SectorB06 using Mongolian language in lure document

threatrecon.nshc.net/2019/04/30/sectorb06-using-mongolian-language-in-lure-document

by ThreatRecon Team April 30, 2019

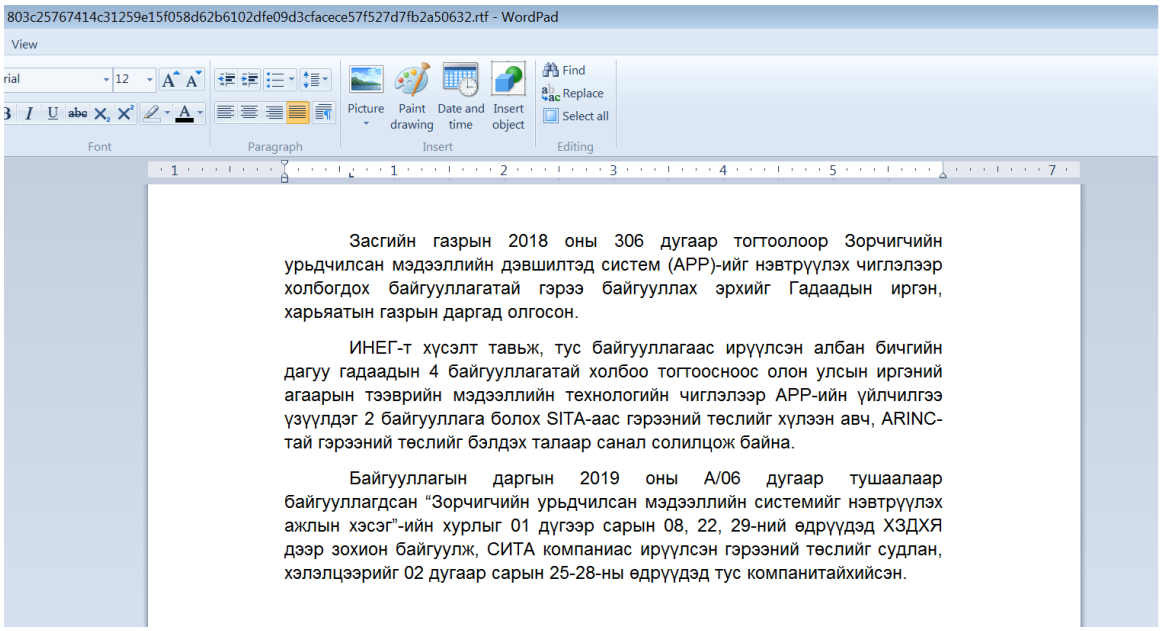
Overview

SectorB06 is a state sponsored threat actor group active especially within Asia. They have been exploiting vulnerabilities in Microsoft Office's Equation Editor [1] which Microsoft removed in January 2018 [2], which in this case seems to be a highly obfuscated version of CVE-2017-11882. The malware we analyzed in this case are sent seemingly only after they already have a basic foothold in their target organizations.

We came across multiple pieces of their malware used in 2019 which appears to be custom compiled on a per target victim per organization basis, with this particular decoy document being uploaded from a Singapore IP address.

Decoy RTF Document

In this example, SectorB06 made use of a Mongolian decoy document to target their victim.



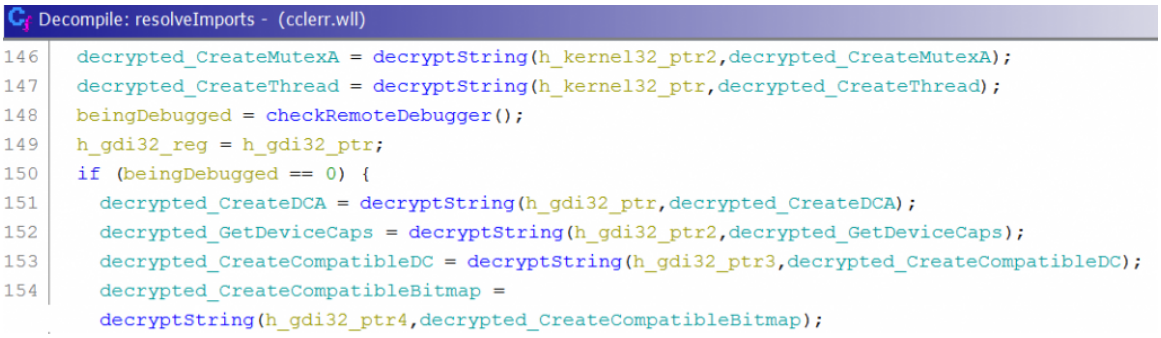
Decoy document written in Mongolian which references the Ministry of Justice and Internal Affairs of Mongolia

If exploitable, the exploit code drops the first-stage malware DLL at "%APPDATA%\Microsoft\Word\STARTUP\cclerr.wll" and runs it.

First Stage Malware (RasTls.dll)

The malware starts off by resolving a list of encoded API addresses by accessing the address of kernel32 from the InMemoryOrderModuleList inside the Process Environment Block (PEB) using FS:[0x30]. It then gets the address of kernel32.LoadLibraryA and kernel32.GetProcAddress from a function which parses kernel32's memory block. This is despite the malware already importing LoadLibraryA() and GetProcAddress(), and is used presumably to prevent automated systems from detecting massive amounts of calls to those functions.

From there, it gets the address of the other libraries it makes use of – Shlwapi.dll, Shell32.dll, Gdi32.dll, User32.dll, and Advapi32.dll. Once that is done, it calls the function which parses the various DLLs again close to 100 times in order to resolve all the APIs it uses. In the middle of those calls, it checks CheckRemoteDebuggerPresent and does not resolve the APIs from the other DLLs if a debugger is found, which will cause the malware to exit later before doing anything malicious.


Decompile: resolveImports - (cclerr.wll)

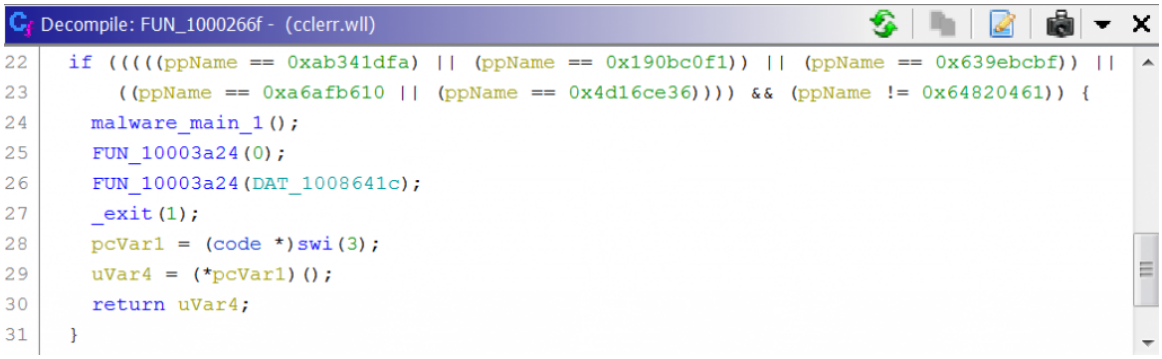
```
146   decrypted_CreateMutexA = decryptString(h_kernel32_ptr2,decrypted_CreateMutexA);
147   decrypted_CreateThread = decryptString(h_kernel32_ptr,decrypted_CreateThread);
148   beingDebugged = checkRemoteDebugger();
149   h_gdi32_reg = h_gdi32_ptr;
150   if (beingDebugged == 0) {
151     decrypted_CreateDCA = decryptString(h_gdi32_ptr,decrypted_CreateDCA);
152     decrypted_GetDeviceCaps = decryptString(h_gdi32_ptr2,decrypted_GetDeviceCaps);
153     decrypted_CreateCompatibleDC = decryptString(h_gdi32_ptr3,decrypted_CreateCompatibleDC);
154     decrypted_CreateCompatibleBitmap =
      decryptString(h_gdi32_ptr4,decrypted_CreateCompatibleBitmap);
```

Malware decrypting the imports it uses via its custom hashing algorithm

It then starts a thread which polls the result of CheckRemoteDebuggerPresent constantly and exits once a debugger is found.

Process Name Hashing

The malware checks for the lower-cased process name it is running under at various steps of execution using a string hashing algorithm. In the first step, it checks against the string hash "0xAB341DFA", "0x190BC0F1", "0x639EBCBF", "0xA6AFB610", "0x4D16CE36", and "0x64820461". It only continues execution if the process name hash is one of the first five hashes and the process name hash is not the last hash. We wrote a custom bruteforcing utility and managed to crack the first five hashes, finding the process names which the attacker expected as "winword.exe", "excel.exe", "powerpnt.exe", "acrord32.exe", and "eqnedt32.exe". While four of these process names are associated with Microsoft Office and the Equation Editor vulnerabilities, "acrord32.exe" (Adobe Reader) is also in the expected process name list because the malware will in some situations rename the legitimate signed Symantec executable file (described later) to "AcroRd32.exe".



Malware making sure the process name is related to the exploit source or itself

Besides this initial check, it also checks the hashes of process names at three other points of execution. Only the hash 0X84F39C89 is checked against the entire process list and is not a lower-case version of the process name.

Hash	Meaning	Description
0X0E867CB6	rundll32.exe	If process is rundll32.exe, do not continue
0XA54ACF71	explorer.exe	If process is not explorer/services.exe, do not continue
0XCD163D44	services.exe	If process is not explorer/services.exe, do not continue
0X84F39C89	<unknown>	If this process exists, do not inject into dllhost.exe

From this we can see there are actually two points from which the malware expects to run from – using the Microsoft Office exploit which injects the second stage malware into dllhost.exe or another path which injects into explorer.exe/services.exe.

Persistence

This first stage malware mainly decompresses and drops two files being used for persistence.

File Name	Description
RasTls.dll	Renamed from cclerr.wll
IntelGraphicsController.exe / AcroRd32.exe	Legitimate signed Symantec file (real name: dot1xtra.exe) from Symantec Network Access Control agent (version 12.1.671.4971) Hash: 724909ba378a872018a3ae0b68afe4949bc404de31bcbd65a6239c12b3a7a3ea

Public examples of a different version of this same signed file being abused in the wild was with version 11.0.4010.7, where the filenames used were rastlsc.exe and iassvcs.exe. Though these files were signed, their certificates have long expired.

The files used for persistence are stored in either the “%AppData%\Intel\Intel(R) Processor Graphic\” or “%PROGRAMFILES%\Intel\Intel(R) Processor Graphics\” directories.

The persistence keys used are in <HIVE>\Software\microsoft\windows\currentversion\run where <HIVE> is either HKLM or HKCU depending on whether there is administrative rights. The name of the registry key used is “IntelGraphicsController” with the value of “<DIRECTORY_TO_INTELGRAPHICSCONTROLLER.EXE> Processid:{0A10C245-2190-7215-A3C5-43215926716A}”.

Commands Ran

The malware runs CreateProcess from a custom command execution function four times, with each run executing takeown/icacls/icacls. The first icacls function attempts to give ownership to the administrators group and the second icacls function attempts to give ownership to the users group. The four runs are for the RasTls.dll file in the %APPDATA% and %PROGRAMFILE% subdirectories and the IntelGraphicsController.exe file in the %APPDATA% and %PROGRAMFILE% subdirectories.

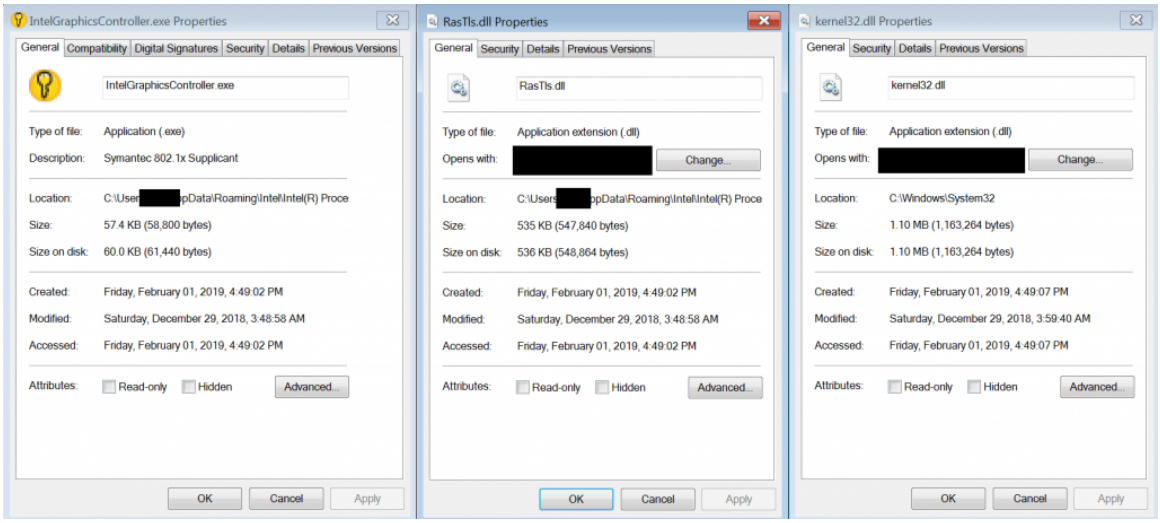
Besides those commands, it also drops two batch files.

<random.bat> – deleting from initial location
1
2
3
Ping 127.0.0.1 -n 10
del “C:\Users\admin\AppData\Roaming\Microsoft\Word\STARTUP\cclerr.wll” /q /f
del %0 /q /f

<random.bat> – attempting to delete winword.exe
1
2
3
Ping 127.0.0.1 -n 10
del “C:\Program Files\Microsoft Office\Office14\WINWORD.EXE” /q /f
del %0 /q /f

Timestomping

The malware uses kernel32’s GetFileTime() and SetFileTime() to get the Creation Time, Last Access Time, and Last Write Time of %windir%\system32\kernel32.dll and saves those same times to the RasTls.dll and IntelGraphicsController.exe files. However, these timestamps are only approximate [3] so the fake times will not be an exact match to kernel32.dll’s file time.



The main two files dropped by the malware for persistence have the approximate timestamps of kernel32.dll

Victim Identification

The malware identifies its victims using <HIVE>\Software\Intel\Java (with <HIVE> being HKLM/HKCU again) with the name “user”. Malware “1-a” referenced below is the current first stage malware we are describing in this post.

Malware “1-a” and “5-a” contain the same victim identifier values, as do “2-a” and “3-a”. This is interesting because of the second stage malware which we describe briefly later.

Malware	Victim Identification Value
1-a	0XdgrHGaayfyBHQ/vCwMP2HE+cNEbzTk 6cZ9bYJOH0R2/z9riKtfcWki36ENBhJ/
2-a	W3qNGgEnxwHShISsHqe4WQlLvmX2q0ms tlCuJVt0/qjwLh7CWXM34rJl66fTyf1u
3-a	4et2q+jmcCeVoPVtVlUeC+Zqq62VN3Q7e7noo8oplXCIv aA22rc7KIYWtv69Nv1rgPeytor20Dv5..oEFQze78uA==
5-a	0XdgrHGaayfyBHQ/vCwMP2HE+cNEbzTk 6cZ9bYJOH0SxvpFWecTmuneM/5p93lQw

Process Injection

Finally, the malware performs process injection into “%windir%\system32\dllhost.exe /Processid:{712459B2-3311-54C3-910D-0327080553246}” without the second stage ever touching the disk. The injected process, dllhost.exe, is typically a container process for running COM DLLs. The list of CLSIDs in a system can be seen in KEY_CLASSES_ROOT\CLSID. We are unsure what the hardcoded CLSID value of “712459B2-3311-54C3-910D-0327080553246” is supposed to represent, but a likely guess is a CLSID used by Symantec since the malware is impersonating their executable file.

Second Stage Malware

While we did not analyze the second stage malware in large detail, we did decode the C2 information among other data such as credentials. The samples we analyzed appear to connect to two external C2 IP addresses 217[.]69[.]8[.]255 and 1[.]187[.]1[.]187 on port 443. It also references an internal IP address, which indicate that these spear phishing documents are sent to targeted victims and only after the attacker already has basic access to the victim's internal network.

One of the purposes of this second stage malware also appears to be for creating a remote command shell.

Malware	Internal IPs Referenced
1-b	192[.]168[.]43[.]234
2-b	192[.]168[.]111[.]111
3-b	192[.]168[.]111[.]111
4-b	192[.]168[.]43[.]234
5-b	192[.]168[.]43[.]234
6-b	192[.]168[.]43[.]234

With the malware trying to target/use the same internal IP but with different user identification values, we see how the attacker is custom compiling each malware executable for each victim/attempt in a specific organization.

Summary

SectorB06 is a threat group with very specific interests and in the case of these malware, appears to either already have a basic foothold in the victim network or has already gained and then lost access to the network. They are actively developing their toolkit and are adept at bypassing security solutions at least statically especially for their exploit document and second stage malware.

Indicators of Compromise

Decoy Hash (SHA-256)

803c25767414c31259e15f058d62b6102dfe09d3cfacece57f527d7fb2a50632

First Stage Hashes (SHA-256)

304115cef6cc7b81f4409178cd0bcea2b22fd68ca18dfd5432c623cbbb507154
6086b407ed69434fce117bc173f70a2ec147fdf119cf38f6031c1889e19ff8bf
240f2c0cd808991b2c77a978203c661612e250df2b0bad9fd452b6c21d60b324
d0ccb9a277b986f7127199f122023c79a7e0253378a4a78806fbf55a87633532

Second Stage Hashes (SHA-256)

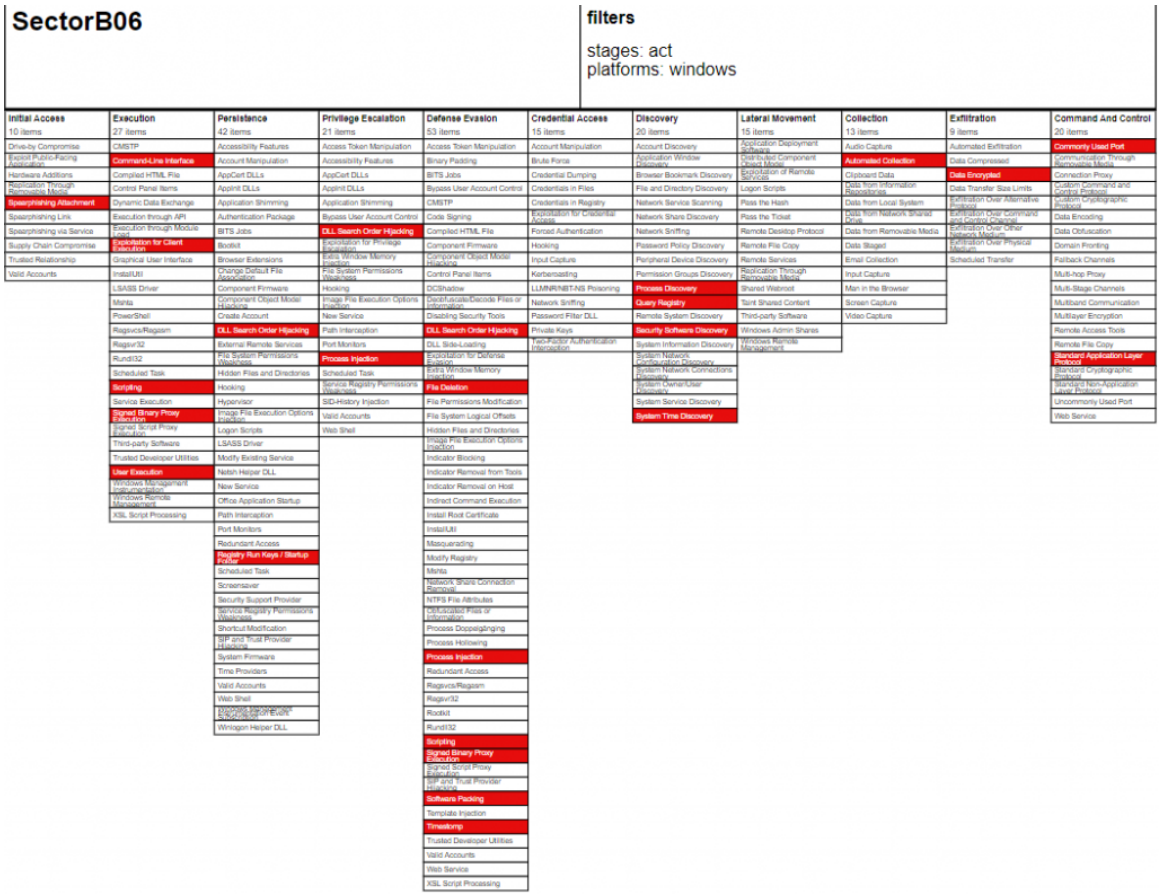
87c4eb8201f9cf92aa5562d112fdd322a01899bcc38ba39e4f6ef92cbf144900
fcb0d071a9384750adf88963eb580690effbe8b29942afa6a8e2566e9a4e94dd
e8446ba200c9d703fab7ddc068b45772585ae782a8bcf4c5f86782d7220405f7
32fbd62a1fde794cdf95a67f22f47b495474cd18419ac4c37fbb5460cdfd1831
85bdd517886e645a3d0e4e4bc16ede5bbb126eaf86c0d14c05a951219f48555a

IP Addresses

217[.]69[.]8[.]255
1[.]187[.]1[.]187

MITRE ATT&CK Techniques

The following is a list of MITRE ATT&CK Techniques we have observed based on our analysis of these malware.



Initial Access

T1193 Spearphishing Attachment

Execution

T1059 Command-Line Interface
T1203 Exploitation for Client Execution
T1064 Scripting

T1204 User Execution
T1218 Signed Binary Proxy Execution

Persistence

T1038 DLL Search Order Hijacking
T1060 Registry Run Keys / Startup Folder

Defense Evasion

T1116 Code Signing
T1038 DLL Search Order Hijacking
T1107 File Deletion
T1055 Process Injection
T1218 Signed Binary Proxy Execution
T1045 Software Packing
T1099 Timestomp

Discovery

T1057 Process Discovery
T1012 Query Registry
T1063 Security Software Discovery
T1124 System Time Discovery

Collection

T1119 Automated Collection

Exfiltration

T1022 Data Encrypted

Command and Control

T1043 Commonly Used Port
T1071 Standard Application Layer Protocol

References

[1] Microsoft Office : List of security vulnerabilities
https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-320/cvssscoremin-9/cvssscoremax-/Microsoft-Office.html
[2] CVE-2018-0802 | Microsoft Office Memory Corruption Vulnerability
https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/CVE-2018-0802
[3] GetFileTime function
https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-getfiletime