Malware Analysis

Minh Dao Nguyen

Abstract

The report is about a malware on the virus share library. Using both static and dynamic analysis, we determine that this is malware is a dropper and Trojan type that target window machine. It spawns malicious exe files that send out profanity and inappropriate language through social media mainly Skype. It also hide its present as well.

1 Introduction

This paper is an analysis of a malware. We will be using static and dynamic analysis through many tools such as PE Detective, CTF Explorer, Ghidra, WiredShark, etc. The malware will be run on a Window 7 virtual machine in order to performed dynamic analysis while protecting the host machine and prevent the malware from escaping into the network.

2 Static Analysis

2.1 Unpacking

2.1.1 PE detective

Using PE Detective from Win7, we can scan the malware for any sign of packaging. This is done to reduce the size and obfuscate the malware. In the case of the malware is packed, we can unpack it to have the full malware for analysis. The scan implies that there are no packing done on the malware.

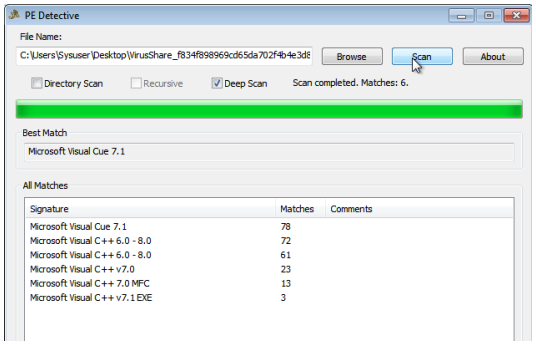

Figure 1: PE Detective scan of Malware in Win7 Vm

However, it does indicated that the malware contain components from Microsoft Visual C++ package. This could mean that the malware is built using C++. Overall, there is no unpacking need to be done on the malware.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2.1.2 CFF Explorer

Using CFF Explore helps show the header information of the malware and other basic information. The malware is a Window executable 32 bit. The file size is the same as the PE size 484 KB. CFF shows the malware MD5 Hash and SHA-1 value.

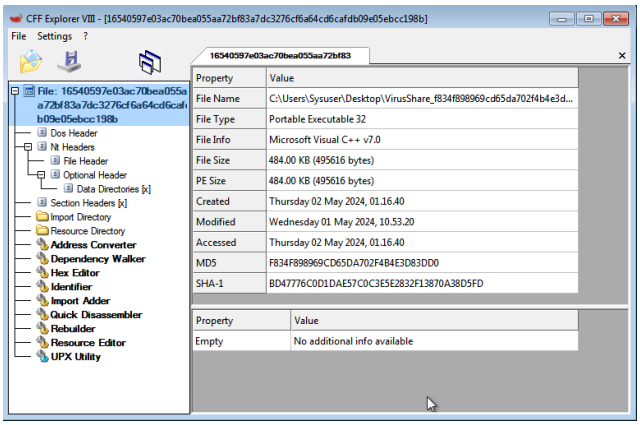

Figure 2: CFF Explorer scan of the Malware

2.2 Strings

Upon inspecting the Strings in the malware using the command strings malware > malString.txt, the produced file is 58KB.
IsDebuggerPresent is presented in the string file. This could indicate that the malware has anti-debugger functionality to prevent analysis done on it. It would be important to take into consideration while inspecting the executable program. This will help to NOP out or adjust the instruction accordingly during dynamic analysis.
ReadFile and WriteFile functions are also presented. The malware could be writting and dropping file in the computer. Inspecting the memory during running malware would be needed.
ShellExecuteA command in the string file could indicate that the malware will spawn another program that would perform malicious activities.
There is no interesting URL presented in the strings. There is also no other file name.

2.3 Import Libraries

There are only 2 imported libraries in this malware, KERNEL32.DLL and SHELL32.DLL. These are basic window API dll. This also means that it also can run on most window machine.What interesting is that there is no MS Visual C++ package presented even though it is detected to have it.

2.4 Ghidra Analysis

Putting the function into Ghidra, we can see the assembly of the malware as well as its decompiled code.

2.4.1 entry

The entry function of the malware is typical for a program. It initialize the stack, virtual memory, etc. The main function is FUN_004016a0 at the address 00405cc8. It takes in 3 parameters. We will analysis the main function more closely.

2.4.2 Main Function (FUN_004016a0)

The main function has some interesting functions call in it. It has 2 infinitive loop (while (true)) at 00401c8a and 00404449. The loop at 00401c8a call ShellExecuteA at 00401d6d. The malware

could be trying to write or read something in the computer, so stopping before the and inspect the parameters during the dynamic analysis would be beneficial.

2.4.3 Reverse Engineer Main Function

Throughout the process, we will rename the variables and put in comments in the decompiled code. Since local_25c = param_3, we will rename local_25c to param3. There are a lot of variables being assigned null variable '\0', so we will rename them to nullVal_num.
Instructions from 004018ec to 00401920 seem to do nothing. It make chnage on a variable, but the variable is never used again. Summary of what other function in main does, and rename them accordingly:

- FUN_00404890 -> deCapitalized: converts uppercase letters in the string to lowercase
- FUN_00401000 -> StrConcat: concatenates second string to end of first string
- FUN_004051e0 ->

```
if (*(char *)param3 == '_') {
    // Initialize variables HEX Values
    int var_1 = 0x6b6f7764;  // Decimal value: 1802206740
    int var_2 = 0x7164716e; // Decimal value: 1919252078
    int var_3 = 0x6477642d;  // Decimal value: 1684828781
    int var_4 = 0;
    int var_5 = 0;

    while( true ) {
        // Iterate over memory locations until null but this LOOP does nothing
        for (int* current = &var_1; *(char *)current != '\0'; current = (int*)((int)
        }

        if ((current - var_1) <= var_5) break;

        // Increment the character located at the calculated memory address by 1
        *(char *)((int)&var_1 + var_5) = *(char *)((int)&var_1 + var_5) + '\x01';

        var_5 = var_5 + 1;
    }

    ShellExecuteA(0, 0, &var_1, &stringMemory_1, 0, 1);
}
```

After analysis, this loop is what create another executable with a string gerenated randomly.

2.4.4 Junk Instruction

Interesting there are alot of junk functions in the program.

```
                        ...
00401960 b0 6f          MOV     AL,0x6f
00401962 b0 ff          MOV     AL,0xff
00401964 b0 f5          MOV     AL,0xf5
00401966 b0 f5          MOV     AL,0xf5
                        ...
```

These instructions moving some bytes into the the register AL. However, the register is never used and the byte keep replacing itself with random values. There are multiple places where these junk instruction occurs. It could be an error in the malware and compiler or purposely placed to obstruct the code.

3 Dynamic Analysis

3.1 Network Traffic

The URL that the malware trying to calls are:	www.ebay.com www.adobe.com www.yahoo.com www.facebook.com The strings in	www.baidu.com www.blogger.com www.youtube.com www.google.com	www.imdb.com www.wikipedia.org www.myspace.com

3.2 Process Monitor

Looking at the process monitor, the malware (virus) creates a file called gegcamvmrn.exe. It then perform alot of execution to write into that file. The malware then execute the gegcamvmrn process. It also try e to connect to Internet Explorer. The gegcamvmrn.exe tries to make calls to all the URL

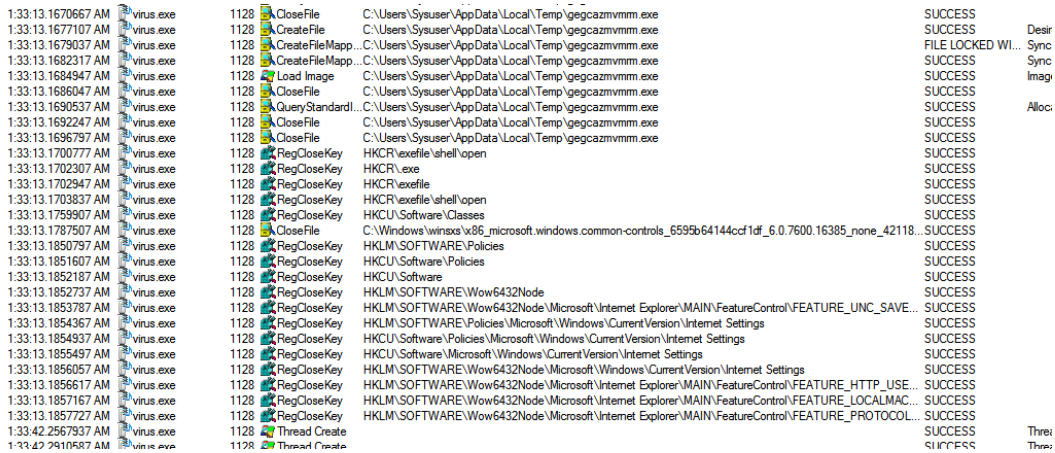

Figure 3: ProcMoc monitors the virus.exe

above. It also tries to open the digger registry in `HKCR/Software`

The virus function creates new thread then exit after 5 minutes.

The malware also try to edit the registies of the computer making it prompts warning. The function also

3.3 OllyDBG

OllyDBG can't run the malware due to underlying debugger check. The VM is killed everything the malware is loaded into OllyDBG.

3.4 Dropped Executable

3.5 Dropped Malware: gegcazmvmrm.exe

The output strings of this program indicates a lot of what it does. Firstly, it contains URL to different social media. This is where the network calls are made with imp_gethostbyaddr.
The part where the malware hide itself is at 0040da4 where it calls Sleep(10800000) along with other malicious functions in a while loop infinitely. The malware will sleep for approximately 3 hours, hiding itself from the user and the process monitors.
This malware also has profanity sentences as strings in its program. This is to send to Skype.

3.6 Dropped Malware: npbcls.exe

This executable try to steal user information. In address 0041e94d, the funciton FUN_0041e94d make multiple call to the FUN_0041d048 with "GET PROFILE ..." and "GET CURRENTUSERHANDLE".

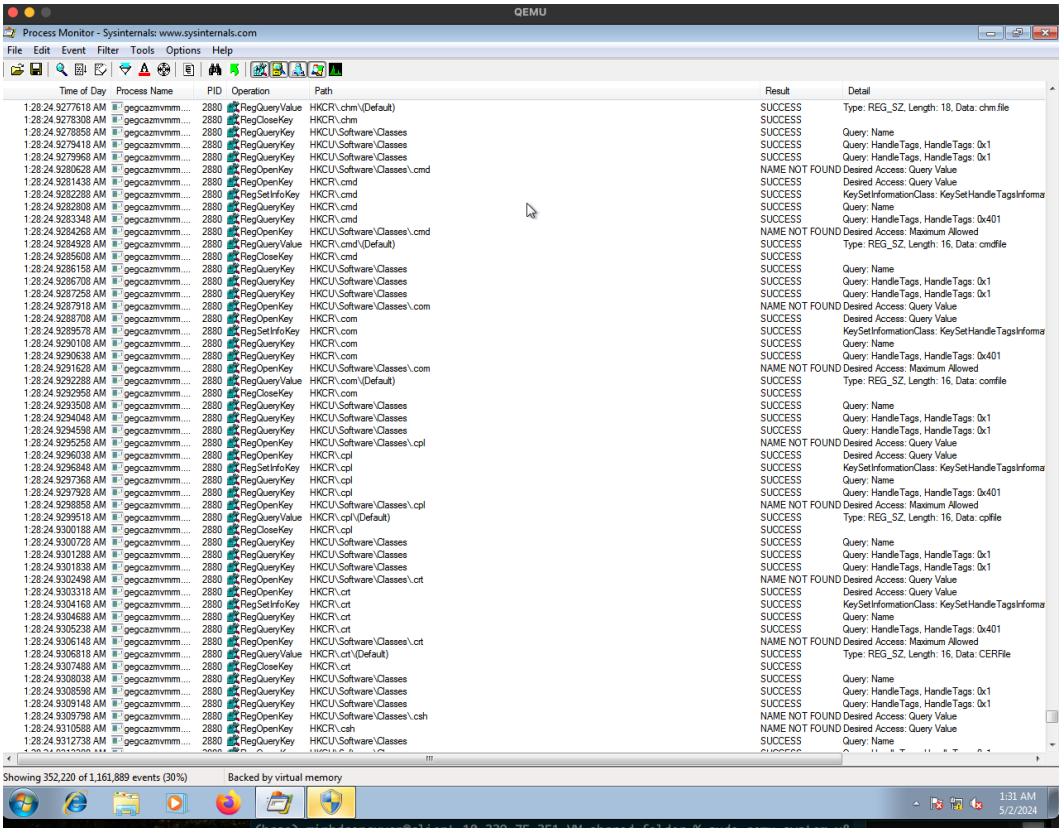

Figure 4: ProcMoc monitors the gegcamvmrn.exe

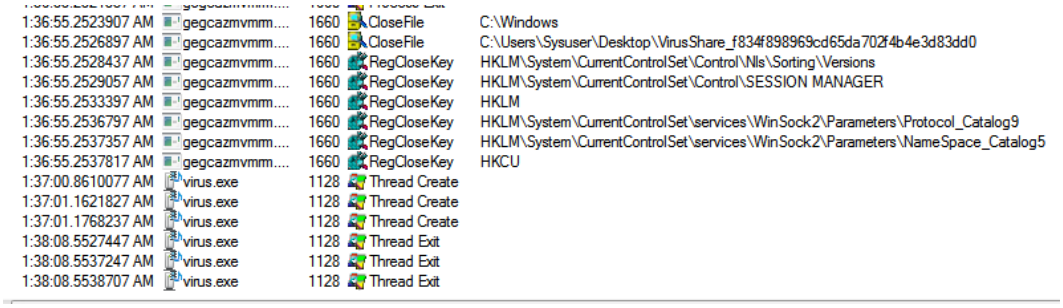

Figure 5: ProcMoc monitors the virus.exe

```
void FUN_0041e94d(void)
{
  Sleep(10);
  FUN_0041d048("GET CURRENTUSERHANDLE");
  Sleep(10);
  FUN_0041d048("GET PROFILE PSTN_BALANCE");
  Sleep(10);
  FUN_0041d048("GET PROFILE FULLNAME");
  Sleep(10);
  FUN_0041d048("GET PROFILE BIRTHDAY");
  Sleep(10);
  FUN_0041d048("GET PROFILE SEX");
  Sleep(10);
  FUN_0041d048("GET PROFILE COUNTRY");
```

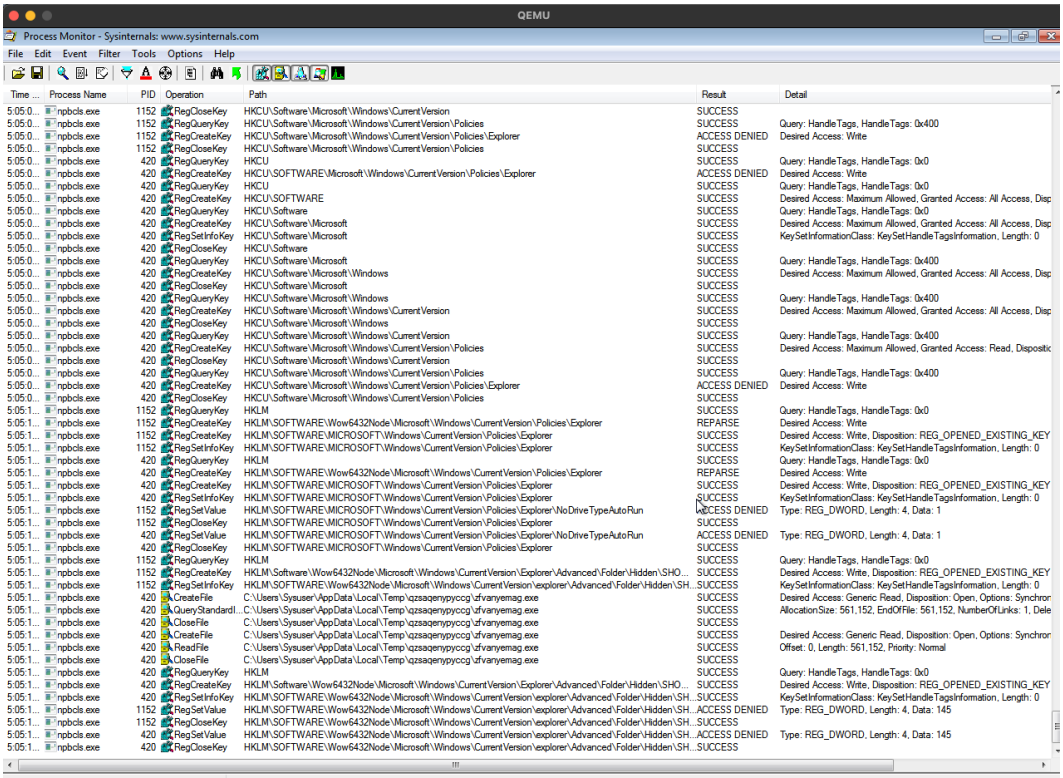

Figure 6: ProcMoc monitors the npbcls.exe

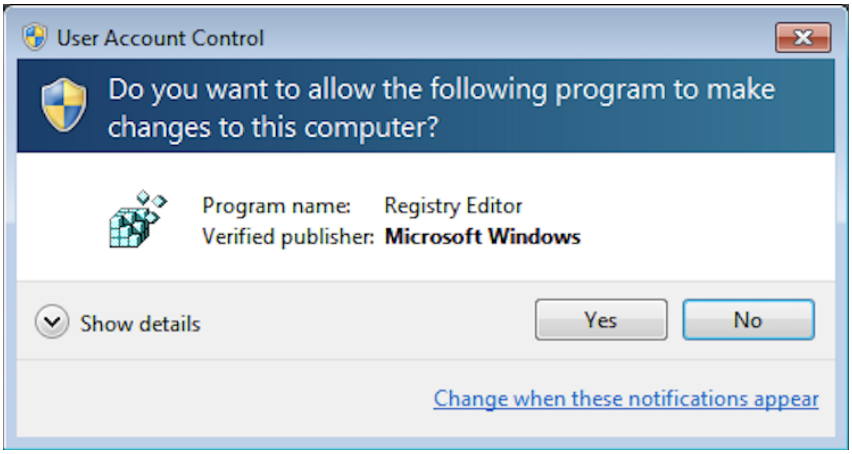

Figure 7: ProcMoc monitors the npbcls.exe

```
Sleep(10);
FUN_0041d048("GET PROFILE IPCOUNTRY");
Sleep(10);
FUN_0041d048("GET PROFILE CITY");
Sleep(10);
FUN_0041d048("GET PROFILE PHONE_HOME");
Sleep(10);
FUN_0041d048("GET PROFILE PHONE_OFFICE");
Sleep(10);
FUN_0041d048("GET PROFILE PHONE_MOBILE");
Sleep(10);
```

```
  FUN_0041d048("GET PROFILE HOMEPAGE");
  Sleep(10);
  FUN_0041d048("GET PROFILE ABOUT");
  Sleep(10);
  FUN_0041d048("GET PROFILE MOOD_TEXT");
  Sleep(10);
  FUN_0041d048("GET CURRENTUSERHANDLE");
  Sleep(10);
  FUN_0041d048("GET PROFILE TIMEZONE");
  Sleep(10);
  return;
}
```

The function FUN_0041d048 add those query to DAT_0045b0e. The function FUN_0040a949 at 0040a949 use that data to query to the register. This is where the Register Edit warning comes up.

4 Conclusion

4.1 How to delete Malware

The best way is to delete the virus.exe file. gegcazmvmrm.exe and npbcls.exe are localted in AppData/Local/Temp. Deleting all 3 executable should delete the process even when it is sleeping. Process Explorer should also be monitored every 3 hours in case some of the malware was not deleted and wake up to execute.

4.2 Summary

This Malware is a dropper and Trojan type of malware where it does malicious activities to the computer as well as hide itself by performing those activities through a different process.
This malware main purpose is to spend profanity text message throught the network, mainly through Skype and it also try to steal user information.

5 Tool for analysis assisted

To assist with the analysis of the decompiled code. I have written a Python script that use OpenAI's AI NLP api to assit with the understand of the code. The script is run in user's environment. It connect to Ghidra using ghidra_bridge. It then get the decompiled code wherever the user is at on Ghidra and set it to the AI. The response will be the explanation of the code. It will be printed in the terminal as well as add a comment to the top of the function in Ghidra

```
import requests
import ghidra_bridge

gb = ghidra_bridge.GhidraBridge(namespace=globals(), hook_import=True)

from ghidra.util.task import TaskMonitor
from ghidra.app.decompiler import DecompInterface

API_KEY = 'YOUR OPEN AI KEY'

def getDecompiledFunc():
    monitor = TaskMonitor.DUMMY
    decompiler = DecompInterface()

    decompiler.openProgram(currentProgram)

    function = getFunctionContaining(currentLocation.getAddress())
```

```
    return decompiler.decompileFunction(function, 30, monitor).
            getDecompiledFunction().getC(), function.getName()


def openai_api(prompt):
    headers = {
        'Content-Type': 'application/json',
        'Authorization': f'Bearer {API_KEY}',
    }
    data = {
        'prompt': prompt,
        'max_tokens': 200,
        'temperature': 0.5,
        'stop': '\n',
    }
    res = requests.post('https://api.openai.com/v1/completions',
            headers=headers, json=data)
    resJSON = res.json()
    return resJSON['choices'][0]['text']


c_code, funcName = getDecompiledFunc()
print("Explanation: ")
prompt = f"This code is a malware that got decompiled in Ghidra into C code:
            \n {c_code} \n Please explain this code."
res = openai_api(prompt)

#Comment on the function in Ghidra
function = getFunction(funcName)
function.setComment(res)

print(res)
```

The reason the program is run outside of Ghidra environment is because all script in Ghidra is using Jython. This limit a lot of functionality for what the possibility of writting script. For example, it is very difficult to install some modules such as requests for API call. The module ghidra_bridge allows user to develop normal python 3 script and also connects to Ghidra and all its API call.

This tool could be improved a lot because there many things can be added onto this program, like tracing function and comments on each of them.