使われます。初手はこのコマンドを使うとよいでしょう。

3.8 Reversing 編

Reversing では基本的にアセンブリを読む作業が必要です。知識、経験、エスパーなどを駆使してプログラムの挙動をつかまないといけません。結構根性が要ります。

アセンブリの読み方は「アセンブリ 入門」などと調べると結構な数がヒットするので、それを参考にするとよいでしょう (部誌で書くには締切が近すぎます)。Reversing や Pwn は高度な知識が必要なので、後回しにしておくとよいのかもしれません。

それでは参考がてらに、アセンブリを使った問題をひとつ。

picoCTF 2019 - asm1

What does asm1(0x6fa) return? Submit the flag as a hexadecimal value (starting with '0x'). NOTE: Your submission for this question will NOT be in the normal flag format.

```
asm1:
    <+0>:   push   ebp
    <+1>:   mov    ebp,esp
    <+3>:   cmp    DWORD PTR [ebp+0x8],0x3a2
    <+10>:  jg     0x512 <asm1+37>
    <+12>:  cmp    DWORD PTR [ebp+0x8],0x358
    <+19>:  jne    0x50a <asm1+29>
    <+21>:  mov    eax,DWORD PTR [ebp+0x8]
    <+24>:  add    eax,0x12
    <+27>:  jmp    0x529 <asm1+60>
    <+29>:  mov    eax,DWORD PTR [ebp+0x8]
    <+32>:  sub    eax,0x12
    <+35>:  jmp    0x529 <asm1+60>
    <+37>:  cmp    DWORD PTR [ebp+0x8],0x6fa
```

```
   <+44>:  jne    0x523 <asm1+54>
   <+46>:  mov    eax,DWORD PTR [ebp+0x8]
   <+49>:  sub    eax,0x12
   <+52>:  jmp    0x529 <asm1+60>
   <+54>:  mov    eax,DWORD PTR [ebp+0x8]
   <+57>:  add    eax,0x12
   <+60>:  pop    ebp
   <+61>:  ret
```

この関数では、アドレス [ebp+0x8] に引数が入っています。この引数を cmp 命令で比較し、その直後の jg や jne 命令で比較結果によって次に実行する命令にジャンプします。

行<+0>・<+1>はおまじないのようなものと思ってください。

<+3>で 0x6fa と 0x3a2 を比較します。0x6fa のほうが大きいので (jg)、<+37>にジャンプします。

次に、<+37>で 0x6fa と 0x6fa を比較します。値が等しいので、ジャンプはせずそのまま次の行に移ります。

<+46>で eax レジスタに値 0x6fa を入れています。その次の行<+49>で eax レジスタの値を 0x12 だけ引いています。つまりこの演算の後、eax レジスタの値は 0x6e8 になっています。

最後に、<+52>の処理で行<+60>にジャンプし、処理終了です。

関数の返り値は eax レジスタに入れることになっているので、答えは 0x6e8 です。

3.9 Web 編

プログラムを解読したり脆弱性を突いたりする Reversing や Pwn とは違い、Web では文字通り Web ページの脆弱性を突きます。このジャンルも Pwn 同様、難しい部類に入ります。

今回は割合に簡単な問題を紹介します。

picoCTF 2019 - where are the robots

Can you find the robots? https://jupiter.challenges.picoctf.org/problem/36474/ or http://jupiter.challenges.picoctf.org:36474

Web サーバを構築したことのあるかたは、問題文を読んでやるべきことが分かるかと思います。

robots.txt は、知られたくない情報を検索エンジンによって収集されないように、クローラー (Web 上の画像や文書を取得しデータベース化するプログラム) を制御するためのファイルです。このファイルはサイトのトップディレクトリに置かれています。

なので、URL 欄の末尾に robots.txt を追加すると...

```
User-agent: *
Disallow: /477ce.html
```

図 3.2: robots.txt

このように robots.txt の中身が表示されてしまいます。そして、あからさまに html ファイルのリンクが書いてあるので、アクセスすると flag が手に入ります。

Web 系の問題は学習難度が高く、なかなかとっつきにくいジャンルとなっています。

3.10 おわりに

今回は CTF の入門的問題ではどのようなものが出題されるか書きました。この記事が CTF を始める人に少しでも役に立てば嬉しいです。

それではよい CTF ライフを。