

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.

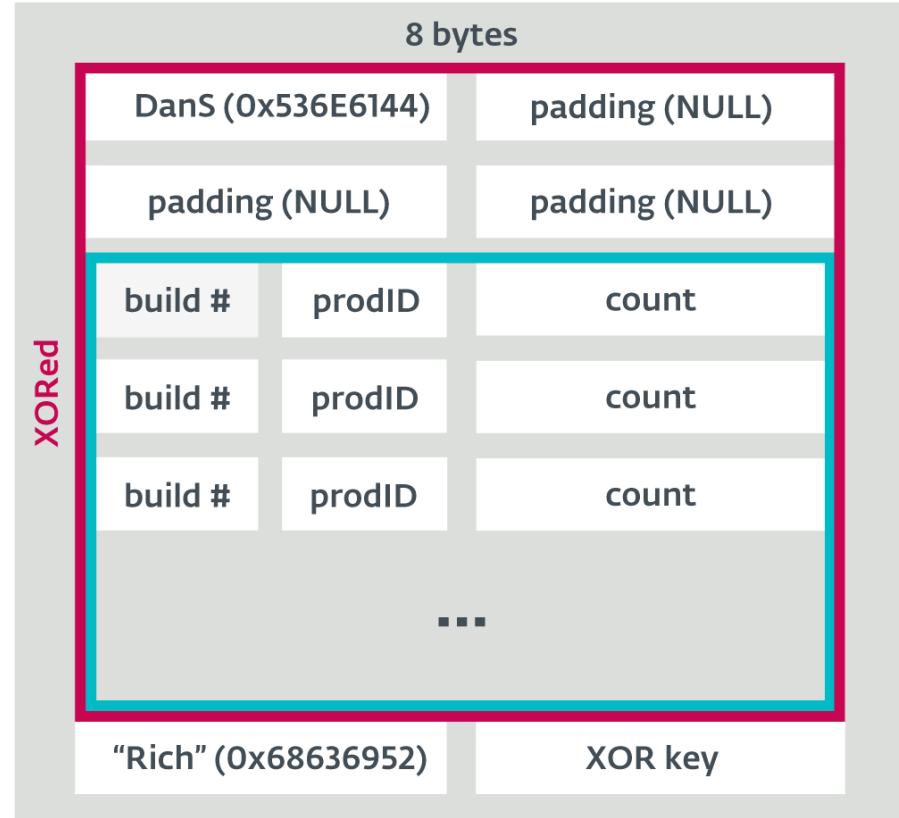
Description of Rich Headers

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 MZÉ
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 65-2F 0D 0D 00-24 00 00 00-00 00 00 00
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 dâ
```



```
00-00 00 00 00 PE
00-00 00 00 00 DanS
01-14 00 00 00 ef
01-09 00 00 00 ef
01-23 00 00 00 j
01-0A 00 00 00 u
00-EA 00 00 00 R
00-01 00 00 00 tk
01-01 00 00 00 tk
00-00 00 00 00 Rich
```

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

#define DwProdidFromProdidwBuild(prodid, wBuild) (((unsigned long) (prodid) << 16) | (wBuild))
#define ProdidFromDwProdid(dwProdid) ((PROID) ((dwProdid) >> 16))
#define WBuildFromDwProdid(dwProdid) ((dwProdid) & 0xFFFF)

// Define the image data format
typedef struct PRODITEM {
    unsigned long dwProdid; // Product identity
    unsigned long dwCount; // Count of objects built with that product
} PRODITEM;

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

Product IDs

'DanS'

'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.

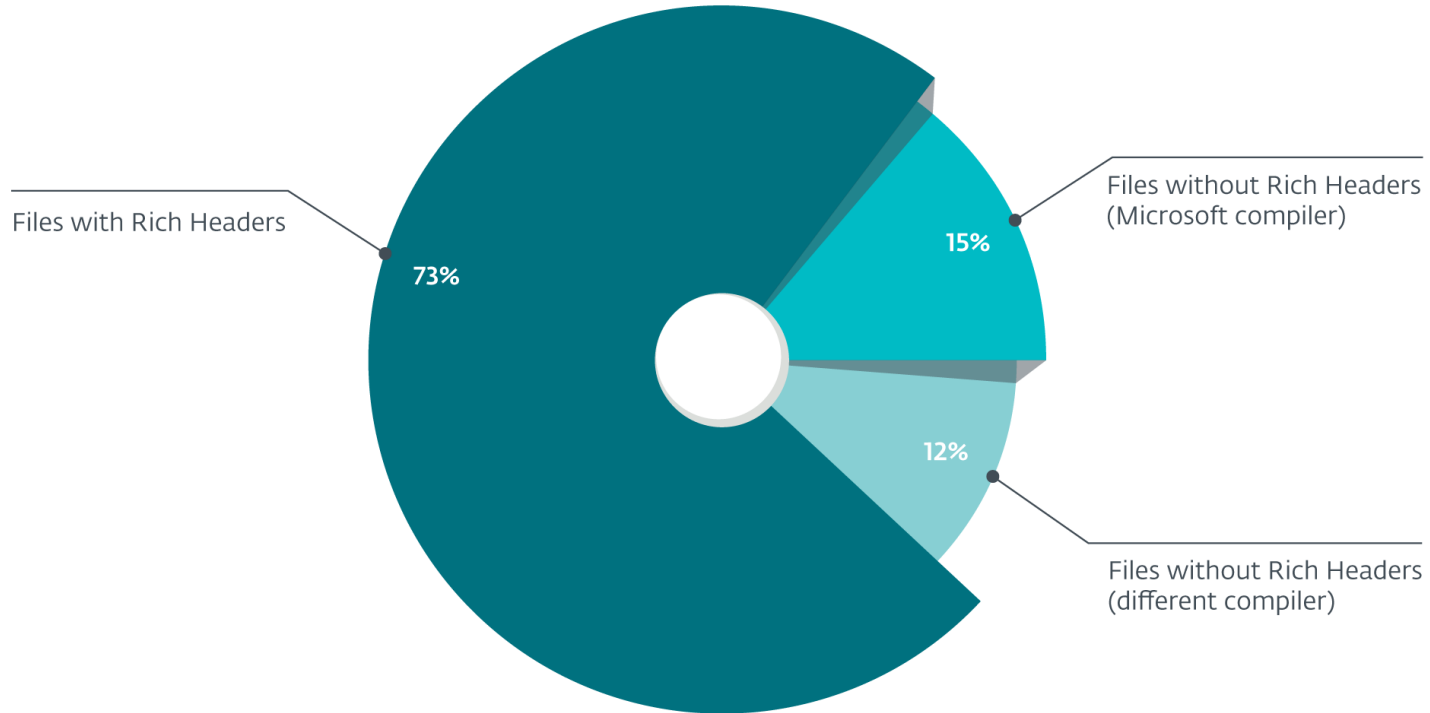
```
struct {
    IMAGE_DOS_HEADER dosHeader;
    BYTE             rgbDosStub[N]; // MS-DOS stub
    PRODITEM         { tagEndID, 0 }; // start of tallies (Masked with dwMask)
    PRODITEM         { 0, 0 }; // end of tallies (Masked with dwMask)
    PRODITEM         rgproditem[]; // variable sized (Masked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

XOR key

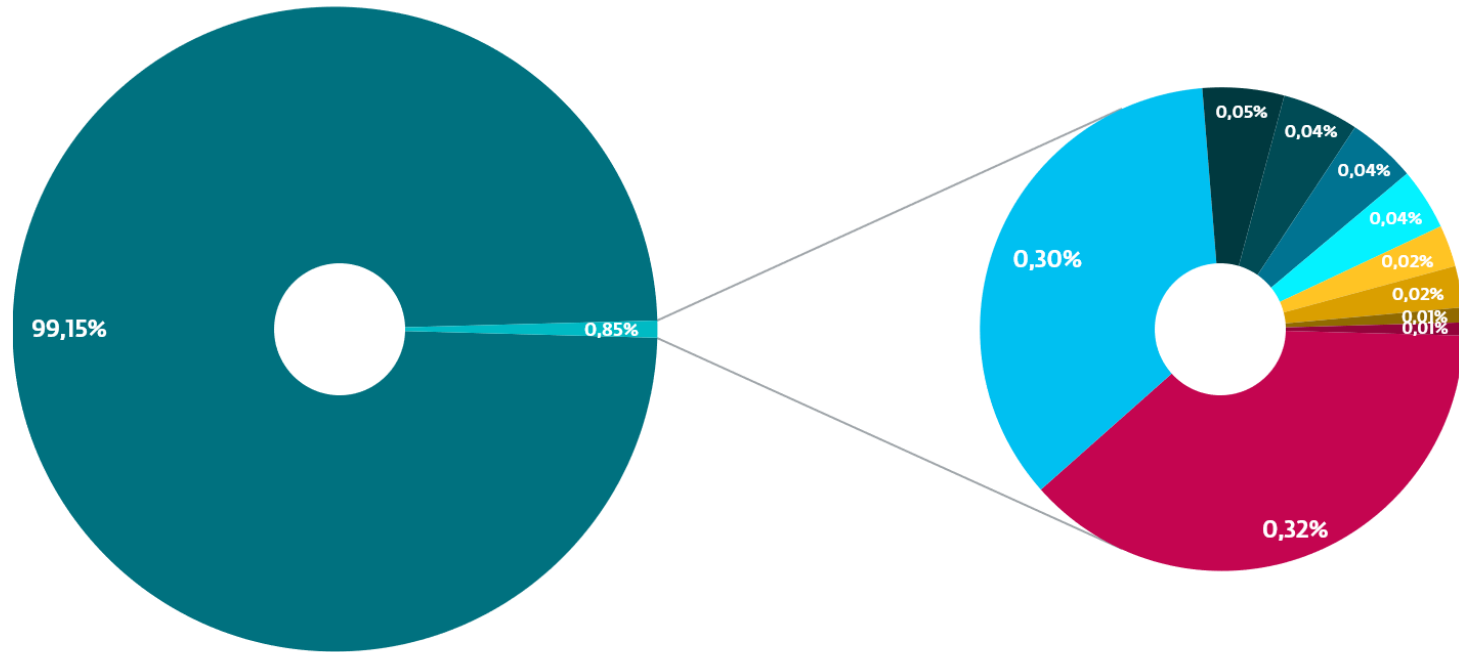
- XOR key generation VS 2019:

```
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 ■ 0x60 ■ 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, AutoIt,

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_LTTCG_CPP (b40219)	22	prodidUtc1600_LTTCG_CPP (b40219)	22-23

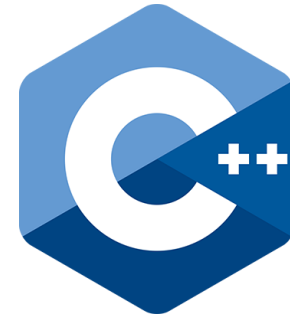
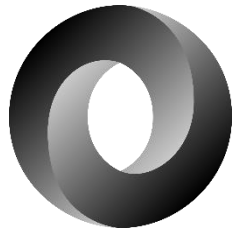
RH Anomalies

- 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



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



`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`

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



PERichFinder on File: Level 0

Di:_VB2019_rich_headers\Win32.Dridex\bot_x86_v1.159.module - PERichFinder

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	

Database: maliciousset

Include Version Count
 Exclude

30729 1

Equals Range

Add Remove

PE Timestamp 2014-11-17 13:6:4

nedelja , 16. novembra 2014 Newer than
utorok , 18. novembra 2014 Older than
 Range

Unique SHA1
 First 1000 Results

Show SQL Query

XOR Key
0x4BFA293D

Real XOR Key
0x4BFA293D

Rich Header CRC
0x43624D3622803975

ProdID-Build CRC
0xE219AC239E8F70E6

ProdID-Build Sorted CRC
0x68EA6BBDEF6C285

ProdID-Count CRC
0x51B716AB36628D2B

Import Count
 Equals Range
6

Image Size
 Equals Range
8496

Object count
 Equals Range
7

Record Count
 Equals Range
0

Search Create Rule Browse Rules Export To Yara

michal.poslusny@eset.cz

PERichFinder on File: Level 0

The screenshot displays the PERichFinder application window titled "Results". At the top, a table lists search results with columns: hex sha1, image_size, bitness, is_dll, pe_timestamp, linker_major_version, linker_minor_version, and impo. The first row is highlighted, showing a SHA1 hash, image size of 298496, bitness of 32, is_dll of 1, pe_timestamp of 17.11.2014 13:06, linker_major_version of 10, linker_minor_version of 0, and impo of 216. Below the table, a status bar indicates "Query took 6 ms".

On the left side, a red box highlights the "Result count: 1" label. Below this, a large greyed-out area represents the search results list.

On the right side, the configuration panel is visible. It includes a date selector set to "18. novembra 2014" and radio buttons for "Older than" and "Range". A list of search criteria is shown, with "Rich Header CRC" checked and its value "0x43624D3622803975" highlighted by a red box. Other criteria include "Real XOR Key", "Image Size", "Object count", and "Record Count".

At the bottom, there are buttons for "Search", "Create Rule", "Browse Rules", and "Export To Yara". The ESET logo and the email "michal.poslusny@eset.cz" are located in the bottom right corner.

hex sha1	image_size	bitness	is_dll	pe_timestamp	linker_major_version	linker_minor_version	impo
712C2D7C6D7CFD8C9A45BCA7C5DADD6722EDA05	298496	32	1	17.11.2014 13:06	10	0	216

PERichFinder on File: Level 1

The screenshot displays the PERichFinder application window titled "Results". At the top, a table lists search results with columns: hex sha1, image_size, bitness, is_dll, pe_timestamp, linker_major_version, linker_minor_version, and impo. The first row is highlighted, showing a SHA1 hash and other file metadata.

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

Below the table, it indicates "Query took 6 ms" and "Result count: 1".

The configuration panel on the right includes several options:

- Real XOR Key (highlighted with a red box) with value 0x4BFA293D
- Rich Header CRC (value: 0x43624D3622803975)
- ProdID-Build CRC (value: 0xE219AC239E8F70E6)
- ProdID-Build Sorted CRC (value: 0x68EA6BBDEF6C285)
- ProdID-Count CRC (value: 0x51B716AB36628D2B)

Other configuration options include Image Size (216), Object count (517), and Record Count (10).

At the bottom, there are buttons for "Search", "Create Rule", "Browse Rules", and "Export To Yara". The ESET logo and the email address "michal.poslusny@eset.cz" are visible in the bottom right corner.

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
	091009C03E2A11454C0901FC957309E1020E132D...	307472	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

Object count
 Equals Range
298496
517

Record Count
 Equals Range
10

ProdID-Build CRC
0xE219AC239E8F70E6

ProdID-Build Sorted CRC
0x68EA6BBDEF6C285

ProdID-Count CRC
0x51B716AB36628D2B

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
<	0A270F0D2C46E2202E67AB42884220AB282A82C1	308224	32	1	18.12.2014 16:07	10	0	200

Query took 8 ms

Result count: 124

Rich Header CRC
0x43624D3622803975

ProdID-Build CRC
0xE2194C239E8E70E6

ProdID-Build Sorted CRC
0x68EA6BBDEF6C285

ProdID-Count CRC
0x51B716AB36628D2B

Image Size
 Equals Range
298496

Object count
 Equals Range
517

Record Count
 Equals Range
10

Unique SHA1
 First 1000 Results
 Show SQL Query

Search

Create Rule

Browse Rules

Export To Yara

michal.poslusny@eset.cz



PERichFinder on Folder: Step 0

Database: maliciouset

Include Version Count
 Exclude

XOR Key:

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

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

PERichFinder on Folder: Step 1

The screenshot shows the PERichFinder application window titled "D:_VB2019_rich_headers_group_059_APT37 - PERichFinder". The main window is divided into several sections:

- Table:** A table with columns "prodID", "version", "count", and "misc". The row for "prodidLinker1400" is highlighted in blue.
- Search Criteria:** A section for "prodidLinker1400" with options for "Include" (selected) and "Exclude". It includes input fields for "Version" (23917), "Count" (1), and "XOR Key" (1). There are "Add" and "Remove" buttons.
- Filters:** Checkboxes for "PE Timestamp", "Unique SHA1", and "First 1000 Results" (checked). The "PE Timestamp" section has date pickers for "Newer than" (sobota, 16. júla 2016) and "Older than" (utorok, 16. januára 2018), along with "Range" radio buttons.
- Counters:** Four sections with checkboxes and radio buttons for "Import Count", "Image Size", "Object Count", and "Record Count", each with an input field and "Equals" or "Range" options.
- Buttons:** "Search", "Create Rule", "Browse Rules", and "Export To Yara" buttons at the bottom.

A red-bordered window titled "prodidLinker1400" is overlaid on the table, showing a list of files:

File name	Count
D:_VB2019_rich_headers_group_059_APT37\calc.exe	1

The text "Press Delete" is written in red next to this window.

Database: maliciousset

Search

Create Rule

Browse Rules

Export To Yara

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
prodIdImpIib900	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
prodIdImpab900	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

prodIdLinker1000 Database: maliciousset

Include Version Count
 Exclude 30319 1 1

XOR Key:

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

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
prodIdImpab900	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

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	FB2BDD4739F39508FA50
FB1C759BBBE2435ABD19FAA101FF322308CD727A	311808	32	0	23.2.2018 7:00	10	0	85	67CBD6C068A58909B0C2

Query took: 6 ms
Result count: 3

Search Create Rule Browse Rules Export To Yara

michal.poslusny@eset.cz

PERichFinder on Folder: Step 5

DA:\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: cleanset

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

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

Examples (0)



Lazarus Group

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


aka: Operation DarkSeoul, Dark Seoul, Hidden Cobra, Hastati Group, Andariel, Unit 121, Bureau 121, NewRomanic Cyber Army Team, Bluenoroff, Subgroup: Bluenoroff, Group 77, Labyrinth Chollima, Operation Troy, Operation GhostSecret, Operation AppleJeus, APT38, APT 38, Stardust Chollima, Whois Hacking Team, Zinc, Appleworm, Nickel Academy, APT-C-26

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

LAZARUS GROUP: A MAHJONG GAME PLAYED WITH DIFFERENT SETS OF TILES

Peter Káinai & Michal Poslušný
ESET, Czech Republic

[peter.kainai, michal.poslूसny@eset.cz](mailto:peter.kainai, michal.poslუსny@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 outbreak, the spear-phishing campaigns against US contractors), and kept up the pace at the turn of the year (the Android-portid payload, the bitcoin-oriented attacks, the Turkish phishing, 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 toolset appears to be modified with every attack. There are several static features that vary between the instances: dynamic Windows API resolution and the obfuscation of procedure and library names, the form of self-deleting batch files, the list of domains leveraged for fake TLS communications, the format strings included in TCP backdoors, the use of commercial packers, etc. The variety is so huge that it suggests that the Lazarus group may be split into multiple, independent, code-sharing cells. Our research investigated this idea further by 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 toolset that have not been publicly reported. Our study of these samples adds some interesting findings to the Lazarus puzzle: the very first iteration of WannaCry from 2016, in-the-wild experimentation with the malicious Java downloaders targeting multiple platforms, the use of a custom malware packer, and the presence of strange artifacts like Chinese language or South Korean cultural references. This paper will present previously unpublished details about the cyber-sabotage attack against an online casino in Central America from late 2017, and we will reveal the modus operandi of the Lazarus cell that was behind that 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. [vrt-090909_012](#) and [090909_00e111](#). However, the first published identification of the so-called Lazarus Group and its toolset was not until many years later in *Novotna's* extensive paper [2] in February 2016. The first mention of Lazarus at a *Virus Bulletin* conference was also in 2016, when Bartholomew and Guertsoo-Saade of *Kaspersky*

WWW.VIRUSBULLETIN.COM/CONFERENCE



Lab described the pseudo-backtivism tendencies of two famous Lazarus attacks [3]. Since 2017, especially after the WannaCry 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 why suspected Lazarus-related cases that we believe are not widely known.

ESET detects known Lazarus malware mostly as [Win32/NikSp8](#), [Win64/NikSp8](#), [Android/NikSp8](#) or [PowerShell/NikSp8](#). *US-CERT* and the FBI call the group Hidden Cobra [4].

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 arm 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-Keroboot](#) or [Win64/Spy-Keroboot](#).

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 statically similar files were later extensively described by *Novotna* [2], which named the attack ‘Operation Blockbuster’. Regarding the more common overlapping characteristics of binaries, this topic report was preceded by *Symantec's* report [7] by several months and, a year later, was followed by *Palo Alto Networks's* series of blog posts: *The Blockbuster Saga* [8], *The Blockbuster Saga Continues* [9] and *Operation 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.

¹For example, [Z:\MissionTeam_Project\2012.6 - HPTTP Troy\HngDbpwrWin32\Release\Payload\idb.job](#) or [D:\Work\CyMissionTeam\Payload\2012.11 - C2\Troy\Payload\Release\Payload\idb.job](#). We have found only two NikSp8 samples with paths: [E:\Connections\win64\blockbuster_gm\idb\blockbuster_idb.job](#); [D:\Source\Source_C\Work_Source\3T.1_Unicode\Server_label\Release\Server\idb.job](#).

²We used a very similar hunting method to the one described in Section 2.1 of [2] to find newly linked Lazarus executables.

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

CONJUNCTION OF RANGES:

win.exaramel (Back to overview)

Exaramel

Propose Change

aka: Crash,
CrashOverride
Actor(s): ELECTRUM



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

Industroyer is a malware that has been used in the power grid on December 23, 2015, a fifth of Kiev, the capital of Ukraine. It is the first ever known malware designed to attack electrical grids.

References

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

tion at this point.

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

```
► [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 [\(Back to overview\)](#)

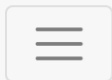


aka: APT 37, Group 123, Group123, Starcraft, Reaper, Reaper Group, Red Eyes, Ricochet Chollima, StarCraft, Operation Daybreak, Operation Erebus, Venus 121

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



Victoria Coach Station
Arrivals



national



Peter Kálnai

Senior Malware Researcher



Michal Poslušný

Malware Researcher



@ESETresearch

