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.

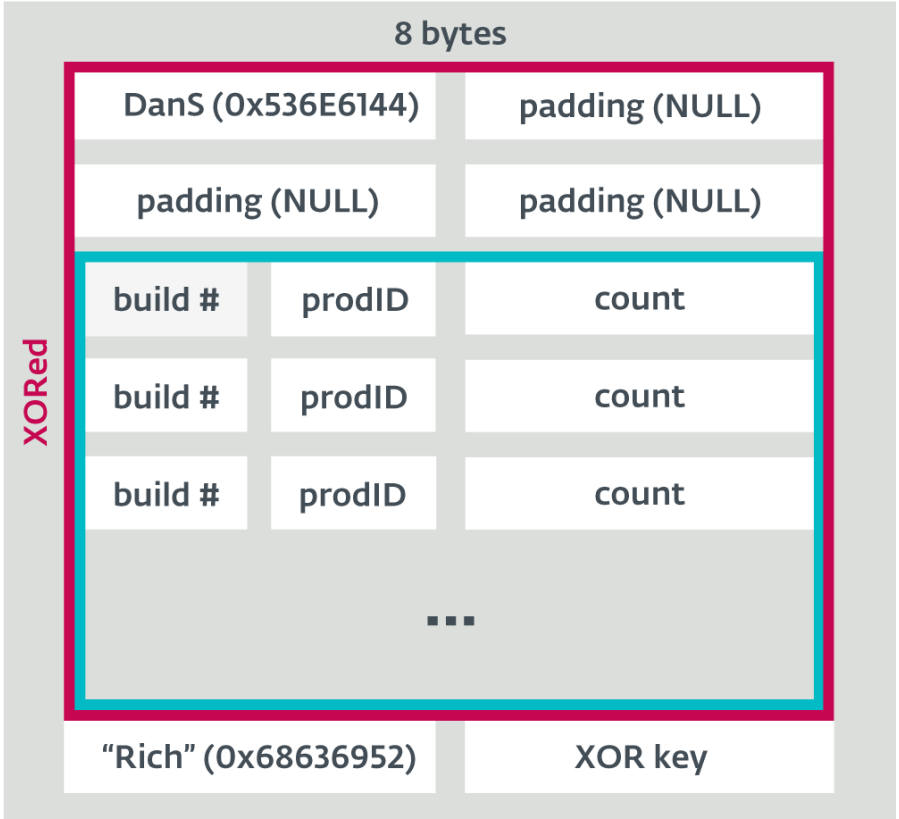
Description of Rich Headers

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

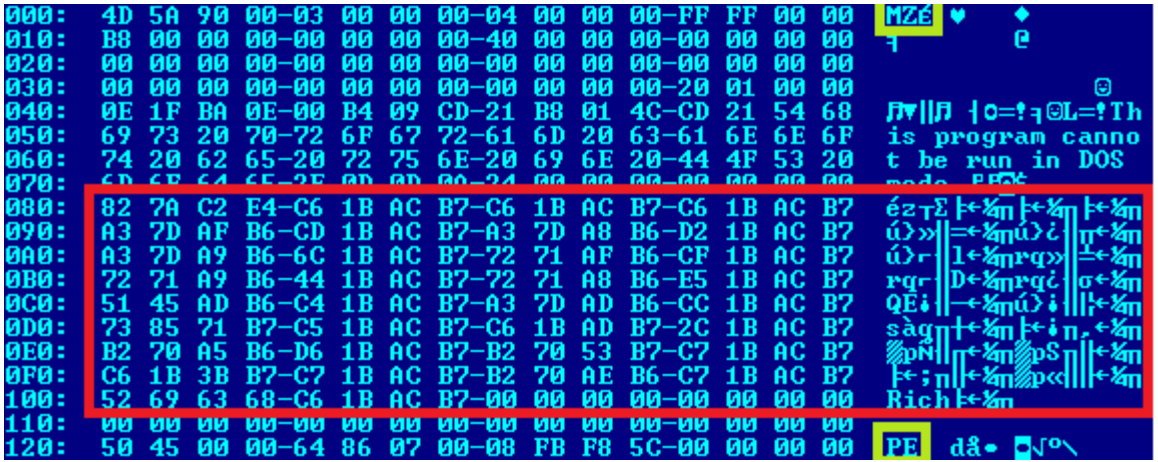
RH Structure

- 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



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)             ((PRODID) ((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;
```

'DanS'

```
enum {
    tagEndID    = 0x536e6144,
    tagBegID    = 0x68636952,
};
```

'Rich'

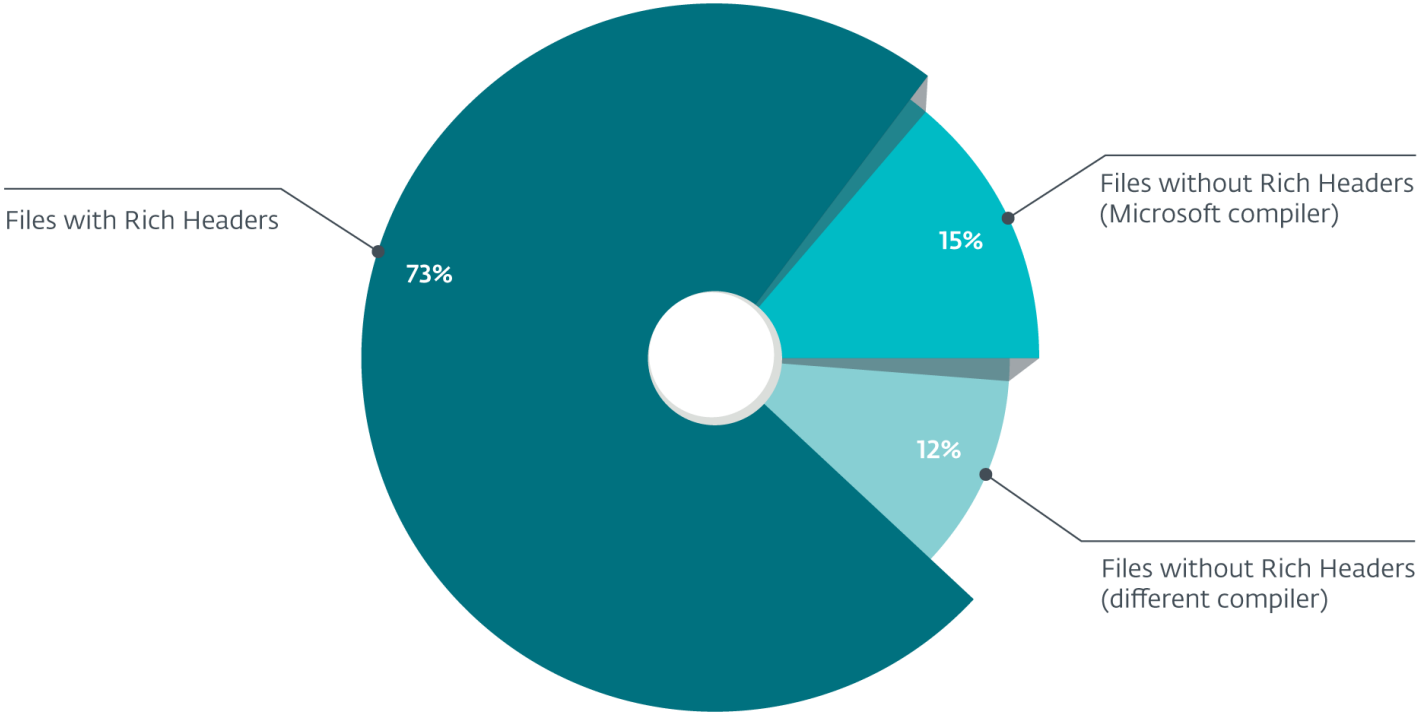
```
/*
  Normally, the DOS header and PE header are contiguous.  We place some data
  in between them if we find at least one tagged object file.
  struct {
    IMAGE_DOS_HEADER dosHeader;
    BYTE             rgbDosStub[N];         // MS-DOS stub
    PRODITEM         { tagEndID, 0 };       // start 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

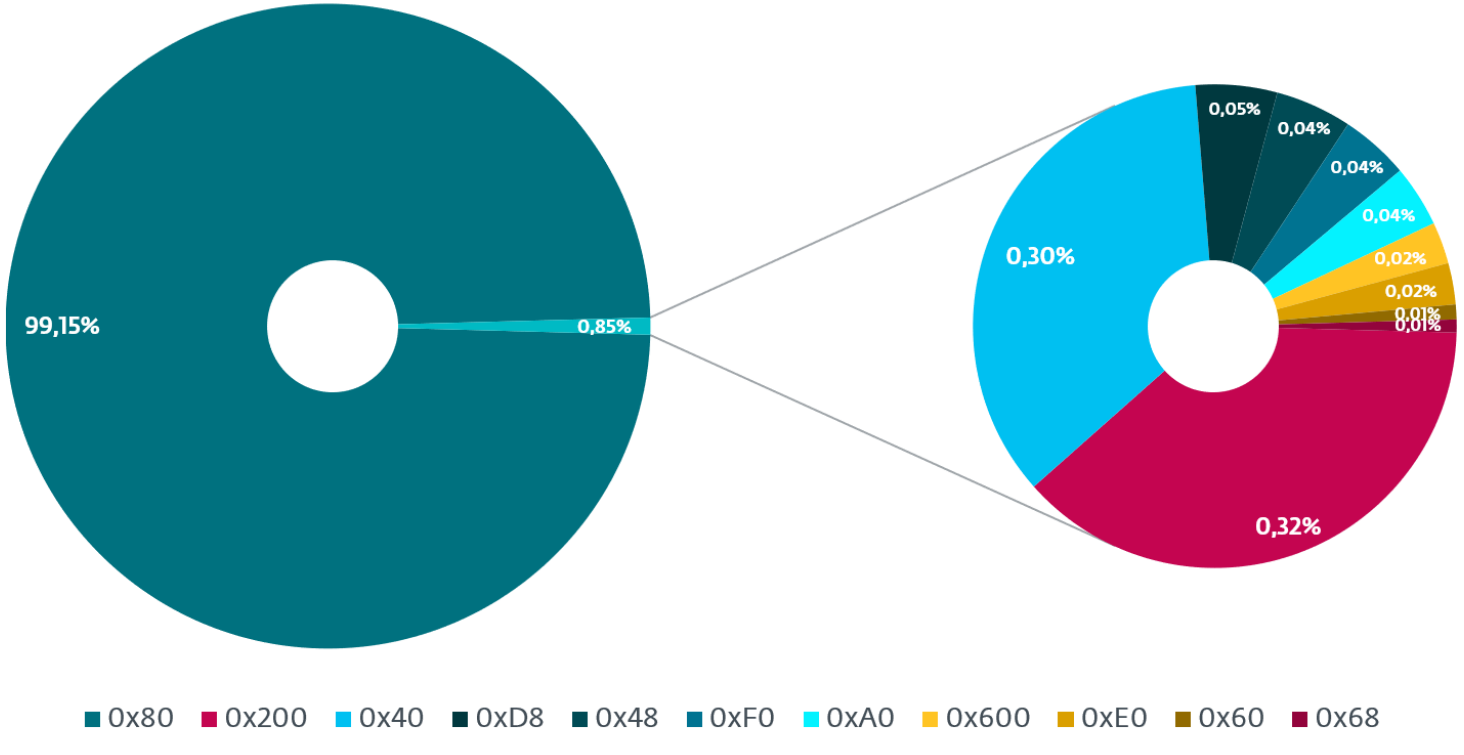
- XOR key generation VS 2019:

```
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


99,15%
0,85%
0,30%
0,32%
0,05%
0,04%
0,04%
0,04%
0,02%
0,02%
0,01%
0,01%
0x80
0x200
0x40
0xD8
0x48
0xF0
0xA0
0x600
0xE0
0x60
0x68

RH Levels of similarity

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

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):

	iPkxP4.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

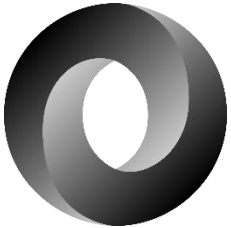
RH Anomalies

- 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.

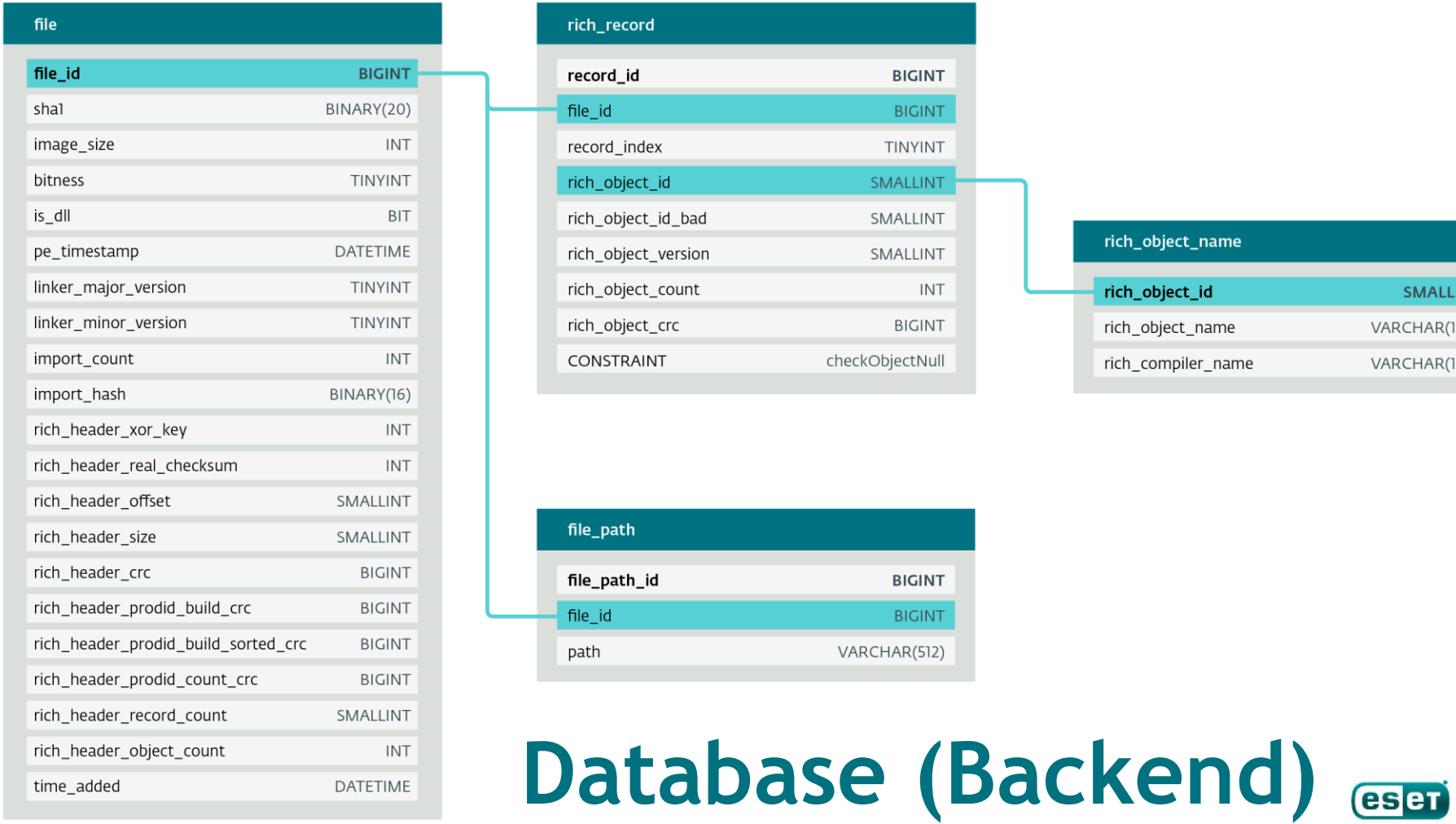
Tooling

Technologies



{fmt}

eset



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:

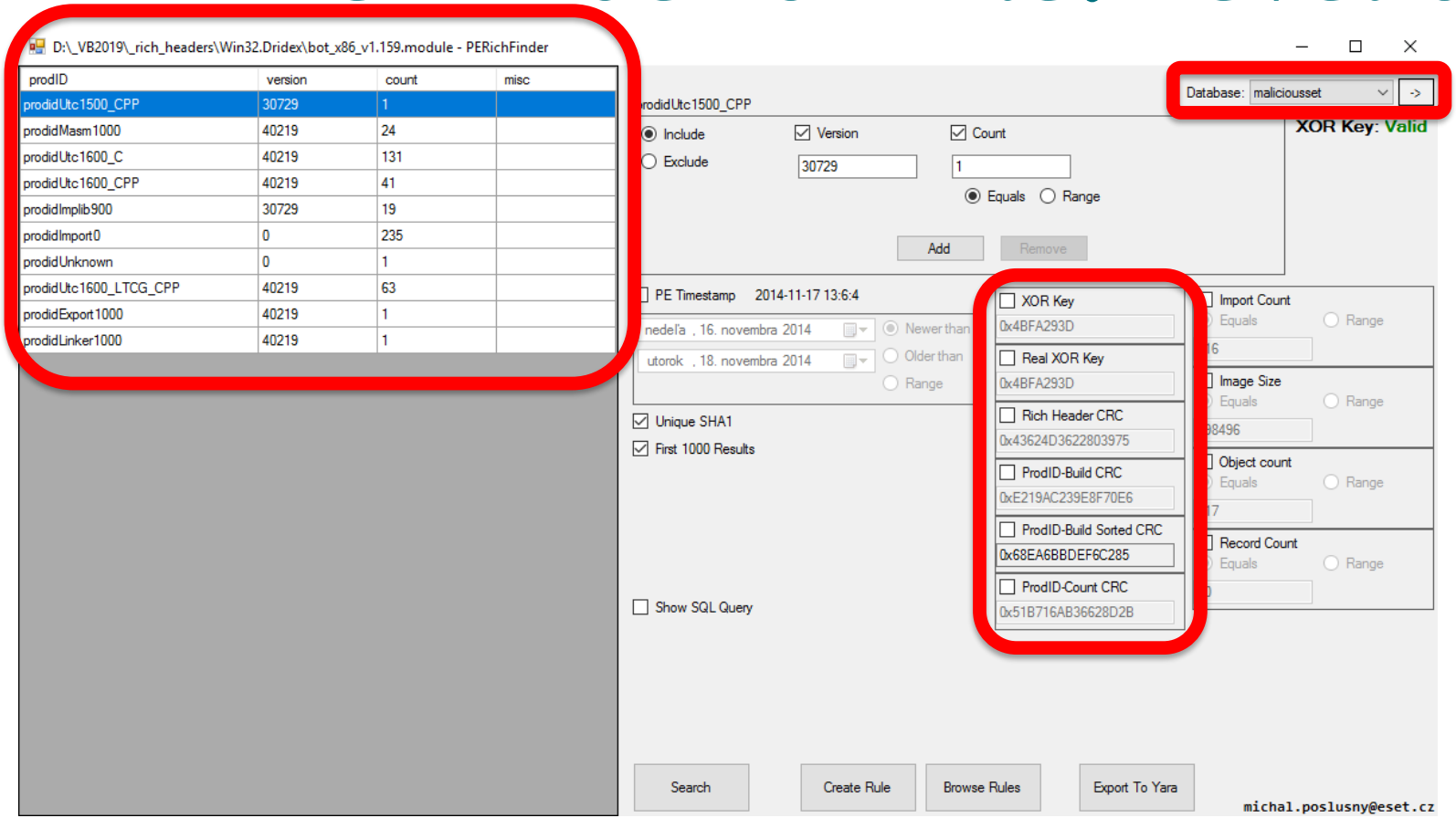
http://www.ntcore.com/files/richsign.htm

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.

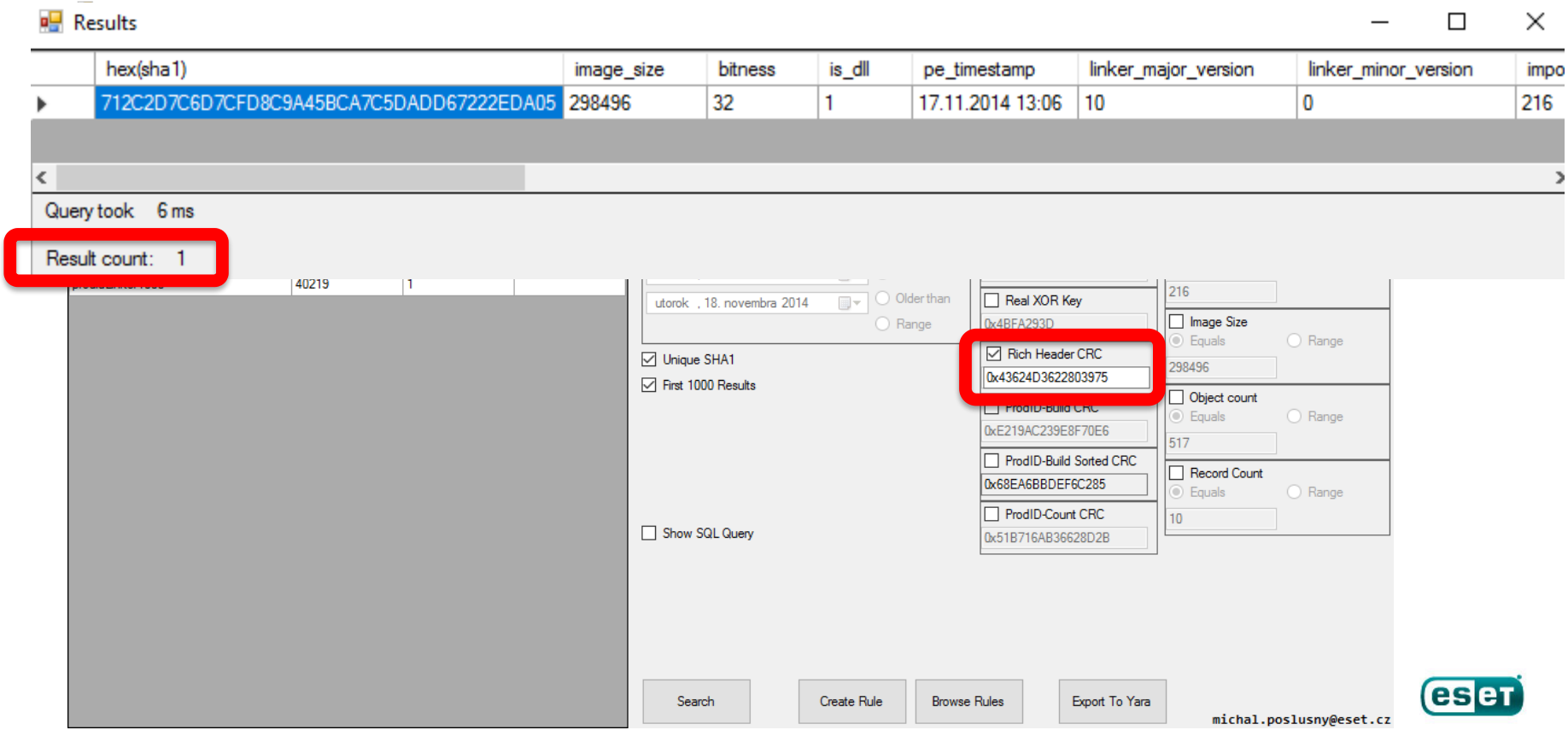
Example: pe.rich_signature.toolid(170, 40219) >= 99 and pe.rich_signature.toolid(170, 40219) <= 143

eset

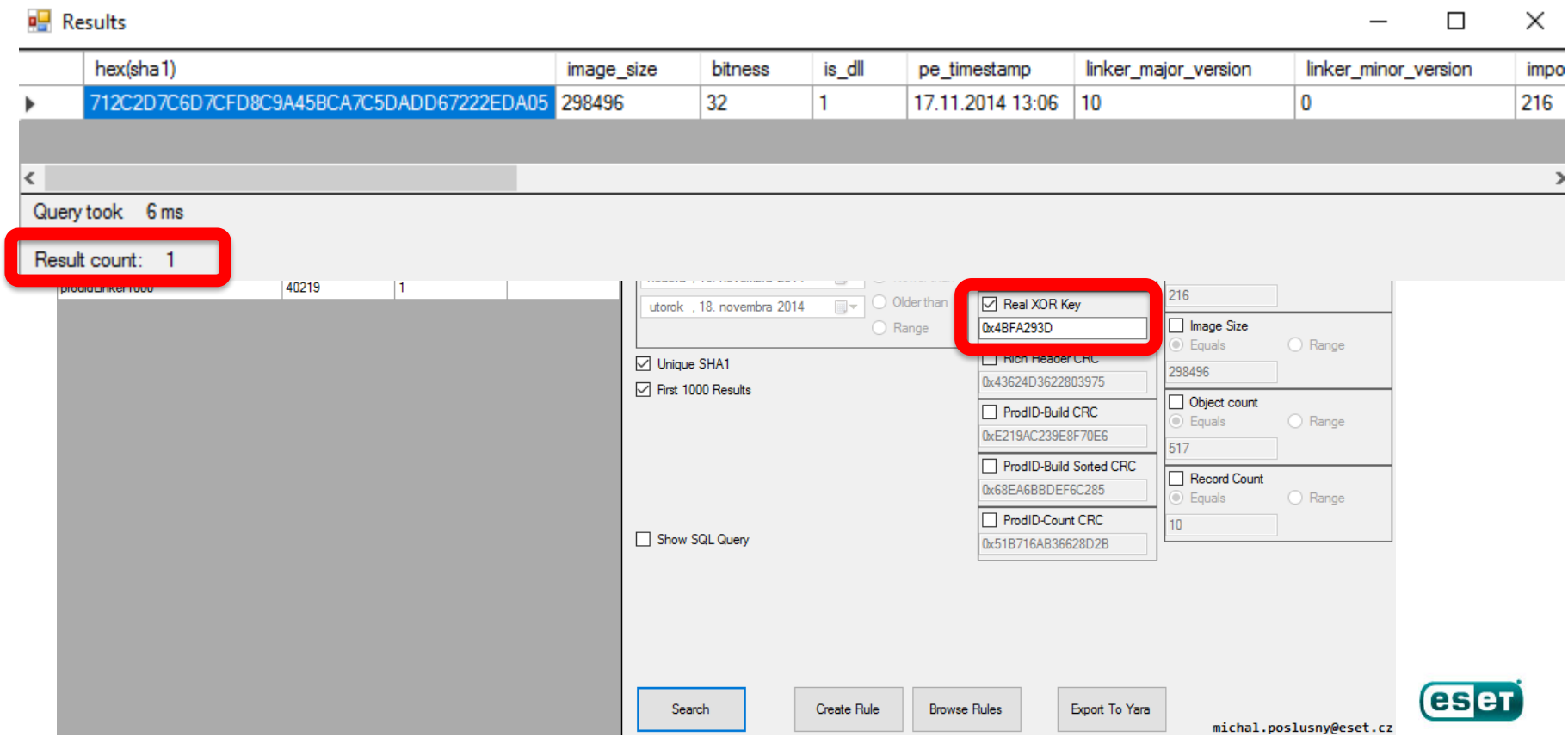
PERichFinder on File: Level 0



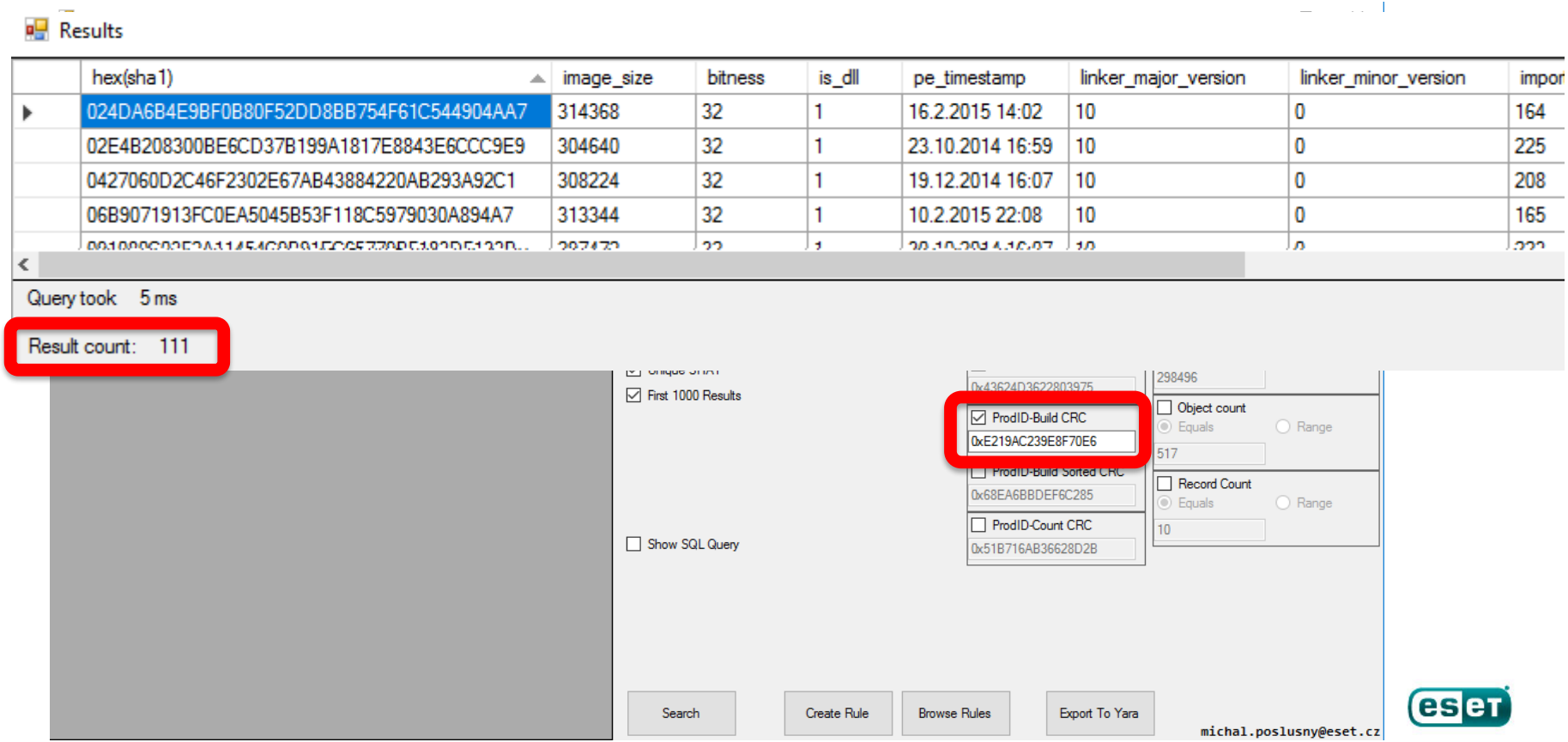
PERichFinder on File: Level 0



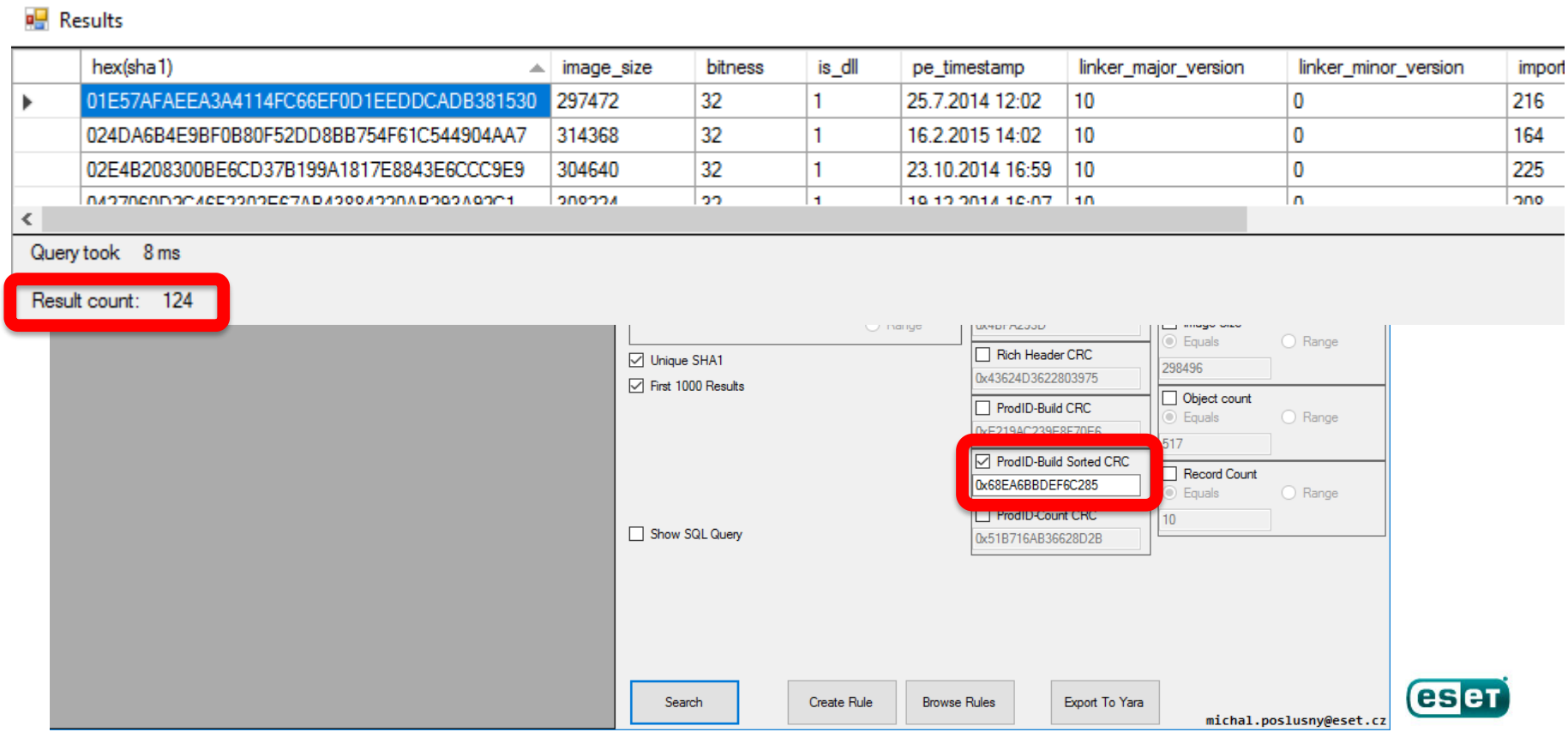
PERichFinder on File: Level 1



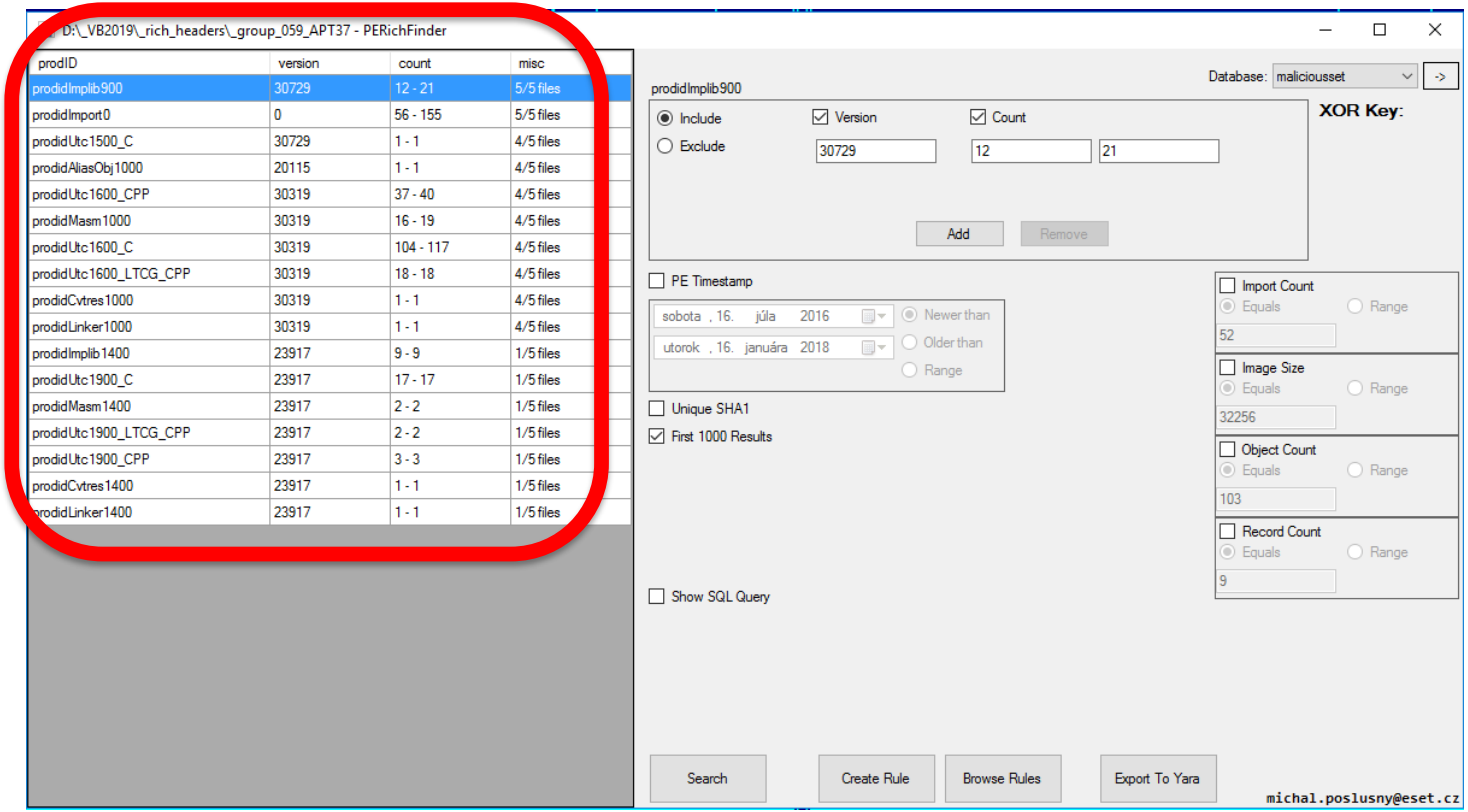
PERichFinder on File: Level 2



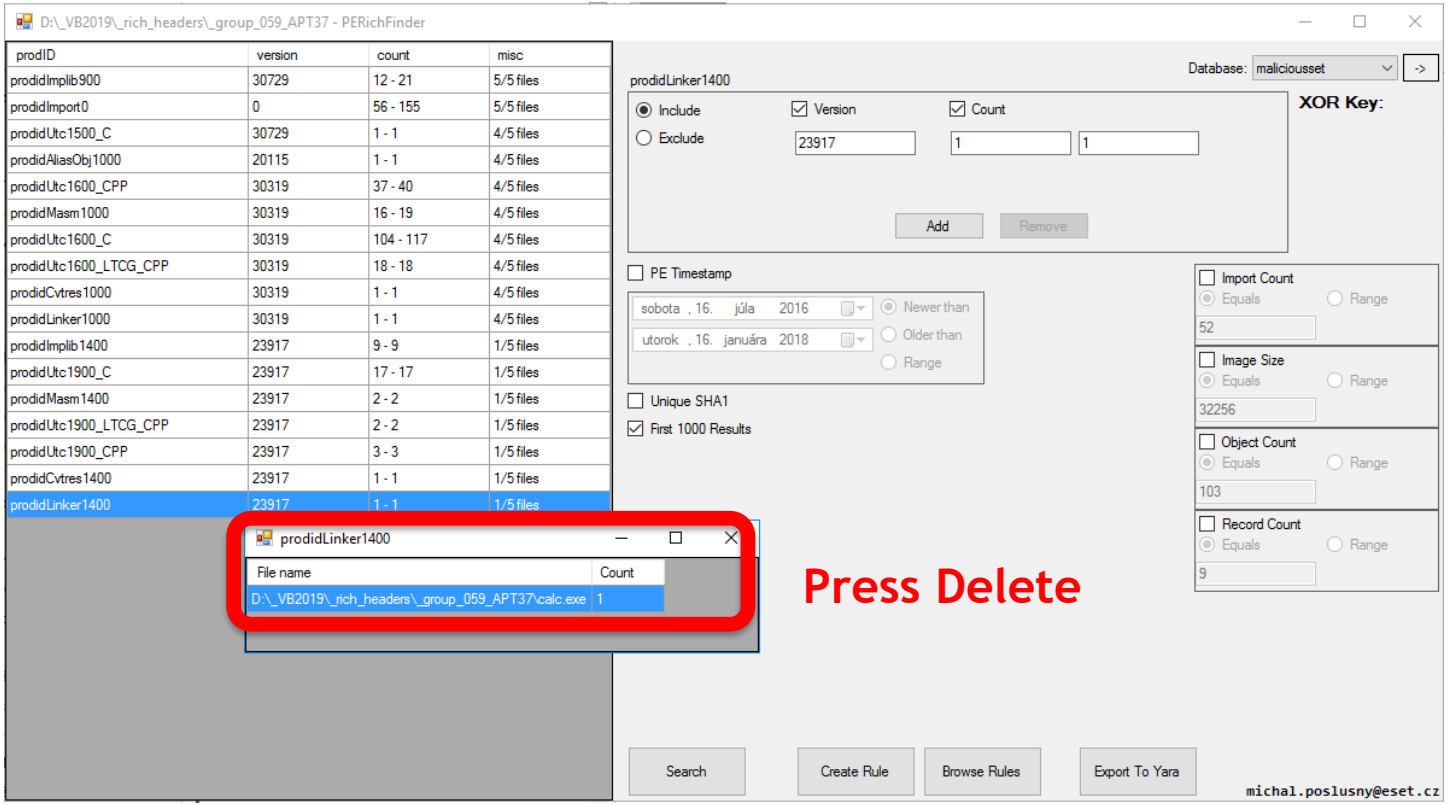
PERichFinder on File: Level 3



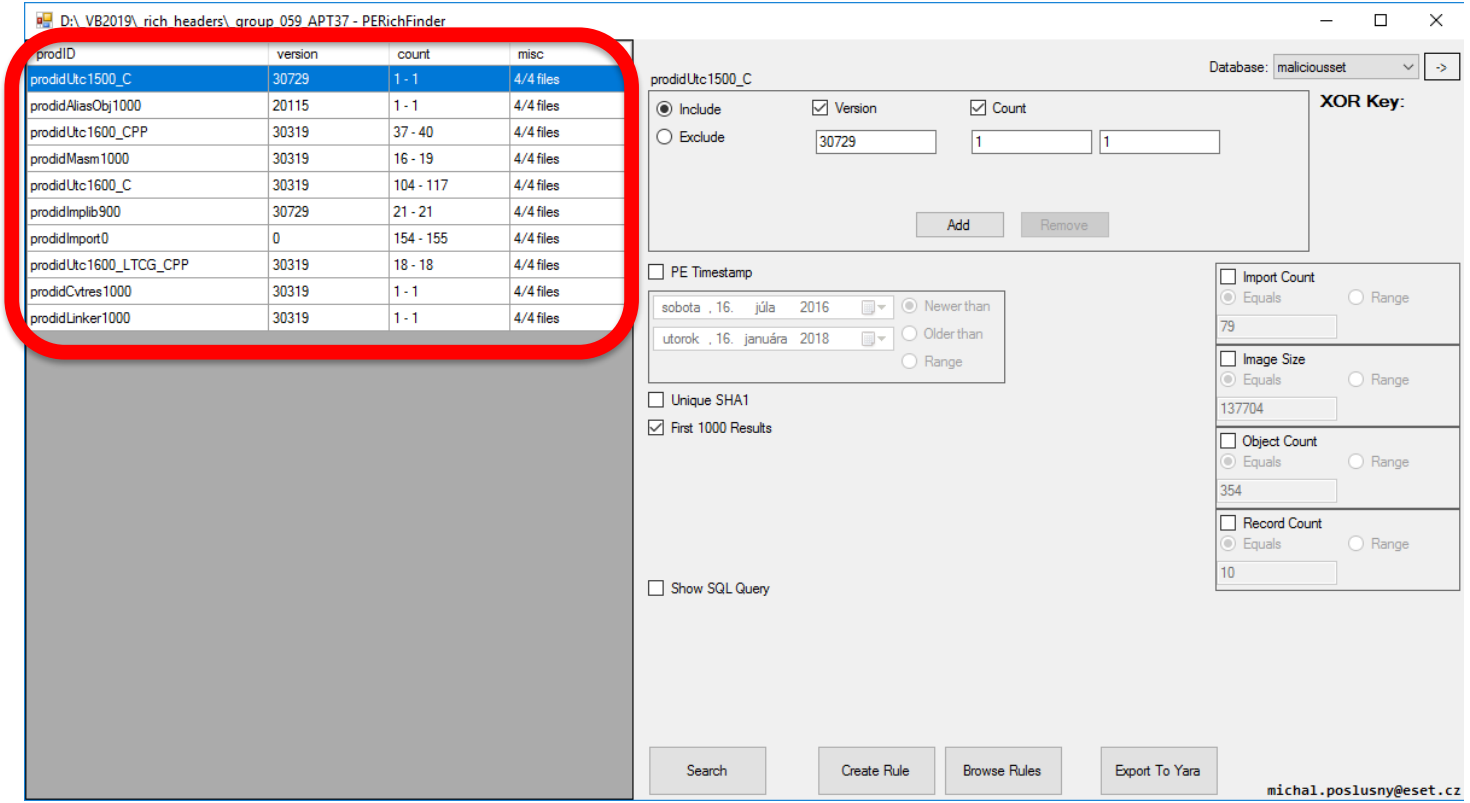
PERichFinder on Folder: Step 0



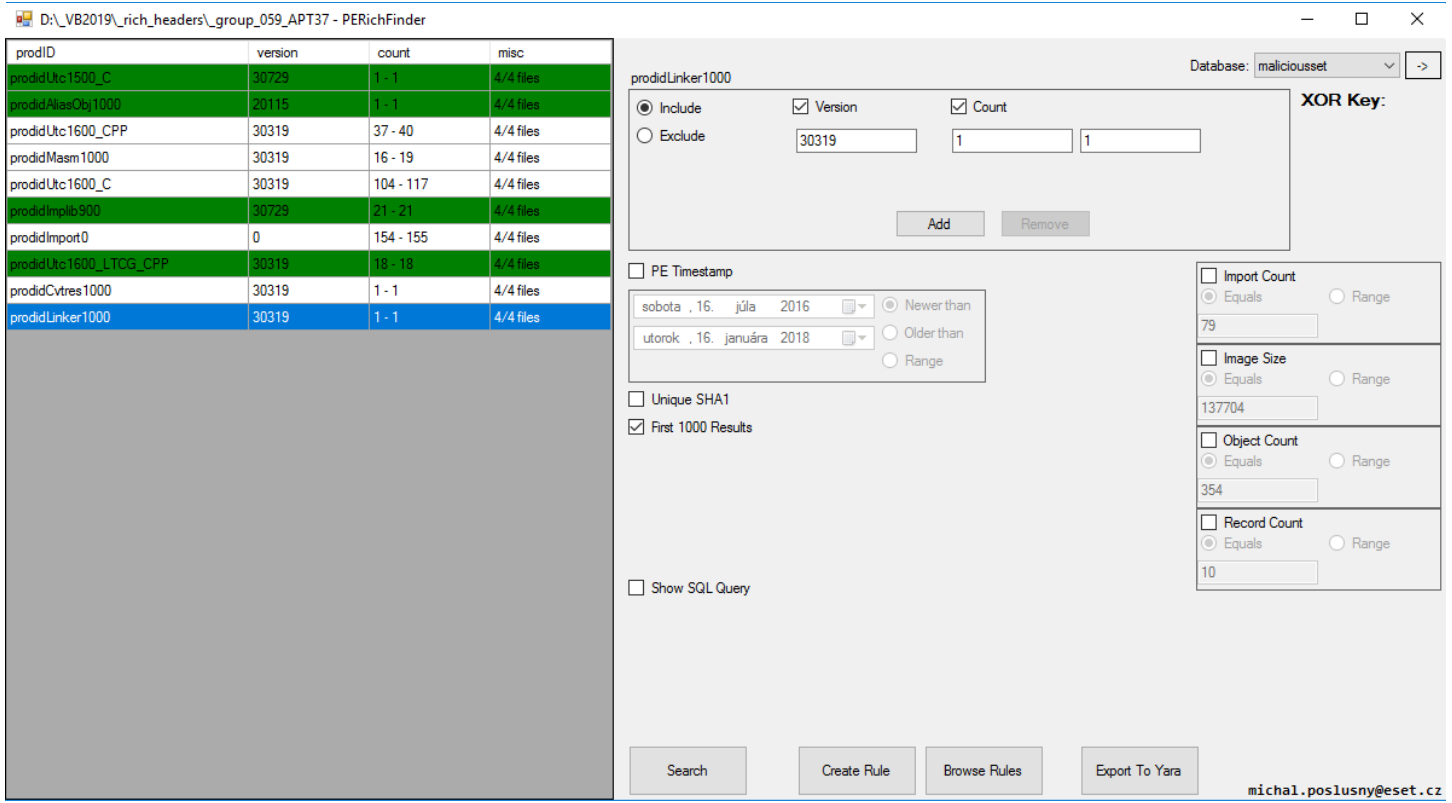
PERichFinder on Folder: Step 1



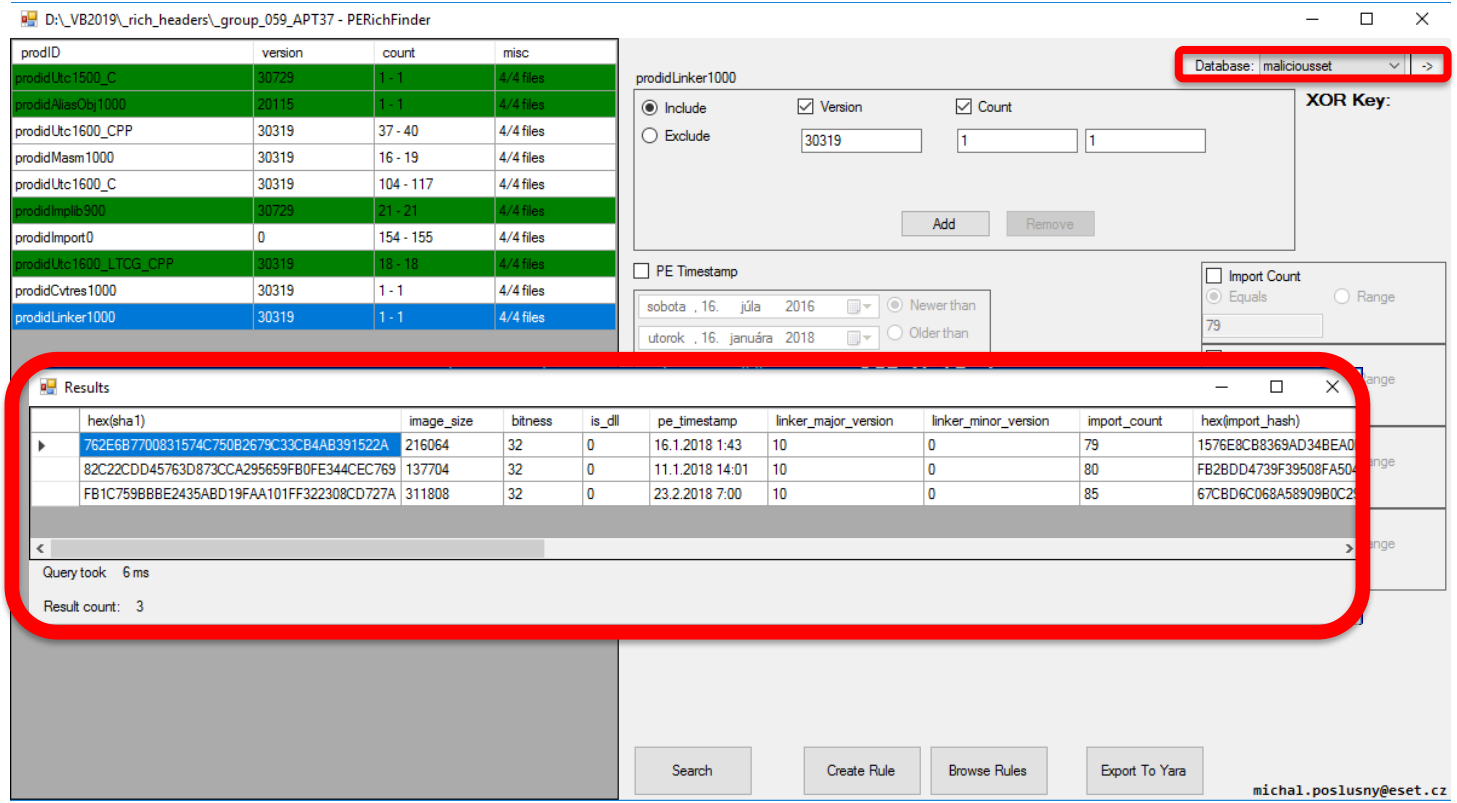
PERichFinder on Folder: Step 2



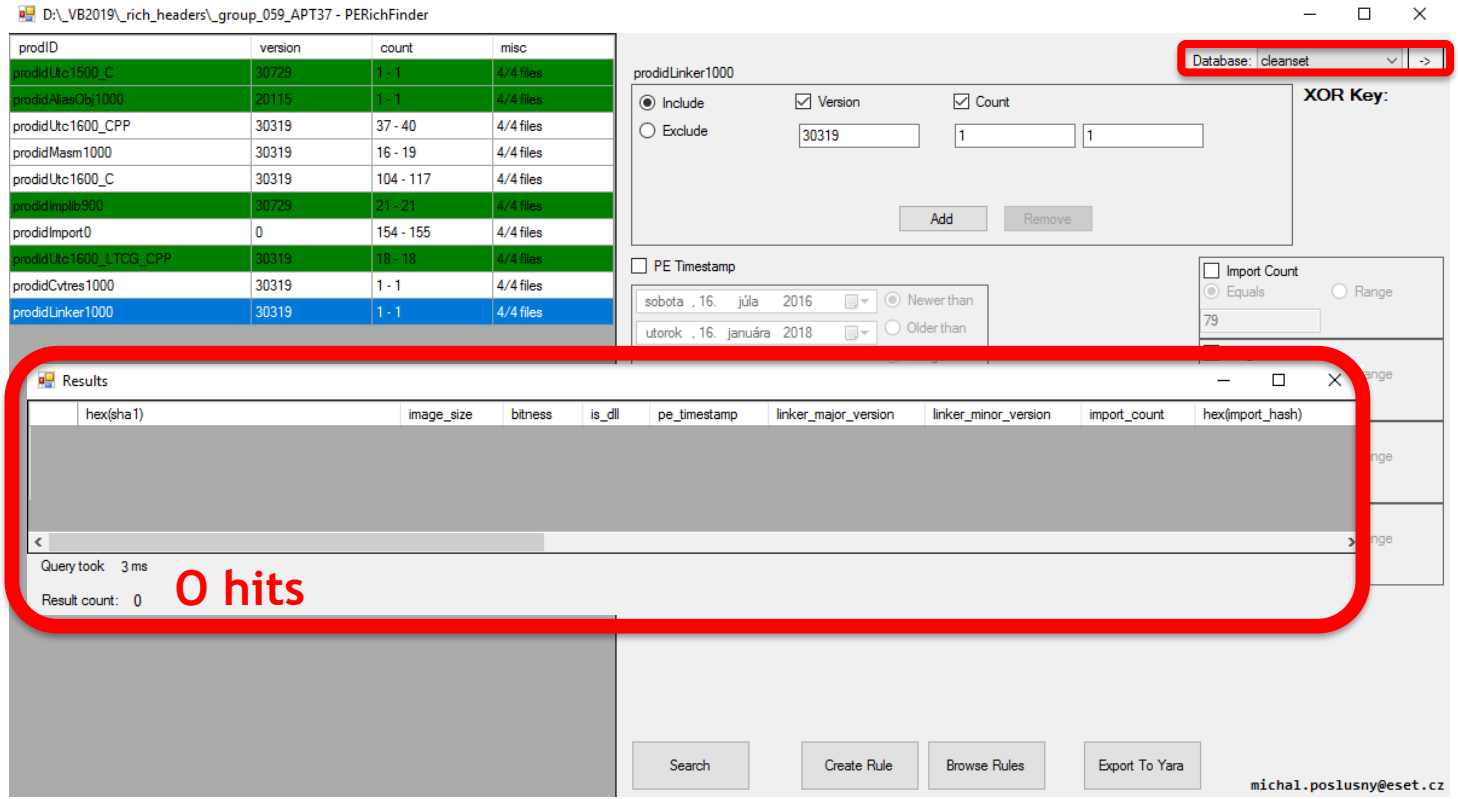
PERichFinder on Folder: Step 3



PERichFinder on Folder: Step 4



PERichFinder on Folder: Step 5



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

malpedia

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

WWW.VIRUSBULLETIN.COM/CONFERENCE vb 2018 MONTREAL 3 – 5 October 2018

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

Peter Kálnai & Michal Poslušný
ESET, Czech Republic

{peter.kalnai, 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 WannaCryptor 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 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 communication, 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 investigates 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 WannaCryptor 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. `netprov.dll` and `dvcmgmt.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-backtivism 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.

ESET detects known Lazarus malware mostly as Win32/NukeSped, Win64/NukeSped, Android/NukeSped or PowerShell/NukeSped. USCERT 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 armed forces and government targets, and ran between the year's 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 paths[1] – 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 statically similar files were later extensively described by *Novetta* [2][7], 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 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.

[1] For example, Z:\UMission\Team_Project\{2012.6 – }HTTP Troy\HttpDr0pper\Win32\Release\3PayloadDll.pdb or D:\Work\Op\Mission\TeamProject\{2012.11–12}\TDrop\Payload1\32\Release\Payload1l32.pdb. We have found only two NukeSped samples with paths: E:\ConstructSystem\widebot\bot_dll\Debug\bot_dll.pdb D:\Source\Source_C\Work_Source\T\3.1_Unicode\Server_\x64\Release\ServerDll.pdb

[2] We 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)

eset

Examples (1)

malpedia

win.industroyer (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
```

malpedia

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)
```

eset

Examples (1)

malpedia

win.industroyer (Back to overview)

Industroyer

Propose Change

aka: Crash, CrashOverride

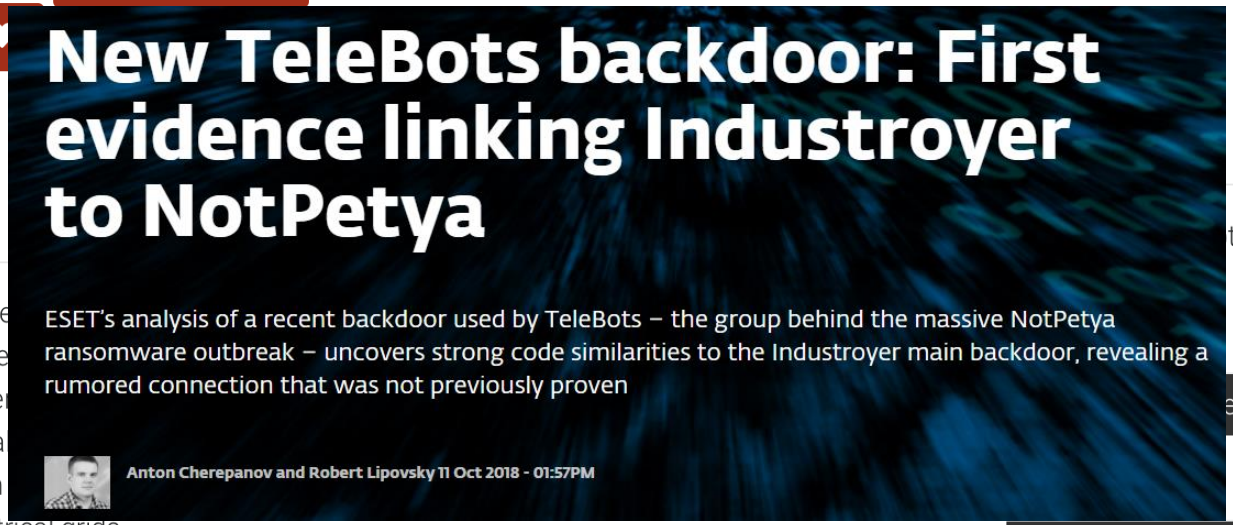
Actor(s): ELECTRUM

Industroyer is a malware
to have been used in the
power grid on Decembe
a fifth of Kiev, the capita
It is the first ever known
designed to attack electrical grids.

References

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

CONJUNCTION OF RANGES:



malpedia

win.exaramel (Back to overview)

Exaramel

Propose Change

tion at this point.

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

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

eset

Examples (2)

malpedia

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

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

…

It's Hacking Team, right?

⇦

No! Just a downloader of Win/Navrat

⇨

malpedia

APT37 (Back to overview)

aka: APT 37, Group 123, Group123, Starcruft, Reaper, Reaper Group, Red Eyes, Ricochet Chollima, StarCruft, 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.starcruft

Examples (3)

malpedia

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

eset

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