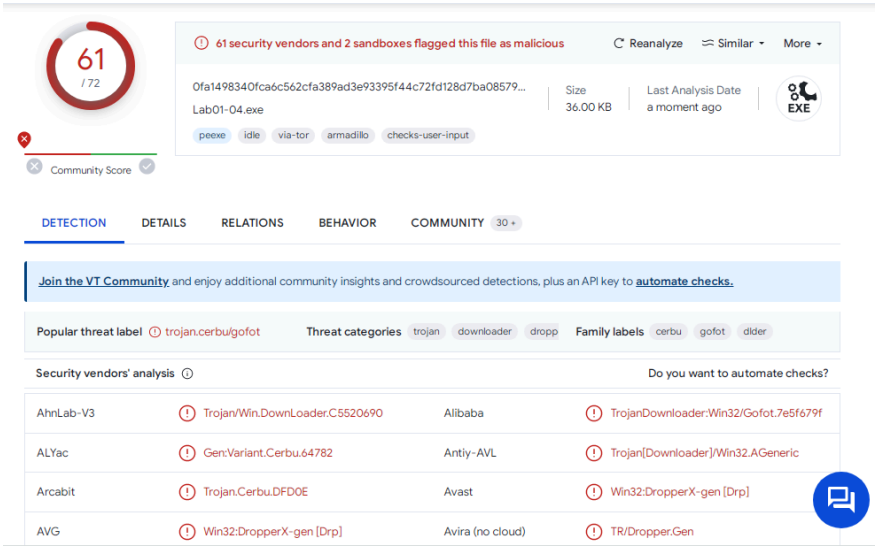
Anvitha Yerneni
ayernen@g.clemson.edu

Malware Lab- 1

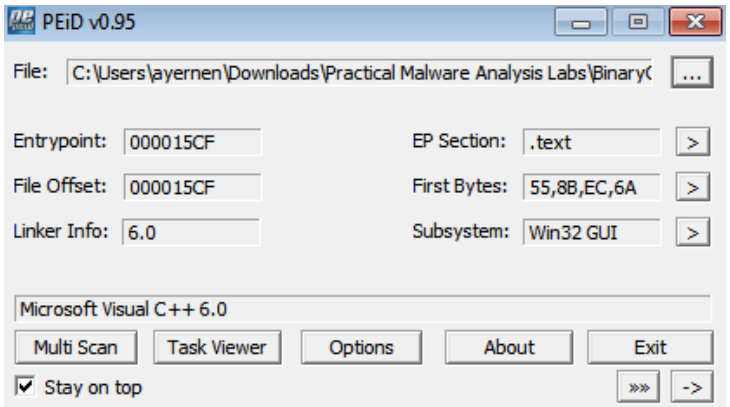
Problem 1 – Static analysis (30 pts): Lab 1-4

1.



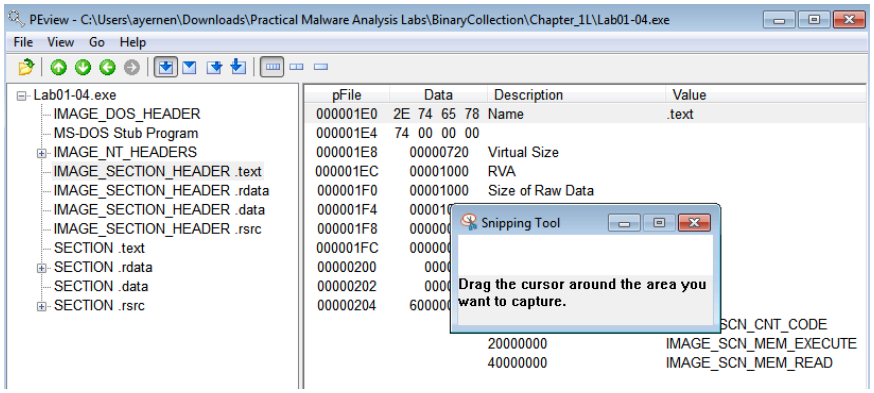
The screenshot provided demonstrates that the malware sample is recognized by antivirus programs, with 61 out of 72 security providers identifying the file as harmful. A subset of these security vendors is also listed for reference.

2.

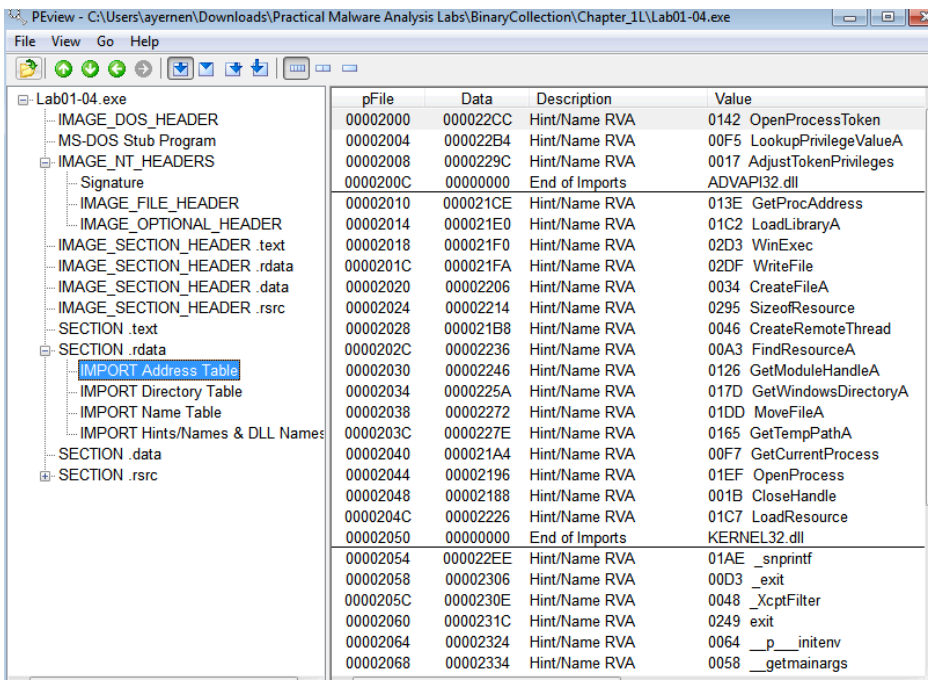


The VirusTotal analysis indicated that the packer used was Microsoft Visual C++. This was confirmed by using the PEiD v0.95 tool, which showed that the packer is Microsoft Visual C++. This suggests that the file has not been packed.

Anvitha Yerneni
ayernen@g.clemson.edu



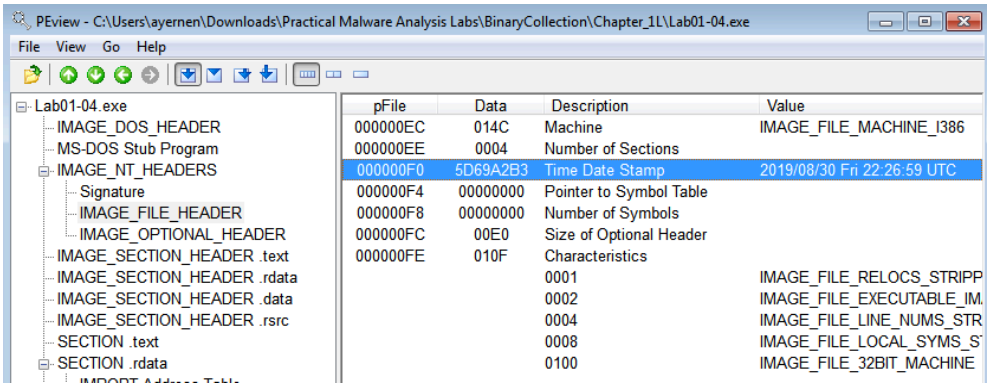
Additionally, I used the PEview tool to obtain more details about the file. As observed in the provided screenshot, the similarity between the virtual size and the raw data size suggests that the file has not been packed.



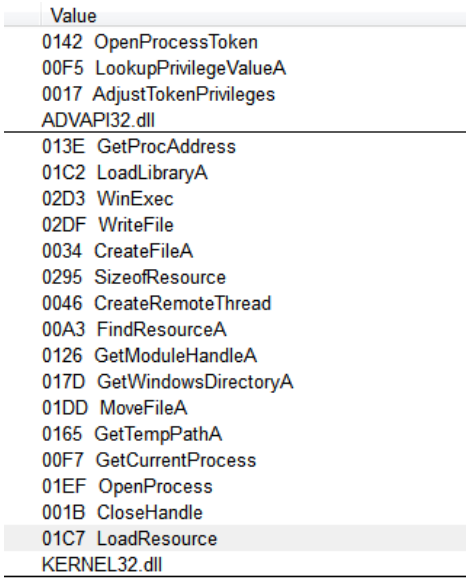
Finally, I reviewed the Address Table in the .rdata section to examine the quantity of imports this malware sample contains. From the image provided, it's evident that the import count is moderate, suggesting that the file isn't compressed using packing techniques, but it appears to be obfuscated instead.

3. The screenshot provided shows that, through the use of the PEview application, it was established that the program's compilation date was Friday, August 30, 2019, at 22:26:59 UTC.

Anvitha Yerneni
ayernen@g.clemson.edu



4.



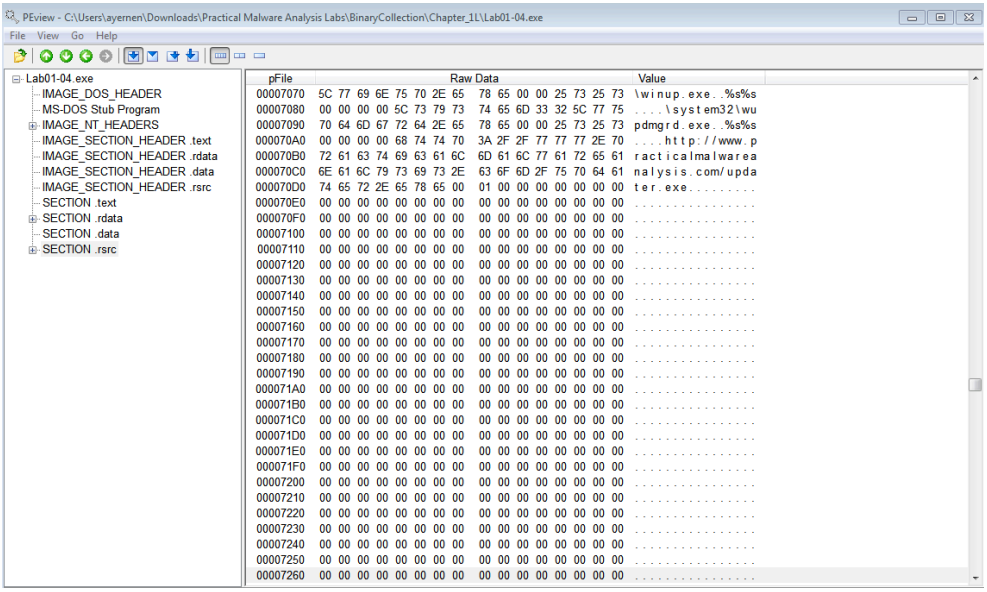
Based on the VirusTotal report's listed imports, it appears that the program in question is designed to identify and access specific resources, as indicated by the use of functions like FindResourceA and LoadResource. Furthermore, the inclusion of functions such as CreateFileA, WriteFile, MoveFileA, and WinExec implies that the program is capable of creating, modifying, and moving files on the disk, in addition to launching executable files. Additionally, the presence of imports like AdjustTokenPrivileges and LookupPrivilegeValueA suggests the program's ability to evaluate and modify the permissions required for executing files, allowing it to alter the privilege levels associated with particular files when necessary.

5.

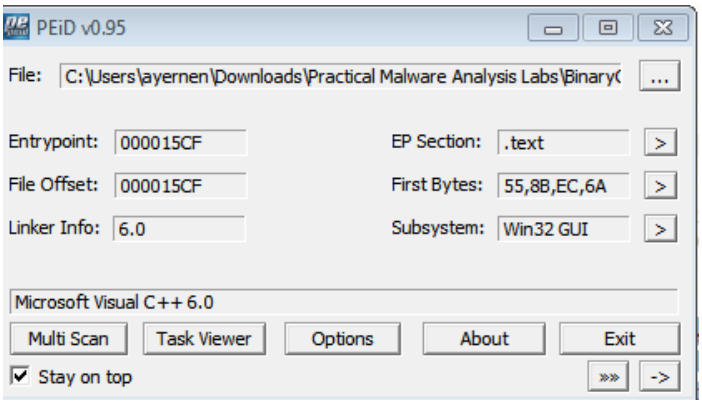
Using the PEview tool, I was able to detect indicators of compromise associated with this malware on both the host and network levels. For host-based detection, the presence of the executable “\system32\wupdmgrd.exe” on a system suggests that the malware has successfully

Anvitha Yerneni
ayernen@g.clemson.edu

infiltrated the device. On the network side, the URL "http://www.practicalmalwareanalysis.com/updater.exe" could serve as a source for further malware downloads, indicating potential network-based infection vectors.

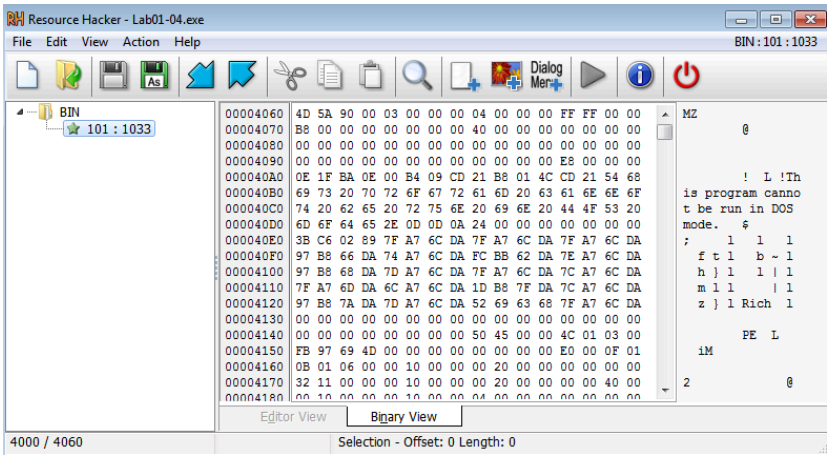


Furthermore, I employed the PEiD String Viewer v0.95 tool to examine the strings present in the malware sample more closely. As depicted in the provided screenshot, this application allows for a clearer visualization of both host-based and network-based indicators.

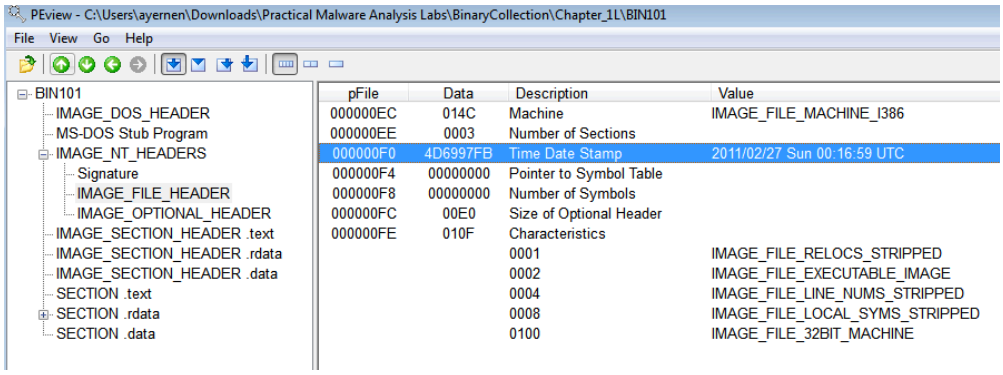


6. Based on the VirusTotal analysis, the malware file contained a single resource, as indicated by the prompt. Utilizing Resource Hacker for examination, I determined this resource to be a binary file and proceeded to extract its contents to analyze further.

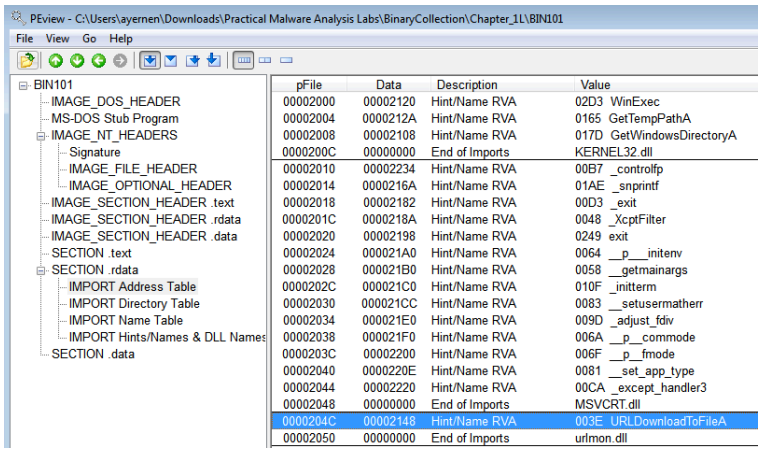
Anvitha Yerneni
ayernen@g.clemson.edu



After utilizing the Resource Hacker tool to extract the binary file, I proceeded to examine its details using the PEview application. From the analysis, it became evident that the software was originally compiled on February 27, 2011, at 00:16:59 UTC, which contradicts the altered date listed in 2019.



Furthermore, the document revealed more imports concealed within it. Notably, URLDownloadToFileA was identified as one of the imports. This function is frequently used by harmful software to download files from the internet and initiate further malicious activities.



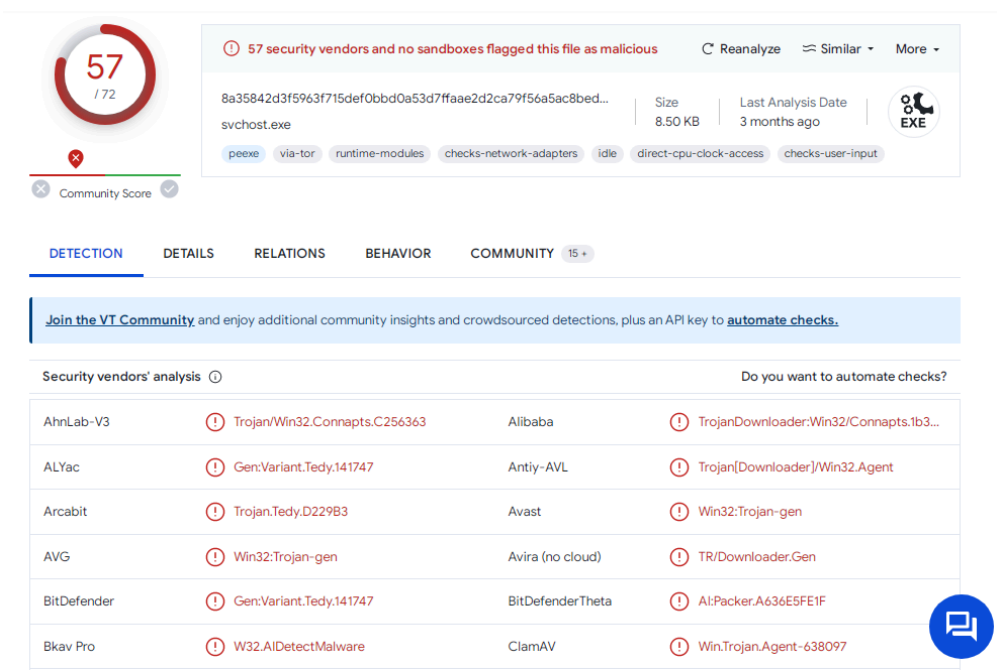
Anvitha Yerneni
ayernen@g.clemson.edu

Problem 2 – Basic static analysis (20 pts):

1.
MD5 sum (Message Digest Algorithm 5) is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value. It's typically expressed as a 32-character hexadecimal number. MD5 is used to ensure data integrity, as even a small change in the input data will produce a significantly different hash value. Below shown is MD5 sum generated for hw1-2.malware using command prompt.

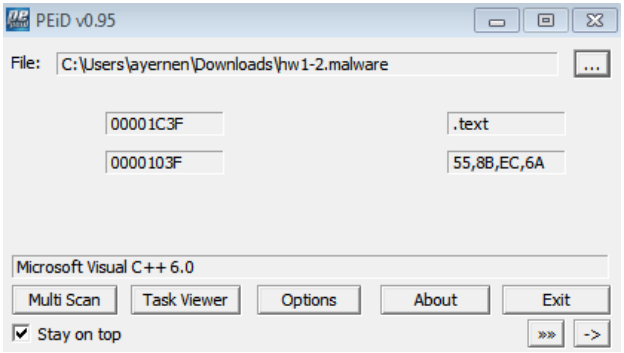
```
C:\Users\ayernen\Downloads>CertUtil -hashfile hw1-2.malware MD5
MD5 hash of file hw1-2.malware:
02 65 8b c9 80 1f 98 df df 16 7a cc f5 7f 6a 36
CertUtil: -hashfile command completed successfully.
```

Upon conducting an initial static analysis of the newly encountered malware sample, several notable findings were uncovered. Notably, when the file was examined through VirusTotal, the results were alarming. A significant majority, specifically 57 out of 72 antivirus vendors, identified the file as harmful. A glimpse of these vendor reports is provided in the screenshot below.

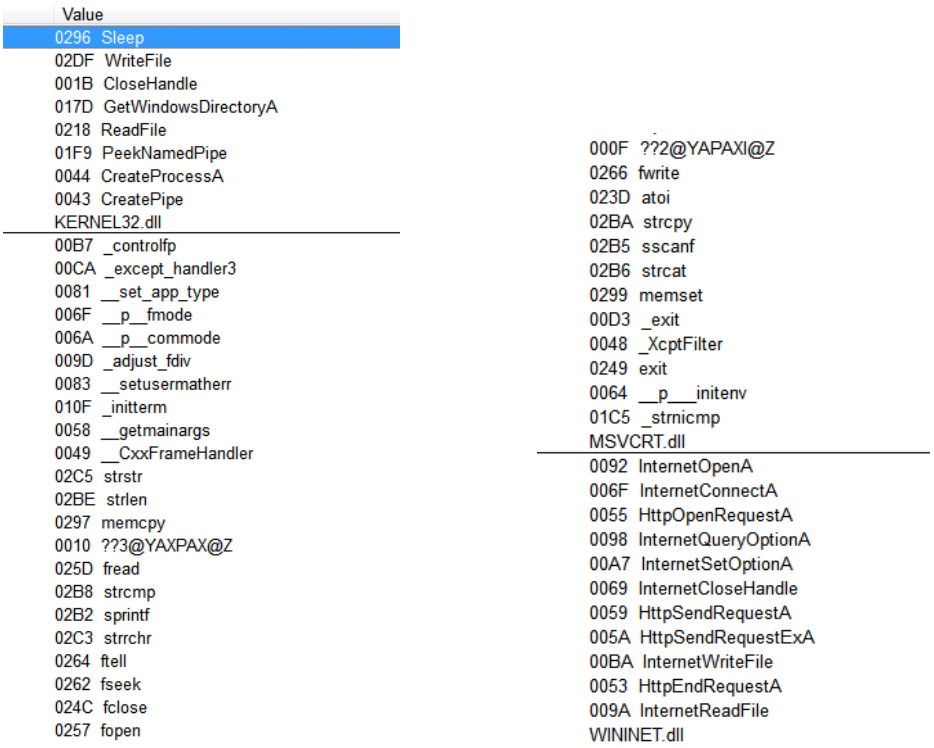


2. Subsequently, I observed that the packer identified by PEiD was Microsoft Visual C++ which suggests that the file was not subjected to packing. This finding was confirmed through the use of the PEiD version 0.95 tool, as illustrated in the screenshot provided below.

Anvitha Yerneni
ayernen@g.clemson.edu



Furthermore, the presence of numerous imports in this file signals that a packer likely wasn't employed. This aspect will be explored in more detail in the subsequent query. Additionally, the discovery of a single resource in the VirusTotal report for this file raises suspicions, similar to what was observed in a prior malware example. This anomaly warrants a deeper examination using the Resource Hacker tool.



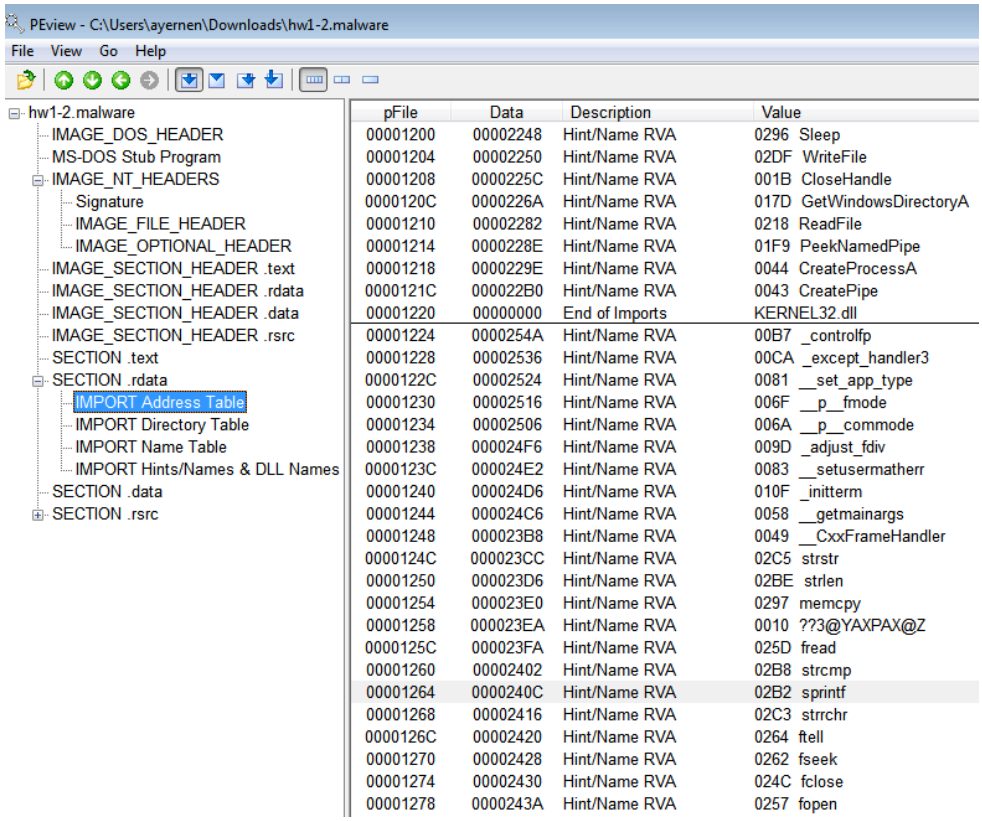
3. Based on the screenshot provided below and the analysis using PEview, it's evident that the program includes functionalities to interact with local files, indicated by imports like fopen, fclose, fread, ReadFile, WriteFile, and fwrite. This implies the software is designed to open, read, and modify files stored on the computer.

Anvitha Yerneni
ayernen@g.clemson.edu

Furthermore, the presence of imports like PeekNamedPipe, CreatePipe, and CreateProcessA raises some red flags. These functions allow the program to create anonymous pipes and spawn new processes on the host system, which could be problematic.

The most alarming aspect is the inclusion of the WININET.dll library, especially since the program doesn't require internet connectivity. Functions such as HttpOpenRequestA, HttpSendRequestA, InternetReadFile, and InternetWriteFile hint at the capability to interact with online resources, suggesting the software might access and manipulate internet-based files.

In summary, the program exhibits capabilities that could be deemed potentially harmful, including file manipulation, process creation, and internet communication. Without a thorough investigation, the true intent behind these functionalities remains uncertain, but they certainly warrant caution.

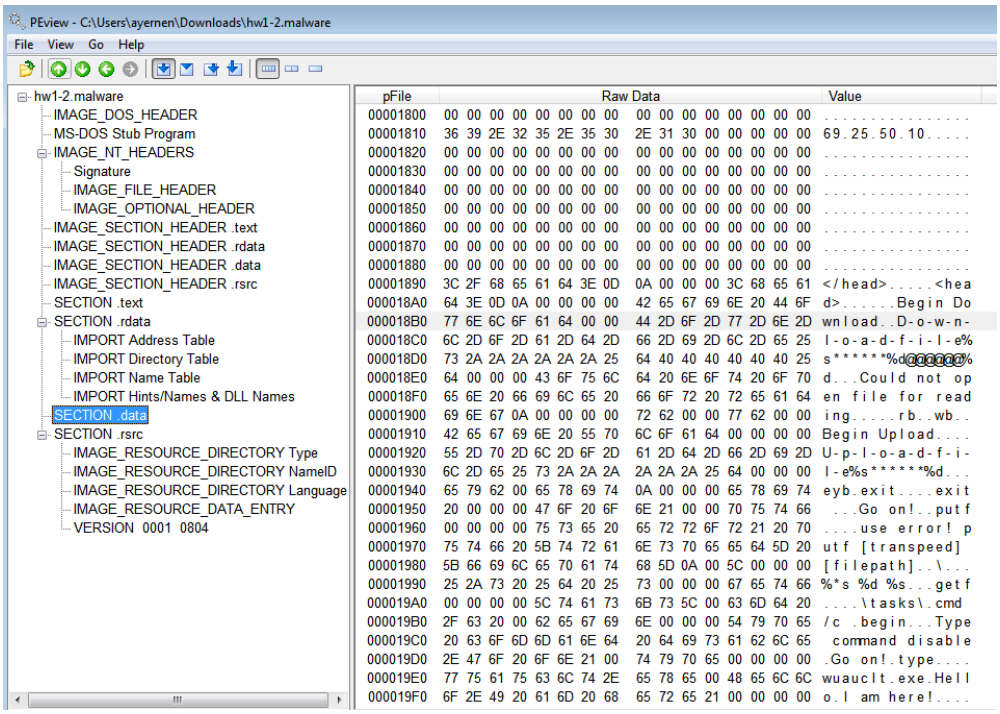


pFile	Data	Description	Value
00001200	00002248	Hint/Name RVA	0296 Sleep
00001204	00002250	Hint/Name RVA	02DF WriteFile
00001208	0000225C	Hint/Name RVA	001B CloseHandle
0000120C	0000226A	Hint/Name RVA	017D GetWindowsDirectoryA
00001210	00002282	Hint/Name RVA	0218 ReadFile
00001214	0000228E	Hint/Name RVA	01F9 PeekNamedPipe
00001218	0000229E	Hint/Name RVA	0044 CreateProcessA
0000121C	000022B0	Hint/Name RVA	0043 CreatePipe
00001220	00000000	End of Imports	KERNEL32.dll
00001224	0000254A	Hint/Name RVA	00B7 _controlfp
00001228	00002536	Hint/Name RVA	00CA _except_handler3
0000122C	00002524	Hint/Name RVA	0081 __set_app_type
00001230	00002516	Hint/Name RVA	006F __p__fmode
00001234	00002506	Hint/Name RVA	006A __p__commode
00001238	000024F6	Hint/Name RVA	009D _adjust_fdiv
0000123C	000024E2	Hint/Name RVA	0083 __setusermatherr
00001240	000024D6	Hint/Name RVA	010F _initterm
00001244	000024C6	Hint/Name RVA	0058 __getmainargs
00001248	000023B8	Hint/Name RVA	0049 __CxxFrameHandler
0000124C	000023CC	Hint/Name RVA	02C5 strstr
00001250	000023D6	Hint/Name RVA	02BE strlen
00001254	000023E0	Hint/Name RVA	0297 memcpy
00001258	000023EA	Hint/Name RVA	0010 ??3@YAXPAX@Z
0000125C	000023FA	Hint/Name RVA	025D fread
00001260	00002402	Hint/Name RVA	02B8 strcmp
00001264	0000240C	Hint/Name RVA	02B2 sprintf
00001268	00002416	Hint/Name RVA	02C3 strrchr
0000126C	00002420	Hint/Name RVA	0264 ftell
00001270	00002428	Hint/Name RVA	0262 fseek
00001274	00002430	Hint/Name RVA	024C fclose
00001278	0000243A	Hint/Name RVA	0257 fopen

4. Using the PEview application, I examined the strings within the malware sample and discovered that the strings in the .data section were particularly noteworthy. I came across phrases like “D-o-w-n-l-o-a-d-i-n-g-f-i-l-e%s********%d@@@@@@@%d” and “Begin Download,” suggesting the malware engages in downloading activities. Similarly, phrases such as “U-p-l-o-a-d-f-i-l-e%s********%d” and “Begin Upload” imply that the malware also has capabilities

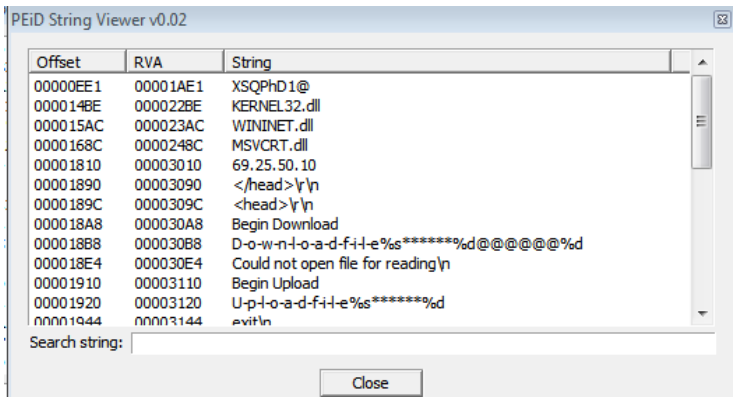
Anvitha Yerneni
ayernen@g.clemson.edu

to upload files. The exact name of the file involved remains unclear due to the obfuscated strings. Furthermore, the malware makes reference to “wuauclt.exe,” a known legitimate file associated with Windows Automatic Updates. Any alterations to this file could serve as an indicator of compromise, signaling that the system may be infected with the malware.

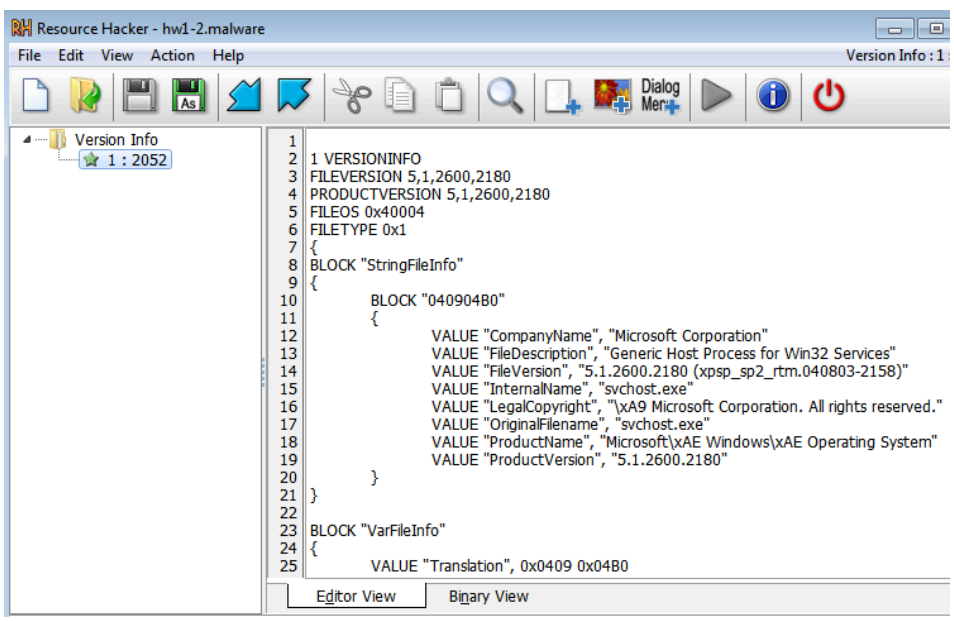


Next as shown below, I directed my analysis towards the PEiD String Viewer version 0.02 to delve deeper into these specific strings. From the provided screenshot, it's evident that there are numerous strings that warrant further examination. The initial string that caught my attention is “XSQPhD1@”, which might seem like nonsensical characters to the layperson; however, I hypothesize that this string serves a specific function. It could potentially be used as a credential—either a username or a password—for gaining access to a distant server. Regarding remote servers, the IP address “69.25.50.10” is notable because the malware might be utilizing it to transfer files back and forth. Since our prior analysis of the imports revealed the program's capability for such actions, interacting with this IP address could signal that a system is compromised by this malware, serving as an indicator of network-based infection.

Anvitha Yerneni
ayernen@g.clemson.edu



Finally, it's worth noting that during my examination of the malware sample, I used the Resource Hacker tool to inspect its resources. The screenshot provided earlier showcases this process. From this inspection, the only significant detail I was able to gather was the original filename of the resource, which is "svchost.exe".



Anvitha Yerneni
ayernen@g.clemson.edu

Problem 3 – Get the IAT hooking code (iathooking.cpp) run on Windows

1. Runned this code on a 64-bit Windows machine:

```cpp
// IATHooking.cpp : This file contains the 'main' function. Program execution begins and ends there
//

#include <iostream>
#include <Windows.h>
#include <winternl.h>

// define MessageBoxA prototype
using PrototypeMessageBox = int (WINAPI*)(HWND hWnd, LPCSTR lpText, LPCSTR lpCaption, UINT uType);

// remember memory address of the original MessageBoxA routine
PrototypeMessageBox originalMsgBox = MessageBoxA;

// hooked function with malicious code that eventually calls the original MessageBoxA
int hookedMessageBox(HWND hWnd, LPCSTR lpText, LPCSTR lpCaption, UINT uType) {
    MessageBoxW(NULL, L"Ola Hooked from a Rogue Senor .o.", L"Ola Senor o/", 0);
    // execute the original NessageBoxA
    return originalMsgBox(hWnd, lpText, lpCaption, uType);
}
```

2. **Environment**
   - Operating System: Windows 10 or newer (64-bit version).
   - User Permissions: Administrator rights may be required to install software or make certain changes to the system.

Tools

- Integrated Development Environment (IDE): Visual Studio 2019 or newer. The Community Edition is free and sufficient for most development tasks.
- Compiler: MSVC (Microsoft Visual C++) that comes with Visual Studio.

Dependencies

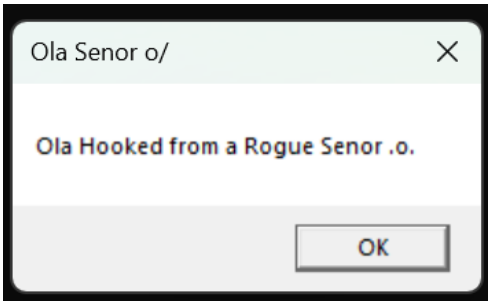
- Windows SDK: The latest version of the Windows 10 SDK, which is typically included with Visual Studio installations.
- C++ Standard Library: Comes with the MSVC compiler.
- Windows API: For functions like MessageBoxA, which are part of the Windows API, and included in the Windows SDK.

3. Result after Building the project : ctrl + shift +B

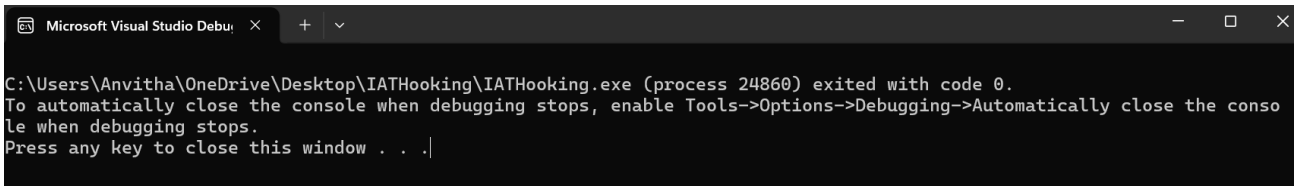
```
Output
Show output from: Build
Build started at 7:20 PM...
========== Build: 0 succeeded, 0 failed, 1 up-to-date, 0 skipped ==========
========== Build completed at 7:20 PM and took 00.200 seconds ==========
```

Result after Running the Project: F5

Below are the message boxes which appear as a result of IAT hooking



The screenshot provided below shows that the program has run and exited with code 0, which typically indicates that the program has finished execution without errors. However, there's no output on the console related to the IAT hooking or MessageBox display.

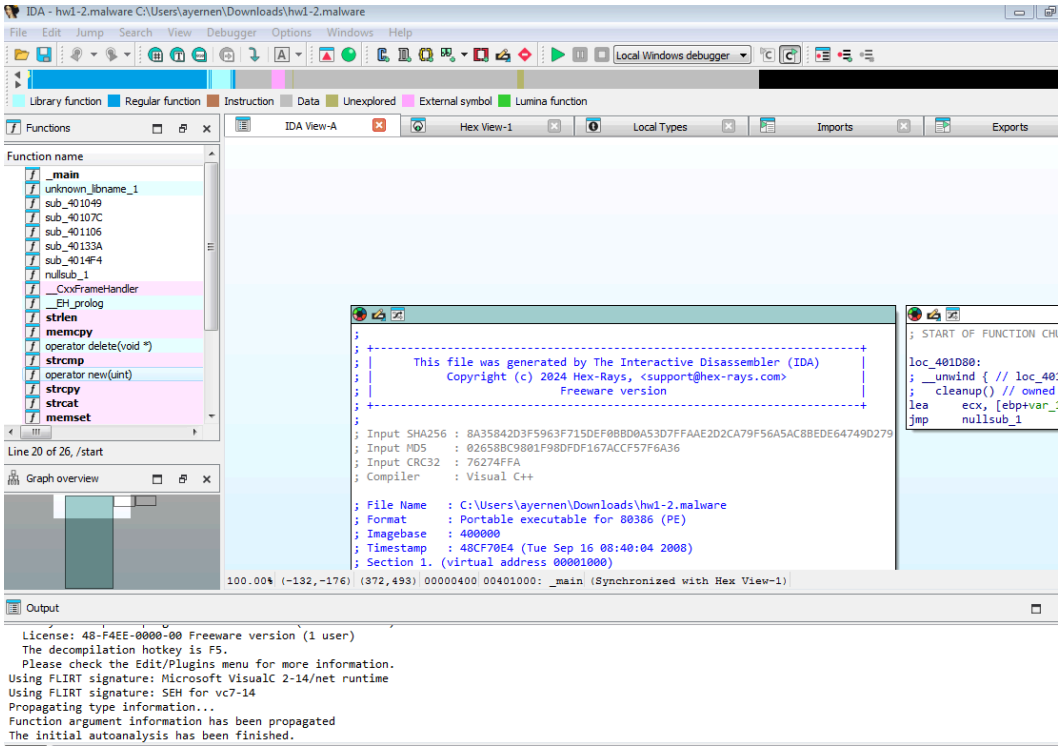


NO modifications were made to the code to compile and run it

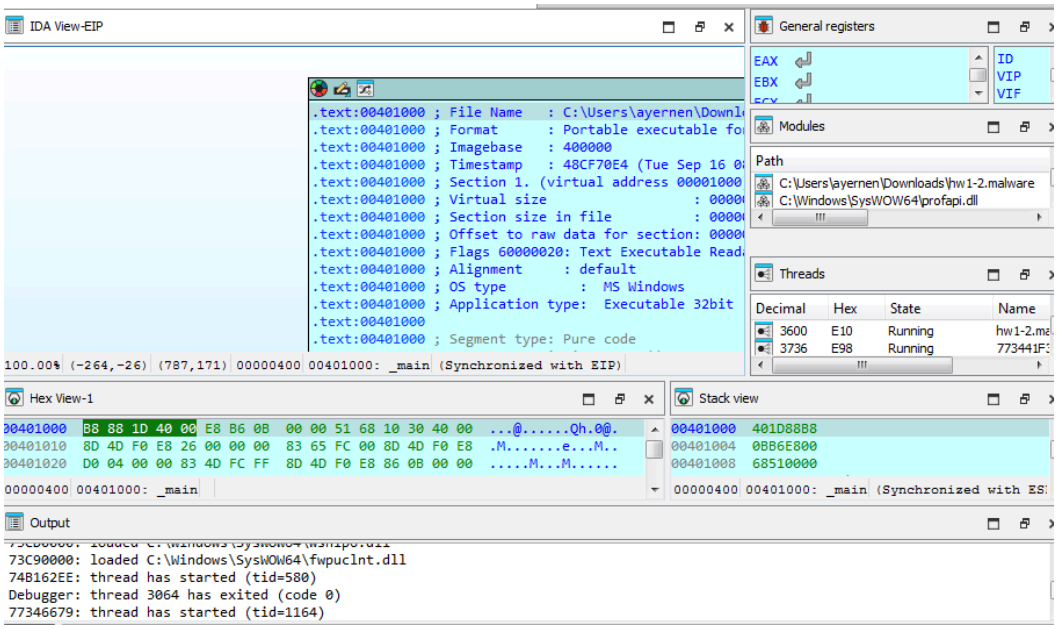
Anvitha Yerneni
ayernen@g.clemson.edu

Problem 4 – Basic dynamic analysis

1. Before running the code output:



After Running the malware:

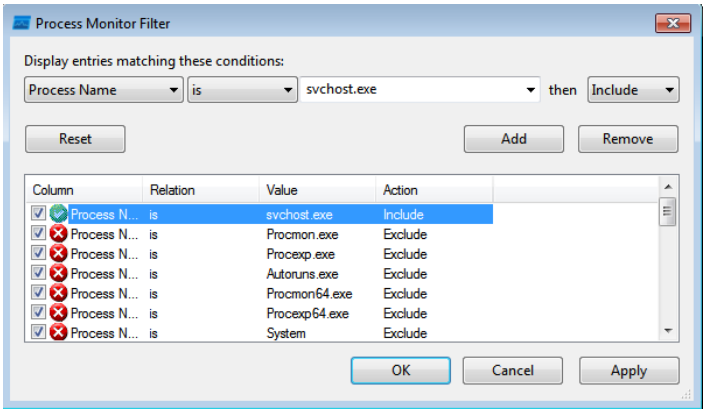


Anvitha Yerneni
ayernen@g.clemson.edu

The output pane shows that certain system DLLs have been loaded, such as winmm.dll and fwpclnt.dll. These are normal Windows DLLs; winmm.dll is related to the Windows multimedia framework, and fwpclnt.dll is related to the Windows Filtering Platform. This does not indicate anything specific about the malware's behavior.

After disassembling the malware sample using the IDA Freeware tool, I established several breakpoints at points that seemed significant and executed the program. Unexpectedly, there were no apparent changes or activities, indicating that a deeper examination of the system is necessary. This can be accomplished with the aid of the Process Monitor utility to gain a better understanding of the malware's behavior.

2.



In the screenshot provided above , the Process Monitor tool was used to narrow down the analysis by applying a filter based on the process name. Specifically, the filter was set to display only the activities of the process named "svchost.exe".

3.

```
call    esi ; CreatePipe
push    44h ; 'D'        ; Size
lea     eax, [ebp+StartupInfo]
push    ebx              ; Val
push    eax              ; void *
call    memset
mov     eax, [ebp+var_24]
add     esp, 0Ch
mov     [ebp+StartupInfo.hStdInput], eax
mov     eax, [ebp+hWritePipe]
mov     [ebp+StartupInfo.hStdError], eax
mov     [ebp+StartupInfo.hStdOutput], eax
lea     eax, [ebp+ProcessInformation]
mov     esi, offset aWuaucltExe ; "wuauclt.exe"
lea     edi, [ebp+CommandLine]
push    eax              ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
mov     [ebp+StartupInfo.dwFlags], 101h
movsd
```

While examining this malware sample using the IDA tool, I identified a signature associated with a host system, specifically an executable file. The executable, as displayed in the screenshot

Anvitha Yerneni
ayernen@g.clemson.edu

provided, it is named "wuauclt.exe." This file is commonly recognized as the executable responsible for Windows Automatic Updates, a legitimate Windows process.

4.

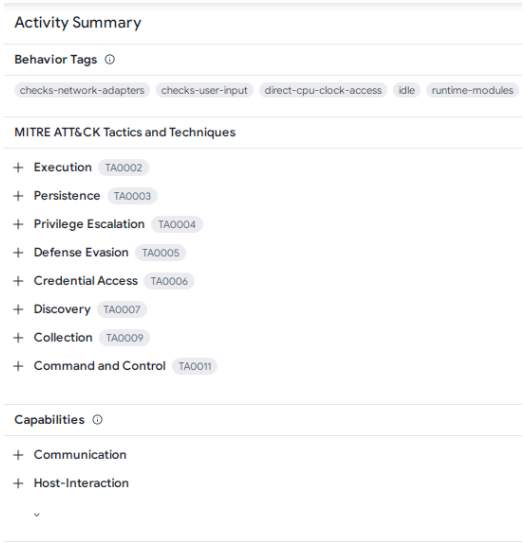
```
; FUNCTION CHUNK AT .text:00401D80 SIZE 00000012 BYTES

; __unwind { // loc_401D88
mov     eax, offset loc_401D88
call    __EH_prolog
push    ecx
push    offset a69255010 ; "69.25.50.10"
lea     ecx, [ebp+var_10]
call    unknown_libname_1 ; Microsoft VisualC 2-14/net runtime
;   try {
and     [ebp+var_4], 0
lea     ecx, [ebp+var_10]
call    sub_4014F4
;   } // starts at 401018
or      [ebp+var_4], 0FFFFFFFFh
lea     ecx, [ebp+var_10]
call    nullsub_1
mov     ecx, [ebp+var_C]
xor     eax, eax
mov     large fs:0, ecx
leave
retn
; } // starts at 401000
_main endp
```

While examining this malware specimen using IDA, I encountered a signature pertaining to networking, specifically an IP address. The address, "69.25.50.10," is highlighted in the program within the screenshot provided. This program is designed to use this IP address to establish a connection to an external server, enabling it to download and upload data.

5. This was my initial experience with reverse-engineering malware, and it presented several challenges. I had to familiarize myself with various tools necessary for the analysis, and setting them up on a Windows 7 Virtual Machine was not straightforward. During the dynamic analysis phase, I had to make do with the IDA Freeware version, which lacks several features found in IDA Pro. This limitation hindered my analysis. Access to a Windows 7-compatible IDA Pro license, rather than the Linux version we were given, would have facilitated a more efficient analysis process.

6. After completing both static and dynamic analyses, a clearer picture has emerged regarding the objectives of the malware depicted in the screenshot. The VirusTotal behavior section reveals that this malware engages in multiple malicious activities. It sets up a reverse shell, tampers with existing Windows services like wuauclt.exe (which handles Windows Updates), initiates new processes, logs system events, and establishes communication with a remote host at the IP address 69.25.50.10. These functions are indicative of a threat actor's intent to exfiltrate sensitive information from the affected machine and potentially gain extensive control over it. The malware's author has also employed sophisticated obfuscation methods to evade detection and operates discreetly to undermine the user's security without their knowledge.

Anvitha Yerneni
ayernen@g.clemson.edu



— THE END —