Detailed threat analysis of Shamoon 2.0 Malware

Gregory Paul and Shaunak

Our Previous post talked about the initial overview of the Shamoon 2.0 sample. This analysis is a continuation of our last post but with a more insight on the working and behavior of the malware.

There are 3 components which are linked with one another which makeup Shamoon 2.0 single malware. We have analyzed each component according to the stages which the Shamoon 2.0 uses for infection on a victim's machine i.e. Dropper Component⇒ Communication Component⇒ Wiper Component.

When Shamoon 1.0 made its first wave of attack in August 2012, it had not just infected 30,000-35,000 computers but it also had crippled the entire organizations altogether which were infected with it. Its effects were seen post attack as many computers were still working irregularly and the time that required to restore the organization's full functionality led to huge loss in not just terms of money but also in terms of company's reputation too.

The second wave Shamoon which is dubbed as Shamoon 2.0 used the similar approach which it had used previously but this time it is predicted that the amount of infection of computers will be more, since last time the attackers were able to retrieve the credentials of users for various organization, The second wave will be using the stolen credentials from the previous attack and the reason this attack is bound to be success is because of lack of awareness among the employees on securing passwords. One survey about the Middle East reports some of the facts mentioned below:

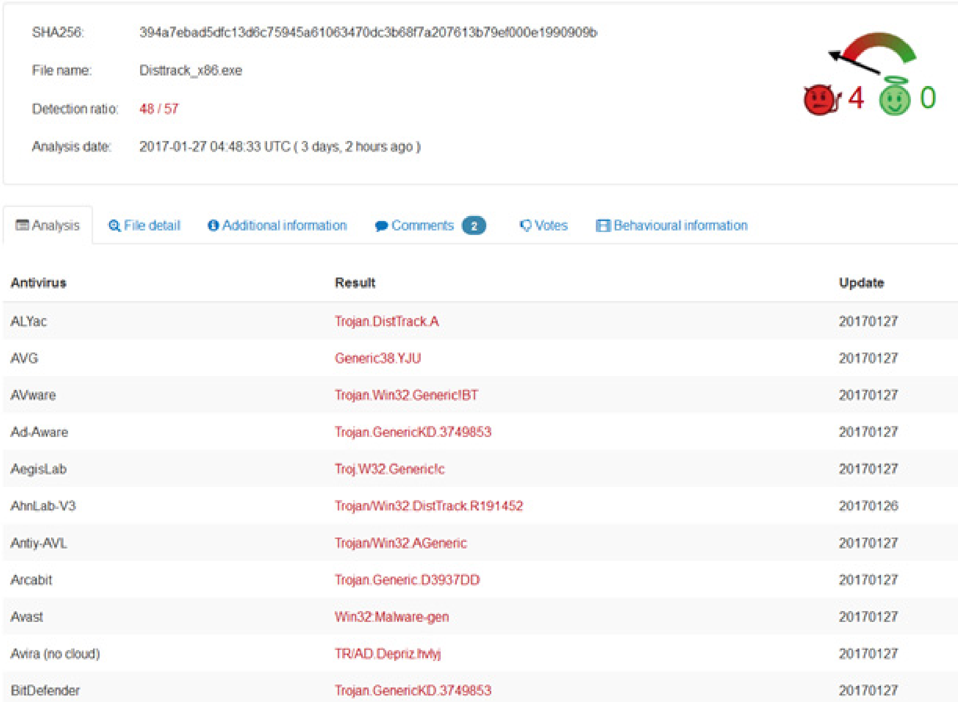
- **More than 70 percent** of the users said that they were storing administrative passwords in plaintext.
- **Over 45 percent** of the users use the same password for over multiple systems.
- **More than 40 percent** users share their passwords.
- **Only 13 percent** users change their passwords once a month.

These facts make the Middle East region more easy as a target for Shamoon 2.0. We have launched a Shamoon detection tool which can detect the new Shamoon 2.0.

Following below is the in-depth analysis that we have done on Shamoon 2.0.

Dropper Component - Disttrack:
Upon computing the hash value of the sample, the SHA256 as
394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b

Doing a quick VirusTotal search we get the following output:



SHA256:	394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b
File name:	Disttrack_x86.exe
Detection ratio:	48 / 57
Analysis date:	2017-01-27 04:48:33 UTC (3 days, 2 hours ago)

Analysis | File detail | Additional information | Comments 2 | Votes | Behavioural information

Antivirus	Result	Update
ALYac	Trojan.DistTrack.A	20170127
AVG	Generic38.YJU	20170127
AVware	Trojan.Win32.Generic!BT	20170127
Ad-Aware	Trojan.GenericKD.3749853	20170127
AegisLab	Troj.W32.Generic!c	20170127
AhnLab-V3	Trojan/Win32.DistTrack.R191452	20170126
Antiy-AVL	Trojan/Win32.AGeneric	20170127
Arcabit	Trojan.Generic.D3937DD	20170127
Avast	Win32:Malware-gen	20170127
Avira (no cloud)	TR/AD.Depriz.hvlyj	20170127
BitDefender	Trojan.GenericKD.3749853	20170127

This assures us that the sample we are analyzing is of Shamoon 2.0. The date of update also tells us that it is the recent Shamoon sample which is dubbed as the Shamoon 2.0.

The sample uses the following evasion techniques for Debugging:

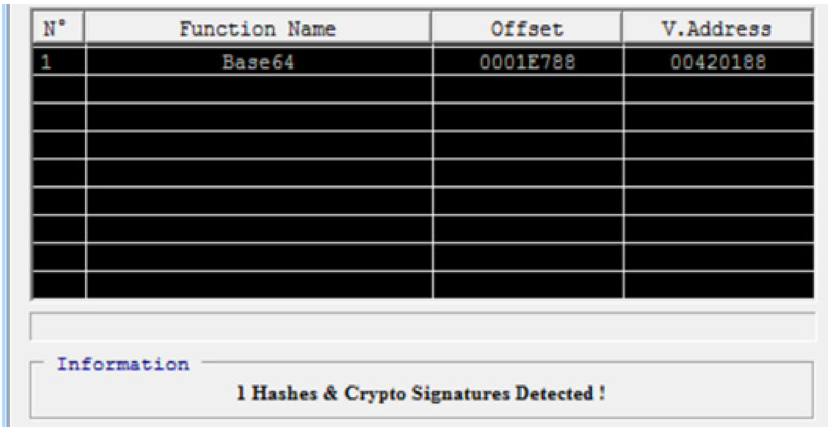
1) GetLastError
2) IsDebuggerPresent
3) Process32FirstW
4) Process32NextW
5) TerminateProcess
6) UnhandledExceptionFilter

The following screenshot gives information of the which compiler was used for compiling the malware, which entry point address is being used, EP section tells us the entry point of the portable executable (PE).

As mentioned earlier above the compiler used is **Microsoft Visual C++ v8.0 2005**

Malware in general use some basic techniques to obfuscate the code so that it is not easily readable when loaded in any debugger and to make it more difficult to reverse the malware. There are many Hashing methods that can be used. Our sample uses the Hash technique known as

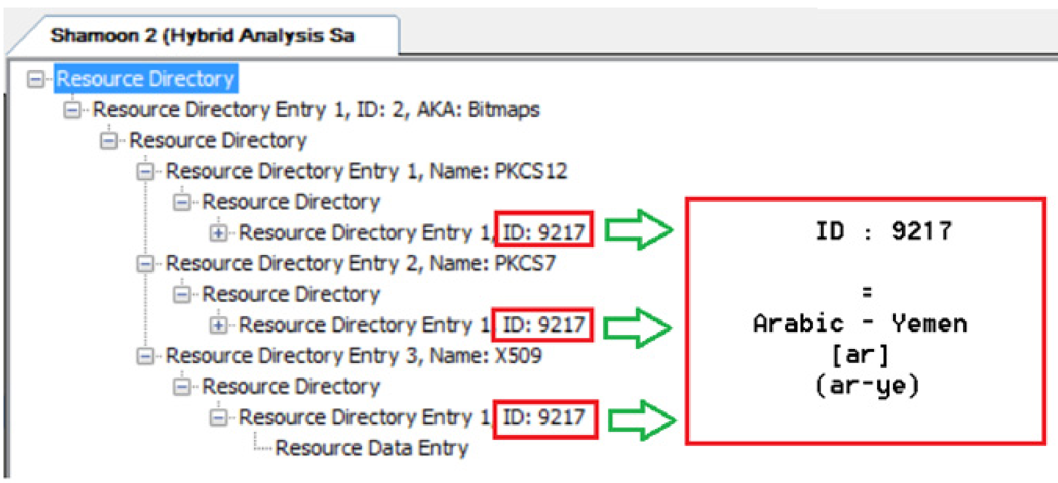
Base64.



We know that Shamoon 2.0 was targeted the Middle East region. The following screenshot is the evidence that this malware is specifically looking for **Arabic -Yemen [ar] (ar-ye)** language settings.

So the malware looks into the keyboard layout and the ID mentioned is in the reference of the keyboard layout, for example ID:1033 corresponds to the English-US [en] (en-us), here we find that the language is of the

ID: 9217 i.e.Arabic -Yemen [ar] (ar-ye).



The following file operations that took place during the execution of the malware are listed as following:

1. **File-Read**
C:\Documents and Settings\student\LocalSettings\Temp\Shamoon-394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b.bin

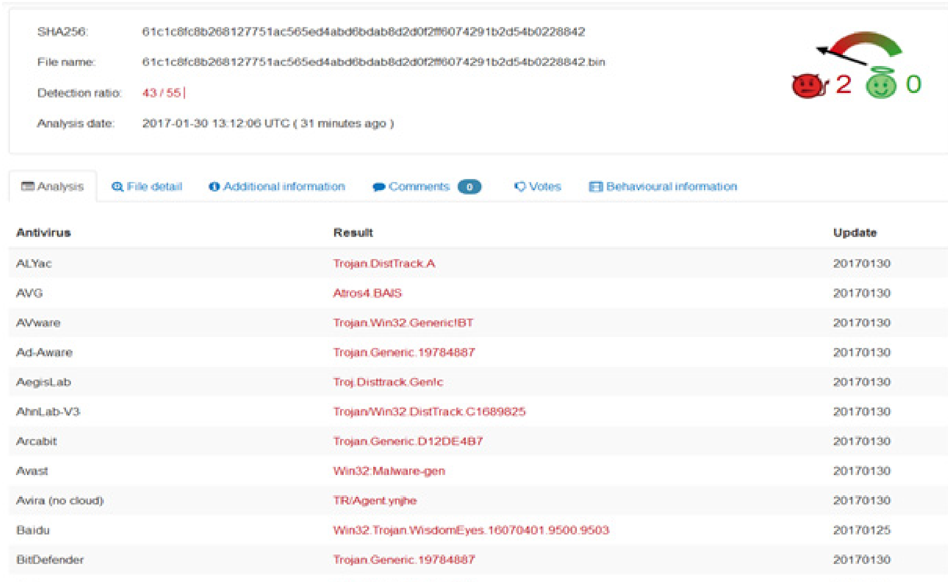
2. **File-Opened**
C:\Documents and Settings\student\LocalSettings\Temp\Shamoon-394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b.bin

C:\WINDOWS\system32\kernel32.dll

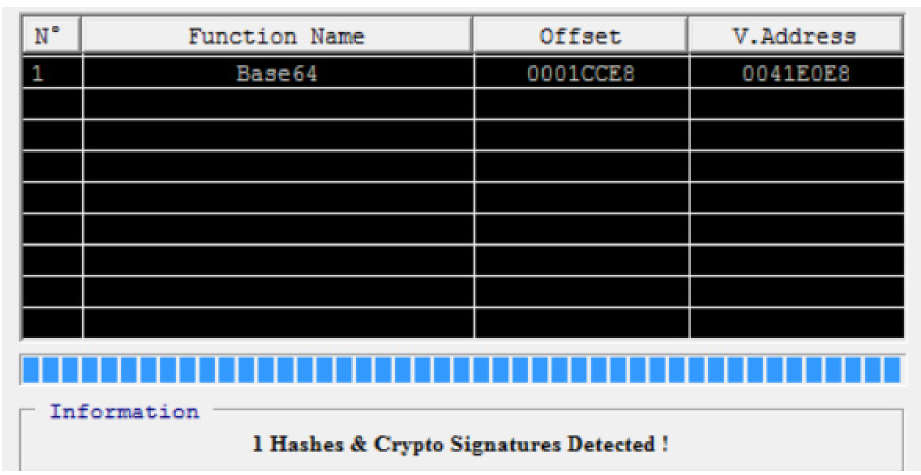
3. **Registry Key-Read**
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\GRE_Initialize\DisableMetaFiles

Communication Component - Disttrack:

Upon computing the hash value of the sample, the SHA256 as **61c1c8fc8b268127751ac565ed4abd6bdab8d2d0f2ff6074291b2d54b0228842**, doing a quick VirusTotal search we verified the sample as a part of the Shamoon 2.0



Following screenshot shows that communication component has the same hash technique as that seen in the dropper component mentioned earlier, i.e. **Base64.**



Since communication component is a part of the Shamoon 2.0 components it will have same compiler used

For compiling the communication component as well which is shown in the screenshot below:

During our analysis, we found that the communication component made many changes in the Registry values of the infected system, these changes are mentioned below:

Registry Key - Opened

1) HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\DnsCache\Parameters
2) HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows NT\DnsClient

3) HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows NT\Rpc
4)HKEY_LOCAL_MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\ImageFile ExecutionOptions\61c1c8fc8b268127751ac565ed4abd6bdab8d2d0f2ff6074291b2d54b0228842.exe\RpcThreadPoolThrottle
5) HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LDAP
6) HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\System\DNSClient
7) HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc
8) HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
9) HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\PagedBuffers
10) HKEY_LOCAL_MACHINE\System\Setup

Registry Key - Read

1. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\UseDomainNameDevolution
2. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\ServerPriorityTimeLimit
3. HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Rpc\MaxRpcSize
4. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\WaitForNameErrorOnAll
5. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DnsQuickQueryTimeouts
6. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DefaultRegistrationRefreshInterval
7. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\RegisterWanAdapters
8. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\DomainNameDevolutionLevel
9. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\AppendToMultiLabelName
10. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DisableAdapterDomainName
11. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\RegisterPrimaryName
12. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\EnableAdapterDomainNameRegistration
13. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\UpdateTopLevelDomainZones
14. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\FilterClusterIp
15. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\DnsTest
16. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\ScreenUnreachableServers
17. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\MulticastListenLevel
18. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\MaxNegativeCacheTtl
19. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\QueryAdapterName
20. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\PrioritizeRecordData
21. HKEY_LOCAL_MACHINE\SYSTEM\Setup\SystemSetupInProgress
22. HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\GRE_Initialize\DisableMetaFiles
23. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DisableReverseAddressRegistrations
24. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\MaxCacheTtl
25. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\UpdateSecurityLevel
26. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\MaxCachedSockets
27. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\RegistrationEnabled
28. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\RegisterAdapterName
29. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\AdapterTimeoutLimit
30. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\UpdateSecurityLevel
31. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\RegistrationMaxAddressCount
32. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DefaultRegistrationTTL
33. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DisableDynamicUpdate
34. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\Hostname
35. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\AllowUnqualifiedQuery
36. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\UpdateZoneExcludeFile
37. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\PrioritizeRecordData
38. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\RegistrationTtl
39. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\UseHostsFile
40. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\AllowUnqualifiedQuery

41. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\RegistrationRefreshInterval
42. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DnsQueryTimeouts
43. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\QueryIpMatching
44. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DnsNbtLookupOrder
45. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\MaxCacheSize
46. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\UseDomainNameDevolution
47. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\UseEdns
48. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\Domain
49. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\ldap\LdapClientIntegrity
50. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DisableWanDynamicUpdate
51. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DnsMulticastQueryTimeouts
52. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\ScreenBadTlds
53. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\RegisterReverseLookup
54. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\MaxNumberOfAddressesToRegister
55. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Dnscache\Parameters\MulticastSendLevel
56. HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters\DomainNameDevolutionLevel

These above results only indicate that the malware sample tries to communicate with the server.

Wiper Component - Disttrack:

The wiper component is the most important component out of the three components of Shamoon 2.0. Upon computing the hash value of the sample, the SHA256 as **128fa5815c6fee68463b18051c1a1ccdf28c599ce321691686b1efa4838a2acd**.

A quick look up with VirusTotal confirms that this indeed is a wiper component of the Shamoon 2.0.

SHA256: 128fa5815c6fee68463b18051c1a1ccdf28c599ce321691686b1efa4838a2acd
File name: 2cd0a5f1e9bcce6807e57ec8477d222a.virus
Detection ratio: 45 / 56
Analysis date: 2017-01-18 06:32:38 UTC (1 week, 5 days ago)

Analysis | File detail | Additional information | Comments 0 | Votes | Behavioural information

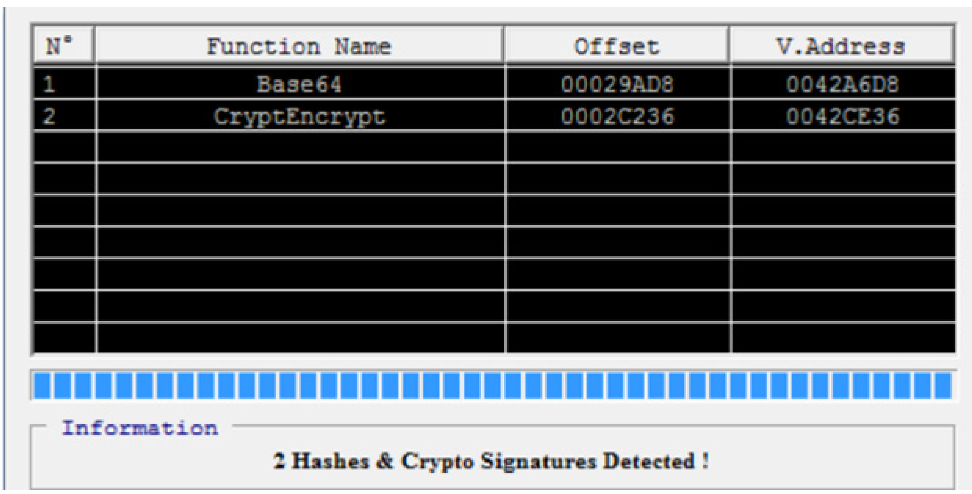
Antivirus	Result	Update
ALYac	Trojan.DistTrack.A	20170118
AVG	Generic38.YOB	20170118
AVware	Trojan.Win32.Generic!BT	20170118
Ad-Aware	Trojan.Generic.19780901	20170118
AegisLab	Troj.W32.Generic!c	20170118
AhnLab-V3	Trojan/Win32.DistTrack.C1689828	20170117
Antiy-AVL	Trojan/Win32.AGeneric	20170118
Arcabit	Trojan.Generic.D12DD525	20170118
Avast	Win32:Malware-gen	20170118
Avira (no cloud)	TR/Agent.axwlp	20170117
BitDefender	Trojan.Generic.19780901	20170118

Initial analysis shows us that apart from using the anti-debugging techniques this component also uses Anti-VM tricks which was not seen pervious dropper sample and communication sample.

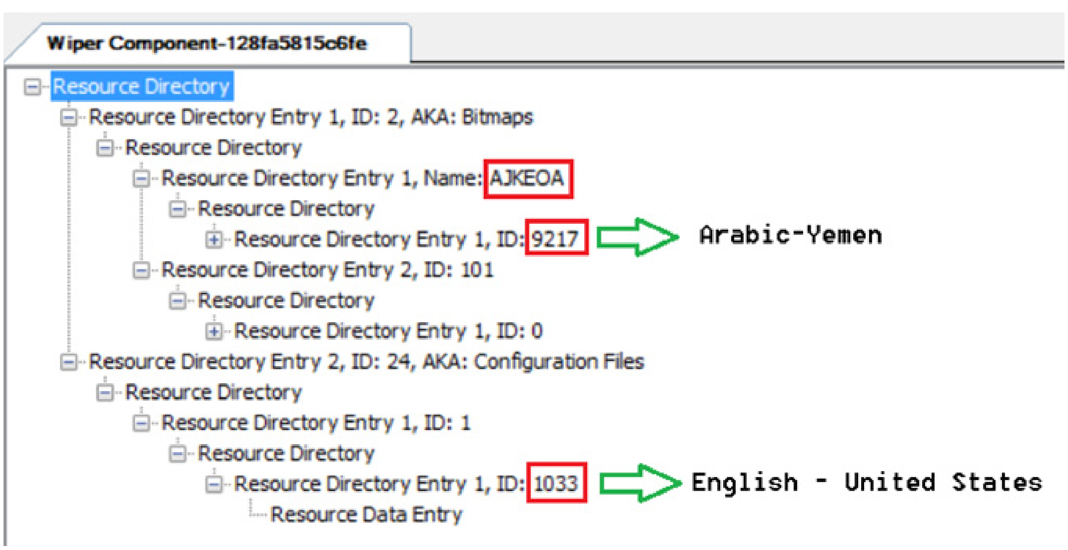
VMCheck.dll is a technique used to check if the sample is in a Virtual machine or not.

Just similar to the Dropper component and Communication component, we find that the Wiper component uses **Microsoft Visual C++ v8.0 2005** shown in the screenshot below.

However, what is different in the Wiper component, which is not present in the dropper or the communication component is it uses and additional hash/crypt function along with the **Base64**. i.e. **CryptEncrypt** Function is also used. The screenshot below shows this, which only means that the malware developers really wanted the make this wiper component not more difficult to understand for researchers but also much more obfuscated than the other components that we discussed above, as obfuscated codes are not detected by Antivirus companies easily.



The language component remains same as that of the dropper with the default English option included as shown below:



In context of the registry changes that the Wiper does is same as it did with the Dropper component:

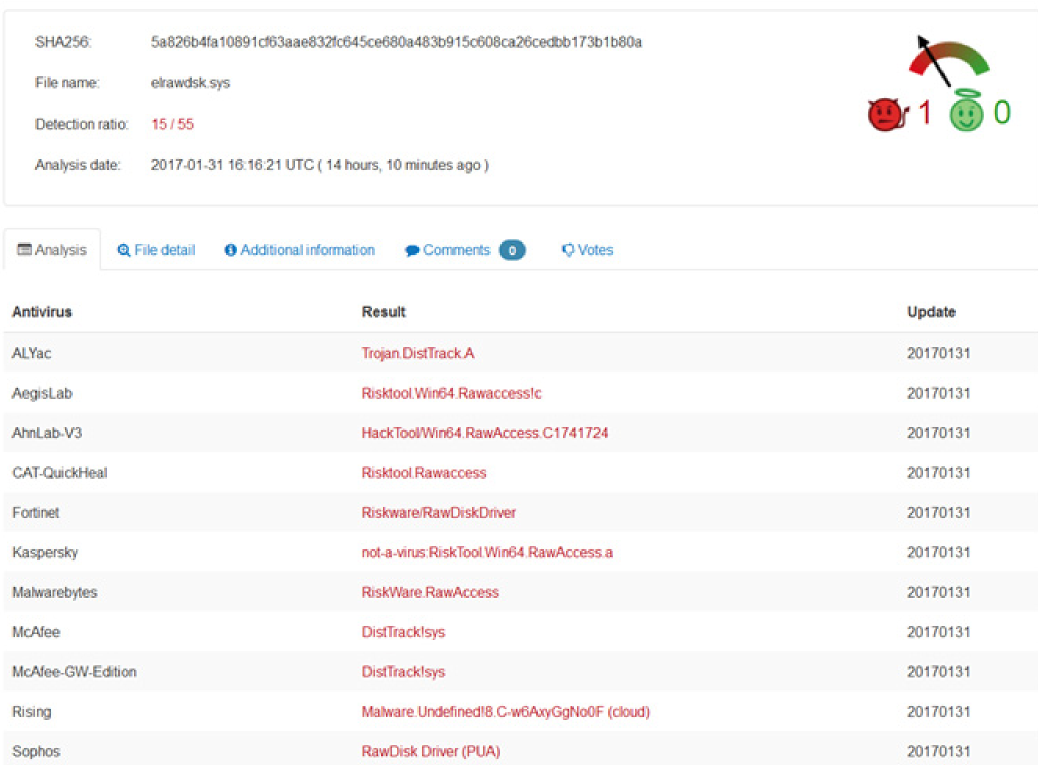
Registry Key-Read

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\GRE_Initialize\DisableMetaFiles

The cryptbase.pdb which contains all the necessary information about the encryption techniques, Keys on encryption and other functions are linked in the following way with the code of the wiper sample:

One more file that we analyzed was which had links to the Shamoon 2.0 was with the SHA256 hash as **5a826b4fa10891cf63aae832fc645ce680a483b915c608ca26cedbb173b1b80a**.

Doing a VirusTotal lookup gives us the following confirmation that the software is malicious in nature and that the detection ratio is very low as well.



Antivirus	Result	Update
ALYac	Trojan.DistTrack.A	20170131
AegisLab	Risktool.Win64.Rawaccess!c	20170131
AhnLab-V3	HackTool/Win64.RawAccess.C1741724	20170131
CAT-QuickHeal	Risktool.Rawaccess	20170131
Fortinet	Riskware/RawDiskDriver	20170131
Kaspersky	not-a-virus:RiskTool.Win64.RawAccess.a	20170131
Malwarebytes	RiskWare.RawAccess	20170131
McAfee	DistTrack!sys	20170131
McAfee-GW-Edition	DistTrack!sys	20170131
Rising	Malware.Undefined!8.C-w6AxyGgNo0F (cloud)	20170131
Sophos	RawDisk Driver (PUA)	20170131

Now the preliminary analysis shows us the following files that were found:

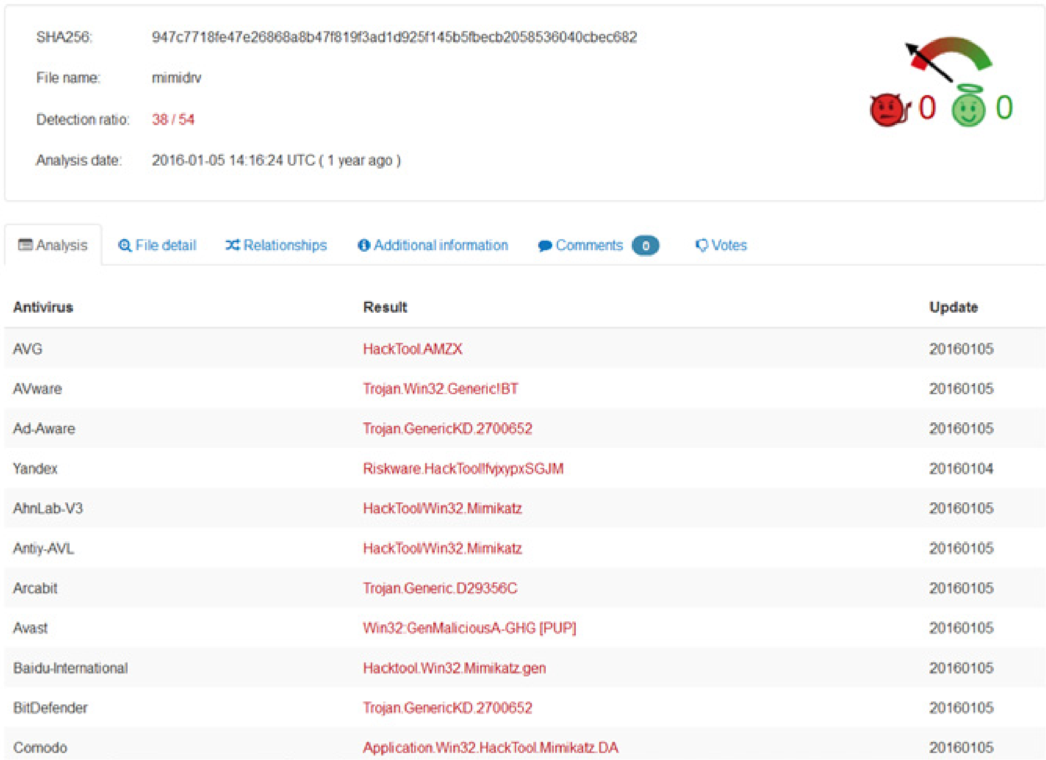
Executable	ntoskrnl.exe
Database	c:\projects\rawdisk\bin\wnet\fre\amd64\elrawdsk.pdb

During the analysis for the file we found the following device name parameters

\#{9A6DB7D2-FECF-41ff-9A92-6EDA696613DF}#
\#{8A6DB7D2-FECF-41ff-9A92-6EDA696613DE}#

The interesting thing is that, this same details were also found in the previous Shamoon attack that took place in 2012.

We also came across these ‘**060523170051Z**’ and ‘**160523171051Z0W1**’ strings. The interesting thing about these numbers is that they are found in a different malware which has a file name ‘mimidrv.sys’. The screenshot of that malware is mentioned below.

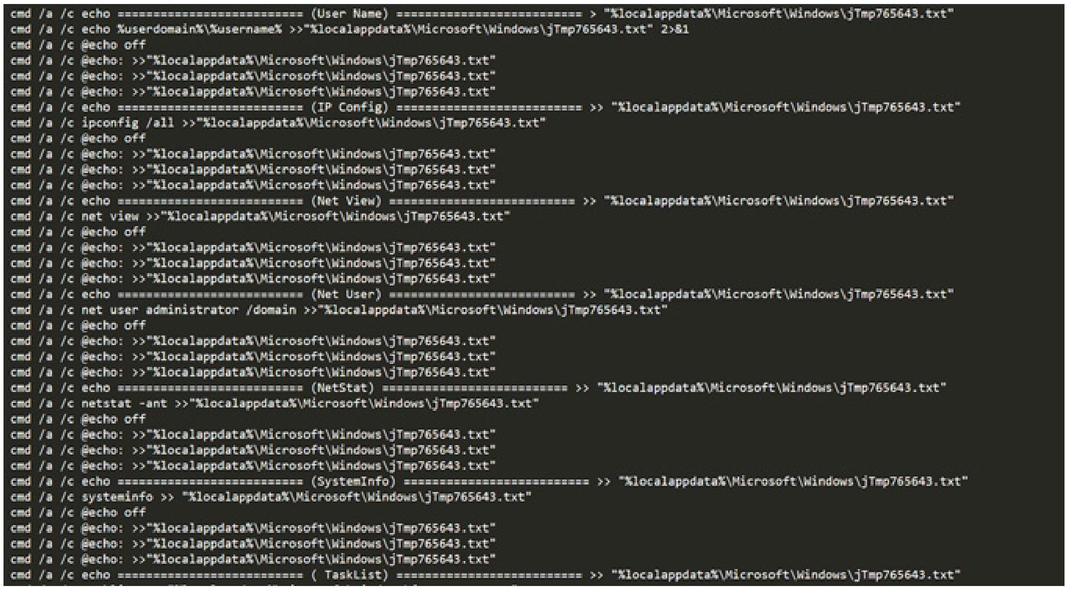


SHA256: 947c7718fe47e26868a8b47f819f3ad1d925f145b5fbecb2058536040cbec682
File name: mimidrv
Detection ratio: 38 / 54
Analysis date: 2016-01-05 14:16:24 UTC (1 year ago)

Antivirus	Result	Update
AVG	HackTool.AMZX	20160105
AVware	Trojan.Win32.Generic!BT	20160105
Ad-Aware	Trojan.GenericKD.2700652	20160105
Yandex	Riskware.HackTool!fvjxypxSGJM	20160104
AhnLab-V3	HackTool/Win32.Mimikatz	20160105
Antiy-AVL	HackTool/Win32.Mimikatz	20160105
Arcabit	Trojan.Generic.D29356C	20160105
Avast	Win32:GenMaliciousA-GHG [PUP]	20160105
Baidu-International	Hacktool.Win32.Mimikatz.gen	20160105
BitDefender	Trojan.GenericKD.2700652	20160105
Comodo	Application.Win32.HackTool.Mimikatz.DA	20160105

This malware mentioned above is basically a ‘hacktool’ Trojan as identified by the other Antivirus companies. There are chances that this is another behavior that our sample also behaves like the sample mentioned below.

We find that the **Mimikatz malware is related with the** PowerShell. We had found out **the same PowerShell which we had reported** in our previous blog. Hence the Shamoon 2.0 has some behaviour with PowerShell. Following screenshot shows the PowerShell commands that Shamoon 2.0 executes:

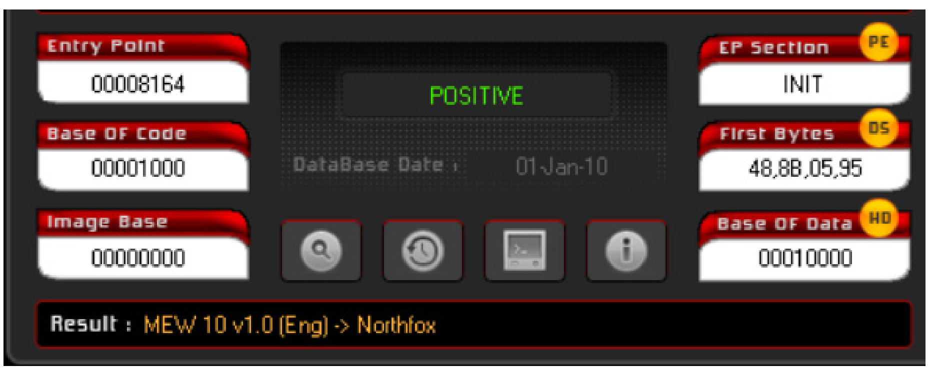


```
cmd /a /c echo ========================= (User Name) ========================= > "%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c echo %userdomain%\%username% >>"%localappdata%\Microsoft\Windows\jTmp765643.txt" 2>&1
cmd /a /c @echo off
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c echo ========================= (IP Config) ========================= >> "%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c ipconfig /all >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo off
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c echo ========================= (Net View) ========================= >> "%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c net view >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo off
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c echo ========================= (Net User) ========================= >> "%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c net user administrator /domain >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo off
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c echo ========================= (NetStat) ========================= >> "%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c netstat -ant >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo off
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c echo ========================= (SystemInfo) ========================= >> "%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c systeminfo >> "%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo off
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c @echo: >>"%localappdata%\Microsoft\Windows\jTmp765643.txt"
cmd /a /c echo ========================= ( TaskList) ========================= >> "%localappdata%\Microsoft\Windows\jTmp765643.txt"
```

From the evidence collected we confirm links between the Shamoon 1.0 to Shamoon 2.0.

Some features that were observed with this sample are that it is using an overlay to hid the packer information:

Analysing further, we found out the MEW 10v1.0 from Northfox packer is used:



Unlike the components that we analyzed so far this particular sample had the CRC16 Hash function which is completely different.

N°	Function Name	Offset	V.Address
1	CRC16	00001512	42001552

Information

1 Hashes & Crypto Signatures Detected !

The malware has a SSL certificate embedded within it. The following screenshot gives the SLL certificate information,

Public key	RSA (1024 Bits)
Authority Key Identifier	KeyID=d2 5b f3 4b 26 4b a5 b0 e7 5d
Authority Information A...	

```
30 81 89 02 81 81 00 e4 06 0a 27 cb b7 0f 30 9a 41
cf fb 9e d7 87 c9 59 a3 91 68 1e c6 98 20 ad 82 07
b0 54 74 d5 46 27 8a 21 2b c5 de f1 95 85 f2 42 f6
60 8b 7b 02 b0 75 b5 ce fe 6b aa 6e 2e b9 07 bd 6f
a8 36 8f 86 12 55 06 fb 5d c3 e1 4a aa 61 36 dd 6b
1c 24 e5 f3 b6 4d 2c e8 ac 55 10 62 ef 09 0f ad 86
46 98 ce 01 b9 00 3f e2 4f e5 4c 1a d5 5b c8 72 ea
66 32 25 f6 68 28 bd c3 18 53 7f 98 1d 0f 55 8b 02
03 01 00 01
```

The certificate is valid from Monday,January 11, 2010 7:49.26 PM to Friday, January 11 2013 7:49:26 PM

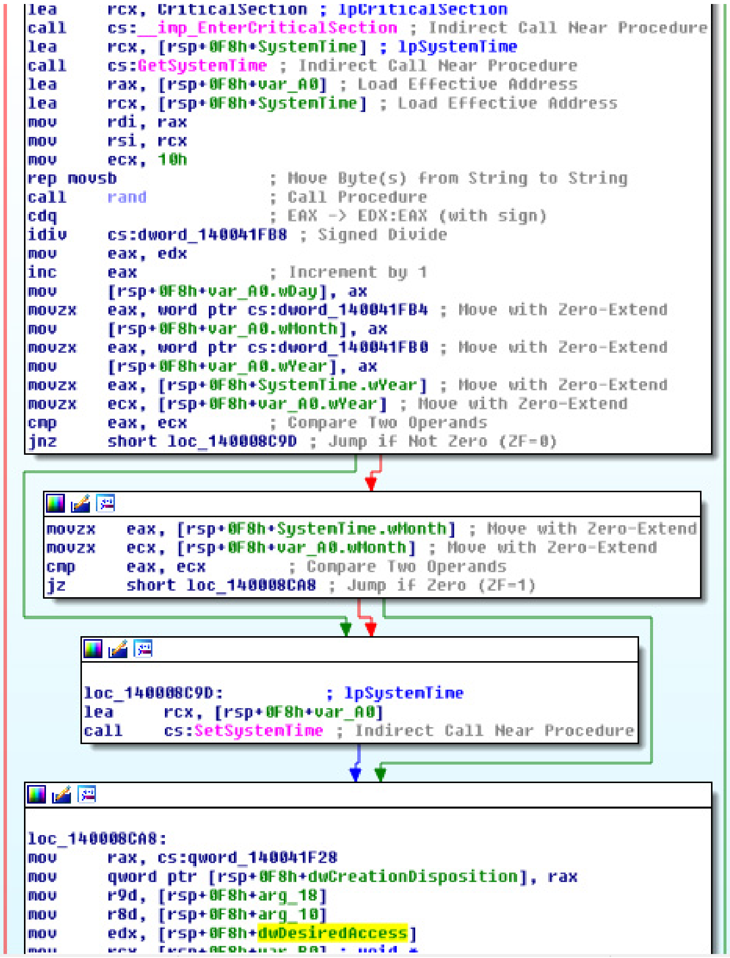
This above date correlates to the hard-coded date inside the program. This hardcoded date allows the program to execute since the date is inside the validity period as mentioned.

```
.rdata:00000000000121F0 ; Debug Directory entries
.rdata:00000000000121F0                 dd 0                    ; Characteristics
.rdata:00000000000121F4                 dd 4EFB4911h            ; TimeDateStamp: Wed Dec 28 16:51:29 2011
.rdata:00000000000121F8                 dw 0                    ; MajorVersion
.rdata:00000000000121FA                 dw 0                    ; MinorVersion
.rdata:00000000000121FC                 dd 2                    ; Type: IMAGE_DEBUG_TYPE_CODEVIEW
.rdata:0000000000012200                 dd 4Ch                  ; SizeOfData
.rdata:0000000000012204                 dd rva asc_122E0        ; AddressOfRawData
.rdata:0000000000012208                 dd 10E0h                ; PointerToRawData
.rdata:000000000001220C                 align 10h
```

The pseudo code explains that Shamoon 2.0 changes the system time, and sets it at random time and date between Monday, January 11, 2010 7:49.26 PM to Friday, January 11 2013 7:49:26 PM.

```
if ( a1 )
{
  EnterCriticalSection(&CriticalSection);
  GetSystemTime(&SystemTime);
  v5 = SystemTime;
  v5.wDay = rand() % dword_438204 + 1;
  v5.wMonth = dword_438200;
  v5.wYear = dword_4381FC;
  if ( SystemTime.wYear != (unsigned __int16)dword_4381FC || SystemTime.wMonth != v5.wMonth )
    SetSystemTime(&v5);
  v3 = sub_4136D0(a1, dwDesiredAccess, dword_4381B8);
  LeaveCriticalSection(&CriticalSection);
  result = v3;
}
```

The above code is derived from this code flow:



The reason for Shamoon 2.0 changes the time and date settings is because we found out that Shamoon 2.0 uses a commercial product which the malware developers are using which is as called RawDisk by EldoS Corporation. This software gives direct access to files, disk and partitions. The temporary license key for this product is between the time mentioned earlier and hence Shamoon 2.0 changes the system time to make the product believe into thinking that it is using a valid key, and thus the overwrite function can take place.

The MBR-Overwriting Techniques that Shamoon 2.0 Implements:

Before explaining the MBR overwriting that the Shamoon 2.0 does we need to understand what is an MBR or the Master Boot Record (MBR). MBR usually is the **first 512 bytes of the disk** which consists of all the important and crucial information about the data in the disk. The breakdown of the 512 bytes is as shown below:
The reason of overwriting the first 512 bytes of data is

Bootstrap code area	446 bytes
Partition entry 1 Partition entry 2 Partition entry 3 Partition entry 4	Partition table (for primary partitions) 16 bytes x 4 (partitions)
Boot signatureBoot signature	2 bytes
Total	512 bytes

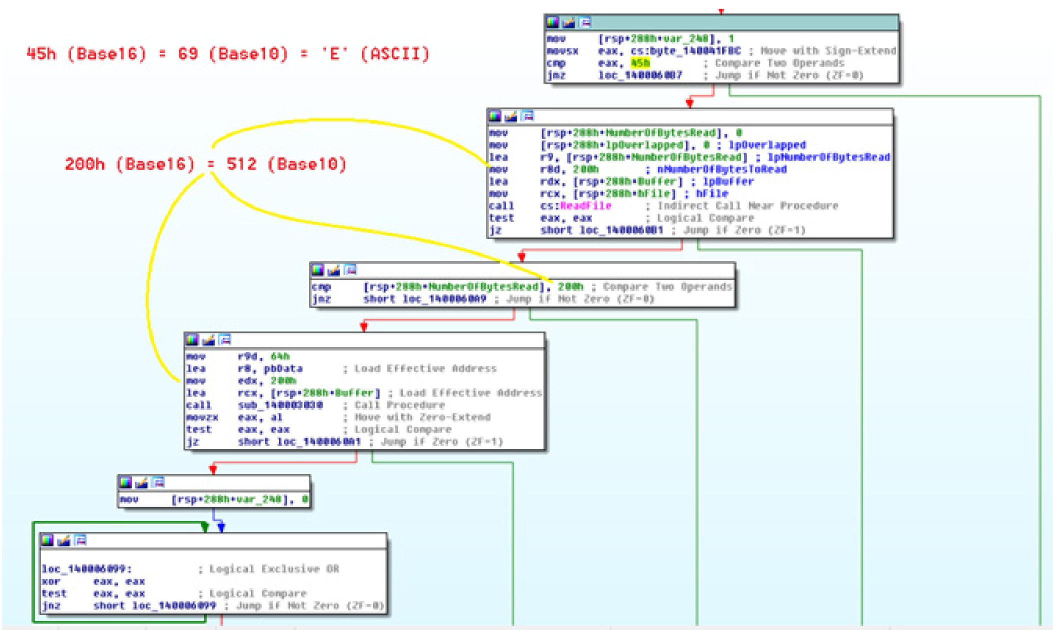
So, that in simple terms mean that target the MBR and lose all data rather than wiping the entire drive all-together.

The following screenshot is a pseudo-code for the MBR-overwrite method that the Shamoon 2.0 uses.

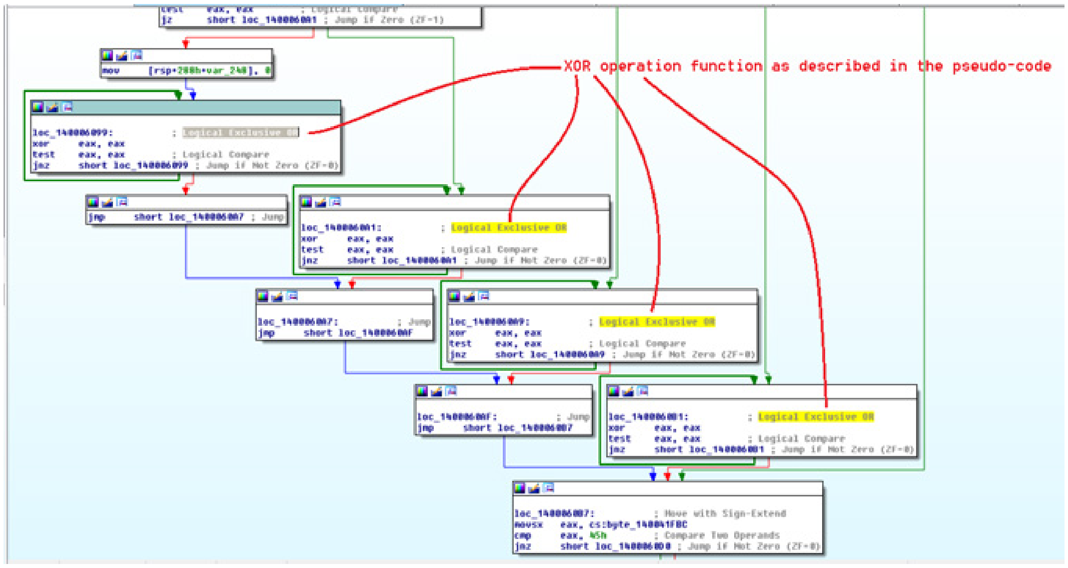
As seen the code above there is a comparison of a variable with '69'. The ASCII equivalent of 69 is 'E'
So, the way the code works in 3 simple steps:

1. Reads the data from the location to overwrite.
2. Uses an XOR table to corrupt the data
3. Write back the XOR'ed values to the location where it read the original data from.

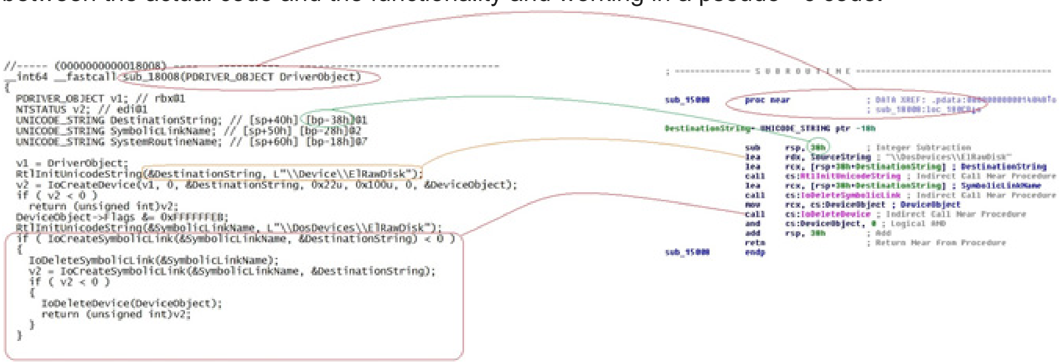
The Above code is derived from the following structure of code:



And the **XOR operations** which are responsible for **overwriting/wiping the data** are in the cascading representations as shown in the image below:



One more module code that we can observe is from the ElRawDisk sample is shown below that shows the correlation between the actual code and the functionality and working in a pseudo - c code.



This particular snippet shows how the **IoDeleteDevice** routine removes a device object from the system, once the MBR is overwritten. This IODeleteDevice routine sends a message to the system notifying that a hard-drive or device is removed, this message is sent because after the MBR is overwritten the system cannot read the drive and this routine tells the system that the device is disconnected from the system and hence the system does not further communicate with the drive. Therefore, the drive is no longer visible on the system.

Conclusion

From the whole analysis, we now can say the following behaviour. The Shamoon sample that is currently spreading is not very different from what spread in its first attack of August 2012. There is a lot of similarity in the previous sample and the new sample. But the new sample is more destructive than the older one. The modules which are split into Dropper, Communication, Wiper are independent and yet linked with one another. From the analysis, we can say that the wiper component is the most important out of the three.

The first stage that the Shamoon 2.0 does it that it checks the system date and compares it with the date embedded. If the date matches it proceeds towards the deletion stage but, otherwise Shamoon 2.0 changes the date into a date which is acceptable to the malware sample and then proceeds to the infection.

The wiper is does not only overwrites the MBR (Master-Boot-Record), but also uses the IODevices module to trigger alerts to the compromised system about the device module being removed from the system altogether making the system completely useless to the user.

The language detected as Arabic Yemen shows that it's a targeted attack towards Middle East.

Indicators of Compromise - SHA 256 hash values

5a826b4fa10891cf63aae832fc645ce680a483b915c608ca26cedbb173b1b80a

c7fc1f9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a
47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51d1fe1ac34
61c1c8fc8b268127751ac565ed4abd6bdab8d2d0f2ff6074291b2d54b0228842
128fa5815c6fee68463b18051c1a1ccdf28c599ce321691686b1efa4838a2acd
394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b

We have launched a Shamoon detection tool which can detect the new Shamoon 2.0.