

Pacific Ring of Fire: PlugX / Kaba

July 24, 2014 | By Geok Meng Ong, Chong Rong Hwa | Threat Intelligence, Threat Research, Advanced Malware, Targeted Attack

As depicted in earlier [FireEye blogs](#), advanced cyber attacks are no strangers to the Asia Pacific region. In this blog, we take a deeper look at some of the advanced persistent threat (APT) malware that have significant presence in the APAC region, starting with PlugX (we detect it as Backdoor.APT.Kaba).

The PlugX / Kaba malware is a well-known remote access tool (RAT) believed to have been around for several years that continues to evolve itself in new attack campaigns. It is often seen used in APT campaigns alongside two other infamous RATs – PoisonIvy and Taidoor. For this blog, FireEye Labs has investigated PlugX samples discovered throughout 2013 as well as recent variants detected between January and June 2014. Countries on both sides of the Pacific including the United States as well as Northeast Asian countries such as South Korea, Hong Kong, Japan and Taiwan were most hit by this malware, with attacks spanning multiple industry verticals. The top 5 most targeted verticals include Technology, Aerospace / Defense, Entertainment / Media, Telecommunications and Government (Federal).

[caption id="attachment_6004" align="alignnone" width="638"]

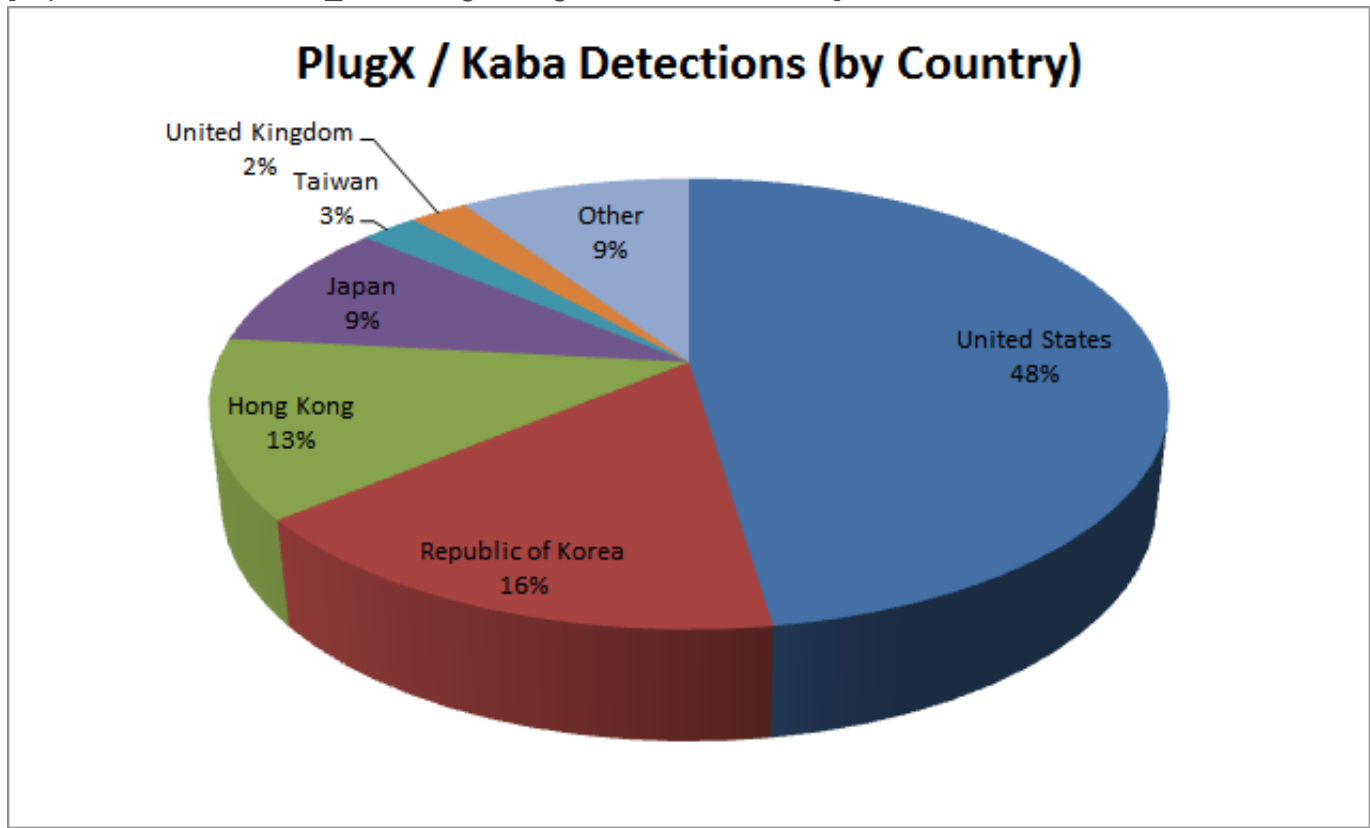


Figure 1:

PlugX / Kaba Detections (by Country)[/caption]

[caption id="attachment_6005" align="alignnone" width="652"]

Industry	Exposure
Technology	47.4%
Aerospace / Defense	30.8%
Entertainment /Media	9.8%
Telecommunication	8.7%
Government (Federal)	3.3%

Table 1:

Top 5 Affected Verticals[/caption]

Delivering the Attacks

PlugX is most commonly distributed via an exploit, but may also be delivered using a RAR self-extracting executable. Amanda Stewart has written [an excellent blog](#) and paper about the common components of the PlugX / Kaba RAT and how it capitalizes on [the DLL side-loading technique](#). In general, the RAT consists of DLL components that are injected into the process memory of svchost.exe. To deliver the DLL components, a “dropper” must first be executed through the use of an exploit, or via social-engineering tactics over e-mail or web to entice the victims to load an executable file.

[caption id="attachment_6006" align="alignnone" width="552"]

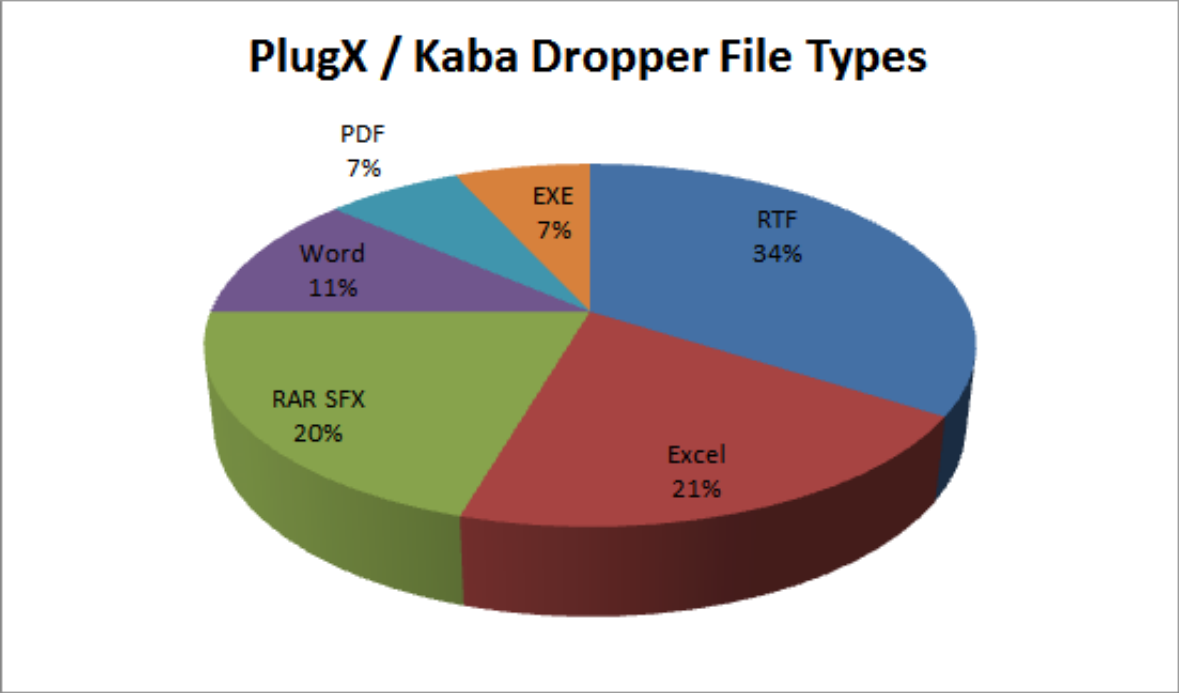


Figure 2: Primary PlugX

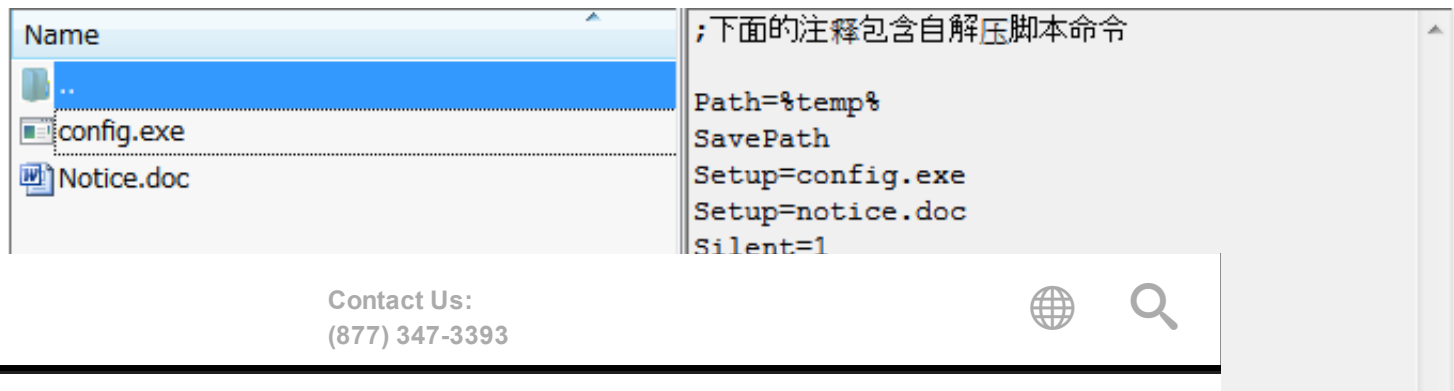
“Dropper” File Types[/caption]

While RTF files exploiting CVE-2012-0158 are nothing new, they are still most frequently used in the delivery of PlugX to its targets. The same vulnerability has also been exploited through Excel spreadsheets and Word document files. More recently, [a Flash zero-day vulnerability has been exploited to deliver a PlugX payload](#).

Where an exploit is not used, RAR self-extracting executable (SFX) files were commonly used throughout 2013. These files often appear to have a Word or PDF icon and launch a decoy document that is displayed to the victim. The PlugX RAT is then loaded in the background without the user's knowledge. While we have noticed a decrease in the use of this vector to deliver PlugX in 2014, it continues to be an effective technique for PlugX and other malware, so we do not expect its use to disappear entirely.

In the below example, the RAR SFX contains a script that loads the RAT (config.exe) and the decoy document (notice.doc).

[caption id="attachment_6007" align="alignnone" width="679"]



Menu

Command and Control

We have found two dominant variants, SideBar and RasTLS, using 4 of the top 10 domains associated with the PlugX / Kaba command and control (C2) infrastructure. In fact, the 4 domains resolved to the same IP range based in Hong Kong likely operated by the same threat group(s).

[caption id="attachment_6008" align="alignnone" width="679"]

Domain	First Seen	Last Seen	IP Address	Callbacks
scqf.bacguarp.com	01/08/2014	01/08/2014	210.56.63.61	21.6%
bbs.zuesinfo.com	10/01/2013	10/01/2013	210.56.63.61	18.9%
scqf.zuesinfo.com	02/14/2014	2014/05/21	210.56.63.61	10.8%
fast.bacguarp.com	10/01/2013	06/27/2014	210.56.63.60	10.8%
vip.kavupdate.com	09/07/2012	04/22/2013	173.13.112.122	8.1%
strnewgm.network-sec.net	03/07/2013	05/03/2013	103.244.149.60	8.1%
cache.mindplat.com	07/11/2013	06/10/2014	202.30.225.15	5.4%
sql.so-webmail.com	11/16/2013	11/27/2013	103.246.112.123	5.4%
philippines.crabdance.com	06/22/2014	06/27/2014	103.226.153.104	5.4%
hansoft.sunsb.net	01/20/2013	06/27/2014	67.229.53.114	5.4%

Table

2: Top Domains used in PlugX / Kaba Callbacks[/caption]

SideBar

The SideBar variant is delivered through RTF, Word and Excel files. Upon successfully exploitation, it drops “dw20.dll” to the %TEMP% folder. This “dw20.dll” continues to install the following files:

- %ALLUSERS%\WS\Gadget.exe (MD5: 6b97b3cd2fcfb4b74985143230441463)
- %ALLUSERS%\WS\SideBar.dll (MD5: 123e1841cc596c1f40e2e6693ea7dcac)
- %ALLUSERS%\WS\SideBar.dll.doc (MD5: a0c93bdc089e1338cc392108a0e57f2f)

A service registry key is created to start “Gadget.exe” upon reboot of the infected system.

"Gadget.exe" is part of a benign “TENCENT SideBar” application digitally signed by “Tencent Technology(Shenzhen) Company Limited “. Using the [DLL-side loading method](#), a malicious version of“SideBar.dll” is loaded and executes the exported function “Main”.

“SideBar.dll” is a loader for “SideBar.dll.doc”, executing code at offset 0. “SideBar.dll.doc” decodes a part of its own data and is responsible for deflating a backdoor component. It spawns a new svchost.exe process and injects the backdoor into memory. This backdoor component remains only in memory, and is never saved to disk.

[caption id="attachment_6009" align="alignnone" width="609"]

00420571	8B46 0C	MOV EAX,DWORD PTR DS:[ESI+0C]	
00420574	83E8 04	SUB EAX,4	
00420577	50	PUSH EAX	
00420578	8B46 08	MOV EAX,DWORD PTR DS:[ESI+8]	
0042057B	83C0 04	ADD EAX,4	
0042057E	50	PUSH EAX	
0042057F	53	PUSH EBX	
00420580	FF75 E8	PUSH DWORD PTR SS:[EBP-18]	
00420583	6A 02	PUSH 2	
00420585	FF55 F4	CALL DWORD PTR SS:[EBP-0C]	ntdll.RtlDecompressBuffer
00420588	85C0	TEST EAX,EAX	
0042058A	74 07	JE SHORT 00420593	
0042058C	6A 00	PUSH 00	
0042058E	E9 9E020000	JMP 00420831	
00420593	395D B4	CMP DWORD PTR SS:[EBP-4C],EBX	
00420596	74 07	JE SHORT 0042059F	
00420598	6A 0E	PUSH 0E	
0042059A	F9 92020000	.IMP 00420831	

Figure 4:

Decompressing Encoded Data[/caption]

Version information can often be found in PlugX's process memory. In SideBar, a DWORD value storing the internal version number was 0x20120123. The path names found in the deflated backdoor's process memory indicating that this PlugX variant is version 6.0:

- "d:\work\plug6.0(360)(gadget)"

The variant connects to fast.bacguarp.com and bbs.zuesinfo.com over port 8080.

RasTLS

While the RasTls variant is also dropped by document exploits, the dropped files are different. RasTls does not use the DLL side-loading method found in older variants [3]. The DWORD used to store the internal version number of RasTls was 0x20130810.

- %ALLUSERS%\DRM\RasTls\RasTls.exe

"RasTls.exe" spawns "svchost.exe" and injects a deflated backdoor component into memory. The deflated backdoor component in memory contains a "XV" marker, instead of "MZ" and "PE" as found in regular Windows portal executable (PE) files. This is because "RasTls.exe" manually loads each section of deflated file into memory, so the file does not have to be a complete PE image.

[caption id="attachment_6010" align="alignnone" width="451"]

Address	Hex dump	ASCII
001D0000	58 56 00 00 00 00 00 00 00 00 00 00 00 00 00 00	XU
001D0010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D00A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D00B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D00C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D00D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D00E0	58 56 00 00 4C 01 04 00 1F F8 D4 52 00 00 00 00	XU L0+ 7°=R
001D00F0	00 00 00 00 E0 00 02 21 0B 01 0A 00 F2 00 00 00	α 0†000 2
001D0100	00 8A 00 00 00 00 00 00 70 14 00 00 00 10 00 00	è 0†000 p7
001D0110	00 10 01 00 00 00 00 10 00 10 00 00 00 02 00 00	† 0 † 0 † 0 † 0
001D0120	05 00 01 00 00 00 00 05 00 01 00 00 00 00 00 00	‡ 0 † 0 † 0 † 0
001D0130	00 00 01 00 00 00 04 00 00 00 00 00 02 00 40 05	‡ 0 † 0 † 0 † 0
001D0140	00 00 10 00 00 10 00 00 00 00 00 10 00 10 00 00	‡ 0 † 0 † 0 † 0
001D0150	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00	‡ 0 † 0 † 0 † 0
001D0160	58 27 01 00 3C 00 00 00 00 00 00 00 00 00 00 00	X'0 <
001D0170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D0180	00 90 01 00 20 00 00 00 00 00 00 00 00 00 00 00	ε0 7
001D0190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D01A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D01B0	00 00 00 00 00 00 00 00 00 00 10 01 00 74 00 00	‡ 0 ö
001D01C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D01D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
001D01E0	AB F0 00 00 00 10 00 00 00 F2 00 00 00 04 00 00	½= † 2 † ,
001D01F0	00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60	
001D0200	00 00 00 00 00 00 00 00 54 1A 00 00 00 10 01 00	T+ † 0
001D0210	00 1C 00 00 00 F6 00 00 00 00 00 00 00 00 00 00	L ÷ 0 0
001D0220	00 00 00 00 40 00 00 40 00 00 00 00 00 00 00 00	0 0 0 0
001D0230	6C 56 00 00 00 30 01 00 00 00 00 00 00 00 00 00	UV 00 0 0
001D0240	00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0	
001D0250	00 00 00 00 00 00 00 00 9C 14 00 00 00 90 01 00	0 0 0 0 0 0
001D0260	00 16 00 00 00 12 01 00 00 00 00 00 00 00 00 00	- 0 0 0 0 0 0
001D0270	00 00 00 00 40 00 00 42 00 00 00 00 00 00 00 00	0 B
001D0280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

Figure 5: Backdoor Component with "XV"

maker instead of "MZ" and "PE"[/caption]

The variant doesn't contain strings implying version. The variant accept commands like as "ST1", "ST2", "TT1", "TT2" which are different from version 6.

In memory space in the "svchost.exe", we can see the decoded configuration information:

[caption id="attachment_6011" align="alignnone" width="488"]

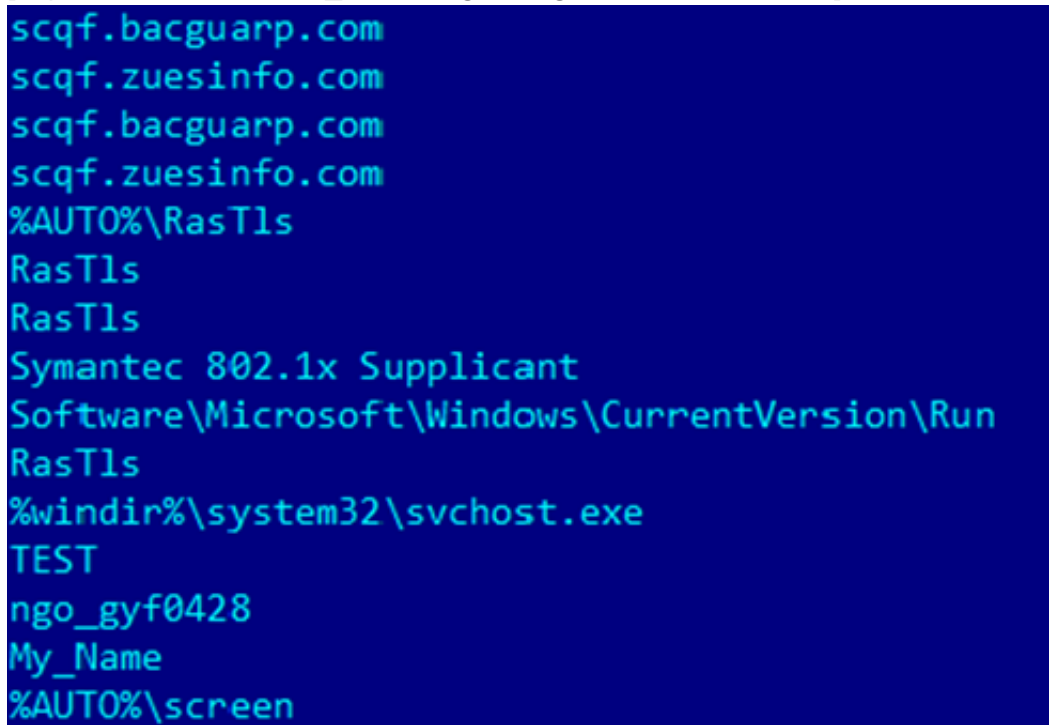


Figure 6: Decoded Configuration

Information[/caption]

All RasTls variants have largely identical configuration and connects to scqf.bacguarp.com and scqf.zuesinfo.com over port 443. The "My_Name" mutex is also common to all RasTls variants.

PlugX Encryption Algorithm

PlugX has a variety of encryption algorithms used to encrypt its data across variants. However, the encryption style is largely similar as depicted in Figure 7.

[caption id="attachment_6012" align="alignnone" width="661"]

```
def DecryptKaba (EncryptedData, EncryptedLen, key):
    sub_key1 = key;
    sub_key2 = key;
    sub_key3 = key;
    sub_key4 = key;
    DecryptedData = "";
    for i in range (EncryptedLen):
        sub_key1 = (sub_key1 + (sub_key1 >> 3) + 3) & 0xFFFFFFFF;
        sub_key2 = (sub_key2 + (sub_key2 >> 5) + 5) & 0xFFFFFFFF;
        sub_key3 = (0xFFFFFFFF81 * sub_key3 - 7) & 0xFFFFFFFF;
        sub_key4 = (0xFFFFFE01 * sub_key4 - 9) & 0xFFFFFFFF;

        xor_key = (sub_key4 + sub_key3 + sub_key2 + sub_key1) & 0xFF;
        DecryptedData = DecryptedData + chr(xor_key ^ ord (EncryptedData[i])) );
    return DecryptedData;
```

Older algorithm in win3dx.dll
Internal version 0x20100921
Reference: MD5 7ADAE0335C9D6C9F3826CDE9747438B7

```
def DecryptKaba (EncryptedData, EncryptedLen):
    key = Dword (EncryptedData);
    if key == 0xFFFFFFFF:
        return ""
    EncryptedLen = EncryptedLen - 4
    decryptedString=""
    for i in range (EncryptedLen):

        key1 = (((key << 7) - (key >> 3) + i + 0x713A8FC1) & 0xFFFFFFFF) >> 16) & 0xFFFFFFFF
        key2 = (((key << 7) - (key >> 3) + i + 0x713A8FC1) & 0xFFFFFFFF) >> 24) & 0xFFFFFFFF
        #update key
        key = ((key << 7) - (key >> 3) + i + 0x713A8FC1) & 0xFFFFFFFF ;
        xor_key = (key & 0xFF) ^ ((key >> 8) & 0xFF) ^ (key1 & 0xFF) ^ (key2 & 0xFF)
        decryptedString = decryptedString + chr(Byte (EncryptedData + i + 4)^xor_key)

    return decryptedString;
```

Decryption algorithm in RasTls.exe
Internal version 0x20130810
Reference: MD5 3055B6A64A8D9C7B02D6A54DC6AD92AE

Figure 7: PlugX Decryption Algorithm[/caption]

In RasTls, the DWORD decryption key was found in the first four bytes of the encrypted string. It was also less aggressive in encrypting and hiding its data. In older variants such as "win3dx.DLL" (MD5: 7ADAE0335C9D6C9F3826CDE9747438B7), most API names were decrypted before loading and nullified after use. This makes understanding the malware slightly more difficult for malware analysts.

The supported functionalities are largely similar where it uses the identical command code. Below is a list of PlugX commands for file system manipulation:

- 0x3000: GetDiskRelatedInformation
- 0x3001: SearchDirectoryForFiles
- 0x3002: SearchDirectoryRecursively
- 0x3004: ReadFile
- 0x3007: WriteFile
- 0x300A: CreateDirectory
- 0x300C: CreateWindowsDesktop
- 0x300D: PerformSH_FileOperation

- 0x300E: ExpandEnvironmentVariable

Some improvements were made by its developers. For example, the key logger function was updated to utilize the GetRawInputData API to collect keystrokes. “RegisterRawInputDevices” and “GetRawInputData” were two of the few API names that remain encrypted in RasTls.

[caption id="attachment_6013" align="alignnone" width="353"]

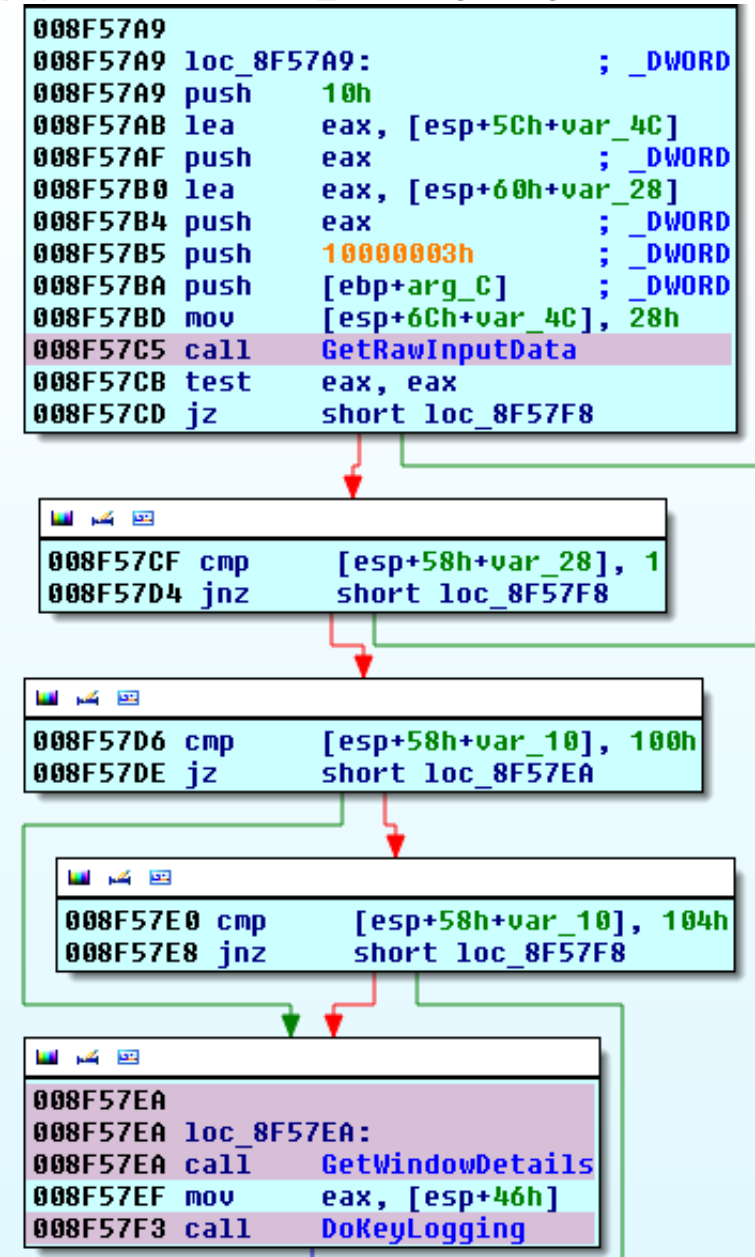


Figure 8: Updated Key Logging Component[/caption]

PlugX / Kaba Trending

Figure 9 shows the trending of total PlugX / Kaba infections and their variants: SideBar and RasTls. The spike in September 2013 was caused by SideBar. In 2014, we see SideBar and RasTls on an inverse trend, with the latter on a steady increase.

[caption id="attachment_6014" align="alignnone" width="735"]

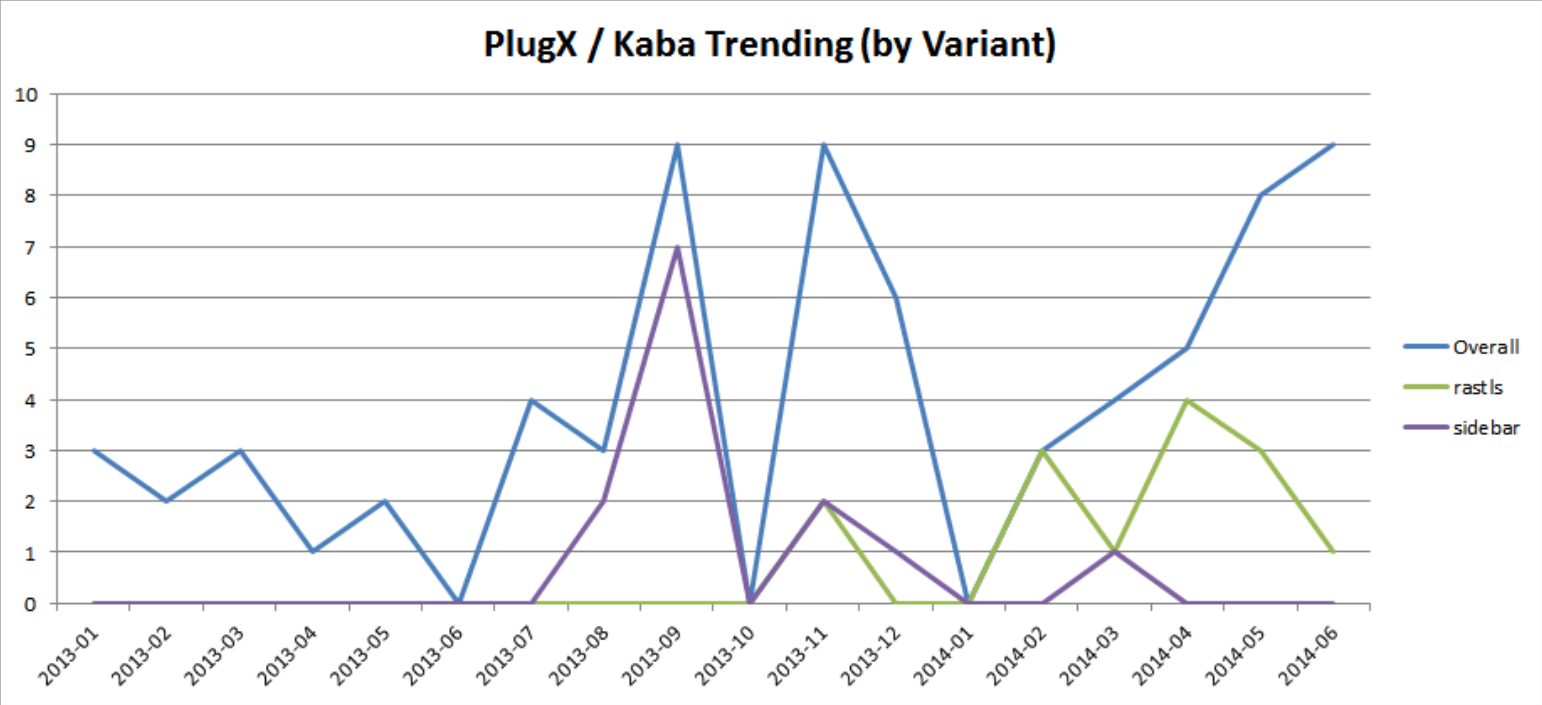


Figure 9: Trending of Overall PlugX / Kaba , SideBar, RasTLS Infections[/caption]

Figure 10 shows the distribution of SideBar/RasTls variants by country. The C2 servers are located in Hong Kong, where much of the attacks have occurred. We also find a variety of countries targeted by these variants. In some of the exploit documents delivering these variants, the content revolves around the theme of NGOs and socio-political events in China and Japan. These are content that would likely be of interest to the victims who would be opening the documents.

[caption id="attachment_6015" align="alignnone" width="728"]

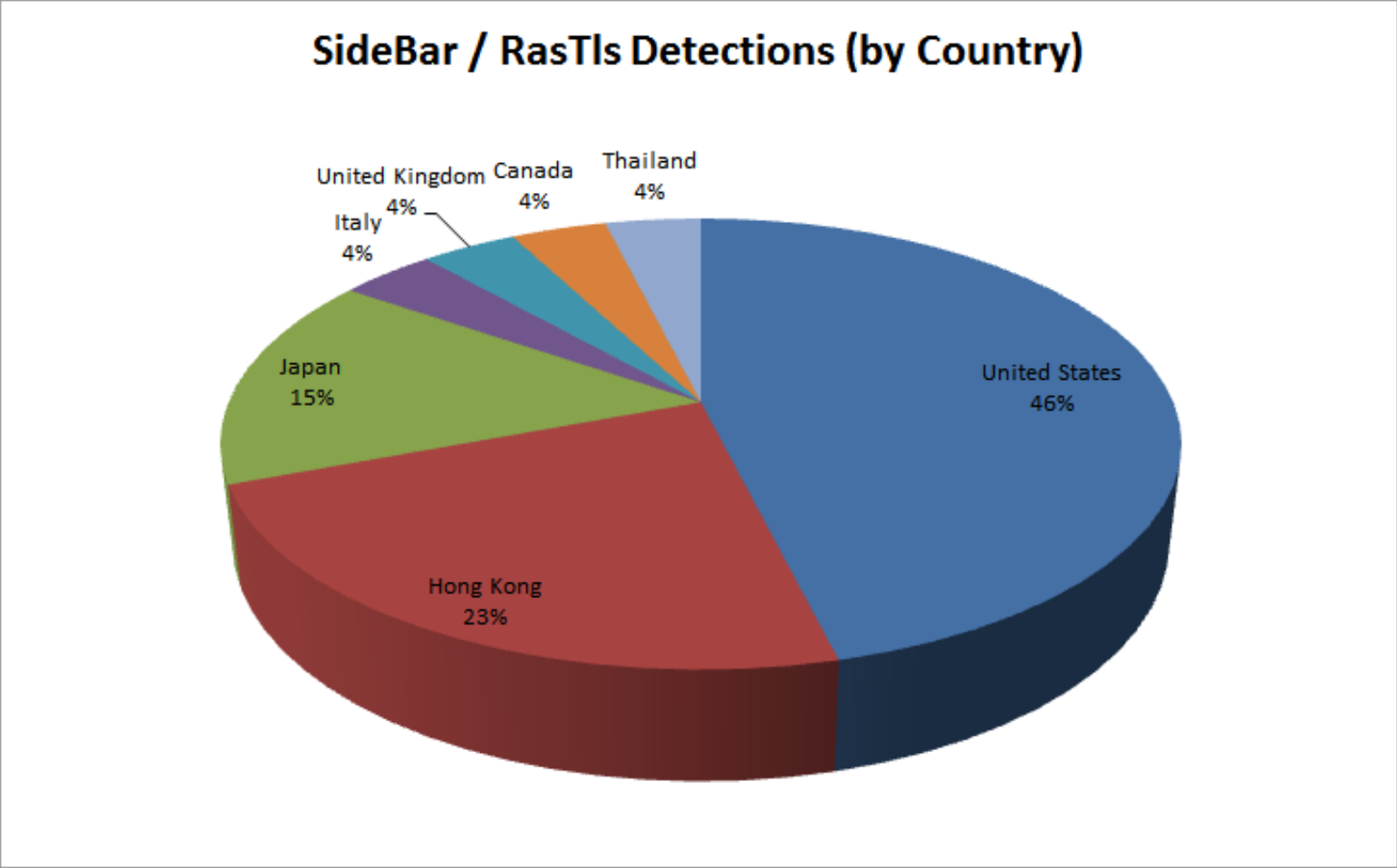


Figure 10: Distribution of SideBar/RasTls Variants by Country[/caption]

Conclusion

The Asia Pacific region remains a highly attractive target of advanced cyber-attacks. Many threat groups have a particular interest in this region, and are likely to continue to launch new attacks against targets here. We recommend that users in this region block access to the above C2 servers. FireEye Labs will continue to monitor and report on new PlugX / Kaba developments.

This entry was posted on Thu Jul 24 21:00:16 EDT 2014 and filed under [Advanced Malware](#), [Advanced Malware](#), [Blog](#), [Chong Rong Hwa](#), [Geok Meng Ong](#), [Targeted Attack](#), [Threat Intelligence](#) and [Threat Research](#).

FireEye Alerts

Be the first to receive information on major cyber attacks from the industry leader!

First Name

Last Name






Email Address

Subscribe



- Cyber Security Fundamentals
- Careers
- Events
- Webinars
- Support
- Partners
- Newsroom
- Blog
- Incident?
- Contact Us
- Communication Preferences
- Report Security Issue
- Supplier Documents

Connect

-  Facebook
-  LinkedIn
-  Twitter
-  Google+
-  YouTube
-  Glassdoor

