BUGS IN MALWARE – UNCOVERING VULNERABILITIES FOUND IN MALWARE PAYLOADS

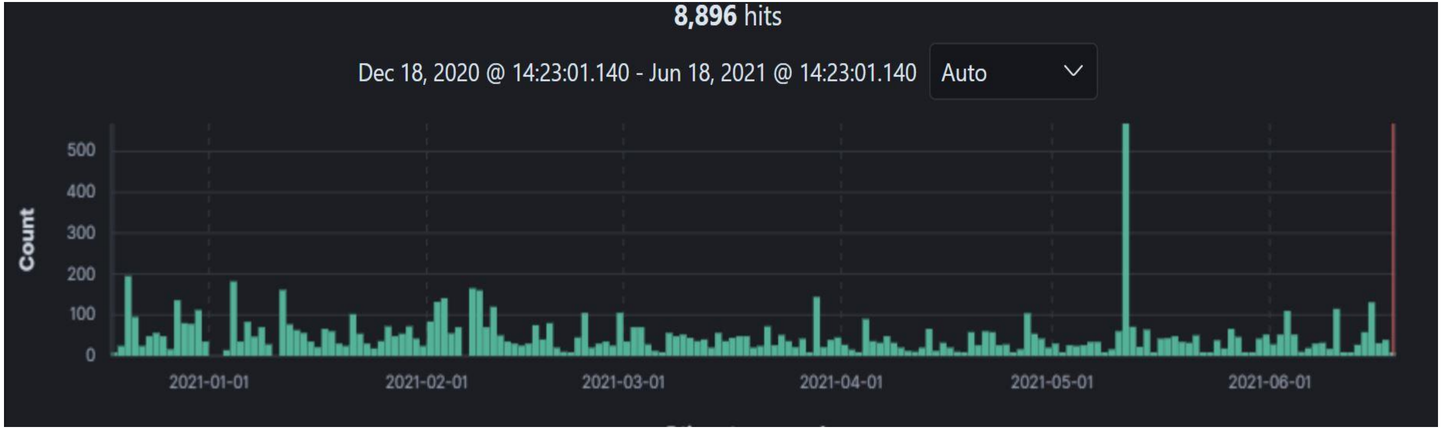
Nirmal Singh,
Uday Pratap Singh
ThreatLabZ

Agenda

- Introduction
- Study approach
- Look at case studies

Introduction

- Malware authors often take advantage of vulnerabilities in popular software.
- A lot of research on anti-VM and anti-sandbox techniques and techniques for bypassing AV product.
- Malware is also prone to bugs and coding errors which can cause it to crash or which can serve as backdoors for whitehats.



8,896 hits

Dec 18, 2020 @ 14:23:01.140 - Jun 18, 2021 @ 14:23:01.140 Auto

- Such bugs can often persist in a family for a long time.

Introduction

- **The purpose of this research is threefold:**

- **To look at what type of vulnerabilities exist in some of the prevalent malware families.**

- **To discuss the use of these bugs/vulnerabilities in preventing malware infection.**

- **To find out whether these are real vulnerabilities/coding errors or escape mechanisms.**

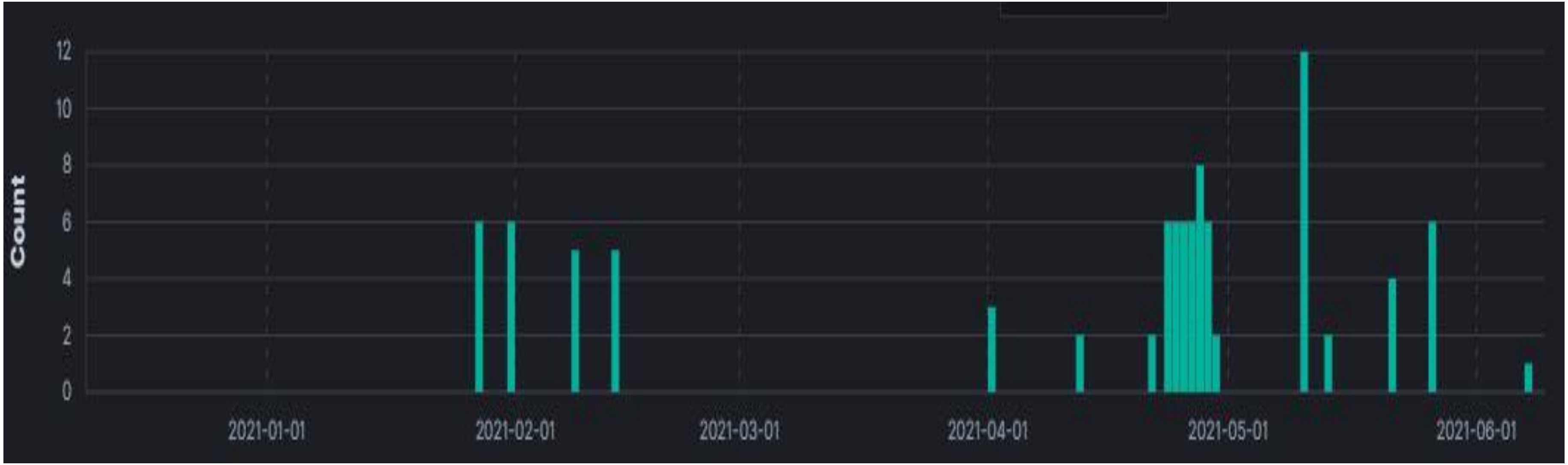
Study Approach

- **Large-scale analysis on a data set of malicious samples collected from the Zscaler Cloud Sandbox based on a few behaviour signatures**
- **Malware samples collected from 2019 to March 2021 in the Zscaler Cloud**
- **Clustering of samples using behavioral similarities**
- **MITRE’s Common Weakness Enumeration (CWE) system used to categorize malware bugs.**

Case Study #1

WIN32.PWS.VIDAR MULTIPLE BUGS IN THE CODE

- **Steals information and cryptocurrency from infected users**
- **Vidar can also scrape an Impressive selection of digital wallets**
- **In the Zscaler Cloud Sandbox, we found 94 samples showing execution errors.**

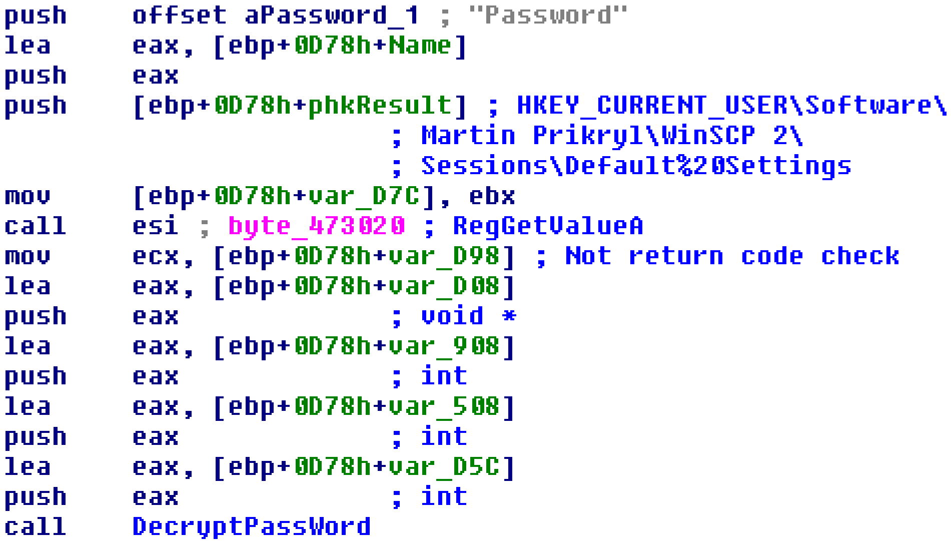


Case Study #1

WIN32.PWS.VIDAR MULTIPLE BUGS IN THE CODE

- **Bug 1: Incorrect check of function return value**

- **This bug is about calling an API and performing an operation without validating the output of that API call**



```
push    offset aPassword_1 ; "Password"
lea     eax, [ebp+0D78h+Name]
push    eax
push    [ebp+0D78h+phkResult] ; HKEY_CURRENT_USER\Software\
                        ; Martin Prikryl\WinSCP 2\
                        ; Sessions\Default%20Settings
mov     [ebp+0D78h+var_D7C], ebx
call    esi ; byte_473020 ; RegGetValueA
mov     ecx, [ebp+0D78h+var_D98] ; Not return code check
lea     eax, [ebp+0D78h+var_D08]
push    eax             ; void *
lea     eax, [ebp+0D78h+var_908]
push    eax             ; int
lea     eax, [ebp+0D78h+var_508]
push    eax             ; int
lea     eax, [ebp+0D78h+var_D5C]
push    eax             ; int
call    DecryptPassWord
```

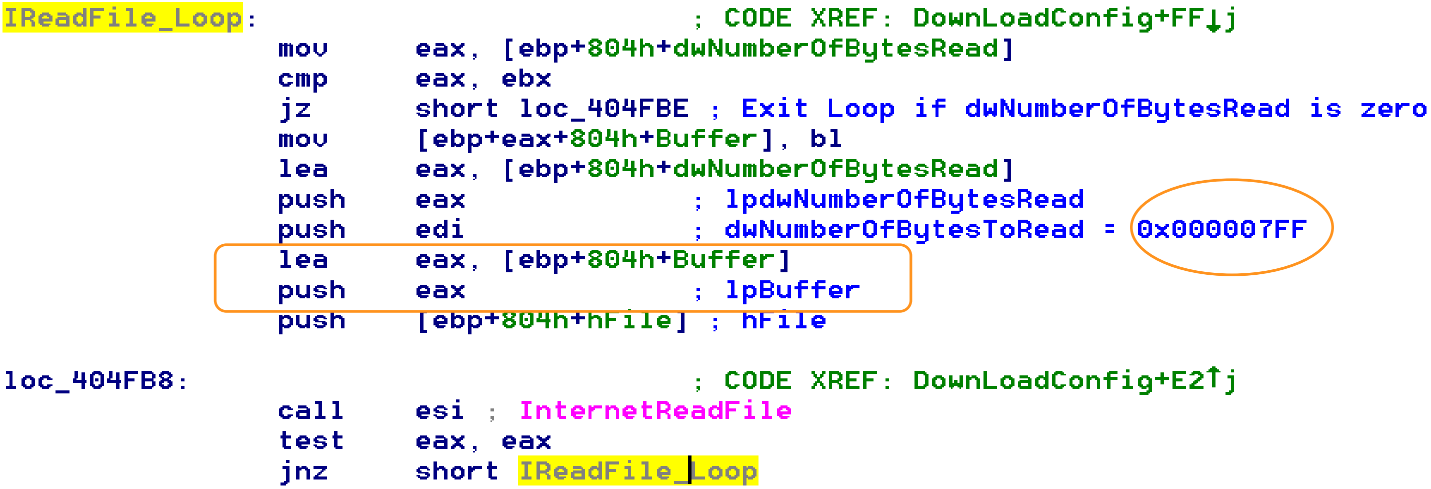
- **This bug is part of CWE-253 and it has consequences such as unexpected state, DoS, crash, exit, or restart of the system.**

Case Study #1

WIN32.PWS.VIDAR MULTIPLE BUGS IN THE CODE

- **Bug 2: Common buffer used by an API to perform multiple tasks & out-of-bounds write**

- **Downloads config files from the C&C using the InternetReadFile**



```
IReadFile_Loop:                                         ; CODE XREF: DownLoadConfig+FF↓j
                mov     eax, [ebp+804h+dwNumberOfBytesRead]
                cmp     eax, ebx
                jz      short loc_404FBE ; Exit Loop if dwNumberOfBytesRead is zero
                mov     [ebp+eax+804h+Buffer], bl
                lea     eax, [ebp+804h+dwNumberOfBytesRead]
                push    eax             ; lpdwNumberOfBytesRead
                push    edi             ; dwNumberOfBytesToRead = 0x000007FF
                lea     eax, [ebp+804h+Buffer]
                push    eax             ; lpBuffer
                push    [ebp+804h+hFile] ; hFile

loc_404FB8:                                             ; CODE XREF: DownLoadConfig+E2↑j
                call    esi ; InternetReadFile
                test    eax, eax
                jnz     short IReadFile_Loop
```

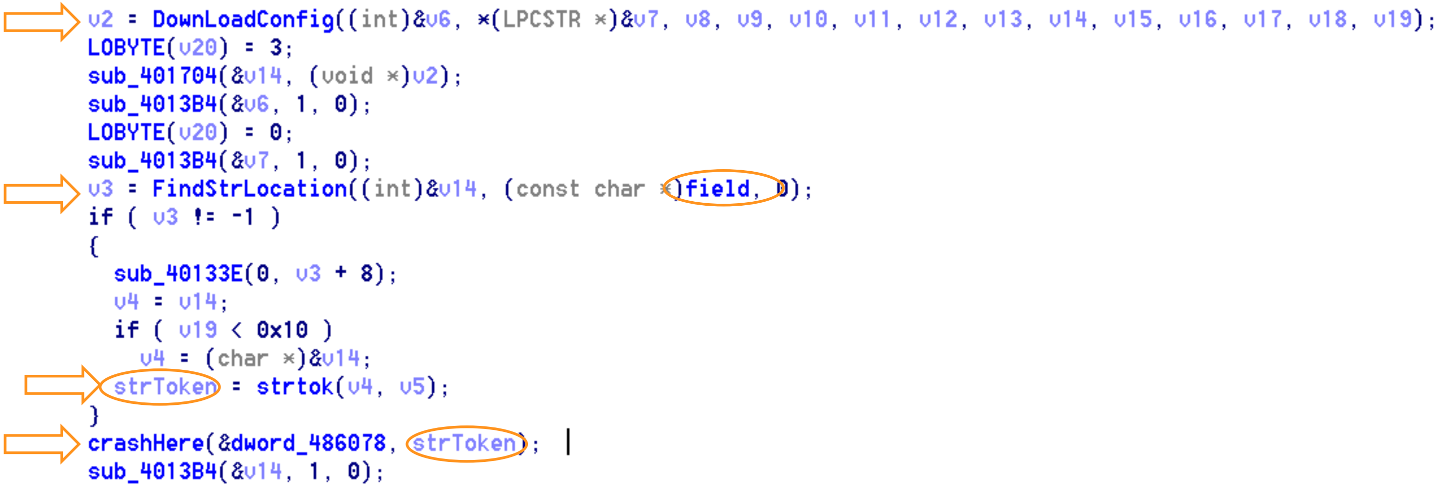
- **This bug is a classic case of CWE-787 where malware writes data past the end of the buffer, which results in the corruption of data, a crash, or code execution.**

Case Study #1

WIN32.PWS.VIDAR MULTIPLE BUGS IN THE CODE

- **Bug 3: Detection of absent string in configuration without any action**

- **Sample crashes it if it’s not able to download data from the C&C or if it’s not able to find a specific string (‘about’) in the downloaded data.**



```
v2 = DownLoadConfig((int)&v6, *(LPCSTR *)&v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19);
LOBYTE(v20) = 3;
sub_401704(&v14, (void *)v2);
sub_4013B4(&v6, 1, 0);
LOBYTE(v20) = 0;
sub_4013B4(&v7, 1, 0);
v3 = FindStrLocation((int)&v14, (const char *)field, 0);
if ( v3 != -1 )
{
  sub_40133E(0, v3 + 8);
  v4 = v14;
  if ( v19 < 0x10 )
    v4 = (char *)&v14;
  strToken = strtok(v4, v5);
}
crashHere(&dword_486078, strToken);
sub_4013B4(&v14, 1, 0);
```

- **Example of CWE-390, where the malware detects an error but doesn’t perform any action to prevent the consequences of the error, which may result in sample crashing.**

Case Study #2

INCORRECT CALCULATION OF BUFFER SIZE

- **WIN32.DOWNLOADER.RUGMI is a downloader which has been seen downloading RATs, e.g. Remcos, and other malware.**

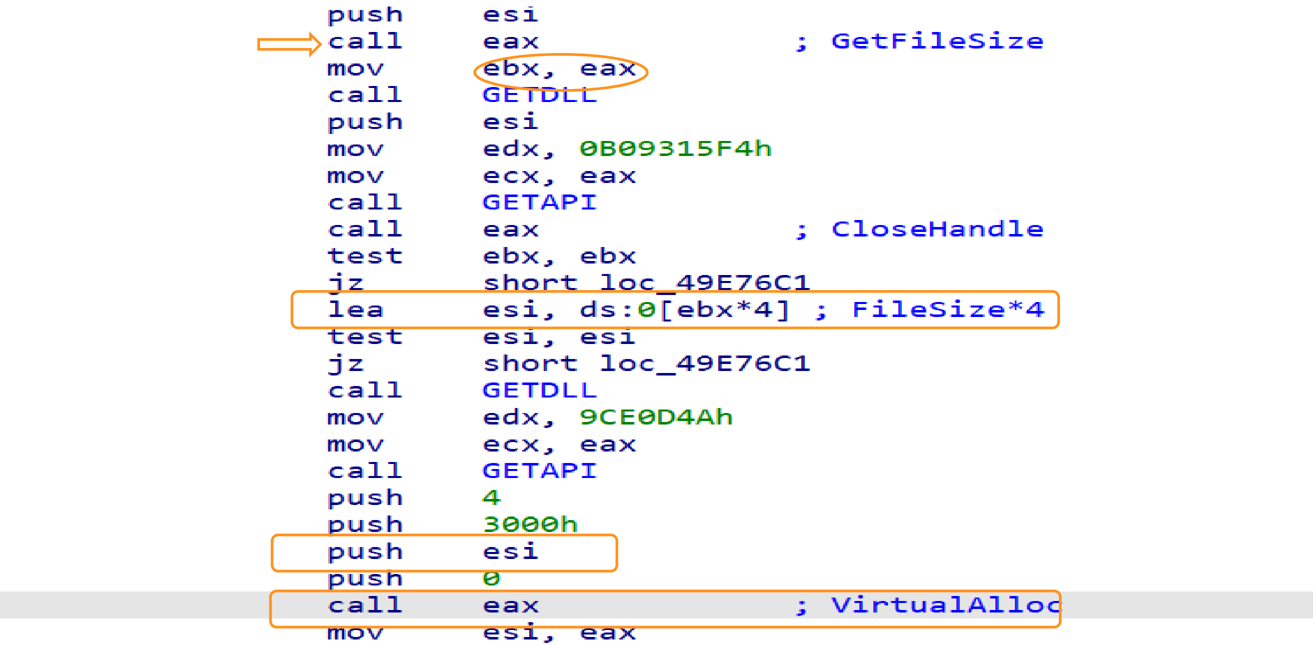
- **Found 17 samples of this malware showing execution errors during a campaign that was active from February to March 2021.**

- **Downloads a PNG file from i[.]imgur[.]com, which contains configuration data and a payload file.**

Case Study #2

INCORRECT CALCULATION OF BUFFER SIZE

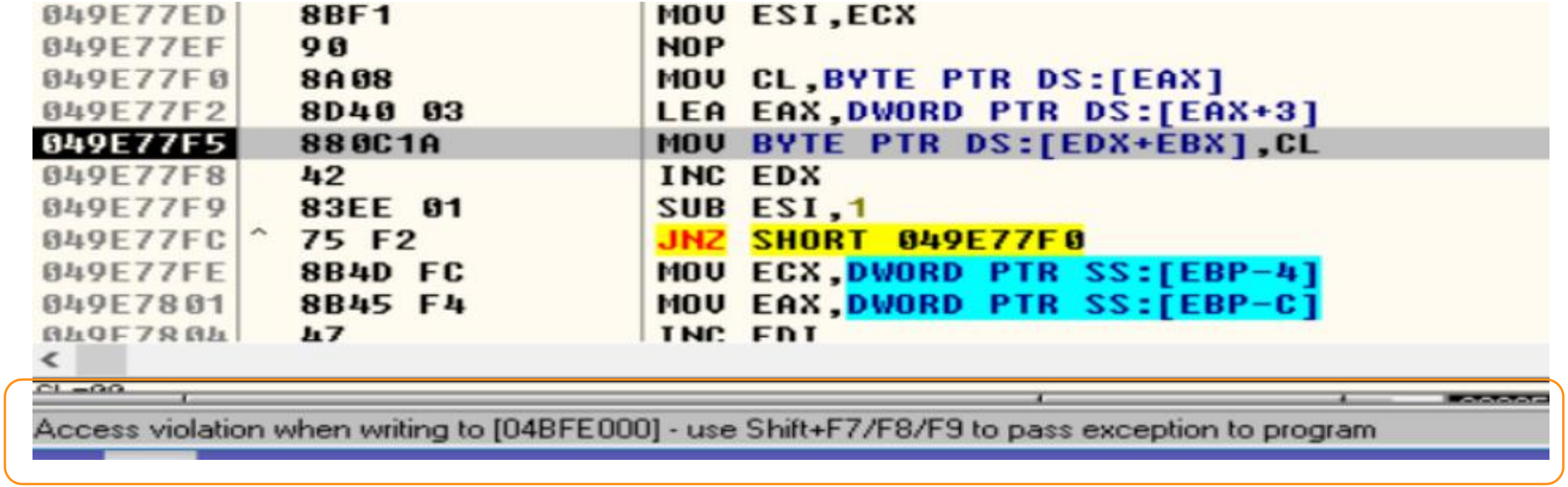
- **The decryption logic assumes that the size of the uncompressed data will be four times the size of the file, so it allocates memory according to that .**



```
push    esi
call    eax                     ; GetFileSize
mov     ebx, eax
call    GETDLL
push    esi
mov     edx, 0B09315F4h
mov     ecx, eax
call    GETAPI
call    eax                     ; CloseHandle
test    ebx, ebx
jz      short loc_49E76C1
lea     esi, ds:0[ebx*4] ; FileSize*4
test    esi, esi
jz      short loc_49E76C1
call    GETDLL
mov     edx, 9CE0D4Ah
mov     ecx, eax
call    GETAPI
push    4
push    3000h
push    esi
push    0
call    eax                     ; VirtualAlloc
mov     esi, eax
```

Case Study #2

INCORRECT CALCULATION OF BUFFER SIZE



```
049E77ED   8BF1       MOV ESI,ECX
049E77EF   90         NOP
049E77F0   8A08       MOV CL,BYTE PTR DS:[EAX]
049E77F2   8D40 03    LEA EAX,DWORD PTR DS:[EAX+3]
049E77F5   880C1A     MOV BYTE PTR DS:[EDX+EBX],CL
049E77F8   42         INC EDX
049E77F9   83EE 01    SUB ESI,1
049E77FC ^ 75 F2      JNZ SHORT 049E77F0
049E77FE   8B4D FC    MOV ECX,DWORD PTR SS:[EBP-4]
049E7801   8B45 F4    MOV EAX,DWORD PTR SS:[EBP-C]
049E7804   47         INC EDI
```

```
CL=00
Access violation when writing to [04BFE000] - use Shift+F7/F8/F9 to pass exception to program
```

- **This bug is mapped to CWE-131. Such bugs may lead to an out-of-bounds read or write, possibly causing a crash, allowing arbitrary code execution, or exposing sensitive data.**

Case Study #3

LOADING UNVALIDATED RELOCATION TABLE

- **Win32.Trojan.Buerloader, active from mid-2019 and seen in the wild downloading other ransomware and banking malware.**
- **Found 19 samples of this variant showing similar behaviour and all were leading to crashes due to similar bugs.**
- **For installation, this sample drops itself in the %PROGRAMDATA% folder and starts a new instance with following command-line parameters:**
  - **C:\ProgramData\Ostersin\gennt.exe "<initial file location>" ensgJJ**
- **Starts the secinit.exe legitimate process in suspended mode using the CreateProcessW API**
- **Writes DLL and initialization code for DLL using the VirtualAlloc and WriteProcessMemory APIs**

Case Study #3

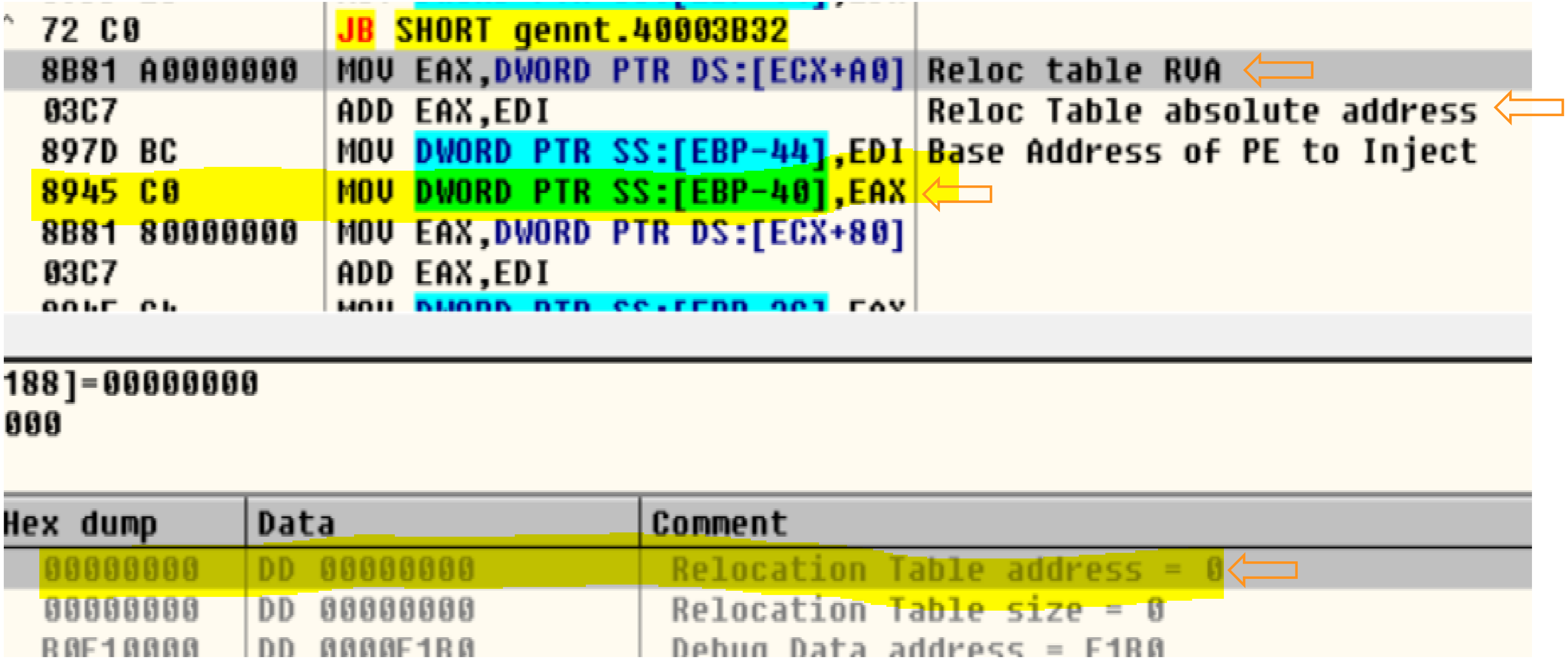
LOADING UNVALIDATED RELOCATION TABLE

- **The DLL initialization code performs the following actions:**
  - **Fixes the DLL offset using the relocation table in the PE header.**
  - **Parse the import table of the DLL and loads the DLLs mentioned in the import table using the LdrLoadDll Windows API.**
  - **Builds the import table using the LdrGetProcedureAddress API.**
  - **Calls the entry point of the DLL**

Case Study #3

LOADING UNVALIDATED RELOCATION TABLE

- **DLL file is compiled with IMAGE_FILE_RELOCS_STRIPPED**



```
72 C0             JB SHORT gennt.40003B32
8B81 A0000000     MOV EAX,DWORD PTR DS:[ECX+A0]   Reloc table RVA
03C7              ADD EAX,EDI                     Reloc Table absolute address
897D BC           MOV DWORD PTR SS:[EBP-44],EDI   Base Address of PE to Inject
8945 C0           MOV DWORD PTR SS:[EBP-40],EAX
8B81 80000000     MOV EAX,DWORD PTR DS:[ECX+80]
03C7              ADD EAX,EDI
```

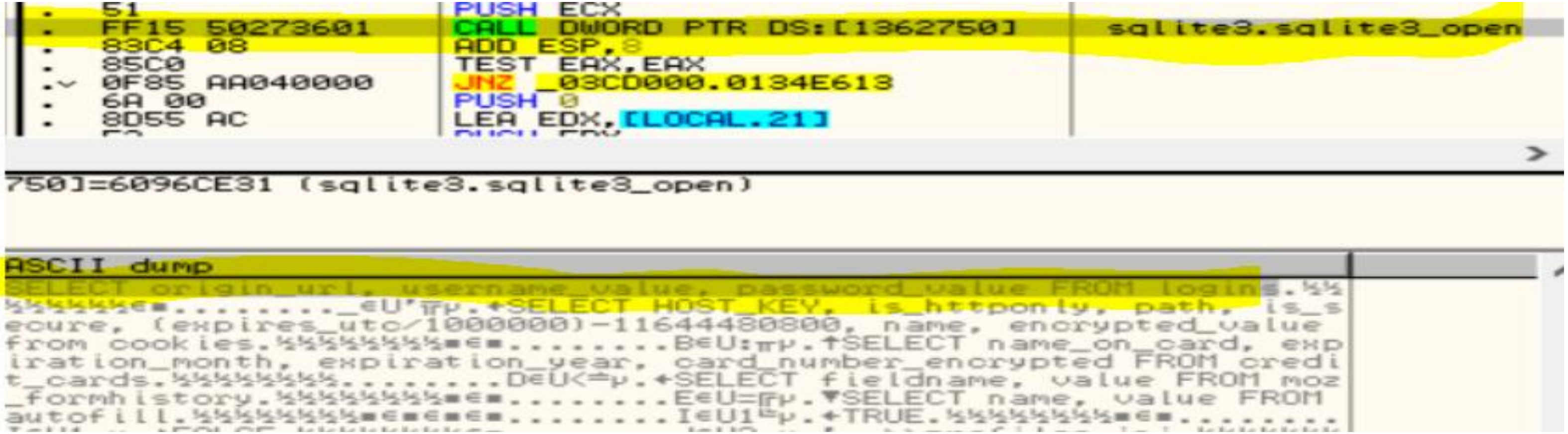
```
188]=00000000
000
```

```
Hex dump    Data           Comment
00000000    DD 00000000     Relocation Table address = 0
00000000    DD 00000000     Relocation Table size = 0
B0F10000    DD 0000F1B0     Debug Data address = F1B0
```

Case Study #4

INCORRECT CHECK OF FUNCTION RETURN VALUE

- **Win32.PWS.Oski introduced in 2019, steals personal and sensitive information from a victim's system.**
- **It also steals passwords stored in Google Chrome.**
- **Copies the 'Login Data' file from the location '%LOCALAPPDATA%\Google\Chrome\User Data\Default' in 'C:\ProgramData\<InstallFolder>\tmp'**
- **Malware extracts origin_url, username_value and password_value**



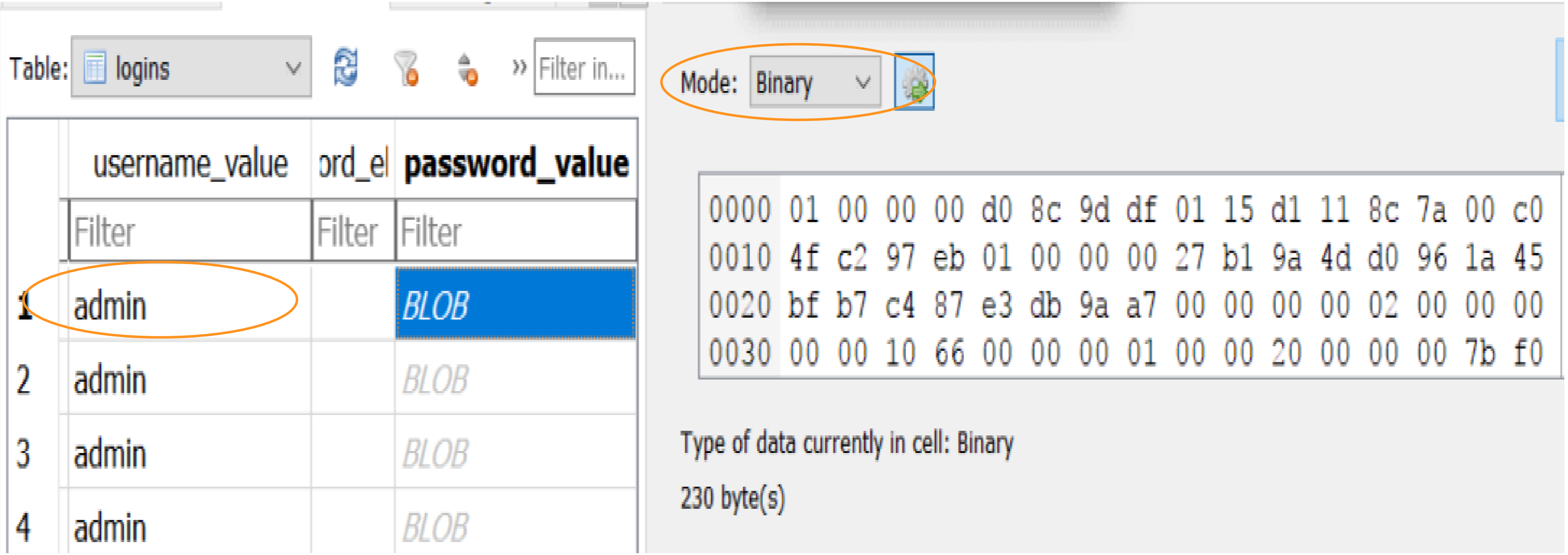
```
 .   51               PUSH ECX
 .   FF15 50273601    CALL DWORD PTR DS:[1362750]      sqlite3.sqlite3_open
 .   83C4 08          ADD ESP,8
 .   85C0             TEST EAX,EAX
 .v  0F85 AA040000    JNZ _03CD000.0134E613
 .   6A 00            PUSH 0
 .   8D55 AC          LEA EDX,[LOCAL.21]

750]=6096CE31 (sqlite3.sqlite3_open)

ASCII dump
SELECT origin_url, username_value, password_value FROM logins.½½
½½½½½½€■........_€U'╥µ.♦SELECT HOST_KEY, is_httponly, path, is_s
ecure, (expires_utc/1000000)-11644480800, name, encrypted_value
from cookies.½½½½½½½½■€■........B€U:╥µ.↑SELECT name_on_card, exp
iration_month, expiration_year, card_number_encrypted FROM credi
t_cards.½½½½½½½½........D€U<═µ.♦SELECT fieldname, value FROM moz
_formhistory.½½½½½½½½■€■........E€U=╔µ.▼SELECT name, value FROM
autofill.½½½½½½½½■€■€■€■........I€U1╘µ.♦TRUE.½½½½½½½½■€■........
```

Case Study #4

INCORRECT CHECK OF FUNCTION RETURN VALUE



	username_value	ord_el	password_value
	Filter	Filter	Filter
1	admin		BLOB
2	admin		BLOB
3	admin		BLOB
4	admin		BLOB

Mode: Binary

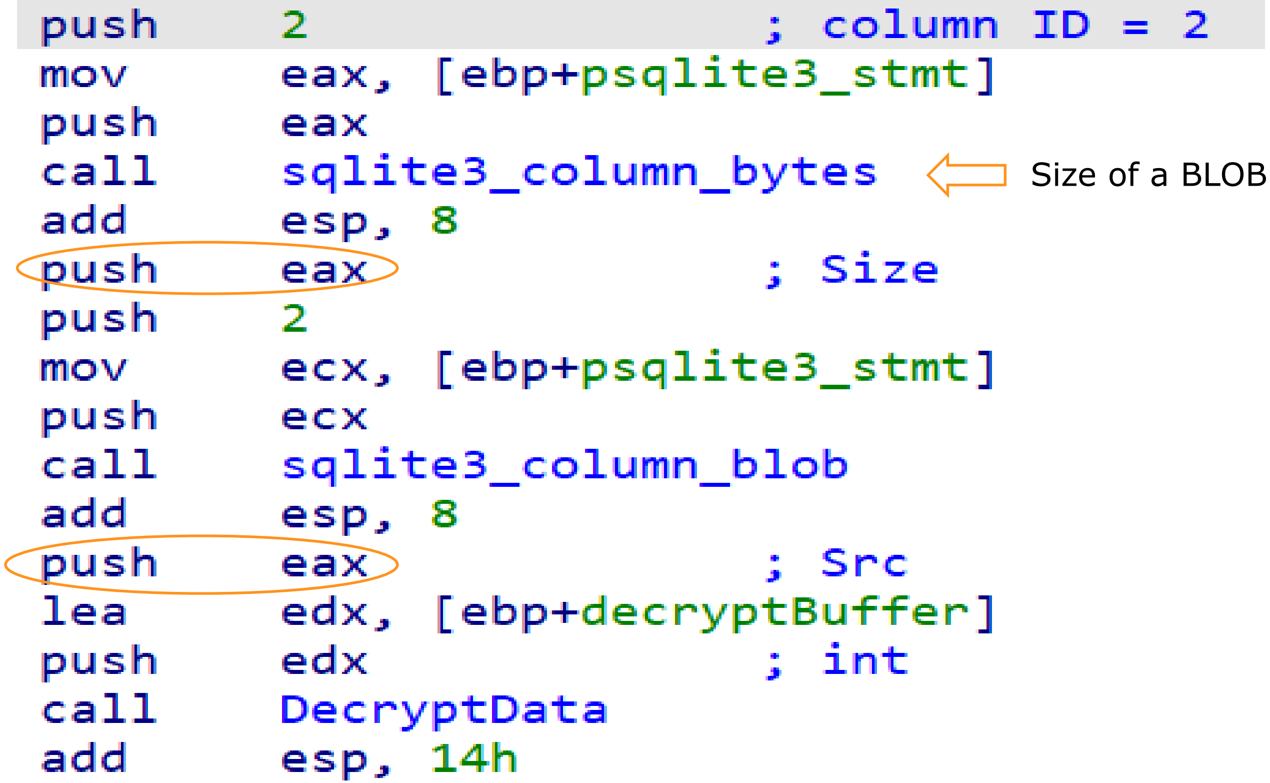
```
0000 01 00 00 00 d0 8c 9d df 01 15 d1 11 8c 7a 00 c0
0010 4f c2 97 eb 01 00 00 00 27 b1 9a 4d d0 96 1a 45
0020 bf b7 c4 87 e3 db 9a a7 00 00 00 00 02 00 00 00
0030 00 00 10 66 00 00 00 01 00 00 20 00 00 00 7b f0
```

Type of data currently in cell: Binary

230 byte(s)

Case Study #4

INCORRECT CHECK OF FUNCTION RETURN VALUE



```
push    2                         ; column ID = 2
mov     eax, [ebp+psqlite3_stmt]
push    eax
call    sqlite3_column_bytes
add     esp, 8
push    eax                       ; Size
push    2
mov     ecx, [ebp+psqlite3_stmt]
push    ecx
call    sqlite3_column_blob
add     esp, 8
push    eax                       ; Src
lea     edx, [ebp+decryptBuffer]
push    edx                       ; int
call    DecryptData
add     esp, 14h
```

Size of a BLOB or a UTF-8 TEXT result in bytes

Case Study #4

INCORRECT CHECK OF FUNCTION RETURN VALUE

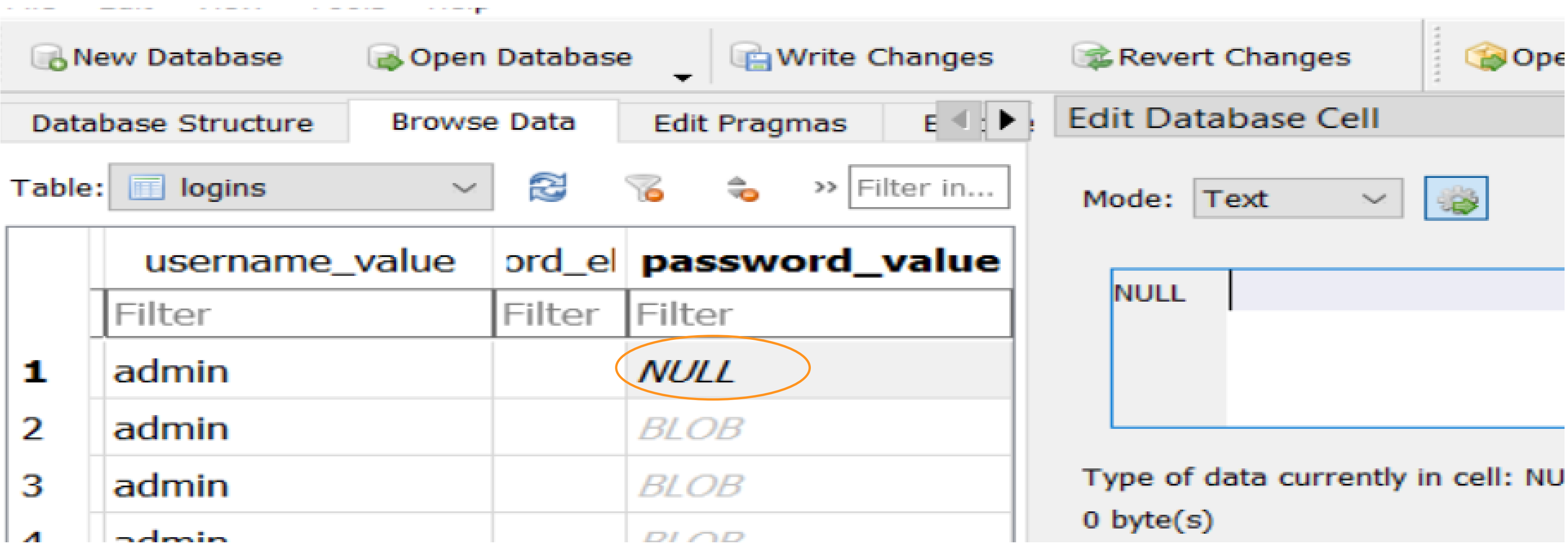


Table: logins

	username_value	ord_el	password_value
	Filter	Filter	Filter
1	admin		*NULL*
2	admin		*BLOB*
3	admin		*BLOB*
4	admin		*BLOB*

Mode: Text

NULL

Type of data currently in cell: NU

0 byte(s)

Case Study #5

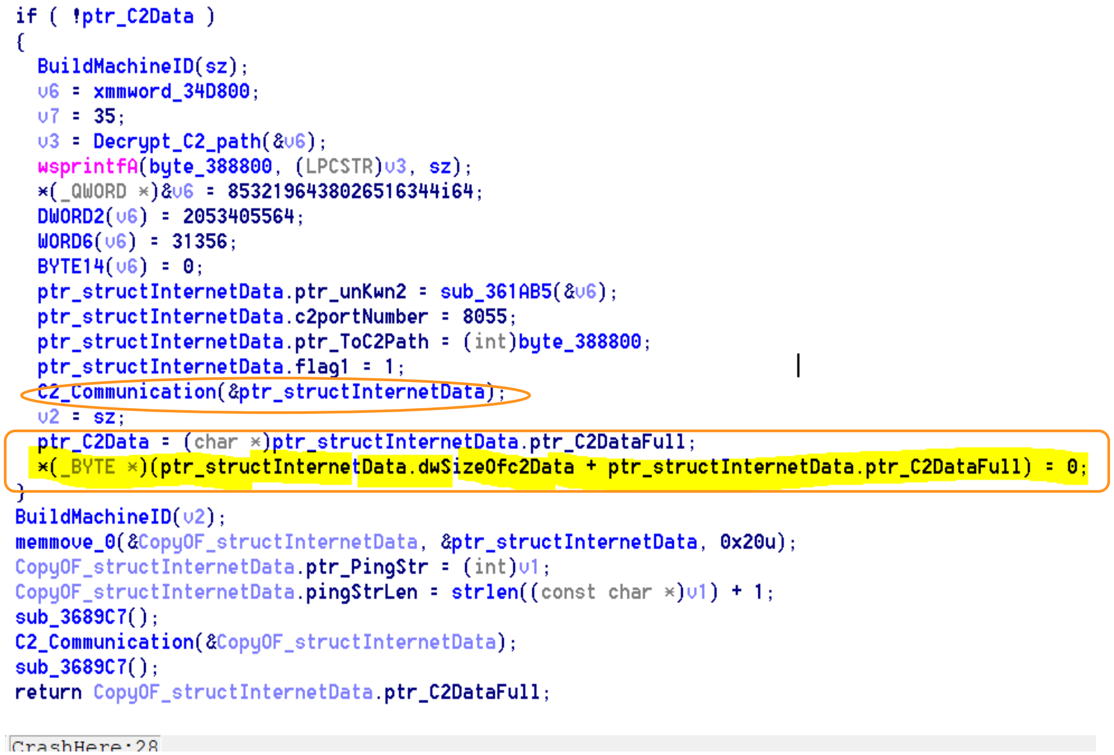
INCONSISTENT INTERPRETATION OF HTTP RESPONSE HANDLING

- **Bug 1: Win32.Downloader.Penguish – no check for InternetReadFile API output**

- **A downloader sample and it shows an execution error when it encounters an unexpected HTTP response from the C2**

- **Doesn’t validate the C2 response read through the InternetReadFile Windows API**

- **Found 100+ similar samples from this family**

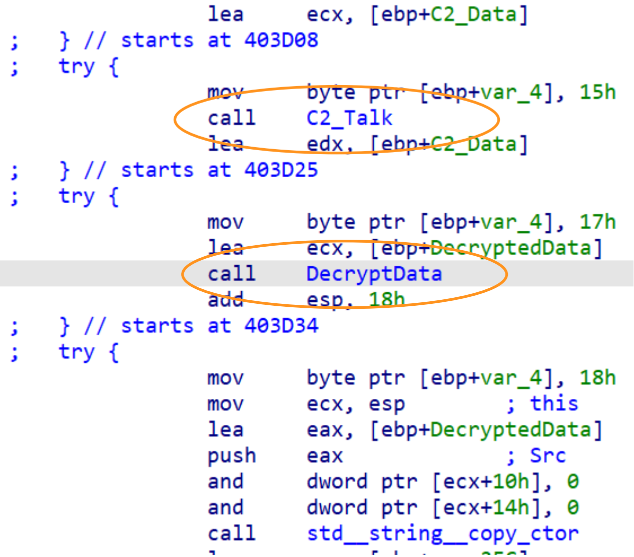


```
if ( !ptr_C2Data )
{
  BuildMachineID(sz);
  v6 = xmmword_34D800;
  v7 = 35;
  v3 = Decrypt_C2_path(&v6);
  wsprintfA(byte_388800, (LPCSTR)v3, sz);
  *(_QWORD *)&v6 = 8532196438026516344i64;
  DWORD2(v6) = 2053405564;
  WORD6(v6) = 31356;
  BYTE14(v6) = 0;
  ptr_structInternetData.ptr_unKwn2 = sub_361AB5(&v6);
  ptr_structInternetData.c2portNumber = 8055;
  ptr_structInternetData.ptr_ToC2Path = (int)byte_388800;
  ptr_structInternetData.flag1 = 1;
  C2_Communication(&ptr_structInternetData);
  v2 = sz;
  ptr_C2Data = (char *)ptr_structInternetData.ptr_C2DataFull;
  *(_BYTE *)(ptr_structInternetData.dwSizeOfc2Data + ptr_structInternetData.ptr_C2DataFull) = 0;
}
BuildMachineID(v2);
memmove_0(&CopyOF_structInternetData, &ptr_structInternetData, 0x20u);
CopyOF_structInternetData.ptr_PingStr = (int)v1;
CopyOF_structInternetData.pingStrLen = strlen((const char *)v1) + 1;
sub_3689C7();
C2_Communication(&CopyOF_structInternetData);
sub_3689C7();
return CopyOF_structInternetData.ptr_C2DataFull;
```

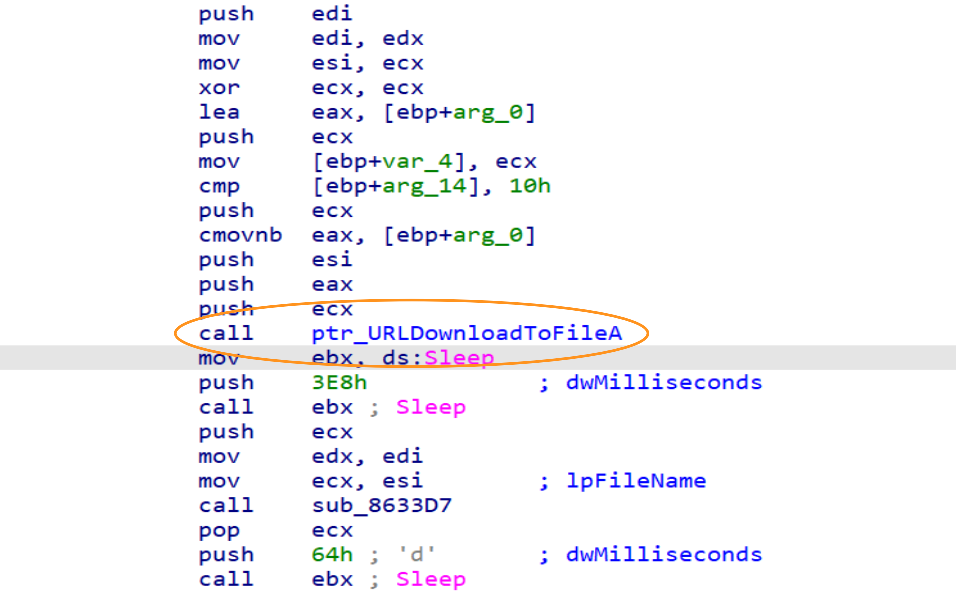
Case Study #5

INCONSISTENT INTERPRETATION OF HTTP RESPONSE HANDLING

- **Bug 2: Win32.Downloader.Glupteba – no check for URLDownloadToFile API output**



```
                lea     ecx, [ebp+C2_Data]
;   } // starts at 403D08
;   try {
                mov     byte ptr [ebp+var_4], 15h
                call    C2_Talk
                lea     edx, [ebp+C2_Data]
;   } // starts at 403D25
;   try {
                mov     byte ptr [ebp+var_4], 17h
                lea     ecx, [ebp+DecryptedData]
                call    DecryptData
                add     esp, 18h
;   } // starts at 403D34
;   try {
                mov     byte ptr [ebp+var_4], 18h
                mov     ecx, esp        ; this
                lea     eax, [ebp+DecryptedData]
                push    eax             ; Src
                and     dword ptr [ecx+10h], 0
                and     dword ptr [ecx+14h], 0
                call    std__string__copy_ctor
```



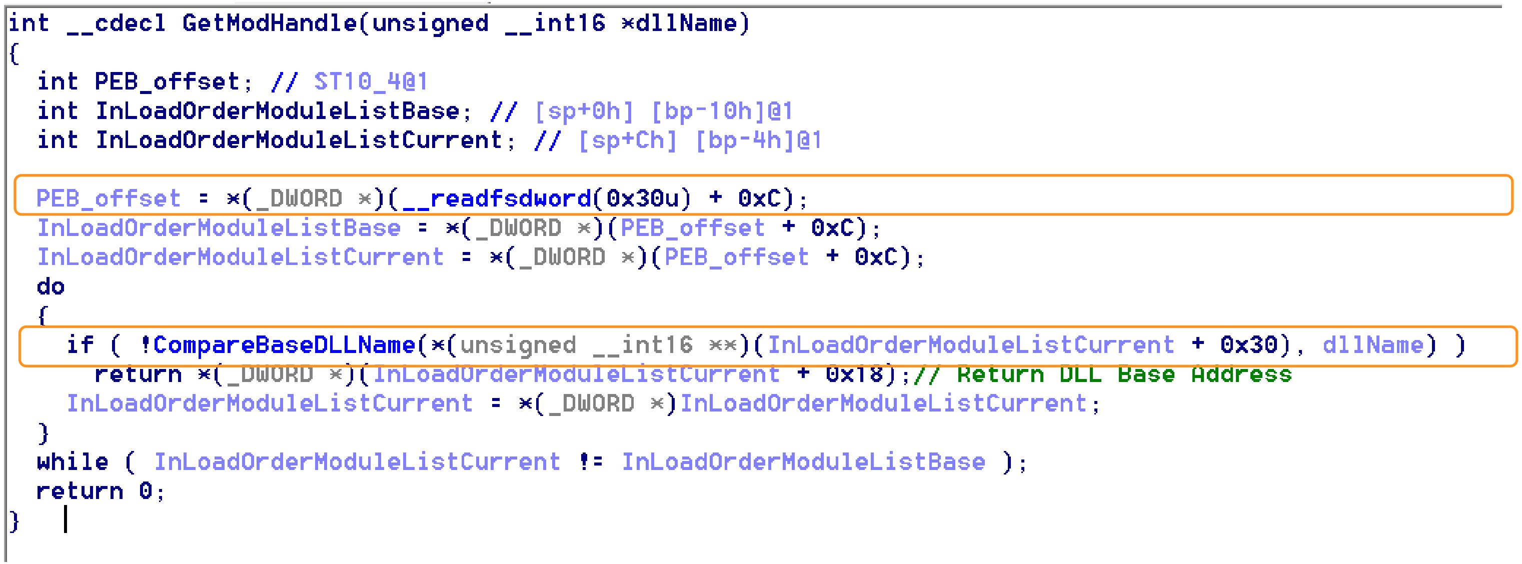
```
push    edi
mov     edi, edx
mov     esi, ecx
xor     ecx, ecx
lea     eax, [ebp+arg_0]
push    ecx
mov     [ebp+var_4], ecx
cmp     [ebp+arg_14], 10h
push    ecx
cmovnb  eax, [ebp+arg_0]
push    esi
push    eax
push    ecx
call    ptr_URLDownloadToFileA
mov     ebx, ds:Sleep
push    3E8h                    ; dwMilliseconds
call    ebx ; Sleep
push    ecx
mov     edx, edi
mov     ecx, esi                ; lpFileName
call    sub_8633D7
pop     ecx
push    64h ; 'd'               ; dwMilliseconds
call    ebx ; Sleep
```

- **Both these bugs are related to the misinterpretation of HTTP response, which falls under CWE-444**

Case Study #6

WILDCARD SEARCH FOR DLL

- **Win32.Backdoor.Emotet a famous malware-as-a-service (MaaS), was first seen in 2014**
- **Found 318 Emotet samples showing execution errors due to different types of bugs.**
- **Issue in the logic it uses to get the address of the NTDLL.DLL system DLL.**



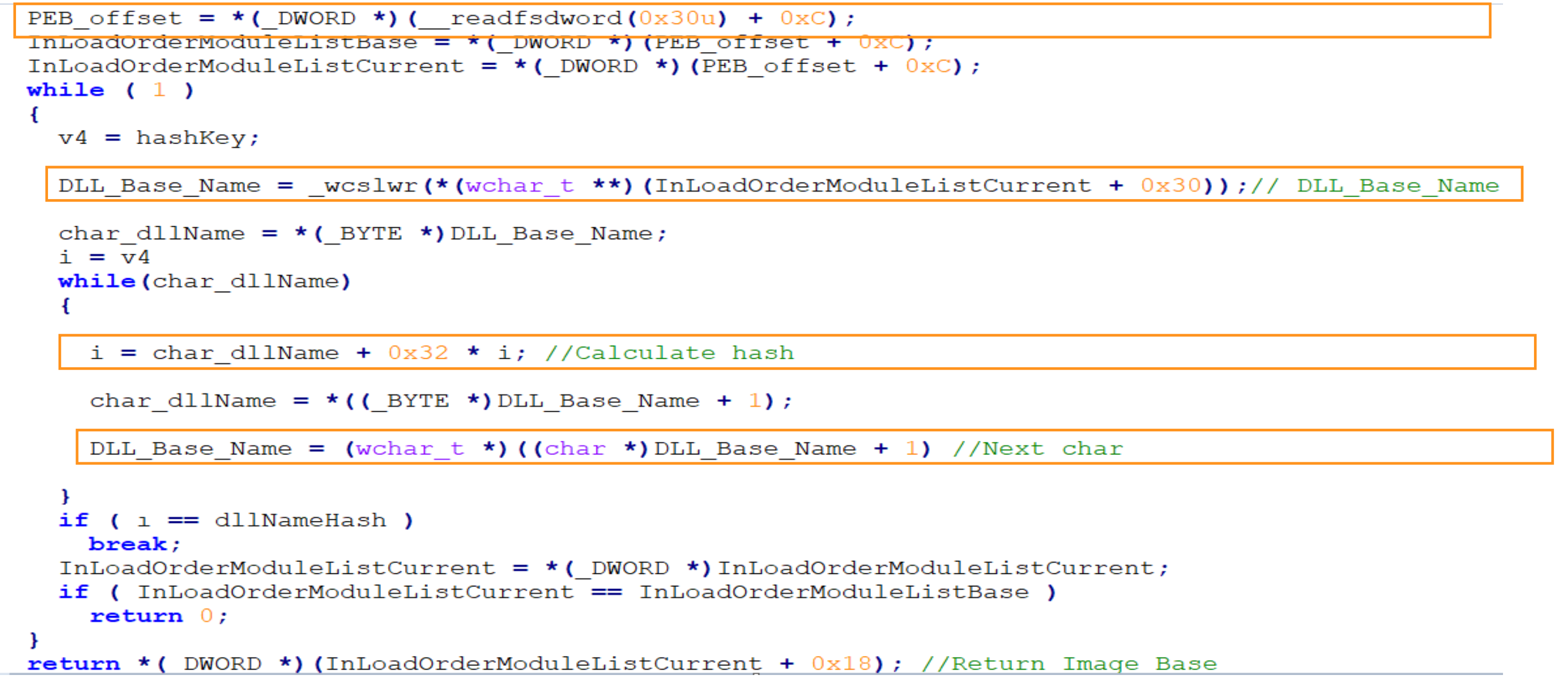
```
int __cdecl GetModHandle(unsigned __int16 *dllName)
{
  int PEB_offset; // ST10_4@1
  int InLoadOrderModuleListBase; // [sp+0h] [bp-10h]@1
  int InLoadOrderModuleListCurrent; // [sp+Ch] [bp-4h]@1

  PEB_offset = *(_DWORD *)(__readfsdword(0x30u) + 0xC);
  InLoadOrderModuleListBase = *(_DWORD *)(PEB_offset + 0xC);
  InLoadOrderModuleListCurrent = *(_DWORD *)(PEB_offset + 0xC);
  do
  {
    if ( !CompareBaseDLLName(*(unsigned __int16 **)(InLoadOrderModuleListCurrent + 0x30), dllName) )
      return *(_DWORD *)(InLoadOrderModuleListCurrent + 0x18);// Return DLL Base Address
    InLoadOrderModuleListCurrent = *(_DWORD *)InLoadOrderModuleListCurrent;
  }
  while ( InLoadOrderModuleListCurrent != InLoadOrderModuleListBase );
  return 0;
}
```

Case Study #6

WILDCARD SEARCH FOR DLL

- A similar issue was found in another sample but for a different DLL –Kernel32.dll
- Change the file name to anything that starts with ‘K’, it will result in the crash.
- Such bugs are covered under CWE-1023 and may lead to altered execution logic, bypass of protection mechanism, etc



```
PEB_offset = *(_DWORD *)(__readfsdword(0x30u) + 0xC);
InLoadOrderModuleListBase = *(_DWORD *)(PEB_offset + 0xC);
InLoadOrderModuleListCurrent = *(_DWORD *)(PEB_offset + 0xC);
while ( 1 )
{
  v4 = hashKey;
  DLL_Base_Name = _wcslwr(*(wchar_t **)(InLoadOrderModuleListCurrent + 0x30));// DLL_Base_Name
  char_dllName = *(_BYTE *)DLL_Base_Name;
  i = v4
  while(char_dllName)
  {
    i = char_dllName + 0x32 * i; //Calculate hash
    char_dllName = *((_BYTE *)DLL_Base_Name + 1);
    DLL_Base_Name = (wchar_t *)((char *)DLL_Base_Name + 1) //Next char
  }
  if ( i == dllNameHash )
    break;
  InLoadOrderModuleListCurrent = *(_DWORD *)InLoadOrderModuleListCurrent;
  if ( InLoadOrderModuleListCurrent == InLoadOrderModuleListBase )
    return 0;
}
return *( DWORD *)(InLoadOrderModuleListCurrent + 0x18); //Return Image Base
```

Case Study #7

USE OF FUNCTION WITH INCONSISTENT IMPLEMENTATIONS

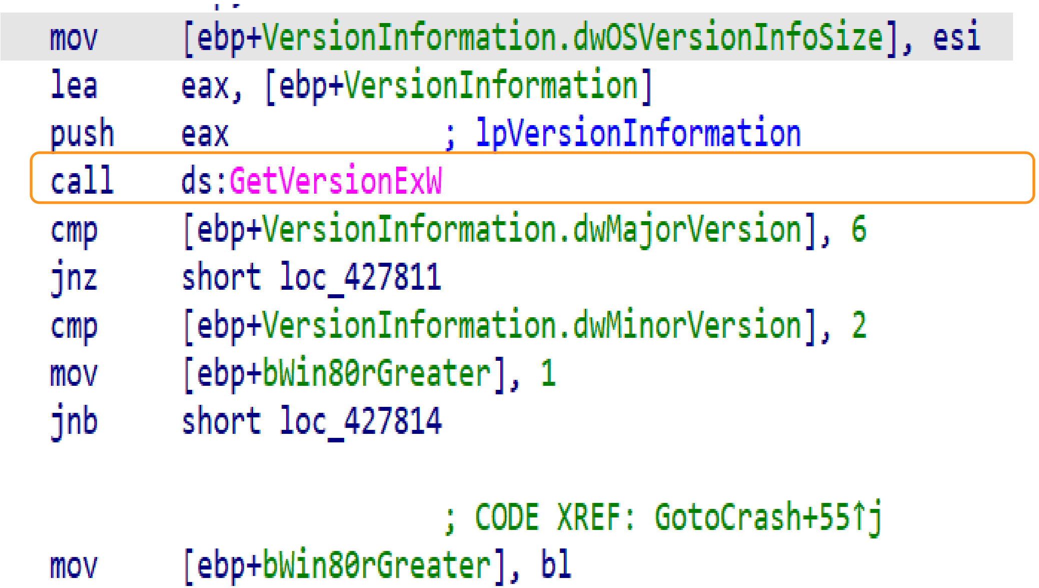
- **Malware samples are usually packed using unknown packers.**
- **Win32.PWS.Raccoon type of malware focused on gathering sensitive information from the infected system.**
- **Extracts and steal credentials stored by Internet Explorer.**
- **Starting with Windows 7, Internet Explorer stores sensitive information including passwords in the Windows Vault.**
- **Malware uses different APIs (VaultOpenVault, VaultCloseVault, VaultEnumerateItems, VaultGetItem and VaultFree) from VAULTCLI.DLL**
- **There is a change in the VaultGetItem API starting from Windows 8**

Case Study #7

USE OF FUNCTION WITH INCONSISTENT IMPLEMENTATIONS

- **As per MSDN documentation, the behaviour of this API has changed, starting from Windows 8.1.**

- **For applications not manifested for 8.1 or Windows 10, this API will always return the Windows 8 OS version value (6.2)**

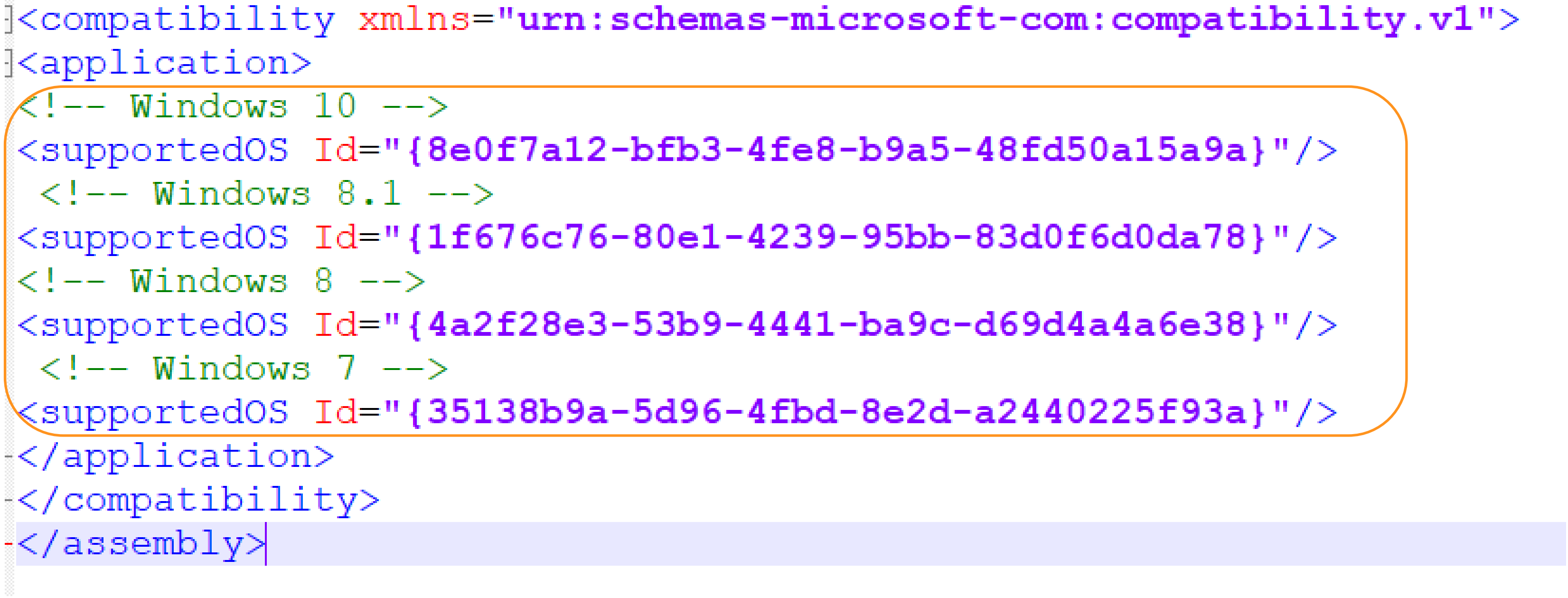


```
mov     [ebp+VersionInformation.dwOSVersionInfoSize], esi
lea     eax, [ebp+VersionInformation]
push    eax             ; lpVersionInformation
call    ds:GetVersionExW
cmp     [ebp+VersionInformation.dwMajorVersion], 6
jnz     short loc_427811
cmp     [ebp+VersionInformation.dwMinorVersion], 2
mov     [ebp+bWin8OrGreater], 1
jnb     short loc_427814

                        ; CODE XREF: GotoCrash+55↑j
mov     [ebp+bWin8OrGreater], bl
```

Case Study #7

USE OF FUNCTION WITH INCONSISTENT IMPLEMENTATIONS

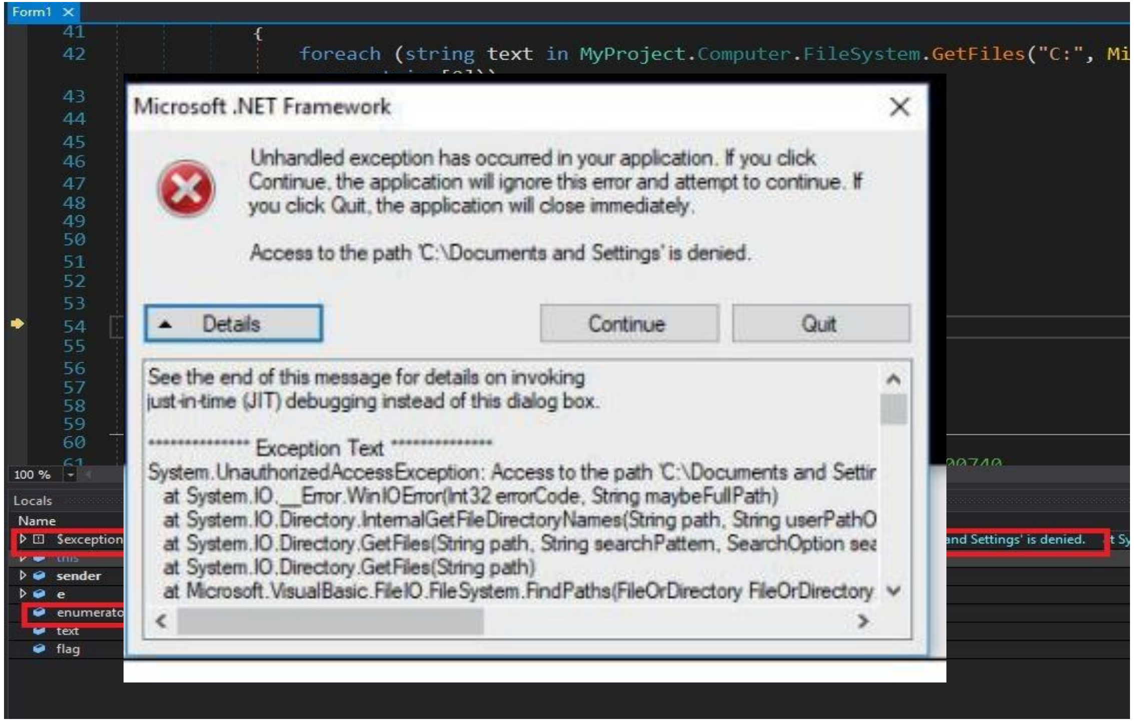


```
<compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
<application>
<!-- Windows 10 -->
<supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}"/>
 <!-- Windows 8.1 -->
<supportedOS Id="{1f676c76-80e1-4239-95bb-83d0f6d0da78}"/>
<!-- Windows 8 -->
<supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>
 <!-- Windows 7 -->
<supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/>
</application>
</compatibility>
</assembly>
```

Case Study #8

IMPROPER HANDLING OF INSUFFICIENT PERMISSIONS / PRIVILEGES

- **Win32.Ransom.Sapphire a type of malware that encrypts a victim's files and demands a ransom.**
- **Encrypts all files in the 'C:\' directory and skips files with the .VIVELAG extension.**
- **Found a variant of this ransomware that doesn't check the permission of directories**



```
{
    foreach (string text in MyProject.Computer.FileSystem.GetFiles("C:", Mi
```

Microsoft .NET Framework

Unhandled exception has occurred in your application. If you click Continue, the application will ignore this error and attempt to continue. If you click Quit, the application will close immediately.

Access to the path 'C:\Documents and Settings' is denied.

```
See the end of this message for details on invoking
just-in-time (JIT) debugging instead of this dialog box.

************** Exception Text **************
System.UnauthorizedAccessException: Access to the path 'C:\Documents and Settir
  at System.IO.__Error.WinIOError(Int32 errorCode, String maybeFullPath)
  at System.IO.Directory.InternalGetFileDirectoryNames(String path, String userPathO
  at System.IO.Directory.GetFiles(String path, String searchPattern, SearchOption sea
  at System.IO.Directory.GetFiles(String path)
  at Microsoft.VisualBasic.FileIO.FileSystem.FindPaths(FileOrDirectory FileOrDirectory
```

Conclusion

- **Looked at multiple examples of malware with different types of vulnerabilities.**
- **Tried to classify all the bugs using MITRE’s CWE list.**
- **This study includes a broad range of malware from stealers and downloaders to ransomware.**
- **This research shows that malware code often contains multiple bugs and indicates that no proper quality assurance checks.**
- **Security vendors can leverage these bugs to write different types of signatures to identify and block such malware attacks**

Thank you!