Assignment 7

Adam Livingston

University Of Arizona

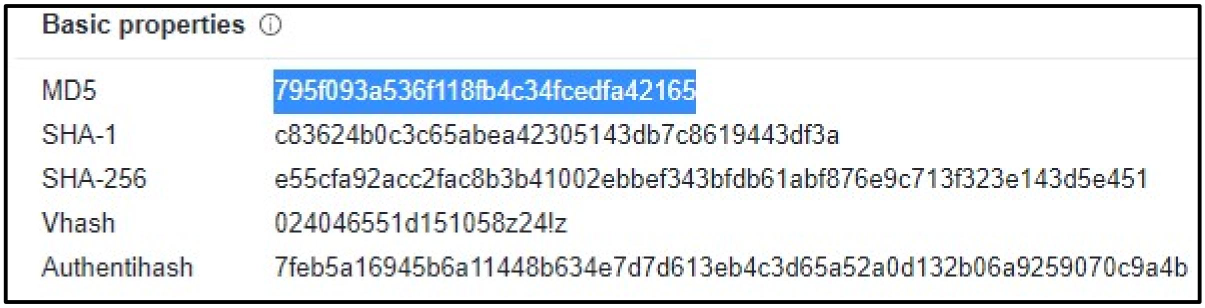
CYBV 454 MALWARE THREATS & ANALYSIS

Professor Galde

3 May 2023

LAB 10-1

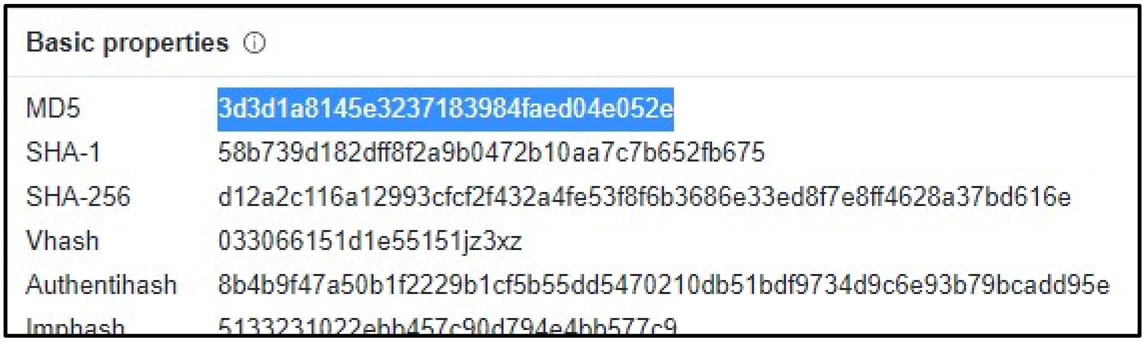
- Lab10-01.exe: 795f093a536f118fb4c34fcedfa42165 (Figure 1)
- Lab10-01.sys: 3d3d1a8145e3237183984faed04e052e (Figure 2)



Basic properties ⓘ

| | |
|---|---|
| MD5 | 795f093a536f118fb4c34fcedfa42165 |
| SHA-1 | c83624b0c3c65abea42305143db7c8619443df3a |
| SHA-256 | e55cfa92acc2fac8b3b41002ebbef343bfdb61abf876e9c713f323e143d5e451 |
| Vhash | 024046551d151058z24!z |
| Authentihash | 7feb5a16945b6a11448b634e7d7d613eb4c3d65a52a0d132b06a9259070c9a4b |

*Figure 1: Virus Total MD5 Hash for file Lab10-01.exe.*

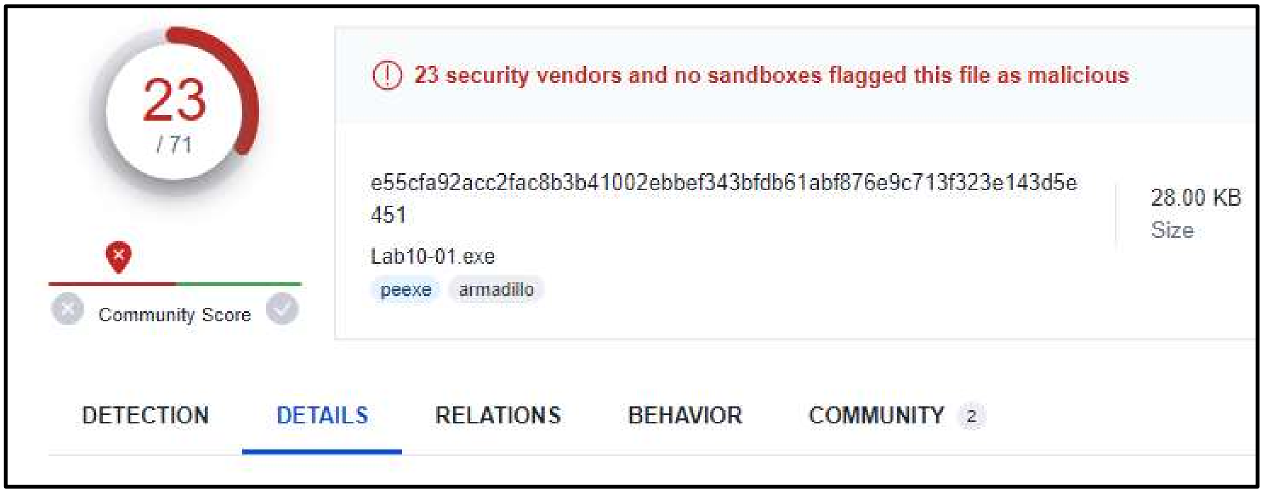


Basic properties ⓘ

| | |
|---|---|
| MD5 | 3d3d1a8145e3237183984faed04e052e |
| SHA-1 | 58b739d182dff8f2a9b0472b10aa7c7b652fb675 |
| SHA-256 | d12a2c116a12993cfcf2f432a4fe53f8f6b3686e33ed8f7e8ff4628a37bd616e |
| Vhash | 033066151d1e55151jz3xz |
| Authentihash | 8b4b9f47a50b1f2229b1cf5b55dd5470210db51bdf9734d9c6e93b79bcadd95e |
| Imphash | 5133231022ebb457c90d794e4bb577c9 |

*Figure 2: Virus Total MD5 Hash for file Lab10-01.sys.*

Virus Total found 23 of 71 matching security vendor signatures for Lab10-01.exe (Figure 3) and has a compilation timestamp of 2011-03-11 at 10:55:44 UTC (Figure 4). There were 29 of 72 signatures for Lab10-01.sys (Figure 5) and has a compilation timestamp of 2012-01-14 at 09:13:34 UTC (Figure 6).



23 security vendors and no sandboxes flagged this file as malicious

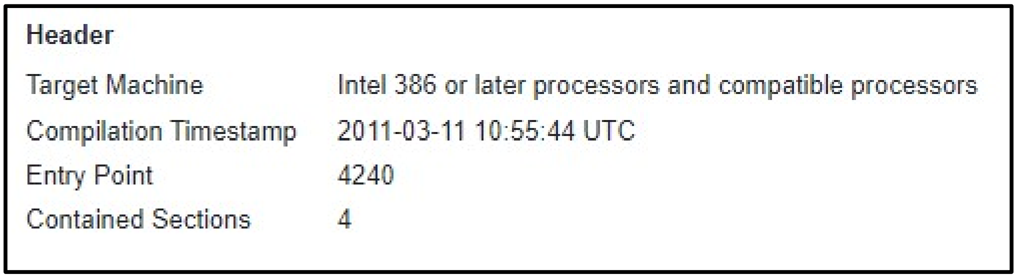
e55cfa92acc2fac8b3b41002ebbef343bfdb61abf876e9c713f323e143d5e451

Lab10-01.exe

peexe armadillo

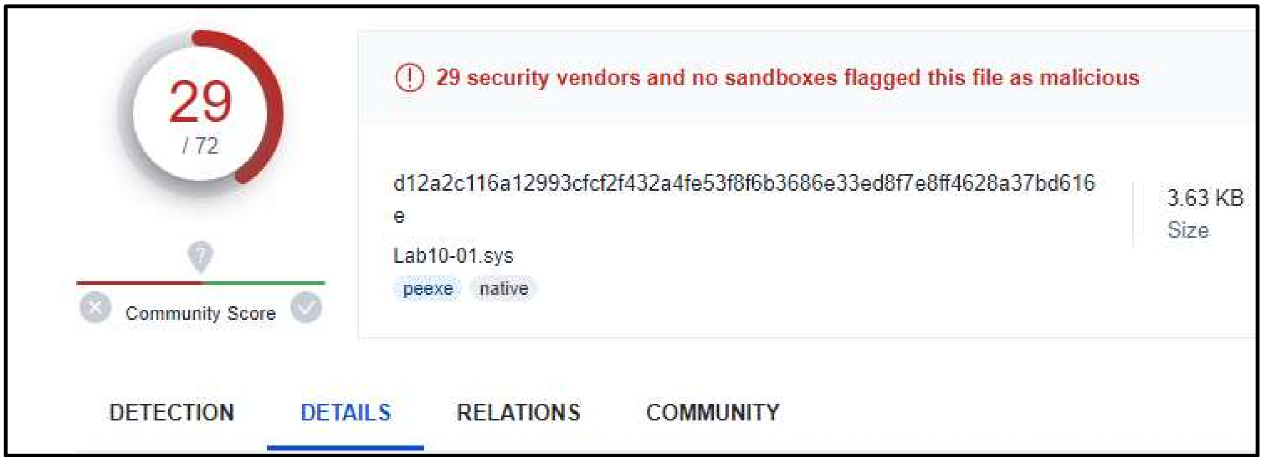
28.00 KB
Size

*Figure 3: Virus Total Findings for file Lab10-01.exe.*



| Header | |
|---|---|
| Target Machine | Intel 386 or later processors and compatible processors |
| Compilation Timestamp | 2011-03-11 10:55:44 UTC |
| Entry Point | 4240 |
| Contained Sections | 4 |

*Figure 4: Virus Total compilation timestamp for Lab10-01.exe.*



29 security vendors and no sandboxes flagged this file as malicious

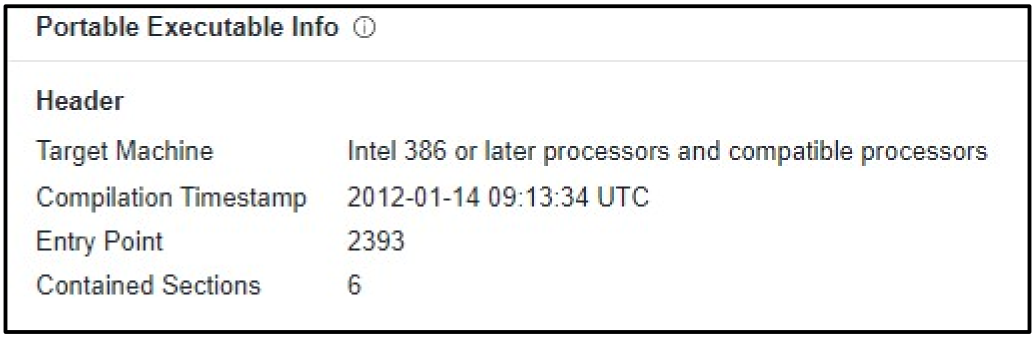
d12a2c116a12993cfcf2f432a4fe53f8f6b3686e33ed8f7e8ff4628a37bd616e

Lab10-01.sys

peexe native

3.63 KB
Size

*Figure 5: Virus Total Findings for Lab10-01.sys.*



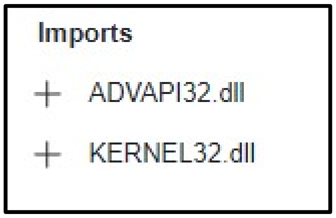
Portable Executable Info ⓘ

Header

| | |
|---|---|
| Target Machine | Intel 386 or later processors and compatible processors |
| Compilation Timestamp | 2012-01-14 09:13:34 UTC |
| Entry Point | 2393 |
| Contained Sections | 6 |

*Figure 6: Virus Total compilation timestamp for Lab10-01.sys.*

The .exe appears to only import two dynamic linked libraries: kerenel32 and advapi32 (Figure 7). Kernel32.dll indicates that it has the capability to access and modify the core OS functions. Advapi32.dll indicates that core Windows components will be altered, such as the Service Manager and Registry. The .sys file only has one import: ntoskrnl.exe (Figure 8). This is the kernel image that is responsible for handling system-level services and is a critical component of Windows OS. This indicates that the malware will manipulate the kernel in some way, potentially using the file to install a rootkit.

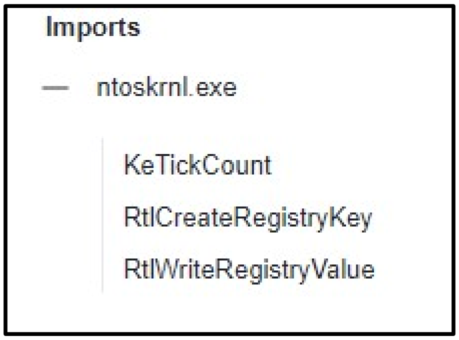


Imports

+ ADVAPI32.dll

+ KERNEL32.dll

*Figure 7: Virus Total imports for Lab10-01.exe.*



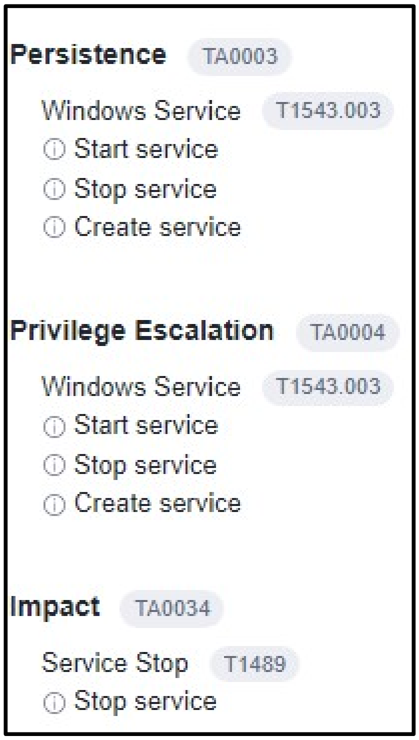
Imports

— ntoskrnl.exe

KeTickCount
RtlCreateRegistryKey
RtlWriteRegistryValue

*Figure 8: Virus Total imports for Lab10-01.sys.*

Virus Total also reports that the .exe file has behaviors of creating, starting, and stopping services, likely indicative of establishing persistence (Figure 9). The .sys file did not show any behaviors from Virus Total, but the naming convention suggests that it is most likely the service that is installed by the .exe.



**Persistence** TA0003

Windows Service T1543.003

ⓘ Start service

ⓘ Stop service

ⓘ Create service

**Privilege Escalation** TA0004

Windows Service T1543.003

ⓘ Start service

ⓘ Stop service

ⓘ Create service

**Impact** TA0034

Service Stop T1489

ⓘ Stop service

*Figure 9: Virus Total behavior for file Lab10-01.exe.*

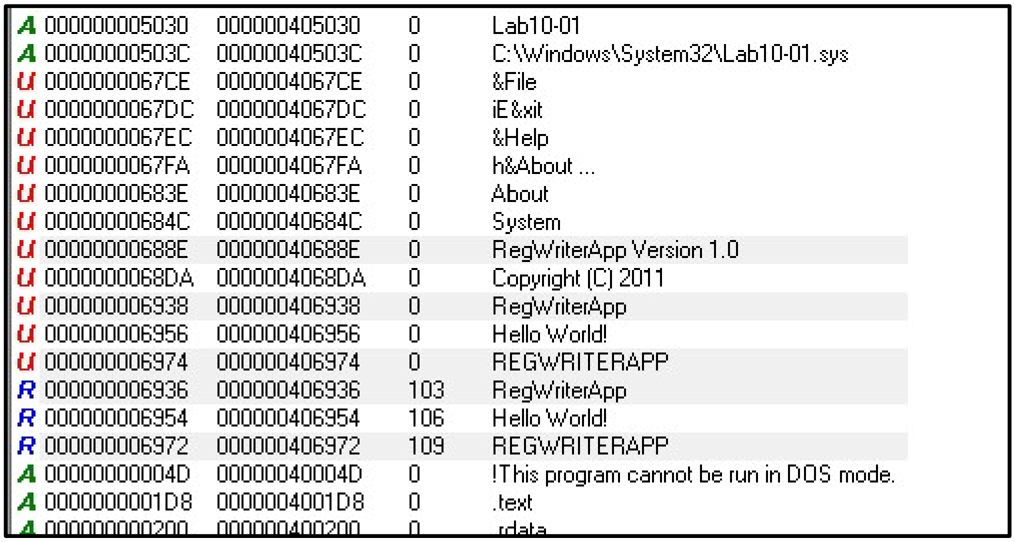
# LAB 10-1

## LAB 10-1 Question 1

**Does this program make any direct changes to the registry? (Use procmon to check.)**

**BLUF**: Yes, using the RegSetValue

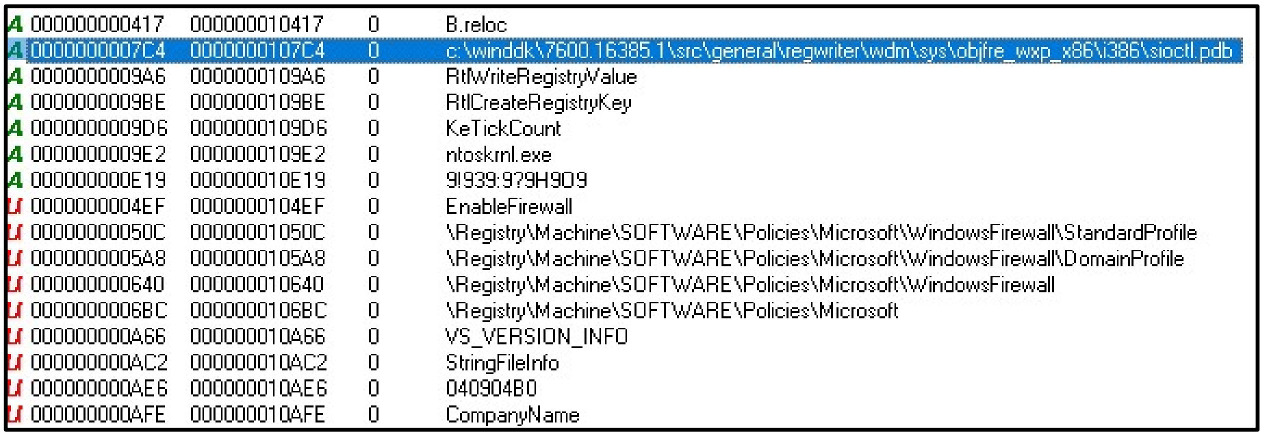
Both files were first run through a brief static analysis. BinText identified some registry-related strings, such as RegWriterApp (Figure 10). Also within Figure 10, the System32 directory is invoked and Lab10-01.sys is referenced. This could potentially be the .exe file copying the .sys file into the System32 directory, most likely where the .sys file will invoke ntoskrnl.exe as shown in Figure 8.



```
A 000000005030   000000405030   0     Lab10-01
A 00000000503C   00000040503C   0     C:\Windows\System32\Lab10-01.sys
U 0000000067CE   0000004067CE   0     &File
U 0000000067DC   0000004067DC   0     iE&xit
U 0000000067EC   0000004067EC   0     &Help
U 0000000067FA   0000004067FA   0     h&About ...
U 00000000683E   00000040683E   0     About
U 00000000684C   00000040684C   0     System
U 00000000688E   00000040688E   0     RegWriterApp Version 1.0
U 0000000068DA   0000004068DA   0     Copyright (C) 2011
U 000000006938   000000406938   0     RegWriterApp
U 000000006956   000000406956   0     Hello World!
U 000000006974   000000406974   0     REGWRITERAPP
R 000000006936   000000406936   103   RegWriterApp
R 000000006954   000000406954   106   Hello World!
R 000000006972   000000406972   109   REGWRITERAPP
A 00000000004D   00000040004D   0     !This program cannot be run in DOS mode.
A 0000000001D8   0000004001D8   0     .text
A 000000000200   000000400200   0     .rdata
```

*Figure 10: Lab10-01.exe calling 'RegWriterApp'.*

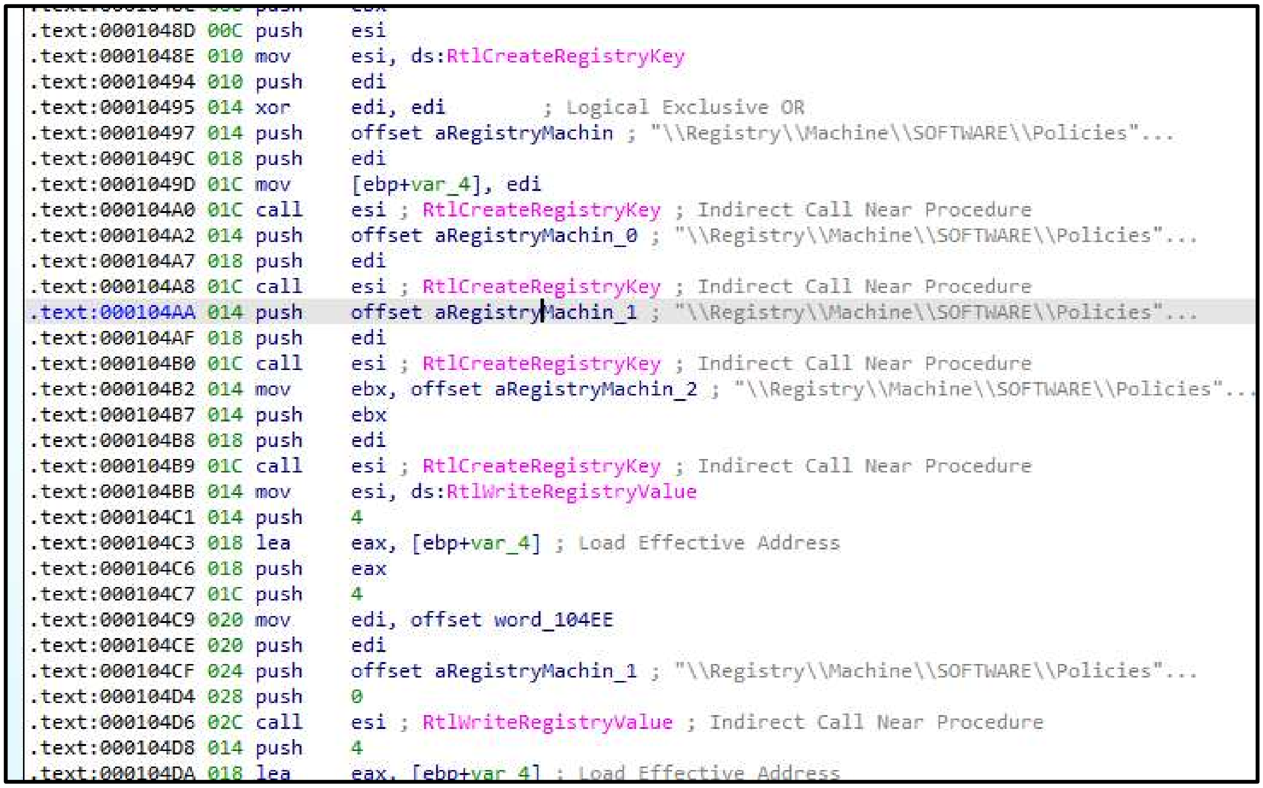
Lab10-01.sys has even more calls to the registry. Some especially-concerning references were to the Windows Firewall (Figure 11). We see calls to "RtlWriteRegistryValue" and "RtlCreateRegistryKey" prior to the Windows Firewall path, both being kernel-mode component functions. There is also an import showing a file path to the C:\ drive in the \winddk\ directory, which is the windows driver kit and contains tools for development.



```
A 000000000417   000000010417   0   B.reloc
A 0000000007C4   0000000107C4   0   c:\winddk\7600.16385.1\src\general\regwriter\wdm\sys\objfre_wxp_x86\i386\sioctl.pdb
A 0000000009A6   0000000109A6   0   RtlWriteRegistryValue
A 0000000009BE   0000000109BE   0   RtlCreateRegistryKey
A 0000000009D6   0000000109D6   0   KeTickCount
A 0000000009E2   0000000109E2   0   ntoskrnl.exe
A 000000000E19   000000010E19   0   9!939:9?9H9O9
U 0000000004EF   0000000104EF   0   EnableFirewall
U 00000000050C   00000001050C   0   \Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall\StandardProfile
U 0000000005A8   0000000105A8   0   \Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall\DomainProfile
U 000000000640   000000010640   0   \Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall
U 0000000006BC   0000000106BC   0   \Registry\Machine\SOFTWARE\Policies\Microsoft
U 000000000A66   000000010A66   0   VS_VERSION_INFO
U 000000000AC2   000000010AC2   0   StringFileInfo
U 000000000AE6   000000010AE6   0   040904B0
U 000000000AFE   000000010AFE   0   CompanyName
```

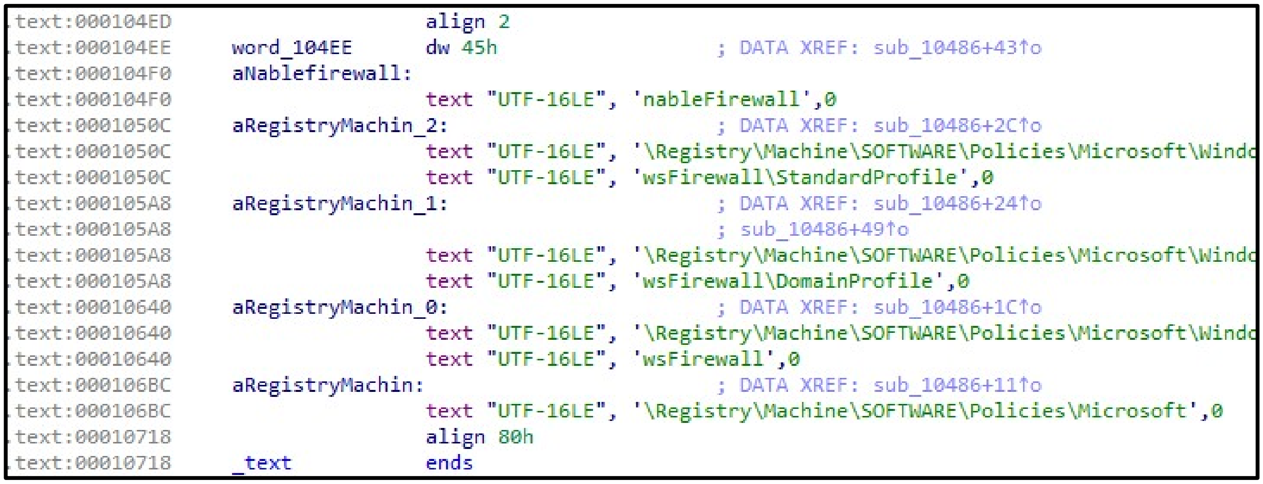
*Figure 11: Lab10-01.sys suspicious driver creation and registry calls.*

We also see in IDA Pro of the .sys file the same paths being pushed onto the stack prior to the registry functions being called (Figures 12 and 13). This means that we can first expect a new key to be created first within SOFTWARE\Policies\WindowsFirewall. Then, the next two keys can be expected to be created in two subfolders: \StandardProfile and \DomainProfile. It will then write a hex value of 0x45 (decimal 69, ASCII "E") to the \DomainProfile key.



```
.text:0001048D 00C push    esi
.text:0001048E 010 mov     esi, ds:RtlCreateRegistryKey
.text:00010494 010 push    edi
.text:00010495 014 xor     edi, edi        ; Logical Exclusive OR
.text:00010497 014 push    offset aRegistryMachin ; "\\Registry\\Machine\\SOFTWARE\\Policies"...
.text:0001049C 018 push    edi
.text:0001049D 01C mov     [ebp+var_4], edi
.text:000104A0 01C call    esi ; RtlCreateRegistryKey ; Indirect Call Near Procedure
.text:000104A2 014 push    offset aRegistryMachin_0 ; "\\Registry\\Machine\\SOFTWARE\\Policies"...
.text:000104A7 018 push    edi
.text:000104A8 01C call    esi ; RtlCreateRegistryKey ; Indirect Call Near Procedure
.text:000104AA 014 push    offset aRegistryMachin_1 ; "\\Registry\\Machine\\SOFTWARE\\Policies"...
.text:000104AF 018 push    edi
.text:000104B0 01C call    esi ; RtlCreateRegistryKey ; Indirect Call Near Procedure
.text:000104B2 014 mov     ebx, offset aRegistryMachin_2 ; "\\Registry\\Machine\\SOFTWARE\\Policies"...
.text:000104B7 014 push    ebx
.text:000104B8 018 push    edi
.text:000104B9 01C call    esi ; RtlCreateRegistryKey ; Indirect Call Near Procedure
.text:000104BB 014 mov     esi, ds:RtlWriteRegistryValue
.text:000104C1 014 push    4
.text:000104C3 018 lea     eax, [ebp+var_4] ; Load Effective Address
.text:000104C6 018 push    eax
.text:000104C7 01C push    4
.text:000104C9 020 mov     edi, offset word_104EE
.text:000104CE 020 push    edi
.text:000104CF 024 push    offset aRegistryMachin_1 ; "\\Registry\\Machine\\SOFTWARE\\Policies"...
.text:000104D4 028 push    0
.text:000104D6 02C call    esi ; RtlWriteRegistryValue ; Indirect Call Near Procedure
.text:000104D8 014 push    4
.text:000104DA 018 lea     eax, [ebp+var_4] ; Load Effective Address
```

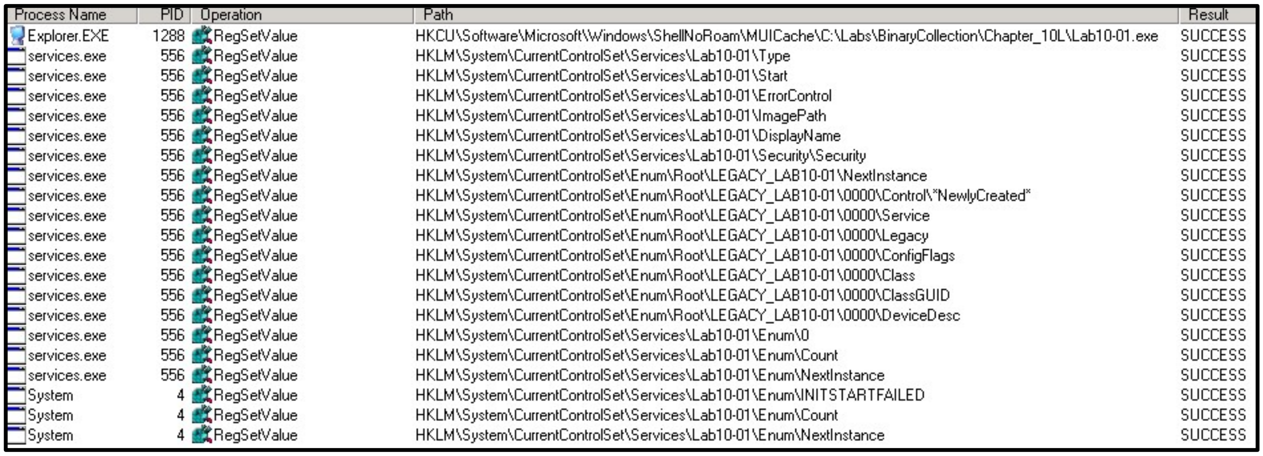
*Figure 12: Lab10-01.sys creating registry keys.*



```
.text:000104ED                 align 2
.text:000104EE     word_104EE      dw 45h                  ; DATA XREF: sub_10486+43↑o
.text:000104F0     aNablefirewall:
.text:000104F0                 text "UTF-16LE", 'nableFirewall',0
.text:0001050C     aRegistryMachin_2:                      ; DATA XREF: sub_10486+2C↑o
.text:0001050C                 text "UTF-16LE", '\Registry\Machine\SOFTWARE\Policies\Microsoft\Windo
.text:0001050C                 text "UTF-16LE", 'wsFirewall\StandardProfile',0
.text:000105A8     aRegistryMachin_1:                      ; DATA XREF: sub_10486+24↑o
.text:000105A8                                         ; sub_10486+49↑o
.text:000105A8                 text "UTF-16LE", '\Registry\Machine\SOFTWARE\Policies\Microsoft\Windo
.text:000105A8                 text "UTF-16LE", 'wsFirewall\DomainProfile',0
.text:00010640     aRegistryMachin_0:                      ; DATA XREF: sub_10486+1C↑o
.text:00010640                 text "UTF-16LE", '\Registry\Machine\SOFTWARE\Policies\Microsoft\Windo
.text:00010640                 text "UTF-16LE", 'wsFirewall',0
.text:000106BC     aRegistryMachin:                        ; DATA XREF: sub_10486+11↑o
.text:000106BC                 text "UTF-16LE", '\Registry\Machine\SOFTWARE\Policies\Microsoft',0
.text:00010718                 align 80h
.text:00010718     _text           ends
```

*Figure 13: Lab10-01.sys registry key paths.*

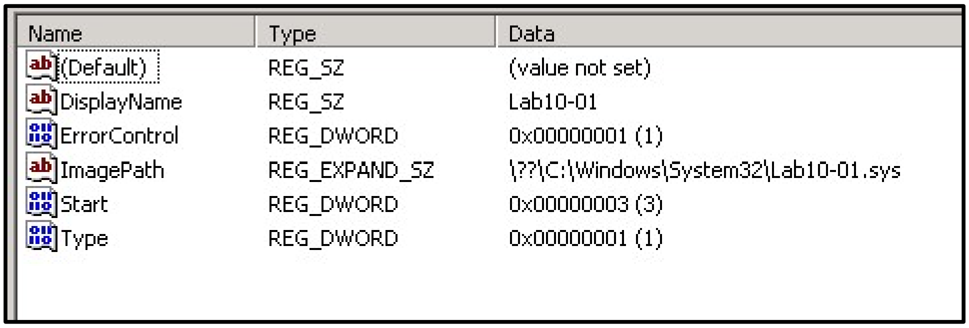
Lab10-01.exe was then executed and activity was captured using procmon. When a filter was set to include the operation "RegCreateKey", nothing populated on the capture. However, multiple events for the operation "RegSetValue" were captured (Figure 14). RegSetValue is a Windows API function used to set data and type of a value in a registry key. Additionally, many of the RegSetValue operations were for the registry location \CurrentControlSet\Services, indicating persistence being established. These were not created by Lab10-01.exe, but rather services.exe.



| Process Name | PID | Operation | Path | Result |
|---|---|---|---|---|
| Explorer.EXE | 1288 | RegSetValue | HKCU\Software\Microsoft\Windows\ShellNoRoam\MUICache\C:\Labs\BinaryCollection\Chapter_10L\Lab10-01.exe | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Type | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Start | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\ErrorControl | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\ImagePath | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\DisplayName | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Security\Security | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Enum\Root\LEGACY_LAB10-01\NextInstance | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Enum\Root\LEGACY_LAB10-01\0000\Control\*NewlyCreated* | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Enum\Root\LEGACY_LAB10-01\0000\Service | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Enum\Root\LEGACY_LAB10-01\0000\Legacy | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Enum\Root\LEGACY_LAB10-01\0000\ConfigFlags | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Enum\Root\LEGACY_LAB10-01\0000\Class | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Enum\Root\LEGACY_LAB10-01\0000\ClassGUID | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Enum\Root\LEGACY_LAB10-01\0000\DeviceDesc | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Enum\0 | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Enum\Count | SUCCESS |
| services.exe | 556 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Enum\NextInstance | SUCCESS |
| System | 4 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Enum\INITSTARTFAILED | SUCCESS |
| System | 4 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Enum\Count | SUCCESS |
| System | 4 | RegSetValue | HKLM\System\CurrentControlSet\Services\Lab10-01\Enum\NextInstance | SUCCESS |

*Figure 14: Lab10-01.sys registry key paths.*

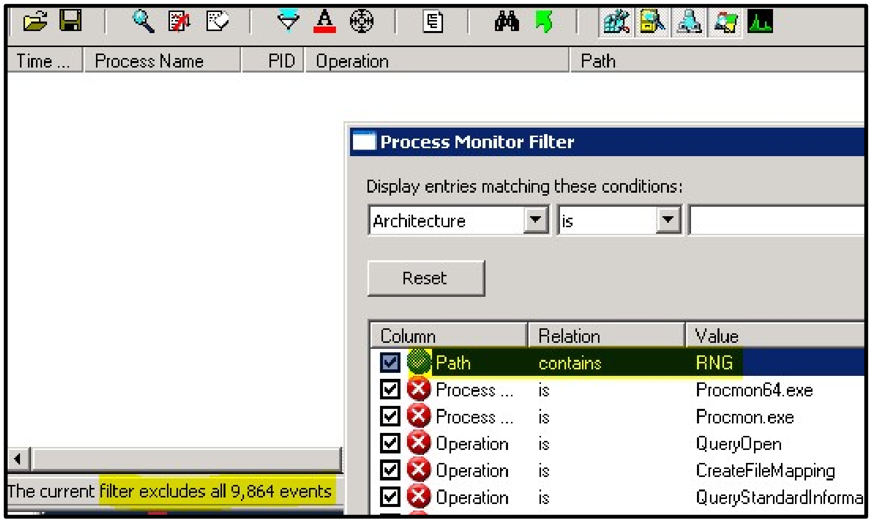
This registry key creation was confirmed when looking at Registry Editor. There is an image path, as expected, for C:\Windows\System32 for the .sys file (Figure 15). However, the file was not found when the directory was examined in File Explorer.



| Name | Type | Data |
|---|---|---|
| (Default) | REG_SZ | (value not set) |
| DisplayName | REG_SZ | Lab10-01 |
| ErrorControl | REG_DWORD | 0x00000001 (1) |
| ImagePath | REG_EXPAND_SZ | \??\C:\Windows\System32\Lab10-01.sys |
| Start | REG_DWORD | 0x00000003 (3) |
| Type | REG_DWORD | 0x00000001 (1) |

*Figure 15: Confirming new service in regedit.*

Additionally in Figure 14, we see new keys being created in CurrentControlSet\Enum\Root, the path containing information about all connected hardware devices. However, this was not found within regedit. There was not any data captured by procmon to the expected registry value being written to Microsoft\Cryptography\RNG\Seed (according to the book). The procmon capture lasted 5 minutes after running Lab10-01.exe, yet no path containing the string “RNG” was captured (Figure 16). This was tested in both Windows XP and Windows 10 OS environments.



Process Monitor Filter

Display entries matching these conditions:

Architecture | is

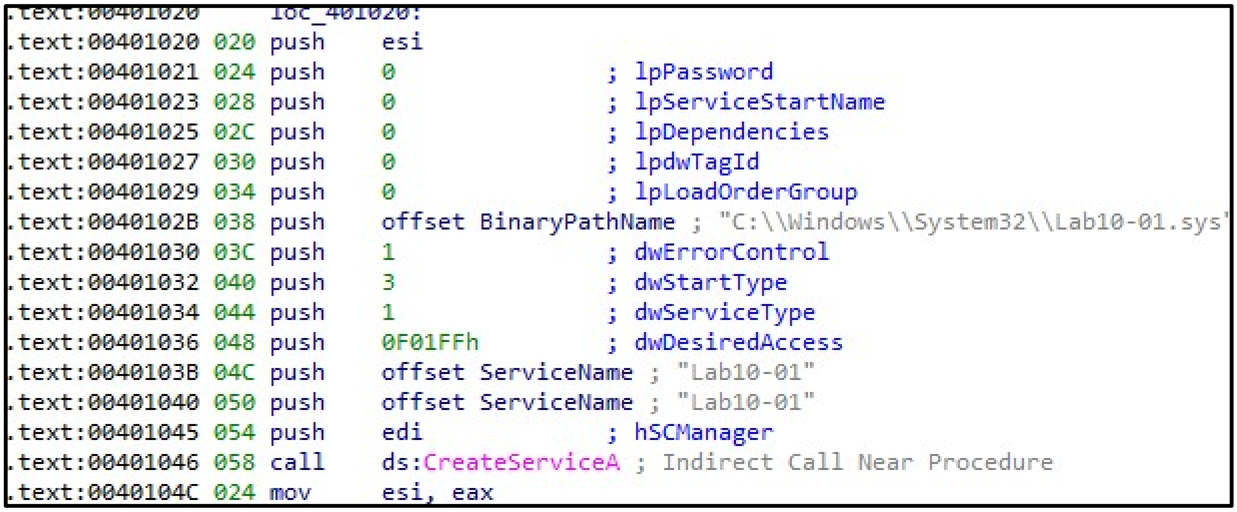
Reset

| | Column | Relation | Value |
|---|---|---|---|
| ☑ | Path | contains | RNG |
| ☑ | Process ... | is | Procmon64.exe |
| ☑ | Process ... | is | Procmon.exe |
| ☑ | Operation | is | QueryOpen |
| ☑ | Operation | is | CreateFileMapping |
| ☑ | Operation | is | QueryStandardInforma |

The current filter excludes all 9,864 events

*Figure 16: Path specified in book not captured in procmon.*

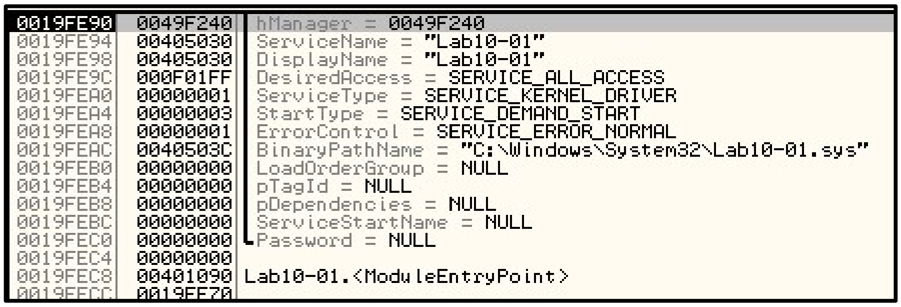
Because the .sys file creates the registry keys and Lab10-01.exe can only be run through OllyDbg, a breakpoint was set at 0x40102B where the .sys file is pushed onto the stack prior to “CreateServiceA” was passed as the lpBinaryPathName parameter (Figure 17).



```
.text:00401020     loc_401020:
.text:00401020 020 push    esi
.text:00401021 024 push    0               ; lpPassword
.text:00401023 028 push    0               ; lpServiceStartName
.text:00401025 02C push    0               ; lpDependencies
.text:00401027 030 push    0               ; lpdwTagId
.text:00401029 034 push    0               ; lpLoadOrderGroup
.text:0040102B 038 push    offset BinaryPathName ; "C:\\Windows\\System32\\Lab10-01.sys"
.text:00401030 03C push    1               ; dwErrorControl
.text:00401032 040 push    3               ; dwStartType
.text:00401034 044 push    1               ; dwServiceType
.text:00401036 048 push    0F01FFh         ; dwDesiredAccess
.text:0040103B 04C push    offset ServiceName ; "Lab10-01"
.text:00401040 050 push    offset ServiceName ; "Lab10-01"
.text:00401045 054 push    edi             ; hSCManager
.text:00401046 058 call    ds:CreateServiceA ; Indirect Call Near Procedure
.text:0040104C 024 mov     esi, eax
```

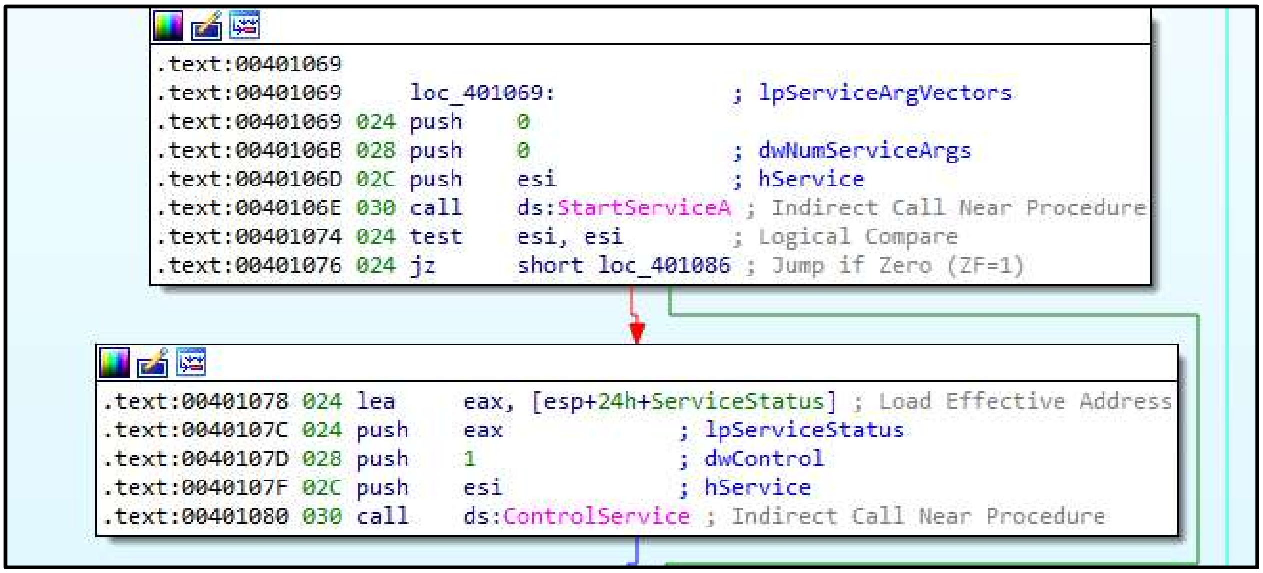
*Figure 17: Path for .sys file prior to call to CreateServiceA.*

Before CreateServiceA is called, we see in the stack that the created service is a service type of SERVICE_KERNEL_DRIVER (Figure 18). Additionally, the .exe file calls the functions StartServiceA at 0x40106E and ControlService at 0x401080 (Figure 19). In the documentation for ControlService, we see that the value of 1 pushed for dwControl means that it will stop the service.



```
0019FE90  0049F240  hManager = 0049F240
0019FE94  00405030  ServiceName = "Lab10-01"
0019FE98  00405030  DisplayName = "Lab10-01"
0019FE9C  000F01FF  DesiredAccess = SERVICE_ALL_ACCESS
0019FEA0  00000001  ServiceType = SERVICE_KERNEL_DRIVER
0019FEA4  00000003  StartType = SERVICE_DEMAND_START
0019FEA8  00000001  ErrorControl = SERVICE_ERROR_NORMAL
0019FEAC  0040503C  BinaryPathName = "C:\Windows\System32\Lab10-01.sys"
0019FEB0  00000000  LoadOrderGroup = NULL
0019FEB4  00000000  pTagId = NULL
0019FEB8  00000000  pDependencies = NULL
0019FEBC  00000000  ServiceStartName = NULL
0019FEC0  00000000  Password = NULL
0019FEC4  00000000
0019FEC8  00401090 Lab10-01.<ModuleEntryPoint>
0019FECC  0019FF70
```

*Figure 18: CreateServiceA arguments on stack.*

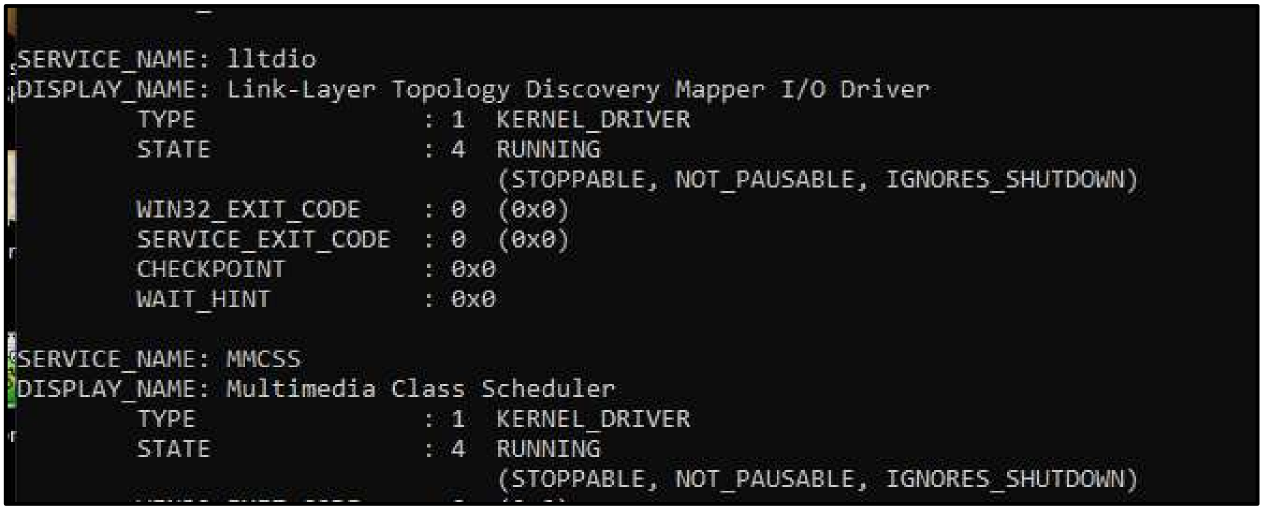


```
.text:00401069
.text:00401069     loc_401069:             ; lpServiceArgVectors
.text:00401069 024 push    0
.text:0040106B 028 push    0               ; dwNumServiceArgs
.text:0040106D 02C push    esi             ; hService
.text:0040106E 030 call    ds:StartServiceA ; Indirect Call Near Procedure
.text:00401074 024 test    esi, esi        ; Logical Compare
.text:00401076 024 jz      short loc_401086 ; Jump if Zero (ZF=1)
```

```
.text:00401078 024 lea     eax, [esp+24h+ServiceStatus] ; Load Effective Address
.text:0040107C 024 push    eax             ; lpServiceStatus
.text:0040107D 028 push    1               ; dwControl
.text:0040107F 02C push    esi             ; hService
.text:00401080 030 call    ds:ControlService ; Indirect Call Near Procedure
```

*Figure 19: CreateServiceA and ControlService in IDA.*

Since Olly can only analyze .sys files that are running, we first confirm the if the service is running. The command line tool "sc" was used to find kernel drivers, but the service was not found (Figure 21) and therefore cannot be debugged with Olly. This was tested on both a Windows 10 and Windows XP VM.

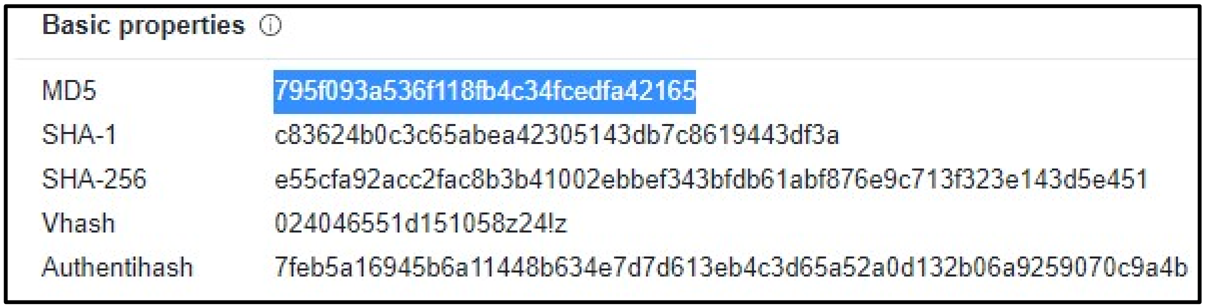


```
SERVICE_NAME: lltdio
DISPLAY_NAME: Link-Layer Topology Discovery Mapper I/O Driver
        TYPE               : 1  KERNEL_DRIVER
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

SERVICE_NAME: MMCSS
DISPLAY_NAME: Multimedia Class Scheduler
        TYPE               : 1  KERNEL_DRIVER
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
```

*Figure 20: CreateServiceA and ControlService in IDA.*

LAB 10-2

- Lab10-02.exe: 3f3a29ca2467d2d05feac9d233366f45 (Figure 21)

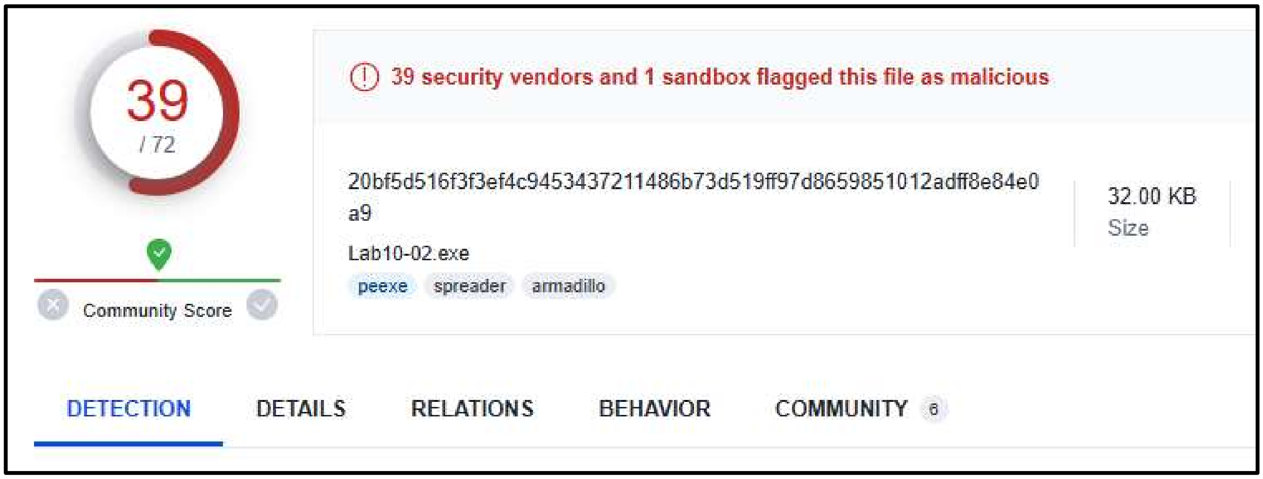


Basic properties ⓘ

| | |
|---|---|
| MD5 | 795f093a536f118fb4c34fcedfa42165 |
| SHA-1 | c83624b0c3c65abea42305143db7c8619443df3a |
| SHA-256 | e55cfa92acc2fac8b3b41002ebbef343bfdb61abf876e9c713f323e143d5e451 |
| Vhash | 024046551d151058z24!z |
| Authentihash | 7feb5a16945b6a11448b634e7d7d613eb4c3d65a52a0d132b06a9259070c9a4b |

*Figure 21: Virus Total MD5 Hash for file Lab10-02.exe.*

Virus Total found 39 of 72 matching security vendor signatures for Lab10-02.exe (Figure 22) and has a compilation timestamp of 2010-12-31 at 15:33:33 UTC (Figure 23).



ⓘ 39 security vendors and 1 sandbox flagged this file as malicious

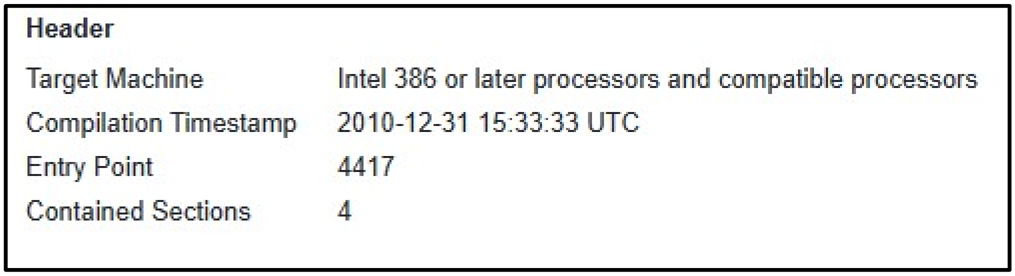
20bf5d516f3f3ef4c9453437211486b73d519ff97d8659851012adff8e84e0a9

Lab10-02.exe

peexe spreader armadillo

32.00 KB
Size

*Figure 22: Virus Total Findings for file Lab10-02.exe.*

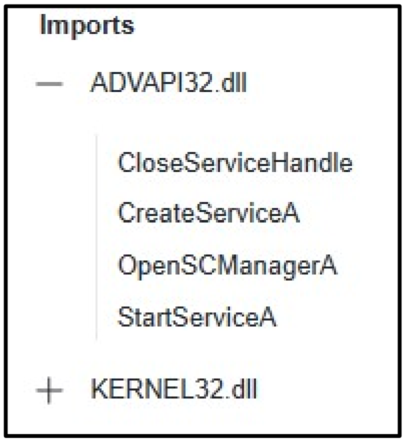


Header

| | |
|---|---|
| Target Machine | Intel 386 or later processors and compatible processors |
| Compilation Timestamp | 2010-12-31 15:33:33 UTC |
| Entry Point | 4417 |
| Contained Sections | 4 |

*Figure 23: Virus Total compilation timestamp for Lab10-02.exe.*

The file appears to only import two dynamic linked libraries: kerenel32 and advapi32 (Figure 24). Kernel32.dll indicates that it has the capability to access and modify the core OS functions. Advapi32.dll indicates that core Windows components will be altered, such as the Service Manager and Registry. It appears that this file will also create and start a service, most likely to establish persistence.



Imports

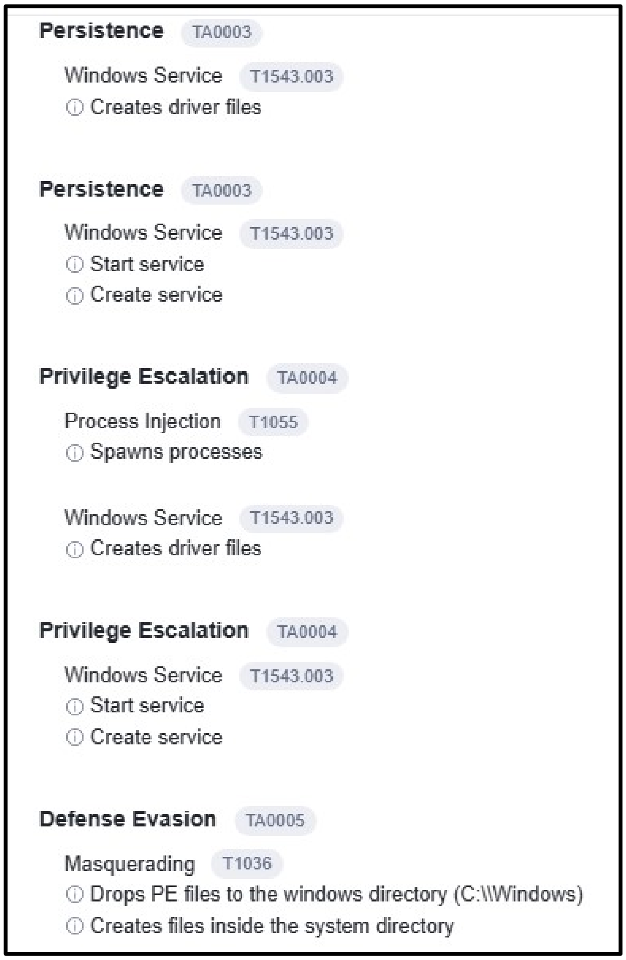
— ADVAPI32.dll

CloseServiceHandle
CreateServiceA
OpenSCManagerA
StartServiceA

+ KERNEL32.dll

*Figure 24: Virus Total imports for Lab10-02.exe.*

Virus Total also reports that the file has behaviors of creating and starting stopping services, likely indicative of establishing persistence (Figure 25). It also shows behaviors of creating driver files and dropping PE files into the C:\Windows directory.



**Persistence** TA0003

Windows Service T1543.003

ⓘ Creates driver files

**Persistence** TA0003

Windows Service T1543.003

ⓘ Start service

ⓘ Create service

**Privilege Escalation** TA0004

Process Injection T1055

ⓘ Spawns processes

Windows Service T1543.003

ⓘ Creates driver files

**Privilege Escalation** TA0004

Windows Service T1543.003

ⓘ Start service

ⓘ Create service

**Defense Evasion** TA0005

Masquerading T1036

ⓘ Drops PE files to the windows directory (C:\\Windows)

ⓘ Creates files inside the system directory

*Figure 25: Virus Total behavior for file Lab10-01.exe.*

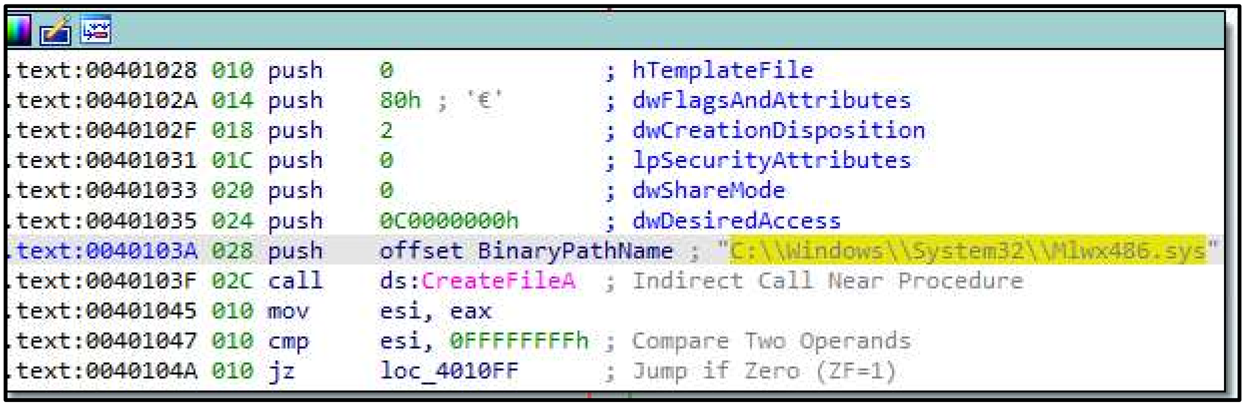
## LAB 10-2

### LAB 10-2 Question 1

**Does this program create any files? If so, what are they?**

**BLUF**:

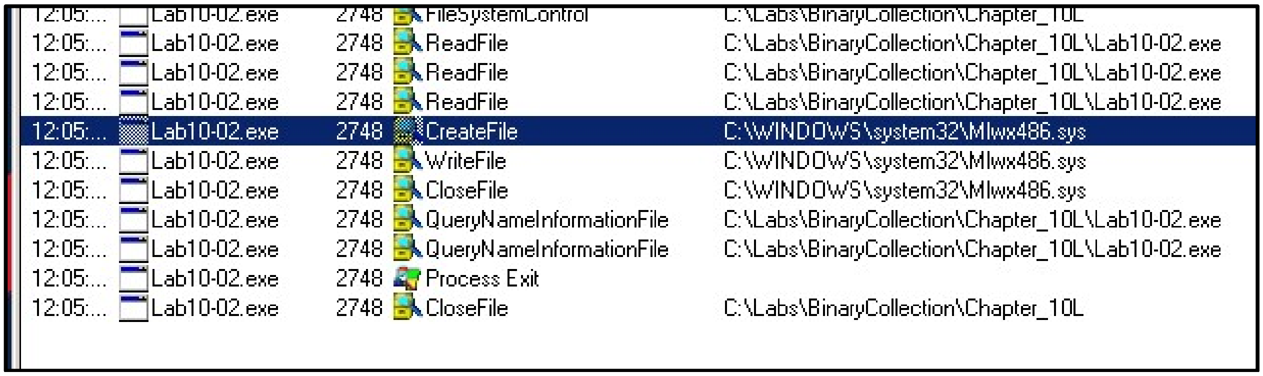
When analyzing the malware within IDA Pro, there is an external function import of "CreateFileA" and only called once at 0x40103F. Preceding this call, there is a string value pushed onto the stack containing the path "C:\Windows\System32\Mlwx486.sys" (Figure 26). It can therefore be assumed that the file name this malware will create will be in the System32 directory and is a driver file named "Mlwx486.sys".



```
.text:00401028 010 push    0                 ; hTemplateFile
.text:0040102A 014 push    80h ; '€'         ; dwFlagsAndAttributes
.text:0040102F 018 push    2                 ; dwCreationDisposition
.text:00401031 01C push    0                 ; lpSecurityAttributes
.text:00401033 020 push    0                 ; dwShareMode
.text:00401035 024 push    0C0000000h        ; dwDesiredAccess
.text:0040103A 028 push    offset BinaryPathName ; "C:\\Windows\\System32\\Mlwx486.sys"
.text:0040103F 02C call    ds:CreateFileA    ; Indirect Call Near Procedure
.text:00401045 010 mov     esi, eax
.text:00401047 010 cmp     esi, 0FFFFFFFFh ; Compare Two Operands
.text:0040104A 010 jz      loc_4010FF        ; Jump if Zero (ZF=1)
```

*Figure 26: IDA Pro showing the path of the created file.*

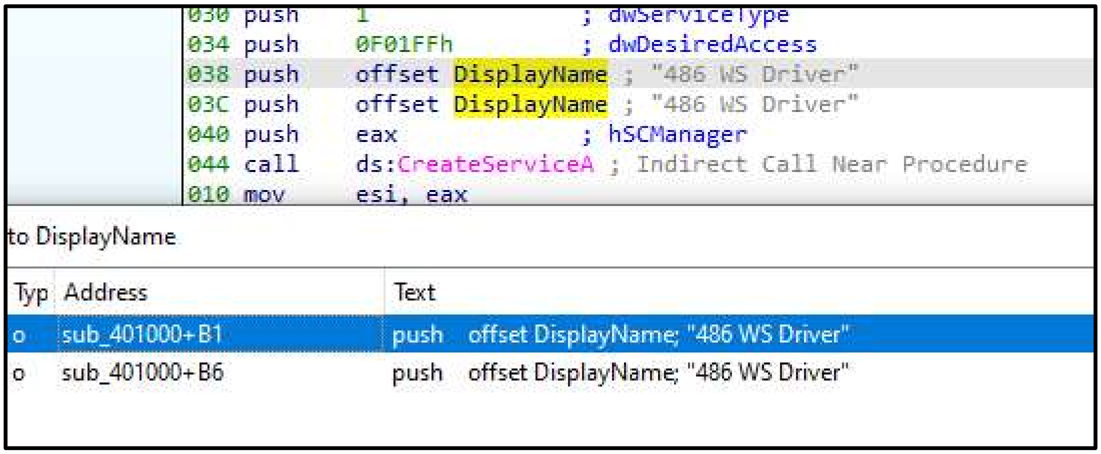
To confirm this, the malware was executed with procmon capturing event activity. The file identified within IDA Pro was confirmed as being created (Figure 27). However, this file could not be found within System32. There was also only one call to this file in IDA Pro to create it, no calls to delete this file in the code, and no file deletion operations done on the file in the procmon capture.



```
12:05:...  Lab10-02.exe  2748  FileSystemControl       C:\Labs\BinaryCollection\Chapter_10L
12:05:...  Lab10-02.exe  2748  ReadFile                C:\Labs\BinaryCollection\Chapter_10L\Lab10-02.exe
12:05:...  Lab10-02.exe  2748  ReadFile                C:\Labs\BinaryCollection\Chapter_10L\Lab10-02.exe
12:05:...  Lab10-02.exe  2748  ReadFile                C:\Labs\BinaryCollection\Chapter_10L\Lab10-02.exe
12:05:...  Lab10-02.exe  2748  CreateFile              C:\WINDOWS\system32\Mlwx486.sys
12:05:...  Lab10-02.exe  2748  WriteFile               C:\WINDOWS\system32\Mlwx486.sys
12:05:...  Lab10-02.exe  2748  CloseFile               C:\WINDOWS\system32\Mlwx486.sys
12:05:...  Lab10-02.exe  2748  QueryNameInformationFile C:\Labs\BinaryCollection\Chapter_10L\Lab10-02.exe
12:05:...  Lab10-02.exe  2748  QueryNameInformationFile C:\Labs\BinaryCollection\Chapter_10L\Lab10-02.exe
12:05:...  Lab10-02.exe  2748  Process Exit
12:05:...  Lab10-02.exe  2748  CloseFile               C:\Labs\BinaryCollection\Chapter_10L
```

*Figure 27: Procmon captured the file creation.*

To find out where this file potentially went, we go back into IDA Pro and find xrefs to CreateServiceA. We see that a service name of “486 WS Driver” is pushed onto the stack twice before the function is called (Figure 28). These to pushes are the only xrefs to “486 WS Driver”.



```
030 push    1                   ; dwServiceType
034 push    0F01FFh             ; dwDesiredAccess
038 push    offset DisplayName  ; "486 WS Driver"
03C push    offset DisplayName  ; "486 WS Driver"
040 push    eax                 ; hSCManager
044 call    ds:CreateServiceA   ; Indirect Call Near Procedure
010 mov     esi, eax
```

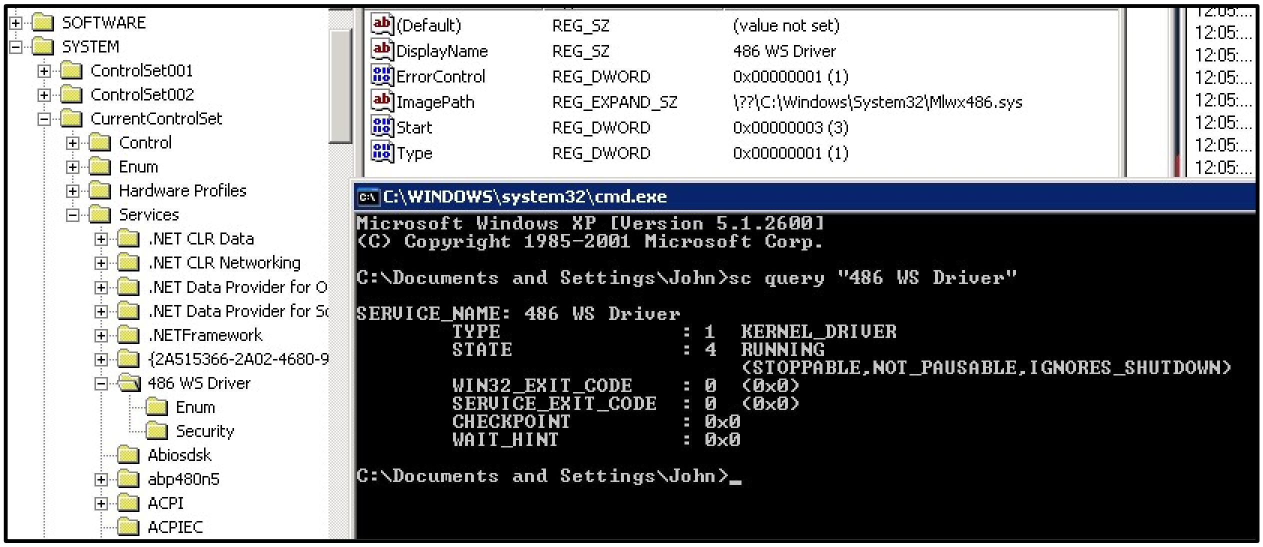
to DisplayName

| Typ | Address | Text |
|---|---|---|
| o | sub_401000+B1 | push offset DisplayName; "486 WS Driver" |
| o | sub_401000+B6 | push offset DisplayName; "486 WS Driver" |

*Figure 28: 486 WS Driver service created.*

In Figure 29, we see that by running on the command line “sc query “486 WS Driver”” that the service is running and is a type of kernel driver. In the same Figure, we also see in regedit that the service is in the Current Control Set and has an image path to the .sys file.

To reiterate the facts: 1) Mlwx486.sys was created with a path to the System32 folder, but 2) Mlwx486.sys could not be found within the System32 folder, 3) 486 WS Driver was created as a service, 4) the service has an image path to Mlwx486.sys, and 5) 486 WS Driver is a service that is currently running. Therefore, it is highly probable that Mlwx486.sys resides in kernel memory, leading to the conclusion that Lab10-02.exe established a rootkit.



| | | |
|---|---|---|
| (Default) | REG_SZ | (value not set) |
| DisplayName | REG_SZ | 486 WS Driver |
| ErrorControl | REG_DWORD | 0x00000001 (1) |
| ImagePath | REG_EXPAND_SZ | \??\C:\Windows\System32\Mlwx486.sys |
| Start | REG_DWORD | 0x00000003 (3) |
| Type | REG_DWORD | 0x00000001 (1) |

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\John>sc query "486 WS Driver"

SERVICE_NAME: 486 WS Driver
        TYPE               : 1  KERNEL_DRIVER
        STATE              : 4  RUNNING
                                (STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

C:\Documents and Settings\John>_
```

*Figure 29: 486 WS Driver running with an image path the .sys file.*

Running the malware through Olly, we see if we can find the file being created by placing a breakpoint prior to the values being pushed onto the stack for the call to CreateFileA. This breakpoint was placed at 0x401028 (Figure 30). The documentation for CreateFileA gives us some information as to the properties of the file and if we need to modify them.

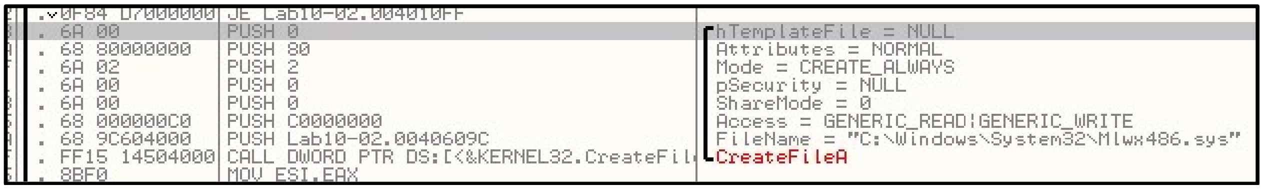
dwDesiredAccess: Generic read and write. This is fine.

dwShareMode: Set to 0. No other process can open the file. Modified to 1 so other processes can read.

lpSecurityAttributes: Set to 0. No modification needed.

dwCreationDisposition: Set to 2. It will always create a new file.

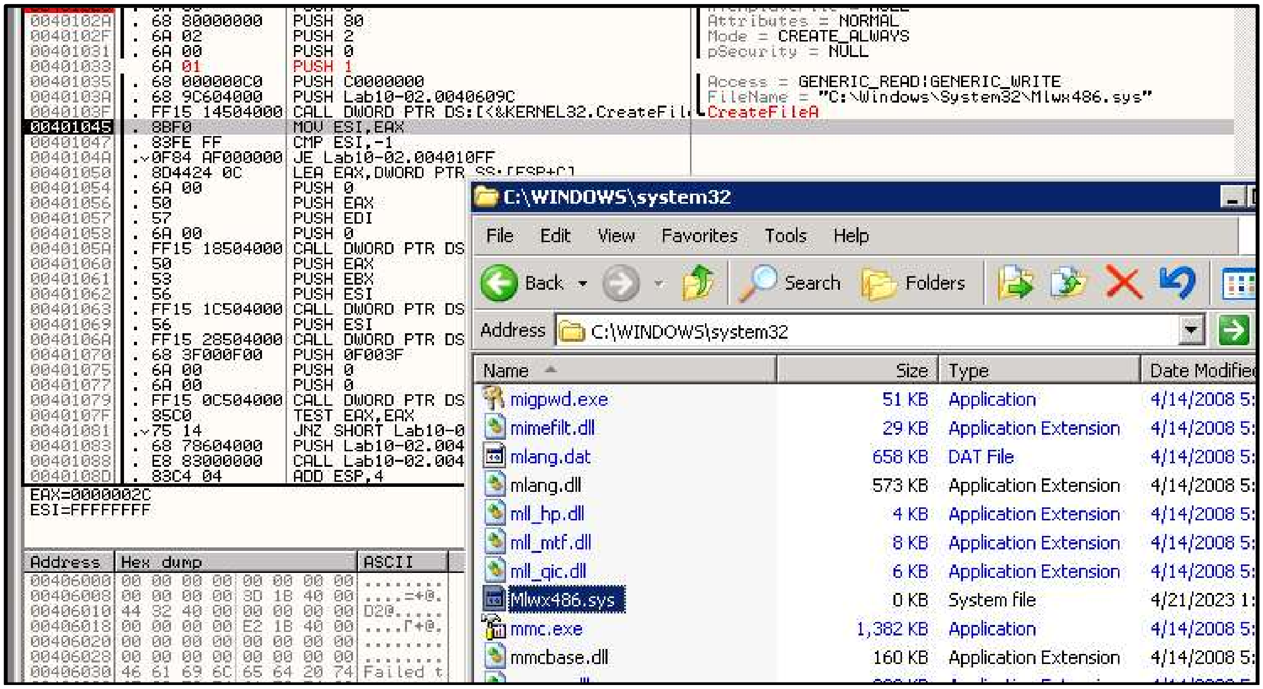
dwFlagsAndAttributes: Set to 0x80 (decimal 128). File attribute is normal.



```
.v0F84 D7000000 JE Lab10-02.004010FF
. 6A 00         PUSH 0                                  ┌hTemplateFile = NULL
. 68 80000000   PUSH 80                                 │Attributes = NORMAL
. 6A 02         PUSH 2                                  │Mode = CREATE_ALWAYS
. 6A 00         PUSH 0                                  │pSecurity = NULL
. 6A 00         PUSH 0                                  │ShareMode = 0
. 68 000000C0   PUSH C0000000                           │Access = GENERIC_READ|GENERIC_WRITE
. 68 9C604000   PUSH Lab10-02.0040609C                  │FileName = "C:\Windows\System32\Mlwx486.sys"
. FF15 14504000 CALL DWORD PTR DS:[<&KERNEL32.CreateFil └CreateFileA
. 8BF0          MOV ESI,EAX
```

*Figure 30: CreateFileA original parameters ShareMode was then set to 1.*

After CreateFileA was stepped over in Olly, the .sys file was confirmed to reside within the system32 folder (Figure 31). The code was then continuously stepped over until it ended. The malware was run again without the modification, yet the file still remained. Both instances stated in the command prompt it uses that it failed to create a service (Figure 32). This is probably due to some sort of functionality and/or checks within Mlwx486.sys that checks if the code is being debugged which causes it to not create the associated service. Since it doesn't create the service, it doesn't take steps to hide the driver file.



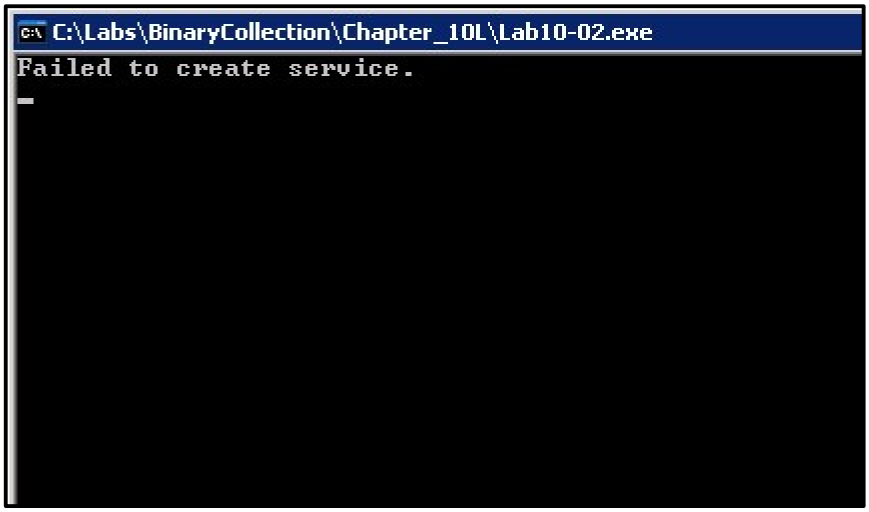
```
0040102A . 68 80000000   PUSH 80                              Attributes = NORMAL
0040102F . 6A 02         PUSH 2                               Mode = CREATE_ALWAYS
00401031 . 6A 00         PUSH 0                               pSecurity = NULL
00401033   6A 01         PUSH 1
00401035 . 68 000000C0   PUSH C0000000                        Access = GENERIC_READ|GENERIC_WRITE
0040103A . 68 9C604000   PUSH Lab10-02.0040609C               FileName = "C:\Windows\System32\Mlwx486.sys"
0040103F . FF15 14504000 CALL DWORD PTR DS:[<&KERNEL32.CreateFil CreateFileA
00401045 . 8BF0          MOV ESI,EAX
00401047 . 83FE FF       CMP ESI,-1
0040104A .v0F84 AF000000 JE Lab10-02.004010FF
00401050 . 8D4424 0C     LEA EAX,DWORD PTR SS:[ESP+C]
00401054 . 6A 00         PUSH 0
00401056 . 50            PUSH EAX
00401057 . 57            PUSH EDI
00401058 . 6A 00         PUSH 0
0040105A . FF15 18504000 CALL DWORD PTR DS
00401060 . 50            PUSH EAX
00401061 . 53            PUSH EBX
00401062 . 56            PUSH ESI
00401063 . FF15 1C504000 CALL DWORD PTR DS
00401069 . 56            PUSH ESI
0040106A . FF15 28504000 CALL DWORD PTR DS
00401070 . 68 3F000F00   PUSH 0F003F
00401075 . 6A 00         PUSH 0
00401077 . 6A 00         PUSH 0
00401079 . FF15 0C504000 CALL DWORD PTR DS
0040107F . 85C0          TEST EAX,EAX
00401081 .v75 14         JNZ SHORT Lab10-0
00401083 . 68 78604000   PUSH Lab10-02.004
00401088 . E8 83000000   CALL Lab10-02.004
0040108D . 83C4 04       ADD ESP,4
EAX=0000002C
ESI=FFFFFFFF

Address  Hex dump                ASCII
00406000 00 00 00 00 00 00 00 00 ........
00406008 00 00 00 00 3D 1B 40 00 ....=+@.
00406010 44 32 40 00 00 00 00 00 D2@.....
00406018 00 00 00 00 E2 1B 40 00 ....Γ+@.
00406020 00 00 00 00 00 00 00 00 ........
00406028 00 00 00 00 00 00 00 00 ........
00406030 46 61 69 6C 65 64 20 74 Failed t
```

C:\WINDOWS\system32

Address C:\WINDOWS\system32

| Name | Size | Type | Date Modified |
|---|---|---|---|
| migpwd.exe | 51 KB | Application | 4/14/2008 5: |
| mimefilt.dll | 29 KB | Application Extension | 4/14/2008 5: |
| mlang.dat | 658 KB | DAT File | 4/14/2008 5: |
| mlang.dll | 573 KB | Application Extension | 4/14/2008 5: |
| mll_hp.dll | 4 KB | Application Extension | 4/14/2008 5: |
| mll_mtf.dll | 8 KB | Application Extension | 4/14/2008 5: |
| mll_qic.dll | 6 KB | Application Extension | 4/14/2008 5: |
| Mlwx486.sys | 0 KB | System file | 4/21/2023 1: |
| mmc.exe | 1,382 KB | Application | 4/14/2008 5: |
| mmcbase.dll | 160 KB | Application Extension | 4/14/2008 5: |

*Figure 31: Mlwx486.sys exists in system32 folder.*



```
Failed to create service.
```

*Figure 32: Error message when debugging.*