Fraunhofer FKIE

Fraunhofer-Institut für
Kommunikation, Informations-
verarbeitung und Ergonomie FKIE

Daniel Plohmann
Virus Bulletin Conference | London | 2023-10-04

Applied one-to-many code similarity analysis using MCRIT

Introduction
$whoami

- Security Researcher @ Fraunhofer FKIE & University of Bonn

Open Source Analysis Tooling:
IDAscope, ApiScout, SMDA, **MCRIT**, …

DGA ARCHIVE
malpedia

Data Sets

Botnet Takedowns

[1] https://github.com/danielplohmann
[2] https://dgarchive.caad.fkie.fraunhofer.de
[3] https://malpedia.io

Outline

- Motivation
- MCRIT: System Overview
  - Methodology
  - Framework
  - Use Cases
- Case Studies
- Summary and Outlook

Motivation

Motivation

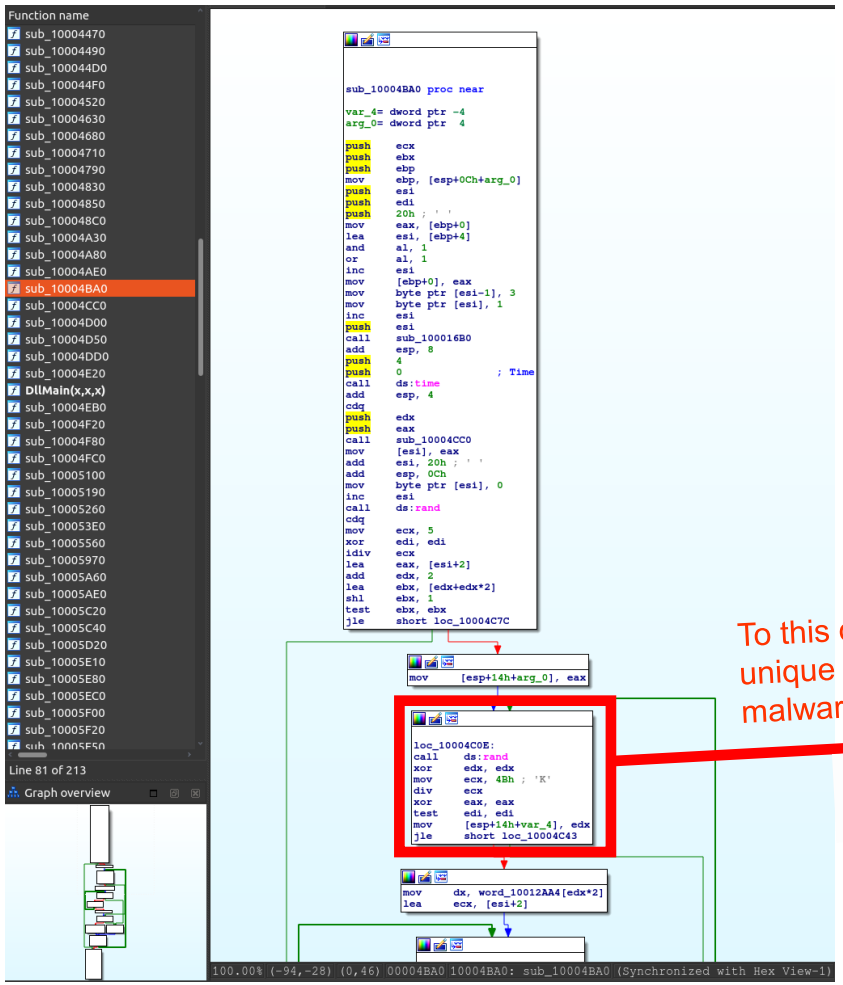
Anzeige im Hauptbahnhof in Chemnitz am Freitag: Forderung von Lösegeld Foto: P. Götzelt/ dpa

- Infamous WannaCry
- Ransomware attack using wormable exploit (EternalBlue)
  - Attack started on May 12th 2017
  - 230k affected systems in ~8 hours
  - Quickly disrupted due to a lucky registration of killswitch domain
- Impact
  - UK NHS disrupted (£100m damage)
  - Nissan, Renault, Telefonica, FedEx, DB, …
- Attack Attribution?

[1] https://en.wikipedia.org/wiki/WannaCry_ransomware_attack
[2] https://twitter.com/neelmehta/status/864164081116225536

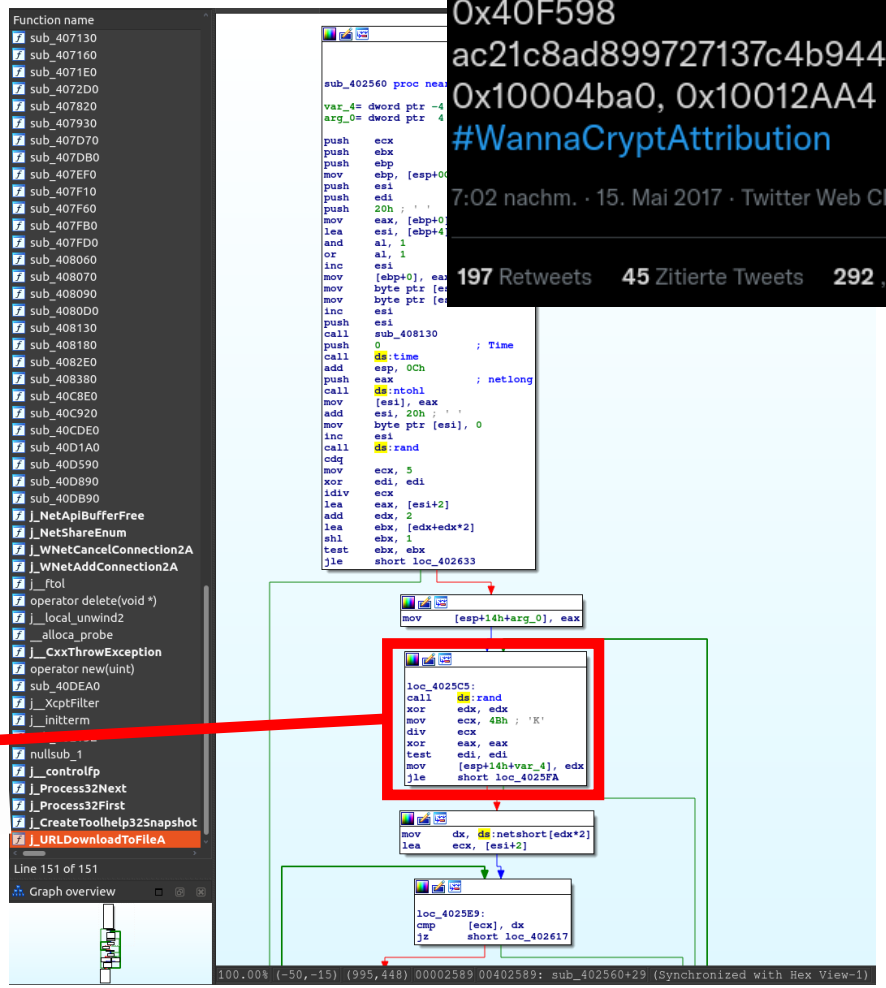
Fraunhofer FKIE

Motivation



To this day…
unique across 1700+
malware families

malpedia



[1] https://twitter.com/neelmehta/status/864164081116225536

Fraunhofer FKIE

Motivation

- Code Similarity Analysis
  - High potential to help analysts and accelerate analysis
    - Source code reuse identification, library filtering, hunting, label transfer, …
  - Existing solutions mostly
    - limited to 1:1 comparison - e.g. BinDiff, Diaphora, …
    - abandoned or proprietary

- My goal with this presentation
  1) Showcase current status and capabilities of MCRIT
  2) Get feedback and further ideas from you!

MCRIT

System Overview

MCRIT
Design

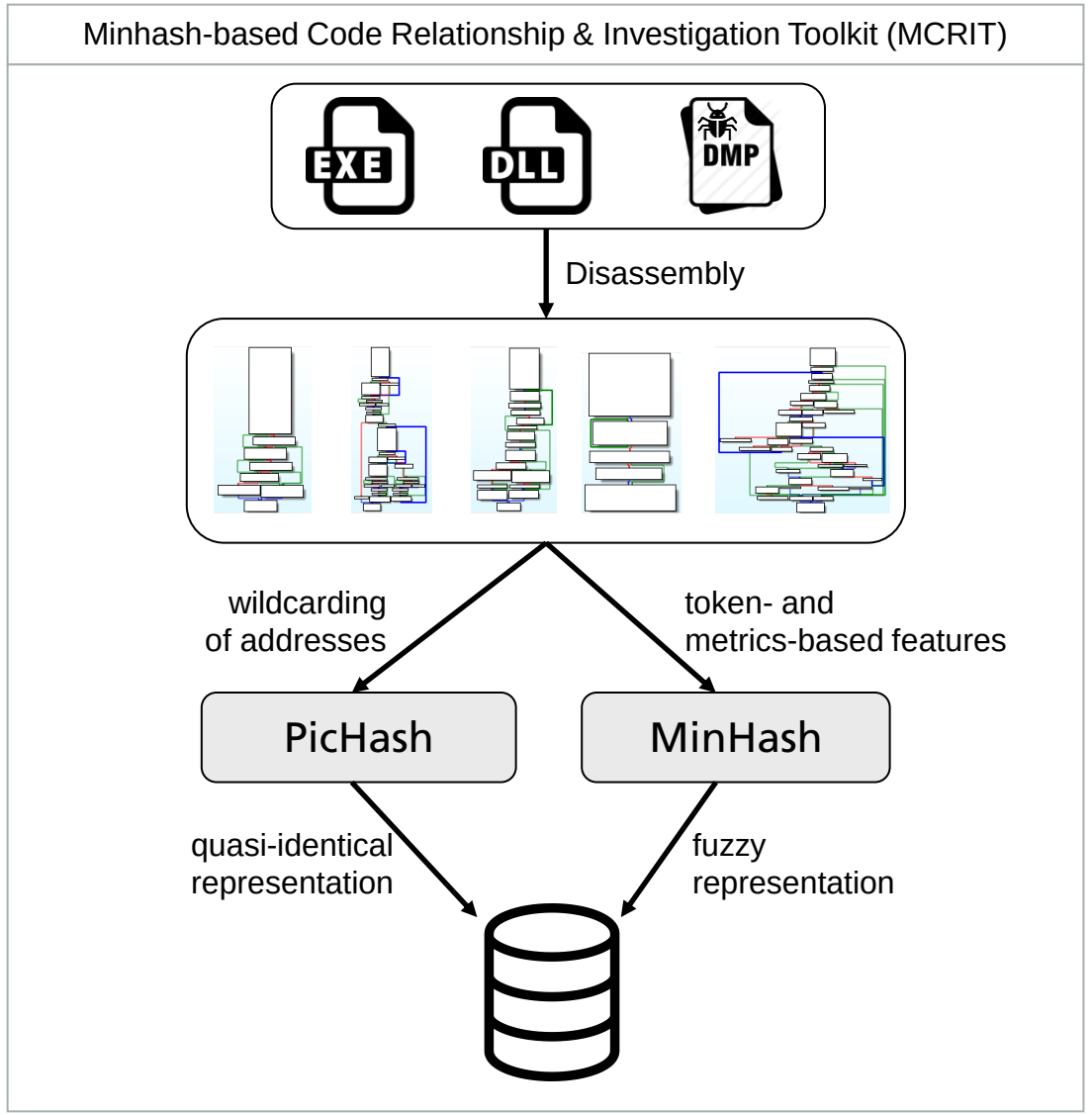
➔ Minhash-based Code Relationship & Investigation Toolkit (MCRIT)

- Goal: Analyze code sharing and third-party library usage in malware
  - Create open source tools to leverage Malpedia binary corpus
  - Don‘t reinvent the wheel: reuse of proven techniques as described in literature
- Requirements:
  - Similarity: reliable, interpretable estimate
  - Scalability: (tens of) millions of functions
  - Efficient representation: (significantly) smaller than indexed code
- Limitation:
  - Limited cross-bitness and no cross-architecture for now

Code Recovery and Similarity Analysis

MCRIT: Overview

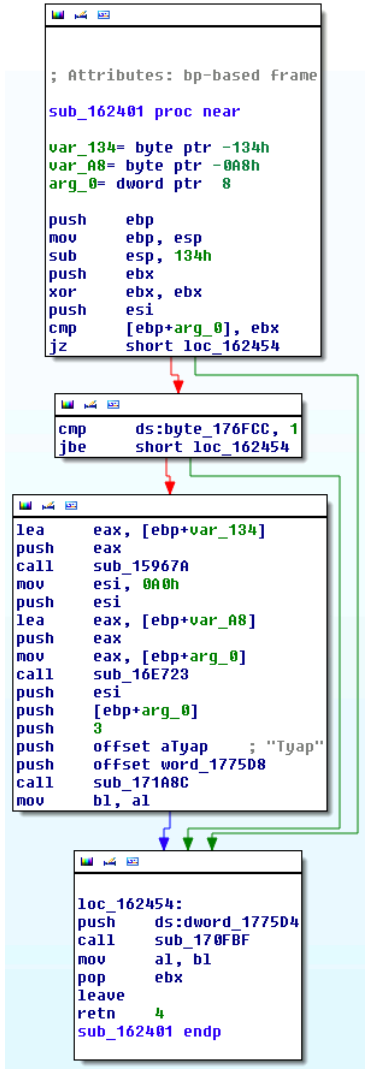
- MCRIT
  - Combines quasi-identical and fuzzy code representation
  - Basic Block & Function-level similarity
  - Efficient 1:n matching via
    - Hashmaps
    - Locality-Sensitive Hashing (LSH)



Project Co-Authors: Paul Hordiienko, Steffen Enders, Manuel Blatt, Daniel Enders

MCRIT

PicHash: Functions



- Quasi-Identical: Position Independent Code (PIC) Hashing
  - On function level (original method as introduced by Cohen and Havrilla [1])
  - On basic block level (more granularity, almost the same speed)

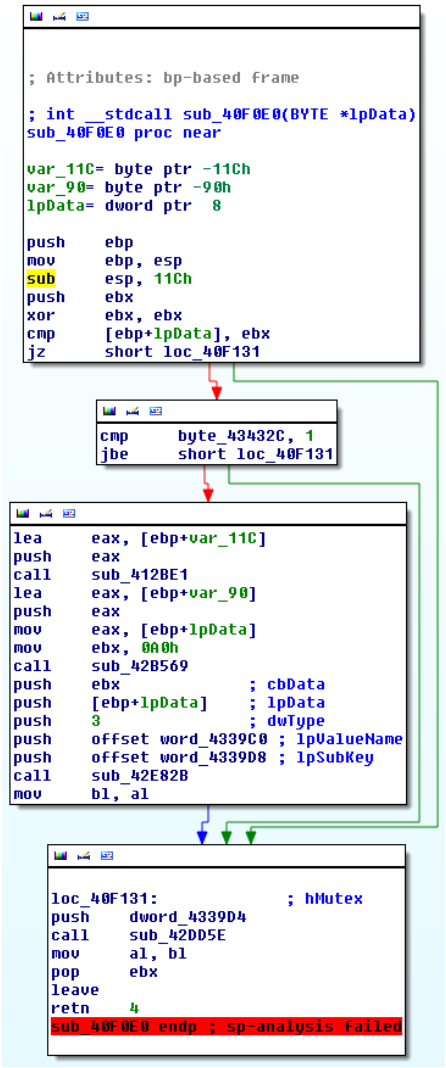
```
pichash(„55 8BEC 81EC34010000 …“)
```

```
-> 80066413841218754O5
```

```
pichash(„55 8BEC 81EC1C010000 …“)
```

```
-> 10270976525648996728
```

- Good for recognizing statically linked code (often binary identical)

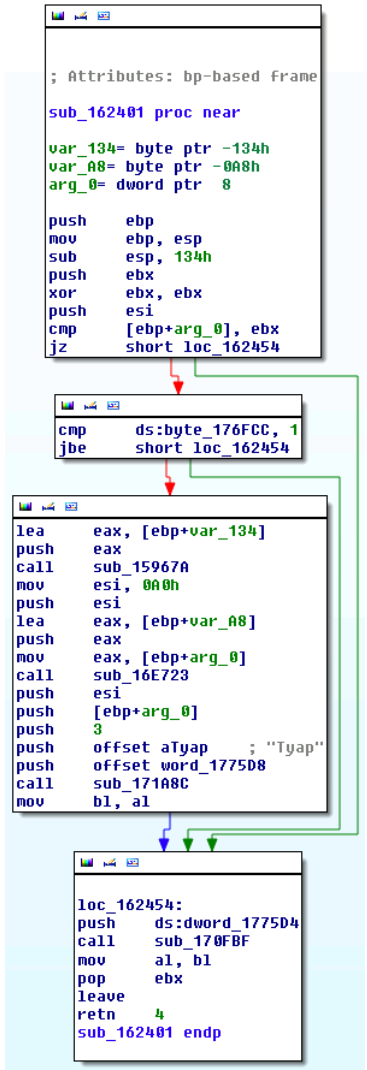


[1] C. Cohen and J. Havrilla, “Function Hashing for Malicious Code Analysis“, tech. rep., SEI, CMU, 2009.

Fraunhofer FKIE

MCRIT

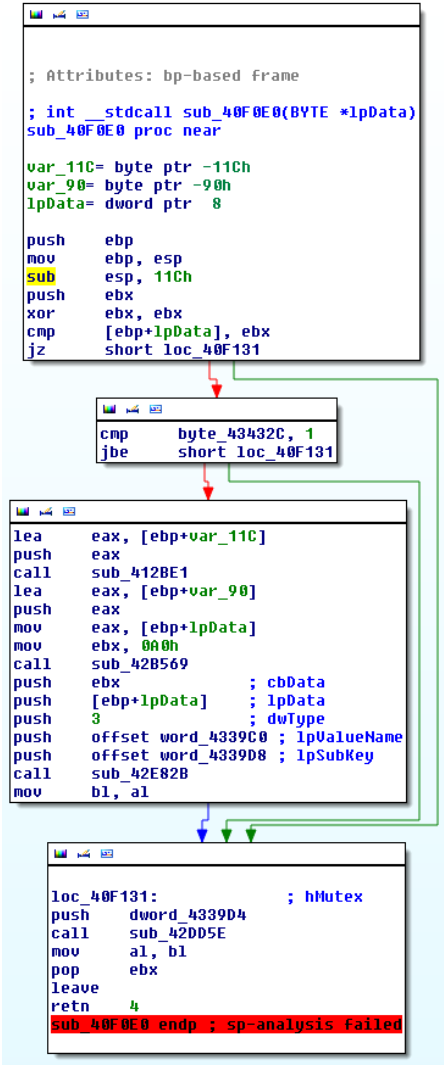
PicHash: Basic Blocks



- Quasi-Identical: Position Independent Code (PIC) Hashing
  - On function level (original method as introduced by Cohen and Havrilla [1])
  - On basic block level, size 4+ (more granularity, almost the same speed)

pichash(„55 8BEC 81EC34010000 …“)	pichash(„55 8BEC 81EC1C010000 …“)
-> 620962970	-> 2827249019
pichash(„8D85CCFEFFFF 50 E8?? …“)	pichash(„8D85E4FEFFFF 50 E8?? …“)
-> 847901973	-> 640583737
pichash(„FF35???????? E8???? …“)	pichash(„FF35???????? E8???? …“)
-> 463987246	-> 463987246

- Good for recognizing partial code reuse (fractions and candidates)



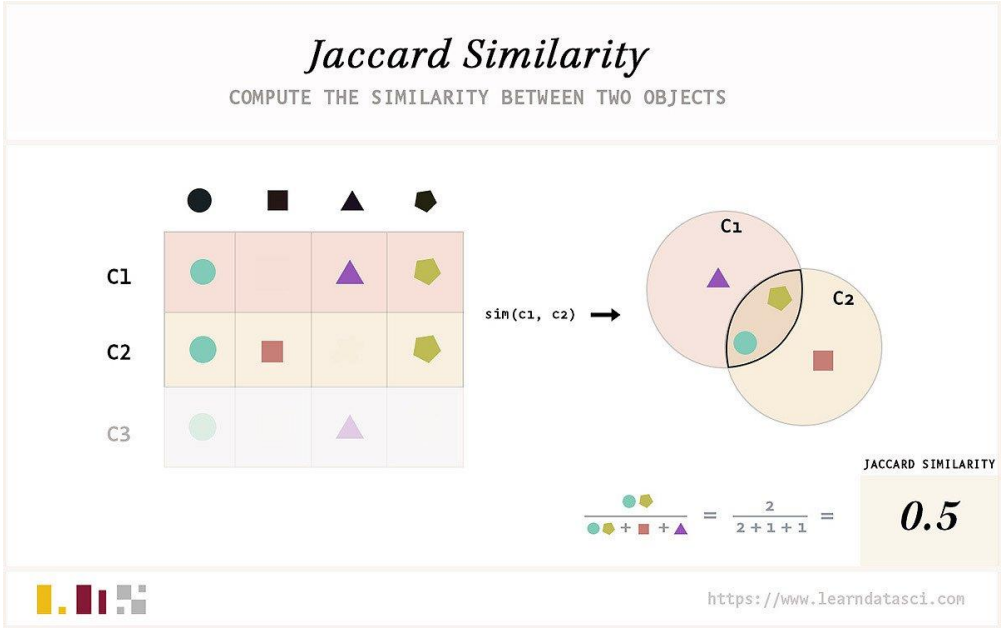
[1] C. Cohen and J. Havrilla, “Function Hashing for Malicious Code Analysis“, tech. rep., SEI, CMU, 2009.

Fraunhofer FKIE

MCRIT
MinHashing

- MinHashing
  - „Min-wise independent permutations“ - Locality Sensitive Hashing (LSH) scheme [1]
  - Fast estimation of set similarity -> approximation of Jaccard similarity coefficient
  - Scalability: O(log *n*) for single lookups

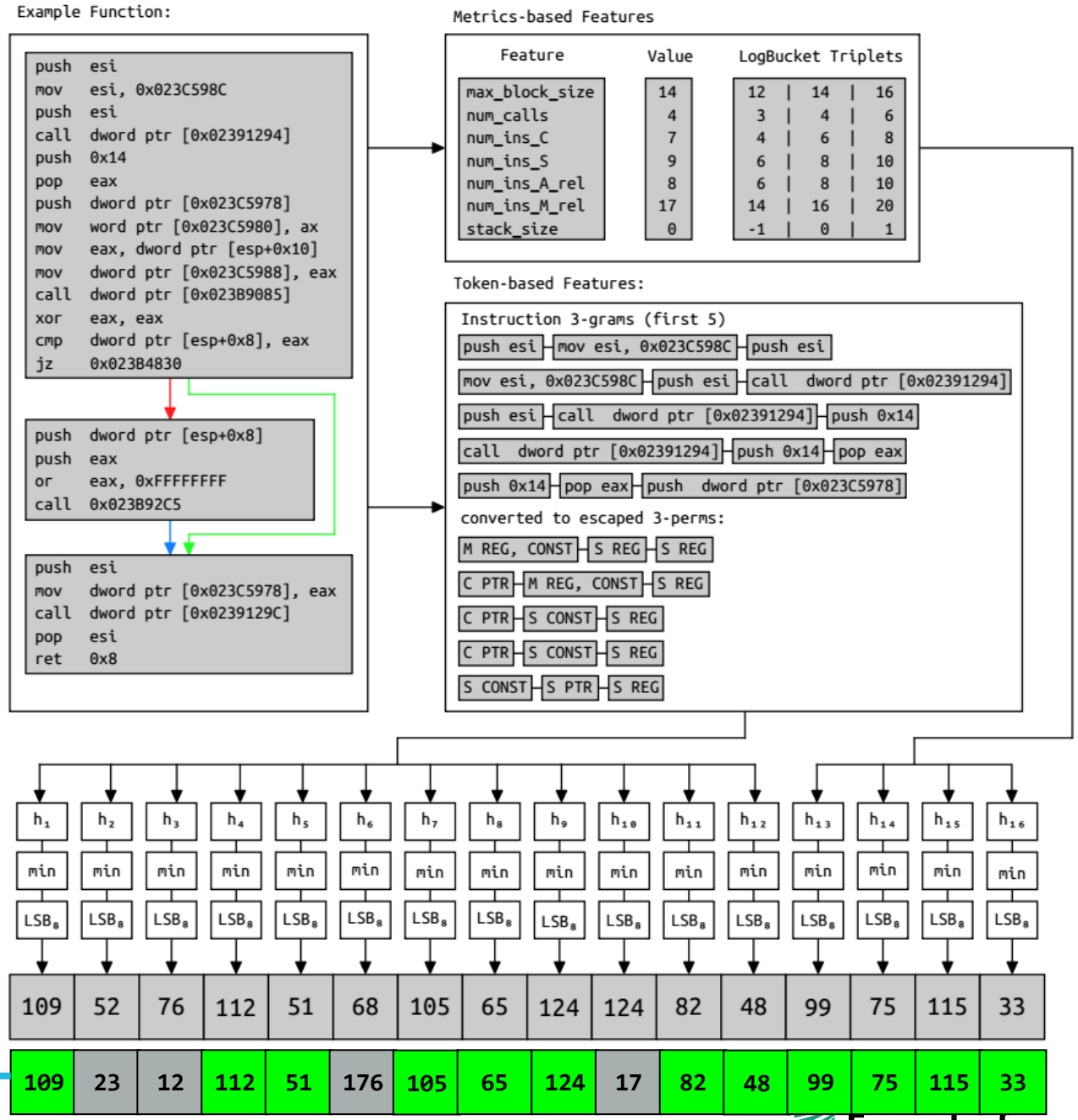
- Use cases:
  - text documents / websites (duplicates, plagiarism)
  - genome sequencing
  - code similarity! [2]



[1] “Min-wise independent permutations”. Broder et al., In: Proceedings of the 30th ACM Symposium on Theory of Computing (STOC '98), New York, NY, USA.
[2] “Binary Function Clustering using Semantic Hashes”. Jin et al., Carnegie Mellon University, 2012.
[3] https://www.learndatasci.com/glossary/jaccard-similarity/

MCRIT
MinHash Composition

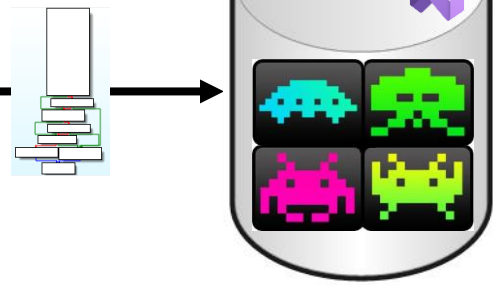
- Token-based features:
  - Instruction 3-grams
  - Abstract semantically
  - Convert to 3-perms (sorted)
- Metrics-based features:
  - Numerically describe structure of a function
  - Normalize & Quantize for fuzzy matching
- MinHash matching:
  - Count same values in same position



MCRIT

Querying the System

- Query with
  - Basic Block
  - Function
  - Sample



Function A	Offset A	Offset B	Function B	Family B	Sample B	Score
4459063	0x406770	0x406770	4990660	win.romeos	4498	100
4459063	0x406770	0x10006b50	6362927	win.romeos	5710	92
4459063	0x406770	0x10006b50	5238246	win.romeos	4730	92
4459063	0x406770	0x405f00	6681224	win.op_blockbuster	5954	89
4459063	0x406770	0x405ff0	1816610	win.op_blockbuster	1859	89
4459063	0x406770	0x10003490	1257856	win.badcall	1288	65
4459063	0x406770	0x10004600	1135454	win.hardrain	1159	62
4459063	0x406770	0x344600	889162	win.hardrain	909	62

For every function, we know how many families we match.

1. We can use this to weigh by occurrence frequency when aggregating to sample matches.
2. We can identify unique matches into just one family and use this as further indication for identity.

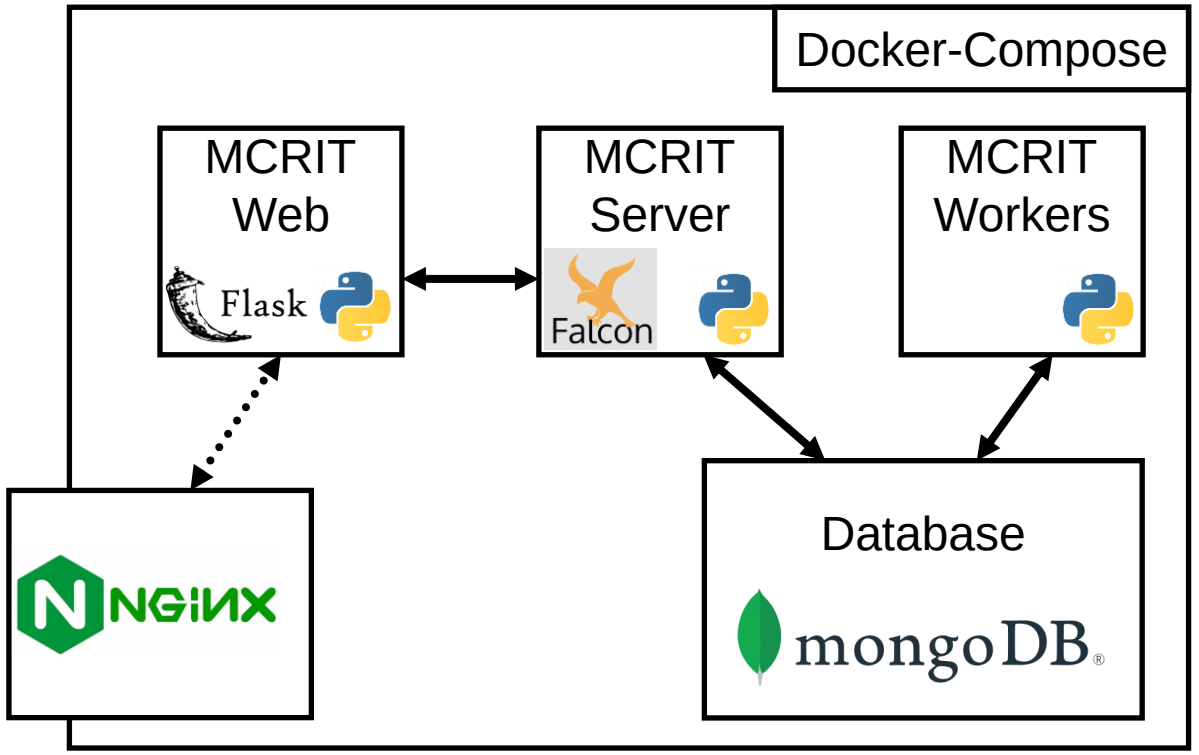
	Version		SHA256	Filename	Bitness	FNs	Min#	Pic#	Lib	Direct		Frequency		Uniq
win.romeos	2014-07-07-alfa	4498	4ec0214b	eff542ac8e...0x00400000	32	311	247	247	71	98	97	47	59	21.49%
win.hardrain		909	c3b1af35	2cc3b5f2df...0x00340000	32	289	138	103	66	50	37	13	13	0.00%
win.op_blockbuster	2015-04-08	1859	2f629c3c	2f629c3c65...3_unpacked	32	341	159	122	70	56	43	12	12	0.00%
win.keymarble	2017-04-12	4543	f19cd9ef	e23900b00f...0x00400000	32	448	168	137	70	53	38	9	8	0.00%

[1] eff542ac8e37db48821cb4e5a7d95c044fff27557763de3a891b40ebeb52cc55 @ 0x406770 | win.romeos

Fraunhofer FKIE

MCRIT
Dockerized Setup

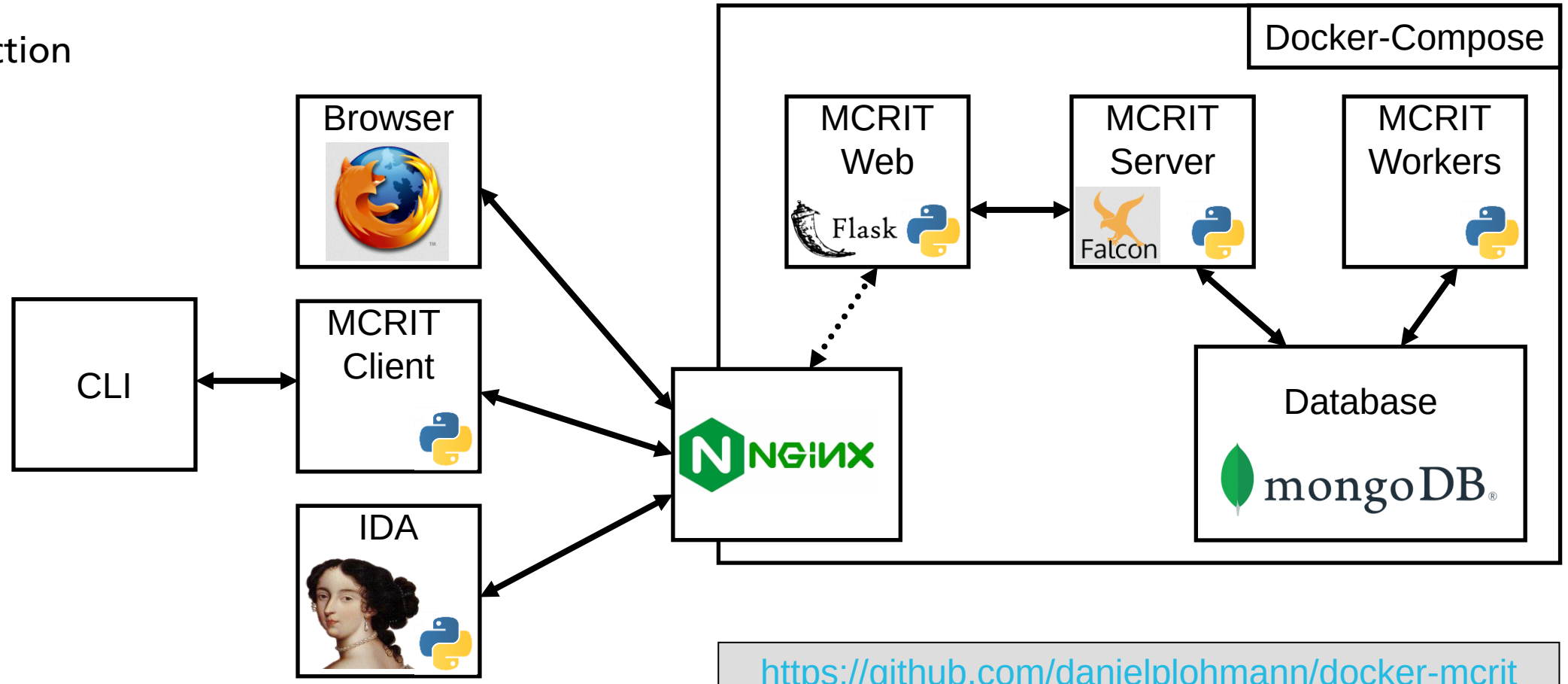
- Database
- MCRIT Server
  - Core of the system
  - Enable access to stored content (API)
  - Create matching jobs
- MCRIT Workers
  - Process jobs from the queue
- MCRIT Web
  - Expose service functionality in a user interface
  - User management
  - API forwarding to MCRIT server



https://github.com/danielplohmann/docker-mcrit

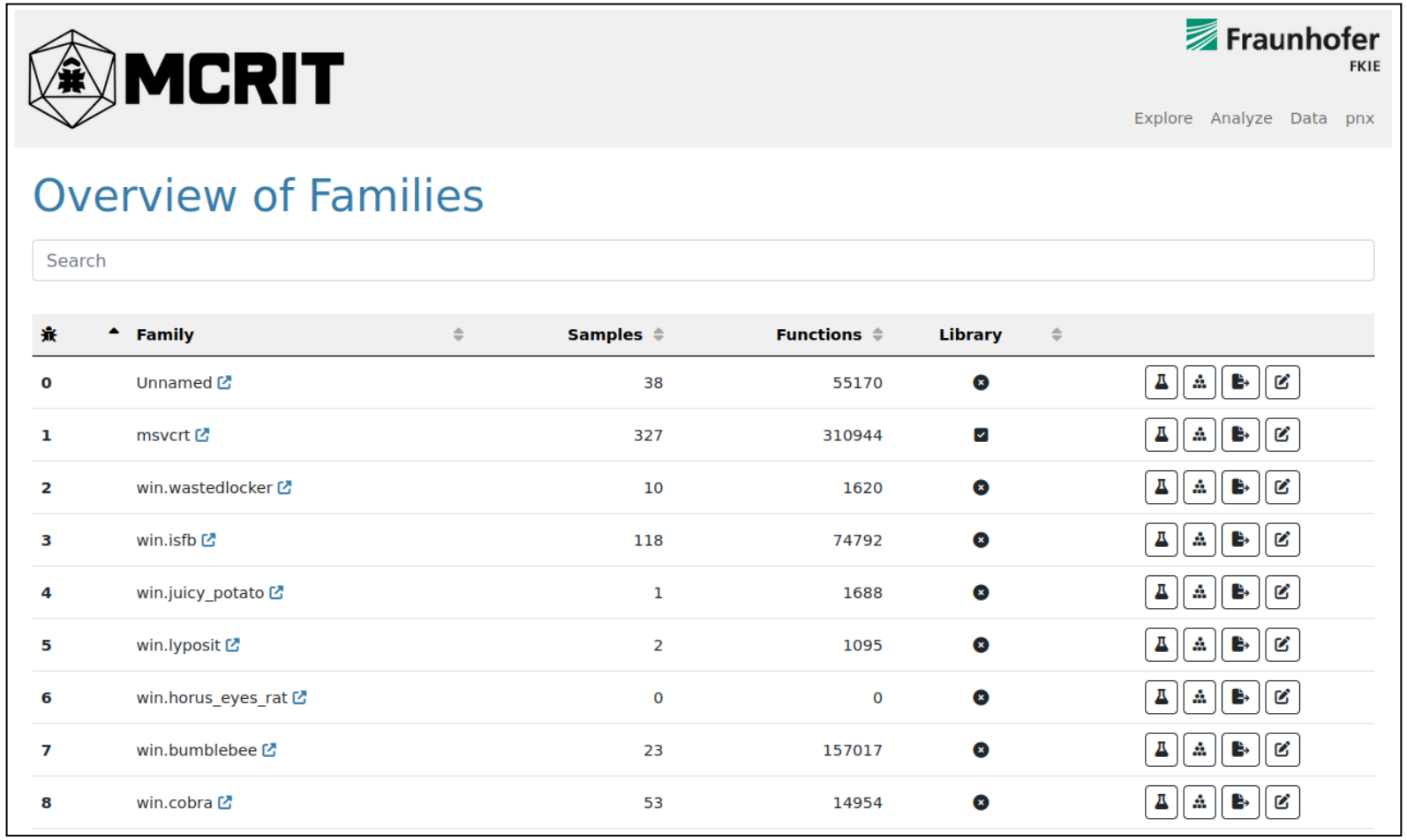
MCRIT
Interaction with MCRIT

- Interaction

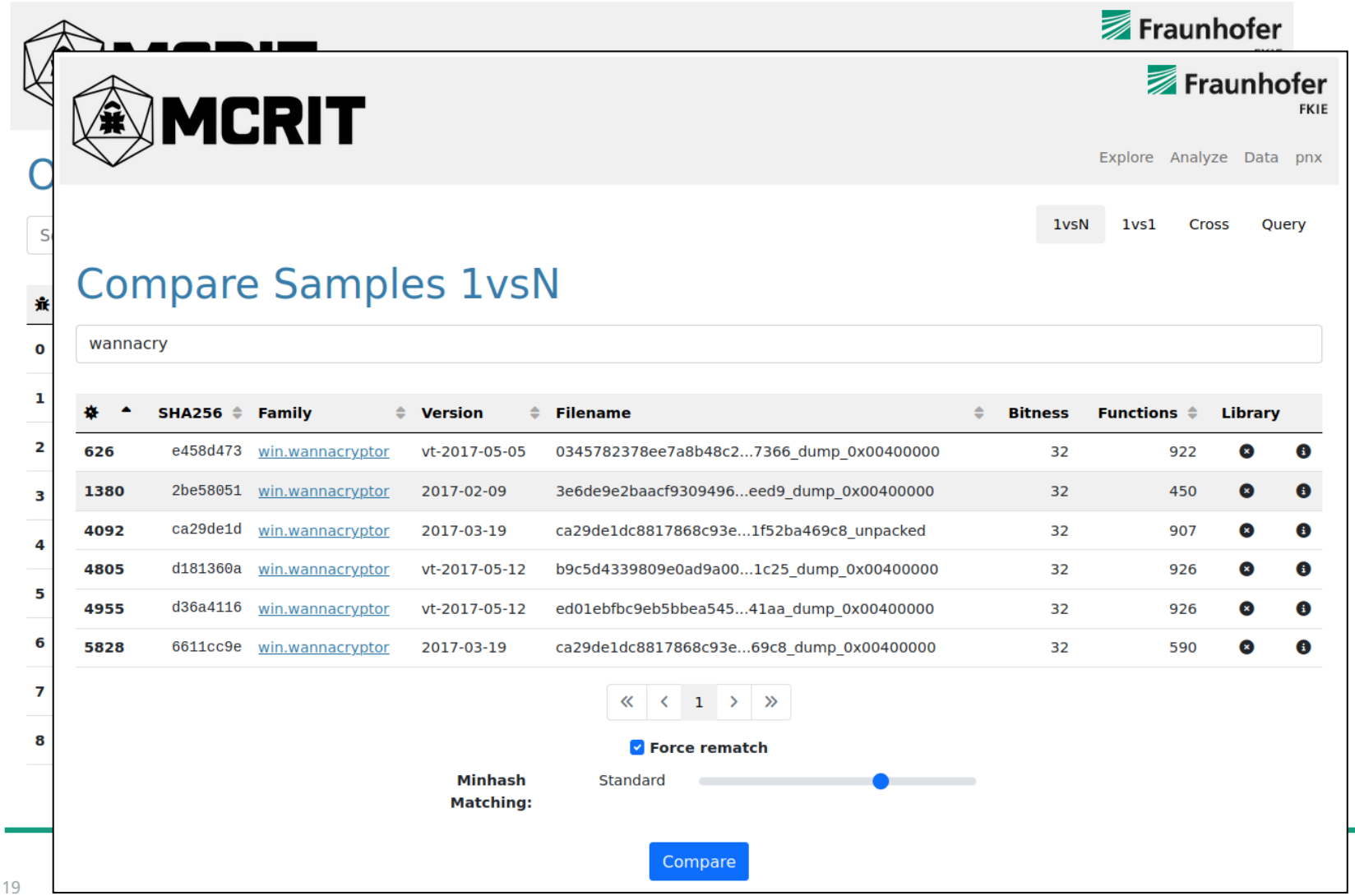


https://github.com/danielplohmann/docker-mcrit

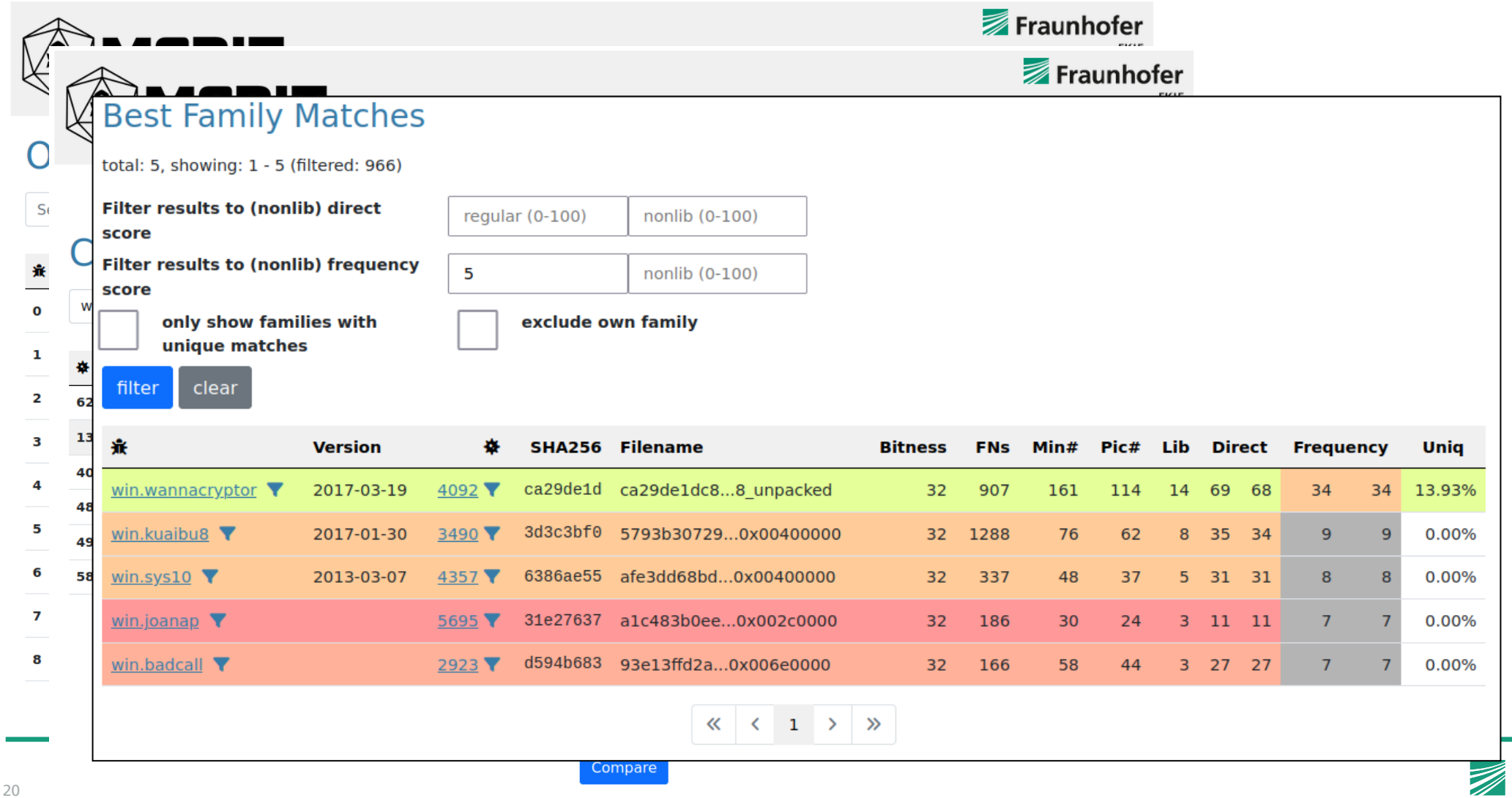
MCRIT
WebUI



MCRIT
WebUI



MCRIT
WebUI



MCRIT

New Features in Release 1.2

- Version 1.0 published in April 2023 at Botconf
  - Matching fully implemented, basic WebUI
- Version 1.2 – Virus Bulletin Release
  - Focus on usability improvements
    - Polishing of the WebUI – filters, …
    - Significant extension of the IDA plugin
      - Just-in-time matching information for currently shown function
      - Import of matching and label information
    - Prototype LinkHunt feature that makes use of ICFG relationship information

Use Cases and Case Studies

Example Use Cases

- Malware family identification and library code differentiation
  - Clustering and comparison
  - Isolation of unique family code
- Lead generation for discovering potentially unknown links
- Label Transfer

malpedia +  - - >

Current stats of my demo instance:

num_samples	7372
num_families	1767
num_functions	9637676
num_pichashes	2083507

8 Cores, 32GB RAM

Use Cases and Case Studies

Malware Family Identification

Example Use Cases

Malware Family Identification and Library Code Differentiation

LockBit ransomware goes 'Green,' uses new Conti-based encryptor

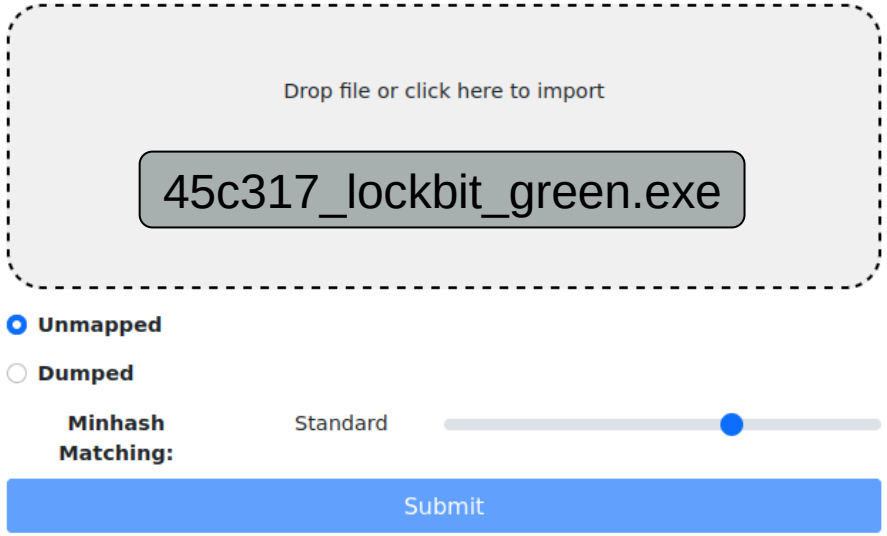
By Lawrence Abrams — February 1, 2023 — 05:48 PM — 0

The LockBit ransomware gang has again started using encryptors based on other operations, this time switching to one based on the leaked source code for the Conti ransomware.

Since its launch, the LockBit operation has gone through numerous iterations of its encryptor, starting with a custom one and moving to LockBit 3.0 (aka LockBit Black), which is derived from the BlackMatter gang's source code.

This week, cybersecurity collective VX-Underground first reported that the ransomware gang is now using a new encryptor named 'LockBit Green,' based on the leaked source code of the now-disbanded Conti gang.

Query Sample



Disassembly + Matching: 35sec

[1] https://www.bleepingcomputer.com/news/security/lockbit-ransomware-goes-green-uses-new-conti-based-encryptor/
[2] lockbit green: 45c317200e27e5c5692c59d06768ca2e7eeb446d6d495084f414d0f261f75315

Fraunhofer FKIE

Best Family Matches

total: 1070, showing: 1 - 10

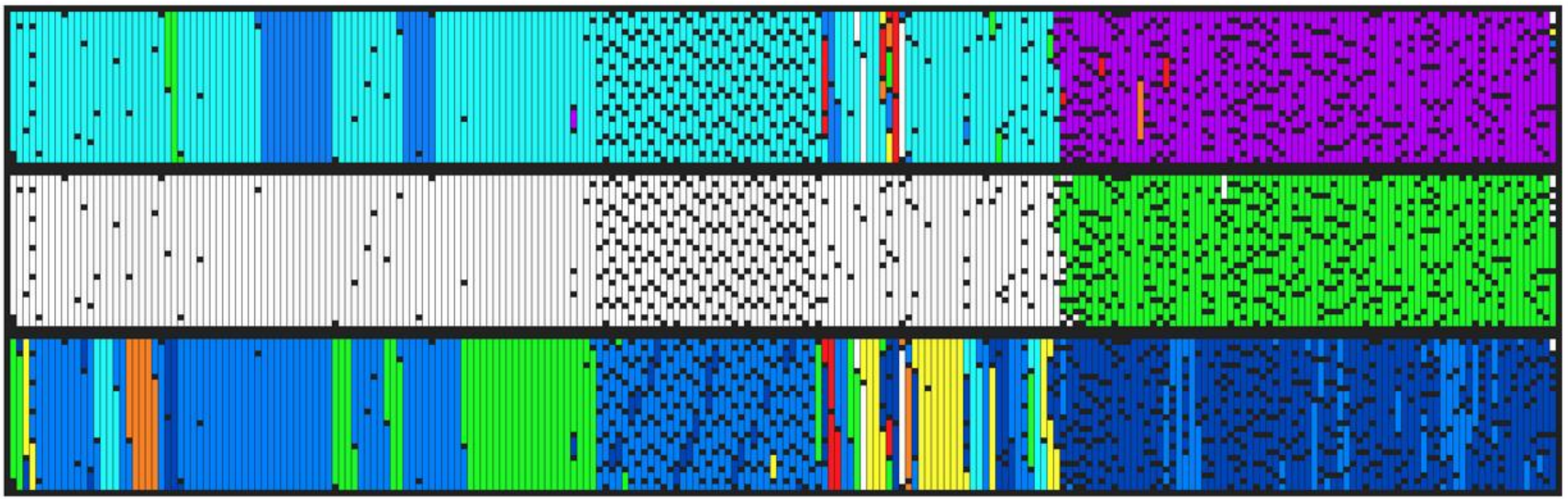
	Version		SHA256	Filename	Bitness	FNs	Min#	Pic#	Lib	Direct		Frequency	
win.meow		2172	222e2b91	222e2b91f5...3_unpacked	32	702	461	126	220	66	71	41	53
win.conti	2021-02-04	5041	a5751a46	a5751a4676...6_unpacked	32	736	516	183	256	59	58	28	36
win.scarecrow		6199	bcf49782	bcf49782d7...a_unpacked	32	653	582	271	334	61	51	27	32
win.lockergoga	2019-03-18	4517	edae201c	c97d9bbc80...0x00400000	32	7847	369	311	352	26	1	3	0
win.void		1460	2fd1863e	2fd1863eb3...c_unpacked	32	7123	366	312	351	26	1	3	0
win.bandook		4792	fabce973	fabce973a9...7_unpacked	32	3229	363	306	349	25	1	3	0

Best Family Matches

total: 1070, showing: 1 - 10

	Version		SHA256	Filename	Bitness	FNs	Min#	Pic#	Lib	Direct		Frequency	
win.meow		2172	222e2b91	222e2b91f5...3_unpacked	32	702	461	126	220	66	71	41	53
win.conti	2021-02-04	5041	a5751a46	a5751a4676...6_unpacked	32	736	516	183	256	59	58	28	36
win.scarecrow		6199	bcf49782	bcf49782d7...a_unpacked	32	653	582	271	334	61	51	27	32

Showing: foreign family match frequency, library matches, best foreign family match scores.

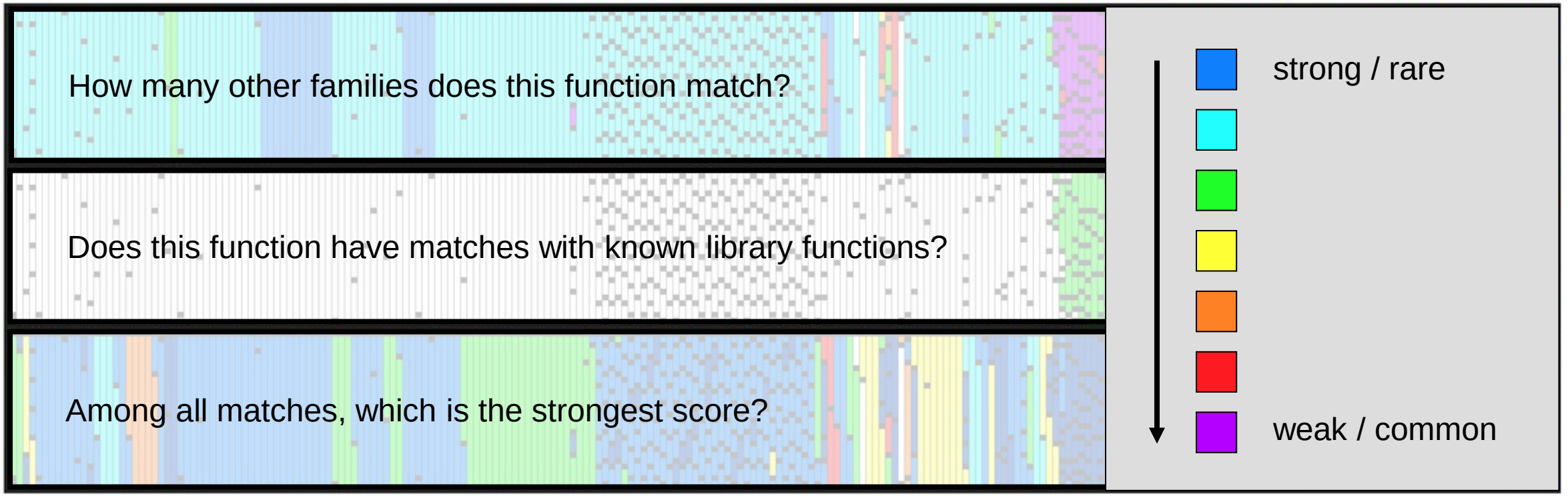


Best Family Matches

total: 1070, showing: 1 - 10

	Version		SHA256	Filename	Bitness	FNs	Min#	Pic#	Lib	Direct		Frequency	
win.meow		2172	222e2b91	222e2b91f5...3_unpacked	32	702	461	126	220	66	71	41	53
win.conti	2021-02-04	5041	a5751a46	a5751a4676...6_unpacked	32	736	516	183	256	59	58	28	36
win.scarecrow		6199	bcf49782	bcf49782d7...a_unpacked	32	653	582	271	334	61	51	27	32

Showing: foreign family match frequency, library matches, best foreign family match scores.

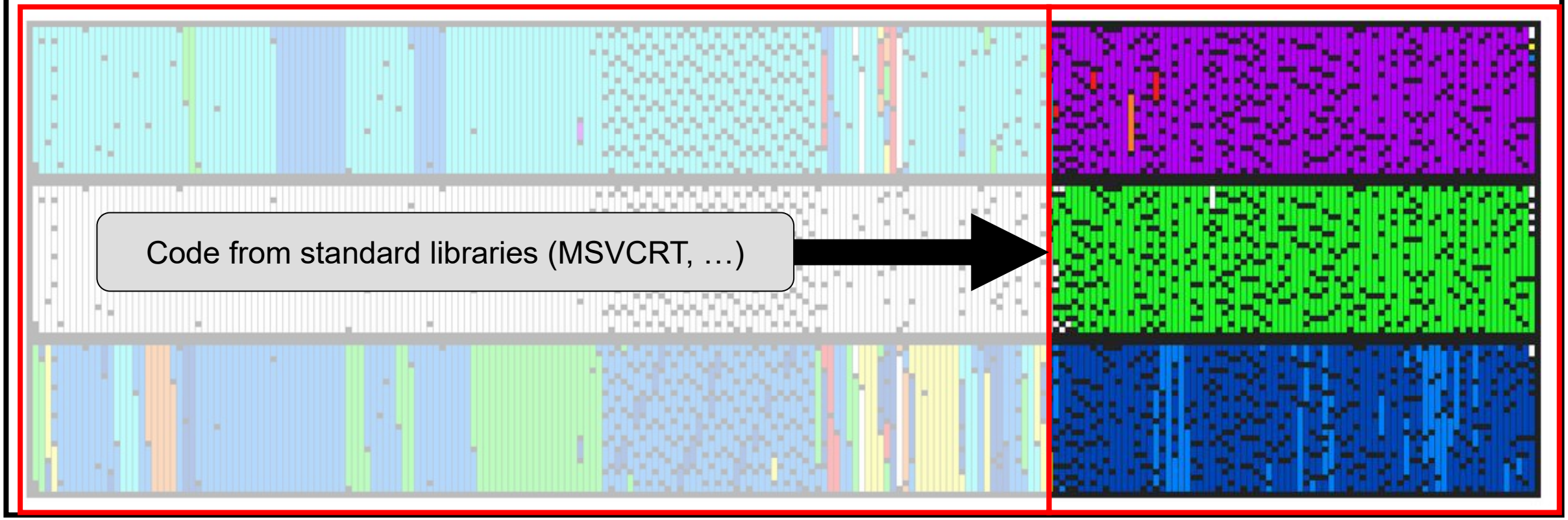


Best Family Matches

total: 1070, showing: 1 - 10

	Version		SHA256	Filename	Bitness	FNs	Min#	Pic#	Lib	Direct		Frequency	
win.meow		2172	222e2b91	222e2b91f5...3_unpacked	32	702	461	126	220	66	71	41	53
win.conti	2021-02-04	5041	a5751a46	a5751a4676...6_unpacked	32	736	516	183	256	59	58	28	36
win.scarecrow		6199	bcf49782	bcf49782d7...a_unpacked	32	653	582	271	334	61	51	27	32

Showing: foreign family match frequency, library matches, best foreign family match scores.

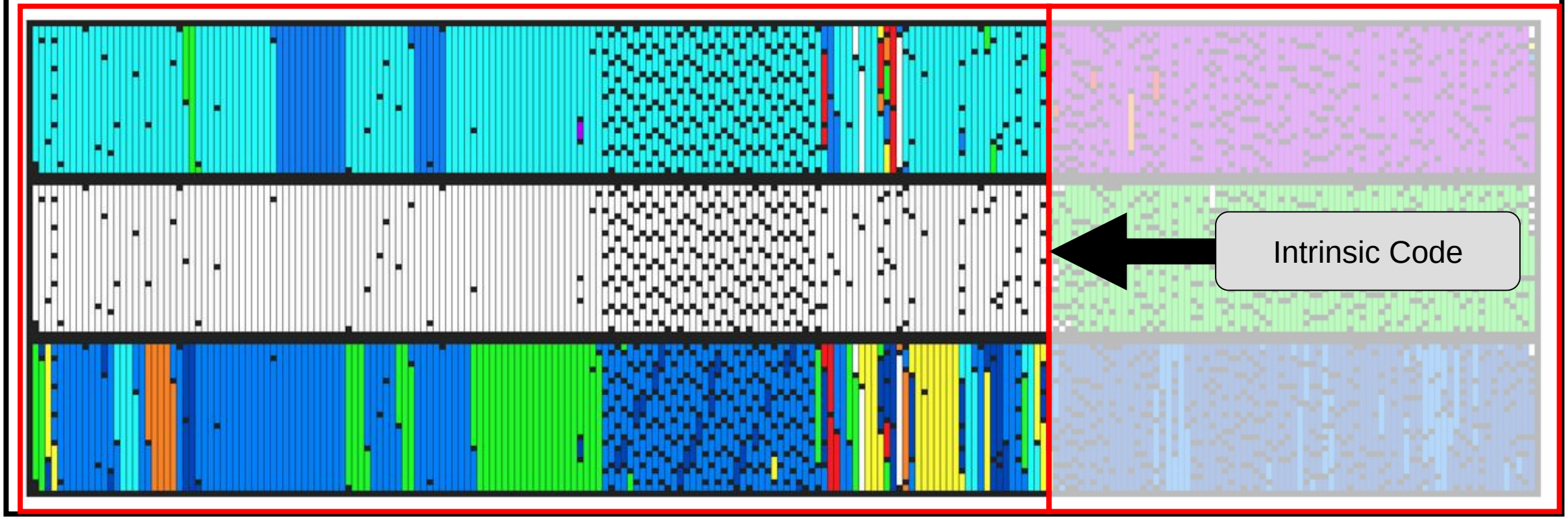


Best Family Matches

total: 1070, showing: 1 - 10

	Version		SHA256	Filename	Bitness	FNs	Min#	Pic#	Lib	Direct		Frequency	
win.meow		2172	222e2b91	222e2b91f5...3_unpacked	32	702	461	126	220	66	71	41	53
win.conti	2021-02-04	5041	a5751a46	a5751a4676...6_unpacked	32	736	516	183	256	59	58	28	36
win.scarecrow		6199	bcf49782	bcf49782d7...a_unpacked	32	653	582	271	334	61	51	27	32

Showing: foreign family match frequency, library matches, best foreign family match scores.



[1] conti: a5751a46768149c5ddf318fd75afc66b3db28a5b76254ee0d6ae27b21712e266

Function CFGs

Show Cycles | Show Loops | Show Loop Boundaries | Enable Tooltip

go to top

lockbit green
@0x401730

conti
@0x10017a0

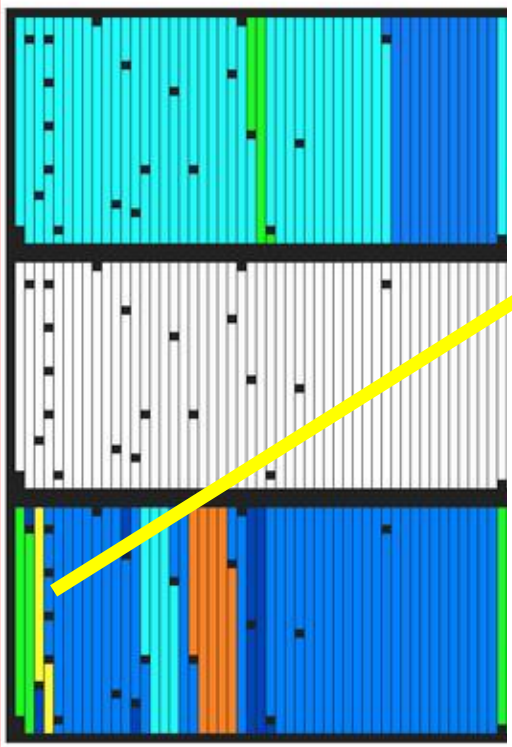
Best Family Match

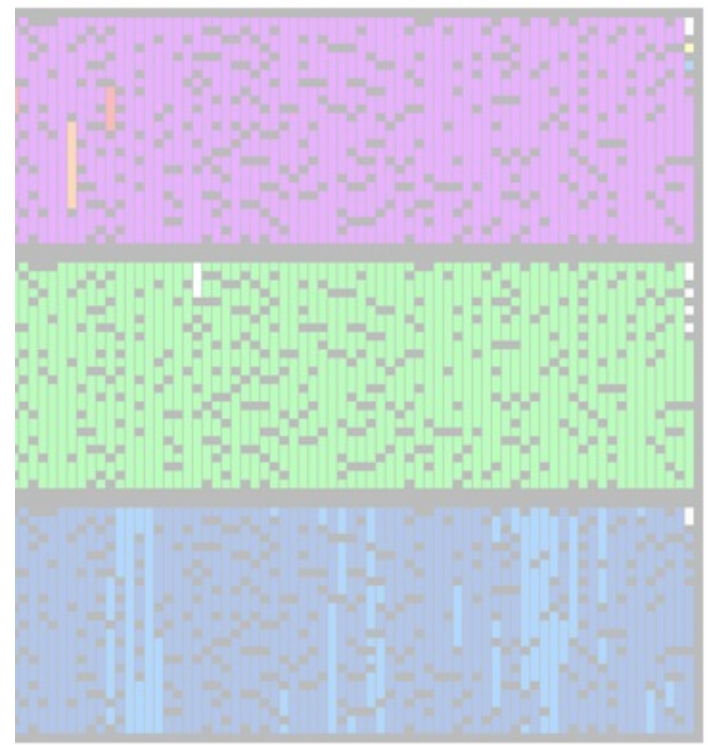
total: 1070, showing: 1 - 10

	Version
win.meow	
win.conti	2021-02-
win.scarecrow	

Min#	Pic#	Lib	Direct		Frequency	
461	126	220	66	71	41	53
516	183	256	59	58	28	36
582	271	334	61	51	27	32

Showing: foreign family match freque





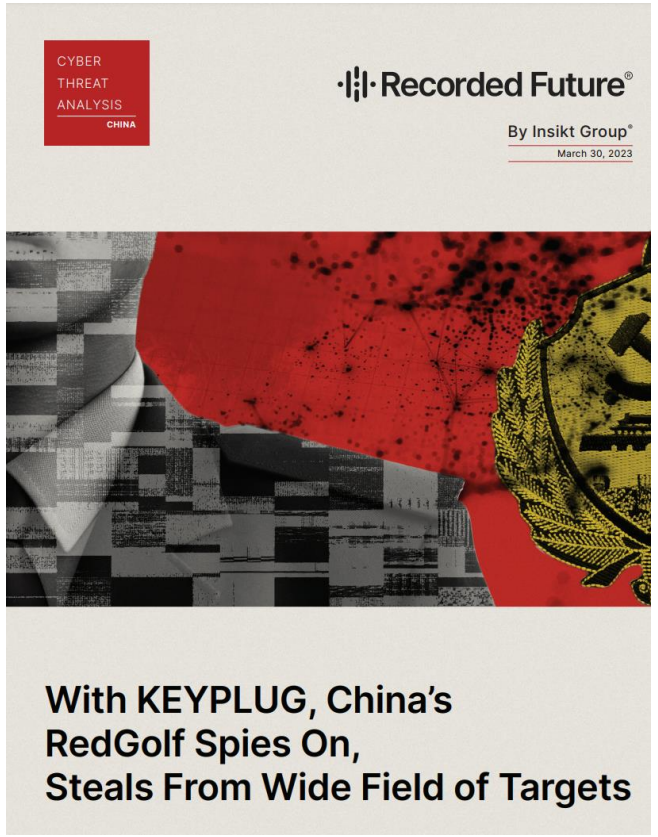
Use Cases and Case Studies

Clustering

Example Use Cases

Clustering: KEYPLUG and Friends

- APThursday @ FKIE
  - Volunteer / Enthusiast Group looking at APT activity
- VT Retrohunted shellcode loader used in samples publicly reported
  - Found / unpacked 13 hits
- Question:
  - What families are these samples?



[1] https://go.recordedfuture.com/hubfs/reports/cta-2023-0330.pdf
[2] https://www.mandiant.com/resources/blog/apt41-us-state-governments

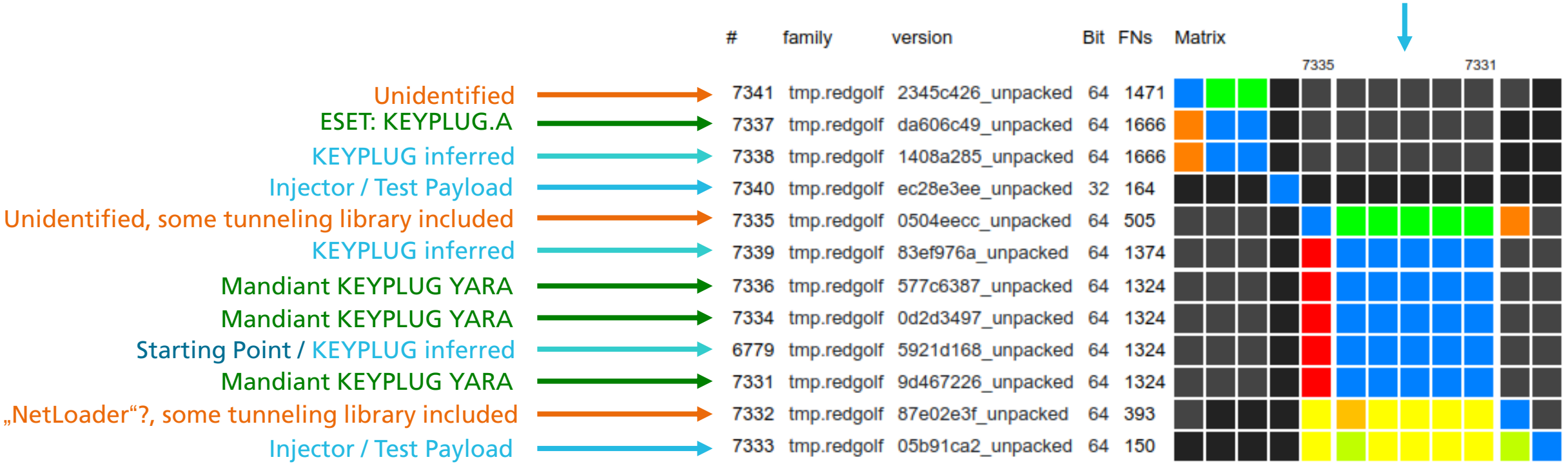
Fraunhofer FKIE

Example Use Cases

Clustering: KEYPLUG and Friends

- Running them through MCRIT
- Labeling of clusters / samples via other sources

Match all samples against each other.
Meaning of colors as before

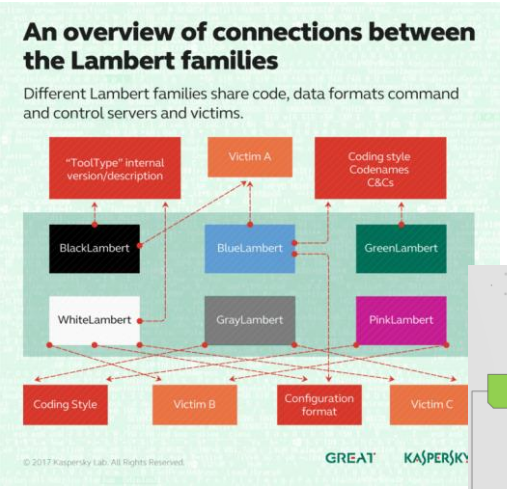


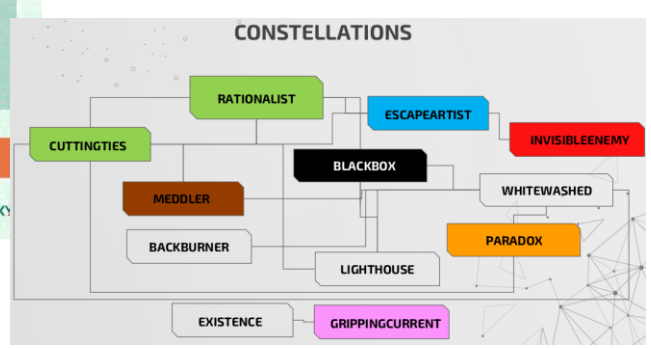
Fraunhofer FKIE

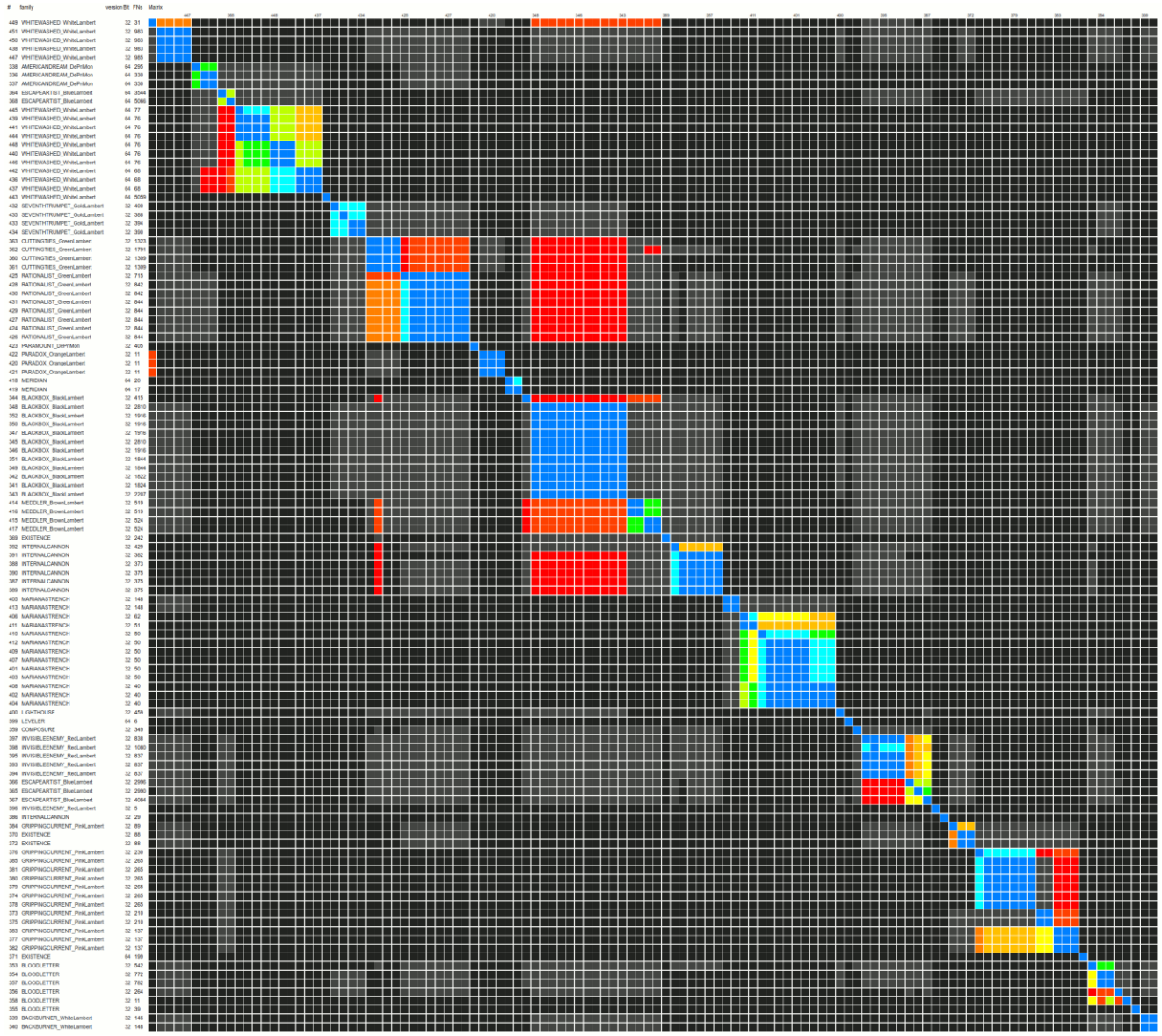
Example Use Cases

Clustering: „Bright Constellation“

- Oh colorful Lamberts…
  - Write-ups by various institutions
  - revisited in Sep 2022 by Greg Lesnewich who shared this data set with me (THX!)
- Experiment: Cluster verification using MCRIT







- MCRIT Processing time: ~35min.

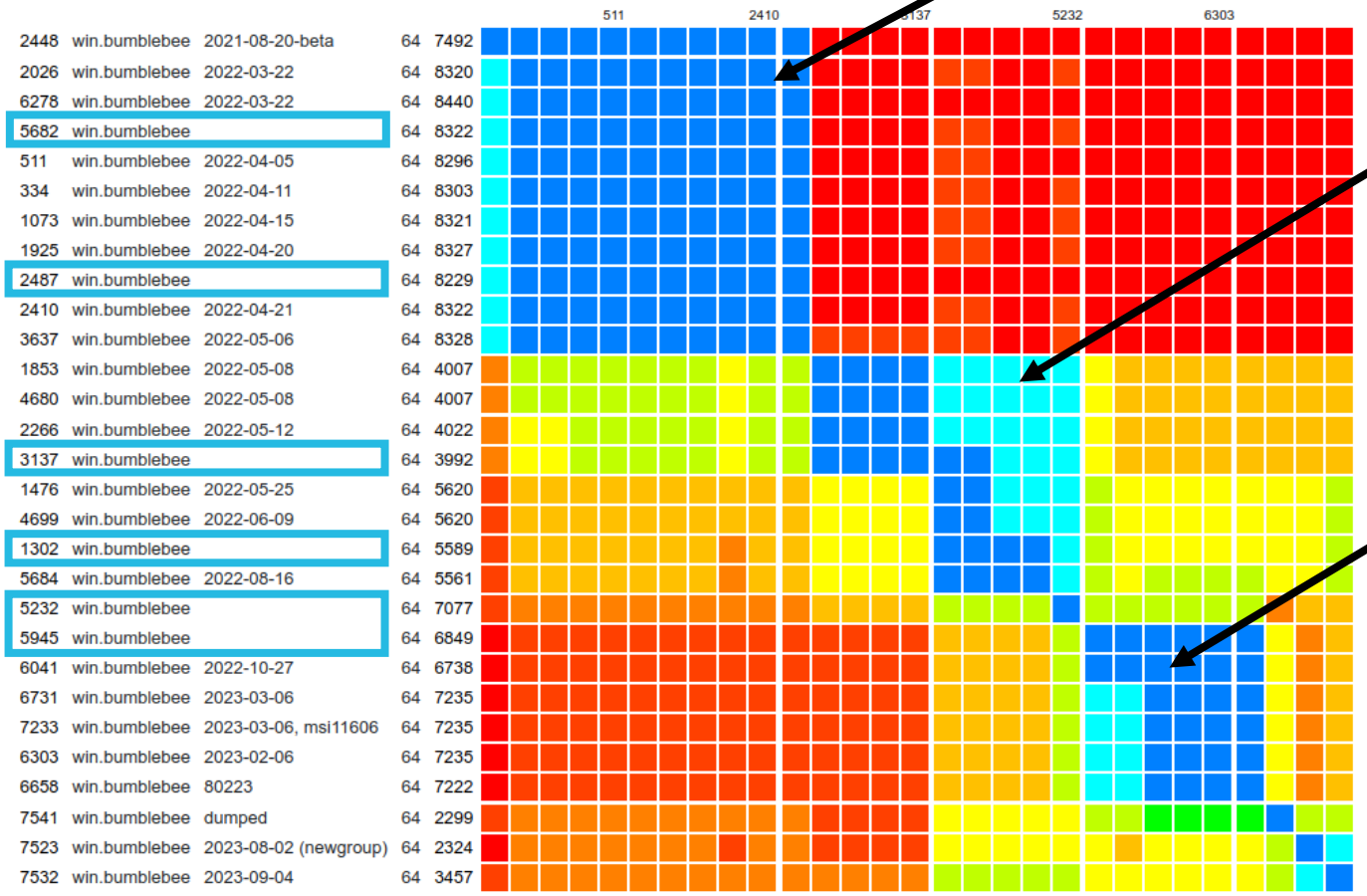
[1] https://securelist.com/unraveling-the-lamberts-toolkit/77990/
[2] https://www.youtube.com/watch?v=aaV7UieJ_I4

Fraunhofer FKIE

Example Use Cases

Clustering: BumbleBee

#	family	version	Bit	FNs
2448	win.bumblebee	2021-08-20-beta	64	7492
2026	win.bumblebee	2022-03-22	64	8320
6278	win.bumblebee	2022-03-22	64	8440
5682	win.bumblebee		64	8322
511	win.bumblebee	2022-04-05	64	8296
334	win.bumblebee	2022-04-11	64	8303
1073	win.bumblebee	2022-04-15	64	8321
1925	win.bumblebee	2022-04-20	64	8327
2487	win.bumblebee		64	8229
2410	win.bumblebee	2022-04-21	64	8322
3637	win.bumblebee	2022-05-06	64	8328
1853	win.bumblebee	2022-05-08	64	4007
4680	win.bumblebee	2022-05-08	64	4007
2266	win.bumblebee	2022-05-12	64	4022
3137	win.bumblebee		64	3992
1476	win.bumblebee	2022-05-25	64	5620
4699	win.bumblebee	2022-06-09	64	5620
1302	win.bumblebee		64	5589
5684	win.bumblebee	2022-08-16	64	5561
5232	win.bumblebee		64	7077
5945	win.bumblebee		64	6849
6041	win.bumblebee	2022-10-27	64	6738
6731	win.bumblebee	2023-03-06	64	7235
7233	win.bumblebee	2023-03-06, msi11606	64	7235
6303	win.bumblebee	2023-02-06	64	7235
6658	win.bumblebee	80223	64	7222
7541	win.bumblebee	dumped	64	2299
7523	win.bumblebee	2023-08-02 (newgroup)	64	2324
7532	win.bumblebee	2023-09-04	64	3457



Initial Version

Internal change of library use?
Extensions of tasks [1]

WebSocket protocol introduced [1]

- MCRIT Processing time: ~20min

[1] “Tracking Bumblebee’s Development”. Suweera De Souza, 2023.

Fraunhofer FKIE

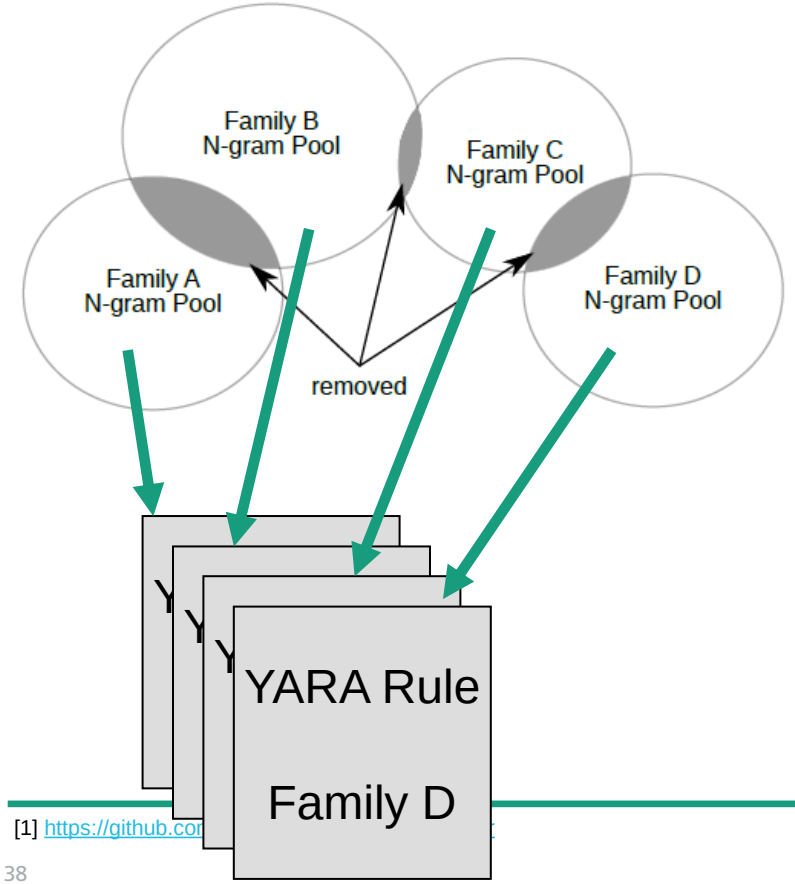
Use Cases and Case Studies

Isolation of Unique Family Code

Example Use Cases

Isolation of Unique Family Code

- Essentially like YARA-Signator, but with basic blocks



[1] https://github.com

Unique Block Isolation Report

Job ID	639358a4e0ff5413a77221e4
Family	win.remcos
Samples	19
Unique Blocks	10028
Has a YARA rule?	True, covers: 19 samples
YARA rule covers all?	True

Statistics | Unique Blocks | YARA Rule

Block Statistics across Samples

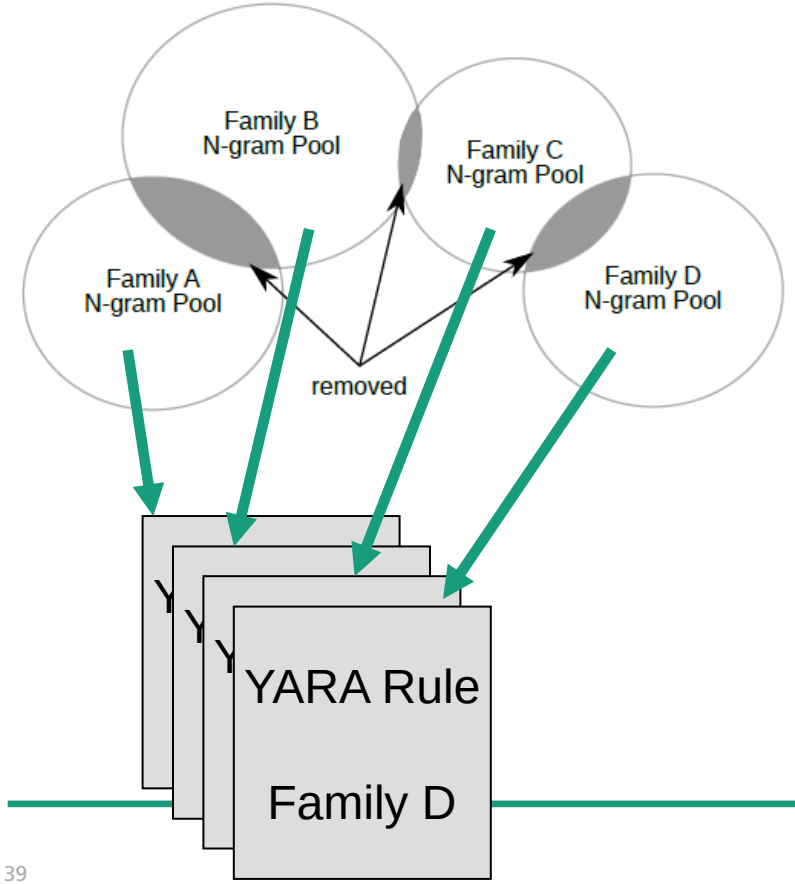
Characteristic blocks are basic blocks only found in this collection of samples (versus rest of the whole data set), unique blocks are only found in the specific sample.

Sample ID	Total Blocks	Characteristic Blocks	Unique Blocks
1111	1104	965 (87.41%)	0 (0.00%)
1342	1178	1056 (89.64%)	567 (48.13%)
1343	1106	967 (87.43%)	1 (0.09%)
1519	746	548 (73.46%)	68 (9.12%)
1968	8020	3423 (42.68%)	188 (2.34%)
2217	867	743 (85.70%)	266 (30.68%)
3729	1190	1041 (87.48%)	32 (2.69%)
4501	8307	3937 (47.39%)	0 (0.00%)
4561	8315	3938 (47.36%)	0 (0.00%)
4575	1299	1137 (87.53%)	74 (5.70%)
4683	876	745 (85.05%)	49 (5.59%)

Example Use Cases

Isolation of Unique Family Code

- Essentially like YARA-Signator, but with basic blocks



Explore Unique Blocks

Filter blocks to min score (0-100) | min block length | max block length

filter

total: 10028, showing: 1 - 100

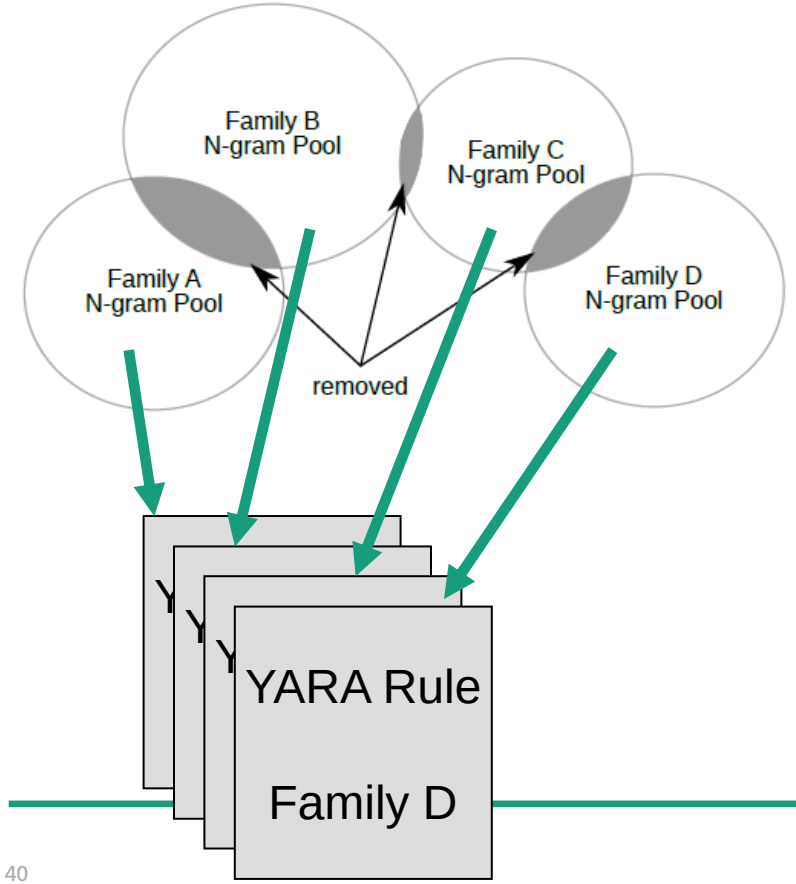
Score	PicBlockHash	Samples	Instructions	Function ID	Block
97.14	0xcadb40567ec2656	19 / 19	6	470589	/* picblockhash: 0xcadb40567ec2656 * 6a09 | push 9 * ff35fc8d4100 | push dword ptr [0x418dfc] * ff1554244100 | call dword ptr [0x412454] * ff35fc8d4100 | push dword ptr [0x418dfc] * ff1588244100 | call dword ptr [0x412488] * eb38 | jmp 0x41157f */ { 6a09 ff35???????? ff15???????? ff35???????? ff15???????? eb?? }
83.16	0x1bb7e1ed48b1d751	15 / 19	7	470361	/* picblockhash: 0x1bb7e1ed48b1d751 * 53 | push ebx * 53 | push ebx * 56 | push esi * 68ed524000 | push 0x4052ed * 53 | push ebx * 53 | push ebx * ffd7 | call edi */ { 53 53 56 68???????? 53 53 ffd7 }
80.30	0x38bb1b656ac38756	15 / 19	6	470428	/* picblockhash: 0x38bb1b656ac38756 * 6890374100 | push 0x413790 * 6874374100 | push 0x413774 * ffd7 | call edi * 50 | push eax * ffd6 | call esi * a3a48a4100 | mov dword ptr [0x418aa4], eax */ { 68???????? 68???????? ffd7 50 ffd6 a3???????? }

Fraunhofer FKIE

Example Use Cases

Isolation of Unique Family Code

- Essentially like YARA-Signator, but with basic blocks



Proposed YARA rule

Copy rule to clipboard!

```
rule mcrit_639358a4e0ff5413a77221e4 {
    meta:
        author = "MCRIT YARA Generator"
        description = "Code-based YARA rule composed from potentially unique basic blocks for the selected set of samples/family."
        date = "2023-03-29"
    strings:
        // Rule generation selected 20 picblocks, covering 19/19 input sample(s).
        /* picblockhash: 0x1bb7e1ed48b1d751 - coverage: 15/19 samples.
         * 53         | push ebx
         * 53         | push ebx
         * 56         | push esi
         * 68ed524000 | push 0x4052ed
         * 53         | push ebx
         * 53         | push ebx
         * ffd7       | call edi
         */
        $blockhash_0x1bb7e1ed48b1d751 = { 53 53 56 68???????? 53 53 ffd7 }

        /* picblockhash: 0x2e684020db1f147c - coverage: 14/19 samples.
         * ff7508       | push dword ptr [ebp + 8]
         * 6a00         | push 0
         * 6800040000   | push 0x400
         * ff15cc204100 | call dword ptr [0x4120cc]
         * 8d4d08       | lea ecx, [ebp + 8]
         * 51           | push ecx
         * 50           | push eax
         * ff15e48d4100 | call dword ptr [0x418de4]
         * 837d0800     | cmp dword ptr [ebp + 8], 0
         * 7504         | jne 0x410ae5
         */
        $blockhash_0x2e684020db1f147c = { ff7508 6a00 6800040000 ff15???????? 8d4d08 51 50 ff15???????? 837d0800 75?? }

        /* picblockhash: 0x38bb1b656ac38756 - coverage: 15/19 samples.
         * 6890374100 | push 0x413790
         * 6874374100 | push 0x413774
         * ffd7       | call edi
         * 50         | push eax
         * ffd6       | call esi
         * a3a48a4100 | mov dword ptr [0x418aa4], eax
         */
        $blockhash_0x38bb1b656ac38756 = { 68???????? 68???????? ffd7 50 ffd6 a3???????? }

        /* picblockhash: 0x3fc01ffe2622434d - coverage: 14/19 samples.
         * ff7508       | push dword ptr [ebp + 8]
         * ff36         | push dword ptr [esi]
         * ff15ac234100 | call dword ptr [0x4123ac]
         * 59           | pop ecx
         * 85c0         | test eax, eax
         * 59           | pop ecx
         * 741f         | je 0x40f2dc
```


Fraunhofer FKIE

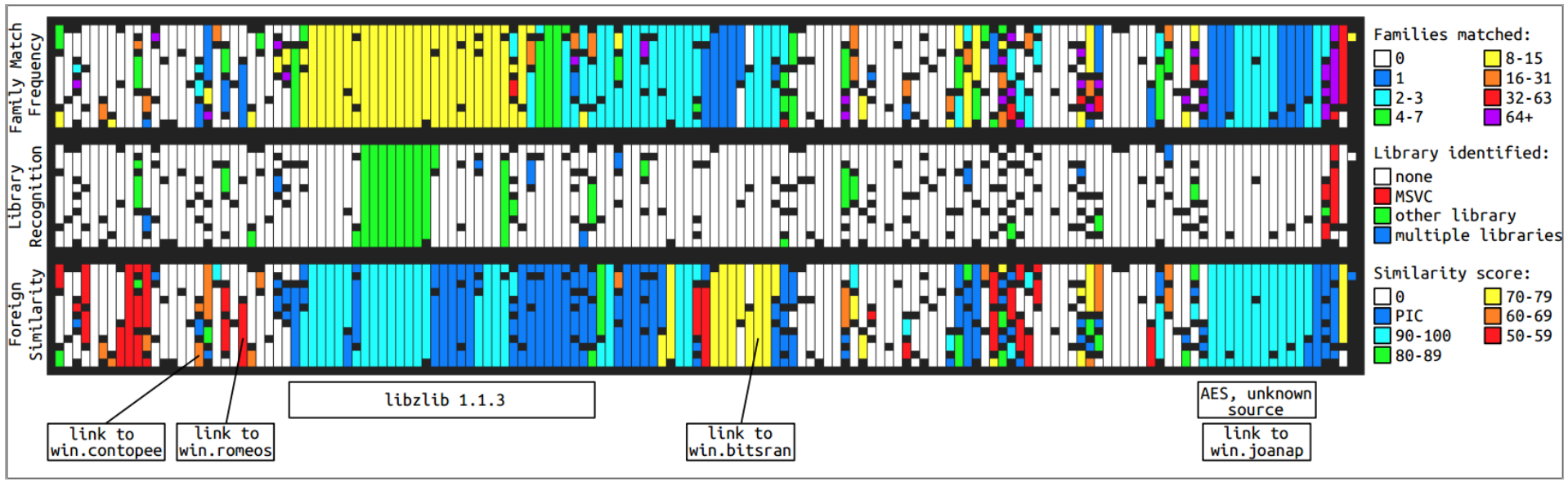
Use Cases and Case Studies

Lead Generation

Example Use Cases

Lead Generation: Reconstructing the WannaCry Hunt

- May 15th 2017 - Tweet by Neel Mehta (Google) with hashes + offsets
  - Earlier version of WannaCry sharing „rare“ code with Contopee
- Identification of similar functions with appearance across few families
  - Potential reuse of non-public code as an indicator for relationship



[1] https://twitter.com/neelmehta/status/864164081116225536
[2] „Classification, Characterization, and Contextualization of Windows Malware using Static Behavior and Similarity Analysis“, D. Plohmann, 2022.

Fraunhofer FKIE

Example Use Cases

Lead Generation: Reconstructing the WannaCry Hunt

Function Matches

selection: 49, showing: 1 - 49 (filtered: 64916)

Filter results to 65 | max score (0-100) | 3

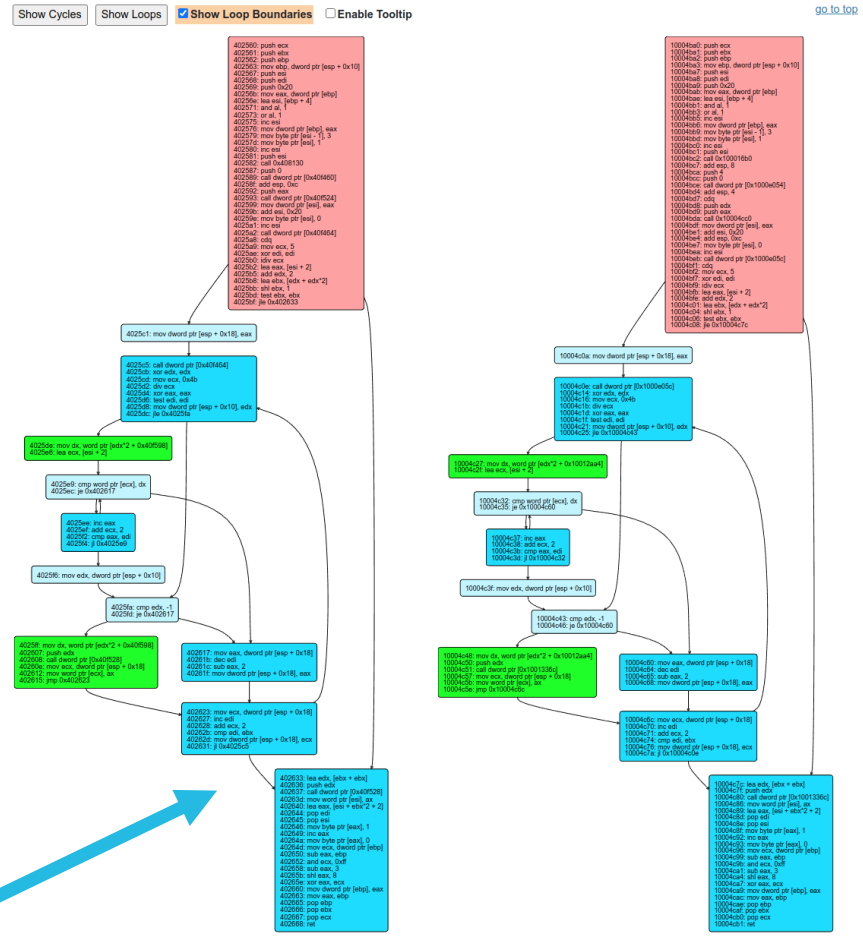
Filter to offset 0x0

exclude functions with library hits | exclude PIC hits | unique only

filter | clear

Function ID	offset	num_bytes	Matched Families	Matched Samples	Matched Functions	Best Score	Min	Pic	Lib	Uniq
1344938	0x401000	58	win.wannacryptor	5	5	100	5	5	0	✓
1344939	0x401040	173	win.wannacryptor	3	3	100	3	3	0	✓
1344941	0x401120	252	win.wannacryptor	3	3	100	3	3	0	✓
1344942	0x401220	69	win.wannacryptor	3	3	100	3	3	0	✓
1344948	0x401360	61	win.wannacryptor	5	5	100	5	3	0	✓
1344953	0x401790	278	win.wannacryptor	5	5	90	5	0	0	✓
1344955	0x4018d0	183	win.wannacryptor	5	5	68	5	0	0	✓
1344961	0x401dd0	42	win.wannacryptor	5	5	100	5	5	0	✓
1344965	0x401e40	145	win.wannacryptor	5	5	100	5	5	0	✓
1344966	0x401ee0	190	win.wannacryptor	5	5	100	5	5	0	✓
1344967	0x401fa0	181	win.wannacryptor	5	5	84	5	0	0	✓
1344968	0x402060	344	win.wannacryptor	5	5	84	5	0	0	✓
1344969	0x4021c0	77	win.wannacryptor	5	5	78	5	0	0	✓
1344970	0x402210	451	win.wannacryptor	5	5	95	5	0	0	✓
1344971	0x4023e0	214	win.wannacryptor	5	5	82	5	0	0	✓
1344975	0x402500	55	3	7	7	100	7	7	0	✗
1344977	0x402560	265	win.contopee	1	1	67	1	0	0	✓

Function CFGs



Fraunhofer FKIE

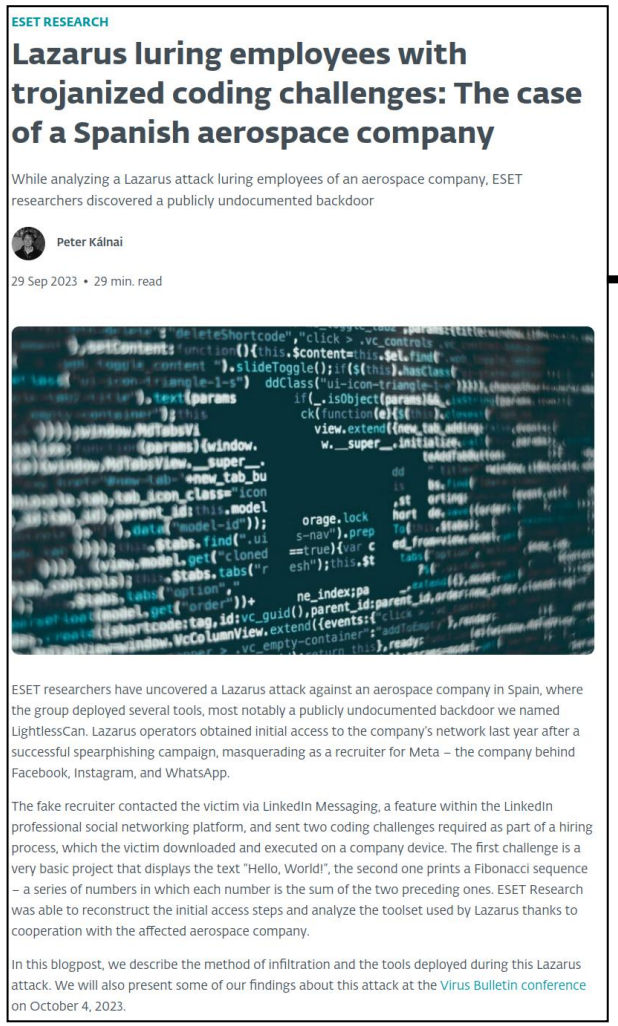
Example Use Cases

Lead Generation: New LinkHunt Feature

- LinkHunt
  - Consider ICFG relationship of functions in a matched sample
    - Do multiple connected functions match another family?
    - If yes, highlight such clusters
  - Otherwise rank functions by combined
    - Matching score
    - Size
    - Position in binary (front)

Example Use Cases

Lead Generation: Recent Lazarus Reporting



ESET RESEARCH

Lazarus luring employees with trojanized coding challenges: The case of a Spanish aerospace company

While analyzing a Lazarus attack luring employees of an aerospace company, ESET researchers discovered a publicly undocumented backdoor

Peter Kálnai

29 Sep 2023 • 29 min. read

ESET researchers have uncovered a Lazarus attack against an aerospace company in Spain, where the group deployed several tools, most notably a publicly undocumented backdoor we named LightlessCan. Lazarus operators obtained initial access to the company's network last year after a successful spearphishing campaign, masquerading as a recruiter for Meta – the company behind Facebook, Instagram, and WhatsApp.

The fake recruiter contacted the victim via LinkedIn Messaging, a feature within the LinkedIn professional social networking platform, and sent two coding challenges required as part of a hiring process, which the victim downloaded and executed on a company device. The first challenge is a very basic project that displays the text "Hello, World!", the second one prints a Fibonacci sequence – a series of numbers in which each number is the sum of the two preceding ones. ESET Research was able to reconstruct the initial access steps and analyze the toolset used by Lazarus thanks to cooperation with the affected aerospace company.

In this blogpost, we describe the method of infiltration and the tools deployed during this Lazarus attack. We will also present some of our findings about this attack at the Virus Bulletin conference on October 4, 2023.

Execution chain 1: miniBlindingCan

One of the payloads downloaded and executed by NickelLoader is miniBlindingCan, a simplified version of the group's flagship BlindingCan RAT. It was reported for the first time by Mandiant in September 2022, under the name AIRDRY.V2.

	Version		SHA256	Filename	Bitness	FNs	Min#	Pic#	Lib	Direct		Frequency		Uniq
win.blindingcan	2020-05-19	3689	58027c80	58027c80c6...d_unpacked	64	348	200	188	182	51	15	9	5	1.37%
win.snatchcrypto		986	bde53bd6	156d33cd77...0180000000	64	2087	221	206	209	52	8	8	2	0.40%
win.neddnloader		7368	e05344f6	454734dca5...6_unpacked	64	263	165	155	150	39	16	8	7	2.04%
win.cloudburst		6367	e1ecf0f7	e1ecf0f7bd...0_unpacked	64	2996	218	207	211	53	5	7	0	0.00%
win.banpolmex		7490	21c83fe2	21c83fe249...f_unpacked	64	2165	220	212	210	53	8	7	1	0.00%
Unnamed		7428	333b4da6	333b4da636...218b52c979	64	5070	218	207	209	52	8	7	1	0.00%
win.soul	soulsearcher	1208	844b9ce5	1af5252cad...0180000000	64	634	219	213	209	51	8	7	1	0.00%
win.unidentified_087		795	1af5252c	1af5252cad...f_unpacked	64	635	219	213	209	51	8	7	1	0.00%
win.bookcodesrat		7277	7695e619	7695e619cb...e_unpacked	64	800	219	213	208	51	8	7	1	0.00%
win.pocodown	2018-07-04	4028	e4a99f7b	b40909ac0b...0180000000	64	9890	221	206	213	50	7	7	1	0.00%

- MCRIT Processing time:
  - Disassembly: 6sec -> 950 functions
  - Matching: 21sec

win.blindingcan (Back to overview)
BLINDINGCAN
aka: AIRDRY, ZetaNile
Actor(s): Lazarus Group

win.snatchcrypto (Back to overview)
SnatchCrypto
Actor(s): Lazarus Group

win.neddnloader (Back to overview)
NedDnLoader
Actor(s): Lazarus Group

malpedia

[1] https://www.welivesecurity.com/en/eset-research/lazarus-luring-employees-trojanized-coding-challenges-case-spanish-aerospace-company/
[2] f69198afcd237af265f1f48203d0db8f5f07cdf53a0f410e4d49f77434b53cff

Fraunhofer FKIE

Example Use Cases

Lead Generation: LinkHunt in Action

Link Clusters

These are clusters of matches with another single family where all functions have a direct code CFG relationship (calls, jumps, ...) among them. Bold matches are also unique with this other family.

Rank	Clusterscore	Best Linkscore	Family	Cluster Size	Unique Matches	Functions		
1	163.93	90.58	(472) win.blindingcan	3	1	**0x18000be80**	0x180002d14	0x1800027d0
2	112.67	73.84	(27) win.neddnloader	2	0		0x180001000	0x180001444

Best Individual Links

Family-unique matches in bold.

Rank	Linkscore	Function ID	Offset	num_bytes	Family	Sample ID	Function ID	Match Score
1	90.58	9595275	0x1800027d0	1347	(472) win.blindingcan	3689	4087967	100.00
2	90.53	9595279	0x180003e38	414	(472) win.blindingcan	3689	**4087998**	100.00
3	90.39	9595321	0x18000b5e8	381	(472) win.blindingcan	3689	**4088036**	100.00
4	86.20	9595275	0x1800027d0	1347	(27) win.neddnloader	7368	9197730	93.75
5	81.88	9595313	0x18000a7d0	171	(472) win.blindingcan	3689	**4088045**	100.00
6	80.70	9595319	0x18000b45c	154	(472) win.blindingcan	3689	**4088034**	100.00
7	75.99	9595285	0x180005330	79	(474) win.spyder	1417	1376061	100.00
8	75.99	9595285	0x180005330	79	(27) win.neddnloader	7368	9197740	100.00
9	73.84	9595257	0x180001444	1041	(27) win.neddnloader	7368	9197762	76.56
10	71.60	9595324	0x18000be80	499	(472) win.blindingcan	3689	**4088049**	73.44
11	67.41	9595294	0x180005dd0	255	(34) win.colony	480	443267	71.88
12	66.18	9595257	0x180001444	1041	(1710) win.httpsuploader	7280	8991609	65.62
13	64.44	9595303	0x180006b2c	142	(191) win.easynight	4869	5407333	78.12
14	63.03	9595294	0x180005dd0	255	(1040) win.pocodown	4028	4442855	65.62
15	59.59	9595285	0x180005330	79	(1737) win.unidentified_106	7366	9190548	76.56
16	57.88	9595303	0x180006b2c	142	(1652) win.r77	6741	7855204	68.75
17	57.88	9595303	0x180006b2c	142	(0) Unnamed	7550	9583145	68.75

Function Comparison: 9595324 vs. 4088049

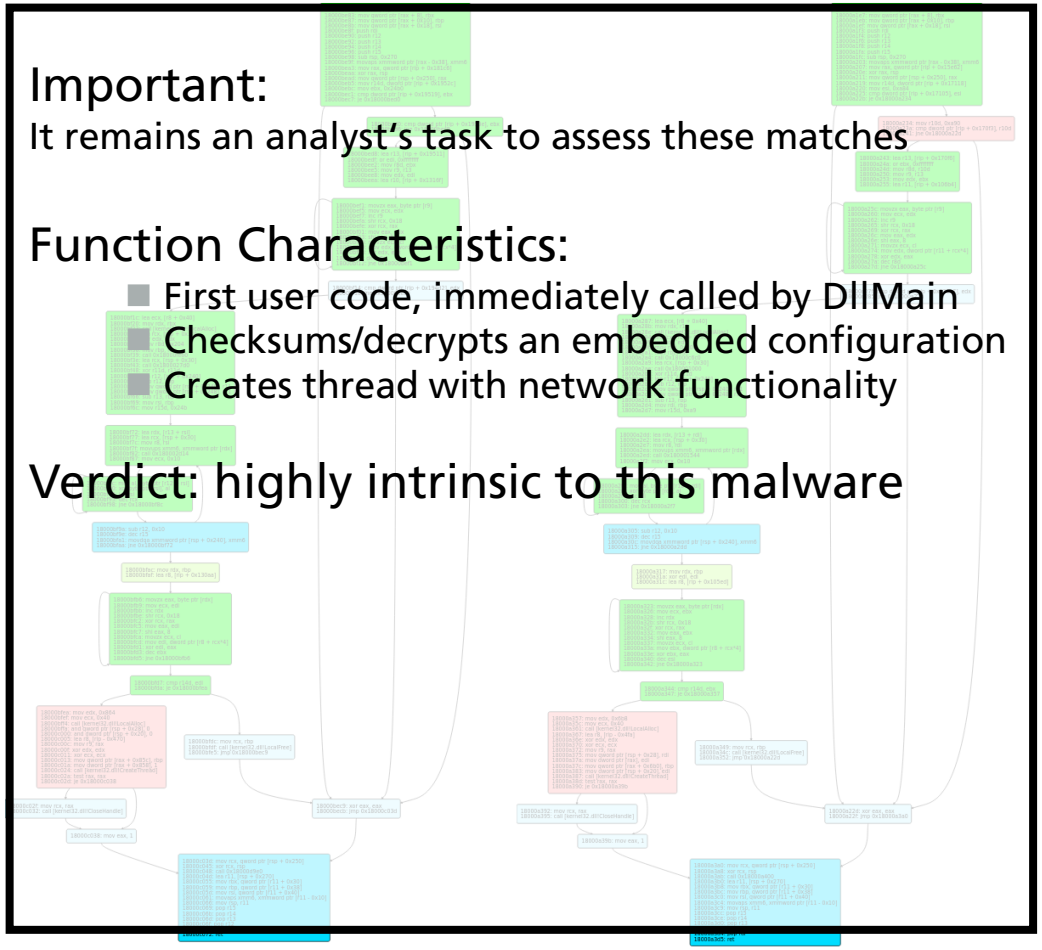
go to cfg

Architecture	intel	intel
Binweight	499	498
Family ID	1791	472
Family Name	unid.lazarus	win.blindingcan
Sample ID	7560	3689
Short Sha256	f69198af	58027c80
Function ID	9595324	4088049
Function Name		
Num Blocks	18	18
Num Instructions	126	128
Offset	0x18000be80	0x18000a1e4
PicHash	0xb38855d9b62981d5 (1 , 1 , 1)	0xdf6b227d8d2df497 (1 , 1 , 1)
Matching Score	73.44	

Function CFGs

Show Cycles | Show Loops | Show Loop Boundaries | Enable Tooltip

go to top

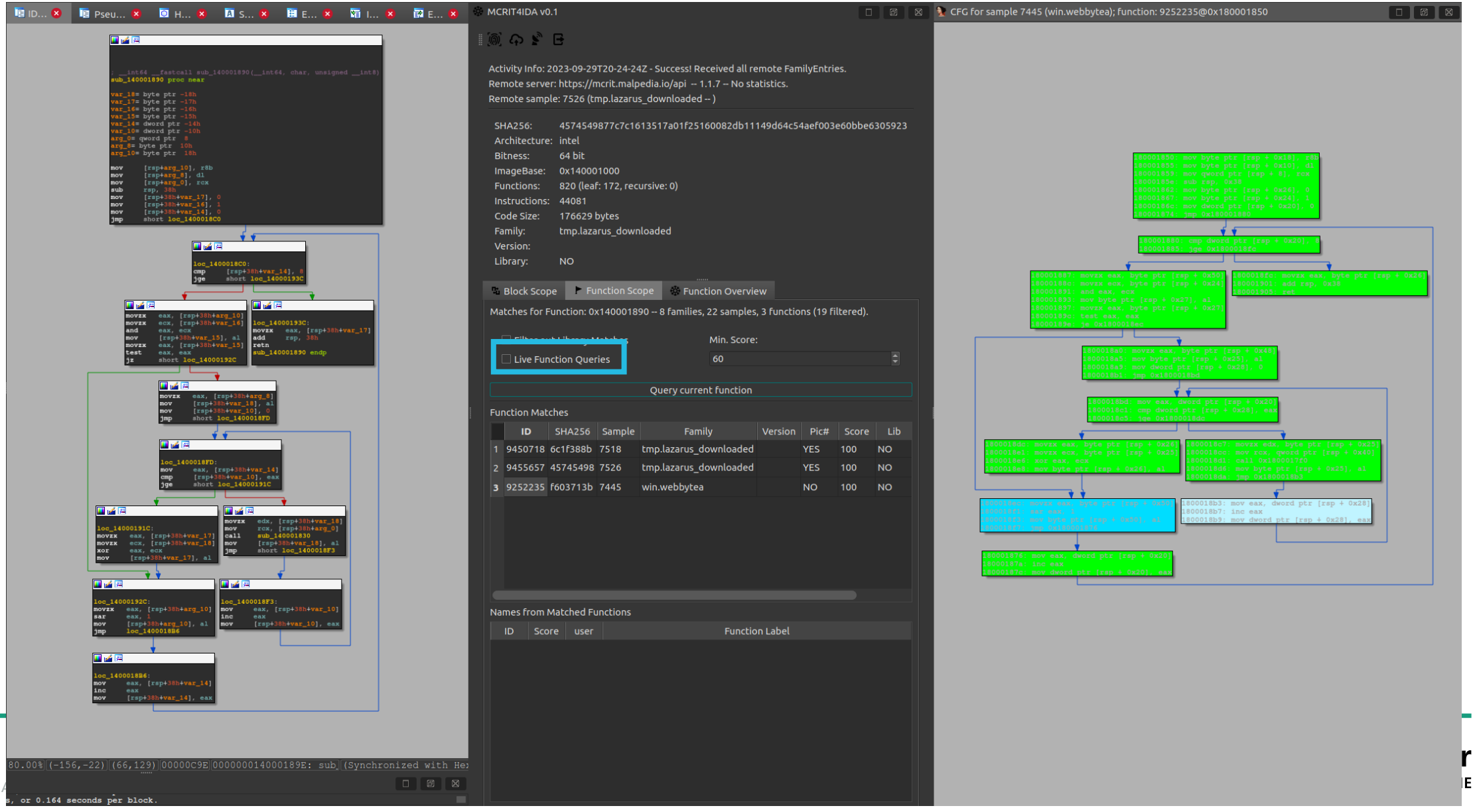


Use Cases and Case Studies

IDA Plugin, Label Transfer, Reference Data

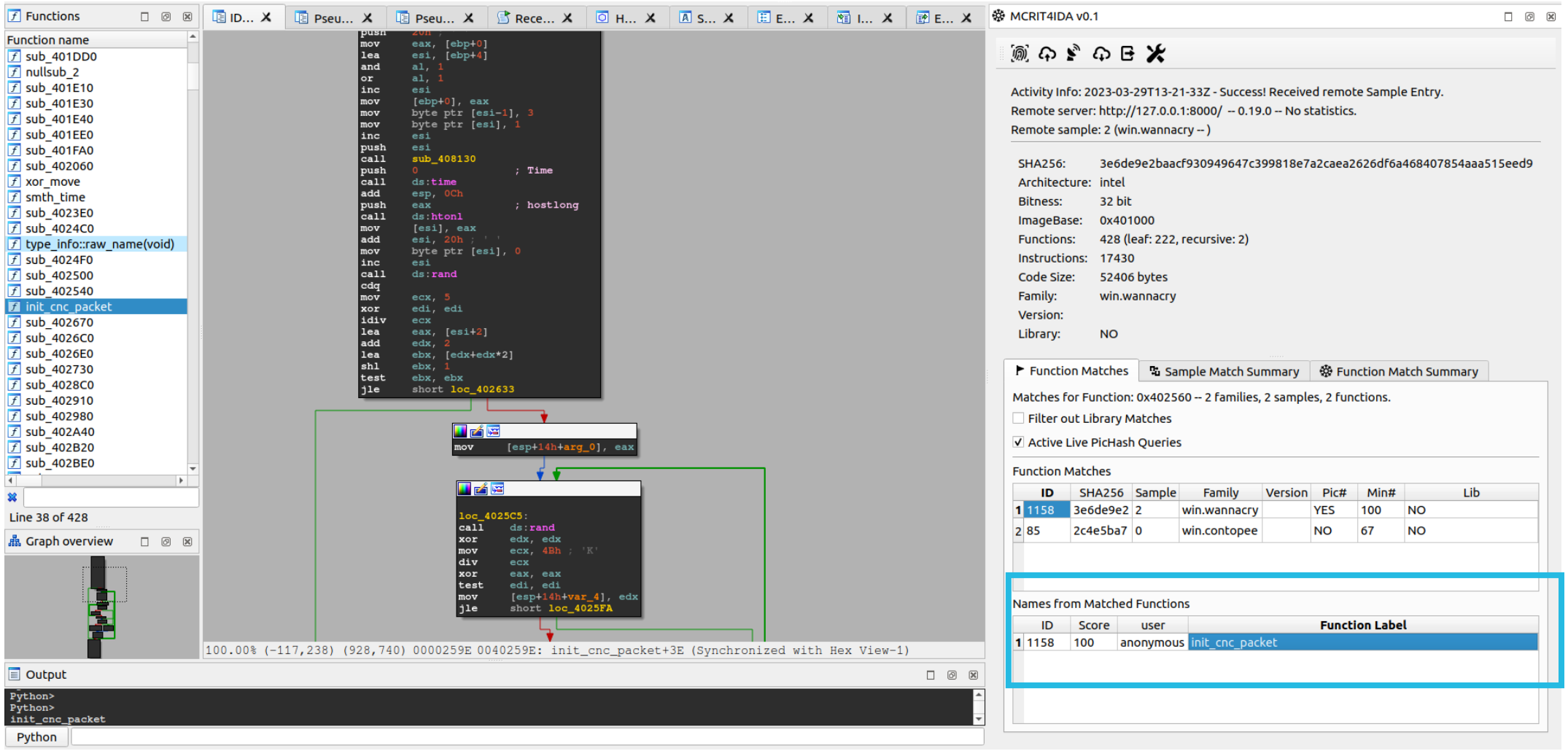
Example Use Cases

IDA Plugin



Example Use Cases

Label Transfer



Fraunhofer FKIE

Example Use Cases
Reference Data

- MCRIT-Data [1]
  - Parsed *.lib/obj files from various sources
  - Long-term goal: cover commonly encountered statically linked code
  - Currently: compilers (MSVC, MinGW, Golang, Nim, Rust) and libraries (aPLib, zlib)
    - 890k functions with symbols
  - Ready to use with MCRIT via Import functionality

[1] https://github.com/danielplohmann/mcrit-data

Summary & Outlook

Summary and Outlook
MCRIT

- Minhash-based Code Relationship & Investigation Toolkit (MCRIT)
  - A framework for quasi-identical and fuzzy 1:n code matching
  - Use Cases: Code identification & library filtering, hunting, label transfer, …
  - Open Source, convenient deployment via Docker [1]

- Outlook
  - Scalability improvements
  - Further usability refinement based on user feedback

[1] https://github.com/danielplohmann/docker-mcrit

Thank you for your attention!

Dr. Daniel Plohmann
daniel.plohmann@fkie.fraunhofer.de
 @push_pnx

Fraunhofer FKIE