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
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)
• 2018: GReAT, The devil’s in the Rich header
• 2018: [K.-P.] Lazarus Group, VB2018 Montreal
• 2019: (July) Maksim Dubyak, 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
Microsoft Windows 2000 source code leaked

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

- Files with Rich Headers: 73%
- Files without Rich Headers (Microsoft compiler): 15%
- Files without Rich Headers (different compiler): 12%
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,
Samples with Identical RH have obviously the same XOR key

Malware packers, e.g. 0x8F44CEBF, 0xAEB29219, 0x8A17753B, 0xD4F1AE19, 0x887F83A7 (the complete RH varied!)

<table>
<thead>
<tr>
<th>Software</th>
<th>RH Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual Basic 6.0</td>
<td>0x886973F3, 0x8869808D, 0x88AA42CF, 0x88AA2A9D, 0x89A99A19, 0x88CECC0B, 0x8897EBCB, 0xAC72CCFA, 0x1AAAA993, 0xD05FECFB, 0x183A2CFD,</td>
</tr>
<tr>
<td>NSIS installer</td>
<td>0xD28650E9, 0x38BF1A05, 0x6A2AD175, 0xD246D0E9, 0x371742A2, 0xAB930178, 0x69EAD975, 0x69EB1175, 0xFB2414A1, 0xFB240DA1</td>
</tr>
<tr>
<td>MoleBox Ultra v4</td>
<td>0x8CABE24D</td>
</tr>
<tr>
<td>WinRar SFX</td>
<td>0xC47CACAA, 0xFDABBB1F, 0xD3254748, 0x557B8C97, 0x8DEFA739, 0x723F06DE, 0x16614BC7</td>
</tr>
<tr>
<td>Microsoft CAB File</td>
<td>0x43FACBB6</td>
</tr>
<tr>
<td>Autoit</td>
<td>0xBEAFE369, 0xC1FC1252, 0xCDA605B9, 0xA9CBC717, 0x8FEDAD28, 0x273B0B7D, 0xECFA7F86</td>
</tr>
</tbody>
</table>
Samples with Identical XOR key have obviously the same unsorted (ProdID, build) pairs

Win64/CoinMiner.DN (0x105E60A5B349F444):

<table>
<thead>
<tr>
<th>Videolan.exe</th>
<th>Update-19.1.10.exe</th>
</tr>
</thead>
<tbody>
<tr>
<td>prodidImport0</td>
<td>323</td>
</tr>
<tr>
<td></td>
<td>328</td>
</tr>
</tbody>
</table>

Win32/Pterodo (0x745E73E5045EE80E):

<table>
<thead>
<tr>
<th>iPkxP4.dll</th>
<th>Y9s9Ow.dll</th>
</tr>
</thead>
<tbody>
<tr>
<td>prodidMasm1210 (b40116)</td>
<td>9</td>
</tr>
<tr>
<td>prodidUtc1810_CPP (b40116)</td>
<td>119</td>
</tr>
<tr>
<td>prodidUtc1810_C (b40116)</td>
<td>24</td>
</tr>
<tr>
<td>prodidMasm1400 (b24123)</td>
<td>19</td>
</tr>
<tr>
<td>prodidUtc1900_CPP</td>
<td>23</td>
</tr>
<tr>
<td>prodidImport0</td>
<td>84</td>
</tr>
</tbody>
</table>
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)

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>prodidMasm1000 (b40219)</td>
<td>15</td>
</tr>
<tr>
<td>prodidUtc1600_C (b40219)</td>
<td>100</td>
</tr>
<tr>
<td>prodidImport0 (0)</td>
<td>130</td>
</tr>
<tr>
<td>prodidUtc1600_C (b40219)</td>
<td>prodidMasm1000 (b40219)</td>
</tr>
<tr>
<td>prodidMasm1000 (b40219)</td>
<td>9</td>
</tr>
<tr>
<td>prodidImport0 (0)</td>
<td>131</td>
</tr>
</tbody>
</table>

Win/GreyEnergy

<table>
<thead>
<tr>
<th>Zlib_x86.dll</th>
<th>Zlib_x64.dll</th>
</tr>
</thead>
<tbody>
<tr>
<td>prodidMasm1000 (b40219)</td>
<td>17</td>
</tr>
<tr>
<td>prodidUtc1600_C (b40219)</td>
<td>110</td>
</tr>
<tr>
<td>prodidImport0</td>
<td>87-88</td>
</tr>
<tr>
<td>prodidUtc1600_LTCG_CPP (b40219)</td>
<td>22</td>
</tr>
</tbody>
</table>
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

Python

VirusTotal yara

Apache Kafka

Visual Studio 2019

C#

fmt

C++
### Database (Backend)

#### File Table
- **file_id** (BIGINT)
  - **shl** (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 Table
- **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)

#### Rich Object Name Table
- **rich_object_id** (SMALLINT)
  - **rich_object_name** (VARCHAR(100))
  - **rich_compiler_name** (VARCHAR(100))

#### File Path Table
- **file_path_id** (BIGINT)
  - **file_id** (BIGINT)
  - **path** (VARCHAR(512))
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
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 r ich_s ignature toolid(170, 40219) >= 99 and pe r ich_s ignature toolid(170, 40219) <= 143
PERichFinder on File: Level 0
**PERichFinder on File: Level 0**

<table>
<thead>
<tr>
<th>hex(sha1)</th>
<th>image_size</th>
<th>bitness</th>
<th>is_dll</th>
<th>pe_timestamp</th>
<th>linker_major_version</th>
<th>linker_minor_version</th>
<th>impo</th>
</tr>
</thead>
<tbody>
<tr>
<td>712C2D7C6D7CFD8C945B2A7C5DADD672226DA05</td>
<td>298496</td>
<td>32</td>
<td>1</td>
<td>17.11.2014 13:06</td>
<td>10</td>
<td>0</td>
<td>216</td>
</tr>
</tbody>
</table>

**Result count:** 1

[Image of PERichFinder interface with highlighted table data and query results]
PERichFinder on File: Level 1
### PERichFinder on File: Level 2

<table>
<thead>
<tr>
<th>hex(sha1)</th>
<th>image_size</th>
<th>bitness</th>
<th>is_dll</th>
<th>pe_timestamp</th>
<th>linker_major_version</th>
<th>linker_minor_version</th>
<th>import</th>
</tr>
</thead>
<tbody>
<tr>
<td>024DA6B4E9BF0B80F52DD88B754F61C544904AA7</td>
<td>314368</td>
<td>32</td>
<td>1</td>
<td>16.2.2015 14:02</td>
<td>10</td>
<td>0</td>
<td>164</td>
</tr>
<tr>
<td>02E4B208300BE6CD37B199A1817E8843E6C6CC9E9</td>
<td>304640</td>
<td>32</td>
<td>1</td>
<td>23.10.2014 16:59</td>
<td>10</td>
<td>0</td>
<td>225</td>
</tr>
<tr>
<td>04270606D2C46F2320E67A43884220AB293A92C1</td>
<td>308224</td>
<td>32</td>
<td>1</td>
<td>19.12.2014 16:07</td>
<td>10</td>
<td>0</td>
<td>208</td>
</tr>
<tr>
<td>06B9071913FC0EA5045B53F118C5979030A894A7</td>
<td>313344</td>
<td>32</td>
<td>1</td>
<td>10.2.2015 22:08</td>
<td>10</td>
<td>0</td>
<td>165</td>
</tr>
</tbody>
</table>

**Query took**: 5 ms

**Result count**: 111

- **ProdID-Build CRC**: 0xE219AC235EBF70E6
- **ProdID-Build CRC**: 0x68EA5EBEDF6CC25
- **ProdID-Count CRC**: 0x51E716A936628D2B

![Image of PERichFinder interface with highlighted data and options](image-url)
PERichFinder on File: Level 3

<table>
<thead>
<tr>
<th>hex(sha1)</th>
<th>image_size</th>
<th>bitness</th>
<th>is_dll</th>
<th>pe_timestamp</th>
<th>linker_major_version</th>
<th>linker_minor_version</th>
<th>import</th>
</tr>
</thead>
<tbody>
<tr>
<td>01E57AFAEEA3A4114FC66EF0D1EEDDCADB381530</td>
<td>297472</td>
<td>32</td>
<td>1</td>
<td>25.7.2014 12:02</td>
<td>10</td>
<td>0</td>
<td>216</td>
</tr>
<tr>
<td>024DA6B4E9BF0B80F52DD8B754F61C544904AA7</td>
<td>314368</td>
<td>32</td>
<td>1</td>
<td>16.2.2015 14:02</td>
<td>10</td>
<td>0</td>
<td>164</td>
</tr>
<tr>
<td>02E4828300BE6CD37B199A181E888436CC9E9</td>
<td>304640</td>
<td>32</td>
<td>1</td>
<td>23.10.2014 16:59</td>
<td>10</td>
<td>0</td>
<td>225</td>
</tr>
</tbody>
</table>

Result count: 124
### PERichFinder on Folder: Step 0

**Table:**

<table>
<thead>
<tr>
<th>prodId</th>
<th>version</th>
<th>count</th>
<th>misc</th>
</tr>
</thead>
<tbody>
<tr>
<td>prodIdImp800</td>
<td>30729</td>
<td>12 - 21</td>
<td>5/5/80a</td>
</tr>
<tr>
<td>prodIdImp0</td>
<td>0</td>
<td>57 - 80</td>
<td>5/5/80a</td>
</tr>
<tr>
<td>prodIdC0</td>
<td>20115</td>
<td>1 - 3</td>
<td>4/4/80a</td>
</tr>
<tr>
<td>prodId1600</td>
<td>30319</td>
<td>39 - 40</td>
<td>4/4/80a</td>
</tr>
<tr>
<td>prodId1600_C</td>
<td>30319</td>
<td>14 - 19</td>
<td>4/4/80a</td>
</tr>
<tr>
<td>prodId1600_CCPP</td>
<td>30319</td>
<td>104 - 117</td>
<td>4/4/80a</td>
</tr>
</tbody>
</table>

**GUI:**

- **Database:** malicious
- **XOR Key:** 21
- **PE Timestamp:**
  - chebota, 16. jula 2016
  - utorak, 16. januara 2018
- **Unique SHA1**
- **First 1000 Results**
- **Show SQL Query**
PERichFinder on Folder: Step 1

Press Delete
PERichFinder on Folder: Step 2
PERichFinder on Folder: Step 3

<table>
<thead>
<tr>
<th>prodID</th>
<th>version</th>
<th>count</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>prodICtrna1000</td>
<td>30319</td>
<td>1-1</td>
<td>4/4</td>
</tr>
<tr>
<td>prodICtrnai1000</td>
<td>30319</td>
<td>1-1</td>
<td>4/4</td>
</tr>
<tr>
<td>prodICtrna1000_C</td>
<td>30319</td>
<td>1-1</td>
<td>4/4</td>
</tr>
<tr>
<td>prodICtrni1000</td>
<td>30319</td>
<td>1-1</td>
<td>4/4</td>
</tr>
<tr>
<td>prodICtrni1000_C</td>
<td>30319</td>
<td>1-1</td>
<td>4/4</td>
</tr>
<tr>
<td>prodILink000</td>
<td>30319</td>
<td>1-1</td>
<td>4/4</td>
</tr>
<tr>
<td>prodILink1000</td>
<td>30319</td>
<td>1-1</td>
<td>4/4</td>
</tr>
</tbody>
</table>

Database: multilist

XOR Key: 1

Include: Version, Count
Exclude: 30319, 1, 1

PE Timestamp:
- Sabota, 16. jula 2016
- Utork, 16. januara 2016

Unique SHA1

File 1000 Results

Show SQL Query

Search, Create Rule, Browse Rules, Export To Yara

michal.poisliny@eset.cz
**PERichFinder on Folder: Step 4**

---

![PERichFinder interface](image)

### Table: Results

<table>
<thead>
<tr>
<th>Hash (SHA-1)</th>
<th>Image Size</th>
<th>bitsize</th>
<th>is_SI</th>
<th>PE Timestamp</th>
<th>Linker_major_version</th>
<th>Linker_minor_version</th>
<th>Import_count</th>
<th>HexImport_hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>622037700381217AC505C2673C3CB430B3152A</td>
<td>216064</td>
<td>32</td>
<td>0</td>
<td>16.1.2018 1:43</td>
<td>10</td>
<td>0</td>
<td>79</td>
<td>157BECB3B393A0D4BEA0</td>
</tr>
<tr>
<td>0352222DDB47630873CA2295095B08FE343C6C765</td>
<td>137704</td>
<td>32</td>
<td>0</td>
<td>11.1.2018 14:21</td>
<td>10</td>
<td>0</td>
<td>80</td>
<td>FB22CE7F3995F8APB0</td>
</tr>
<tr>
<td>F17C5F58BED285354D135F610F58823200CD727X</td>
<td>311000</td>
<td>32</td>
<td>0</td>
<td>23.2.2018 7:00</td>
<td>10</td>
<td>0</td>
<td>85</td>
<td>67C2E0C650880603C2</td>
</tr>
</tbody>
</table>

---

Query look: 6ms
Result count: 3

---

* michal.poslusny@eset.cz
PERichFinder on Folder: Step 5

0 hits
SQL & YARA rules

```
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 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
Main subgroups
1) x86 VS98 + x64 VS2010
2) x86 VS2010 + x64 VS2010

..and anomalies...

VS98 + VS2013, not expected VS98 + VS2010

Lazarus Group

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)
### Examples (1)

**CONJUNCTION OF RANGES:**

<table>
<thead>
<tr>
<th>Object Count</th>
<th>350..460</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utc1810_CPP(40116)</td>
<td>120..121</td>
</tr>
<tr>
<td>Masm1400 (24123)</td>
<td>17...18</td>
</tr>
<tr>
<td>Utc1900_CPP(24123)</td>
<td>29...33</td>
</tr>
<tr>
<td>Import0</td>
<td>100..200</td>
</tr>
<tr>
<td>Cvtres1400 (24210)</td>
<td>1....1</td>
</tr>
</tbody>
</table>

**Industroyer**

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**

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

Examples (1)

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

References

Examples (2)

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

...It’s Hacking Team, right?

No! Just a downloader of Win/Navrat
Examples (3)

- Suspicious File called etwdrv.dll
- Export Name: LCrPsdNew.dll

- 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