

RICH HEADERS: LEVERAGING THE MYSTERIOUS ARTEFACT OF THE PE FORMAT

Peter Kálnai
Malware Researcher

peter.kalnai@eset.cz

Michal Poslušný
Malware Researcher

michal.poslusny@eset.cz



@ESETresearch





Description of Rich Headers (RH)



Tooling



Lessons learned

Implemented since VS
97 SP3, Microsoft has
never announced,
documented or allowed
to opt out this feature.

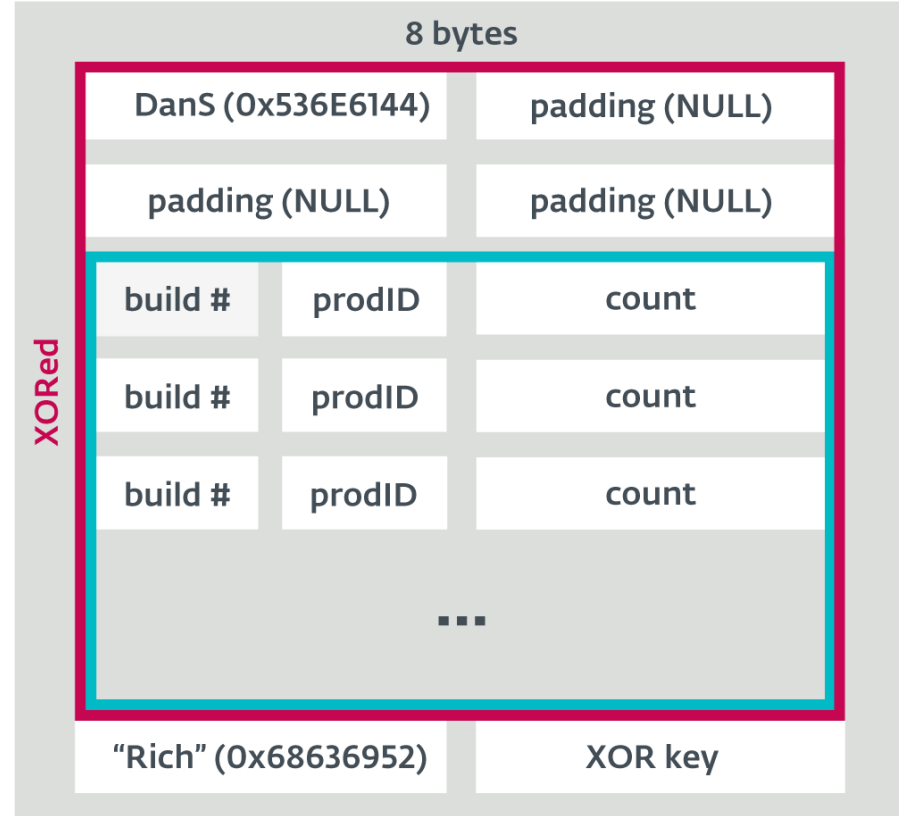
Description of Rich Headers

Timeline Highlights

- 1997: Visual Studio 97 SP3 introduced RH
- 2004: :lifewire / ikx, : things they didn't tell you about ms link and the pe header :
- 2010: Pistelli, *Microsoft's Rich Signature (undocumented)*
- 2017: Webster et al. *Finding the Needle: A Study of the PE32 Rich Header and Respective Malware Triage*
- 2018: GReAT, *The devil's in the Rich header*
- 2018: [K.-P.] *Lazarus Group*, VB2018 Montreal
- 2019: (July) Maksim Dubyk, *Leveraging the PE Rich Header for Static Malware Detection and Linking*, SANS
- 2019: (October) Todd Plantenga, Ben Wilson, *Fingerprinting Binaries Using Rich Headers: Tales from Our Analysis*, Fireeye Cyber Defense Summit 2019

RH Structure

- Overlooked by the security industry (and us) for many years
- Contains valuable information when interpreted correctly



RH Basic Facts

- Between IMAGE_DOS_HEADER and IMAGE_NT_HEADERS
- ['DanS' .. 'Rich'], 4-byte XOR key → data between

```
000: 4D 5A 90 00-03 00 00 00-04 00 00 00-FF FF 00 00 MZ
010: B8 00 00 00-00 00 00 00-40 00 00 00-00 00 00 00
020: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
030: 00 00 00 00-00 00 00 00-00 00 00 00-20 01 00 00
040: 0E 1F BA 0E-00 B4 09 CD-21 B8 01 4C-CD 21 54 68
050: 69 73 20 70-72 6F 67 72-61 6D 20 63-61 6E 6E 6F
060: 74 20 62 65-20 72 75 6E-20 69 6E 20-44 4F 53 20
070: 6D 6E 64 65-20 72 75 6E-20 69 6E 20-44 4F 53 20
080: 82 7A C2 E4-C6 1B AC B7-C6 1B AC B7-C6 1B AC B7
090: A3 7D AF B6-CD 1B AC B7-A3 7D A8 B6-D2 1B AC B7
0A0: A3 7D A9 B6-6C 1B AC B7-72 71 AF B6-CF 1B AC B7
0B0: 72 71 A9 B6-44 1B AC B7-72 71 A8 B6-E5 1B AC B7
0C0: 51 45 AD B6-C4 1B AC B7-A3 7D AD B6-CC 1B AC B7
0D0: 73 85 71 B7-C5 1B AC B7-C6 1B AD B7-2C 1B AC B7
0E0: B2 70 A5 B6-D6 1B AC B7-B2 70 53 B7-C7 1B AC B7
0F0: C6 1B 3B B7-C7 1B AC B7-B2 70 AE B6-C7 1B AC B7
100: 52 69 63 68-C6 1B AC B7-00 00 00 00-00 00 00 00
110: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
120: 50 45 00 00-64 86 07 00-08 FB F8 5C-00 00 00 00 PE
```



```
00-00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00-00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01-14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01-09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01-23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01-0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00-EA 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00-01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01-01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00-00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

RH Structure

- Microsoft Windows 2000 source code leaked

```
prodidCvtomf511    = 0x0011,    // LINK 5.11 OMf to COFF conversion
prodidMasm614      = 0x0012,    // MASM 6.14 (MMX2 support)
prodidLinker512    = 0x0013,    // LINK 5.12
prodidCvtomf512    = 0x0014,    // LINK 5.12 OMf to COFF conversion
```

-Product IDs

```
#define DwProdidFromProdidWBuild(prodid, wBuild) (((((unsigned long) (prodid) << 16) | (wBuild))
#define ProdidFromDwProdid(dwProdid)            ((PRODIG) ((dwProdid) >> 16))
#define WBuildFromDwProdid(dwProdid)            ((dwProdid) & 0xFFFF)
```

// Define the image data format

```
typedef struct PRODIGITEM {
    unsigned long    dwProdid;        // Product identity
    unsigned long    dwCount;        // Count of objects built with that product
} PRODIGITEM;
```

'DanS'

```
enum {
    tagEndID    = 0x536e0144,
    tagBegID    = 0x68636952,
};
/*
```

-'Rich'

Normally, the DOS header and PE header are contiguous. We place some data in between them if we find at least one tagged object file.

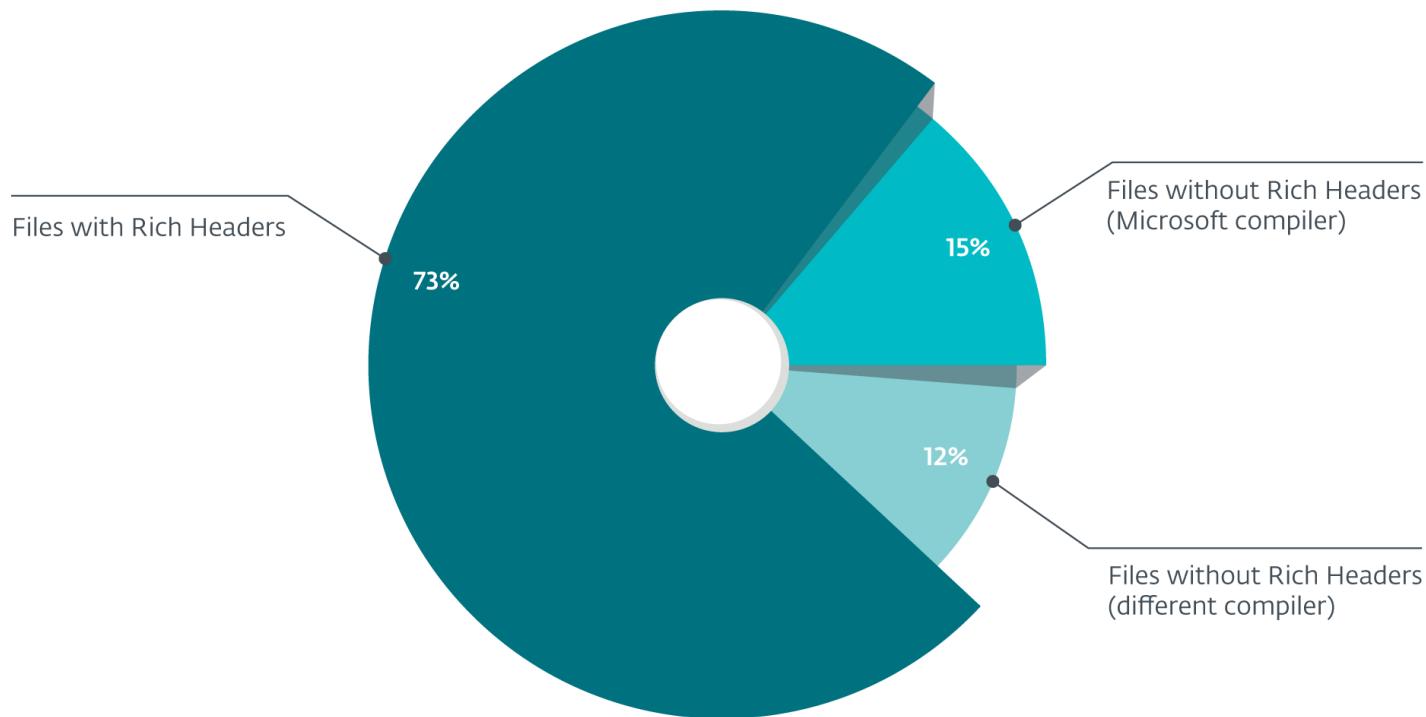
```
struct {
    IMAGE_DOS_HEADER dosHeader;
    BYTE             rgbDosStub[N];    // MS-DOS stub
    PRODIGITEM       { tagEndID, 0 };  // start of tallies (Masked with dwMask)
    PRODIGITEM       { 0, 0 };         // end of tallies (Masked with dwMask)
    PRODIGITEM       rgbproditem[];    // variable sized (Masked with dwMask)
    PRODIGITEM       { tagBegID, dwMask }; // end of tallies
    PRODIGITEM       { 0, 0 };         // variable sized
    IMAGE_PE_HEADER  peHeader;
};
```

XOR key

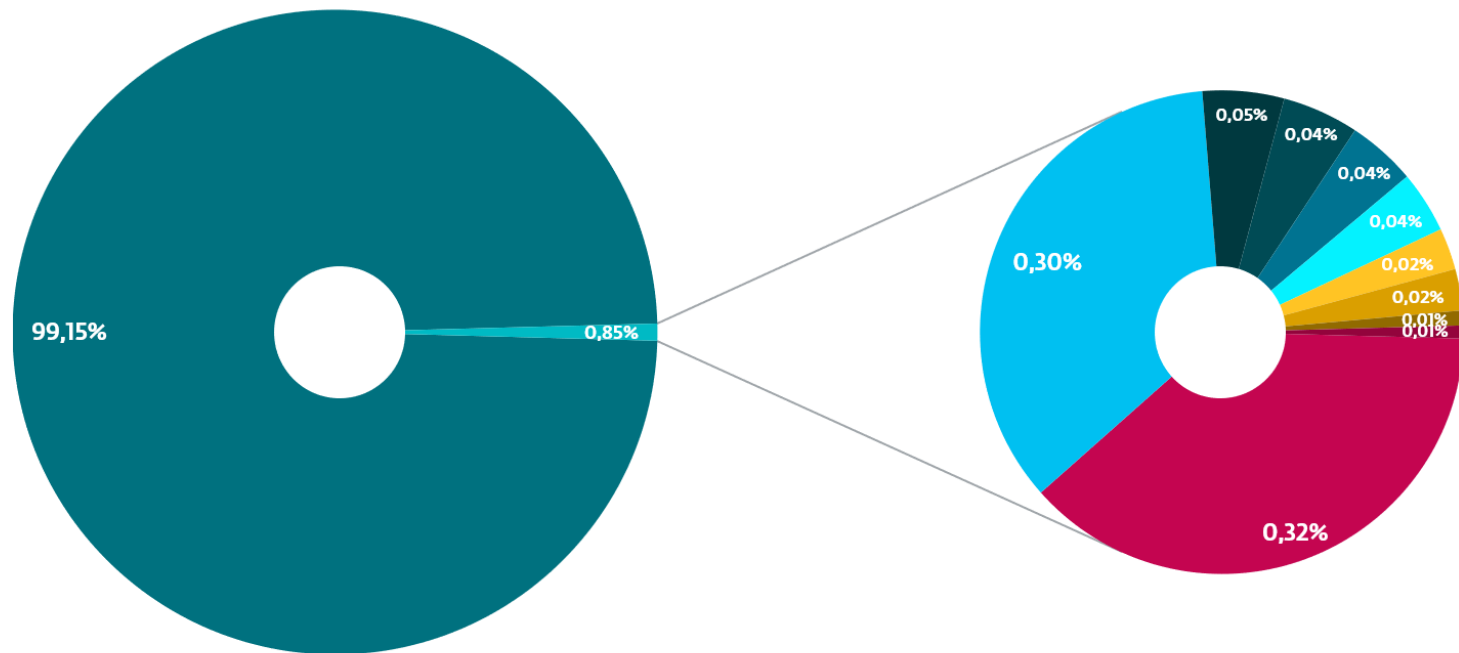
- XOR key generation VS 2019:

```
dosStubSize = *(v3 + 164);
i = 0;
checkSum = dosStubSize;
if ( dosStubSize )
{
    do
    {
        checkSum += __ROL4__(data->dosStub[i], i);
        ++i;
    }
    while ( i < dosStubSize );
    v5 = v25;
}
for ( richHeaderList = v5; richHeaderList; checkSum += __ROL4__(dwProdid, dwCount) )
{
    dwCount = richHeaderList->dwCount;
    dwProdid = richHeaderList->dwProdid;
    richHeaderList = richHeaderList->next;
}
```

RH Occurrence in the malicious set



RH Offsets



■ 0x80 ■ 0x200 ■ 0x40 ■ 0xD8 ■ 0x48 ■ 0xF0 ■ 0xA0 ■ 0x600 ■ 0xE0 ■ 0x60 ■ 0x68

RH Levels of similarity

- Identical Rich Headers
- Identical XOR Keys
- Identical Unsorted ProdlDs + builds
- Identical Sorted ProdlDs + builds
- Conjunctions of various ProdlDs

RH Level 0 - Identical RH

- Themida-, Enigma- & VMProtect-ed samples with the unprotected one, e.g. PredatorStealer, Win/NukeSped
- Fake/Copied RH, e.g. Olympic Destroyer, explorer.exe
- Clusters of known File Formats, e.g. WinRAR SFX, Autolt,

RH Level 1 - Identical XOR key

- Samples with Identical RH have obviously the same XOR key
- Malware packers, e.g. 0x8F44CEBF, 0xAEb29219, 0x8A17753B, 0xD4F1AE19, 0x887F83A7 (the complete RH varied!)

Visual Basic 6.0	0x886973F3, 0x8869808D, 0x88AA42CF, 0x88AA2A9D, 0x89A99A19, 0x88CECC0B, 0x8897EBCB, 0xAC72CCFA, 0x1AAAA993, 0xD05FECFB, 0x183A2CFD,
NSIS installer	0xD28650E9, 0x38BF1A05, 0x6A2AD175, 0xD246D0E9, 0x371742A2, 0xAB930178, 0x69EAD975, 0x69EB1175, 0xFB2414A1, 0xFB240DA1
MoleBox Ultra v4	0x8CABE24D
WinRar SFX	0xC47CACAA, 0xFDAFBB1F, 0xD3254748, 0x557B8C97, 0x8DEFA739, 0x723F06DE, 0x16614BC7
Microsoft CAB File	0x43FACBB6
Autoit	0xBEAFE369, 0xC1FC1252, 0xCDA605B9, 0xA9CBC717, 0x8FEDAD28, 0x273B0B7D, 0xECFA7F86

RH Level 2 - Unsorted (ProdID, build)

- Samples with Identical XOR key have obviously the same unsorted (ProdID, build) pairs

Win64/CoinMiner.DN (0x105E60A5B349F444):

	Videolan.exe	Update-19.1.10.exe
prodidImport0	323	328

Win32/Pterodo (0x745E73E5045EE80E):

	iPxxP4.dll	Y9s9Ow.dll
prodidMasm1210 (b40116)	9	15
prodidUtc1810_CPP (b40116)	119	159
prodidUtc1810_C (b40116)	24	29
prodidMasm1400 (b24123)	19	24
prodidUtc1900_CPP	23	51
prodidImport0	84	96

RH Level 3 - Sorted (ProdID, build)

- Samples with the same unsorted (ProdID, build) pairs have obviously the same sorted (ProdID, build) pairs

Win64/NukeSped.Z (0x1108557B575DE91F)

rds.dll (2016-10-24 8:32:11)		res.dll (2016-10-24 8:32:11)	
prodidMasm1000 (b40219)	15	prodidUtc1600_C (b40219)	94
prodidUtc1600_C (b40219)	100	prodidMasm1000 (b40219)	9
prodidImport0 (0)	130	prodidImport0 (0)	131

Win/GreyEnergy

Zlib_x86.dll		Zlib_x64.dll	
prodidMasm1000 (b40219)	17	prodidUtc1600_C (b40219)	107
prodidUtc1600_C (b40219)	110	prodidMasm1000 (b40219)	9
prodidImport0	87-88	prodidImport0	88-89
prodidUtc1600_LTCG_CPP (b40219)	22	prodidUtc1600_LTCG_CPP (b40219)	22-23

RH Anomalies

- Invalid RH values
- Duplicate Rich Header values
- Invalid XOR key checksum
- Rich Header offsets
- PE Optional Header linker information mismatch
- Imports & Resources mismatch

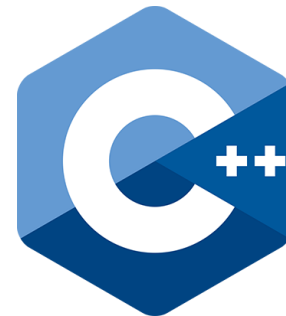
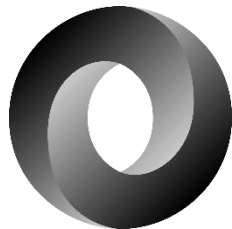
A small team of developers or even a single dedicated person could build the required infrastructure.

Tooling

Technologies



{fmt}



file	
file_id	BIGINT
sha1	BINARY(20)
image_size	INT
bitness	TINYINT
is_dll	BIT
pe_timestamp	DATETIME
linker_major_version	TINYINT
linker_minor_version	TINYINT
import_count	INT
import_hash	BINARY(16)
rich_header_xor_key	INT
rich_header_real_checksum	INT
rich_header_offset	SMALLINT
rich_header_size	SMALLINT
rich_header_crc	BIGINT
rich_header_prodid_build_crc	BIGINT
rich_header_prodid_build_sorted_crc	BIGINT
rich_header_prodid_count_crc	BIGINT
rich_header_record_count	SMALLINT
rich_header_object_count	INT
time_added	DATETIME

rich_record	
record_id	BIGINT
file_id	BIGINT
record_index	TINYINT
rich_object_id	SMALLINT
rich_object_id_bad	SMALLINT
rich_object_version	SMALLINT
rich_object_count	INT
rich_object_crc	BIGINT
CONSTRAINT	checkObjectNull

rich_object_name	
rich_object_id	SMALLINT
rich_object_name	VARCHAR(100)
rich_compiler_name	VARCHAR(100)

file_path	
file_path_id	BIGINT
file_id	BIGINT
path	VARCHAR(512)

Database (Backend)



PERichMiner (Backend)

- Parses PE files and stores the data into RH database
- High performance (C++, low-level API)
- Supports multiple operation modes:
 - Live-feed processing (various internal live-feeds)
 - On-demand scan of a file system

PERichFinder (Frontend)

- .NET desktop app
- Lookup of similar files in the RH DB
- Find commonalities among a set of files
- Notifications & YARA Rules

SOL & YARA Rules

VirusTotal / yara

Watch

270

<> Code

Issues 105

Pull requests 21

Projects 0

Wiki

Security

Insights

PE rich signature improvements #1135

Merged

plusvic merged 3 commits into VirusTotal:master from eset:rich_signature_improvements 10 days ago

Conversation 3

Commits 3

Checks 0



Commits on Sep 13, 2019

rich_internal function now returns the sum of counts ...

mposlusny authored and marc-etienne committed on Jul 24

Update documentation of the `toolid` and `version` in PE module

mposlusny authored and marc-etienne committed on Jul 29

Added test for the updated rich_signature function

mposlusny authored and marc-etienne committed on Jul 30

`toolid(toolid, [version])`

New in version 3.5.0.

Function returning a sum of count values of all matching `toolid` records. Provide the optional `version` argument to only match when both match for one entry. More information can be found here:

<http://www.ntcore.com/files/richsign.htm>

Note: Prior to version 3.11.0, this function returns only a boolean value (0 or 1) if the given `toolid` and optional `version` is present in an entry.

Example: `pe.rich_signature.toolid(170, 40219) >= 99` and `pe.rich_signature.toolid(170, 40219) <= 143`



PERichFinder on File: Level 0

The screenshot displays the PERichFinder application window. The title bar indicates the file path: `D:\_VB2019\_rich_headers\Win32.Dridex\bot_x86_v1.159.module - PERichFinder`.

Search Results Table:

prodID	version	count	misc
prodIdUtc1500_CPP	30729	1	
prodIdMasm1000	40219	24	
prodIdUtc1600_C	40219	131	
prodIdUtc1600_CPP	40219	41	
prodIdImplib900	30729	19	
prodIdImport0	0	235	
prodIdUnknown	0	1	
prodIdUtc1600_LTCG_CPP	40219	63	
prodIdExport1000	40219	1	
prodIdLinker1000	40219	1	

Configuration Panel:

- Database:** maliciousset (dropdown menu)
- XOR Key:** Valid
- Include/Exclude:** ☒ Include, ☐ Exclude
- Version:** ☒ (Value: 30729)
- Count:** ☒ (Value: 1)
- Comparison:** ☒ Equals, ☐ Range
- PE Timestamp:** 2014-11-17 13:6:4
- Filters:** ☐ Newer than, ☐ Older than, ☐ Range
- Unique SHA1:** ☒
- First 1000 Results:** ☒
- Show SQL Query:** ☐

Advanced Search Criteria (Red Boxed):

- ☐ XOR Key: 0x4BFA293D
- ☐ Real XOR Key: 0x4BFA293D
- ☐ Rich Header CRC: 0x43624D3622803975
- ☐ ProdID-Build CRC: 0xE219AC239E8F70E6
- ☐ ProdID-Build Sorted CRC: 0x68EA6BBDEF6C285
- ☐ ProdID-Count CRC: 0x51B716AB36628D2B

Other Settings:

- Import Count:** ☐ Equals, ☐ Range (Value: 16)
- Image Size:** ☐ Equals, ☐ Range (Value: 88496)
- Object count:** ☐ Equals, ☐ Range (Value: 7)
- Record Count:** ☐ Equals, ☐ Range (Value: 0)

Buttons: Search, Create Rule, Browse Rules, Export To Yara

Footer: eset, michal.poslusny@eset.cz

PERichFinder on File: Level 0

Results

hex sha1	image_size	bitness	is_dll	pe_timestamp	linker_major_version	linker_minor_version	impo
712C2D7C6D7CFD8C9A45BCA7C5DADD67222EDA05	298496	32	1	17.11.2014 13:06	10	0	216

Query took 6 ms

Result count: 1

40219 1

utorok , 18. novembra 2014

☒ Unique SHA1
☒ First 1000 Results

☐ Show SQL Query

☐ Real XOR Key
0x4BFA293D

☒ Rich Header CRC
0x43624D3622803975

☐ ProdID-Build CRC
0xE219AC239E8F70E6

☐ ProdID-Build Sorted CRC
0x68EA6BBDEF6C285

☐ ProdID-Count CRC
0x51B716AB36628D2B

☐ Image Size
☒ Equals ☐ Range
298496

☐ Object count
☒ Equals ☐ Range
517

☐ Record Count
☒ Equals ☐ Range
10

Search Create Rule Browse Rules Export To Yara

eset

michal.poslusny@eset.cz

PERichFinder on File: Level 1

Results

	hex(sha1)	image_size	bitness	is_dll	pe_timestamp	linker_major_version	linker_minor_version	impo
▶	712C2D7C6D7CFD8C9A45BCA7C5DADD67222EDA05	298496	32	1	17.11.2014 13:06	10	0	216

Query took 6 ms

Result count: 1

prodlinker1000 40219 1

utorok , 18. novembra 2014

☒ Unique SHA1
☒ First 1000 Results
☐ Show SQL Query

☒ Real XOR Key
0x4BFA293D

☐ Rich Header CRC
0x43624D3622803975
☐ ProdID-Build CRC
0xE219AC239E8F70E6
☐ ProdID-Build Sorted CRC
0x68EA6BBDEF6C285
☐ ProdID-Count CRC
0x51B716AB36628D2B

216
☐ Image Size
☒ Equals ☐ Range
298496
☐ Object count
☒ Equals ☐ Range
517
☐ Record Count
☒ Equals ☐ Range
10

Search Create Rule Browse Rules Export To Yara

michal.poslusny@eset.cz



PERichFinder on File: Level 2

Results

	hex(sha1)	image_size	bitness	is_dll	pe_timestamp	linker_major_version	linker_minor_version	import
▶	024DA6B4E9BF0B80F52DD8BB754F61C544904AA7	314368	32	1	16.2.2015 14:02	10	0	164
	02E4B208300BE6CD37B199A1817E8843E6CCC9E9	304640	32	1	23.10.2014 16:59	10	0	225
	0427060D2C46F2302E67AB43884220AB293A92C1	308224	32	1	19.12.2014 16:07	10	0	208
	06B9071913FC0EA5045B53F118C5979030A894A7	313344	32	1	10.2.2015 22:08	10	0	165
	001000C03E2A11454C0001FC05730DE302D5132D...	307424	32	1	20.10.2014 16:07	10	0	222

Query took 5 ms

Result count: 111

☒ Unique SHA1
☒ First 1000 Results

☐ Show SQL Query

☒ ProdID-Build CRC
0xE219AC239E8F70E6

☐ ProdID-Build Sorted CRC
0x68EA6B8DEF6C285

☐ ProdID-Count CRC
0x51B716AB36628D2B

☐ Object count
☒ Equals ☐ Range
298496

☐ Record Count
☒ Equals ☐ Range
10

Search

Create Rule

Browse Rules

Export To Yara

michal.poslusny@eset.cz



PERichFinder on File: Level 3



Results

	hex(sha1)	image_size	bitness	is_dll	pe_timestamp	linker_major_version	linker_minor_version	import
▶	01E57AF4EEA3A4114FC66EF0D1EEDDCADB381530	297472	32	1	25.7.2014 12:02	10	0	216
	024DA6B4E9BF0B80F52DD8BB754F61C544904AA7	314368	32	1	16.2.2015 14:02	10	0	164
	02E4B208300BE6CD37B199A1817E8843E6CCC9E9	304640	32	1	23.10.2014 16:59	10	0	225
<	0A27060D3C46E2207E67AB42884220AB282A82C1	308224	32	1	18.12.2014 16:07	10	0	200

Query took 8 ms

Result count: 124

- ☒ Unique SHA1
- ☒ First 1000 Results

☐ Show SQL Query

Rich Header CRC

0x43624D3622803975

ProdID-Build CRC

0xE2184C239E8E70E6

☒ ProdID-Build Sorted CRC

0x68EA6BBDEF6C285

ProdID-Count CRC

0x51B716AB36628D2B

image_size

☒ Equals

298496

☐ Object count

☒ Equals

517

☐ Record Count

☒ Equals

10

Search

Create Rule

Browse Rules

Export To Yara

michal.poslusny@eset.cz



PERichFinder on Folder: Step 0

DA:\VB2019\rich_headers\group_059_APT37 - PERichFinder

prodID	version	count	misc
prodidImplib900	30729	12 - 21	5/5 files
prodidImport0	0	56 - 155	5/5 files
prodidUtc1500_C	30729	1 - 1	4/5 files
prodidAliasObj1000	20115	1 - 1	4/5 files
prodidUtc1600_CPP	30319	37 - 40	4/5 files
prodidMasm1000	30319	16 - 19	4/5 files
prodidUtc1600_C	30319	104 - 117	4/5 files
prodidUtc1600_LTCG_CPP	30319	18 - 18	4/5 files
prodidCvtres1000	30319	1 - 1	4/5 files
prodidLinker1000	30319	1 - 1	4/5 files
prodidImplib1400	23917	9 - 9	1/5 files
prodidUtc1900_C	23917	17 - 17	1/5 files
prodidMasm1400	23917	2 - 2	1/5 files
prodidUtc1900_LTCG_CPP	23917	2 - 2	1/5 files
prodidUtc1900_CPP	23917	3 - 3	1/5 files
prodidCvtres1400	23917	1 - 1	1/5 files
prodidLinker1400	23917	1 - 1	1/5 files

prodidImplib900 Database: maliciouset ->

☒ Include ☒ Version ☒ Count
☐ Exclude 30729 12 21

Add Remove

☐ PE Timestamp
sobota , 16. júla 2016 ☒ Newer than
utorok , 16. januára 2018 ☐ Older than
☐ Range

☐ Unique SHA1
☒ First 1000 Results

☐ Show SQL Query

☐ Import Count
☒ Equals ☐ Range
52

☐ Image Size
☒ Equals ☐ Range
32256

☐ Object Count
☒ Equals ☐ Range
103

☐ Record Count
☒ Equals ☐ Range
9

Search Create Rule Browse Rules Export To Yara

michal.poslusny@eset.cz

PERichFinder on Folder: Step 1

The screenshot shows the PERichFinder application window titled "D:_VB2019_rich_headers_group_059_APT37 - PERichFinder". It features a table of products and various search filters.

prodID	version	count	misc
prodidImplib900	30729	12 - 21	5/5 files
prodidImport0	0	56 - 155	5/5 files
prodidUtc1500_C	30729	1 - 1	4/5 files
prodidAliasObj1000	20115	1 - 1	4/5 files
prodidUtc1600_CPP	30319	37 - 40	4/5 files
prodidMasm1000	30319	16 - 19	4/5 files
prodidUtc1600_C	30319	104 - 117	4/5 files
prodidUtc1600_LTCG_CPP	30319	18 - 18	4/5 files
prodidCvtres1000	30319	1 - 1	4/5 files
prodidLinker1000	30319	1 - 1	4/5 files
prodidImplib1400	23917	9 - 9	1/5 files
prodidUtc1900_C	23917	17 - 17	1/5 files
prodidMasm1400	23917	2 - 2	1/5 files
prodidUtc1900_LTCG_CPP	23917	2 - 2	1/5 files
prodidUtc1900_CPP	23917	3 - 3	1/5 files
prodidCvtres1400	23917	1 - 1	1/5 files
prodidLinker1400	23917	1 - 1	1/5 files

Search filters on the right include:

- prodidLinker1400** (selected)
- Database:** maliciousset
- XOR Key:**
- Include/Exclude:** Version (checked), Count (checked)
- PE Timestamp:** sobota, 16. júla 2016 (Newer than), utorok, 16. januára 2018 (Older than)
- Unique SHA1:** First 1000 Results (checked)
- Import Count:** Equals (selected), Range (unchecked), 52
- Image Size:** Equals (selected), Range (unchecked), 32256
- Object Count:** Equals (selected), Range (unchecked), 103
- Record Count:** Equals (selected), Range (unchecked), 9

A red box highlights the 'prodidLinker1400' entry in the table. A smaller window titled 'prodidLinker1400' shows details for this entry:

File name	Count
D:_VB2019_rich_headers_group_059_APT37\calc.exe	1

Press Delete

Buttons at the bottom: Search, Create Rule, Browse Rules, Export To Yara.

Footer: michal.poslusny@eset.cz

PERichFinder on Folder: Step 2

prodID version count misc

prodIdUtc1500_C	30729	1 - 1	4/4 files
prodIdAliasObj1000	20115	1 - 1	4/4 files
prodIdUtc1600_CPP	30319	37 - 40	4/4 files
prodIdMasm1000	30319	16 - 19	4/4 files
prodIdUtc1600_C	30319	104 - 117	4/4 files
prodIdImplb900	30729	21 - 21	4/4 files
prodIdImport0	0	154 - 155	4/4 files
prodIdUtc1600_LTCG_CPP	30319	18 - 18	4/4 files
prodIdCvtres1000	30319	1 - 1	4/4 files
prodIdLinker1000	30319	1 - 1	4/4 files

prodIdUtc1500_C Database: maliciousset

☒ Include ☒ Version ☒ Count
☐ Exclude 30729 1 1

Add Remove

☐ PE Timestamp
sobota , 16. júla 2016 ☐ Newer than
utorok , 16. januára 2018 ☐ Older than
☐ Range

☐ Unique SHA1
☒ First 1000 Results

☐ Show SQL Query

☐ Import Count
☒ Equals ☐ Range
79

☐ Image Size
☒ Equals ☐ Range
137704

☐ Object Count
☒ Equals ☐ Range
354

☐ Record Count
☒ Equals ☐ Range
10

Search Create Rule Browse Rules Export To Yara

michal.poslusny@eset.cz

PERichFinder on Folder: Step 3

D:_VB2019_rich_headers_group_059_APT37 - PERichFinder

prodID	version	count	misc
prodIdUtc1500_C	30729	1 - 1	4/4 files
prodIdAliasObj1000	20115	1 - 1	4/4 files
prodIdUtc1600_CPP	30319	37 - 40	4/4 files
prodIdMasm1000	30319	16 - 19	4/4 files
prodIdUtc1600_C	30319	104 - 117	4/4 files
prodIdImpak900	30729	21 - 21	4/4 files
prodIdImport0	0	154 - 155	4/4 files
prodIdUtc1600_LTCG_CPP	30319	18 - 18	4/4 files
prodIdCvtes1000	30319	1 - 1	4/4 files
prodIdLinker1000	30319	1 - 1	4/4 files

prodIdLinker1000 Database: maliciousset

☒ Include ☒ Version ☒ Count
☐ Exclude 30319 1 1

Add Remove

☐ PE Timestamp
sobota , 16. júla 2016 ☒ Newer than
utorok , 16. januára 2018 ☐ Older than
☐ Range

☐ Unique SHA1
☒ First 1000 Results

☐ Show SQL Query

☐ Import Count
☒ Equals ☐ Range
79

☐ Image Size
☒ Equals ☐ Range
137704

☐ Object Count
☒ Equals ☐ Range
354

☐ Record Count
☒ Equals ☐ Range
10

Search Create Rule Browse Rules Export To Yara

michal.poslusny@eset.cz

PERichFinder on Folder: Step 4

D:_VB2019_rich_headers_group_059_APT37 - PERichFinder

prodID	version	count	misc
prodIDUtc1500_C	30729	1 - 1	4/4 files
prodIDAliasObj1000	20115	1 - 1	4/4 files
prodIDUtc1600_CPP	30319	37 - 40	4/4 files
prodIDMasm1000	30319	16 - 19	4/4 files
prodIDUtc1600_C	30319	104 - 117	4/4 files
prodIDImpLib900	30729	21 - 21	4/4 files
prodIDImport0	0	154 - 155	4/4 files
prodIDUtc1600_LTCG_CPP	30319	18 - 18	4/4 files
prodIDCvres1000	30319	1 - 1	4/4 files
prodIDLinker1000	30319	1 - 1	4/4 files

prodIDLinker1000

Database: maliciouset

☒ Include ☒ Version ☒ Count

☐ Exclude

30319 1 1

Add Remove

XOR Key:

☐ PE Timestamp

sobota , 16. júla 2016 ☒ Newer than

utorok , 16. januára 2018 ☐ Older than

☐ Import Count

☒ Equals ☐ Range

79

Results

	hex(sha1)	image_size	bitness	is_dll	pe_timestamp	linker_major_version	linker_minor_version	import_count	hex(import_hash)
▶	762E6B77D0831574C750B2679C33CB4AB391522A	216064	32	0	16.1.2018 1:43	10	0	79	1576E8CB8369AD34BEA0
	82C22CDD45763D873CCA295659FB0FE344CEC769	137704	32	0	11.1.2018 14:01	10	0	80	FB2BDD4739F39508FA50A
	FB1C759BBBE2435ABD19FAA101FF322308CD727A	311808	32	0	23.2.2018 7:00	10	0	85	67C8D6C068A58909B0C2

Query took: 6 ms

Result count: 3

Search Create Rule Browse Rules Export To Yara

michal.poslusny@eset.cz

PERichFinder on Folder: Step 5

D:_VB2019_rich_headers\group_059_APT37 - PERichFinder

prodID	version	count	misc
prodIdUtc1500_C	30729	1 - 1	4/4 files
prodIdAlasObj 1000	20115	1 - 1	4/4 files
prodIdUtc1600_CPP	30319	37 - 40	4/4 files
prodIdMasm1000	30319	16 - 19	4/4 files
prodIdUtc1600_C	30319	104 - 117	4/4 files
prodIdImpLib900	30729	21 - 21	4/4 files
prodIdImport0	0	154 - 155	4/4 files
prodIdUtc1600_LTCG_CPP	30319	18 - 18	4/4 files
prodIdCvres1000	30319	1 - 1	4/4 files
prodIdLinker1000	30319	1 - 1	4/4 files

prodIdLinker1000

☒ Include ☒ Version ☒ Count
☐ Exclude

30319 1 1

Add Remove

☐ PE Timestamp

sobota , 16. júla 2016 ☒ Newer than
utorok , 16. januára 2018 ☐ Older than

☐ Import Count
☒ Equals ☐ Range
79

XOR Key:

Results

hex(sha1)	image_size	bitness	is_dll	pe_timestamp	linker_major_version	linker_minor_version	import_count	hex(import_hash)
-----------	------------	---------	--------	--------------	----------------------	----------------------	--------------	------------------

Query took: 3 ms
Result count: 0

0 hits

Search Create Rule Browse Rules Export To Yara

michal.poslusny@eset.cz

SQL & YARA rules

```
SELECT hex(sha1) from file where
file_id in
(SELECT file_id from rich_record where rich_object_id = 170 and rich_object_version = 40219 and rich_object_count >= 99 and rich_object_count <= 143)
and file_id in
(SELECT file_id from rich_record where rich_object_id = 171 and rich_object_version = 40219 and rich_object_count >= 32 and rich_object_count <= 48)
and file_id in
(SELECT file_id from rich_record where rich_object_id = 158 and rich_object_version = 40219 and rich_object_count >= 23 and rich_object_count <= 26)
and file_id in
(SELECT file_id from rich_record where rich_object_id = 0 and rich_object_version = 0 and rich_object_count = 1) LIMIT 1000
```

```
import "pe"
rule Dridex_v1_v2_v3
{
    condition:
        pe.rich_signature.toolid(170, 40219) >= 99 and pe.rich_signature.toolid(170, 40219) <= 143 and
        pe.rich_signature.toolid(171, 40219) >= 32 and pe.rich_signature.toolid(171, 40219) <= 48 and
        pe.rich_signature.toolid(158, 40219) >= 23 and pe.rich_signature.toolid(158, 40219) <= 26 and
        pe.rich_signature.toolid(0, 0) == 1
}
```

SQL & YARA Rules

- **SQL rule:** SELECT into the RH database
- **YARA rule:** Conjunction of toolids and lengths ranges
- 1-1 correspondence between SQL and YARA rules !!!
- Enhancing the YARA project necessary (Counts, Import Function etc.)
- ~200 rules covering mostly APT toolsets

RH is a static feature that can *reasonably* distinguish malicious projects from the clean ones and classify their clusters.

Lessons learned

Lessons learned - (Dis)Advantages

- + RH is a small piece of data easily stored and quickly accessed
- + Malicious projects are of a small size
- + Multi-stage threats and 32/64-bit variants often covered with a single rule
- + Creation of anomalies leads to malware verdict
- The rule needs to exist (no proactivity)
- Tracking is lost with the update of Visual Studio or a larger project refactoring

Lessons learned - FPs

- “For each of the 200 rules already exists a false positive.”
- The nature of FPs:
 - + various Proof-of-Concepts
 - + (signed) tools from specialized software
 - + (unsigned) small components of projects of unknown origin and functionality
 - + unrelated malware families

(Back to



Since 2009, HIDDEN COBRA actors have leveraged their capabilities to target and compromise a range of victims; some intrusions have resulted in the exfiltration of data while others have been disruptive in nature. Commercial reporting has referred to this activity as Lazarus Group and Guardians of Peace. Tools

Examples (0)

WWW.VIRUSBULLETIN.COM/CONFERENCE



Peter Kálnai & Michal Poslušný
ESET, Czech Republic

(peter.kalnaj, michal.poslusny)@eset.cz

ABSTRACT

The number of incidents attributed to the Lazarus Group, a.k.a. Hidden Cobra, has grown rapidly since its estimated establishment in 2009. This notorious group intensified its efforts in 2017 (e.g. the attacks on Polish and Mexican banks, the WannaCry/Cryptor outbreak, the spear-phishing campaign against US contractors), and kept up the pace at the turn of the year (the *Android*-ported payloads, the bitcoin-oriented attacks, the Turkish campaign, and more). Attribution of these newer cases was determined by observing similarities with previously resolved cases: specific chunks of code, unique data, and network infrastructure. In this paper we summarize the crucial links that played a role in these major cases.

The source code of the group's tool appears to be modified with every attack. There are several static features that vary between the instances: digital Windows API resolution and the use of the `System.out.println()` method. The code for deleting batch files, the list of domains leveraged for file TLP communication, the format strings included in TCP backlogs, the use of commercial packages, etc. The variety is so huge that it is not possible to find a single instance that fits all the ideas that the Lazarus group might be split into multiple subgroups. The code is also modified to include specific metadata when exploring the undocumented PE Rich Header metadata, which once again indicates that there are various development environments producing the malicious binaries.

There are also several binaries from the Lazarus toolkit that have been modified to include specific metadata. One of the most interesting findings to the Lazarus puzzle: the very first release of WannaCrypt from 2016, in the wild-peripatetic with the malicious file downloaders targeting multiple platforms, the code is a copy of the Lazarus group's code. It is a curious artifact like Chinese language or South Korean cultural references. This paper will present previously unpublished details about the cyber-subspace attack against an online casino in South Korea, the first time that the Lazarus group's malware (corrupted `1-888-848-7243` file) was behind the attack.

INTRODUCTION

The activity of Lazarus toolset components can be traced back as far as 2009. Several typical Lazarus backdoors were uploaded to VirusTotal that year, e.g. `netprow.dll` and `divmcent.exe` [1]. However, the first published identification of the so-called Lazarus Group and its toolkit was not until many years later in Novetta's extensive papers [2] in February 2016. The first mention of Lazarus at a Virus Bulletin conference was also in 2016, when Bartholomew and Guerrero-Saade of Kaspersky

Lab described the pseudo-hacktivism tendencies of two famous Lazarus attacks [3]. Since 2017, especially after the WannaCryptor outbreak, the number of Lazarus-related reports has proliferated. In this paper, we summarize the crucial fingerprints that led malware researchers to attribute the famous cases to the group, and discuss the main characteristics that have helped us to ascribe further samples to the group. Finally, we show six suspected Lazarus-related cases that we believe are not widely known.

REPORTED CASES

Operation Troy and DarkSeoul

Lazarus Group first came into the spotlight in 2013, when reports about two of their campaigns in South Korea were published for the first time. The long-term campaign called Operation Troy was a cyber espionage operation against South Korean armed forces and government targets, and ran between the years 2009 and 2012. The second of these campaigns, called DarkSeoul, occurred in 2013 and mainly targeted the South Korean financial sector.

Binaries involved in these operations often preserved symbol paths¹ – details can be found in [5, 6]. ESET detects most of the malware known to have been used in these campaigns or similar as Win32/Spy.Keydoor or Win64/Spy.Keydoor.

Operation Blockbuster – the saga, the sequel and going mobile

Sony Pictures Entertainment went through a very tough period in 2014, when the company was the victim of one of the most destructive cyber attacks against a commercial entity to date. The attack caused major damage to the company, and many of its internal files and documents were stolen, leaked or deleted. The binaries involved in the attack, as well as legions of statistically similar files were later extensively described by Novetta [2], which named the attack 'Operation Blockbuster'. Regarding the many common overlapping characteristics of binaries, this epic report was preceded by Symantec's report [7] by several months and, a year later, was followed by Palo Alto Networks' series of blog posts *The Blockbuster Sequel [8]*, *The Blockbuster Sequel 2 [9]*, *Blockbuster Sequel 3 [10]*, and *Blockbuster Goes Mobile [10]*. The new attacks were tied to Lazarus by the re-use of self-deleting batch files, format strings in the TCP backdoors, dynamic API loading routines, obfuscation of function names, and the use of fake TLS communications.

³ For example, `Z:\Mission\Team_Project\2012.6 - HTTP Troy\HtpDrOpfer\Win32\Release\PayloadDll.pdb` or `D:\Work\Op\Mission\TeamProject\2012.11-12\TDrop\Payload\32\Release\PayloadDll32.pdb`. We have found only two *NukeSped* samples with paths:

E:\ConstructSystem\widebor\bot_dll\Debug\bot_dll.pdb
D:\Source\Source_C\Work_Source\T(3.1_Unicode)\Server_x64\Release\ServerDll.pdb

Main subgroups

- 1) x86 VS98 + x64 VS2010
- 2) x86 VS2010 + x64 VS2010

..and anomalies...



VS98 + VS2013, not
expected VS98 + VS2010)

Examples (1)

win.industry (Back to overview)

 Industroyer

Propose Change

aka: Crash,
CrashOverride

Actor(s): ELECTRUM



Industroyer is a malware framework considered to have been used in the cyberattack on Ukraine's power grid on December 17, 2016. The attack cut a fifth of Kiev, the capital, off power for one hour. It is the first ever known malware specifically designed to attack electrical grids.

References

<https://en.wikipedia.org/wiki/Industroyer>

CONJUNCTION OF RANGES:

Object Count	350..460
Utc1810_CPP(40116)	120..121
Masm1400 (24123)	17...18
Utc1900_CPP(24123)	29...33
Import0	100..200
Cvtres1400 (24210)	1....1

win.exaramel (Back to overview)

 Exaramel

Propose Change

Actor(s): TeleBots



There is no description at this point.

References

<https://www.welivesecurity.com/2018/10/11/...>

Yara Rules

```
► [TLP:WHITE] win_exaramel_auto
(20190620 | autogenerated rule
brought to you by yara-signator)
```

win.industroyer (Back to overview)

Industroyer

Propose Change

aka: Crash,
CrashOverride
Actor(s): **ELECTRUM**



Industroyer is a malware that has been used in the power grid on December a fifth of Kiev, the capital. It is the first ever known designed to attack electrical grids.

References

<https://en.wikipedia.org/wiki/Industroyer>

CONJUNCTION OF RANGES:

win.exaramel (Back to overview)

Exaramel

Propose Change



tion at this point.

esecurity.com/2018/10/11/...

New TeleBots backdoor: First evidence linking Industroyer to NotPetya

ESET's analysis of a recent backdoor used by TeleBots – the group behind the massive NotPetya ransomware outbreak – uncovers strong code similarities to the Industroyer main backdoor, revealing a rumored connection that was not previously proven



Anton Cherepanov and Robert Lipovsky 11 Oct 2018 - 01:57PM

► [TLP:WHITE] win_exaramel_auto
(20190620 | autogenerated rule
brought to you by yara-signator)



Hacking Team [\(Back to overview\)](#)



The many 0-days that had been collected by Hacking Team and which became publicly available during the breach of their organization in 2015, have been used by several APT groups since. Since being founded in 2003, the Italian spyware vendor Hacking Team gained notoriety for selling surveillance tools to governments and their agencies across the world. The capabilities of its flagship product, the Remote Control System (RCS), include extracting files from a targeted device, intercepting emails and instant messaging, as well as remotely activating a device's webcam and microphone. The company has been criticized for selling these capabilities to authoritarian governments – an allegation it has consistently denied. When the tables turned

Examples (2)

- happynewyear-gpj.exe
- Character strings: SCOUTSCOUTSCOUT
- Methods of Dynamic Calls

...

It's Hacking Team, right?



No! Just a downloader of Win/Navrat



APT37 [\(Back to overview\)](#)



aka: APT 37, Group 123, Group123, Starcraft, Reaper, Reaper Group, Red Eyes, Ricochet Chollima, StarCraft, Operation Daybreak, Operation Erebus, Venus 121

APT37 has likely been active since at least 2012 and focuses on targeting the public and private sectors primarily in South Korea. In 2017, APT37 expanded its targeting beyond the Korean peninsula to include Japan, Vietnam and the Middle East, and to a wider range of industry verticals, including chemicals, electronics, manufacturing, aerospace, automotive and healthcare entities

Associated Families

win.final1stspy win.freenki
win.navrat win.nokki win.poohmilk
win.rokrat win.starcraft

Examples (3)

win.prikormka ([Back to overview](#))

 Prikormka

[Propose Change](#)

Actor(s):

Groundbait



There is no description at this point.

References

<https://www.welivesecurity.com/wp-content/u...>

Yara Rules

```
► [TLP:WHITE] win_prikormka_auto   
(20190620 | autogenerated rule  
brought to you by yara-signator)
```

- Suspicious File called etwdrv.dll
- Export Name: LCrPsdNew.dll
- PE TimeStamp: 29.9.2017 11:06:41



- Export Name: loadCryptPsd.dll
- PE TimeStamp: 5.1.2017 11:30:15

Detection: Win64/Prikormka.BF trojan

Summary

Summary

- Implemented since VS 97 SP3, Microsoft has never announced, documented or allowed to opt out this feature.
- A small team of developers or even a single dedicated person could build the required infrastructure.
- RH is a static feature that can *reasonably* distinguish malicious projects from the clean ones and classify their clusters.

Questions & Answers





Peter Kálnai

Senior Malware Researcher



Michal Poslušný

Malware Researcher



@ESETresearch

