ChChes – Malware that Communicates with C&C Servers Using Cookie Headers - JPCERT/CC Eyes

By JPCERT/CC

Published: 2017-02-14 · Archived: 2026-04-05 18:27:25 UTC

- ChChes

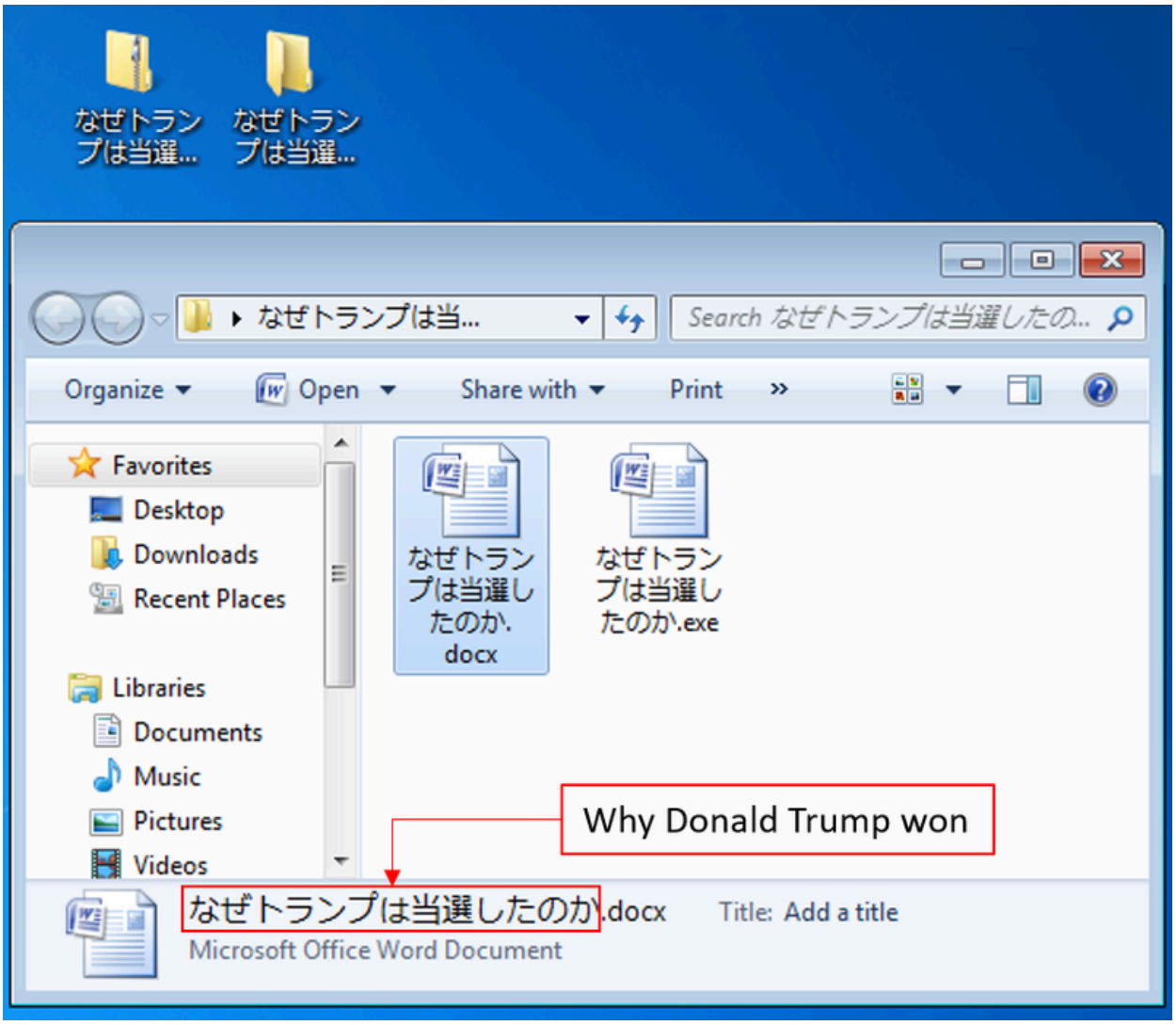
Since around October 2016, JPCERT/CC has been confirming emails that are sent to Japanese organisations with a ZIP file attachment containing executable files. The targeted emails, which impersonate existing persons, are sent from free email address services available in Japan. Also, the executable files' icons are disguised as Word documents. When the recipient executes the file, the machine is infected with malware called ChChes.

This blog article will introduce characteristics of ChChes, including its communication.

ZIP files attached to Targeted Emails

While some ZIP files attached to the targeted emails in this campaign contain executable files only, in some cases they also contain dummy Word documents. Below is the example of the latter case.

Figure 1: Example of an attached ZIP file



In the above example, two files with similar names are listed: a dummy Word document and an executable file whose icon is disguised as a Word document. By running this executable file, the machine will be infected with ChChes. JPCERT/CC has confirmed the executable files that have signatures of a specific code signing certificate. The dummy Word document is harmless, and its contents are existing online articles related to the file name “Why Donald Trump won”. The details of the code signing certificate is described in Appendix A.

Communication of ChChes

ChChes is a type of malware that communicates with specific sites using HTTP to receive commands and modules. There are only few functions that ChChes can execute by itself. This means it expands its functions by receiving modules from C&C servers and loading them on the memory.

The following is an example of HTTP GET request that ChChes sends. Sometimes, HEAD method is used instead of GET.

```
 GET /X4iBJjp/MtD1xyoJMQ.htm HTTP/1.1
Cookie: uHa5=kXFGd3JqQHMfnMbi9mFZAJHCGja0ZLs%3D;KQ=yt%2Fe(omitted)
```

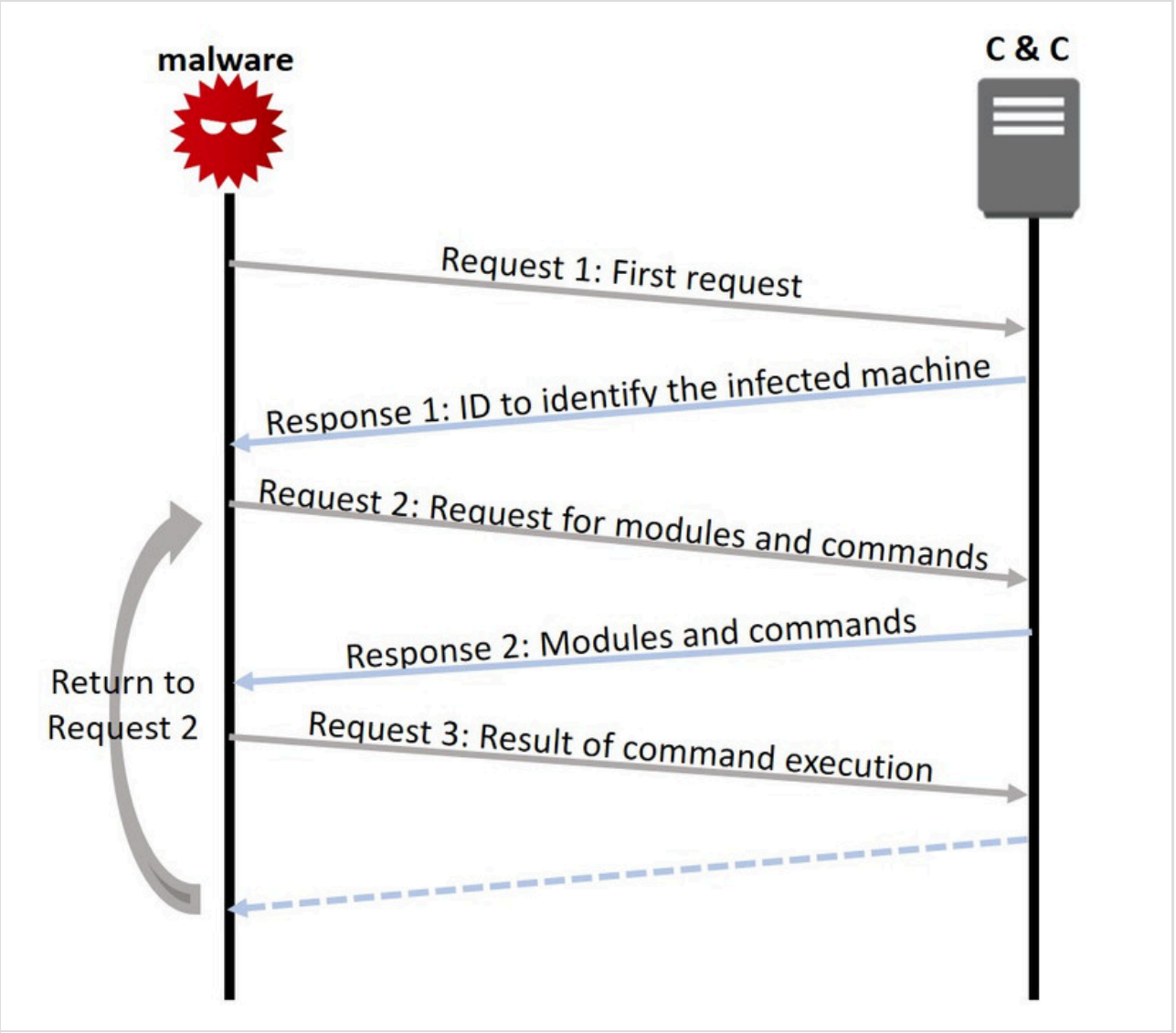
```
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: [user agent]
Host: [host name]
Connection: Keep-Alive
Cache-Control: no-cache
```

As you can see, the path for HTTP request takes */[random string]*.htm, however, the value for the Cookie field is not random but encrypted strings corresponding to actual data used in the communication with C&C servers. The value can be decrypted using the below Python script.

```
 data_list = cookie_data.split(';')
dec = []
for i in range(len(data_list)):
    tmp = data_list[i]
    pos = tmp.find("=")
    key = tmp[0:pos]
    val = tmp[pos:]
    md5 = hashlib.md5()
    md5.update(key)
    rc4key = md5.hexdigest()[8:24]
    rc4 = ARC4.new(rc4key)
    dec.append(rc4.decrypt(val.decode("base64"))[len(key):])
print("[*] decoded: " + "".join(dec))
```

The following is the flow of communication after the machine is infected.

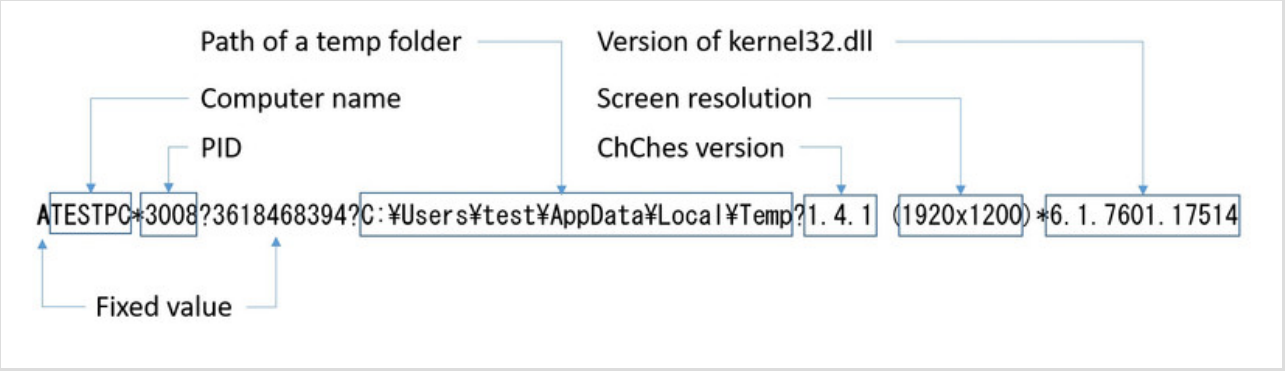
Figure 2: Flow of communication



The First Request

The value in the Cookie field of the HTTP request that ChChes first sends (Request 1) contains encrypted data starting with ‘A’. The following is an example of data sent.

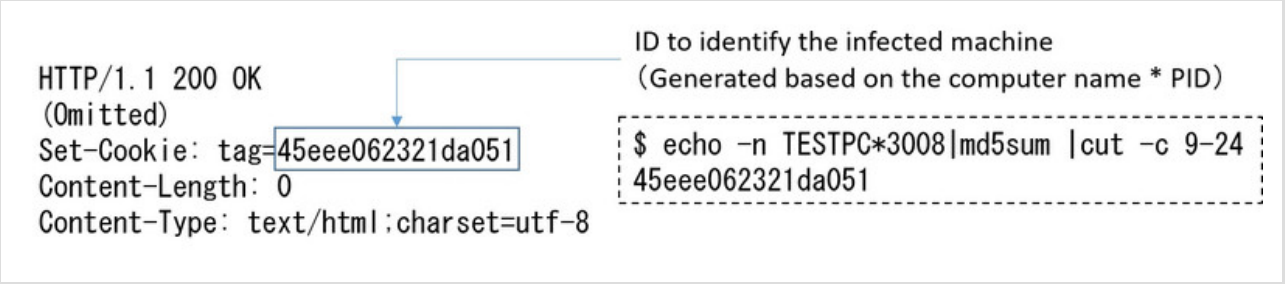
Figure 3: Example of the first data sent



As indicated in Figure 3, the data which is sent contains information including computer name. The format of the encrypted data differs depending on ChChes’s version. The details are specified in Appendix B.

As a response to Request 1, ChChes receives strings of an ID identifying the infected machine from C&C servers (Response 1). The ID is contained in the Set-Cookie field as shown below.

Figure 4: Example response to the first request



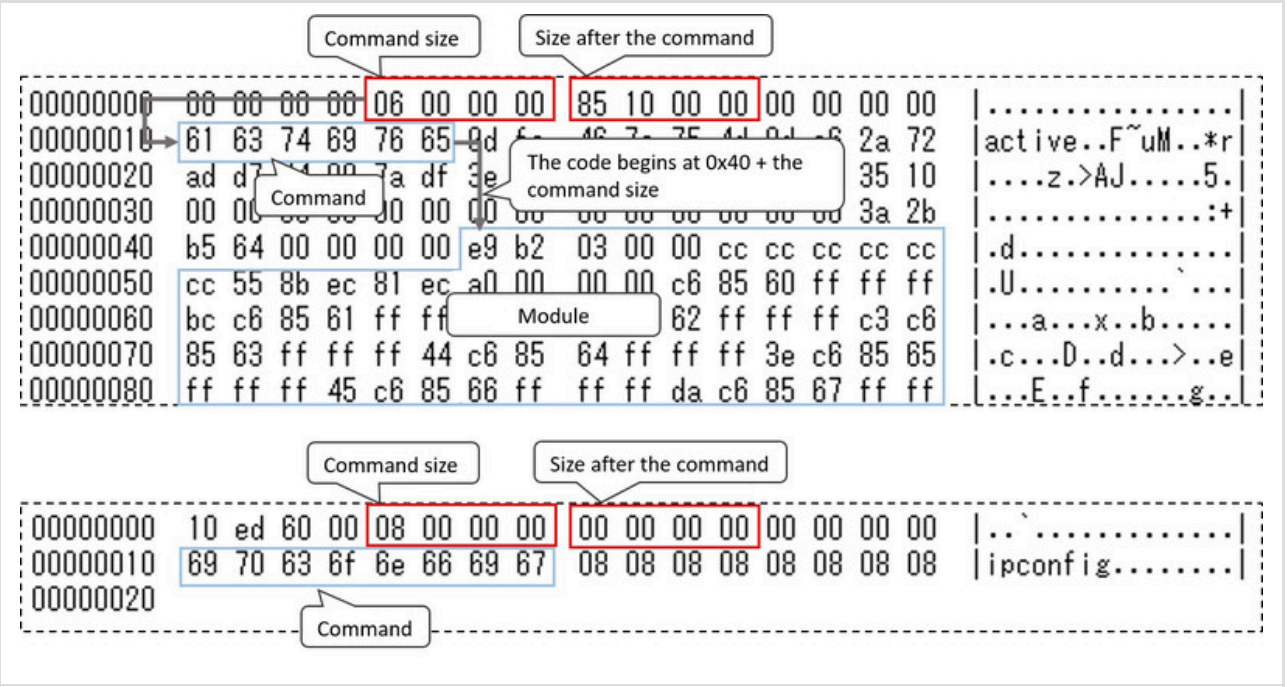
Request for Modules and Commands

Next, ChChes sends an HTTP request to receive modules and commands (Request 2). At this point, the following data starting with 'B' is encrypted and contained in the Cookie field.

```
B[ID to identify the infected machine]
```

As a response to Request 2, encrypted modules and commands (Response 2) are sent from C&C servers. The following shows an example of received modules and commands after decryption.

Figure 5: Decrypted data of modules and commands received



Commands are sent either together with modules as a single data (as above), or by itself. Afterwards, execution results of the received command are sent to C&C servers, and it returns to the process to receive modules and commands. This way, by repeatedly receiving commands from C&C servers, the infected machines will be controlled remotely.

JPCERT/CC's research has confirmed modules with the following functions, which are thought to be the bot function of ChChes.

- Encrypt communication using AES
- Execute shell commands
- Upload files
- Download files
- Load and run DLLs
- View tasks of bot commands

Especially, it was confirmed that the module that encrypts the communication with AES is received in a relatively early stage after the infection. With this feature, communication with C&C servers after this point will be encrypted in AES on top of the existing encryption method.

Summary

ChChes is a relatively new kind of malware which has been seen since around October 2016. As this may be continually used for targeted attacks, JPCERT/CC will keep an eye on ChChes and attack activities using the malware.

The hash values of the samples demonstrated here are described in Appendix C. The malware's destination hosts that JPCERT/CC has confirmed are listed in Appendix D. We recommend that you check if your machines are communicating with such hosts.

Thanks for reading.

- Yu Nakamura

(Translated by Yukako Uchida)

Appendix A: Code signing certificate

The code signing certificate attached to some samples are the following:

```
 $ openssl x509 -inform der -text -in mal.cer
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            3f:fc:eb:a8:3f:e0:0f:ef:97:f6:3c:d9:2e:77:eb:b9
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network, OU=Terms of use at https://www.verisign.com/r
        Validity
            Not Before: Aug  5 00:00:00 2011 GMT
            Not After : Aug  4 23:59:59 2012 GMT
        Subject: C=IT, ST=Italy, L=Milan, O=HT Srl, OU=Digital ID Class 3 - Microsoft Software Validation v2, CN
```

```
        Subject Public Key Info:
(Omitted)
```

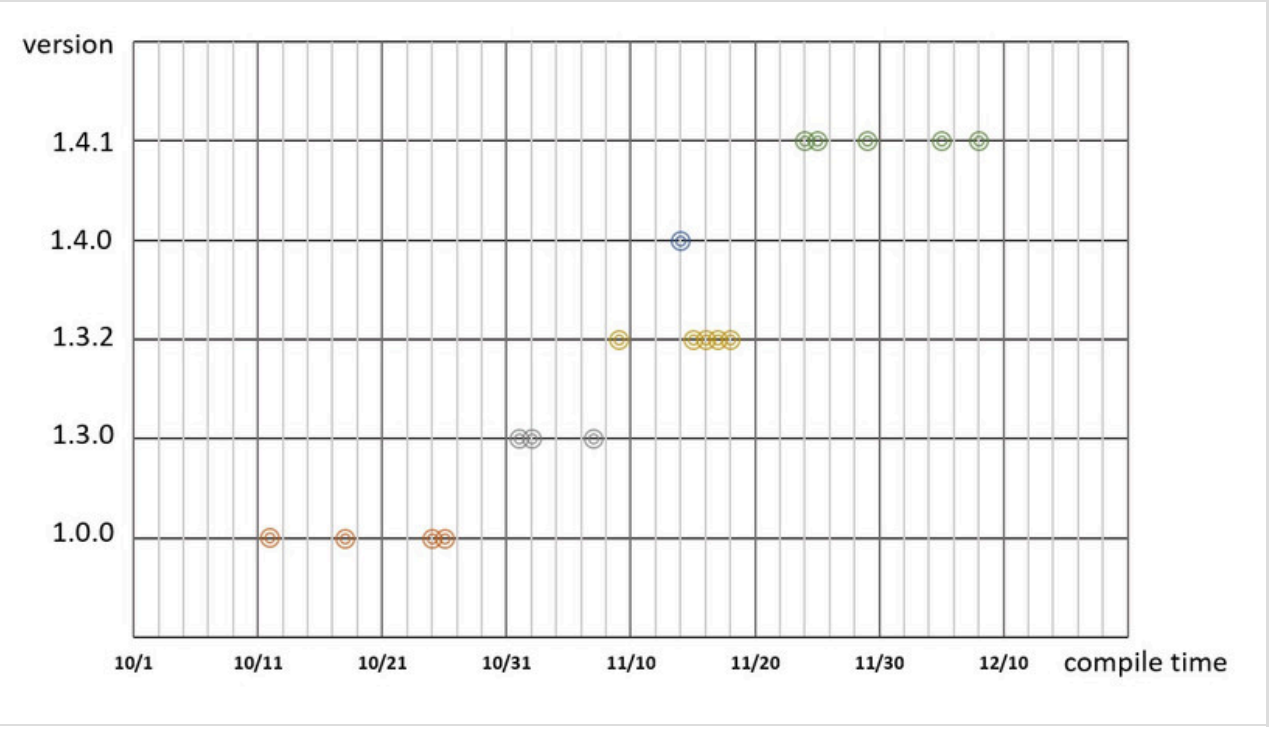
Figure 6: Code signing certificate


Fig6code_signing_certificate

Appendix B: ChChes version

The graph below shows the relation between the version numbers of the ChChes samples that JPCERT/CC has confirmed and the compile times obtained from their PE headers.

Figure 7: Compile time for each ChChes version



The lists below describe encrypted data contained in the first HTTP request and explanation of the values for each ChChes version.

Table 1: Sending format of each version

Version	**Format**
1.0.0	A<a>*<b>?3618468394?<c>?<d>*<f>
1.2.2	A<a>*<b>?3618468394?<c>?<d>*<f>
1.3.0	A<a>*<b>?3618468394?<c>?<d>*<f>
1.3.2	A<a>*<b>?3618468394?<c>?<d>*<g>
1.4.0	A<a>*<b>?3618468394?<c>?<d>*<g>

Version	Format
1.4.1	A<a>*<b>?3618468394?<c>?<d> (<e>)*<g>
1.6.4	A<a>*<b>*<h>?3618468394?<c>?<d> (<e>)*<g>

Table 2: Description of <a> to <h>

Letter	Data	Size	Details
<a>	Computer name	Variable	Capital alphanumeric characters
<b>	Process ID	Variable	Capital alphanumeric characters
<c>	Path of a temp folder	Variable	%TEMP% value
<d>	Malware version	Variable	e.g. 1.4.1
<e>	Screen resolution	Variable	e.g. 1024x768
<f>	explorer.exe version	Variable	e.g. 6.1.7601.17567
<g>	kernel32.dll version	Variable	e.g. 6.1.7601.17514
<h>	Part of MD5 value of SID	16 bytes	e.g. 0345cb0454ab14d7

Appendix C: SHA-256 Hash value of the samples

ChChes

- 5961861d2b9f50d05055814e6bfd1c6291b30719f8a4d02d4cf80c2e87753fa1
- ae6b45a92384f6e43672e617c53a44225e2944d66c1ffb074694526386074145
- 2c71eb5c781daa43047fa6e3d85d51a061aa1dfa41feb338e0d4139a6dfd6910
- 19aa5019f3c00211182b2a80dd9675721dac7cfb31d174436d3b8ec9f97d898b
- 316e89d866d5c710530c2103f183d86c31e9a90d55e2ebc2dda94f112f3bdb6d
- efa0b414a831cbf724d1c67808b7483dec22a981ae670947793d114048f88057
- e90064884190b14a6621c18d1f9719a37b9e5f98506e28ff0636438e3282098b
- 9a6692690c03ec33c758cb5648be1ed886ff039e6b72f1c43b23fbd9c342ce8c
- bc2f07066c624663b0a6f71cb965009d4d9b480213de51809cdc454ca55f1a91
- e6ecb146f469d243945ad8a5451ba1129c5b190f7d50c64580dbad4b8246f88e
- e88f5bf4be37e0dc90ba1a06a2d47faaeea9047fec07c17c2a76f9f7ab98acf0
- d26dae0d8e5c23ec35e8b9cf126cded45b8096fc07560ad1c06585357921eeed
- 2965c1b6ab9d1601752cb4aa26d64a444b0a535b1a190a70d5ce935be3f91699
- 312dc69dd6ea16842d6e58cd7fd98ba4d28eefeb4fd4c4d198fac4eee76f93c3
- 4ff6a97d06e2e843755be8697f3324be36e1ebeb280bb45724962ce4b6710297
- 45d804f35266b26bf63e3d616715fc593931e33aa07feba5ad6875609692efa2
- cb0c8681a407a76f8c0fd2512197aafad8120aa62e5c871c29d1fd2a102bc628
- 75ef6ea0265d2629c920a6a1c0d1dd91d3c0eda86445c7d67ebb9b30e35a2a9f

- 471b7edbd3b344d3e9f18fe61535de6077ea9fd8aa694221529a2ff86b06e856
- ae0dd5df608f581bbc075a88c48eedeb7ac566ff750e0a1baa7718379941db86
- 646f837a9a5efbbdde474411bb48977bff37abfefaa4d04f9fb2a05a23c6d543
- 3d5e3648653d74e2274bb531d1724a03c2c9941fdf14b8881143f0e34fe50f03
- 9fbd69da93fbe0e8f57df3161db0b932d01b6593da86222fabef2be31899156d
- 723983883fc336cb575875e4e3ff0f19bcf05a2250a44fb7c2395e564ad35d48
- f45b183ef9404166173185b75f2f49f26b2e44b8b81c7caf6b1fc430f373b50b

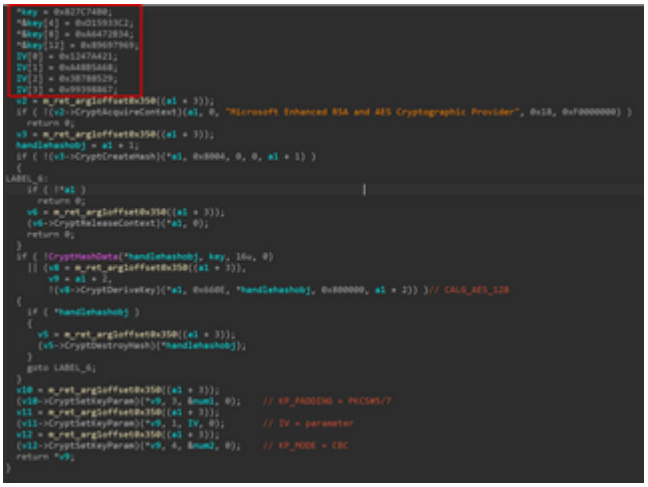
Appendix D: List of communication destination

- area.wthelpdesk.com
- dick.ccfchrist.com
- kawasaki.cloud-maste.com
- kawasaki.unhamj.com
- sakai.unhamj.com
- scorpion.poulsenv.com
- trout.belowto.com
- zebra.wthelpdesk.com
- hamiltion.catholicmmb.com
- gavin.ccfchrist.com

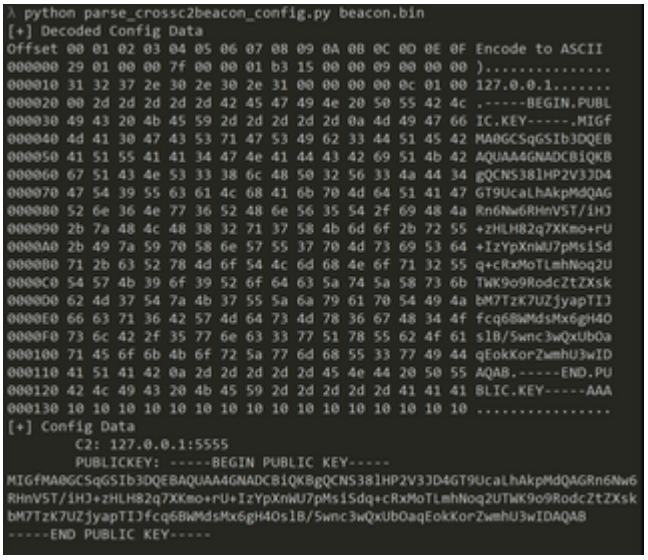
JPCERT/CC

Please use the below contact form for any inquiries about the article.

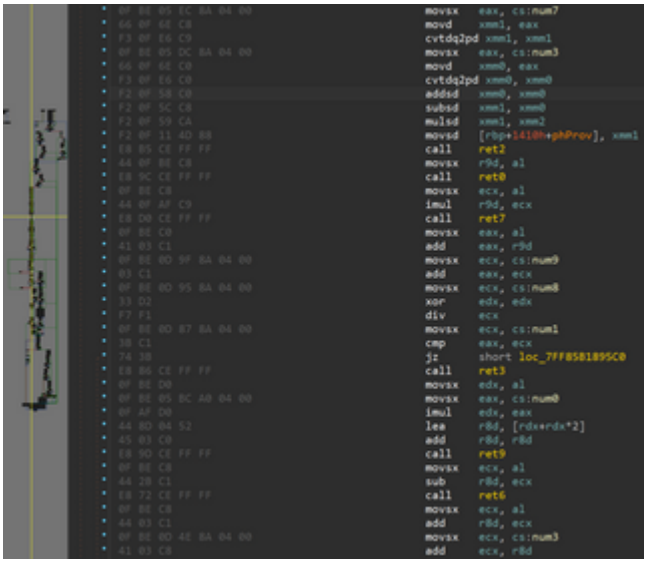
Related articles

- 

[Update on Attacks by Threat Group APT-C-60]



[CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks]

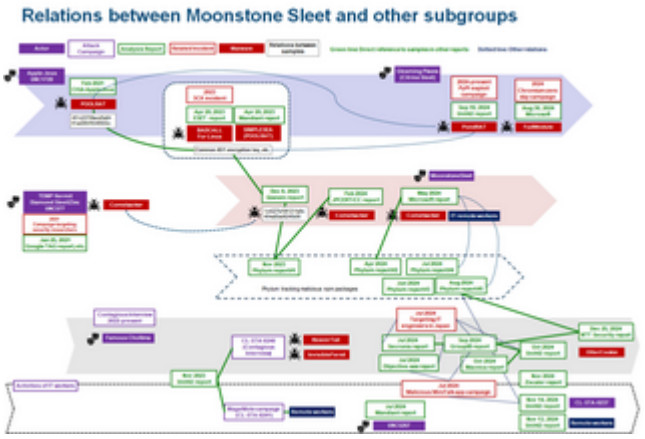


[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities]

```
__int64 __fastcall mal_decode(__int64 encbuf, int bufsize)
{
  __int64 j_1; // rax
  int i; // [rsp+18h] [rbp-Ch]

  if ( encbuf )
  {
    for ( i = 0; ; ++i )
    {
      j_1 = (unsigned int)i;
      if ( i >= bufsize )
        break;
      *(_BYTE *)(encbuf + i) ^= Key1to7[i % 7];
    }
  }
  return j_1;
}
```

- [DslogdRAT Malware Installed in Ivanti Connect Secure]



- [Tempted to Classifying APT Actors: Practical Challenges of Attribution in the Case of Lazarus's Subgroup]

Source: https://blogs.jpcert.or.jp/en/2017/02/chches-malware--93d6.html