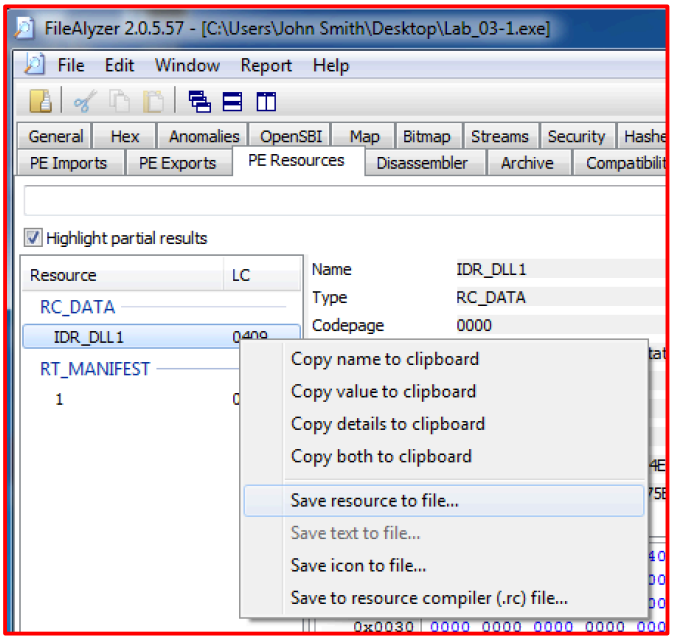
Lab 03-1.malware

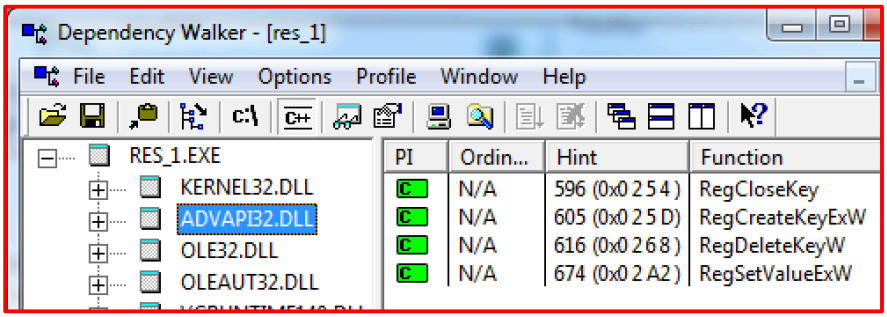
1. **Did you find any interesting resources? If so, how did you extract it?**

One of the resources in the resource section was a DLL. I extracted the nested DLL file using FileAlyzer2.



2. List at least 3 imports or sets of imports you haven't seen before, what their purpose is (from MSDN), and how the malware might use them.

I. *ADVAPI32.dll (from DLL found in resources)* This provides advanced access to core Windows components like the registry. The malware can use this API to change, add, or delete registry values.



II. *OLE32.dll (from DLL found in resources)* The OLE DLL is used for Object Linking and Embedding, which allows for objects created in one application to be embedded in other applications. The malware uses this library to initialize COM functionality, evident by the import of *CoInitialize* function.
III. *KERNEL32.dll* The portable executable imports functions like *CreateFileW* and *WriteFile* from the kernel DLL, indicating there may be some easy host-based signatures for this malware sample.

2. List at least 3 strings that stick out to you and describe how they might relate to malicious activity.

I. *http://blog.rpis.ec/* This could be a callback or C&C server used by the malware
II. *regsvr32 /s C:\Windows\atidrv.dll* This command is used to register a new DLL in the Windows registry. This could be the malicious DLL that was nested in the resources section found in the PE.
III. *SetUnhandledExceptionFilter* This function has the ability to supersede the top-level exception handler of each thread of a process. This could allow malware to take control of a process if it is able to raise an error.

3. What persistence mechanism is used by this malware? What host-based signatures can you gather from this?

Upon running, this malware creates a DLL at *C:\Windows\atidrv.dll*. It then runs the command found at 3.2 to register the DLL (using the /s flag to do so silently). The presence of this file is a strong host-based signature.

4. What is the CLSID served by this malware?

The CLSID is 3543619C-D563-43f7-95EA-4DA7E1CC396A. This was a Unicode string found in the malware sample in numerous locations.

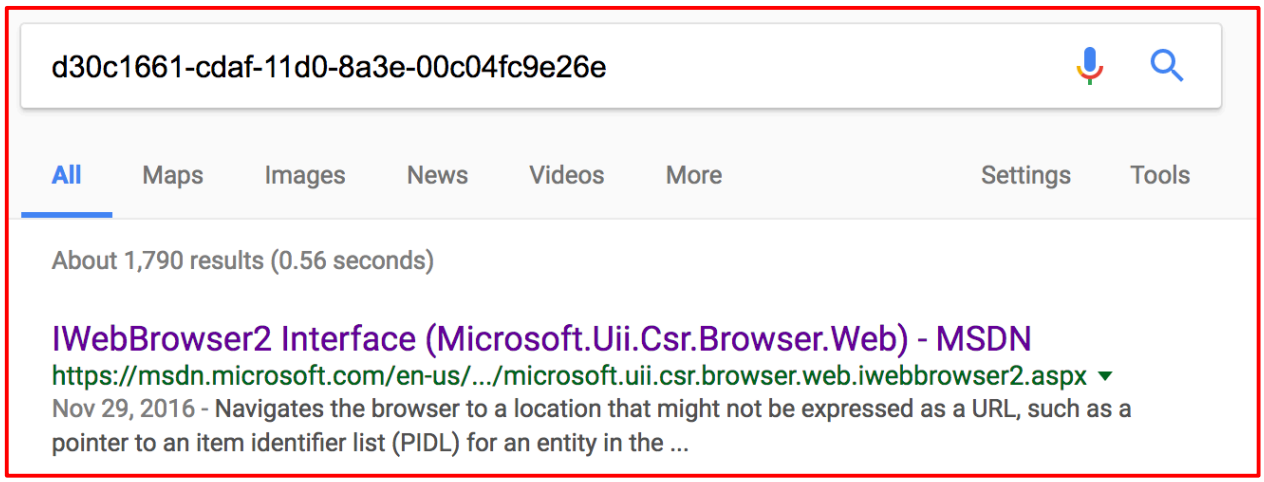
5. What is the name of the COM interface that this malware makes use of?

This malware uses the IWebBrowser2 COM interface. This is evident in the argument to the *CoCreateInstance* function at 0x10001B77 in the DLL.

```
10001B73
10001B73                        loc_10001B73:
10001B73 8D 55 D4               lea     edx, [ebp+ppv]
10001B76 52                     push    edx             ; ppv
10001B77 68 80 41 00 10         push    offset IID_IWebBrowser2 ; riid
10001B7C 6A 04                  push    4               ; dwClsContext
10001B7E 6A 00                  push    0               ; pUnkOuter
10001B80 68 A0 41 00 10         push    offset rclsid   ; rclsid
10001B85 FF 15 10 41 00 10      call    ds:CoCreateInstance
10001B8B 89 45 D0               mov     [ebp+var_30], eax
10001B8E 83 7D D0 00            cmp     [ebp+var_30], 0
10001B92 75 63                  jnz     short loc_10001BF7
```

```
.rdata:10004180 ; IID IID_IWebBrowser2
.rdata:10004180 IID_IWebBrowser2 dd 0D30C1661h          ; Data1
.rdata:10004180                                         ; DATA XREF: sub_10001AD0+A
.rdata:10004180                                         ; sub_100020B0+31↑o
.rdata:10004180                 dw 0CDAFh               ; Data2
.rdata:10004180                 dw 11D0h                ; Data3
.rdata:10004180                 db 8Ah, 3Eh, 0, 0C0h, 4Fh, 0C9h, 0E2h, 6Eh; Data4
```

In these screenshots, the IID is already renamed to IWebBrowser2. A simple google search for the IID as shown below reveals the interface being used.



6. What two COM functions does this malware call from the above COM interface and what are they used for? (hint: Check the PMA book)

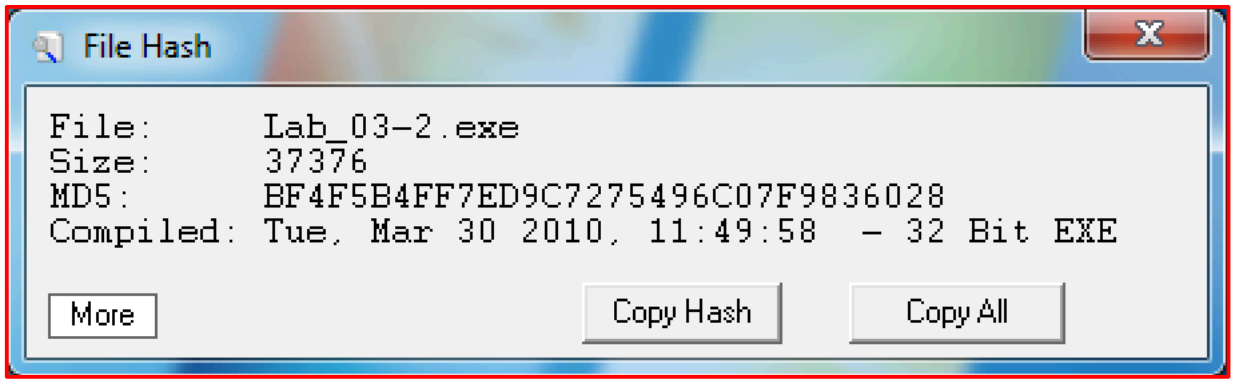
The two COM functions this malware calls are put_Visible and Navigate. This creates a visible Internet Explorer window, and navigates to a random URL listed higher up in the function.

```
10001BBB mov     ecx, [eax]
10001BBD mov     edx, [ebp+ppv]
10001BC0 push    edx
10001BC1 mov     eax, [ecx+IWebBrowser2Vtbl.put_Visible]
10001BC7 call    eax
10001BC9 push    0
10001BCB push    0
10001BCD push    0
10001BCF push    0
10001BD1 lea     ecx, [ebp+var_34]
10001BD4 call    sub_10001550
10001BD9 push    eax
10001BDA mov     ecx, [ebp+ppv]
10001BDD mov     edx, [ecx]
10001BDF mov     eax, [ebp+ppv]
10001BE2 push    eax
10001BE3 mov     ecx, [edx+IWebBrowser2Vtbl.Navigate]
10001BE6 call    ecx
10001BE8 mov     [ebp+var_4], 0FFFFFFFFh
10001BEF lea     ecx, [ebp+var_34]
```

```
10001AF8 mov     [ebp+var_38], ecx
10001AFB mov     [ebp+var_28], offset aHttpRpis_ec ; "http://rpis.ec/"
10001B02 mov     [ebp+var_24], offset aHttpRpis_ecBin ; "http://rpis.ec/binexp"
10001B09 mov     [ebp+var_20], offset aHttpsTwitter_c ; "https://twitter.com/RPISEC"
10001B10 mov     [ebp+var_1C], offset aHttpsWww_faceb ; "https://www.facebook.com/RPI-Computer-S"...
10001B17 mov     [ebp+var_18], offset aHttpBlog_rpis_ ; "http://blog.rpis.ec/"
10001B1E mov     [ebp+var_14], offset aHttpSecurity_c ; "http://security.cs.rpi.edu/courses/bine"...
```

Lab 03-2.malware

1. What is the md5sum? What of interest does VirusTotal report?



VirusTotal reports this file as a Trojan. Some of the names reported are *Troj.W32.Generic!c, Trojan.DarkHotel.1, Win32:Trojan-gen*.

2. List at least 3 imports or sets of imports you haven't seen before, what their purpose is (from MSDN), and how the malware might use them.

I. *PeekNamedPipe* This function allows the caller to view the data in a pipe without altering the pipe. This could allow malware to spy on other processes' network connections, open file streams, and user input/output.
II. *GetDriveType* This function allows a program to determine the types of logical drives connected to the computer (fixed, removable storage, network file share, etc.) Malware can use this function to determine what drives to act on, i.e. a worm trying to spread over a network file share.
III. *Sleep* The sleep function allows a thread to be suspended for a specified amount of time. Malware can use this function if it is waiting on the user to perform a certain action, and it does not want to bring attention to itself by using computing resources.

3. List at least 3 strings that stick out to you and describe how they might relate to malicious activity.

I. *SOFTWARE\Microsoft\Windows\CurrentVersion\Run* This is the registry key that is used for autostarting programs at login. It can be used as a form of persistence for malware.

II. *cmd.exe* The command prompt executable launches a shell which could be used remotely by an attacker.

III. *%4d-%02d-%02d %02d:%02d:%02d* This seems to be a timestamp format string from the C/C++ malware code, indicating that timestamps and/or logging could be an important feature of the malware.

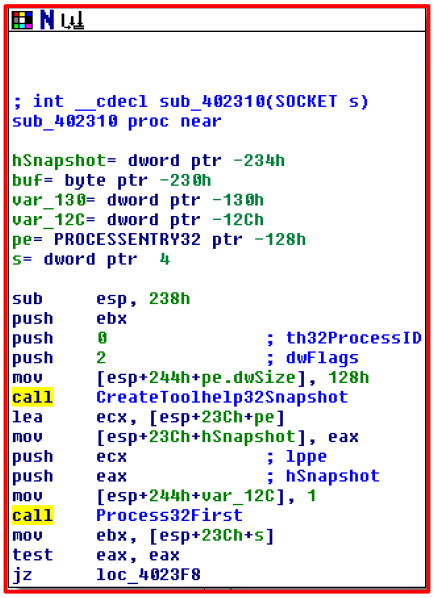
4. What persistence mechanism is used by this malware? What host-based signatures can you gather from this?

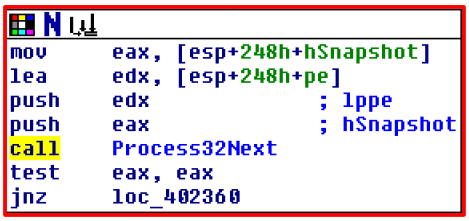
The malware adds itself to the *HKEY_CURRENT_USER\Software\Microsoft\Windows \CurrentVersion\Run* registry value, which contains a list of all the programs that launch when a user logs in. The malware's hash is the easiest and most indicative host-based signature. Any applications in this registry value that share a hash (program names can be easily changed) with the malware are malicious.

Malware Functionality: Listing processes

5. **What is the address of the subroutine that handles this functionality?**

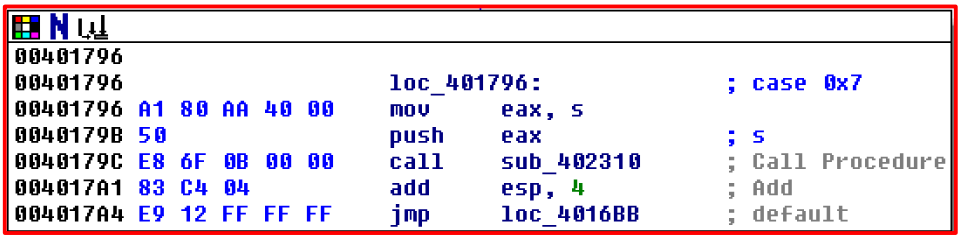
sub_402310 (address 0x402310) is the subroutine that handles this functionality. It does so through using a while loop, and the *ProcessFirst* and *ProcessNext* functions.





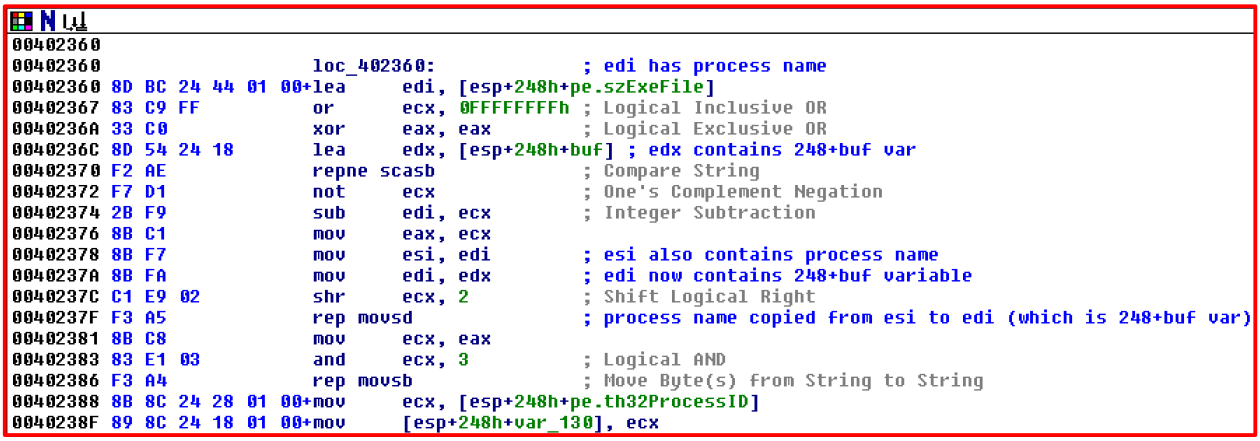
6. What is the command ID?

The command ID for this function is 0x7.



7. Does the subroutine return anything to the attacker, if so, what?

The subroutine returns the process ID and name back to the server. The name is XOR'd with 0x55 (85 in base 10) as shown below, presumably to mitigation detection when inspecting web traffic.



8. Name 3 Windows API calls (besides send/recv) used and how they might contribute to the functionality.

This subroutine calls 3 Windows API functions besides *send/recv*. These are *CreateToolhelp32Snapshot*, *ProcessFirst*, and *ProcessNext*. The first function creates a

snapshot of all running processes on the system, and the *ProcessFirst* and *Next* functions iterate through the processes, fetching information to be sent back to the attacker.

Malware Functionality: Interactive remote shell

5. What is the address of the subroutine that handles this functionality?

The functionality is split between two subroutines, 0x402490 and 0x4017D7.

6. What is the command ID?

The command IDs are 0x9 and 0xA (10).

7. Does the subroutine return anything to the attacker, if so, what?

The 0x9 subroutine spawns a new process (shell) and returns a Pipe handle for 0xA to use. 0xA receives a command, XOR's it with 0x55 to decrypt it, runs the command, and pipes the output back to the attacker (XOR'd again with 0x55).

8. Name 3 Windows API calls (besides send/recv) used and how they might contribute to the functionality.

I. *CreatePipe* Creates a named pipe, and returns a handle to the read and write end. This is used in the future to input commands to the remote shell and to read the command's output.
II. *PeekNamedPipe* Reads the content of a pipe without modifying its contents. This is used by the subroutine to read the output of a remote shell command.
III. *WriteFile* This function is used by the subroutine to write to the pipe created in 0x402490.

Malware Functionality: Upload file

5. What is the address of the subroutine that handles this functionality?

The address of the subroutine that handles the file uploading functionality is 0x402210.

6. What is the command ID?

The command ID is 0x6.

7. Does the subroutine return anything to the attacker, if so, what?

The subroutine returns a copy of the specified file to the attacker. Before sending the file back to the C&C server, the file's data is XOR'd with 0x55.

8. Name 3 Windows API calls (besides send/recv) used and how they might contribute to the functionality.

The functions *CreateFile*, *CreateFileMapping*, and *MapViewOfFile* are used to open a specified file with read permissions, and map the file into the process's memory for XOR-ing and exfiltration.

9. Did the networking guys miss anything? Briefly name/describe 3 more functionalities offered by the malware. Provide the command IDs.

I. *Command ID 1* Returns a list of all logical drives and their type (fixed/removable/network storage/etc.).
II. *Command ID 2* Returns information on every file in the system using a combination of the *FindFirstFile*, *FindNextFile*, and *SHGetFileInfoA* functions.
III. *Command ID 4* Deletes a specified file.

The status and/or output of the above commands are all returned to the attacker using the *send* socket function.