# Reverse Engineering Malware Assignment

# Sample Report

| | |
|---|---|
| Type | |
| Filename | |
| Md5hash | |
| URL Download | |
| VirusTotal | |

## Table of Contents

## Introduction

This report documents the analysis of the malware, SogouPY Config, using both advanced static and dynamic analysis techniques. SogouPY Config was downloaded from http://www.tekdefense.com/downloads/malware-samples/ (0.exe.zip) and executed in a safe and isolated environment for analysing purposes. A series of code analysis were conducted and documented in this report.

## 1. Lab Setup

The tools used to analyse the SogouPY Config malware include the VMware Workstation 14 Player, Interactive Disassembler Professional (IDA Pro) and OllyDBG.

It is important to ensure that the malware is isolated from the host's Operating System when analysing the malware to prevent any damages or unauthorised changes to the host system. Therefore, a virtual machine will be set up to execute and analyse the malware in a safe environment.

After setting up the virtual machine, the malware will then be disassembled by using IDA Pro and OllyDBG. They trace registers, recognize procedures, API calls, switches, tables, constants and strings, as well as locate routines from object files and libraries.

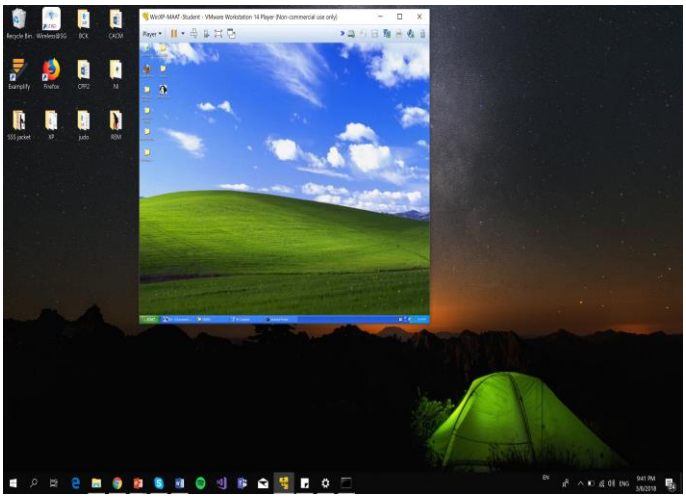
OllyDBG will also be used for the patching of the malware. The malicious codes will be replaced with no operation codes that will stop the actions performed by the malware.

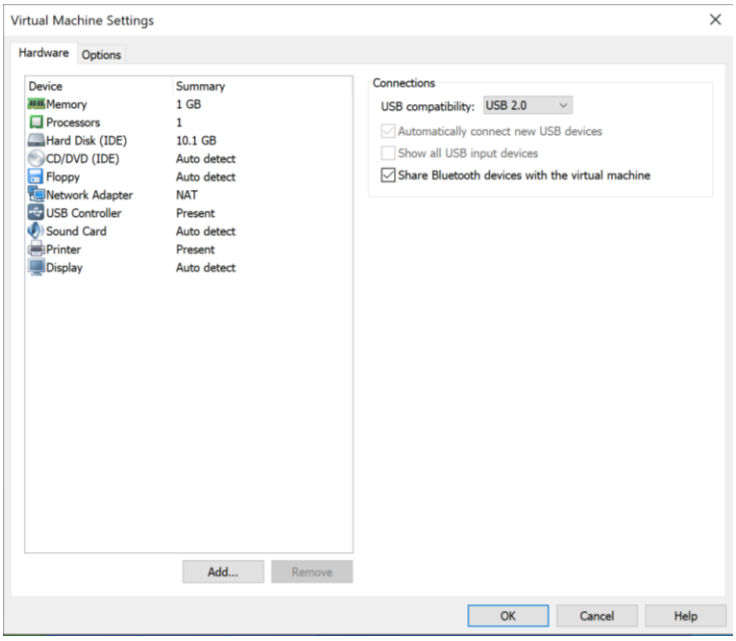
### 1.1. VMware Setup

**Host Operating System:** Windows 10

**Virtual Guest Operating System:** Windows XP

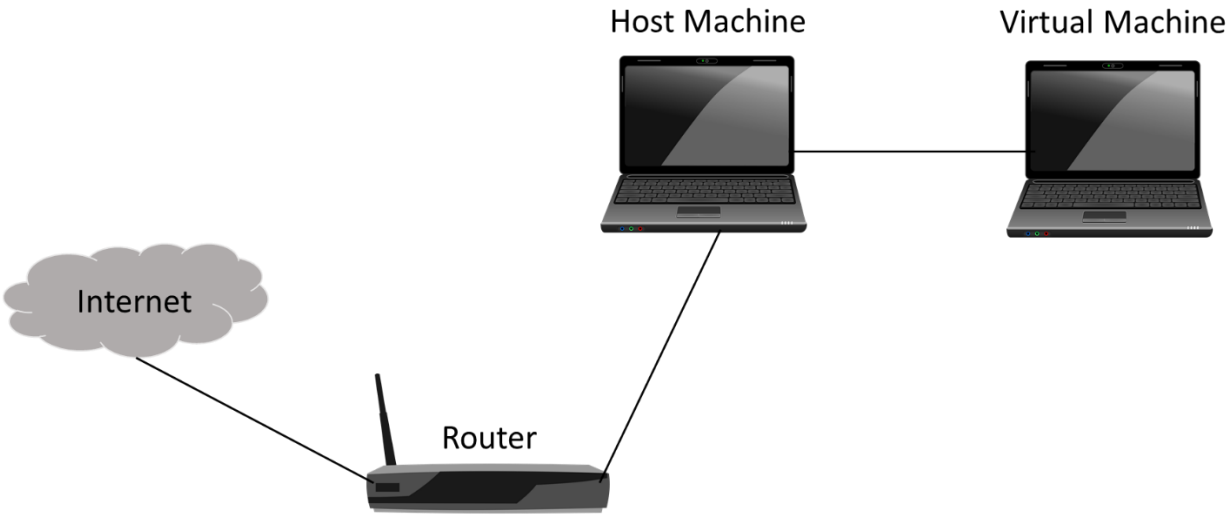
**Version of VMware:** Workstation 14 Player






VMware Workstation 14 player is installed and will be running on the Windows 10 host machine. The guest machine (running on the application VMware Workstation 14 Player) operates on Windows XP with 1GB memory, 10GB of hard disk space and uses the network address translation (NAT) network adapter mode.

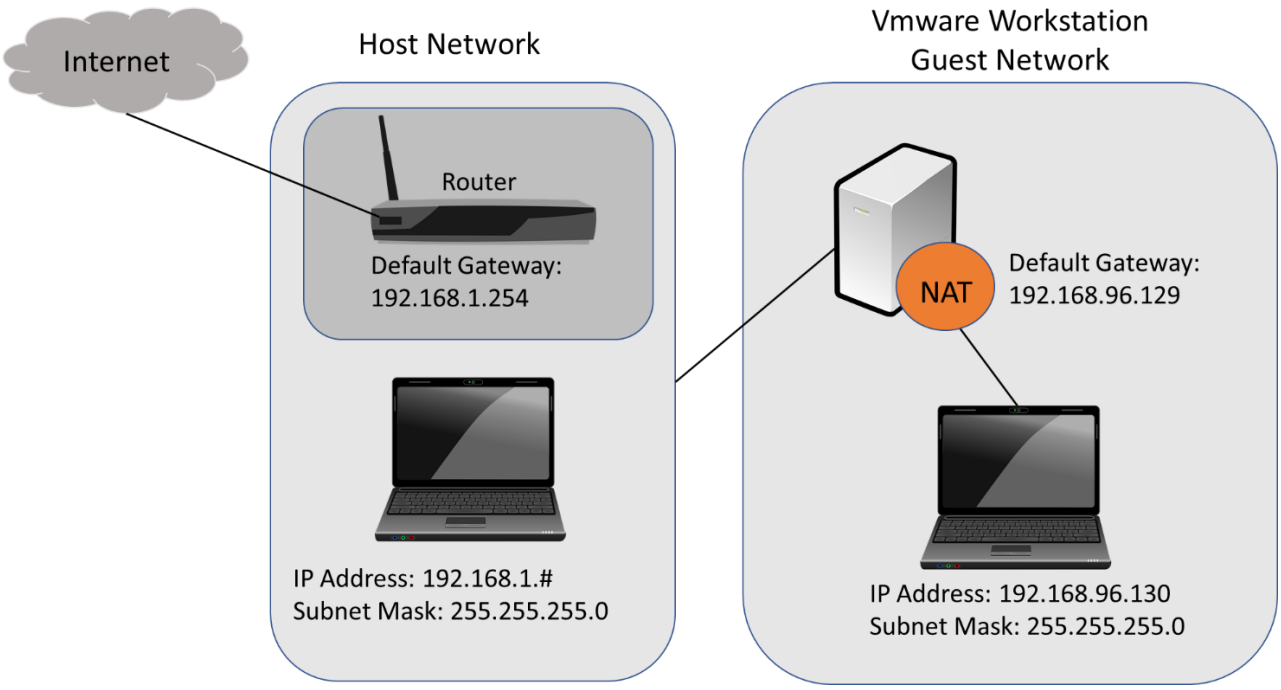
## 1.2. Network Diagram




The Windows XP virtual machine was set up on the host machine to provide a safe and isolated environment. Hence, any malicious actions or code executed by the malware during analysis will not affect the host machine. If the virtual machine were to get infected, it can be easily deleted and recreated to continue the analysis.

In addition, the host machine is connected to the router which has access to the internet. The virtual machine is running in the NAT network connection mode, which is used to share the same IP address as the host machine.

### 1.3. Network Configuration




This diagram depicts the connection and configuration of the network of the virtual guest machine running on the host machine.

## 2. Passive Information Gathering (IDA Pro)

### 2.1. Imported APIs

#### 2.1.1 FindResourceA

```
.idata:0040103C ; HRSRC __stdcall FindResourceA(HMODULE hModule,LPCSTR lpName,LPCSTR lpType)
.idata:0040103C                 extrn FindResourceA:dword ; DATA XREF: sub_4011D5+F↓r
```

**Purpose:** This function determines the location of a resource with the specified type and name in the specified module.

**Parameter:** hModule, lpName, lpType

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/ms908411.aspx

### 2.1.2 LoadResource

```
.idata:00401048 ; HGLOBAL __stdcall LoadResource(HMODULE hModule,HRSRC hResInfo)
.idata:00401048                 extrn LoadResource:dword ; DATA XREF: sub_4011D5+22↓r
```

**Purpose:** Retrieves a handle that can be used to obtain a pointer to the first byte of the specified resource in memory.

**Parameter:** hModule, hResInfo

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms648046(v=vs.85).aspx

### 2.1.3 LockResource

```
.idata:0040104C ; LPVOID __stdcall LockResource(HGLOBAL hResData)
.idata:0040104C                 extrn LockResource:dword ; DATA XREF: sub_4011D5+2B↓r
```

**Purpose:** Retrieves a pointer to the specified resource in memory.

**Parameter:** hResData

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms648047(v=vs.85).aspx

### 2.1.4 SizeOfResource

```
.idata:00401050 ; DWORD __stdcall SizeofResource(HMODULE hModule,HRSRC hResInfo)
.idata:00401050                 extrn SizeofResource:dword ; DATA XREF: sub_4011D5+36↓r
```

**Purpose:** Retrieves the size, in bytes, of the specified resource.

**Parameter:** hModule, hResInfo

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms648048(v=vs.85).aspx

### 2.1.5 DeleteFileA

```
.idata:0040105C ; BOOL __stdcall DeleteFileA(LPCSTR lpFileName)
.idata:0040105C                 extrn DeleteFileA:dword ; DATA XREF: sub_4011D5+42↓r
```

**Purpose:** Deletes an existing file.

**Parameter:** lpFileName

**MSDN reference:** https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-deletefilea

### 2.1.6 CreateFileA

```
data:00401060 ; HANDLE __stdcall CreateFileA(LPCSTR lpFileName,DWORD dwDesiredAccess,DWORD dwShareMode,LPSECURITY_ATTRIBUTES lpSecurityAttributes,DWORD dwCreationDispos
data:00401060                 extrn CreateFileA:dword ; DATA XREF: sub_4011D5+5B↓r
data:00401060                                         ; sub_401310+4B↓r
```

**Purpose:** Creates or opens a file or I/O device. The most commonly used I/O devices are as follows: file, file stream, directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, and pipe. The function returns a handle that can be used to access the file or device for various types of I/O depending on the file or device and the flags and attributes specified.

**Parameters:** lpFileName, dwDesiredAccess, dwShareMode, lpSecurityAttributes, dwCreationDisposition, dwFlagsAndAttributes, hTemplateFile

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-createfilea

### 2.1.7 WriteFile

```
.idata:00401064 ; BOOL __stdcall WriteFile(HANDLE hFile,LPCVOID lpBuffer,DWORD nNumberOfBytesToWrite,LPDWORD lpNumberOfBytesWritten,LPOVERLAPPED lpOverlapped)
.idata:00401064                 extrn WriteFile:dword   ; DATA XREF: sub_4011D5+72↓r
.idata:00401064                                         ; sub_401310+6D↓r ...
```

**Purpose:** Writes data to the specified file or input/output (I/O) device. This function is designed for both synchronous and asynchronous operation.

**Parameters:** hFile, lpBuffer, nNumberOfBytesToWrite, lpNumberOfBytesWritten, lpOverlapped

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-writefile

### 2.1.8 Sleep

```
.idata:0040106C ; void __stdcall Sleep(DWORD dwMilliseconds)
.idata:0040106C                 extrn Sleep:dword       ; DATA XREF: sub_4011D5+78↓r
.idata:0040106C                                         ; sub_4011D5+80↓r ...
```

**Purpose:** Suspends the execution of the current thread until the time-out interval elapses.

**Parameters:** dwMilliseconds

MSDN **Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/synchapi/nf-synchapi-sleep

### 2.1.9 FreeResource

```
.idata:00401070 ; BOOL __stdcall FreeResource(HGLOBAL hResData)
.idata:00401070                 extrn FreeResource:dword ; DATA XREF: sub_4011D5+83↓r
```

**Purpose:** Decrements (decreases by one) the reference count of a loaded resource. When the reference count reaches zero, the memory occupied by the resource is freed.

**Parameters:** hglbResource

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms648044(v=vs.85).aspx

## 2.1.10 CloseHandle

```
.idata:00401068 ; BOOL __stdcall CloseHandle(HANDLE hObject)
.idata:00401068                 extrn CloseHandle:dword ; DATA XREF: sub_4011D5+8A↓r
.idata:00401068                                         ; sub_401310+B3↓r ...
```

**Purpose:** Closes an open object handle.

**Parameters:** hObject [in]

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms724211(v=vs.85).aspx

## 2.1.11 GetModuleFileNameA

```
.idata:00401058 ; DWORD __stdcall GetModuleFileNameA(HMODULE hModule,LPSTR lpFilename,DWORD nSize)
.idata:00401058                 extrn GetModuleFileNameA:dword ; DATA XREF: sub_401310+30↓r
```

**Purpose:** Retrieves the fully qualified path for the file that contains the specified module. The module must have been loaded by the current process.

**Parameters**: hModule, lpFilename, nSize

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms683197(v=vs.85).aspx

## 2.1.12 RegCreateKeyExA

```
.idata:00401000 ; LONG __stdcall RegCreateKeyExA(HKEY hKey,LPCSTR lpSubKey,DWORD Reserved,LPSTR lpClass,DWORD dwOptions,REGSAM samDesired,LPSECURITY_ATTRIBUTES lpSe
.idata:00401000                 extrn RegCreateKeyExA:dword ; DATA XREF: sub_401271+51↓r
.idata:00401004 ; BOOL __stdcall ControlService(SC_HANDLE hService,DWORD dwControl,LPSERVICE_STATUS lpServiceStatus)
```

**Purpose**: Creates the specified registry key. If the key already exists, the function opens it.

**Parameters:** hKey, lpSubKey, Reserved, lpClass, dwOptions, samDesired, lpSecurityAttributes, phkResult, lpdwDisposition

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/winreg/nf-winreg-regcreatekeyexa

### 2.1.13 RegSetValueExA

```
.idata:00401010 ; LONG __stdcall RegSetValueExA(HKEY hKey,LPCSTR lpValueName,DWORD Reserved,DWORD dwType,const BYTE *lpData,DWORD cbData)
.idata:00401010                 extrn RegSetValueExA:dword ; DATA XREF: sub_401271+89↓r
```

**Purpose:** Sets the data and type of a specified value under a registry key.

**Parameters:** hKey, lpValueName, Reserved, dwType, lpData, cbData

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/winreg/nf-winreg-regsetvalueexa

### 2.1.14 RegCloseKey

```
.idata:00401014 ; LONG __stdcall RegCloseKey(HKEY hKey)
.idata:00401014                 extrn RegCloseKey:dword ; DATA XREF: sub_401271+92↓r
```

**Purpose:** Closes a handle to the specified registry key.

**Parameters:** hKey

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/winreg/nf-winreg-regclosekey

### 2.1.15 OpenSCManagerA

```
.idata:0040101C ; SC_HANDLE __stdcall OpenSCManagerA(LPCSTR lpMachineName,LPCSTR lpDatabaseName,DWORD dwDesiredAccess)
.idata:0040101C                 extrn OpenSCManagerA:dword ; DATA XREF: WinMain(x,x,x,x)+F6↓r
.idata:0040101C                                            ; Establish a connection to the service
.idata:0040101C                                            ; control manager on the specified computer
.idata:0040101C                                            ; and opens the specified database
```

**Purpose:** Establishes a connection to the service control manager on the specified computer and opens the specified service control manager database.

**Parameters:** lpMachineName, lpDatabaseName, dwDesiredAccess

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/winsvc/nf-winsvc-openscmanagera

### 2.1.16 OpenServiceA

```
.idata:00401018 ; SC_HANDLE __stdcall OpenServiceA(SC_HANDLE hSCManager,LPCSTR lpServiceName,DWORD dwDesiredAccess)
.idata:00401018                 extrn OpenServiceA:dword ; DATA XREF: WinMain(x,x,x,x)+107↓r
```

**Purpose:** Opens an existing service.

**Parameters:** hscManager, lpServiceName, dwDesiredAccess

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/winsvc/nf-winsvc-openservicea

### 2.1.17 ChangeServiceConfigA

```
.idata:00401020 ; BOOL __stdcall ChangeServiceConfigA(SC_HANDLE hService,DWORD dwServiceType,DWORD dwStartType,DWORD dwErrorControl,LPCSTR lpBinaryPathName,
.idata:00401020                 extrn ChangeServiceConfigA:dword
.idata:00401020                                         ; DATA XREF: WinMain(x,x,x,x)+121↓r
```

**Purpose:** Changes the configuration parameters of a service.

**Parameters:** hService, dwServiceType, dwStartType, dwErrorControl, lpBinaryPathName, lpLoadOrderGroup, lpdwTagId, lpDependencies, lpServiceStartName, lpPassword, lpDisplayName

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/winsvc/nf-winsvc-changeserviceconfiga

### 2.1.18 ControlService

```
.idata:00401004 ; BOOL __stdcall ControlService(SC_HANDLE hService,DWORD dwControl,LPSERVICE_STATUS lpServiceStatus)
.idata:00401004                 extrn ControlService:dword
.idata:00401004                                         ; DATA XREF: WinMain(x,x,x,x)+13D↓r
.idata:00401004                                         ; Send a control code to a Win32 service
```

**Purpose:** Sends a control code to a service.

**Parameters:** hService, dwControl, lpServiceStatus

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/winsvc/nf-winsvc-controlservice

### 2.1.19 StartServiceA

```
.idata:00401008 ; BOOL __stdcall StartServiceA(SC_HANDLE hService,DWORD dwNumServiceArgs,LPCSTR *lpServiceArgVectors)
.idata:00401008                 extrn StartServiceA:dword ; DATA XREF: WinMain(x,x,x,x)+155↓r
```

**Purpose:** Starts a service.

**Parameters:** hService, dwNumServiceArgs, lpServiceArgVectors

**MSDN Reference:** https://docs.microsoft.com/en-us/windows/desktop/api/winsvc/nf-winsvc-startservicea

### 2.1.20 GetWindowsDirectoryA

```
.idata:00401074 ; UINT __stdcall GetWindowsDirectoryA(LPSTR lpBuffer,UINT uSize)
.idata:00401074                 extrn GetWindowsDirectoryA:dword
.idata:00401074                                         ; DATA XREF: WinMain(x,x,x,x)+9E↓r
```

**Purpose:** Retrieves the path of the Windows directory.

**Parameters:** lpBuffer, uSize

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms724454(v=vs.85).aspx

### 2.1.21 LoadLibraryA

```
.idata:00401038 ; HMODULE __stdcall LoadLibraryA(LPCSTR lpLibFileName)
.idata:00401038                 extrn LoadLibraryA:dword ; DATA XREF: WinMain(x,x,x,x)+16F↓r
.idata:00401038                                         ; WinMain(x,x,x,x)+1BA↓r
```

**Purpose:** Loads the specified module into the address space of the calling process.

**Parameters:** lpFileName (a .dll file or .exe file)

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms684175(v=vs.85).aspx

### 2.1.22 GetProcAddress

```
.idata:00401034 ; FARPROC __stdcall GetProcAddress(HMODULE hModule,LPCSTR lpProcName)
.idata:00401034                 extrn GetProcAddress:dword
.idata:00401034                                         ; DATA XREF: WinMain(x,x,x,x)+176↓r
```

**Purpose:** Retrieves the address of an exported function or variable from the specified dynamic-link library (DLL).

**Parameters:** hModule, lpProcName

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms683212(v=vs.85).aspx

### 2.1.23 CreateEventA

```
.idata:00401030 ; HANDLE __stdcall CreateEventA(LPSECURITY_ATTRIBUTES lpEventAttributes,BOOL bManualReset,BOOL bInitialState,LPCSTR lpName)
.idata:00401030                 extrn CreateEventA:dword ; DATA XREF: WinMain(x,x,x,x)+18D↓r
```

**Purpose:** This function creates a named or an unnamed event object.

**Parameters:** lpEventAttributes, bManualReset, bInitialState, lpName

**MSDN Reference:** https://msdn.microsoft.com/en-us/library/ms919029.aspx

## 3. Code Analysis (IDA Pro)

### 3.1. Graph of Major Subroutines from Main Function

Our major subroutine is **WinMain**. 'WinMain' is the conventional name used for the application entry point. This subroutine is called by the **start** subroutine (

```
call    WinMain@16      ; WinMain(x,x,x,x)
```

). The graph below shows our major subroutine.

```
; Attributes: bp-based frame

; int __stdcall WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR lpCmdLine,int nShowCmd)
_WinMain@16 proc near

var_158= dword ptr -158h
Data= byte ptr -148h
ServiceStatus= _SERVICE_STATUS ptr -44h
WndClass= WNDCLASSA ptr -28h
hService= dword ptr  8
hPrevInstance= dword ptr  0Ch
lpCmdLine= dword ptr  10h
nShowCmd= dword ptr  14h

push    ebp
mov     ebp, esp
sub     esp, 148h
push    ebx
push    esi
push    edi
mov     esi, [ebp+hService]
push    0Ah
xor     eax, eax
pop     ecx
lea     edi, [ebp+WndClass]
rep stosd
push    2
xor     ebx, ebx
pop     edi
mov     [ebp+WndClass.lpfnWndProc], offset loc_4011D0
push    69h             ; lpIconName
push    esi             ; hInstance
mov     [ebp+WndClass.style], edi
mov     [ebp+WndClass.cbClsExtra], ebx
mov     [ebp+WndClass.cbWndExtra], ebx
mov     [ebp+WndClass.hInstance], esi
call    LoadIconA
```

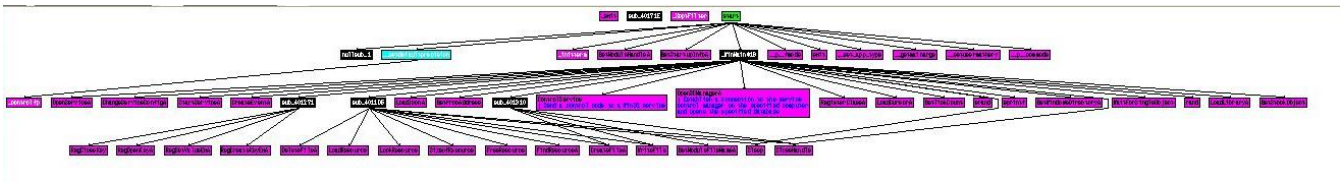
```
push    6Fh             ; lpCursorName
push    esi             ; hInstance
mov     [ebp+WndClass.hIcon], eax
call    LoadCursorA
push    edi             ; int
mov     [ebp+WndClass.hCursor], eax
call    GetStockObject
mov     [ebp+WndClass.hbrBackground], eax
lea     eax, [ebp+WndClass]
push    eax             ; lpWndClass
mov     [ebp+WndClass.lpszMenuName], 6Bh
mov     [ebp+WndClass.lpszClassName], offset aMyWin32 ; "My Win32"
call    RegisterClassA
mov     esi, GetTickCount
call    esi ; GetTickCount
push    eax             ; uSize
call    srand
push    40h
xor     eax, eax
pop     ecx
lea     edi, [ebp-147h]
mov     [ebp+Data], bl
rep stosd
stosw
stosb
lea     eax, [ebp+Data]
mov     [esp+158h+var_158], 104h
push    eax             ; lpBuffer
call    GetWindowsDirectoryA
mov     [ebp-145h], bl
call    rand
imul    eax, 64h
push    eax
lea     eax, [ebp+Data]
push    eax
lea     eax, [ebp+Data]
push    offset aSD_dll  ; "%s%d.dll"
push    eax             ; char *
call    sprintf
```

```
push    offset aCpp      ; "CPP"
lea     eax, [ebp+Data]
push    66h              ; nNumberOfBytesToWrite
push    eax              ; lpFileName
call    sub_4011D5
lea     eax, [ebp+Data]
push    eax              ; lpBuffer
call    sub_401310
add     esp, 20h
push    0F003Fh          ; dwDesiredAccess
push    ebx              ; lpDatabaseName
push    ebx              ; lpMachineName
call    OpenSCManagerA   ; Establish a connection to the service
                         ; control manager on the specified computer
                         ; and opens the specified database
push    0F01FFh          ; dwDesiredAccess
push    offset ServiceName ; "RemoteAccess"
push    eax              ; hSCManager
call    OpenServiceA
push    ebx              ; lpDisplayName
push    ebx              ; lpPassword
push    ebx              ; lpServiceStartName
mov     [ebp+hService], eax
push    ebx              ; lpDependencies
push    ebx              ; lpdwTagId
push    ebx              ; lpLoadOrderGroup
push    ebx              ; lpBinaryPathName
push    0FFFFFFFFh       ; dwErrorControl
push    3                ; dwStartType
push    110h             ; dwServiceType
push    eax              ; hService
call    ChangeServiceConfigA
push    6
xor     eax, eax
pop     ecx
lea     edi, [ebp+ServiceStatus.dwCurrentState]
mov     [ebp+ServiceStatus.dwServiceType], ebx
rep stosd
lea     eax, [ebp+ServiceStatus]
push    eax              ; lpServiceStatus
push    1                ; dwControl
```

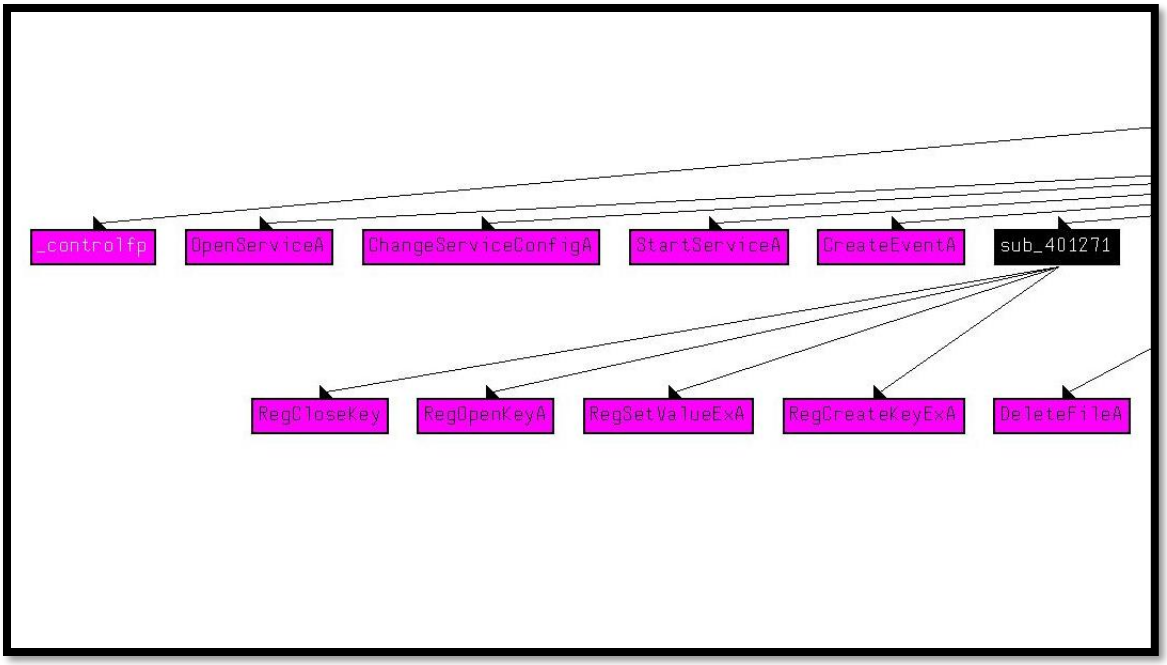
```
push    [ebp+hService]   ; hService
call    ControlService   ; Send a control code to a Win32 service
lea     eax, [ebp+Data]
push    eax              ; lpData
call    sub_401271
pop     ecx
push    ebx              ; lpServiceArgVectors
push    ebx              ; dwNumServiceArgs
push    [ebp+hService]   ; hService
call    StartServiceA
mov     edi, Sleep
push    1                ; dwMilliseconds
call    edi ; Sleep
push    offset ProcName ; "CloseServiceHandle"
push    offset LibFileName ; "Advapi32.dll"
call    LoadLibraryA
push    eax              ; hModule
call    GetProcAddress
push    [ebp+hService]
call    eax
push    1                ; dwMilliseconds
call    edi ; Sleep
push    offset Name      ; "Glable__Wait"
push    ebx              ; bInitialState
push    ebx              ; bManualReset
push    ebx              ; lpEventAttributes
call    CreateEventA
mov     ebx, eax
call    esi ; GetTickCount
```

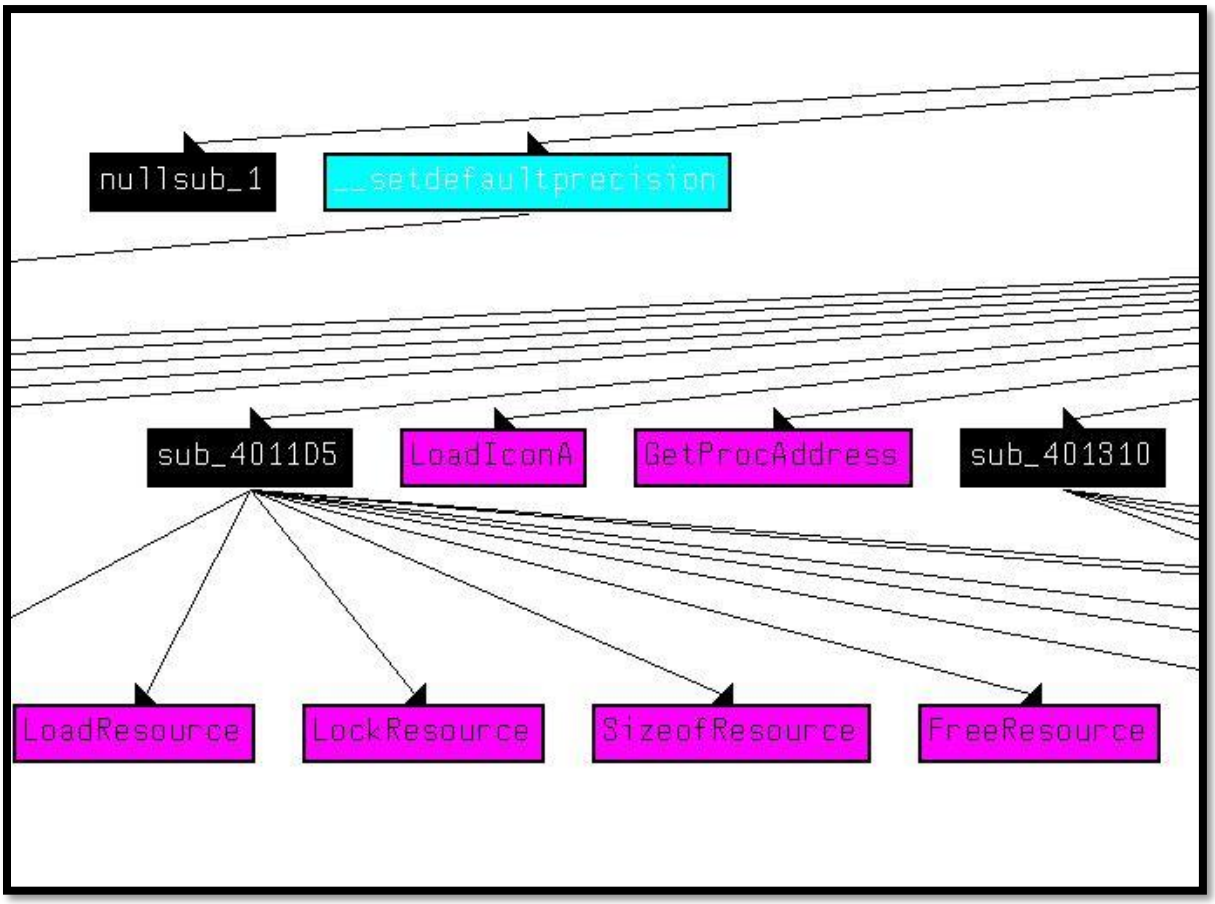
```
mov     [ebp+hService], eax

loc_401568:                     ; dwMilliseconds
push    64h
push    ebx                     ; hHandle
call    WaitForSingleObject
test    eax, eax
jz      short loc_40158E

call    esi ; GetTickCount
sub     eax, [ebp+hService]
cmp     eax, 1388h
jbe     short loc_401568

lea     eax, [ebp+Data]
push    eax                     ; lpLibFileName
call    LoadLibraryA

loc_40158E:                     ; dwMilliseconds
push    1
call    edi ; Sleep
push    ebx                     ; hObject
call    CloseHandle
pop     edi
pop     esi
xor     eax, eax
pop     ebx
leave
retn    10h

_WinMain@16 endp
```

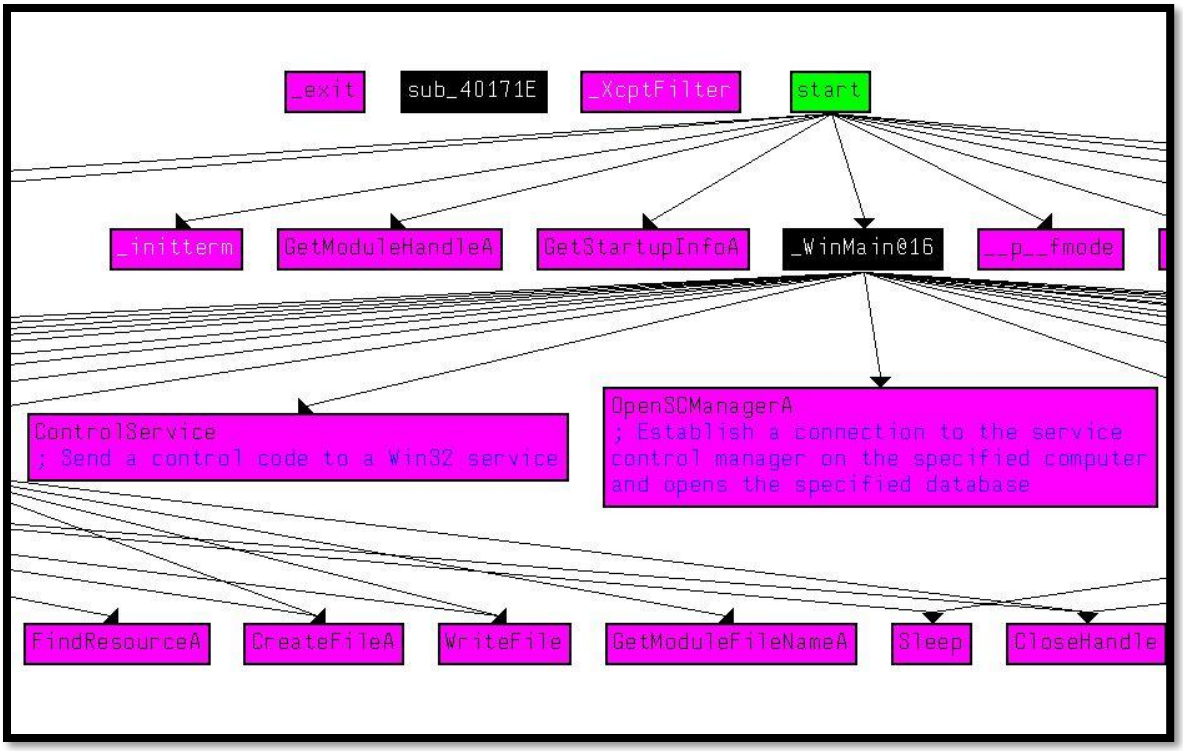
## Graph of Function Calls for Major Subroutine:

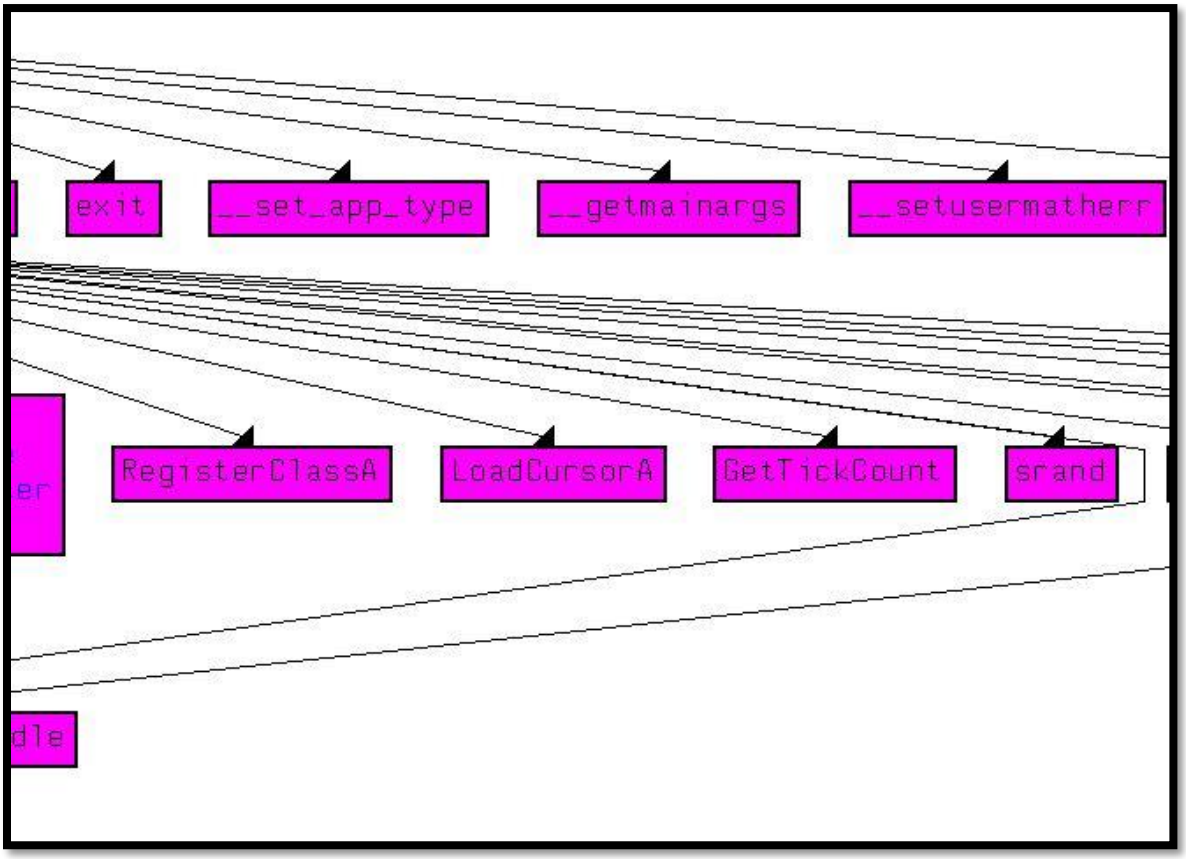


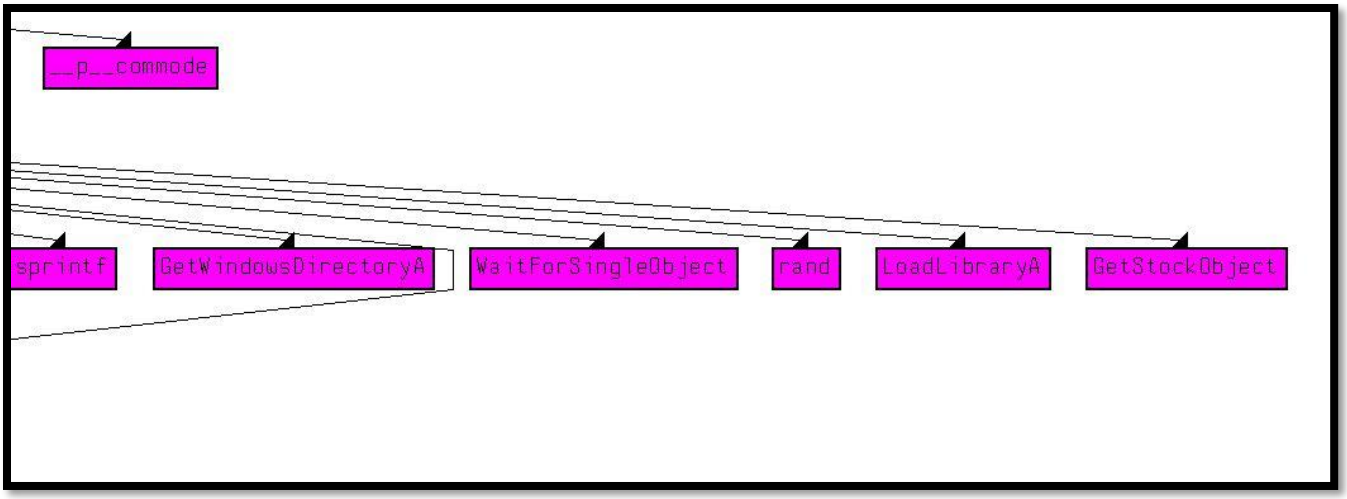
From top to bottom, it shows the graph spanning from left to right.


_controlfp
OpenServiceA
ChangeServiceConfigA
StartServiceA
CreateEventA
sub_401271
RegCloseKey
RegOpenKeyA
RegSetValueExA
RegCreateKeyExA
DeleteFileA


nullsub_1
__setdefaultprecision
sub_4011D5
LoadIconA
GetProcAddress
sub_401310
LoadResource
LockResource
SizeofResource
FreeResource


_exit
sub_40171E
_XcptFilter
start
_initterm
GetModuleHandleA
GetStartupInfoA
_WinMain@16
__p__fmode
ControlService
; Send a control code to a Win32 service
OpenSCManagerA
; Establish a connection to the service control manager on the specified computer and opens the specified database
FindResourceA
CreateFileA
WriteFile
GetModuleFileNameA
Sleep
CloseHandle


exit
__set_app_type
__getmainargs
__setusermatherr
RegisterClassA
LoadCursorA
GetTickCount
srand




As seen from the major subroutine, the malware calls 3 different subroutines (**sub_4011D5**, **sub_401310** and **sub_401271**).

**<u>Purpose of the major subroutine:</u>**

The LoadIconA, LoadCursorA, GetStockObject and RegisterClassA function is like part of the start up of the malware. For instance, the GetStockObject function in this malware retrieves a handle to one of the stock pens, brushes, fonts or palettes. The GetTickCount function retrieves the number of milliseconds that have elapsed since the system was started whereas the srand function sets the seed value for the pseudorandom number generator. The GetWindowsDirectoryA function is then called next. This function will retrieve the path of the Windows directory.

After the program called out two subroutines (sub_4011D5 and sub_401310), the OpenSCManagerA function is called. This function establishes a connection to the service control manager on the specified computer and opens the specified database. After which, a service named “RemoteAccess” is opened when the OpenServiceA function is called. The malware will then change the configuration settings of the service when the ChangeServiceConfigA is called out. With the help of the ControlService function, the program gets to send a control code to a Win32 service. A control code can be used to control a device.

The main subroutine will then call the sub_401271 subroutine. When the program returns back to the main subroutine, it will start a service when the StartServiceA function is called. This execution will be suspended for 1ms due to the Sleep function.

The program will then load the “Advapi32.dll” library, retrieve the address of an exported function (GetProcAddress function) and suspend its execution for 1ms. After which, the CreateEventA function is called. This function will create a named event object called “Glable__Wait” and wait until the specified object is in the signaled state when the WaitForSingleObject function is called. If test eax, eax returns a zero (jz), then the program will jump to the next location, loc_40158E. Otherwise, it will call out the GetTickCount and LoadLibrary function again.

Loc_40158E is where the endpoint of the main function is located. The program will first close an open object handle before ending this main procedure. Thus, the WinMain function contains the main function of the malware.

## 3.2. Description of Subroutines

### 3.2.1. Subroutine 1: sub_4011D5

```
; Attributes: bp-based frame

; int __cdecl sub_4011D5(LPCSTR lpFileName,DWORD nNumberOfBytesToWrite,DWORD NumberOfBytesWritten)
sub_4011D5 proc near

lpBuffer= dword ptr -4
lpFileName= dword ptr  8
nNumberOfBytesToWrite= dword ptr  0Ch
NumberOfBytesWritten= dword ptr  10h

push    ebp
mov     ebp, esp
push    ecx
push    ebx
push    esi
push    [ebp+NumberOfBytesWritten] ; lpType
xor     esi, esi
push    [ebp+nNumberOfBytesToWrite] ; lpName
push    esi             ; hModule
call    FindResourceA
mov     ebx, eax
cmp     ebx, esi
jnz     short loc_4011F4
```

```
xor     eax, eax
jmp     short loc_40126D
```

```
loc_4011F4:
push    edi
push    ebx                 ; hResInfo
push    esi                 ; hModule
call    LoadResource
mov     edi, eax
push    edi                 ; hResData
call    LockResource
push    ebx                 ; hResInfo
push    esi                 ; hModule
mov     [ebp+lpBuffer], eax
call    SizeofResource
push    [ebp+lpFileName] ; lpFileName
mov     [ebp+nNumberOfBytesToWrite], eax
call    DeleteFileA
push    esi                 ; hTemplateFile
push    80h                 ; dwFlagsAndAttributes
push    2                   ; dwCreationDisposition
push    esi                 ; lpSecurityAttributes
push    2                   ; dwShareMode
push    40000000h           ; dwDesiredAccess
push    [ebp+lpFileName] ; lpFileName
call    CreateFileA
mov     ebx, eax
lea     eax, [ebp+NumberOfBytesWritten]
push    esi                 ; lpOverlapped
push    eax                 ; lpNumberOfBytesWritten
push    [ebp+nNumberOfBytesToWrite] ; nNumberOfBytesToWrite
mov     [ebp+NumberOfBytesWritten], esi
push    [ebp+lpBuffer]  ; lpBuffer
push    ebx                 ; hFile
call    WriteFile
mov     esi, Sleep
push    1                   ; dwMilliseconds
call    esi ; Sleep
push    edi                 ; hResData
call    FreeResource
push    ebx                 ; hObject
call    CloseHandle
push    2                   ; dwMilliseconds
call    esi ; Sleep
push    1
pop     eax
pop     edi
```

```
loc_40126D:
pop     esi
pop     ebx
leave
retn
sub_4011D5 endp
```

**Purpose of sub_4011D5:**

This subroutine will first call the **FindResourceA** function. The purpose of this function is to determine the location of a resource with the specified type and name in the specified module. In this case, the specified type and name can be retrieved from the FindResourceA parameters: **lpName and lpType**. The **hModule** parameter in this function is a handle to the module whose executable file contains the resource. If function FindResourceA **does not return a zero** (**jnz** condition, zero flag is not set), it will jump to the next location **loc_4011F4**.

Under **loc_4011F4**, the **LoadResource** function is called out first. This function retrieves a handle that can be used to obtain a pointer to the first byte of the specified resource in memory. The parameters this function accepts are the **hModule** and **hResInfo** parameters.

```
push    ebx             ; hResInfo
push    esi             ; hModule
call    LoadResource
```

. The hResInfo parameter is the handle to the resource to be loaded and this handle was created in the **FindResource** function in the previous step.

After the **LoadResource** function is called, the **LockResource** function will be called out next. This function retrieves a pointer to the specified resource in memory and it only accepts the parameter **hResData**. This parameter is a handle to the resource to be accessed.

The **SizeOfResource** function is called next. This function basically retrieves the size of the specified resource in bytes.

Under the same **loc_4011F4**, a file is deleted when the **DeleteFileA** function is called. The name of the file to be deleted is determined by the parameter **lpFileName** in the **DeleteFileA** function.

After deleting a file, a new file will be created using the **CreateFileA** function. Upon creating a file, the **WriteFile** function will write into that created file and the **Sleep** function will suspend this execution for 1ms (parameter dwMilliseconds). After

suspending its execution, the memory occupied by this resource is freed when the **FreeResource** function is called.

At the end, this subroutine calls the **CloseHandle** function to close the open object handle and sleeps for another 2ms. It will then return to the **_WinMain@16** subroutine after all the above is done. Before returning to the main subroutine, all eax, edi, esi and ebx registers are restored (using **pop**).

In summary, this subroutine will find and load the resources necessary for the malware to work. It will also delete, create and update file(s). After completing all the steps mentioned, it will then close that handle, sleep for 2ms and return to the main subroutine.

## Sub_4011D5 Parameter values in OllyDBG.

We ran the malware in OllyDBG and analysed the values that were inputted into each function's parameters in sub_4011D5. Note that some values may vary when the malware runs again in OllyDBG (such as the filename of a deleted file).

FindResourceA:

| Parameter | Values |
|---|---|
| **hModule** | NULL (since the parameter is NULL, the function searches the module used to create the current process) |
| **lpName** | 66 |
| **lpType** | CPP |

LoadResource:

| Parameter | Values |
|---|---|
| **hResInfo** | 00402208 |
| **hModule** | NULL |

SizeOfResource:

| Parameter | Values |
|---|---|
| **hResInfo** | 00402208 |
| **hModule** | NULL |

DeleteFileA:

| Parameter | Value |
|---|---|
| **lpFileName** | C:\1576800.dll |

CreateFileA:

| Parameter | Values |
|---|---|
| **hTemplateFile** | NULL |
| **dwFlagsAndAttributes** | NORMAL |
| **dwCreationDisposition** | CREATE_ALWAYS |
| **lpSecurityAttributes** | NULL |
| **dwShareMode** | FILE_SHARE_WRITE |
| **dwDesiredAccess** | GENERIC_WRITE |
| **lpFileName** | C:\1576800.dl |

WriteFile:

| Parameter | Values |
|---|---|
| **lpOverlapped** | NULL |
| **lpNumberOfBytesWritten** | 0012FDBC |
| **nNumberOfBytesToWrite** | 10000 (65536.) |
| **lpBuffer** | 0.004022A0 |
| **hFile** | 00000034 (window) |

Sleep:

| Parameter | Values |
|---|---|
| **dwMilliseconds** | 1. ms |

FreeResource:

| Parameter | Values |
|---|---|
| **hResData** | 004022A0 |

CloseHandle:

| Parameter | Values |
|---|---|
| **CloseHandle** | 00000034 (window) |

Sleep:

| Parameter | Values |
|---|---|
| **dwMilliseconds** | 2. ms |

### 3.2.2. Subroutine 2: sub_401310

```
; Attributes: bp-based frame

; int __cdecl sub_401310(LPCVOID lpBuffer)
sub_401310 proc near

Buffer= byte ptr -10Ch
NumberOfBytesWritten= dword ptr -8
hObject= dword ptr -4
lpBuffer= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 10Ch
push    ebx
push    edi
push    40h
xor     ebx, ebx
pop     ecx
xor     eax, eax
lea     edi, [ebp-10Bh]
mov     [ebp+Buffer], bl
rep stosd
stosw
stosb
lea     eax, [ebp+Buffer]
push    104h            ; nSize
push    eax             ; lpFilename
push    ebx             ; hModule
call    GetModuleFileNameA
push    ebx             ; hTemplateFile
push    80h             ; dwFlagsAndAttributes
push    2               ; dwCreationDisposition
push    ebx             ; lpSecurityAttributes
push    2               ; dwShareMode
push    40000000h       ; dwDesiredAccess
push    offset FileName ; "c:\\NT_Path.jpg"
call    CreateFileA
or      ecx, 0FFFFFFFFh
mov     [ebp+hObject], eax
cmp     eax, ecx
jz      short loc_4013CA
```

```
push    esi
lea     eax, [ebp+NumberOfBytesWritten]
push    ebx             ; lpOverlapped
push    eax             ; lpNumberOfBytesWritten
lea     edi, [ebp+Buffer]
xor     eax, eax
repne scasb
not     ecx
mov     esi, WriteFile
dec     ecx
lea     eax, [ebp+Buffer]
push    ecx             ; nNumberOfBytesToWrite
push    eax             ; lpBuffer
mov     [ebp+NumberOfBytesWritten], ebx
push    [ebp+hObject]   ; hFile
call    esi ; WriteFile
lea     eax, [ebp+NumberOfBytesWritten]
push    ebx             ; lpOverlapped
push    eax             ; lpNumberOfBytesWritten
push    1               ; nNumberOfBytesToWrite
push    offset Buffer   ; "\n"
push    [ebp+hObject]   ; hFile
call    esi ; WriteFile
mov     edi, [ebp+lpBuffer]
lea     eax, [ebp+NumberOfBytesWritten]
push    ebx             ; lpOverlapped
push    eax             ; lpNumberOfBytesWritten
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
dec     ecx
push    ecx             ; nNumberOfBytesToWrite
push    [ebp+lpBuffer]  ; lpBuffer
push    [ebp+hObject]   ; hFile
call    esi ; WriteFile
push    [ebp+hObject]   ; hObject
call    CloseHandle
pop     esi
```

```
loc_4013CA:
pop     edi
pop     ebx
leave
retn
sub_401310 endp
```

**Purpose of sub_401310:**

This subroutine will first retrieve the fully qualified path for the file that contains the specified module with the **GetModuleFileName** function. Then, it will create a new file with the filename “c:\\NT_Path.jpg” when the **CreateFileA** function is called out. When

comparing eax and ecx, if it **IS** zero (jz), then it will jump to the next location **loc_4013CA.** If it is **NOT** zero, then the program will call out the **WriteFile** function **4 times** before successfully reaching loc_4013CA. Loc_4013CA is where the endpoint of the subroutine is located (sub_401310 **endp**). The "**sub_401310 endp"** will then lead the program back to the main function again.

The picture below shows the call for 4 **WriteFile** functions if the **jz short loc_4013CA** condition is not met.

```
mov     esi, WriteFile
dec     ecx
lea     eax, [ebp+Buffer]
push    ecx             ; nNumberOfBytesToWrite
push    eax             ; lpBuffer
mov     [ebp+NumberOfBytesWritten], ebx
push    [ebp+hObject]   ; hFile
call    esi ; WriteFile
lea     eax, [ebp+NumberOfBytesWritten]
push    ebx             ; lpOverlapped
push    eax             ; lpNumberOfBytesWritten
push    1               ; nNumberOfBytesToWrite
push    offset Buffer   ; "\n"
push    [ebp+hObject]   ; hFile
call    esi ; WriteFile
mov     edi, [ebp+lpBuffer]
lea     eax, [ebp+NumberOfBytesWritten]
push    ebx             ; lpOverlapped
push    eax             ; lpNumberOfBytesWritten
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
dec     ecx
push    ecx             ; nNumberOfBytesToWrite
push    [ebp+lpBuffer]  ; lpBuffer
push    [ebp+hObject]   ; hFile
call    esi ; WriteFile
push    [ebp+hObject]   ; hObject
call    CloseHandle
```

## Sub_401310 Parameter values in OllyDBG.

We ran the malware in OllyDBG and analysed the values that were inputted into each function's parameters in sub_401310. Note that some values may vary when the malware runs again in OllyDBG.

GetModuleFileNameA:

| Parameter | Values |
|---|---|
| **nSize** | 104 (260.) |
| **lpFileName** | 0012FC9C |
| **hModule** | NULL |

CreateFileA:

| Parameter | Values |
|---|---|
| **hTemplateFile** | NULL |
| **dwFlagsAndAttributes** | NORMAL |
| **dwCreationDisposition** | CREATE_ALWAYS |
| **LpSecurityAttributes** | NULL |
| **dwShareMode** | FILE_SHARE_WRITE |
| **dwDesiredAccess** | GENERIC_WRITE |
| **lpFileName** | **C:\NT_Path.jpg** |

WriteFile:

| Parameter | Values |
|---|---|
| **lpOverlapped** | NULL |
| **lpNumberOfBytesWritten** | 0012FDA0 |

WriteFile:

| Parameter | Values |
|---|---|
| **nNumberOfBytesToWrite** | 2F (47.) |
| **lpBuffer** | 0012FC9C |
| **hFile** | 00000034 (window) |

WriteFile:

| Parameter | Values |
|---|---|
| **lpOverlapped** | NULL |
| **lpNumberOfBytesWritten** | 0012FDA0 |
| **nNumberOfBytesToWrite** | 1 |
| **lpBuffer** | 0.0040113C (\n) |
| **hFile** | 00000034 (window) |

WriteFile:

| Parameter | Values |
|---|---|
| **lpOverlapped** | NULL |
| **lpNumberOfBytesWritten** | 0012FDA0 |
| **nNumberOfBytesToWrite** | E (14.) |

| | |
|---|---|
| **lpBuffer** | 0012FDDC |
| **hFile** | 00000034(window) |

CloseHandle

| Parameter | Values |
|---|---|
| **hObject** | 00000034 (window) |

### 3.2.3. Subroutine 3: sub_401271

```
; Attributes: bp-based frame

; int __cdecl sub_401271(BYTE *lpData)
sub_401271 proc near

SubKey= byte ptr -108h
var_C7= dword ptr -0C7h
dwDisposition= dword ptr -8
hKey= dword ptr -4
lpData= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 108h
push    ebx
push    esi
push    edi
push    10h
pop     ecx
mov     esi, offset aSystemCurrentc ; "SYSTEM\\CurrentControlSet\\Services\\Remot"...
lea     edi, [ebp+SubKey]
push    2Fh
rep movsd
movsb
pop     ecx
xor     eax, eax
lea     edi, [ebp+var_C7]
xor     ebx, ebx
rep stosd
stosw
stosb
lea     eax, [ebp+dwDisposition]
mov     esi, 80000002h
push    eax             ; lpdwDisposition
lea     eax, [ebp+hKey]
push    eax             ; phkResult
push    ebx             ; lpSecurityAttributes
push    2               ; samDesired
push    ebx             ; dwOptions
push    ebx             ; lpClass
lea     eax, [ebp+SubKey]
push    ebx             ; Reserved
push    eax             ; lpSubKey
push    esi             ; hKey
mov     [ebp+dwDisposition], 1
call    RegCreateKeyExA
test    eax, eax
jz      short loc_4012DE
```

```
lea     eax, [ebp+hKey]
push    eax             ; phkResult
lea     eax, [ebp+SubKey]
push    eax             ; lpSubKey
push    esi             ; hKey
call    RegOpenKeyA
```

```
loc_4012DE:
mov     edi, [ebp+lpData]
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
dec     ecx
push    ecx             ; cbData
push    [ebp+lpData]    ; lpData
push    2               ; dwType
push    ebx             ; Reserved
push    offset ValueName ; "DLLPath"
push    [ebp+hKey]      ; hKey
call    RegSetValueExA
push    [ebp+hKey]      ; hKey
call    RegCloseKey
pop     edi
pop     esi
xor     eax, eax
pop     ebx
leave
retn
sub_401271 endp
```

**Purpose of sub_401271:**

This subroutine will first call the **RegCreateKeyExA** function, which creates a specified registry key. The RegCreateKeyExA function has a parameter called **hKey**, which is a handle to an open registry key. In this program, the value of this parameter is set to the following predefined key: “HKEY_LOCAL_MACHINE”. It will also create a subkey called “SYSTEM\CurrentControlSet\Services\RemoteAccess\RouterManagers\Ip” (the value of lpSubKey). In addition, the dwOptions parameter in the RegCreatekeyExA function is set to “REG_OPTION_NON_VOLATILE”. This ensures that information is stored in a file and is preserved when the system is restarted. If the **jz short loc_4012DE** is met, (jump to location if test eax,eax sets zero flag to 1) then the program will jump to the

next location, **loc_4012DE**. *Otherwise, the RegOpenKeyA function is called* which opens the specified registry key.

Under loc_4012DE, **RegSetValueExA** function is called. This function sets the data and type of a specified value under a registry key. In this case, the IpValueName parameter is set to “DLLPath”. The **dwType** parameter is set to “REG_EXPAND_SZ” which is a null-terminated string that contains unexpanded references to environment variables (for instance, "%PATH%").

Finally, the **RegCloseKey** function is called. This function closes the handle to the specified registry key.

In conclusion, this subroutine handles the registry keys that are created.

## Sub_401271 Parameter values in OllyDBG.

We ran the malware in OllyDBG and analysed the values that were inputted into each function’s parameters in sub_401271. Note that some values may vary when the malware runs again in OllyDBG.

RegCreateKeyExA:

```
hKey = HKEY_LOCAL_MACHINE
Subkey = "SYSTEM\CurrentControlSet\Services\RemoteAccess\RouterManagers\Ip"
Reserved = 0
Class = NULL
Options = REG_OPTION_NON_VOLATILE
Access = KEY_SET_VALUE
pSecurity = NULL
pHandle = 0012FDC0
pDisposition = 0012FDBC
```

RegSetValueExA:

| Parameters | Values |
|---|---|
| **cbData** | E (14.) |
| **lpData** | 0012FDDC |

| | |
|---|---|
| **dwType** | REG_EXPAND_SZ |
| **Reserved** | 0 |
| **lpValueName** | DLLPath |
| **hKey** | 58 |

RegCloseKey:

```
hKey = 00000058
```

## 4. Patching (OllyDBG)

### 4.1. Main Routine

#### 4.1.1. GetWindowsDirectoryA

**Before**

```
0040146B  . 50            PUSH EAX                                  Buffer
0040146C  . FF15 74104000 CALL DWORD PTR DS:[<&KERNEL32.GetWindow   GetWindowsDirectoryA
```

**Replaced with NOP**

```
0040146B  . 50            PUSH EAX                                  Buffer
0040146C    90            NOP                                       GetWindowsDirectoryA
0040146D    90            NOP
0040146E    90            NOP
0040146F    90            NOP
00401470    90            NOP
00401471    90            NOP
```

**Patched**

```
00401464  .| C70424 04010000 MOV DWORD PTR SS:[ESP],104
0040146B  .| 50              PUSH EAX
0040146C  .| 90              NOP
0040146D  .| 90              NOP
0040146E  .| 90              NOP
0040146F  .| 90              NOP
00401470  .| 90              NOP
00401471  .| 90              NOP
00401472  .| 889D BBFEFFFF   MOV BYTE PTR SS:[EBP-145],BL
```

### 4.1.2. OpenSCManagerA

**Before**

```
004014C4   FF15 1C104000   CALL DWORD PTR DS:[<&ADVAPI32.OpenSCManager  ADVAPI32.OpenSCManagerA
004014CA   68 FF01DF00     PUSH 0F01FF
004014CF   68 84114000     PUSH 0(Main).00401184                       ASCII "RemoteAccess"
004014D4   50              PUSH EAX
004014D5   FF15 18104000   CALL DWORD PTR DS:[<&ADVAPI32.OpenServiceA>  ADVAPI32.OpenServiceA
```

**Replaced with NOP**

```
004014C4   90              NOP
004014C5   90              NOP
004014C6   90              NOP
004014C7   90              NOP
004014C8   90              NOP
004014C9   90              NOP
004014CA   68 FF01DF00     PUSH 0F01FF
004014CF   68 84114000     PUSH 0(Main).00401184                       ASCII "RemoteAccess"
004014D4   50              PUSH EAX
004014D5   FF15 18104000   CALL DWORD PTR DS:[<&ADVAPI32.OpenServiceA>  ADVAPI32.OpenServiceA
```

**Patched**

```
004014C4   90              NOP
004014C5   90              NOP
004014C6   90              NOP
004014C7   90              NOP
004014C8   90              NOP
004014C9   90              NOP
004014CA   68 FF01DF00     PUSH 0F01FF
004014CF   68 84114000     PUSH 0(Main)-.00401184                      ASCII "RemoteAccess"
004014D4   50              PUSH EAX
004014D5   FF15 18104000   CALL DWORD PTR DS:[<&ADVAPI32.OpenServiceA>  ADVAPI32.OpenServiceA
```

### 4.1.3. OpenServiceA

**Before**

```
004014CF   68 84114000     PUSH 0(Main)-.00401184                      ASCII "RemoteAccess"
004014D4   50              PUSH EAX
004014D5   FF15 18104000   CALL DWORD PTR DS:[<&ADVAPI32.OpenServiceA>  ADVAPI32.OpenServiceA
004014DB   53              PUSH EBX
004014DC   53              PUSH EBX
004014DD   53              PUSH EBX
```

**Replaced with NOP**

```
004014CF   |68 84114000   PUSH 0(Main)-.00401184      ASCII "RemoteAccess"
004014D4  .|50            PUSH EAX
004014D5   |90            NOP
004014D6   |90            NOP
004014D7   |90            NOP
004014D8   |90            NOP
004014D9   |90            NOP
004014DA   |90            NOP
004014DB  .|53            PUSH EBX
004014DC  .|53            PUSH EBX
004014DD  .|53            PUSH EBX
```

**Patched**

```
004014CF  .|68 84114000   PUSH 0(Main)-.00401184      ASCII "RemoteAccess"
004014D4  .|50            PUSH EAX
004014D5  .|90            NOP
004014D6  .|90            NOP
004014D7  .|90            NOP
004014D8  .|90            NOP
004014D9  .|90            NOP
004014DA  .|90            NOP
004014DB  .|53            PUSH EBX
004014DC  .|53            PUSH EBX
004014DD  .|53            PUSH EBX
```

### 4.1.4. ChangeServiceConfigA

**Before**

```
004014D5  . FF15 18104000  CALL DWORD PTR DS:[<&ADVAPI32.OpenServi  ADVAPI32.OpenServiceA
004014DB  . 53             PUSH EBX                                 DisplayName
004014DC  . 53             PUSH EBX                                 Password
004014DD  . 53             PUSH EBX                                 ServiceStartName
004014DE  . 8945 08        MOV DWORD PTR SS:[EBP+8],EAX
004014E1  . 53             PUSH EBX                                 pDependencies
004014E2  . 53             PUSH EBX                                 pTagId
004014E3  . 53             PUSH EBX                                 LoadOrderGroup
004014E4  . 53             PUSH EBX                                 BinaryPathName
004014E5  . 6A FF          PUSH -1                                  ErrorControl = SERVICE_NO_CHANGE
004014E7  . 6A 03          PUSH 3                                   StartType = SERVICE_DEMAND_START
004014E9  . 68 10010000    PUSH 110                                 ServiceType = SERVICE_WIN32_OWN_PROCESS|SERVICE_INTERACTIVE_PROCESS
004014EE  . 50             PUSH EAX                                 hService
004014EF  . FF15 20104000  CALL DWORD PTR DS:[<&ADVAPI32.ChangeSer  ChangeServiceConfigA
004014F5  . 6A 06          PUSH 6
```

**Replaced with NOP**

```
004014D5  . FF15 18104000  CALL DWORD PTR DS:[<&ADVAPI32.OpenServi  ADVAPI32.OpenServiceA
004014DB  . 53             PUSH EBX                                 DisplayName
004014DC  . 53             PUSH EBX                                 Password
004014DD  . 53             PUSH EBX                                 ServiceStartName
004014DE  . 8945 08        MOV DWORD PTR SS:[EBP+8],EAX
004014E1  . 53             PUSH EBX                                 pDependencies
004014E2  . 53             PUSH EBX                                 pTagId
004014E3  . 53             PUSH EBX                                 LoadOrderGroup
004014E4  . 53             PUSH EBX                                 BinaryPathName
004014E5  . 6A FF          PUSH -1                                  ErrorControl = SERVICE_NO_CHANGE
004014E7  . 6A 03          PUSH 3                                   StartType = SERVICE_DEMAND_START
004014E9  . 68 10010000    PUSH 110                                 ServiceType = SERVICE_WIN32_OWN_PROCESS|SERVICE_INTERACTIVE_PROCESS
004014EE  . 50             PUSH EAX                                 hService
004014EF    90             NOP                                      ChangeServiceConfigA
004014F0    90             NOP
004014F1    90             NOP
004014F2    90             NOP
004014F3    90             NOP
004014F4    90             NOP
```

**Patched**

```
004014E9  .| 68 10010000   PUSH 110
004014EE  .| 50            PUSH EAX
004014EF  .| 90            NOP
004014F0  .| 90            NOP
004014F1  .| 90            NOP
004014F2  .| 90            NOP
004014F3  .| 90            NOP
004014F4  .| 90            NOP
004014F5  .| 6A 06         PUSH 6
```

### 4.1.5. ControlService

**Before**

```
0040150B  .| FF15 04104000  CALL DWORD PTR DS:[<&ADVAPI32.ControlServic  ADVAPI32.ControlService
```

**Replaced with NOP**

```
0040150B   | 90            NOP
0040150C   | 90            NOP
0040150D   | 90            NOP
0040150E   | 90            NOP
0040150F   | 90            NOP
00401510   | 90            NOP
00401511  .| 8D85 B8FEFFFF  LEA EAX,DWORD PTR SS:[EBP-148]
```

**Patched**

```
00401508  .| FF75 08        PUSH DWORD PTR SS:[EBP+8]
0040150B  .| 90             NOP
0040150C  .| 90             NOP
0040150D  .| 90             NOP
0040150E  .| 90             NOP
0040150F  .| 90             NOP
00401510  .| 90             NOP
00401511  .| 8D85 B8FEFFFF  LEA EAX,DWORD PTR SS:[EBP-148]
```

### 4.1.6. StartServiceA

**Before**

```
00401511  .| 8D85 B8FEFFFF  LEA EAX,DWORD PTR SS:[EBP-148]
00401517  .| 50             PUSH EAX
00401518  .| E8 54FDFFFF    CALL lastsubp.00401271
0040151D  .| 59             POP ECX
0040151E  .| 53             PUSH EBX
0040151F  .| 53             PUSH EBX
00401520  .| FF75 08        PUSH DWORD PTR SS:[EBP+8]
00401523  .| FF15 08104000  CALL DWORD PTR DS:[<&ADVAPI32.StartServiceA  ADVAPI32.StartServiceA
```

**Replaced with NOP**

```
00401520  .| FF75 08        PUSH DWORD PTR SS:[EBP+8]
00401523   | 90             NOP
00401524   | 90             NOP
00401525   | 90             NOP
00401526   | 90             NOP
00401527   | 90             NOP
00401528   | 90             NOP
00401529  .| 8B3D 6C104000  MOV EDI,DWORD PTR DS:[<&KERNEL32.Sleep>]  kernel32.Sleep
```

**Patched**

```
00401520  .| FF75 08        PUSH DWORD PTR SS:[EBP+8]
00401523  .| 90             NOP
00401524  .| 90             NOP
00401525  .| 90             NOP
00401526  .| 90             NOP
00401527  .| 90             NOP
00401528  .| 90             NOP
00401529  .| 8B3D 6C104000  MOV EDI,DWORD PTR DS:[<&KERNEL32.Sleep>]  kernel32.Sleep
```

### 4.1.7. Sleep

**Before**

```
00401529  . 8B3D 6C104000  MOV EDI,DWORD PTR DS:[<&KERNEL32.Sleep>]  kernel32.Sleep
0040152F  . 6A 01          PUSH 1                                   Timeout = 1. ms
00401531  . FFD7           CALL EDI                                 Sleep
```

**Replaced with NOP**

```
0040152F  . 6A 01          PUSH 1                                   Timeout = 1. ms
00401531    90             NOP                                      Sleep
00401532    90             NOP
```

**Patched**

```
00401529  .| 8B3D 6C104000  MOV EDI,DWORD PTR DS:[<&KERNEL32.Sleep>]  kernel32.Sleep
0040152F  .| 6A 01          PUSH 1
00401531  .| 90             NOP
00401532  .| 90             NOP
```

### 4.1.8. LoadLibraryA

**Before**

```
00401533  . 68 70114000    PUSH 0(Sub2).00401170                    ProcNameOrOrdinal = "CloseServiceHandle"
00401538  . 68 60114000    PUSH 0(Sub2).00401160                    FileName = "Advapi32.dll"
0040153D  . FF15 38104000  CALL DWORD PTR DS:[<&KERNEL32.LoadLibra  LoadLibraryA
```

**Replaced with NOP**

```
00401533  . 68 70114000    PUSH 0(Sub2).00401170                    ProcNameOrOrdinal = "CloseServiceHandle"
00401538  . 68 60114000    PUSH 0(Sub2).00401160                    FileName = "Advapi32.dll"
0040153D    90             NOP                                      LoadLibraryA
0040153E    90             NOP
0040153F    90             NOP
00401540    90             NOP
00401541    90             NOP
00401542    90             NOP
```

**Patched**

```
00401538  . 68 60114000    PUSH 0(Main).00401160                    ProcNameOrOrdinal = "Advapi32.dll"
0040153D  . 90             NOP
0040153E  . 90             NOP
0040153F  . 90             NOP
00401540  . 90             NOP
00401541  . 90             NOP
00401542  . 90             NOP
```

### 4.1.9. GetProcAddress

**Before**

```
00401543  . 50             PUSH EAX                                 hModule
00401544  . FF15 34104000  CALL DWORD PTR DS:[<&KERNEL32.GetProcAd  GetProcAddress
0040154A  . FF75 08        PUSH DWORD PTR SS:[EBP+8]
0040154D  . FFD0           CALL EAX
```

**Replaced with NOP**

```
00401543  . 50          PUSH EAX                                  hModule
00401544    90          NOP                                       GetProcAddress
00401545    90          NOP
00401546    90          NOP
00401547    90          NOP
00401548    90          NOP
00401549    90          NOP
```

**Patched**

```
00401543  . 50          PUSH EAX
00401544  . 90          NOP
00401545  . 90          NOP
00401546  . 90          NOP
00401547  . 90          NOP
00401548  . 90          NOP
00401549  . 90          NOP
```

### 4.1.10. CreateEventA

**Before**

```
00401551  . FFD7          CALL EDI
00401553  . 68 50114000   PUSH 0(Sub2).00401150          EventName = "Glable__Wait"
00401558  . 53            PUSH EBX                       InitiallySignaled
00401559  . 53            PUSH EBX                       ManualReset
0040155A  . 53            PUSH EBX                       pSecurity
0040155B  . FF15 30104000 CALL DWORD PTR DS:[<&KERNEL32.CreateEve  CreateEventA
```

**Replaced with NOP**

```
00401553  . 68 50114000   PUSH 0(Sub2).00401150          EventName = "Glable__Wait"
00401558  . 53            PUSH EBX                       InitiallySignaled
00401559  . 53            PUSH EBX                       ManualReset
0040155A  . 53            PUSH EBX                       pSecurity
0040155B    90            NOP                            CreateEventA
0040155C    90            NOP
0040155D    90            NOP
0040155E    90            NOP
0040155F    90            NOP
00401560    90            NOP
```

**Patched**

```
00401553  . 68 50114000   PUSH 0(Main).00401150          ASCII "Glable__Wait"
00401558  . 53            PUSH EBX
00401559  . 53            PUSH EBX
0040155A  . 53            PUSH EBX
0040155B  . 90            NOP
0040155C  . 90            NOP
0040155D  . 90            NOP
0040155E  . 90            NOP
0040155F  . 90            NOP
00401560  . 90            NOP
```

### 4.1.11. WaitForSingleObject

**Before**

```
00401565  . 8945 08       MOV DWORD PTR SS:[EBP+8],EAX
00401568  > 6A 64         PUSH 64                        Timeout = 100. ms
0040156A  . 53            PUSH EBX                       hObject
0040156B  . FF15 40104000 CALL DWORD PTR DS:[<&KERNEL32.WaitForS  WaitForSingleObject
00401571  . 85C0          TEST EAX,EAX
```

**Replaced with NOP**

```
00401565  . 8945 08        MOV DWORD PTR SS:[EBP+8],EAX
00401568  > 6A 64          PUSH 64                          Timeout = 100. ms
0040156A  . 53             PUSH EBX                         hObject
0040156B    90             NOP                              WaitForSingleObject
0040156C    90             NOP
0040156D    90             NOP
0040156E    90             NOP
0040156F    90             NOP
00401570    90             NOP
```

**Patched**

```
00401565  . 8945 08        MOV DWORD PTR SS:[EBP+8],EAX
00401568  > 6A 64          PUSH 64
0040156A  . 53             PUSH EBX
0040156B  . 90             NOP
0040156C  . 90             NOP
0040156D  . 90             NOP
0040156E  . 90             NOP
0040156F  . 90             NOP
00401570  . 90             NOP
```

### 4.1.12. LoadLibraryA (if jz condition not met)

**Before**

```
00401581  . 8D85 B8FEFFFF  LEA EAX,DWORD PTR SS:[EBP-148]
00401587  . 50             PUSH EAX                         FileName
00401588  . FF15 38104000  CALL DWORD PTR DS:[<&KERNEL32.LoadLibra  LoadLibraryA
```

**Replaced with NOP**

```
00401581  . 8D85 B8FEFFFF  LEA EAX,DWORD PTR SS:[EBP-148]
00401587  . 50             PUSH EAX                         FileName
00401588    90             NOP                              LoadLibraryA
00401589    90             NOP
0040158A    90             NOP
0040158B    90             NOP
0040158C    90             NOP
0040158D    90             NOP
```

**Patched**

```
00401581  . 8D85 B8FEFFFF  LEA EAX,DWORD PTR SS:[EBP-148]
00401587  . 50             PUSH EAX
00401588  . 90             NOP
00401589  . 90             NOP
0040158A  . 90             NOP
0040158B  . 90             NOP
0040158C  . 90             NOP
0040158D  . 90             NOP
```

### 4.1.13. CloseHandle

**Before**

```
00401592  . 53             PUSH EBX                         hObject
00401593  . FF15 68104000  CALL DWORD PTR DS:[<&KERNEL32.CloseHand  CloseHandle
00401599  . 5F             POP EDI
```

**Replaced with NOP**

```
00401590  . FFD7          CALL EDI
00401592  . 53            PUSH EBX                    hObject
00401593    90            NOP                         CloseHandle
00401594    90            NOP
00401595    90            NOP
00401596    90            NOP
00401597    90            NOP
00401598    90            NOP
```

**Patched**

```
00401590  . FFD7          CALL EDI
00401592  . 53            PUSH EBX
00401593  . 90            NOP
00401594  . 90            NOP
00401595  . 90            NOP
00401596  . 90            NOP
00401597  . 90            NOP
00401598  . 90            NOP
```

## 4.2. **Subroutine 1:** sub_4011D5

### 4.2.1. FindResourceA

**Before**

```
004011DB  . FF75 10       PUSH DWORD PTR SS:[EBP+10]          ResourceType
004011DE  . 33F6          XOR ESI,ESI
004011E0  . FF75 0C       PUSH DWORD PTR SS:[EBP+C]           ResourceName
004011E3  . 56            PUSH ESI                            hModule => NULL
004011E4    FF15 3C104000 CALL DWORD PTR DS:[<&KERNEL32.FindResou  FindResourceA
```

**Replaced with NOP**

```
004011DA  . 56            PUSH ESI
004011DB  . FF75 10       PUSH DWORD PTR SS:[EBP+10]          ResourceType
004011DE  . 33F6          XOR ESI,ESI
004011E0  . FF75 0C       PUSH DWORD PTR SS:[EBP+C]           ResourceName
004011E3  . 56            PUSH ESI                            hModule => NULL
004011E4    90            NOP                                 FindResourceA
004011E5    90            NOP
004011E6    90            NOP
004011E7    90            NOP
004011E8    90            NOP
004011E9    90            NOP
```

**Patched**

```
004011DA  . 56            PUSH ESI
004011DB  . FF75 10       PUSH DWORD PTR SS:[EBP+10]
004011DE  . 33F6          XOR ESI,ESI
004011E0  . FF75 0C       PUSH DWORD PTR SS:[EBP+C]
004011E3  . 56            PUSH ESI
004011E4  . 90            NOP
004011E5  . 90            NOP
004011E6  . 90            NOP
004011E7  . 90            NOP
004011E8  . 90            NOP
004011E9  . 90            NOP
004011EA  . 8BD8          MOV EBX,EAX
```

### 4.2.2. LoadResource

**Before**

```
004011F5  . 53            PUSH EBX                                   hResource
004011F6  . 56            PUSH ESI                                   hModule
004011F7    FF15 48104000 CALL DWORD PTR DS:[<&KERNEL32.LoadResou    LoadResource
004011FD  . 8BF8          MOV EDI,EAX
```

**Replaced with NOP**

```
004011F4  > 57            PUSH EDI
004011F5  . 53            PUSH EBX                                   hResource
004011F6  . 56            PUSH ESI                                   hModule
004011F7    90            NOP                                        LoadResource
004011F8    90            NOP
004011F9    90            NOP
004011FA    90            NOP
004011FB    90            NOP
004011FC    90            NOP
004011FD  . 8BF8          MOV EDI,EAX
```

**Patched**

```
004011F4  > 57            PUSH EDI
004011F5  . 53            PUSH EBX
004011F6  . 56            PUSH ESI
004011F7  . 90            NOP
004011F8  . 90            NOP
004011F9  . 90            NOP
004011FA  . 90            NOP
004011FB  . 90            NOP
004011FC  . 90            NOP
004011FD  . 8BF8          MOV EDI,EAX
```

### 4.2.3. SetHandleCount/LockResource

**Before**

```
004011FD  . 8BF8          MOV EDI,EAX
004011FF  . 57            PUSH EDI                                   nHandles
00401200    FF15 4C104000 CALL DWORD PTR DS:[<&KERNEL32.LockResou    SetHandleCount
00401206  . 53            PUSH EBX                                   hResource
```

**Replaced with NOP**

```
004011FD  . 8BF8          MOV EDI,EAX
004011FF  . 57            PUSH EDI                                  nHandles
00401200    90            NOP                                       SetHandleCount
00401201    90            NOP
00401202    90            NOP
00401203    90            NOP
00401204    90            NOP
00401205    90            NOP
```

**Patched**

```
004011FD  . 8BF8          MOV EDI,EAX
004011FF  . 57            PUSH EDI
00401200  . 90            NOP
00401201  . 90            NOP
00401202  . 90            NOP
00401203  . 90            NOP
00401204  . 90            NOP
00401205  . 90            NOP
```

### 4.2.4. SizeofResource

**Before**

```
00401206  . 53            PUSH EBX                                  hResource
00401207  . 56            PUSH ESI                                  hModule
00401208  . 8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
0040120B    FF15 50104000 CALL DWORD PTR DS:[<&KERNEL32.SizeofRes   SizeofResource
```

**Replaced with NOP**

```
00401206  . 53            PUSH EBX                                  hResource
00401207  . 56            PUSH ESI                                  hModule
00401208  . 8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
0040120B    90            NOP                                       SizeofResource
0040120C    90            NOP
0040120D    90            NOP
0040120E    90            NOP
0040120F    90            NOP
00401210    90            NOP
```

**Patched**

```
00401206  . 53            PUSH EBX
00401207  . 56            PUSH ESI
00401208  . 8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
0040120B  . 90            NOP
0040120C  . 90            NOP
0040120D  . 90            NOP
0040120E  . 90            NOP
0040120F  . 90            NOP
00401210  . 90            NOP
```

### 4.2.5. DeleteFileA

**Before**

```
00401211  . FF75 08        PUSH DWORD PTR SS:[EBP+8]                  FileName
00401214  . 8945 0C        MOV DWORD PTR SS:[EBP+C],EAX
00401217    FF15 5C104000  CALL DWORD PTR DS:[<&KERNEL32.DeleteFil    DeleteFileA
```

**Replaced with NOP**

```
00401211  . FF75 08        PUSH DWORD PTR SS:[EBP+8]                  FileName
00401214  . 8945 0C        MOV DWORD PTR SS:[EBP+C],EAX               DeleteFileA
00401217    90             NOP
00401218    90             NOP
00401219    90             NOP
0040121A    90             NOP
0040121B    90             NOP
0040121C    90             NOP
```

**Patched**

```
00401217  . 90             NOP
00401218  . 90             NOP
00401219  . 90             NOP
0040121A  . 90             NOP
0040121B  . 90             NOP
0040121C  . 90             NOP
```

### 4.2.6. CreateFileA

**Before**

```
0040121D  . 56             PUSH ESI                                   hTemplateFile
0040121E  . 68 80000000    PUSH 80                                    Attributes = NORMAL
00401223  . 6A 02          PUSH 2                                     Mode = CREATE_ALWAYS
00401225  . 56             PUSH ESI                                   pSecurity
00401226  . 6A 02          PUSH 2                                     ShareMode = FILE_SHARE_WRITE
00401228  . 68 00000040    PUSH 40000000                              Access = GENERIC_WRITE
0040122D  . FF75 08        PUSH DWORD PTR SS:[EBP+8]                  FileName
00401230  . FF15 60104000  CALL DWORD PTR DS:[<&KERNEL32.CreateFil    CreateFileA
```

**Replaced with NOP**

```
0040121C    90             NOP
0040121D  . 56             PUSH ESI                                   hTemplateFile
0040121E  . 68 80000000    PUSH 80                                    Attributes = NORMAL
00401223  . 6A 02          PUSH 2                                     Mode = CREATE_ALWAYS
00401225  . 56             PUSH ESI                                   pSecurity
00401226  . 6A 02          PUSH 2                                     ShareMode = FILE_SHARE_WRITE
00401228  . 68 00000040    PUSH 40000000                              Access = GENERIC_WRITE
0040122D  . FF75 08        PUSH DWORD PTR SS:[EBP+8]                  FileName
00401230    90             NOP                                        CreateFileA
00401231    90             NOP
00401232    90             NOP
00401233    90             NOP
00401234    90             NOP
00401235    90             NOP
```

**Patched**

```
0040121C  . 90            NOP
0040121D  . 56            PUSH ESI
0040121E  . 68 80000000   PUSH 80
00401223  . 6A 02         PUSH 2
00401225  . 56            PUSH ESI
00401226  . 6A 02         PUSH 2
00401228  . 68 00000040   PUSH 40000000
0040122D  . FF75 08       PUSH DWORD PTR SS:[EBP+8]
00401230  . 90            NOP
00401231  . 90            NOP
00401232  . 90            NOP
00401233  . 90            NOP
00401234  . 90            NOP
00401235  . 90            NOP
```

### 4.2.7. WriteFile

**Before**

```
00401238  . 8D45 10       LEA EAX,DWORD PTR SS:[EBP+10]
0040123B  . 56            PUSH ESI                          pOverlapped
0040123C  . 50            PUSH EAX                          pBytesWritten
0040123D  . FF75 0C       PUSH DWORD PTR SS:[EBP+C]         nBytesToWrite
00401240  . 8975 10       MOV DWORD PTR SS:[EBP+10],ESI
00401243  . FF75 FC       PUSH DWORD PTR SS:[EBP-4]         Buffer
00401246  . 53            PUSH EBX                          hFile
00401247  . FF15 64104000 CALL DWORD PTR DS:[<&KERNEL32.WriteFile  WriteFile
```

**Replaced with NOP**

```
0040123B  . 56            PUSH ESI                          pOverlapped
0040123C  . 50            PUSH EAX                          pBytesWritten
0040123D  . FF75 0C       PUSH DWORD PTR SS:[EBP+C]         nBytesToWrite
00401240  . 8975 10       MOV DWORD PTR SS:[EBP+10],ESI
00401243  . FF75 FC       PUSH DWORD PTR SS:[EBP-4]         Buffer
00401246  . 53            PUSH EBX                          hFile
00401247    90            NOP                               WriteFile
00401248    90            NOP
00401249    90            NOP
0040124A    90            NOP
0040124B    90            NOP
0040124C    90            NOP
```

**Patched**

```
0040123B  . 56            PUSH ESI
0040123C  . 50            PUSH EAX
0040123D  . FF75 0C       PUSH DWORD PTR SS:[EBP+C]
00401240  . 8975 10       MOV DWORD PTR SS:[EBP+10],ESI
00401243  . FF75 FC       PUSH DWORD PTR SS:[EBP-4]
00401246  . 53            PUSH EBX
00401247  . 90            NOP
00401248  . 90            NOP
00401249  . 90            NOP
0040124A  . 90            NOP
0040124B  . 90            NOP
0040124C  . 90            NOP
```

### 4.2.8. Sleep

**Before**

```
00401253  . 6A 01          PUSH 1                                   Timeout = 1. ms
00401255  . FFD6           CALL ESI                                 Sleep
```

**Replaced with NOP**

```
00401253  . 6A 01          PUSH 1                                   Timeout = 1. ms
00401255    90             NOP                                      Sleep
00401256    90             NOP
```

**Patched**

```
00401253  . 6A 01          PUSH 1
00401255  . 90             NOP
00401256  . 90             NOP
```

### 4.2.9. FreeResource

**Before**

```
00401257  . 57             PUSH EDI                                 hResource
00401258  . FF15 70104000  CALL DWORD PTR DS:[<&KERNEL32.FreeResou  FreeResource
```

**Replaced with NOP**

```
00401257  . 57             PUSH EDI                                 hResource
00401258    90             NOP                                      FreeResource
00401259    90             NOP
0040125A    90             NOP
0040125B    90             NOP
0040125C    90             NOP
0040125D    90             NOP
```

**Patched**

```
00401257  . 57             PUSH EDI
00401258  . 90             NOP
00401259  . 90             NOP
0040125A  . 90             NOP
0040125B  . 90             NOP
0040125C  . 90             NOP
0040125D  . 90             NOP
```

### 4.2.10. CloseHandle

**Before**

```
0040125E  . 53              PUSH EBX                                  hObject
0040125F  . FF15 68104000   CALL DWORD PTR DS:[<&KERNEL32.CloseHand    CloseHandle
```

**Replaced with NOP**

```
0040125E  . 53              PUSH EBX                                  hObject
0040125F    90              NOP                                       CloseHandle
00401260    90              NOP
00401261    90              NOP
00401262    90              NOP
00401263    90              NOP
00401264    90              NOP
```

**Patched**

```
0040125E  . 53              PUSH EBX
0040125F  . 90              NOP
00401260  . 90              NOP
00401261  . 90              NOP
00401262  . 90              NOP
00401263  . 90              NOP
00401264  . 90              NOP
```

### 4.2.11. Sleep

**Before**

```
00401265  . 6A 02           PUSH 2                                    Timeout = 2. ms
00401267  . FFD6            CALL ESI                                  Sleep
```

**Replaced with NOP**

```
00401265  . 6A 02           PUSH 2                                    Timeout = 2. ms
00401267    90              NOP                                       Sleep
00401268    90              NOP
```

**Patched**

```
00401265  . 6A 02           PUSH 2
00401267  . 90              NOP
00401268  . 90              NOP
```

## 4.3. Subroutine 2: sub_401310

### 4.3.1. GetModuleFileNameA

**Before**

```
00401339  . 68 04010000   PUSH 104                                  BufSize = 104 (260.)
0040133E  . 50            PUSH EAX                                  PathBuffer
0040133F  . 53            PUSH EBX                                  hModule => NULL
00401340  . FF15 58104000 CALL DWORD PTR DS:[<&KERNEL32.GetModule   GetModuleFileNameA
```

**Replaced with NOP**

```
00401339  . 68 04010000   PUSH 104                                  BufSize = 104 (260.)
0040133E  . 50            PUSH EAX                                  PathBuffer
0040133F  . 53            PUSH EBX                                  hModule => NULL
00401340    90            NOP                                       GetModuleFileNameA
00401341    90            NOP
00401342    90            NOP
00401343    90            NOP
00401344    90            NOP
00401345    90            NOP
```

**Patched**

```
00401339  . 68 04010000   PUSH 104
0040133E  . 50            PUSH EAX
0040133F  . 53            PUSH EBX
00401340  . 90            NOP
00401341  . 90            NOP
00401342  . 90            NOP
00401343  . 90            NOP
00401344  . 90            NOP
00401345  . 90            NOP
```

### 4.3.2. CreateFileA

**Before**

```
00401346  . 53            PUSH EBX                                  hTemplateFile => NULL
00401347  . 68 80000000   PUSH 80                                   Attributes = NORMAL
0040134C  . 6A 02         PUSH 2                                    Mode = CREATE_ALWAYS
0040134E  . 53            PUSH EBX                                  pSecurity => NULL
0040134F  . 6A 02         PUSH 2                                    ShareMode = FILE_SHARE_WRITE
00401351  . 68 00000040   PUSH 40000000                             Access = GENERIC_WRITE
00401356  . 68 40114000   PUSH 0(Sub1).00401140                     FileName = "c:\NT_Path.jpg"
0040135B  . FF15 60104000 CALL DWORD PTR DS:[<&KERNEL32.CreateFil   CreateFileA
```

**Replaced with NOP**

```
00401346  . 53              PUSH EBX                                 hTemplateFile => NULL
00401347  . 68 80000000     PUSH 80                                  Attributes = NORMAL
0040134C  . 6A 02           PUSH 2                                   Mode = CREATE_ALWAYS
0040134E  . 53              PUSH EBX                                 pSecurity => NULL
0040134F  . 6A 02           PUSH 2                                   ShareMode = FILE_SHARE_WRITE
00401351  . 68 00000040     PUSH 40000000                            Access = GENERIC_WRITE
00401356  . 68 40114000     PUSH 0(Sub1).00401140                    FileName = "c:\NT_Path.jpg"
0040135B    90              NOP                                      CreateFileA
0040135C    90              NOP
0040135D    90              NOP
0040135E    90              NOP
0040135F    90              NOP
00401360    90              NOP
```

**Patched**

```
00401346  . 53              PUSH EBX
00401347  . 68 80000000     PUSH 80
0040134C  . 6A 02           PUSH 2
0040134E  . 53              PUSH EBX
0040134F  . 6A 02           PUSH 2
00401351  . 68 00000040     PUSH 40000000
00401356  . 68 40114000     PUSH 0(Sub2).00401140                    ASCII "c:\NT_Path.jpg"
0040135B  . 90              NOP
0040135C  . 90              NOP
0040135D  . 90              NOP
0040135E  . 90              NOP
0040135F  . 90              NOP
00401360  . 90              NOP
```

### 4.3.3. WriteFile, WriteFile, WriteFile (if jz condition not met)

**Before**

```
0040136F  . 53              PUSH EBX                                 pOverlapped => NULL
00401370  . 50              PUSH EAX                                 pBytesWritten
00401371  . 8DBD F4FEFFFF   LEA EDI,DWORD PTR SS:[EBP-10C]
00401377  . 33C0            XOR EAX,EAX
00401379  . F2:AE           REPNE SCAS BYTE PTR ES:[EDI]
0040137B  . F7D1            NOT ECX
0040137D  . 8B35 64104000   MOV ESI,DWORD PTR DS:[<&KERNEL32.WriteF  kernel32.WriteFile
00401383  . 49              DEC ECX
00401384  . 8D85 F4FEFFFF   LEA EAX,DWORD PTR SS:[EBP-10C]
0040138A  . 51              PUSH ECX                                 nBytesToWrite
0040138B  . 50              PUSH EAX                                 Buffer
0040138C  . 895D F8         MOV DWORD PTR SS:[EBP-8],EBX
0040138F  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                hFile
00401392    FFD6            CALL ESI                                 WriteFile
00401394    8D45 F8         LEA EAX,DWORD PTR SS:[EBP-8]
00401397    53              PUSH EBX                                 pOverlapped => NULL
00401398  . 50              PUSH EAX                                 pBytesWritten
00401399  . 6A 01           PUSH 1                                   nBytesToWrite = 1
0040139B  . 68 3C114000     PUSH 0(Sub1).0040113C                    Buffer = 0(Sub1).0040113C
004013A0  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                hFile
004013A3  . FFD6            CALL ESI                                 WriteFile
004013A5  . 8B7D 08         MOV EDI,DWORD PTR SS:[EBP+8]
004013A8  . 8D45 F8         LEA EAX,DWORD PTR SS:[EBP-8]
004013AB  . 53              PUSH EBX                                 pOverlapped => NULL
004013AC  . 50              PUSH EAX                                 pBytesWritten
004013AD  . 83C9 FF         OR ECX,FFFFFFFF
004013B0  . 33C0            XOR EAX,EAX
004013B2  . F2:AE           REPNE SCAS BYTE PTR ES:[EDI]
004013B4  . F7D1            NOT ECX
004013B6  . 49              DEC ECX
004013B7  . 51              PUSH ECX                                 nBytesToWrite
004013B8  . FF75 08         PUSH DWORD PTR SS:[EBP+8]                Buffer
004013BB  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                hFile
004013BE  . FFD6            CALL ESI                                 WriteFile
```

**Replaced with NOP**

```
0040136F  . 53              PUSH EBX                                  pOverlapped => NULL
00401370  . 50              PUSH EAX                                  pBytesWritten
00401371  . 8DBD F4FEFFFF   LEA EDI,DWORD PTR SS:[EBP-10C]
00401377  . 33C0            XOR EAX,EAX
00401379  . F2:AE           REPNE SCAS BYTE PTR ES:[EDI]
0040137B  . F7D1            NOT ECX
0040137D  . 8B35 64104000   MOV ESI,DWORD PTR DS:[<&KERNEL32.WriteF   kernel32.WriteFile
00401383  . 49              DEC ECX
00401384  . 8D85 F4FEFFFF   LEA EAX,DWORD PTR SS:[EBP-10C]
0040138A  . 51              PUSH ECX                                  nBytesToWrite
0040138B  . 50              PUSH EAX                                  Buffer
0040138C  . 895D F8         MOV DWORD PTR SS:[EBP-8],EBX
0040138F  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                 hFile
00401392    90              NOP                                       WriteFile
00401393    90              NOP
00401394    8D45 F8         LEA EAX,DWORD PTR SS:[EBP-8]
00401397    53              PUSH EBX                                  pOverlapped => NULL
00401398  . 50              PUSH EAX                                  pBytesWritten
00401399  . 6A 01           PUSH 1                                    nBytesToWrite = 1
0040139B  . 68 3C114000     PUSH 0(Sub1).0040113C                     Buffer = 0(Sub1).0040113C
004013A0  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                 hFile
004013A3    90              NOP                                       WriteFile
004013A4    90              NOP
004013A5  . 8B7D 08         MOV EDI,DWORD PTR SS:[EBP+8]
004013A8  . 8D45 F8         LEA EAX,DWORD PTR SS:[EBP-8]
004013AB  . 53              PUSH EBX                                  pOverlapped => NULL
004013AC  . 50              PUSH EAX                                  pBytesWritten
004013AD  . 83C9 FF         OR ECX,FFFFFFFF
004013B0  . 33C0            XOR EAX,EAX
004013B2  . F2:AE           REPNE SCAS BYTE PTR ES:[EDI]
004013B4  . F7D1            NOT ECX
004013B6  . 49              DEC ECX
004013B7  . 51              PUSH ECX                                  nBytesToWrite
004013B8  . FF75 08         PUSH DWORD PTR SS:[EBP+8]                 Buffer
004013BB  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                 hFile
004013BE    90              NOP                                       WriteFile
004013BF    90              NOP
004013C0  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                 hObject
```

**Patched**

```
00401397  . 53              PUSH EBX
00401398  . 50              PUSH EAX
00401399  . 6A 01           PUSH 1
0040139B  . 68 3C114000     PUSH 0(Sub2).0040113C
004013A0  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]
004013A3  . 90              NOP
004013A4  . 90              NOP
004013A5  . 8B7D 08         MOV EDI,DWORD PTR SS:[EBP+8]
004013A8  . 8D45 F8         LEA EAX,DWORD PTR SS:[EBP-8]
004013AB  . 53              PUSH EBX
004013AC  . 50              PUSH EAX
004013AD  . 83C9 FF         OR ECX,FFFFFFFF
004013B0  . 33C0            XOR EAX,EAX
004013B2  . F2:AE           REPNE SCAS BYTE PTR ES:[EDI]
004013B4  . F7D1            NOT ECX
004013B6  . 49              DEC ECX
004013B7  . 51              PUSH ECX
004013B8  . FF75 08         PUSH DWORD PTR SS:[EBP+8]
004013BB  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]
004013BE  . 90              NOP
004013BF  . 90              NOP
```

### 4.3.4. CloseHandle

**Before**

```
004013C0  . FF75 FC         PUSH DWORD PTR SS:[EBP-4]                 hObject
004013C3  . FF15 68104000   CALL DWORD PTR DS:[<&KERNEL32.CloseHand   CloseHandle
```

**Replaced with NOP**

```
004013C0  . FF75 FC      PUSH DWORD PTR SS:[EBP-4]      hObject
004013C3    90           NOP                            CloseHandle
004013C4    90           NOP
004013C5    90           NOP
004013C6    90           NOP
004013C7    90           NOP
004013C8    90           NOP
```

**Patched**

```
004013C0  . FF75 FC      PUSH DWORD PTR SS:[EBP-4]
004013C3  . 90           NOP
004013C4  . 90           NOP
004013C5  . 90           NOP
004013C6  . 90           NOP
004013C7  . 90           NOP
004013C8  . 90           NOP
```

## 4.4. Subroutine 3: sub_401271

### 4.4.1. RegCreateKeyExA

**Before**

```
004012A8  .| 50            PUSH EAX                                  pDisposition
004012A9  .| 8D45 FC       LEA EAX,DWORD PTR SS:[EBP-4]
004012AC  .| 50            PUSH EAX                                  pHandle
004012AD  .| 53            PUSH EBX                                  pSecurity => NULL
004012AE  .| 6A 02         PUSH 2                                    Access = KEY_SET_VALUE
004012B0  .| 53            PUSH EBX                                  Options => REG_OPTION_NON_VOLATILE
004012B1  .| 53            PUSH EBX                                  Class => NULL
004012B2  .| 8D85 F8FEFFFF LEA EAX,DWORD PTR SS:[EBP-108]
004012B8  .| 53            PUSH EBX                                  Reserved => 0
004012B9  .| 50            PUSH EAX                                  Subkey
004012BA  .| 56            PUSH ESI                                  hKey => HKEY_LOCAL_MACHINE
004012BB  .| C745 F8 010000 MOV DWORD PTR SS:[EBP-8],1
004012C2  .| FF15 00104000 CALL DWORD PTR DS:[<&ADVAPI32.RegCreateKeyE  RegCreateKeyExA
```

**Replaced with NOP**

```
004012A8  .| 50              PUSH EAX                                  pDisposition
004012A9  .| 8D45 FC         LEA EAX,DWORD PTR SS:[EBP-4]
004012AC  .| 50              PUSH EAX                                  pHandle
004012AD  .| 53              PUSH EBX                                  pSecurity => NULL
004012AE  .| 6A 02           PUSH 2                                    Access = KEY_SET_VALUE
004012B0  .| 53              PUSH EBX                                  Options => REG_OPTION_NON_VOLATILE
004012B1  .| 53              PUSH EBX                                  Class => NULL
004012B2  .| 8D85 F8FEFFFF   LEA EAX,DWORD PTR SS:[EBP-108]
004012B8  .| 53              PUSH EBX                                  Reserved => 0
004012B9  .| 50              PUSH EAX                                  Subkey
004012BA  .| 56              PUSH ESI                                  hKey => HKEY_LOCAL_MACHINE
004012BB  .| C745 F8 010000  MOV DWORD PTR SS:[EBP-8],1
004012C2  | 90               NOP                                       RegCreateKeyExA
004012C3  | 90               NOP
004012C4  | 90               NOP
004012C5  | 90               NOP
004012C6  | 90               NOP
004012C7  | 90               NOP
```

**Patched**

```
004012A8  .| 50              PUSH EAX
004012A9  .| 8D45 FC         LEA EAX,DWORD PTR SS:[EBP-4]
004012AC  .| 50              PUSH EAX
004012AD  .| 53              PUSH EBX
004012AE  .| 6A 02           PUSH 2
004012B0  .| 53              PUSH EBX
004012B1  .| 53              PUSH EBX
004012B2  .| 8D85 F8FEFFFF   LEA EAX,DWORD PTR SS:[EBP-108]
004012B8  .| 53              PUSH EBX
004012B9  .| 50              PUSH EAX
004012BA  .| 56              PUSH ESI
004012BB  .| C745 F8 010000  MOV DWORD PTR SS:[EBP-8],1
004012C2  .| 90              NOP
004012C3  .| 90              NOP
004012C4  .| 90              NOP
004012C5  .| 90              NOP
004012C6  .| 90              NOP
004012C7  .| 90              NOP
```

### 4.4.2. RegSetValueExA

**Before**

```
004012EB  .| 51              PUSH ECX                                  BufSize
004012EC  .| FF75 08         PUSH DWORD PTR SS:[EBP+8]                 Buffer
004012EF  .| 6A 02           PUSH 2                                    ValueType = REG_EXPAND_SZ
004012F1  .| 53              PUSH EBX                                  Reserved
004012F2  .| 68 F0104000     PUSH 0(Main)-.004010F0                    ValueName = "DLLPath"
004012F7  .| FF75 FC         PUSH DWORD PTR SS:[EBP-4]                 hKey
004012FA  .| FF15 10104000   CALL DWORD PTR DS:[<&ADVAPI32.RegSetValueEx  RegSetValueExA
```

**Replaced with NOP**

```
004012EB  .| 51            PUSH ECX                          BufSize
004012EC  .| FF75 08       PUSH DWORD PTR SS:[EBP+8]         Buffer
004012EF  .| 6A 02         PUSH 2                            ValueType = REG_EXPAND_SZ
004012F1  .| 53            PUSH EBX                          Reserved
004012F2  .| 68 F0104000   PUSH 0(Main)-.004010F0            ValueName = "DLLPath"
004012F7  .| FF75 FC       PUSH DWORD PTR SS:[EBP-4]         hKey
004012FA  | 90             NOP                               RegSetValueExA
004012FB  | 90             NOP
004012FC  | 90             NOP
004012FD  | 90             NOP
004012FE  | 90             NOP
004012FF  | 90             NOP
```

**Patched**

```
004012EB  .| 51            PUSH ECX
004012EC  .| FF75 08       PUSH DWORD PTR SS:[EBP+8]
004012EF  .| 6A 02         PUSH 2
004012F1  .| 53            PUSH EBX
004012F2  .| 68 F0104000   PUSH 0(Main)-.004010F0            ASCII "DLLPath"
004012F7  .| FF75 FC       PUSH DWORD PTR SS:[EBP-4]
004012FA  .| 90            NOP
004012FB  .| 90            NOP
004012FC  .| 90            NOP
004012FD  .| 90            NOP
004012FE  .| 90            NOP
004012FF  .| 90            NOP
```

### 4.4.3. RegCloseKey

**Before**

```
00401300  .| FF75 FC       PUSH DWORD PTR SS:[EBP-4]         hKey
00401303  .| FF15 14104000 CALL DWORD PTR DS:[<&ADVAPI32.RegCloseKey>] RegCloseKey
```

**Replaced with NOP**

```
00401300  .| FF75 FC       PUSH DWORD PTR SS:[EBP-4]         hKey
00401303  | 90             NOP                               RegCloseKey
00401304  | 90             NOP
00401305  | 90             NOP
00401306  | 90             NOP
00401307  | 90             NOP
00401308  | 90             NOP
```

**Patched**

```
00401300  .| FF75 FC      PUSH DWORD PTR SS:[EBP-4]
00401303  .| 90           NOP
00401304  .| 90           NOP
00401305  .| 90           NOP
00401306  .| 90           NOP
00401307  .| 90           NOP
00401308  .| 90           NOP
```

## 5. General Analysis

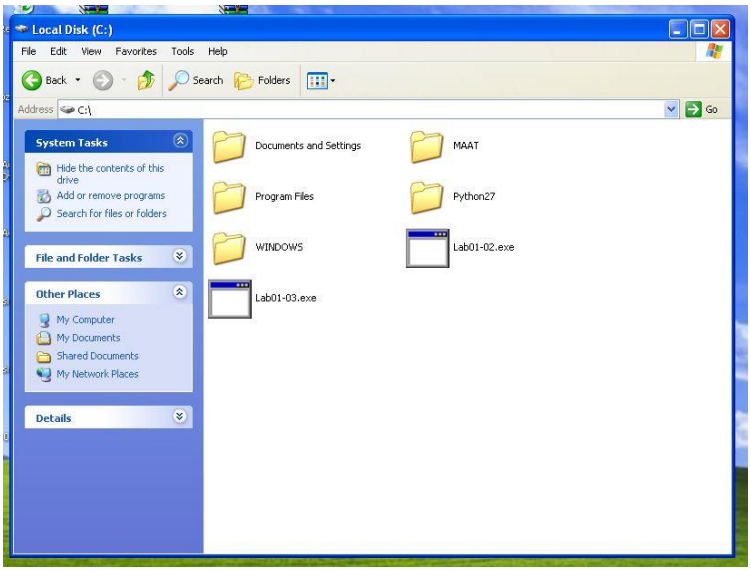
### What type of malware is it?

This malware is a special type of Trojan known as a Remote Access Trojan (RAT). It gives hackers unauthorized remote access to a user's computer using specially configured communication protocols.

### What are the functionalities of the malware?

This malware has the ability to open, write, delete, and copy files in the computer.

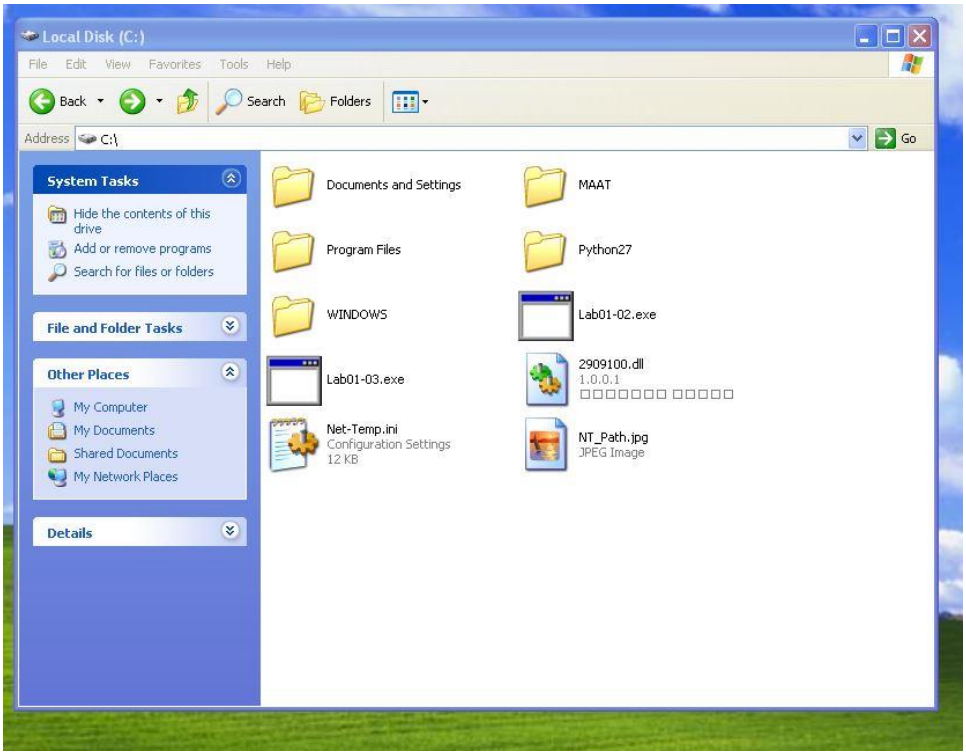
**Before:**



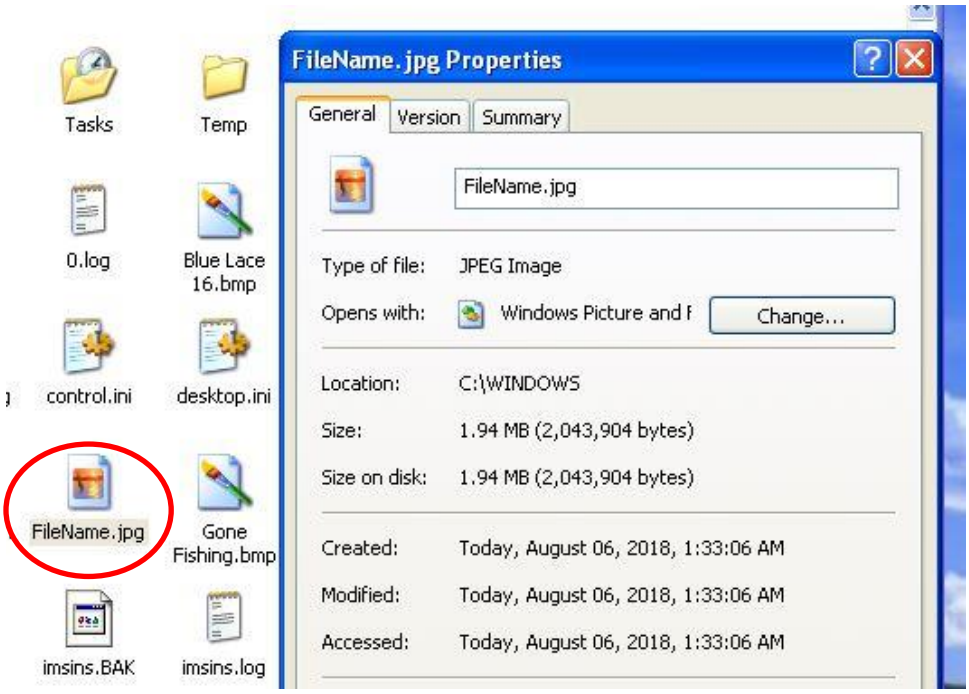
**After:**

When we ran the malware for the first time, the files below were created. However, they were immediately deleted on their own. When we executed the malware the second time, the files shown below did not delete on their own, but stayed in the local disk drive (C:\).

From the screenshot below, the files created under the local disk (C:\) drive were - "**Net-Temp.ini", "NT_Path.jpg"** and "**2909100.dll**" (this dll file name may vary).



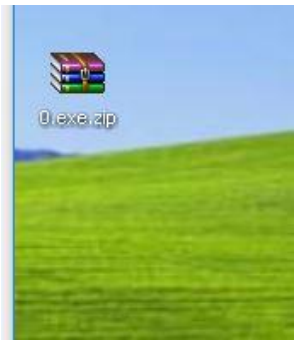

**FileName.jpg** was also created under **C:\WINDOWS**:



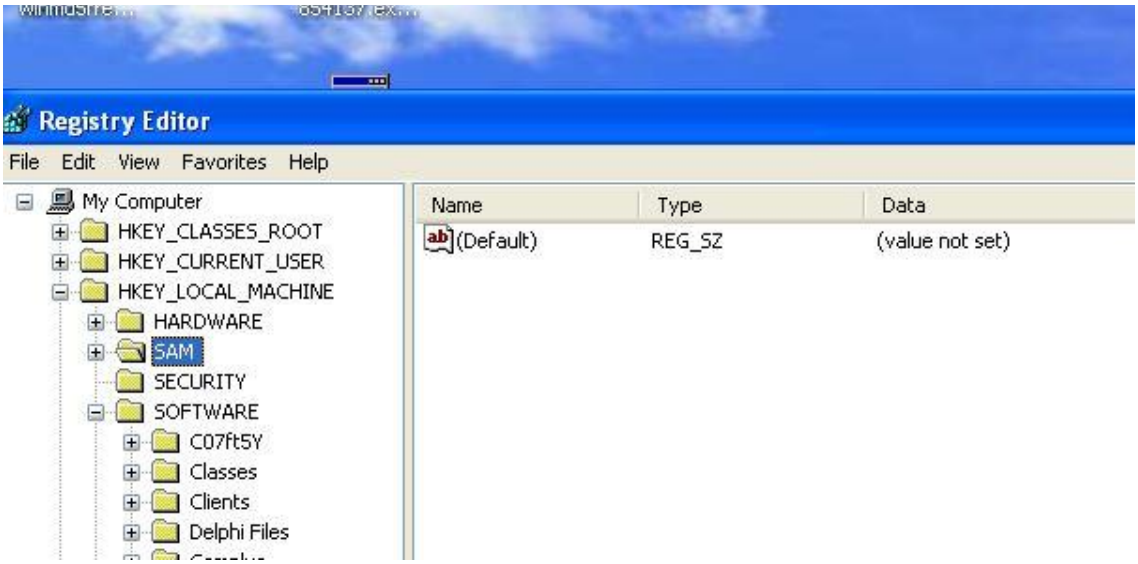

We tried to delete **FileName.jpg** but we could not because we were not authorized to do so. The snapshot below shows an error popup when we tried to delete the FileName.jpg file.


Additionally, we also noticed that when we **first** ran 0.exe, the 0.exe file also deleted itself.

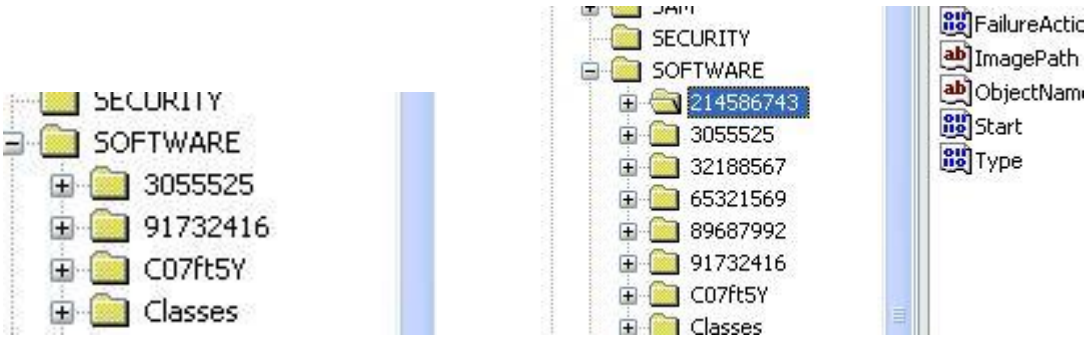
**Before:** 


**After:** 

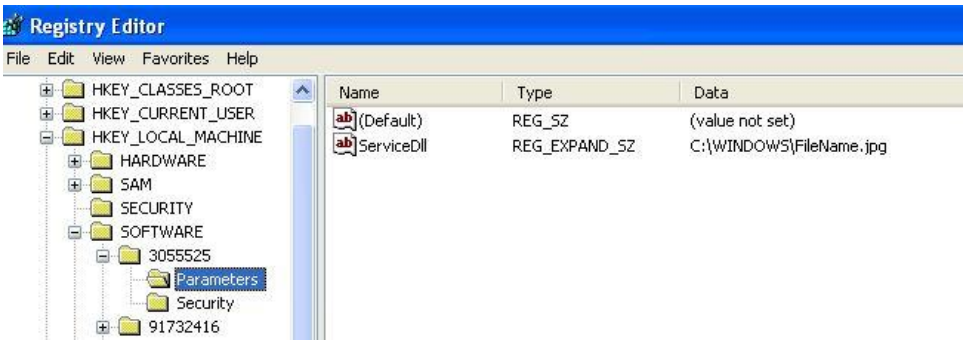

The malware can also create, open registry keys and set/modify registry key values. The snapshot below shows \HKEY_LOCAL_MACHINE\SOFTWARE files before the malware is run:




The snapshots below show the new \HKEY_LOCAL_MACHINE\SOFTWARE when the malware ran. The first snapshot (left) is when the malware first ran, while the second snapshot (right) is when the malware runs for the second time (in the same VM):




When we opened one of the folders, there were subfolders in it showing the data of the registry key. This shows that the malware can also set the data and type values of the registry.




Furthermore, the malware can also modify the registry of a service such as the one under \SYSTEM\ControlSet002\Services\RemoteAccess\RouterManagers\Ip\DLLPath.

The malware is also able to request for privileges and terminate processes. It opens, creates, starts, and stops the services that are running on the computer it has infected. It is also able to load modules into the computer.

**Were you able to interact with the malware? How?**

We had to interact with the malware to find the files/registry that were created, used, or deleted. For our analysis, we used OllyDBG to run the malware and made use of the

step over feature. Those actions allowed us to discover the parameter values of the functions, some of which are the names of the files used, created, or deleted by the malware.