



WHY CORRUPTED (?) SAMPLES IN RECENT APT? — CASE OF JAPAN AND TAIWAN

By Suguru Ishimaru

GREAT
Dec 2016

Introduction

Introduction

\$ whoami

Suguru_ISHIMARU

\$ whois suguru_ishimaru

Job_title: Researcher

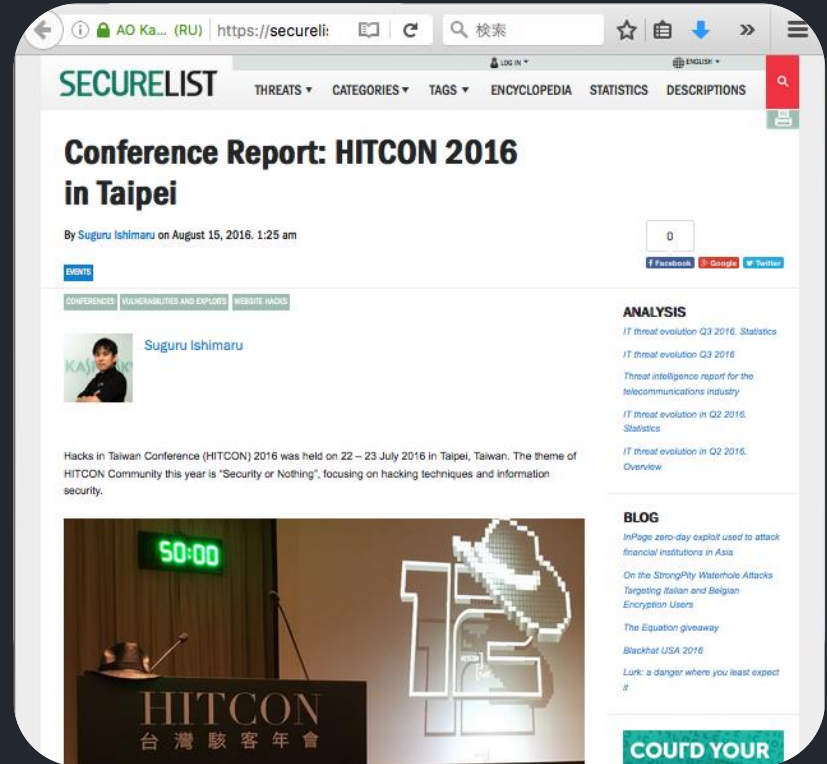
Department: Global Research Analysis Team

Organization: Kaspersky Labs

E-mail: suguru.ishimaru[at]kaspersky.com



My last blogpost was Conference Report:
HITCON 2016 in Taipei



Contents

Contents

\$ history | tail -n5

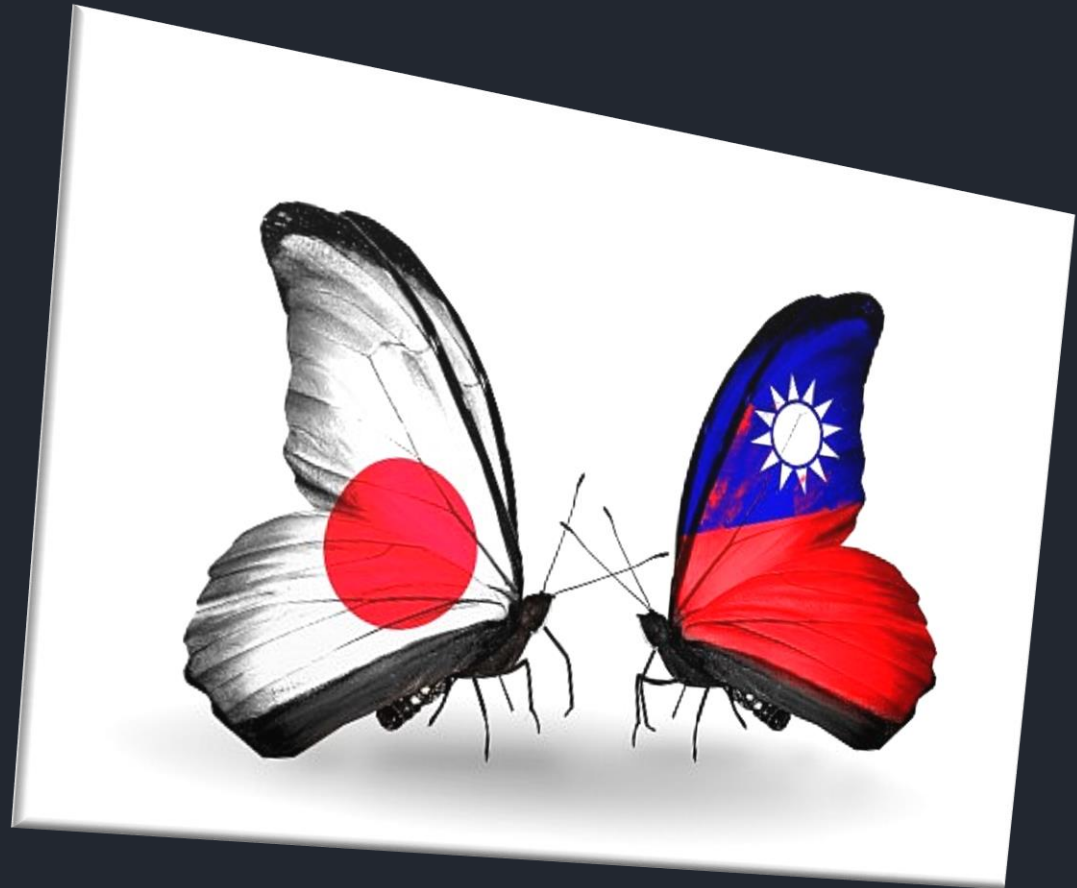
139 problem

140 motivation

141 emdivi

143 elirks

144 conclusion



Problem

Problem: A lot of targeted attacks



More than 40 APT

Problem: The biggest issue is...

Question: What is the biggest problem in APT seen from antivirus side?

Hard work

No sample

No detect

We collect mass spread samples. However, we could not get APT samples easily. Especially, second stage sample is extremely rare.

Problem: Corrupted samples

We found samples, sometimes they were corrupted. That means they are executable but crashing:

1. Memory dump
2. Unknown binary data
3. Broken data
4. Cured by Anti Virus
5. Quarantined file
6. Password encrypted archive without password

Problem: Why corrupted samples?

Question: Why corrupted samples in recent APT?

*I will tell you my answer
in conclusion*

Motivation

Motivation: What should we do?

Question: What should we do when we got corrupted malware in APT?

Just Ignore

Make AV signature

Deep Analysis

1. Checking really corrupted or not
2. Getting information of related others

Motivation: Two recent APT cases

Probably corrupted (?) samples have found in two recent APT.

Emdivi



Elirks



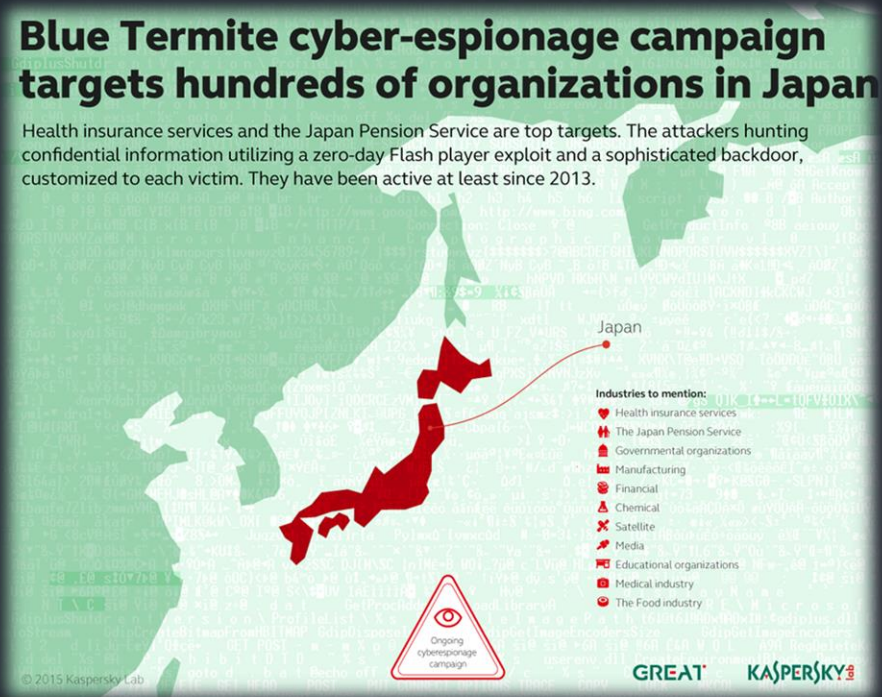
Emdivi

Emdivi: Overview

1. The Blue Termite APT campaign
2. Target region is Japan mainly
3. C2s on compromised legitimate sites
4. spear phishing email
5. drive-by download
6. Watering hole attacks
7. CVE-2014-7247
8. CVE-2015-5119

Blue Termite cyber-espionage campaign targets hundreds of organizations in Japan

Health insurance services and the Japan Pension Service are top targets. The attackers hunting confidential information utilizing a zero-day Flash player exploit and a sophisticated backdoor, customized to each victim. They have been active at least since 2013.



Emdivi: History

JUL 2011

Target to web site in Taiwan

NOV 2013

Oldest sample of Emdivi

NOV 2014

Operation CloudyOmega by Symantec

MAY 2015

Japan pension service Emdivi + PlugX

JUL 2015

Attacks of Flash Player 0day (CVE-2015-5119) by Trendmicro

AUG 2015

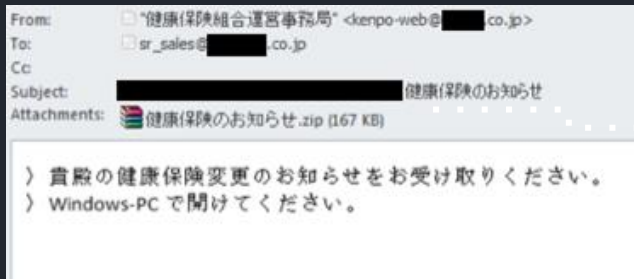
New activity of the Blue Termite APT by Kaspersky

MAY 2016

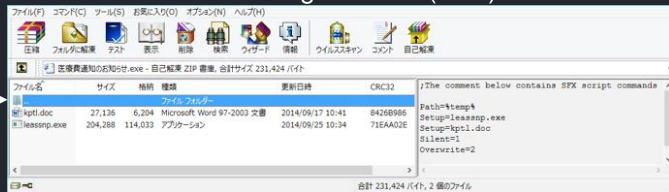
Security report about APT (Emdivi) by Macnica

Emdivi: Infection vector

spear phishing e-mail



self-extracting archives (SFX) file



drive by download



```
<param name=\"allowScriptAccess\" value=\"always\">
<embed src=\"movie.swf\" width=\"0\" height=\"0\"
iptAccess=\"always\" type=\"application/x-shockwave

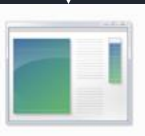
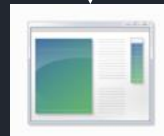
```

CVE-2015-5119

```
5A 57 53 11 59 7E 03 00 66 48 03 00 5D 00 00 80
00 00 3B FF FC 8E 19 FA DF E7 66 08 80 3D 3E 85
F5 75 6D 00 7E 61 4D 56 0F 22 60 75 3C 7A 2F BD
AD A7 3B 8A E7 8A A6 1B 8E FA 1F 3E 64 47 50 A0
41 84 81 30 0D A1 E0 74 CA 7A 8A FE 06 20 62 59
98 65 40 18 89 B4 5F 9C E3 F8 47 BF CA C8 42 0F
7B 58 94 0A 92 82 32 3B DD BB 38 32 10 A7 68 E3
05 FC 7E 9E AD CA 95 32 C6 66 5E C5 EB D4 D9 A7
48 85 9E E2 89 11 88 40 0
75 C1 99 2C 05 5B F8 C3 4
push ebx
call [ebp+writefile]
edx, [ebp+var_420]
push edx
call [ebp+closehandle]
push 0
mov eax, [ebp+var_430]; %temp%%fdrdrv.exe
eax
push eax
call [ebp+irExec]

```

emdivi t17



emdivi t20

watering hole attacks

Emdivi: Target

Regions:

- Japan
- Taiwan

Industries:

1. Government
2. Universities
3. Financial services
4. Energy
5. Food
6. Heavy industry
7. Chemical
8. News media
9. Health care
10. Insurance
11. Security researcher
12. Internet service provider

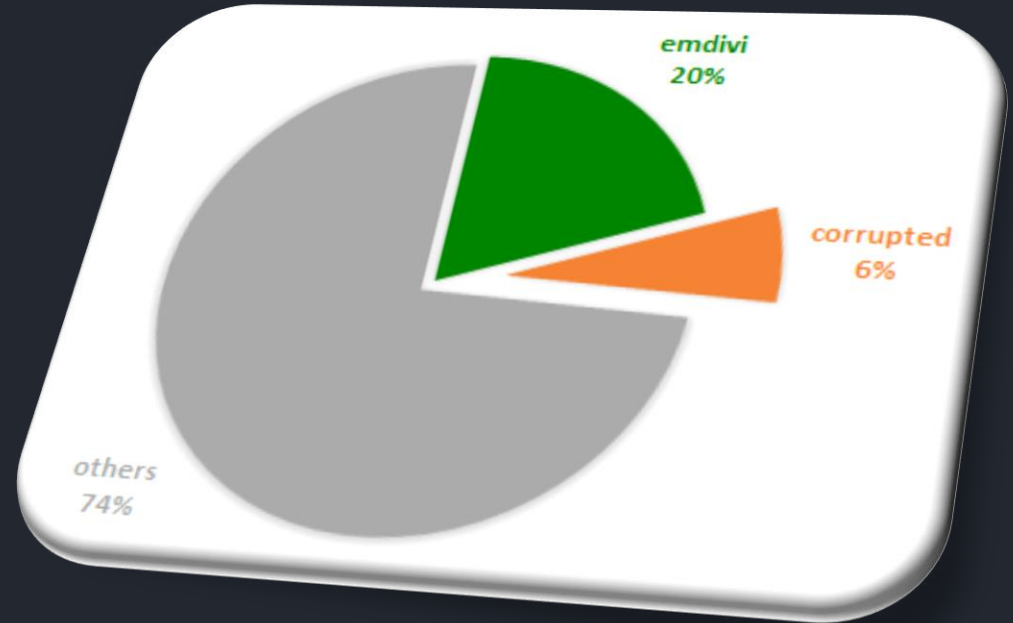
To create infrastructure

Japan Hosting provider

Taiwan web site

Emdivi: Corrupted (?) samples

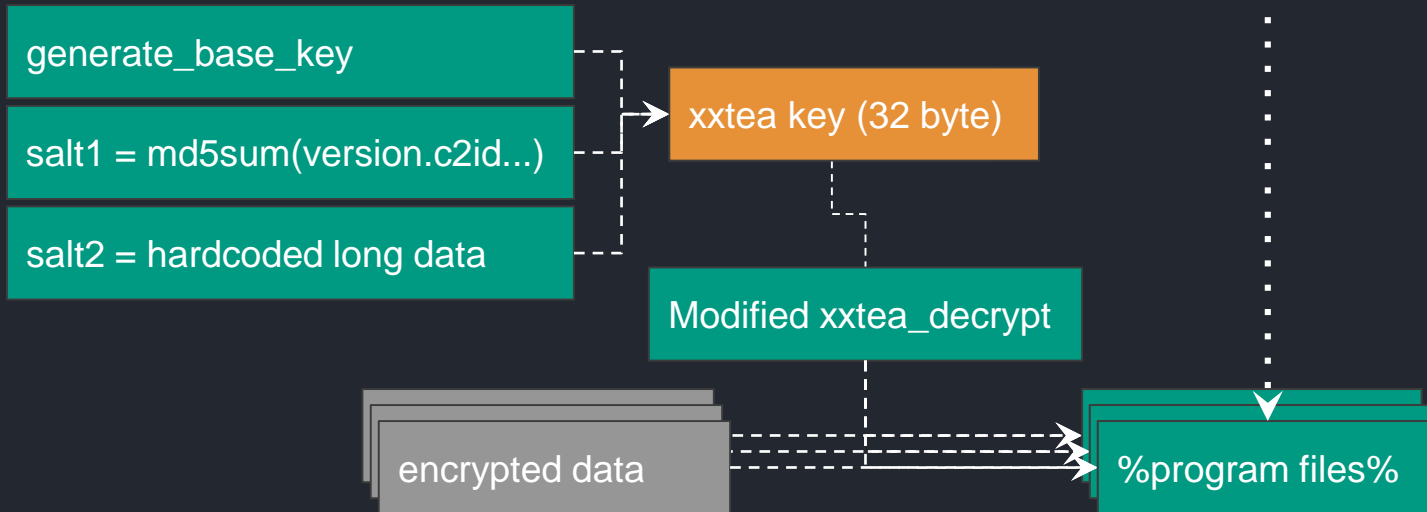
We collected more than 600 samples related to this attacks, around 25 percents were Emdivi samples. Among them, 6 percents did not work.



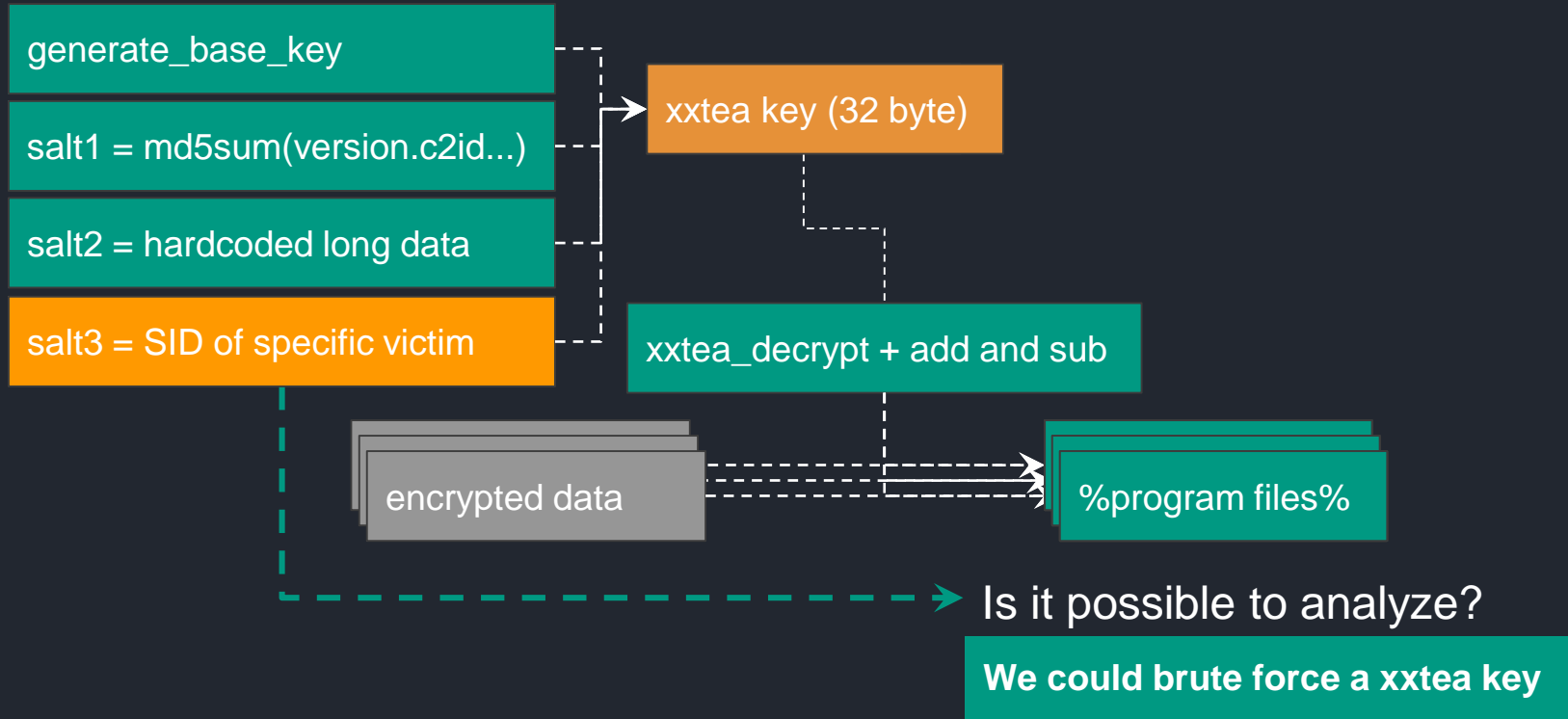
Emdivi: Important data was encrypted

Emdivi family stores encrypted important data:

C2, API name, strings for anti-analysis, value of mutexes, as well as the md5 checksum of backdoor commands and the internal proxy information



Emdivi: Corrupted (?) customized samples



Emdivi: Corrupted (?) customized samples

SECURELIST

THREATS ▾ CATEGORIES ▾ TAGS ▾ ENCYCLOPEDIA ▾

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 = '120.22.1.8750.2081.4208.0'
mov     ecx, [ebp+var_10]
push   ebx
mov     eax, esi
call   base64_enc         : base64(salt1)
xor     esi, 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
push   [ebp+var_4]
call   _J__free
lea    eax, [ebp+var_30]
push   offset salt2       : '5rva+L188PGeY504ZabY00TX3R8@un000051'...
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]
lea    dword ptr [eax]
push   [ebp+var_4]
call   strcat             : md5(base64(salt1)) + md5(salt2)
lea    esi, [ebp+var_30]
lea    byte ptr [ebp+var_4], 0
mov     [ebp+var_4], 0
call   free
lea    eax, [ebp+var_30]
push   eax
call   _getSID            : salt3 = 'S-1-5-21-XXXXXXXXXX-YYYYYYYY-ZZZZZZZZ'
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]
lea    byte ptr [ebp+var_4], 0
mov     [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.

We published the details as a blog in securelist.com

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*

→ Is it possible to analyze?

No

Emdivi: DEMO

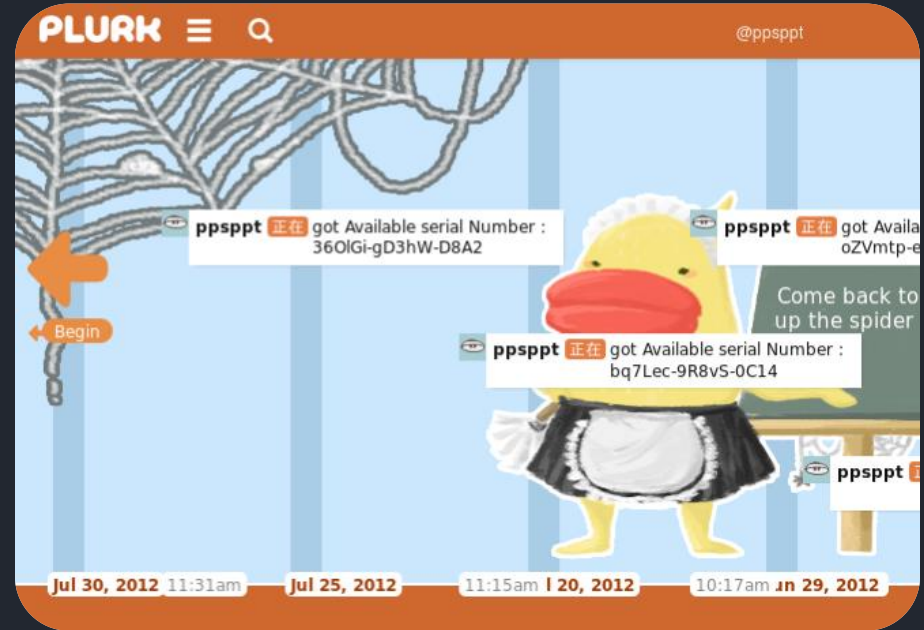
Emdivi t20 AES + SID

Elinks

Elirks: Overview

1. As known as PLURK
2. The Elirks APT campaign
3. Unique schema to connect real C2
4. Target Regions are Taiwan, Japan
5. Trojan dropper is fake folder icon
6. Decoys were sometimes airline e-ticket

This group uses several types of malware Elirks, Ymailer, Ymailer-mini and Micrass. This presentation is focusing Elirks



Elirks: History

MAR 2010

Oldest Elirks sample

JUL 2012

Chasing Advanced Persistent Threats (APT) by SecureWorks

JUL 2013

Hunting the Shadows by Fyodor Yarochkin, Pei Kan PK Tsung, Ming-Chang Jeremy Chiu, Ming-Wei Benson Wu

AUG 2015

Let's Play Hide and Seek In the Cloud by Ashley, Belinda

MAR 2016

Japan Tourist Bureau (JTB) Elirks + PlugX

JUN 2016

Tracking Elirks Variants in Japan: Similarities to Previous Attacks by paloalto

SEP 2016

MILE TEA: Cyber Espionage Campaign Targets Asia Pacific Businesses and Government Agencies by paloalto

OCT 2016

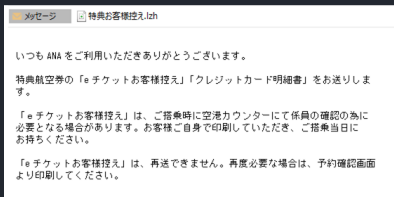
BLACKGEAR Espionage Campaign Evolves by trendmicro

NOV 2016

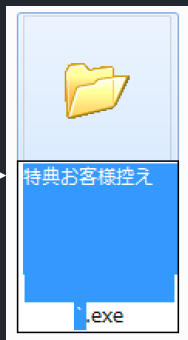
Japan Business Federation Elirks + PlugX

Elirks: Infection vector

spear phishing e-mail

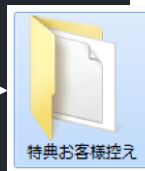


Trojan dropper spoofing folder icon



fake folder icon: 78 %

create dir, decoy and delete it self



Name	PID	CPU	I/O tot...	Private ...	Description
System Idle Process	0	98.84		0	
csrss.exe	352			1.93 MB	クライアント サーバー ラン...
wininit.exe	404			1.35 MB	Windows スタートアップア...
csrss.exe	412	0.05		10.19 MB	クライアント サーバー ラン...
winlogon.exe	468			2.26 MB	Windows ログオン アプリケ...
explorer.exe	2092	0.05		117.37 ...	エクスプローラー
vmtoolsd.exe	2188	0.03		13.74 MB	VMware Tools Core Service
ProcessHacker.exe	1916	0.25		8.96 MB	Process Hacker
svchost.exe	2672	0.17		2.13 MB	

Elirks malware

Elirks: Target

Regions:

- Taiwan
- Japan

Industries:

1. Government
2. Universities
3. Heavy industry
4. News media
5. Trading
6. Airline
7. Travel agency

e チケットお客様控え
ELECTRONIC TICKET ITINERARY/RECEIPT

ANA
eチケットお客様控え
ELECTRONIC TICKET ITINERARY/RECEIPT

このたびは日本航空「e チケット」をご利用いただきありがとうございます。

お客様
航空券
チケット
この e
ようお
Thank
You air
this e
Itinerar
retain t

お名前
NAME
航空券
TICKE
発券
TICKE

搭乗者名: ████████ KA MS
航空券番号: ████████
発行所: JAPAN - NH
発行日: 15MAR16
発行店舗コード: 16391351

Japan

旅程表 ITINERARY

都市/空港 CITY/AIRPORT	0-124 TOKYO (HANEDA)	運名 FLY/OP. NO.	運日 DATE	曜日 DAY	時刻 TIME	751 CLASS	運賃 FARE BASIS	予約状況 STATUS	手荷物 WEIGHT	乗換 TRANSFER
(1) TOKYO (HANEDA)	INT	██████	21MAY16	SAT	0925	X(Y)	XLBPOOH	OK	2PC	
到着 AIRRIVAL	北京 PEKING	運名 FLY/OP. NO.	運日 DATE	曜日 DAY	時刻 TIME	751 CLASS	運賃 FARE BASIS	予約状況 STATUS	手荷物 WEIGHT	乗換 TRANSFER
到着 AIRRIVAL	BEIJING	3	21MAY16	SAT	1235	ALL WOPON AIRWAYS				
出発 AIRFLIGHT	北京 PEKING	運名 FLY/OP. NO.	運日 DATE	曜日 DAY	時刻 TIME	751 CLASS	運賃 FARE BASIS	予約状況 STATUS	手荷物 WEIGHT	乗換 TRANSFER
出発 AIRFLIGHT	BEIJING	3	25MAY16	WED	0830	X(Y)	XLBPOOH	OK	2PC	
到着 AIRRIVAL	東京 HANEDA	運名 FLY/OP. NO.	運日 DATE	曜日 DAY	時刻 TIME	751 CLASS	運賃 FARE BASIS	予約状況 STATUS	手荷物 WEIGHT	乗換 TRANSFER
到着 AIRRIVAL	TOKYO (HANEDA)	INT	25MAY16	WED	1245	ALL WOPON AIRWAYS				

信用卡授權書
復興航空有限公司

中華航空
CHINA AIRLINES

付款同意書

費, 尊敬的旅客:
請再次填寫并確認您所訂購之下列機票:

商店搭客姓名: _____

持卡人 啟程班號: _____ 行程: _____

姓 訂位代號: _____

電話 本人, 姓名: _____, 護照號碼/證件號碼: _____

地址: _____ 電話: _____

信用卡
特此同意以下述之本人信用卡支付中華航空:
機票金額: _____
(票號: _____)

持卡人
信用卡卡別: _____ 信用卡卡號: _____
日後若接獲中華航空公司通知不接受該信用卡付款時, 本人同意以現金支付上述款項。

信用卡持卡人姓名(請用正楷): _____

Decoys of airline e-ticket

Elirks: Unique schema to connect real C2

The Elirks malware has unique schema to connect real C2. It connects blogpost of legitimate site getting encrypted real C2 information.

```
unicode 0, <code [redacted]og.jp/[redacted]20>,0
unicode 0, <blog [redacted].jp/[redacted]ya>,0
unicode 0, <sect [redacted]me.net>,0
unicode 0, <20140210>,0
dw 0
dw 0
dw 0
dw 0
dw 0
dw 0
dw 0
db 'CodeTrend Trying to pick between two',0
db ', their trends and make sure you pick',0
db 0
db 0
```

Malware config

A post in legitimate blog

Or search for a technology by name:
Wynne CodeTrend Trying to pick between two like oEWFYtYa, their trends and make sure you pick B36F.

Decrypt function

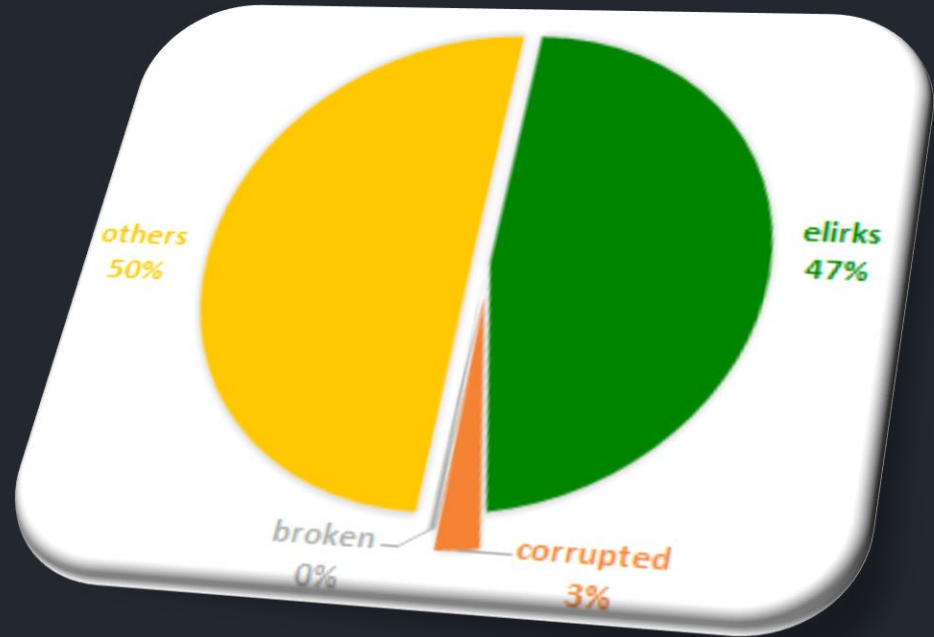


Elirks: Corrupted (?) samples

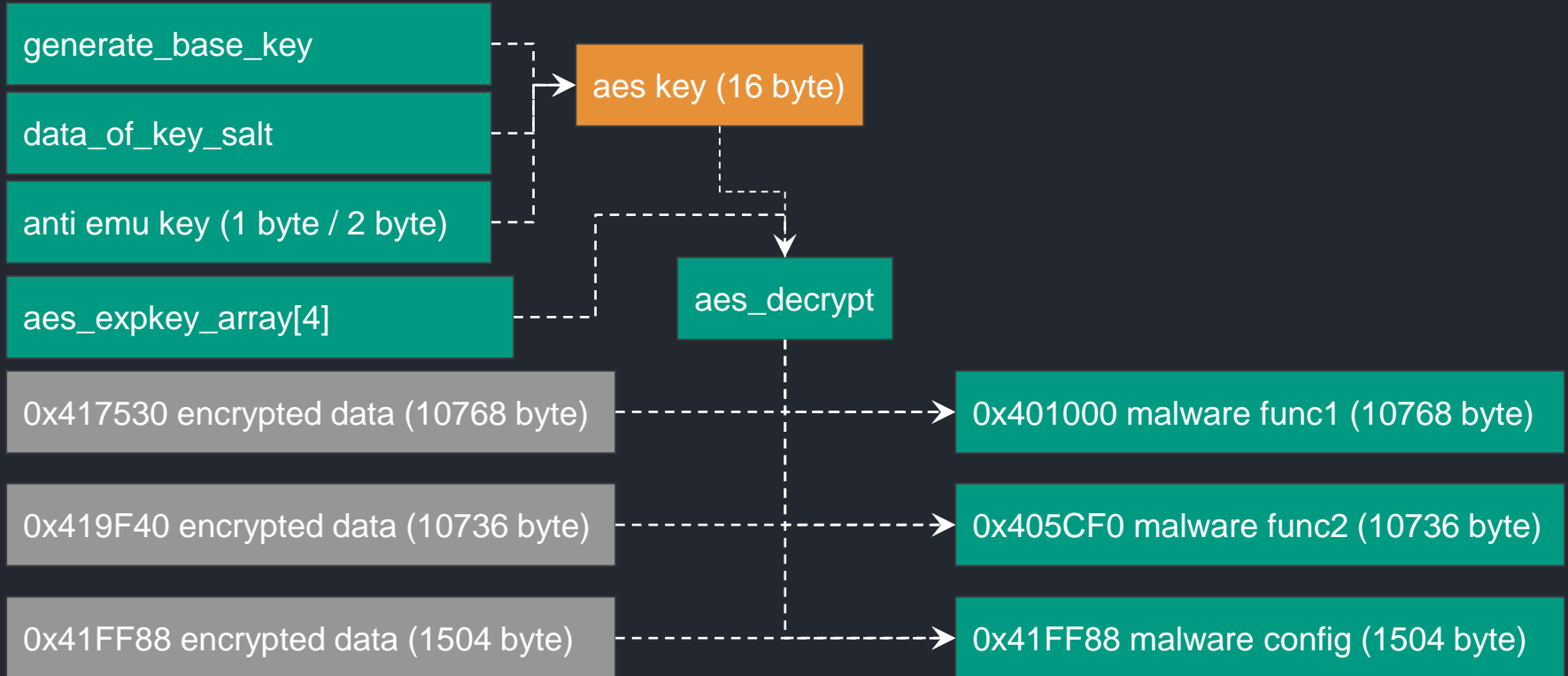
We collected more than 200 samples.

Among them, less than 3 percent were probably corrupted.

Then we confirmed why these samples does not work.



Elirks: Elirks has three encrypted data



Elirks: Decrypted Elirks

```
.text:00401000 83 out_decrypted_data1 db 83h ; DATA XREF: sub_401000
.text:00401001 EC db 05Ch
.text:00401002 30 db 30h; 0
.text:00401003 56 db 56h; V
.text:00401004 8B db 8Bh; t
.text:00401005 74 db 74h; c
.text:00401006 24 db 24h; $
.text:00401007 38 db 38h; 8
.text:00401008 8D db 8Dh
.text:00401009 54 db 54h; T
.text:0040100A 24 db 24h; $
.text:0040100B 0C db 0Ch
.text:0040100C 66
.text:0040100D 8E
.text:0040100E 4B
```

0x401000 unknown data (10768 byte)

```
.text:00405CF8 83 out_decrypted_data2 db 83h ; CODE XREF: .text:00405CF0
.text:00405CF0 44 db 0C4h; c
.text:00405CF1 C4 db 4
.text:00405CF2 04 db 66h; f
.text:00405CF3 66 db 33h; 3
.text:00405CF4 33 db 0C0h; h
.text:00405CF5 C0 db 5Fh; ^
.text:00405CF6 5F db 5Eh; ^
.text:00405CF7 5E db 5Dh; ^
.text:00405CF8 3D db 5Bh; i
.text:00405CF9 5B db 83h; l
.text:00405CFA 83
.text:00405CFB C4
.text:00405CFC 48
.text:00405CFD C3
```

0x405cf0 unknown data (10736 byte)

```
.data:0041FF88 FB enc_data3 db 0FBh; c ; DATA XREF: sub_405250+1657
.data:0041FF89 DF db 0DFh; c ; sub_405250+1657
.data:0041FF8A 05 db 5 ; malware config
.data:0041FF8B CB db 0CBh; c
.data:0041FF8C 0D db 0Dh
.data:0041FF8D 6A db 6Ah; j
.data:0041FF8E 3C db 3Ch; <
.data:0041FF8F 7D db 7Dh; }
.data:0041FF90 E6 db 0E6h; c
.data:0041FF91 72 db 72h; r
.data:0041FF92 64 db 64h; d
.data:0041FF93 68 db 68h; h
.data:0041FF94 A9 db 0A9h; c
.data:0041FF95 F4 db 0F4h; c
.data:0041FF96 50 db 50h; P
.data:0041FF97 77 db 77h; w
.data:0041FF98 10 db 10h; h
.data:0041FF99 86 db 86h; c
.data:0041FF9A 62
.data:0041FF9B 48
.data:0041FF9C B4
.data:0041FF9D 8A
```

0x41FF88 encrypted data (1504 byte)

```
.text:00401000 sub_401000 proc near ; CODE XREF: ma_main_tread+3751p
.text:00401000 ; DATA XREF: sub_405250+871e
.text:00401000
.text:00401000 hKey = dword ptr -14h
.text:00401000 hObject = dword ptr -10h
.text:00401000 Type = dword ptr -0Ch
.text:00401000 cbData = dword ptr -8
.text:00401000 phkResult = dword ptr -4
.text:00401000
.text:00401000 55 push ebp
.text:00401001 8B EC mov ebp, esp
.text:00401003 83 EC 18 sub esp, 18h
.text:0041006 36
.text:00401007 37
.text:00401008 C7 45 F8 00 01 00 00
.text:00401009 C7 45 EC 01 00 00 80
```

0x401000 malware func1 (10768 byte)

```
.text:00405CF0 sub_405CF0 proc near ; CODE XREF: .text:00413FC6j
.text:00405CF0 ; DATA XREF: sub_405250+1147e
.text:00405CF0 57 push edi
.text:00405CF1 8D 86 5C 03 00 00 lea eax, [esi+35Ch]
.text:00405CF7 50 push eax ; lpCriticalSection
.text:00405CF8 FF 15 78 40 41 00 call ds:DeleteCriticalSection
.text:00405CFE 8B 86 28 03 00 00 mov eax, [esi+328h]
.text:00405D04 8B 3D EC 41 41 00 mov edi, ds:GlobalFree
.text:00405D0A 85 C0 test eax, eax
.text:00405D0C 74 0D jz short loc_405D1B
.text:00405D0E 30
.text:00405D0F FF D7 call edi; GlobalFree
.text:00405D11 C7 86 28 03 00 00 00+
.text:00405D1B
```

0x405CF0 malware func2 (10736 byte)

```
.data:0041FF88 ; const WCHAR malconf
.data:0041FF89 malconf: ; DATA XREF: sub_405250
.data:0041FF8A ; sub_405250+1657To
.data:0041FF8B 63 00 6F 00 64 00 65 00+ unicode 0, <code [redacted]og.jp/[redacted]>,0
.data:0041FF8C aBloc [redacted].jp: unicode 0, <blog [redacted].jp/[redacted]ya>,0
.data:0041FF8D aSect [redacted]: unicode 0, <sect [redacted]me.net>,0
.data:0041FF8E a20140210: unicode 0, <20140210>,0
.data:00420024 32 00 30 00 31 00 34 00+ dw 0
.data:00420036 00 00 dw 0
.data:00420038 00 00 dw 0
.data:0042003A 00 00 dw 0
.data:0042003C 00 00 dw 0
.data:0042003E 00 00 dw 0
.data:00420040 00 00 dw 0
.data:00420042 00 00 dw 0
.data:00420044 43 6F 64 65 54 72 65 6E+aCodetrendTryin 'CodeTrend Trying to pick between two',0
.data:00420049 00 00 dw 0
.data:0042006A 2C 20 74 68 65 69 72 20+aTheirTrendsAnd db ', their trends and make sure you pick',0
.data:00420090 00
.data:00420091 00
.data:00420092 1F
.data:00420093 B2
```

0x41FF88 malware config (1504 byte)

Elirks: Corrupted (?) samples

A corrupted (?) sample does not decrypt malware config.

That means does not work and can not analyze.

```
.data:0041CE28 0C          unk_41CE28      db  0Ch
.data:0041CE28
.data:0041CE29 AA          db  0AAh ;  x
.data:0041CE2A BD          db  0BDh ;  7
.data:0041CE2B 34          db  34h ;  4
.data:0041CE2C C2          db  0C2h ;  ツ
.data:0041CE2D 46          db  46h ;  F
.data:0041CE2E CD          db  0CDh ;  ^
.data:0041CE2F 27          db  27h ;  '
.data:0041CE30 4D          db  4Dh ;  M
.data:0041CE31 82          db  82h ;
.data:0041CE32 FC          db  0FCh ;
.data:0041CE33 BA          db  0BAh ;  コ
.data:0041CE34 B8          db  0B8h ;  夕
.data:0041CE35 36          db  36h ;  6
.data:0041CE36 49          db  49h ;  I
.data:0041CE37 69          db  69h ;  i
.data:0041CE38 D4          db  D4h ;
.data:0041CE39 6E          db  6Eh ;
```

0x41CE28 encrypted data (1504 byte)

```
.data:0041CE28 D1          conf_41CE28    db  0D1h ;  4
.data:0041CE28
.data:0041CE29 59          db  59h ;  Y
.data:0041CE2A 83          db  83h ;
.data:0041CE2B 9D          db  9Dh ;
.data:0041CE2C 7E          db  7Eh ;  ~
.data:0041CE2D A0          db  0A0h ;
.data:0041CE2E 09          db  9
.data:0041CE2F 44          db  44h ;  D
.data:0041CE30 14          db  14h ;
.data:0041CE31 D2          db  0D2h ;  2
.data:0041CE32 36          db  36h ;  6
.data:0041CE33 B1          db  0B1h ;  7
.data:0041CE34 E2          db  0E2h ;
.data:0041CE35 36          db  36h ;  6
.data:0041CE36 C6          db  0C6h ;  二
.data:0041CE37 87          db  87h ;
.data:0041CE38 41          db  41h ;
.data:0041CE39 B6          db  B6h ;
```

0x41CE28 malware config (1504 byte)

Elirks: DEMO

Elirks probably corrupted (?) sample

Elirks: ~~Corrupted (?)~~ customized samples

It was customized sample for specific victims

Compare specific dir and current dir to extract 4 bytes xor key as part of generate AES key

aes key (16 byte)

```

.data:0041CE28 0C             unk_41CE28     db  0Ch
.data:0041CE28             db  0AAh ; ɚ
.data:0041CE29 AA             db  0BDh ; ʀ
.data:0041CE2A BD             db  34h ; 4
.data:0041CE2B 34             db  0C2h ; ツ
.data:0041CE2C C2             db  46h ; F
.data:0041CE2D 46             db  0CDh ; ^
.data:0041CE2E CD             db  27h ; '
.data:0041CE2F 27             db  4Dh ; M
.data:0041CE30 4D             db  82h ;
.data:0041CE31 82             db  0FCh ;
.data:0041CE32 FC             db  0BAh ; コ
.data:0041CE33 BA             db  0B8h ; ク
.data:0041CE34 B8             db  36h ; 6
.data:0041CE35 36             db  49h ; I
.data:0041CE36 49             db  69h ; i
.data:0041CE37 69             db  00h ;
.data:0041CE38 00             db  00h ;
.data:0041CE39 00             db  00h ;

```

0x41CE28 encrypted data (1504 byte)

```

.data:0041CE28 ; const WCHAR String
.data:0041CE28 String:                               ; DATA XREF: .text:00404
.data:0041CE28                               ; .text:00404516↑o ...
.data:0041CE28                               unicode 0, <210.209.██████>,0
.data:0041CE46 dw 0
.data:0041CE48 dw 0
.data:0041CE4A dw 0
.data:0041CE4C aSoftwareMicros:
.data:0041CE4C                               unicode 0, <SOFTWARE#Microsoft#Windows#CurrentVe
.data:0041CE4C                               unicode 0, <ORE.EXE>,0
.data:0041CECE aIexplore_exe_l:
.data:0041CECE                               unicode 0, <¥IEXPLORE.EXE.Ink>,0
.data:0041CECE dw 0
.data:0041CEFE dw 0
.data:0041CF00 dw 0
.data:0041CF02 dw 0
.data:0041CF04 dw 0
.data:0041CF06 dw 0
.data:0041CF08 aWinTheCompetio db 'win the competition award',0
.data:0041CF0A db 0
.data:0041CF0C db 0
.data:0041CF0E aWellKnownForTh db ', well known for the series of',0
.data:0041CF10 db 0
.data:0041CF12 db 0
.data:0041CF14 db 0
.data:0041CF16 db 0
.data:0041CF18 db 0

```

0x41CE28 malware config (1504 byte)

Conclusion

Conclusion: Answer of my title's question

Question: Why corrupted (?) samples in recent APT?

*It's not corrupted.
The attacker developed
customized malware*

*When you find corrupted sample,
It might to be chance of analysis very interesting APT malware*

Conclusion: Whitelist approach in APT

Common malware should work in any environment.

APT malware have to work in specific environment.

This approach and introduced new techniques are very simple ,However it works effectively.



Thank You

[suguru.ishimaru\[at\]kaspersky.com](mailto:suguru.ishimaru[at]kaspersky.com)