SANS DFIR
DIGITAL FORENSICS & INCIDENT RESPONSE

Human Fingerprints and Cyber Threat Intelligence

Tobias Johansson || Robert M. Lee

Outline

- The Adversary
- Human Fingerprints on Malware
- Header Metadata
- Code Reuse
- Configuration Data
- Conclusion

The Adversary

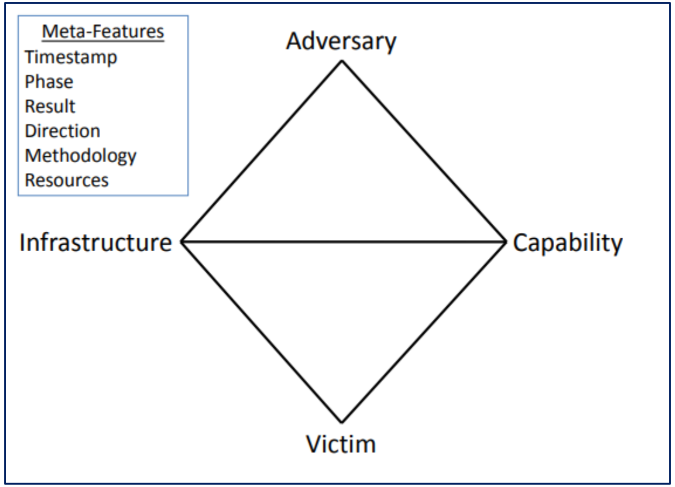
Increase Adversary knowledge

SANS DFIR

DIGITAL FORENSICS & INCIDENT RESPONSE

Diamond Model

- The Diamond Model's core features are:
  - Adversary, Capability (TTPs), Infrastructure, and Victim
- The Diamond Model's meta-features are:
  - Timestamp (start and end), phase, result, direction, methodology, and resources
- Each core feature and its meta-features should have a confidence value
  - Definition up to each user; utilize at minimum High, Moderate, and Low weightings



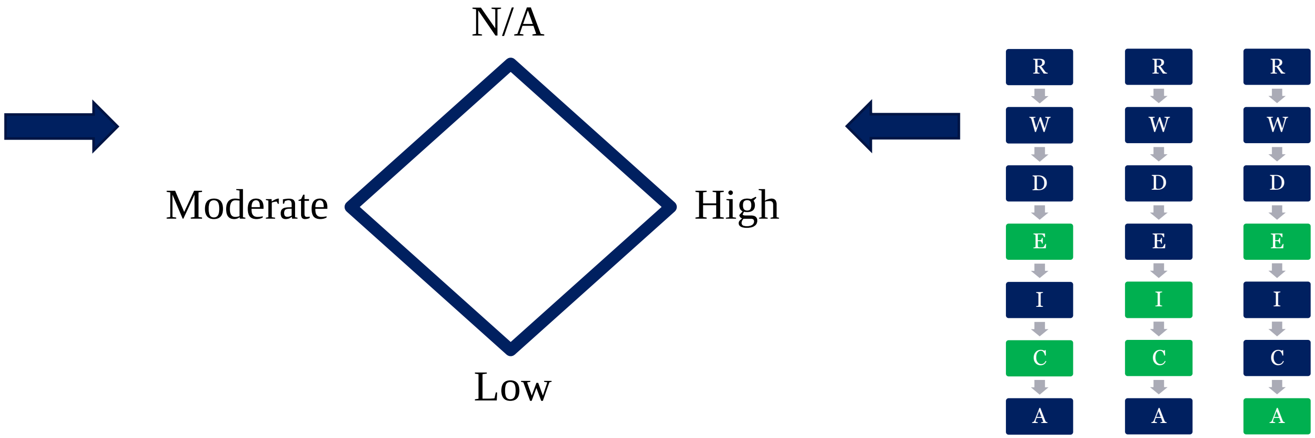
Adversary Event is formally defined as "E"

$$
\begin{aligned}
\boldsymbol{E} = \langle & \langle Adversary, Confidence_{adversary} \rangle, \\
& \langle Capability, Confidence_{capability} \rangle, \\
& \langle Infrastructure, Confidence_{infrastructure} \rangle, \\
& \langle Victim, Confidence_{victim} \rangle, \\
& \langle Timestamp_{start}, Confidence_{timestamp_{start}} \rangle, \\
& \langle Timestamp_{end}, Confidence_{timestamp_{end}} \rangle, \\
& \langle Phase, Confidence_{phase} \rangle, \\
& \langle Result, Confidence_{result} \rangle, \\
& \langle Direction, Confidence_{direction} \rangle, \\
& \langle Methodology, Confidence_{methodology} \rangle, \\
& \langle Resources, Confidence_{resources} \rangle \rangle
\end{aligned}
$$

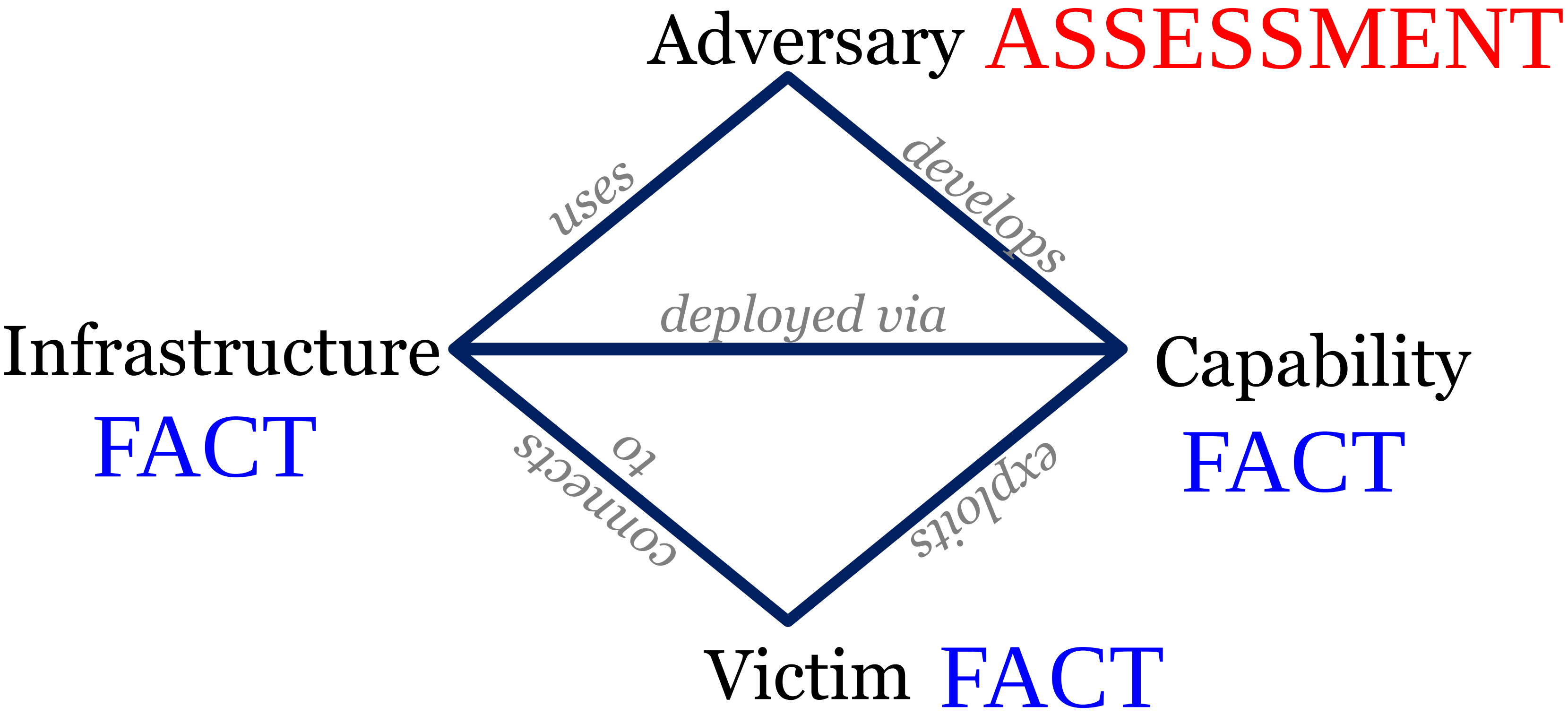
Creating an Activity Group

- Six distinct steps to creating an Activity Group
  - Step 1: Analytical Problem (define what you want to solve, your intelligence requirement)
  - Step 2: Feature Selection (event features and weighting of what’s important to you)
  - Step 3: Creation (analyze events/intrusions and compare against the model to cluster)
  - Step 4: Growth (compare new events and classify them into the Activity Group)
  - Step 5: Analysis (analyze the Activity Group itself to address the Analytical Problem)
  - Step 6: Redefinition (redefine the model as your needs change and more to new cluster)

Problem: Identify unique adversary tradecraft to create scalable detections

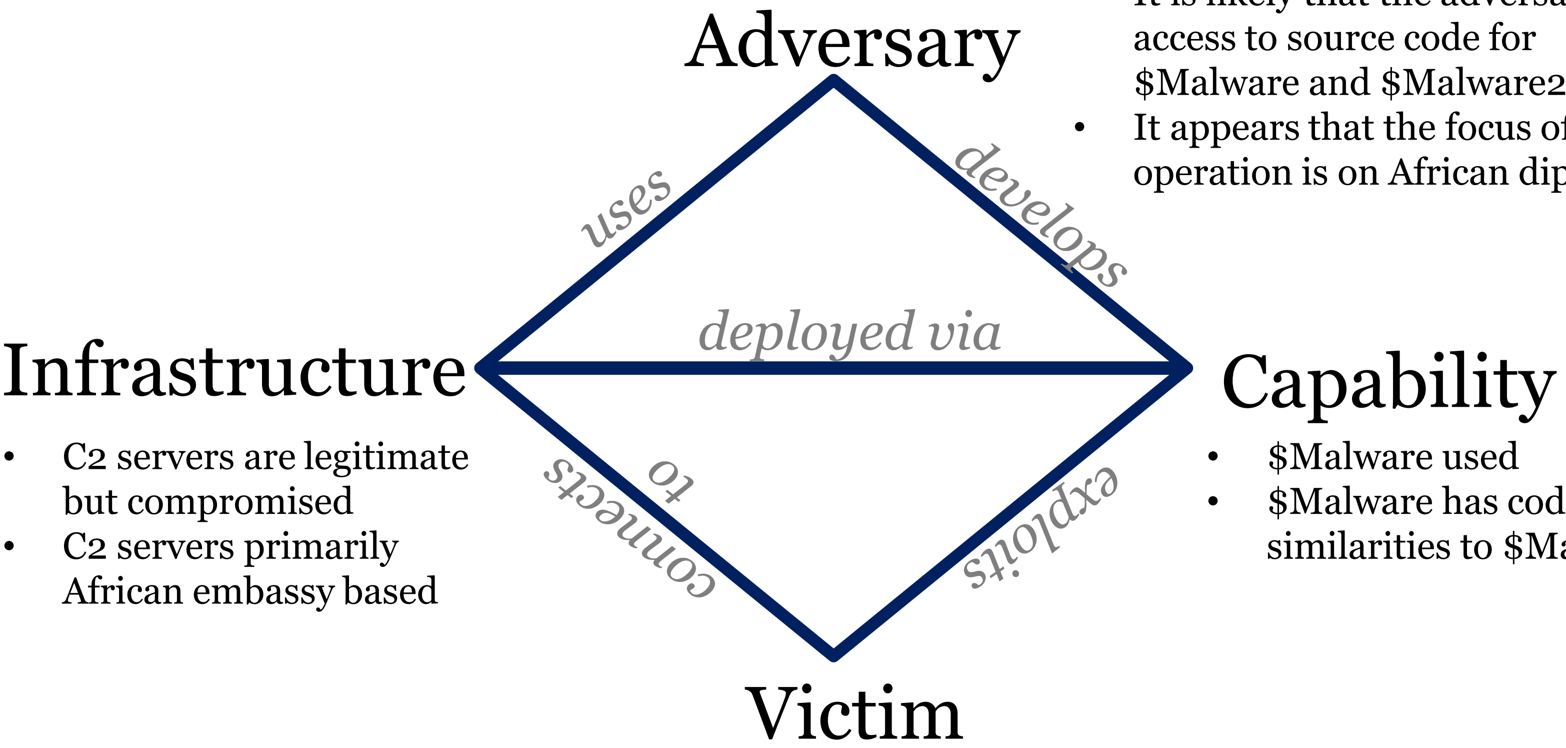


Diamond Model and Analytic Findings



Adversary Assessment Examples

- It is likely that the adversary has access to source code for $Malware and $Malware2
- It appears that the focus of the operation is on African diplomats



- C2 servers are legitimate but compromised
- C2 servers primarily African embassy based

- $Malware used
- $Malware has code similarities to $Malware2

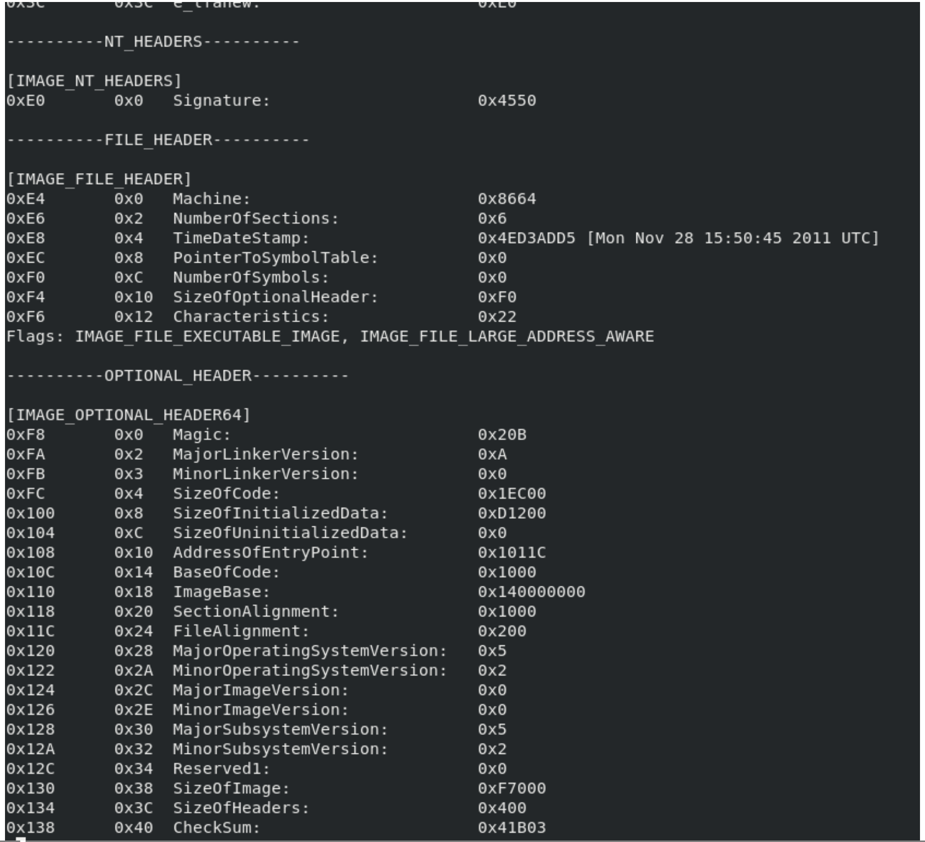
Human Fingerprints on Malware

Human Fingerprints on Malware

Behind every malware is a human developer

- Adversary choices can be found in the malware
  - Leads to the Adversary?
  - Leads to malware?
- Three types of artifacts to look for
  - Header Metadata
  - Code Reuse
  - Configuration Data

Human Fingerprints on Malware – Header Metadata



```
0x3C       0x3C  e_lfanew:                       0xE0

----------NT_HEADERS----------

[IMAGE_NT_HEADERS]
0xE0       0x0   Signature:                      0x4550

----------FILE_HEADER----------

[IMAGE_FILE_HEADER]
0xE4       0x0   Machine:                        0x8664
0xE6       0x2   NumberOfSections:               0x6
0xE8       0x4   TimeDateStamp:                  0x4ED3ADD5 [Mon Nov 28 15:50:45 2011 UTC]
0xEC       0x8   PointerToSymbolTable:           0x0
0xF0       0xC   NumberOfSymbols:                0x0
0xF4       0x10  SizeOfOptionalHeader:           0xF0
0xF6       0x12  Characteristics:                0x22
Flags: IMAGE_FILE_EXECUTABLE_IMAGE, IMAGE_FILE_LARGE_ADDRESS_AWARE

----------OPTIONAL_HEADER----------

[IMAGE_OPTIONAL_HEADER64]
0xF8       0x0   Magic:                          0x20B
0xFA       0x2   MajorLinkerVersion:             0xA
0xFB       0x3   MinorLinkerVersion:             0x0
0xFC       0x4   SizeOfCode:                     0x1EC00
0x100      0x8   SizeOfInitializedData:          0xD1200
0x104      0xC   SizeOfUninitializedData:        0x0
0x108      0x10  AddressOfEntryPoint:            0x1011C
0x10C      0x14  BaseOfCode:                     0x1000
0x110      0x18  ImageBase:                      0x140000000
0x118      0x20  SectionAlignment:               0x1000
0x11C      0x24  FileAlignment:                  0x200
0x120      0x28  MajorOperatingSystemVersion:    0x5
0x122      0x2A  MinorOperatingSystemVersion:    0x2
0x124      0x2C  MajorImageVersion:              0x0
0x126      0x2E  MinorImageVersion:              0x0
0x128      0x30  MajorSubsystemVersion:          0x5
0x12A      0x32  MinorSubsystemVersion:          0x2
0x12C      0x34  Reserved1:                      0x0
0x130      0x38  SizeOfImage:                    0xF7000
0x134      0x3C  SizeOfHeaders:                  0x400
0x138      0x40  CheckSum:                       0x41B03
```

Header Metadata

- Compilation timestamp
- PDB string
- Rich Header
- …

Human Fingerprints on Malware – Header Metadata

GravityRAT samples containing Compilation timestamp and PDB path

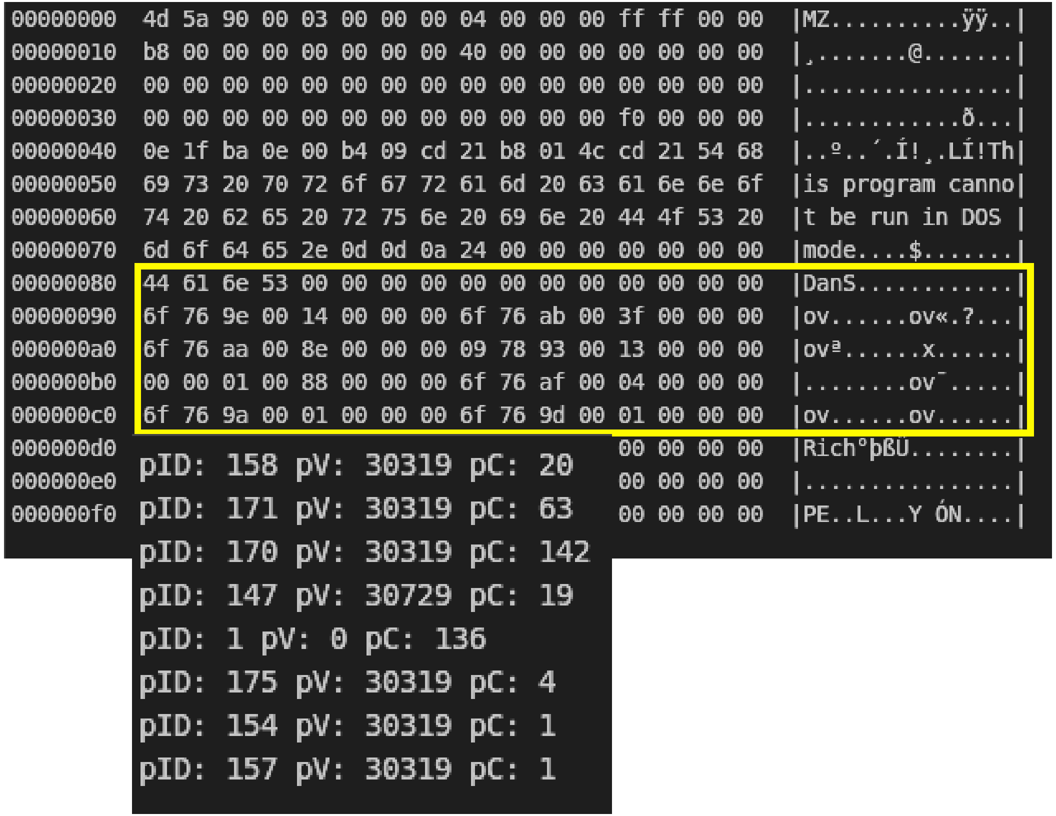
G1:
SHA256: 9f30163c0fe99825022649c5a066a4c972b76210368531d0cfa4c1736c32fb3a
Compiled: 2016-12-22 06:34:24
PDB: f:\F\Windows Work\G1\Adeel's Laptop\G1 Main Virus\systemInterrupts\gravity\obj\x86\Debug\systemInterrupts.pdb

G2:
SHA256: 1993f8d2606c83e22a262ac93cc9f69f972c04460831115b57b3f6244ac128bc
Compiled: 2017-07-31 10:04:20
PDB: e:\Windows Work\G2\G2 Main Virus\Microsoft Virus Solutions (G2 v5) (Current)\Microsoft Virus Solutions\obj\Debug\Windows Wireless 802.11.pdb

G3:
SHA256: 99dd67915566c0951b78d323bb066eb5b130cc7ebd6355ec0338469876503f90
Compiled: 2017-08-21 21:28:31
PDB: F:\Projects\g3\G3 Version 4.0\G3\G3\obj\Release\Intel Core.pdb

GX:
SHA256: 1c0ea462f0bbd7acfdf4c6daf3cb8ce09e1375b766fbd3ff89f40c0aa3f4fc96
Compiled: 2017-12-06 07:52:11
PDB: C:\Users\The Invincible\Desktop\gx\gx-current-program\LSASS\obj\Release\LSASS.pdb

Human Fingerprints on Malware – Header Metadata



```
00000000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00  |MZ..........ÿÿ..|
00000010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  |¸.......@.......|
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00  |............ð...|
00000040  0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68  |..º..´.Í!¸.LÍ!Th|
00000050  69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f  |is program canno|
00000060  74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00  |mode....$.......|
00000080  44 61 6e 53 00 00 00 00 00 00 00 00 00 00 00 00  |DanS............|
00000090  6f 76 9e 00 14 00 00 00 6f 76 ab 00 3f 00 00 00  |ov......ov«.?...|
000000a0  6f 76 aa 00 8e 00 00 00 09 78 93 00 13 00 00 00  |ovª......x......|
000000b0  00 00 01 00 88 00 00 00 6f 76 af 00 04 00 00 00  |........ov¯.....|
000000c0  6f 76 9a 00 01 00 00 00 6f 76 9d 00 01 00 00 00  |ov......ov......|
000000d0                                      00 00 00 00  |Rich°þßÜ........|
000000e0                                      00 00 00 00  |................|
000000f0                                      00 00 00 00  |PE..L...Y ÓN....|
```

```
pID: 158 pV: 30319 pC: 20
pID: 171 pV: 30319 pC: 63
pID: 170 pV: 30319 pC: 142
pID: 147 pV: 30729 pC: 19
pID: 1 pV: 0 pC: 136
pID: 175 pV: 30319 pC: 4
pID: 154 pV: 30319 pC: 1
pID: 157 pV: 30319 pC: 1
```

Rich Header

- Compilers involved in building binary
- Insight into compiling and linking environment
- RichPV fingerprinting build environment

Human Fingerprints on Malware – Code Reuse

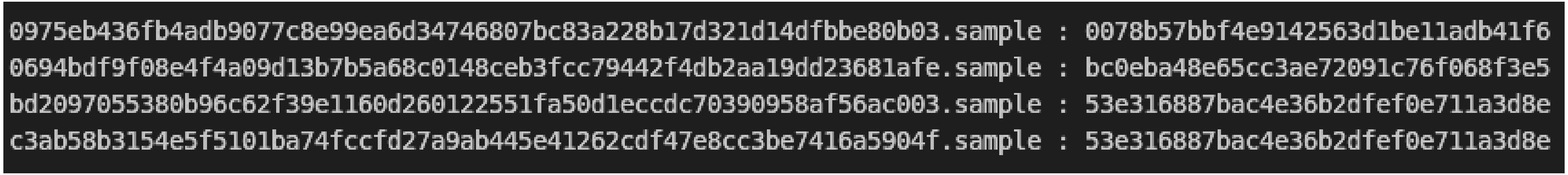
Code Reuse

- Code overlap vs. Code reuse
- Finding new samples
- Finding new malware

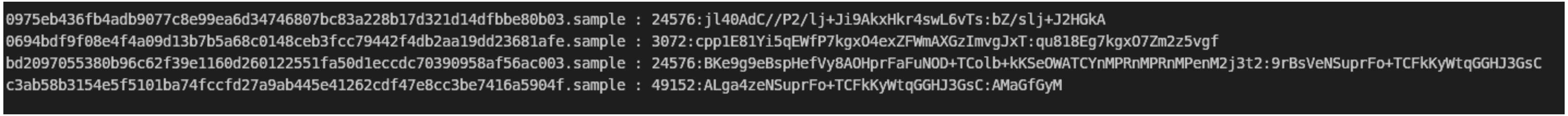
Human Fingerprints on Malware – Code Reuse

Finding new samples

- Imphash
- Fuzzy matching

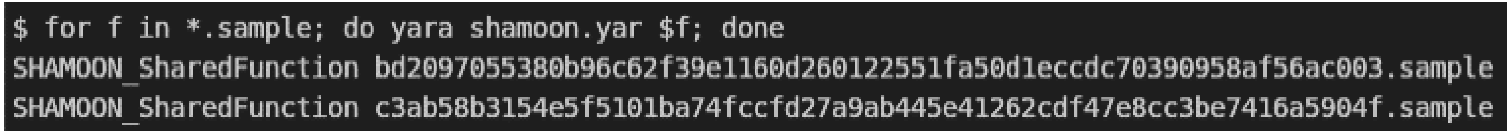


```
0975eb436fb4adb9077c8e99ea6d34746807bc83a228b17d321d14dfbbe80b03.sample : 0078b57bbf4e9142563d1be11adb41f6
0694bdf9f08e4f4a09d13b7b5a68c0148ceb3fcc79442f4db2aa19dd23681afe.sample : bc0eba48e65cc3ae72091c76f068f3e5
bd2097055380b96c62f39e1160d260122551fa50d1eccdc70390958af56ac003.sample : 53e316887bac4e36b2dfef0e711a3d8e
c3ab58b3154e5f5101ba74fccfd27a9ab445e41262cdf47e8cc3be7416a5904f.sample : 53e316887bac4e36b2dfef0e711a3d8e
```

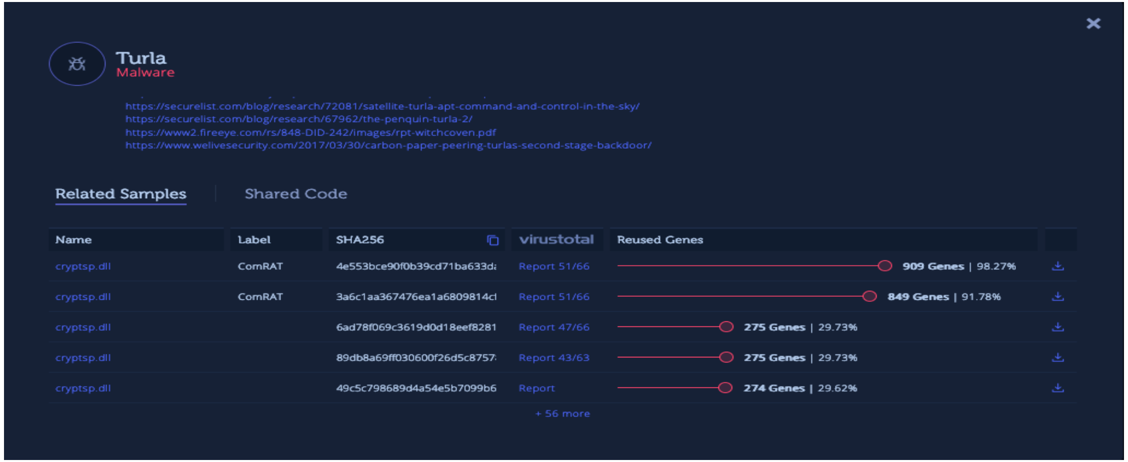


```
0975eb436fb4adb9077c8e99ea6d34746807bc83a228b17d321d14dfbbe80b03.sample : 24576:jl40AdC//P2/lj+Ji9AkxHkr4swL6vTs:bZ/slj+J2HGkA
0694bdf9f08e4f4a09d13b7b5a68c0148ceb3fcc79442f4db2aa19dd23681afe.sample : 3072:cpp1E81Yi5qEWfP7kgxO4exZFWmAXGzImvgJxT:qu818Eg7kgxO7Zm2z5vgf
bd2097055380b96c62f39e1160d260122551fa50d1eccdc70390958af56ac003.sample : 24576:BKe9g9eBspHefVy8AOHprFaFuNOD+TColb+kKSeOWATCYnMPRnMPRnMPenM2j3t2:9rBsVeNSuprFo+TCFkKyWtqGGHJ3GsC
c3ab58b3154e5f5101ba74fccfd27a9ab445e41262cdf47e8cc3be7416a5904f.sample : 49152:ALga4zeNSuprFo+TCFkKyWtqGGHJ3GsC:AMaGfGyM
```

Human Fingerprints on Malware – Code Reuse



```
$ for f in *.sample; do yara shamoon.yar $f; done
SHAMOON_SharedFunction bd2097055380b96c62f39e1160d260122551fa50d1eccdc70390958af56ac003.sample
SHAMOON_SharedFunction c3ab58b3154e5f5101ba74fccfd27a9ab445e41262cdf47e8cc3be7416a5904f.sample
```



Turla
Malware

https://securelist.com/blog/research/72081/satellite-turla-apt-command-and-control-in-the-sky/
https://securelist.com/blog/research/67962/the-penquin-turla-2/
https://www2.fireeye.com/rs/848-DID-242/images/rpt-witchcoven.pdf
https://www.welivesecurity.com/2017/03/30/carbon-paper-peering-turlas-second-stage-backdoor/

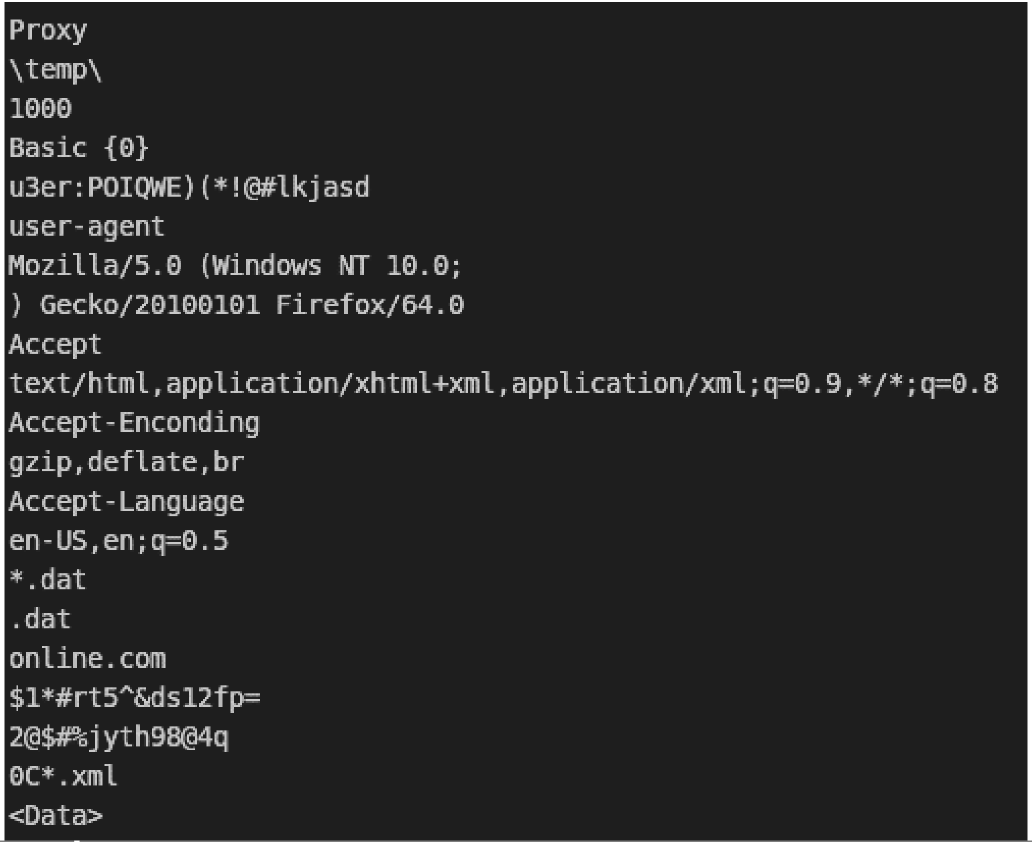
Name	Label	SHA256	virustotal	Reused Genes
cryptsp.dll	ComRAT	4e553bce90f0b39cd71ba633d	Report 51/66	909 Genes | 98.27%
cryptsp.dll	ComRAT	3a6c1aa367476ea1a6809814c	Report 51/66	849 Genes | 91.78%
cryptsp.dll		6ad78f069c3619d0d18eef8281	Report 47/66	275 Genes | 29.73%
cryptsp.dll		89db8a69ff030600f26d5c8757	Report 43/63	275 Genes | 29.73%
cryptsp.dll		49c5c798689d4a54e5b7099b6	Report	274 Genes | 29.62%

+ 56 more

Finding new malware

- YARA
- Community tools
- Other databases

Human Fingerprints on Malware – Configuration Data

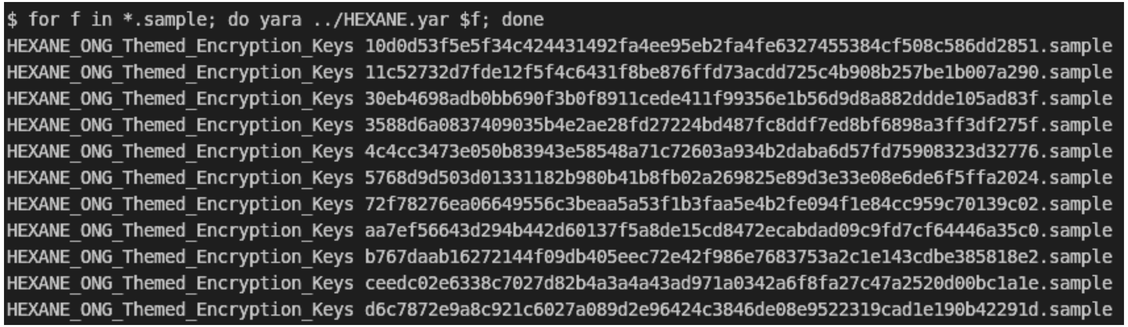


```
Proxy
\temp\
1000
Basic {0}
u3er:POIQWE)(*!@#lkjasd
user-agent
Mozilla/5.0 (Windows NT 10.0;
) Gecko/20100101 Firefox/64.0
Accept
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Enconding
gzip,deflate,br
Accept-Language
en-US,en;q=0.5
*.dat
.dat
online.com
$1*#rt5^&ds12fp=
2@$#%jyth98@4q
0C*.xml
<Data>
```

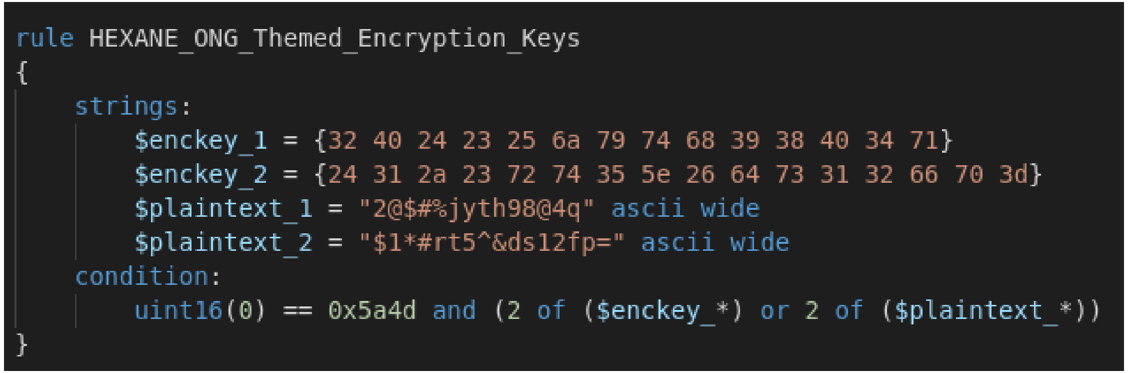
Configuration Data

- Developer or Operator
- Configure malware for current campaign
- Adversary preferences

Human Fingerprints on Malware – Configuration Data



```
$ for f in *.sample; do yara ../HEXANE.yar $f; done
HEXANE_ONG_Themed_Encryption_Keys 10d0d53f5e5f34c424431492fa4ee95eb2fa4fe6327455384cf508c586dd2851.sample
HEXANE_ONG_Themed_Encryption_Keys 11c52732d7fde12f5f4c6431f8be876ffd73acdd725c4b908b257be1b007a290.sample
HEXANE_ONG_Themed_Encryption_Keys 30eb4698adb0bb690f3b0f8911cede411f99356e1b56d9d8a882ddde105ad83f.sample
HEXANE_ONG_Themed_Encryption_Keys 3588d6a0837409035b4e2ae28fd27224bd487fc8ddf7ed8bf6898a3ff3df275f.sample
HEXANE_ONG_Themed_Encryption_Keys 4c4cc3473e050b83943e58548a71c72603a934b2daba6d57fd75908323d32776.sample
HEXANE_ONG_Themed_Encryption_Keys 5768d9d503d01331182b980b41b8fb02a269825e89d3e33e08e6de6f5ffa2024.sample
HEXANE_ONG_Themed_Encryption_Keys 72f78276ea06649556c3beaa5a53f1b3faa5e4b2fe094f1e84cc959c70139c02.sample
HEXANE_ONG_Themed_Encryption_Keys aa7ef56643d294b442d60137f5a8de15cd8472ecabdad09c9fd7cf64446a35c0.sample
HEXANE_ONG_Themed_Encryption_Keys b767daab16272144f09db405eec72e42f986e7683753a2c1e143cdbe385818e2.sample
HEXANE_ONG_Themed_Encryption_Keys ceedc02e6338c7027d82b4a3a4a43ad971a0342a6f8fa27c47a2520d00bc1a1e.sample
HEXANE_ONG_Themed_Encryption_Keys d6c7872e9a8c921c6027a089d2e96424c3846de08e9522319cad1e190b42291d.sample
```

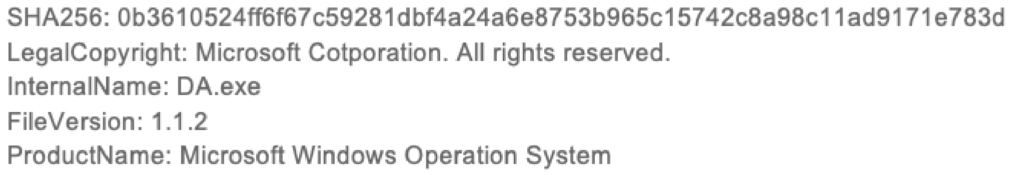


```
rule HEXANE_ONG_Themed_Encryption_Keys
{
    strings:
        $enckey_1 = {32 40 24 23 25 6a 79 74 68 39 38 40 34 71}
        $enckey_2 = {24 31 2a 23 72 74 35 5e 26 64 73 31 32 66 70 3d}
        $plaintext_1 = "2@$#%jyth98@4q" ascii wide
        $plaintext_2 = "$1*#rt5^&ds12fp=" ascii wide
    condition:
        uint16(0) == 0x5a4d and (2 of ($enckey_*) or 2 of ($plaintext_*))
}
```

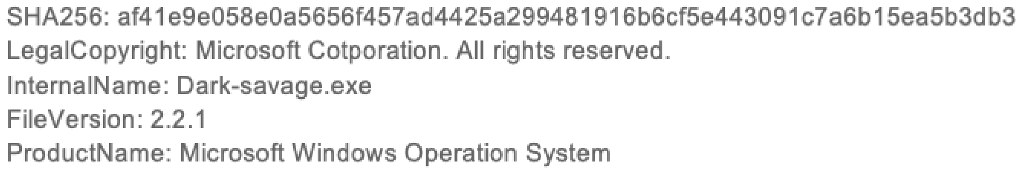
Finding new samples:

- Unique strings -> new samples

Human Fingerprints on Malware – Configuration Data



```
SHA256: 0b3610524ff6f67c59281dbf4a24a6e8753b965c15742c8a98c11ad9171e783d
LegalCopyright: Microsoft Cotporation. All rights reserved.
InternalName: DA.exe
FileVersion: 1.1.2
ProductName: Microsoft Windows Operation System
```

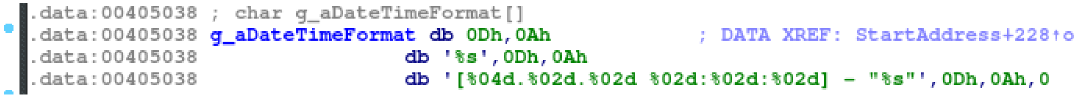


```
SHA256: af41e9e058e0a5656f457ad4425a299481916b6cf5e443091c7a6b15ea5b3db3
LegalCopyright: Microsoft Cotporation. All rights reserved.
InternalName: Dark-savage.exe
FileVersion: 2.2.1
ProductName: Microsoft Windows Operation System
```

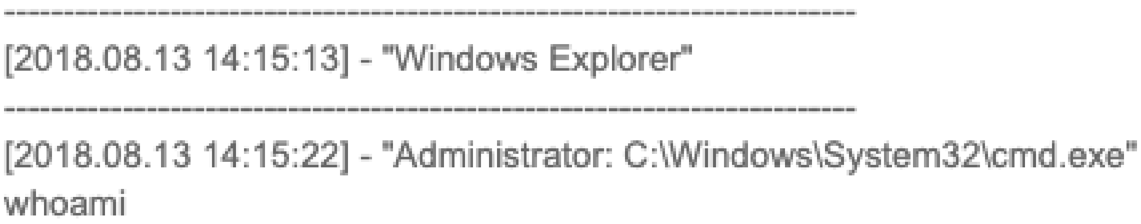
Finding new samples:

- Typos -> new samples

Human Fingerprints on Malware – Configuration Data



```
.data:00405038 ; char g_aDateTimeFormat[]
.data:00405038 g_aDateTimeFormat db 0Dh,0Ah            ; DATA XREF: StartAddress+228↑o
.data:00405038                  db '%s',0Dh,0Ah
.data:00405038                  db '[%04d.%02d.%02d %02d:%02d:%02d] - "%s"',0Dh,0Ah,0
```

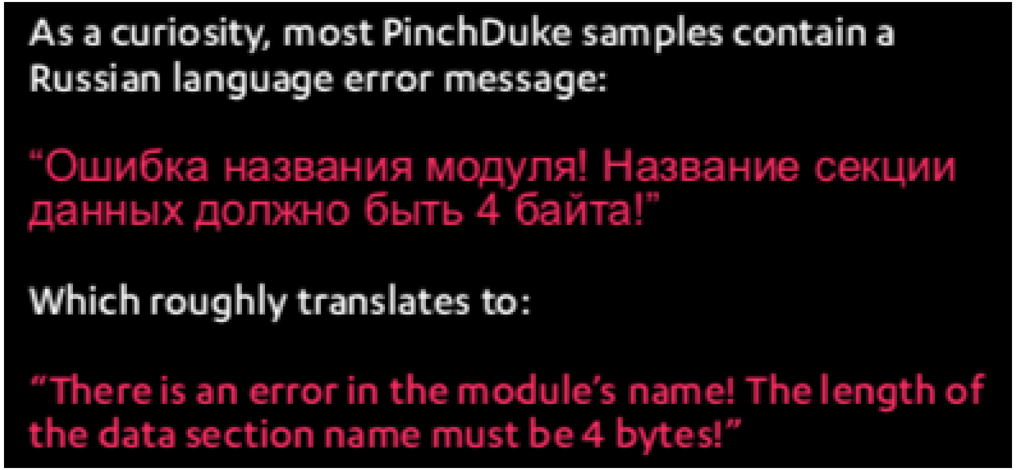


```
-----------------------------------------------------------------------
[2018.08.13 14:15:13] - "Windows Explorer"
-----------------------------------------------------------------------
[2018.08.13 14:15:22] - "Administrator: C:\Windows\System32\cmd.exe"
whoami
```

Adversary knowledge:

- Unique formats -> preference

Human Fingerprints on Malware – Configuration Data



As a curiosity, most PinchDuke samples contain a Russian language error message:

"Ошибка названия модуля! Название секции данных должно быть 4 байта!"

Which roughly translates to:

"There is an error in the module's name! The length of the data section name must be 4 bytes!"

Adversary knowledge:

- Local language -> preference

https://www.f-secure.com/documents/996508/1030745/dukes_whitepaper.pdf

Human Fingerprints on Malware – Caveats

- Assess credibility of fingerprints
  - Timestamps, PDB path, Rich Header, Language
- Common strings
  - Associated with public code or common OS interaction
- Common code
  - “Public” code from public library, forum snippets open to anyone
  - “Private” code from alliances, partnerships, supply chains not visible to defender

Human Fingerprints on Malware – Getting started

Header Metadata

- Pefile by Ero Carrera on Github
- FireEye blog: Definitive Dossier of Devilish Debug Details by Steve Miller
- SANS reading room: Leveraging the PE Rich Header for Static Malware Detection and Linking by Maksim Dubyk

Code Reuse

- yara_fn.py on GitHub by Willi Ballenthin
- YarGen on GitHub by Florian Roth (Neo23x0)

Configuration Data

- Analyze all the malware
- SANS FOR610 Reverse-Engineering Malware

Human Fingerprints on Malware – References

- The concept of import hashing was defined by FireEye in Tracking Malware with Import Hashing https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html
- The concept of ssdeep is provided by the ssdeep Project https://ssdeep-project.github.io/ssdeep/index.html
- Smart Whitelisting Using Locality Sensitive Hashing https://blog.trendmicro.com/trendlabs-security-intelligence/smart-whitelisting-using-locality-sensitive-hashing/
- YARA was originally developed by Victor Alvarez of VirusTotal https://virustotal.github.io/yara/

Conclusion