Another day, another IR...Please help us to understand this ransomware-sample. We were not able to get a sample of the payload when it ran but we got the original first stage. Please help us and find out how it works. We attached one of the encrypted files we need to decrypt urgently.

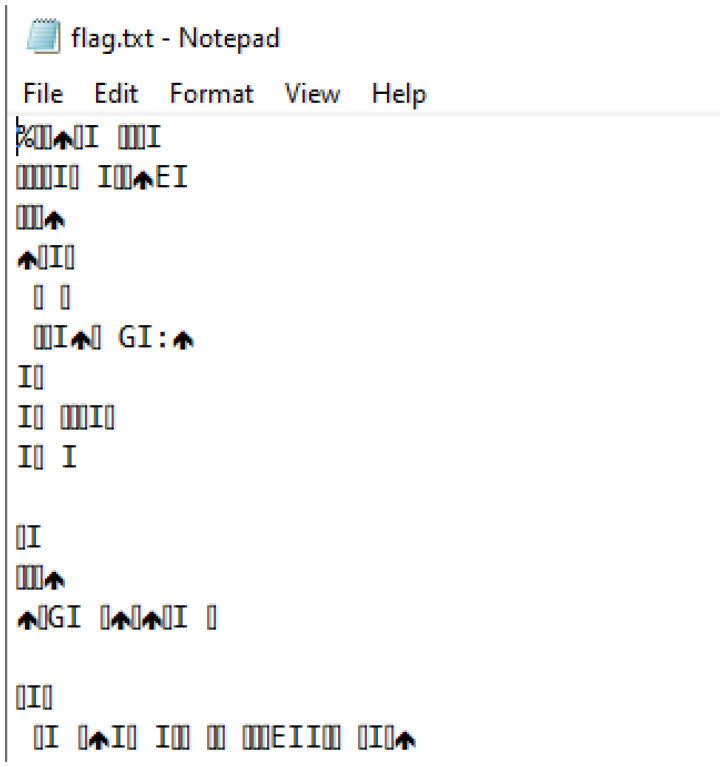
Task:

Find out how the ransomware works and get the flag!

PW: infected

---

1. Extract the file and inspect the flag.txt. It seems the file is completely encrypted



2. As first stage it seems we are dealing with a SFX-Archiv. Instead of running it directly we will decrypt it manually. To see what it is supposed to do we can use Winrar and simply open it and check the textbox on the right:
   The archive is likely being extracted to %temp% and will then run the file a.bat from %temp%. Instead of doing that we will inspect the batch-file first.

```
;The comment below contains SFX script commands

Path=%temp%
Setup=%temp%\a.bat
Silent=1
```

3. The batch-file is heavily obfuscated. Instead of deobfuscating it, we will create a snapshot and run it to see what happens. If you run it from another cmd-window we should be able to see what is going on
4. We can see that the batch is trying to delete all shadow copies, create a task to run the word-file and then will try to delete itself from %temp%. This means that we can ignore the batch-file and inspect the word-file

Developer Command Prompt for VS 2022

```
C:\Users\MalDevUser\Desktop\New folder\public\p.pdf>vssadmin delete shadows /all /quiet
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

Error: You don't have the correct permissions to run this command.  Please run this utility from a command
window that has elevated administrator privileges.


C:\Users\MalDevUser\Desktop\New folder\public\p.pdf>schtasks /create /tn "a" /tr "C:\Users\MALDEV~1\AppData\Local\Temp\p
asswords.docm" /sc minute /mo 1
SUCCESS: The scheduled task "a" has successfully been created.

C:\Users\MalDevUser\Desktop\New folder\public\p.pdf>del C:\Users\MALDEV~1\AppData\Local\Temp\a.bat
Could Not Find C:\Users\MALDEV~1\AppData\Local\Temp\a.bat

C:\Users\MalDevUser\Desktop\New folder\public\p.pdf>
```

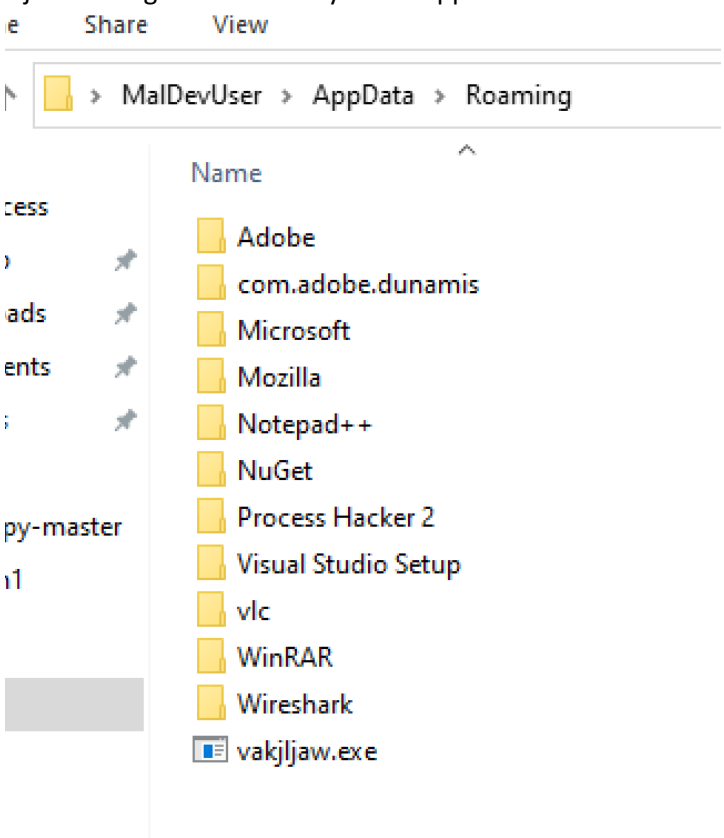
5. Trying to inspect the macros fails as they are likely modified. We can try to restore the function and unhide the macros with EvilClippy.

```
C:\Users\MalDevUser>cd C:\Users\MalDevUser\Downloads\EvilClippy-master\EvilClippy-master

C:\Users\MalDevUser\Downloads\EvilClippy-master\EvilClippy-master>EvilClippy.exe -gg p.docm
Unhiding module: NewMacros

C:\Users\MalDevUser\Downloads\EvilClippy-master\EvilClippy-master>
```

6. We the macros now visible we can inspect them. The interesting macro here is “aaa”. After a bit of Anti-Debug we can see a “call shell” to a s.js in %temp%. Instead of running the script we can delete this line and then inspect the written script in %temp%

```
        Dim scriptContent As String
        scriptContent = GetScriptFromWordDocument

        Dim filePath As String
        filePath = Environ("TEMP") & "\s.js"

        SaveStringToFile scriptContent, filePath

        Call Shell("wscript.exe " & filePath, vbNormalFocus)

    End Sub
```

7. The s.js is a very long jscript and seems to be doing something with a base64-string and then running another executable. You could either take the base64-string and try to save it as binary or run it and see what is going on. Inspecting the operation with procmon we can see that the s.js is writing another binary into %appdata%



8. With IDA we can see that this file is trying to read the file “flag.txt” from the current directory (%appdata%)

```
lea     rcx, unk_1400210A2
call    j___CheckForDebuggerJustMyCode
lea     rax, aFlagTxt    ; "flag.txt"
mov     [rbp+950h+lpFileName], rax
lea     rax, aFlagEncryptedT ; "flag_encrypted.txt"
mov     [rbp+950h+var_348], rax
mov     [rsp+990h+hTemplateFile], 0 ; hTemplateFile
mov     [rsp+990h+dwFlagsAndAttributes], 80h ; dwFlagsAndAttributes
mov     [rsp+990h+dwCreationDisposition], 3 ; dwCreationDisposition
xor     r9d, r9d         ; lpSecurityAttributes
xor     r8d, r8d         ; dwShareMode
mov     edx, 80000000h   ; dwDesiredAccess
mov     rcx, [rbp+950h+lpFileName] ; lpFileName
call    cs:CreateFileA
mov     [rbp+950h+hFile], rax
cmp     [rbp+950h+hFile], 0FFFFFFFFFFFFFFFFh
jnz     short loc_14001185F
```

9. If this works it will later on write into the file “flag_encrypted.txt” into the same directory. That means that if we put the flag.txt into this directory It will likely try to encrypt the file.

```
loc_1400118EE:           ; hTemplateFile
mov     [rsp+990h+hTemplateFile], 0
mov     [rsp+990h+dwFlagsAndAttributes], 80h ; dwFlagsAndAttributes
mov     [rsp+990h+dwCreationDisposition], 2 ; dwCreationDisposition
xor     r9d, r9d         ; lpSecurityAttributes
xor     r8d, r8d         ; dwShareMode
mov     edx, 40000000h   ; dwDesiredAccess
mov     rcx, [rbp+950h+var_348] ; lpFileName
call    cs:CreateFileA
mov     [rbp+950h+hObject], rax
cmp     [rbp+950h+hObject], 0FFFFFFFFFFFFFFFFh
jnz     short loc_140011931
```

10. Looking at the xor-encryption we can see that attacker made a mistake by only using xor-encryption. This allows us to simply drop our encrypted file and get it decrypted again.

```
mov     rax, [rbp+950h+var_328]
movzx   eax, [rbp+rax+950h+Buffer]
xor     eax, 69h
mov     rcx, [rbp+950h+var_328]
mov     [rbp+rcx+950h+Buffer], al
jmp     short loc_1400118B2
```

11. Dropping the flag.txt into the same folder and running the binary will result in the flag in flag_encrypted.txt

WinRAR	7/18/2023 6:23 AM	File folder
Wireshark	10/27/2023 1:07 AM	File folder
flag.txt	10/28/2023 10:57 PM	Text Document
flag_encrypted.txt	11/1/2023 6:16 AM	Text Document
vakjljaw.exe	11/1/2023 6:07 AM	Application

```
ıd

lelp
ıam finibus. DS-CTF{D4mn, l00ks lik3 I 4m n07 g37ting p41d}. Duis auct

:nteger posuere sem non lectus dapibus bibendum. Quisque eu nisi vel j
```