Family Ties: A Close Look at the Influence of Static Features on the Precision of Malware Family Clustering

Antonino Vitale, **Kevin van Liebergen**, Juan Caballero, Savino Dambra,
Platon Kotzias, Simone Aonzo, Davide Balzarotti

EURECOM Sophia Antipolis

institute imdea software

BforeAI

Gen™

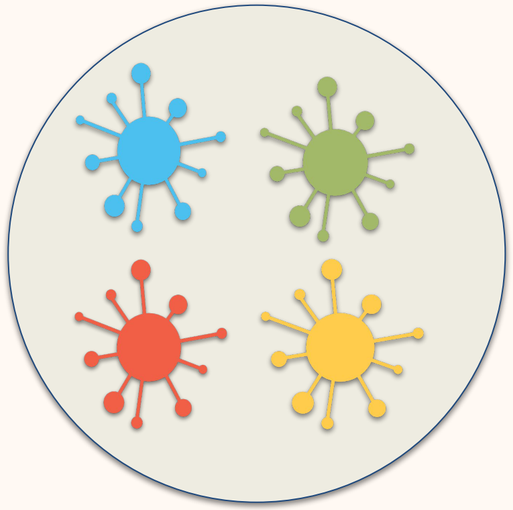
Malware Explosion

Millions of new malware daily

Polymorphism create many variants

Impossible to manually analyze all

Malware Family Clustering



Malware Family

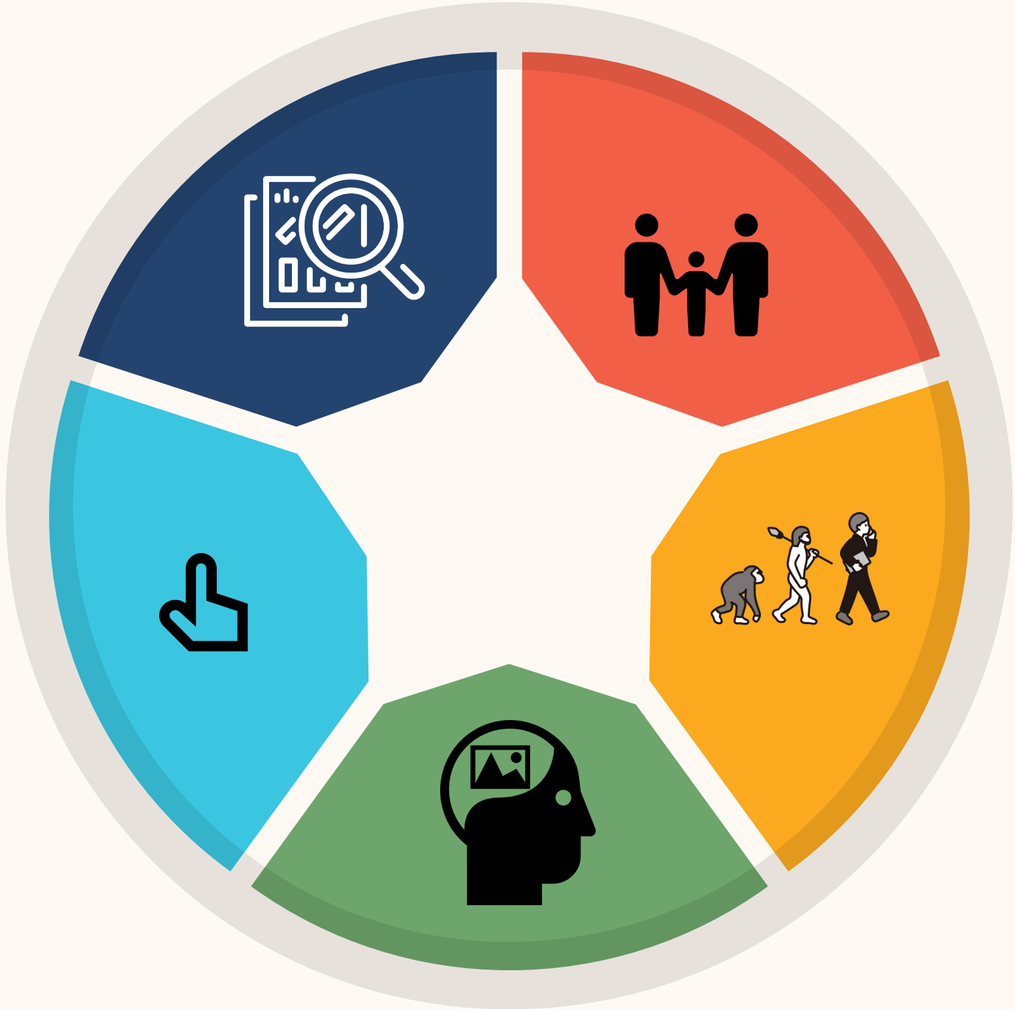
- Samples derived from same code base
- Share common:
  - Characteristics
  - Behavior
  - Attribution to same authors

- **One representative sample is often enough to understand the family's behavior**

Family Clustering Use Cases

Analysis Reduction

Attribution



Lineage Reconstruction

Malware Evolution

Behavior Understanding

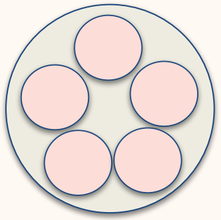
Clustering Accuracy

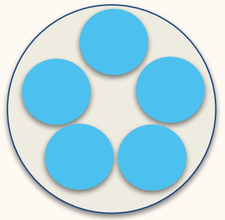
Malware Family A

Malware Family B

Precision

Recall



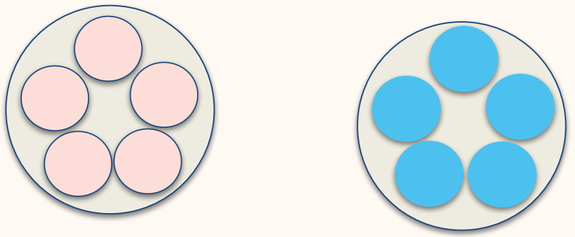


Clustering Accuracy

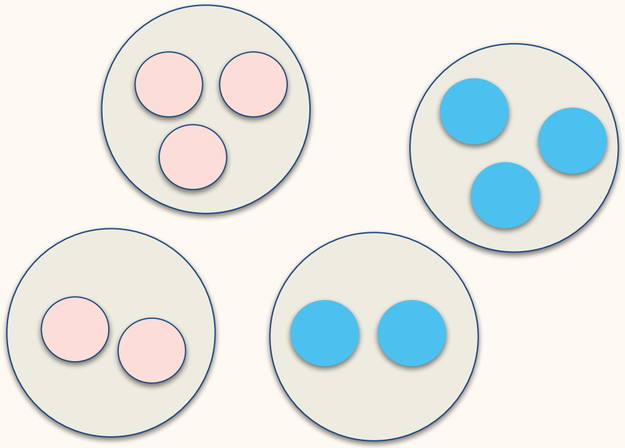
Malware Family A

Malware Family B

Precision Recall



Precision Recall

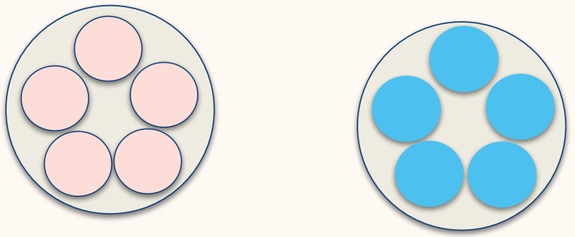


Clustering Accuracy

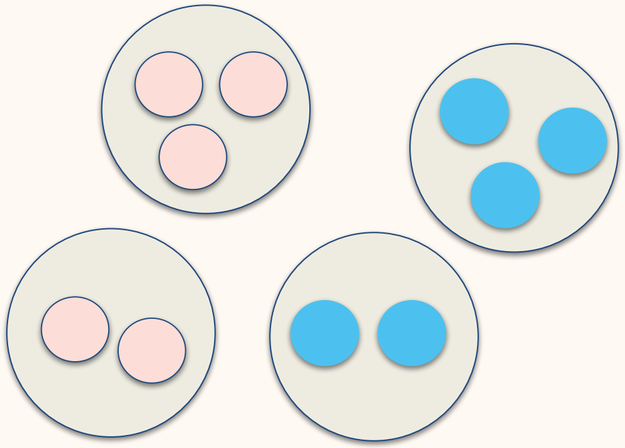
Malware Family A

Malware Family B

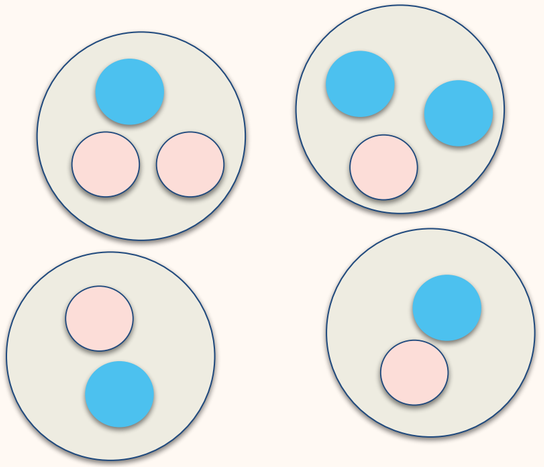
Precision Recall



Precision Recall



Precision

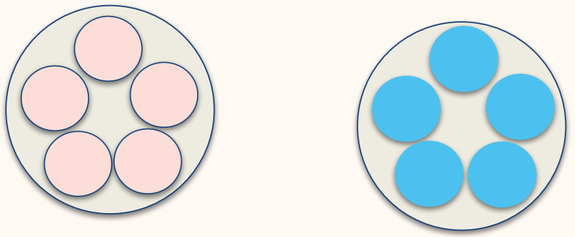


Clustering Accuracy

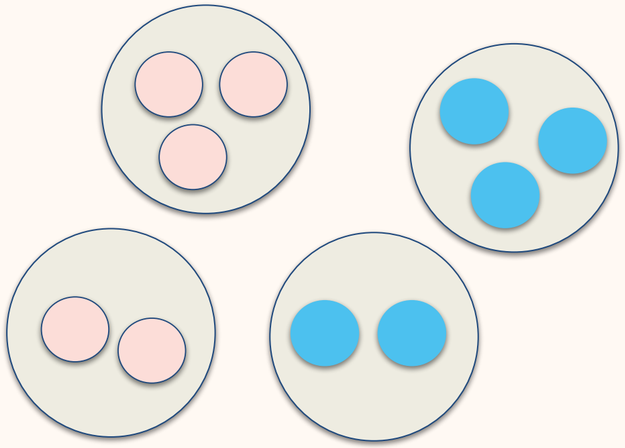
Malware Family A

Malware Family B

Precision Recall

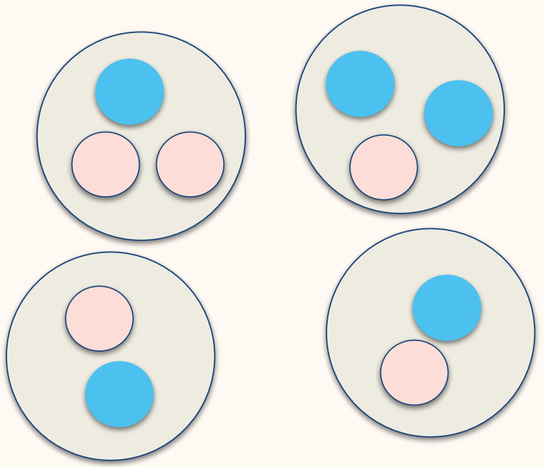


Precision Recall

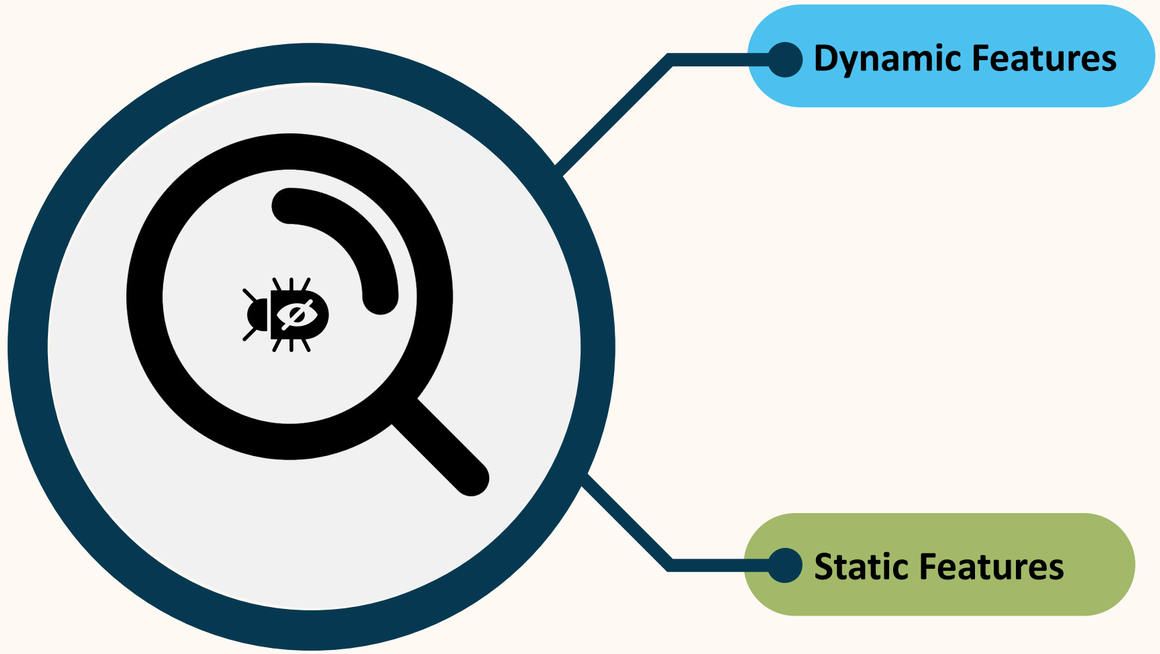


Precision

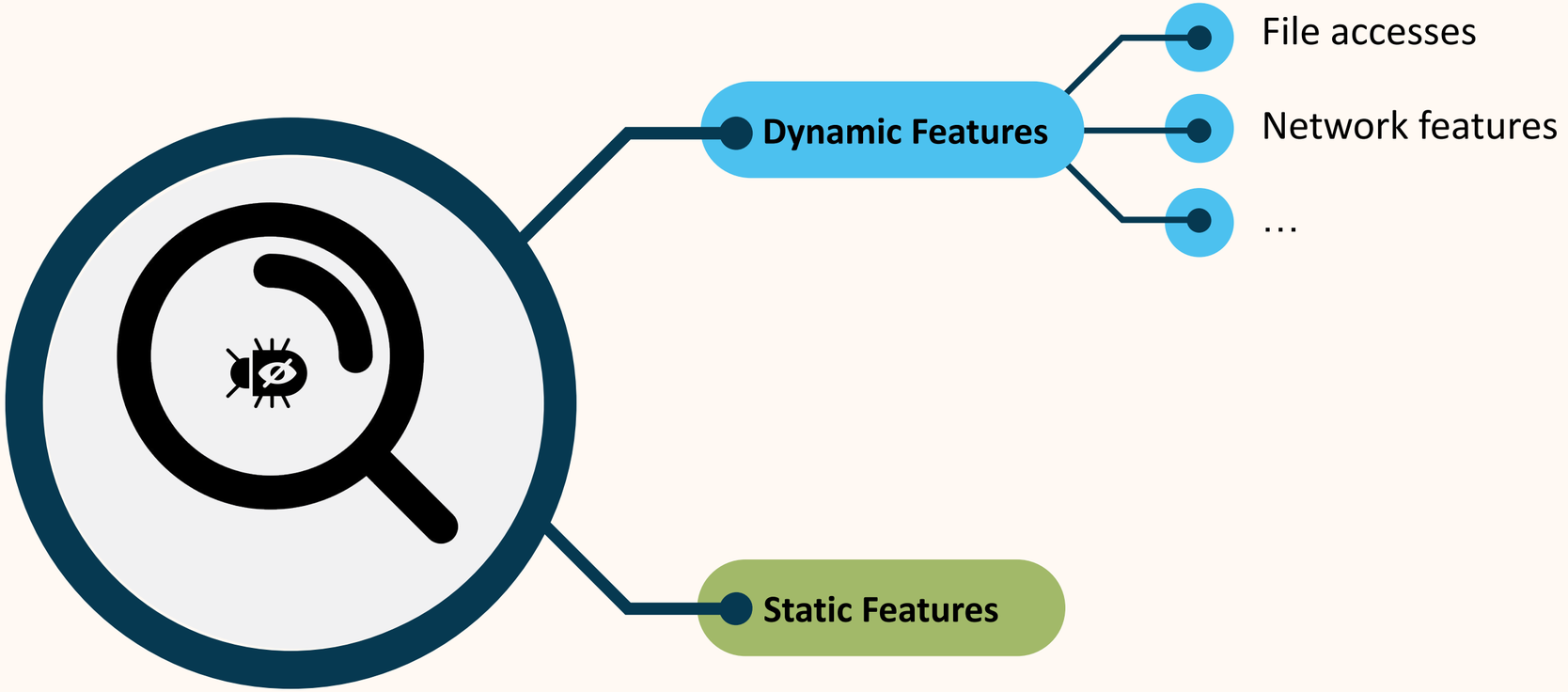
Mixed clusters



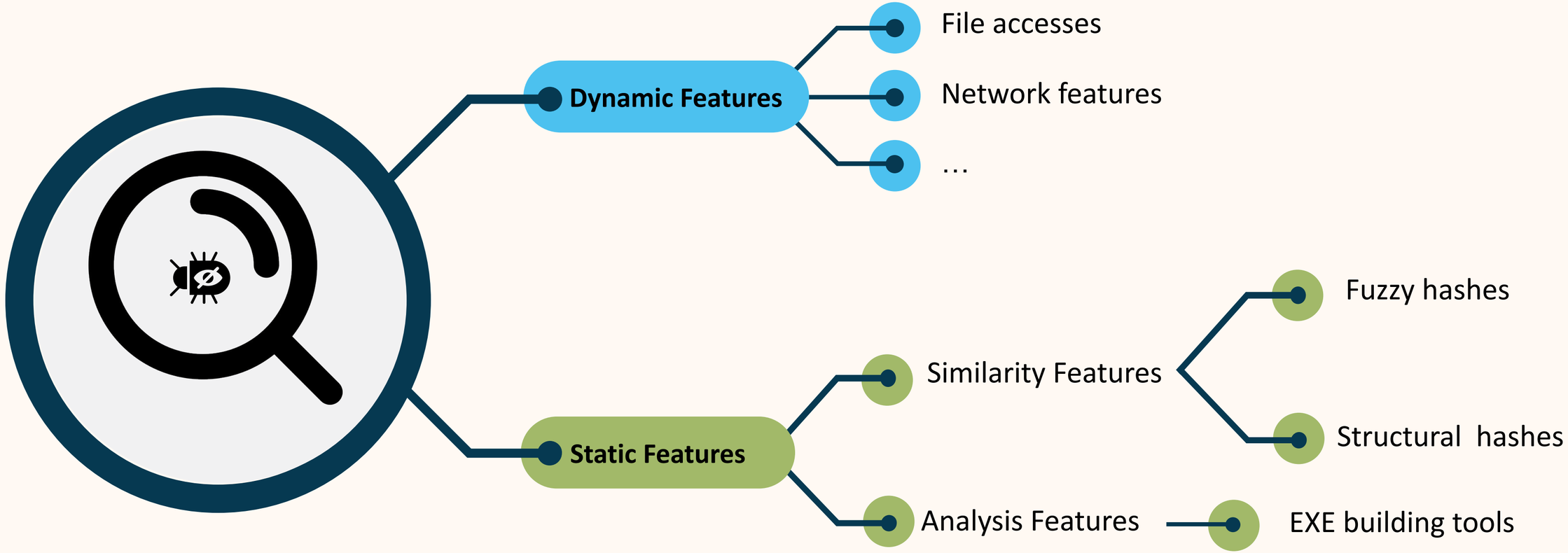
Feature Types



Feature Types



Feature Types



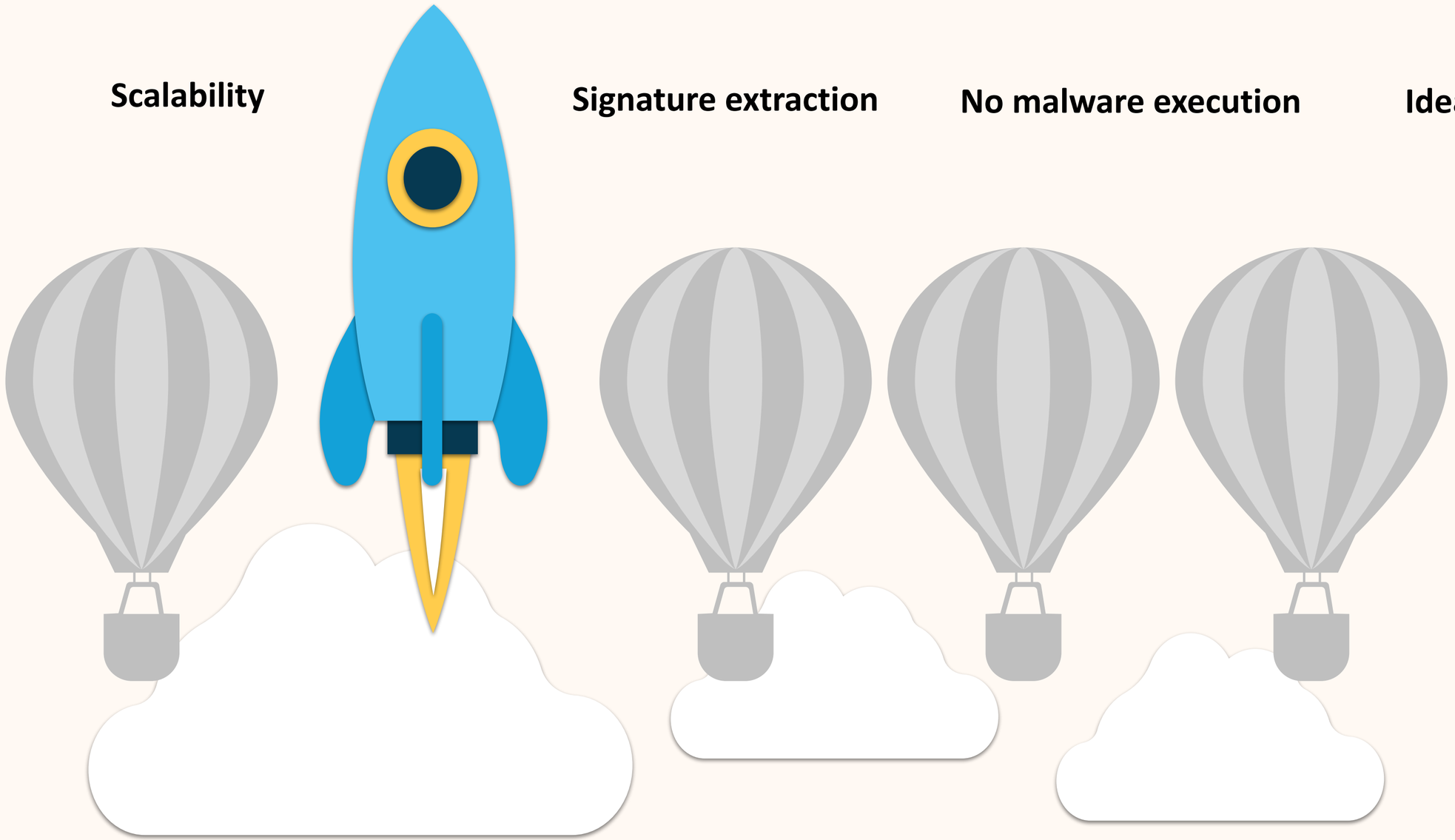
Static Feature Benefits

Scalability

Signature extraction

No malware execution

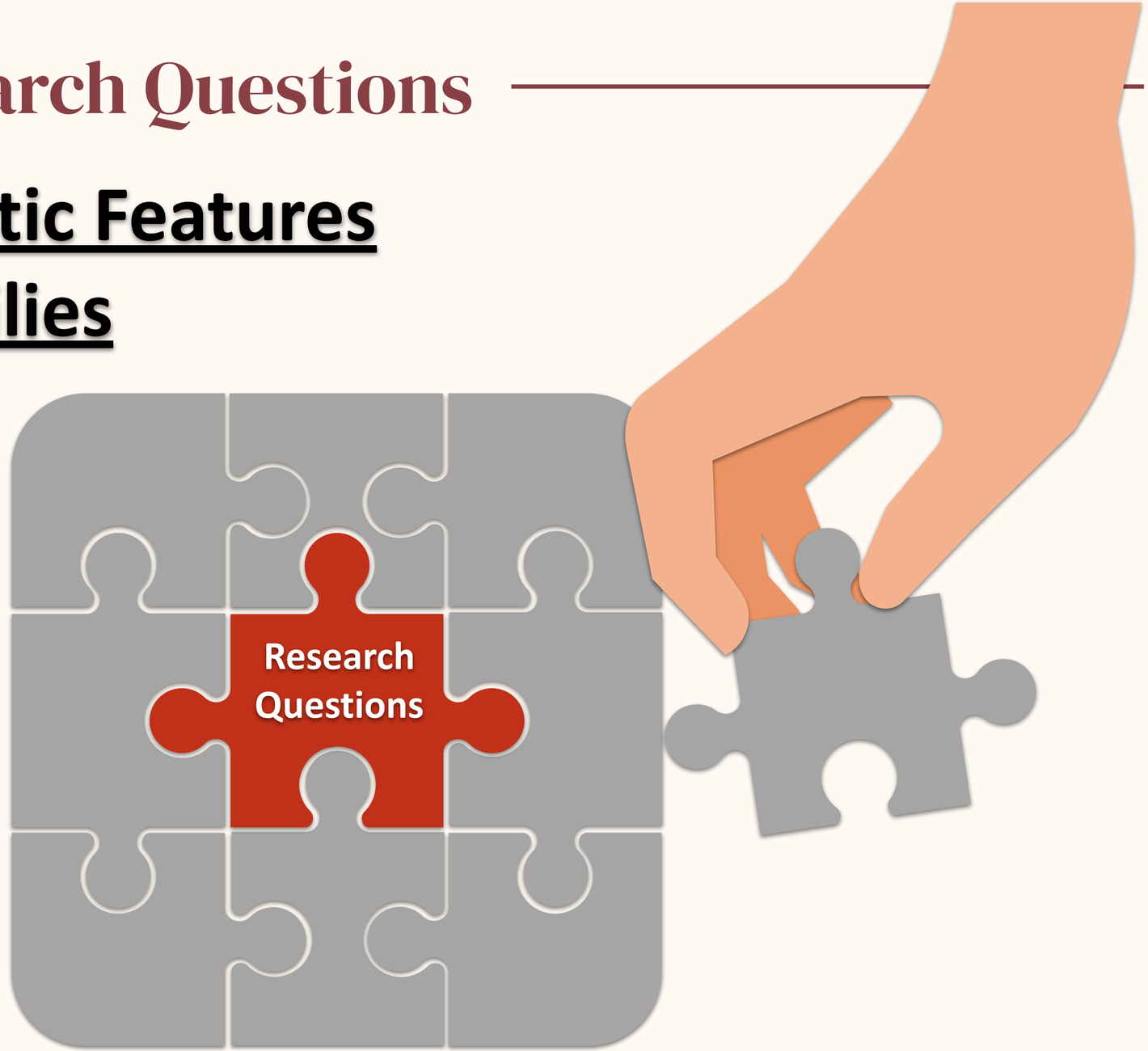
Ideal for large scale



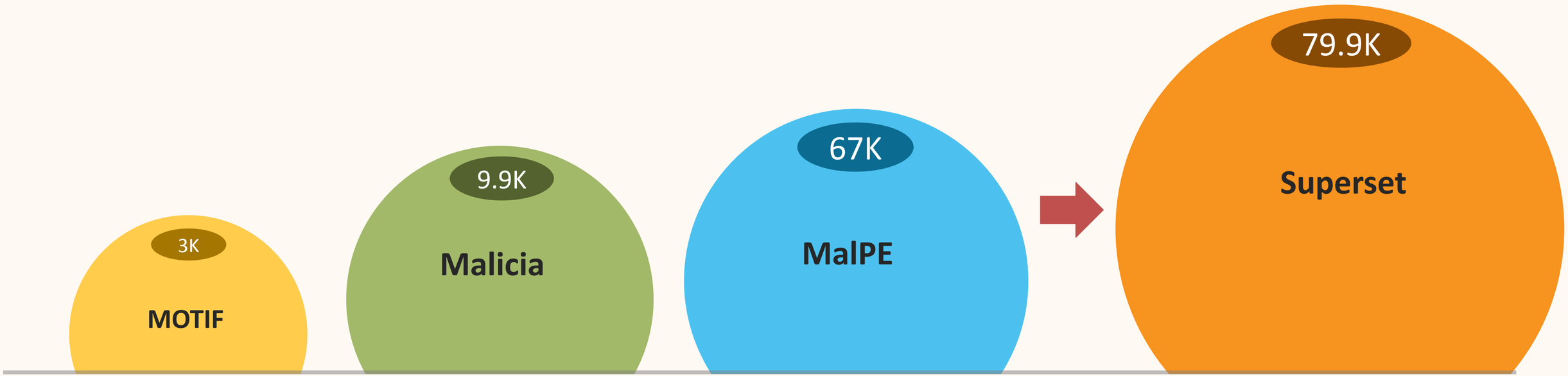
Research Questions

Precision Evaluation of Static Features in Windows Malware Families

- RQ1. Similarity Feature Precision
  - How often static features create mixed clusters?
- RQ2. Similarity Feature Limits
  - Reasons for mixed clusters?



Windows Malware Datasets



454 families

- 2016 - 2020

53 families

- 2012 - 2013

670 families

- 2012 - 2022

1,1K families

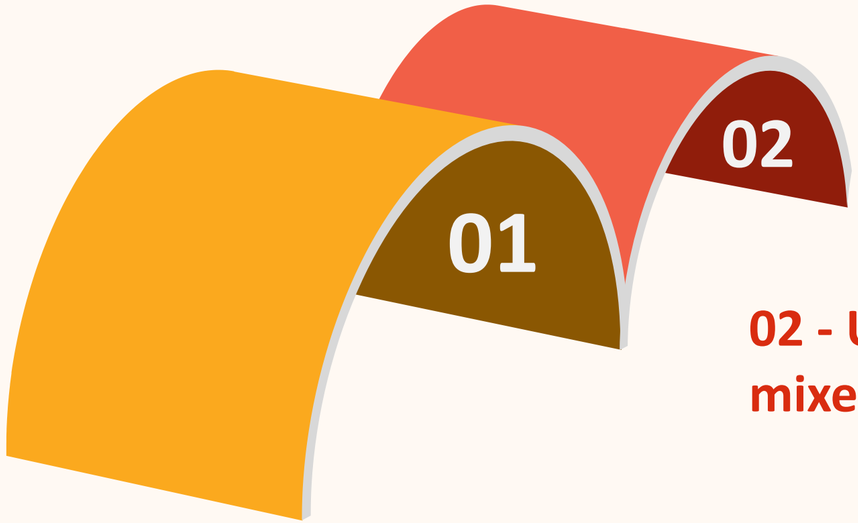
- 2012 - 2022

Approach Overview

01 - Clustering samples using one similarity feature

Approach Overview

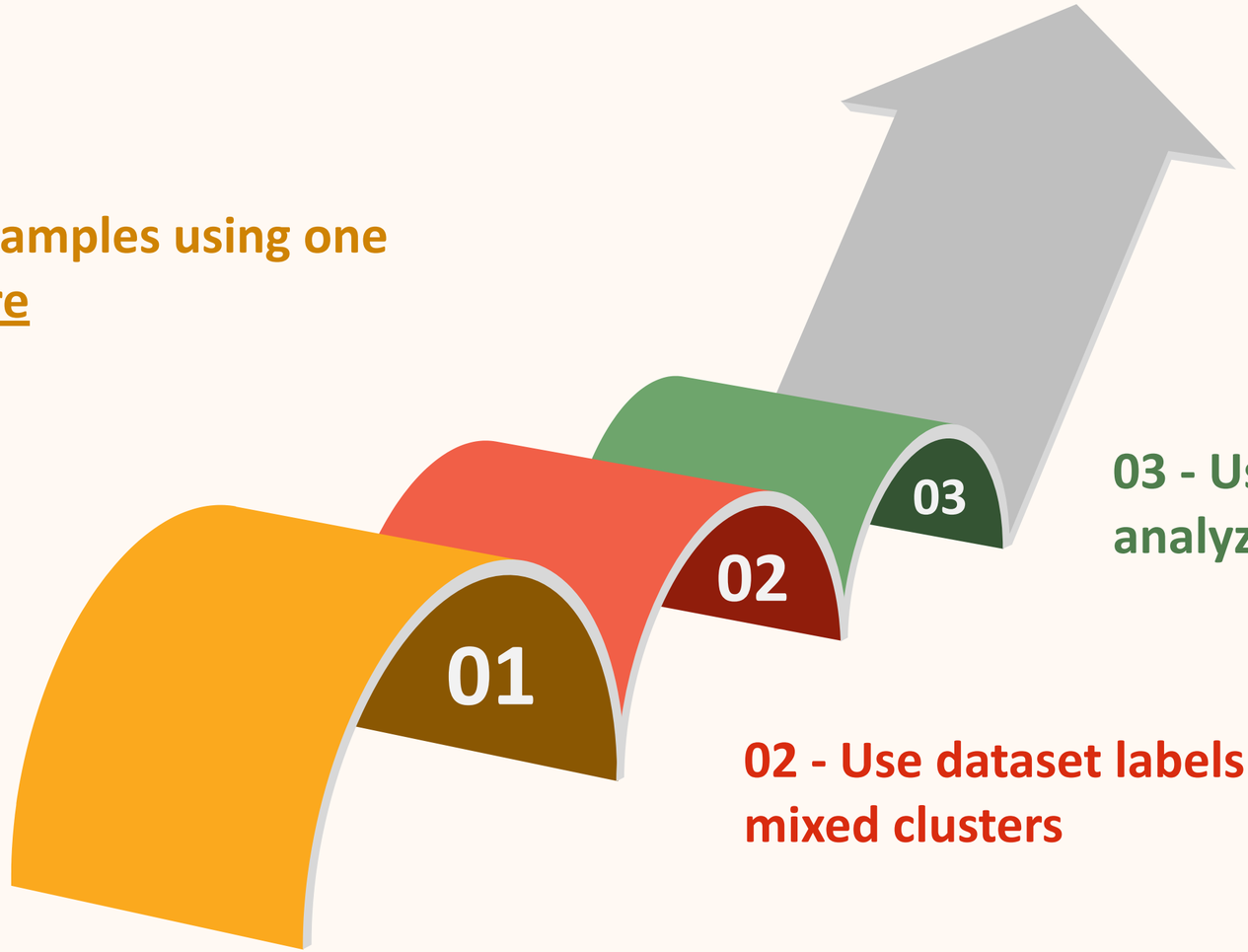
01 - Clustering samples using one similarity feature



02 - Use dataset labels to identify mixed clusters

Approach Overview

01 - Clustering samples using one similarity feature

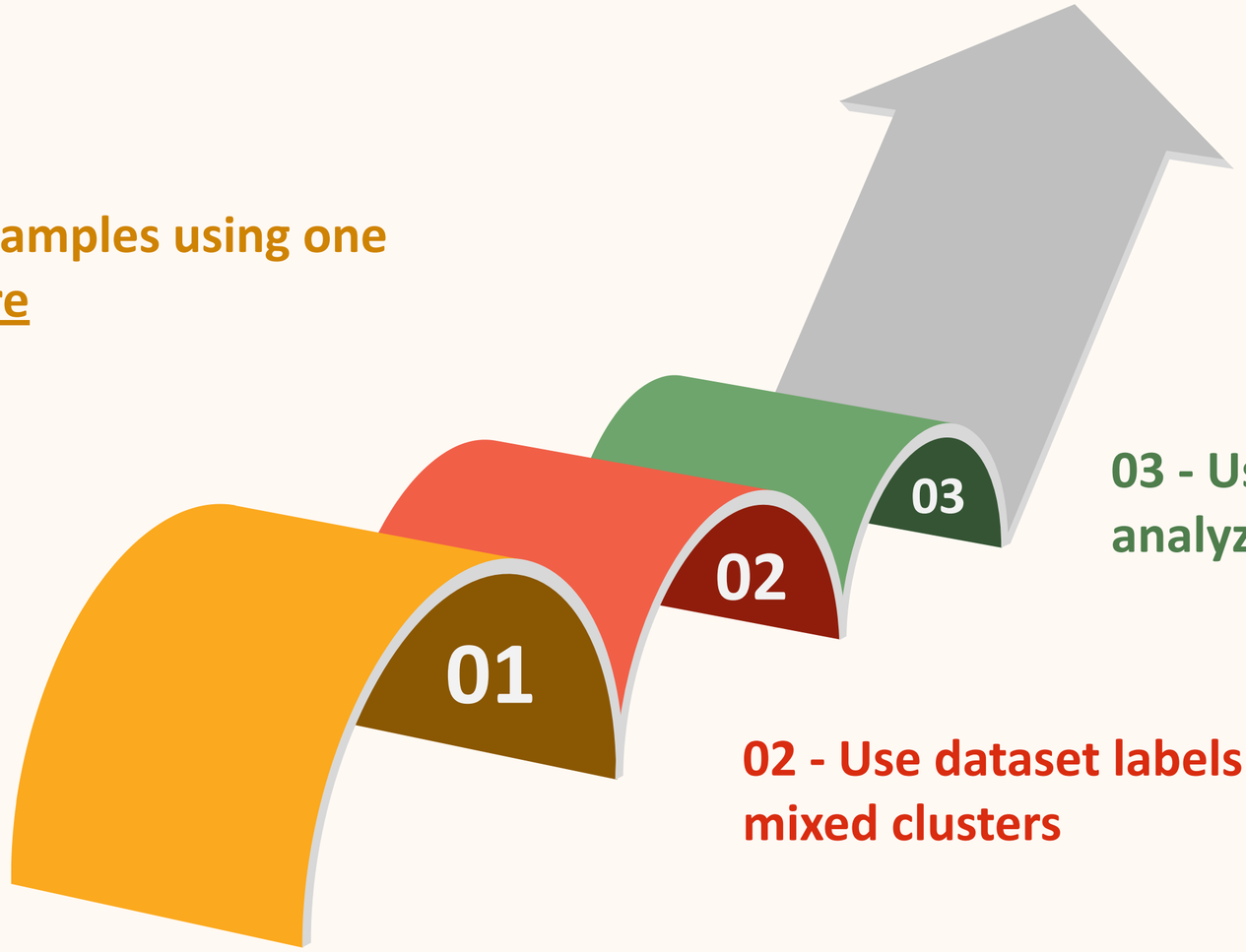


02 - Use dataset labels to identify mixed clusters

03 - Use analysis features to analyze mixed cluster

Approach Overview

01 - Clustering samples using one similarity feature



02 - Use dataset labels to identify mixed clusters

03 - Use analysis features to analyze mixed cluster

Our goal is not to develop a new family clustering approach, but to understand the limits of selected static features

Similarity Features

Structural hashes

Different input -> Different output

“hello” -> “5d4140…”
“hellp” -> “8e4d53…”

Equality comparison

- **Imphash**: Imported libraries
- **Pehash**: Program Executable header
- **Richpe:** Rich header (compiler metadata)
- **Certificate**: Thumbprint
- **Authentihash:** Exclude code signing fields
- **Icon hash:** Extracted icon
- **vhash:** VirusTotal’s propietary hash

Similarity Features

Structural hashes

Different input -> Different output

“hello” -> “5d4140…”
“hellp” -> “8e4d53…”

Equality comparison

- **Imphash**: Imported libraries
- **Pehash**: Program Executable header
- **Richpe:** Rich header (compiler metadata)
- **Certificate**: Thumbprint
- **Authentihash:** Exclude code signing fields
- **Icon hash:** Extracted icon
- **vhash:** VirusTotal’s propietary hash

Fuzzy hashes

Similar input -> Similar output

“hello” -> “abcd…”
“hellp” -> “abxd…”

Similarity comparison

Whole-file:

- **Ssdeep**
- **TLSH**

Extracted icon:

- **Dhash**

Similarity Features

Structural hashes

Different input -> Different output

“hello” -> “5d4140…”
“hellp” -> “8e4d53…”

Equality comparison

- **Imphash**: Imported libraries
- **Pehash**: Program Executable header
- **Richpe:** Rich header (compiler metadata)
- **Certificate**: Thumbprint
- **Authentihash:** Exclude code signing fields
- **Icon hash:** Extracted icon
- **vhash:** VirusTotal’s propietary hash

Fuzzy hashes

Similar input -> Similar output

“hello” -> “abcd…”
“hellp” -> “abxd…”

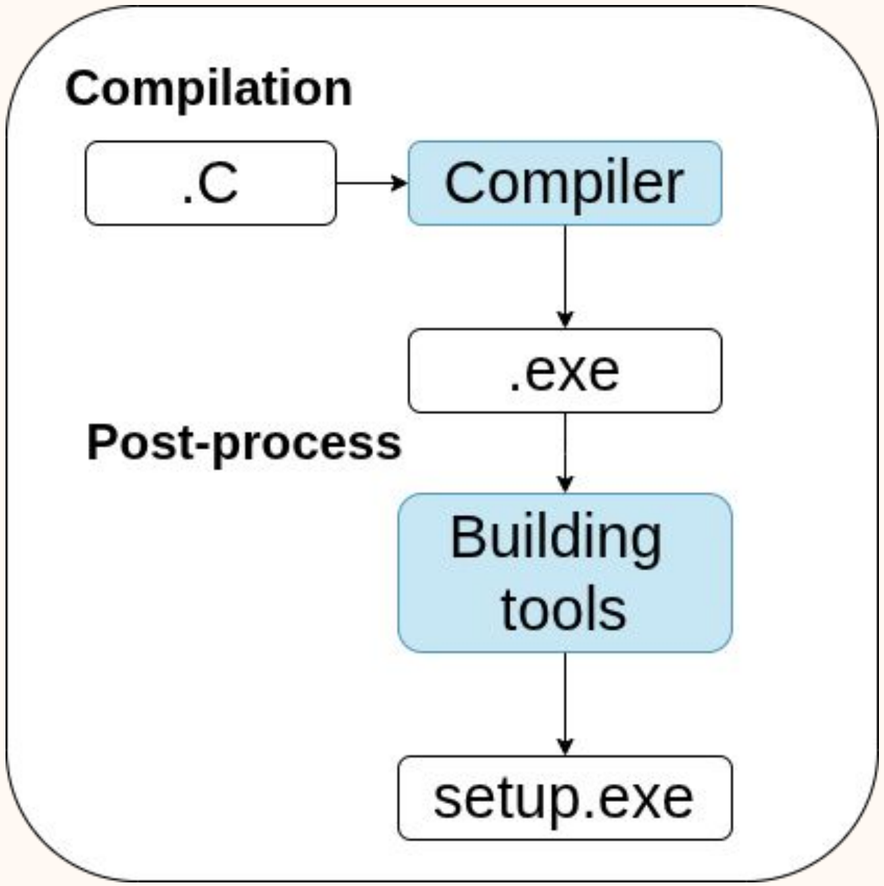
Similarity comparison

Whole-file:
- **Ssdeep**
- **TLSH**

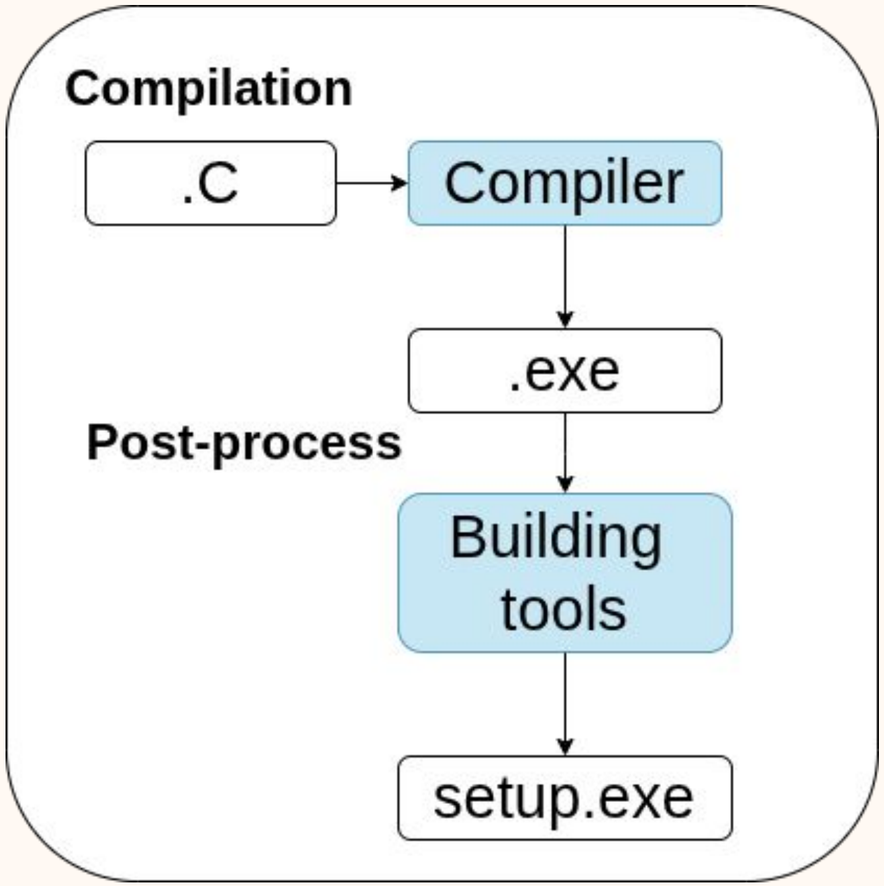
Extracted icon:
- **Dhash**

- **Proven effectiveness and interpretability**
- **Ease of extraction and reproducibility**

Analysis Features - EXE Building Tools



Analysis Features - EXE Building Tools

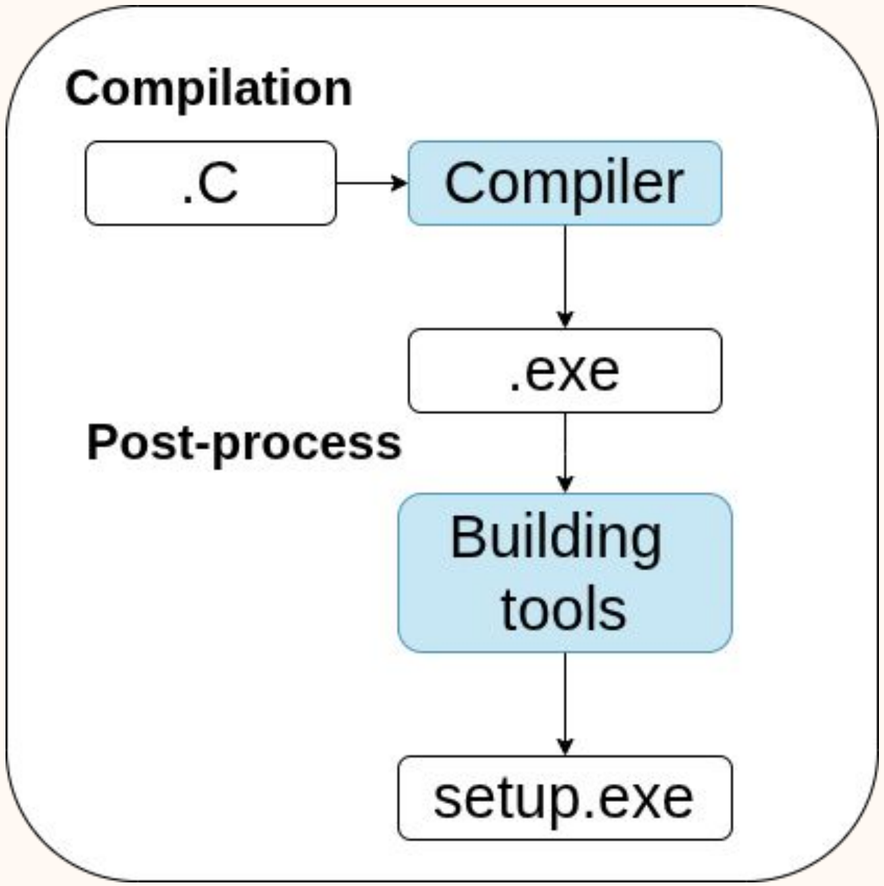


Packer

Code Shield

- Encrypt and decrypt at runtime
- Obfuscation oriented
- UPX, PECompact…

Analysis Features - EXE Building Tools



Packer

Code Shield

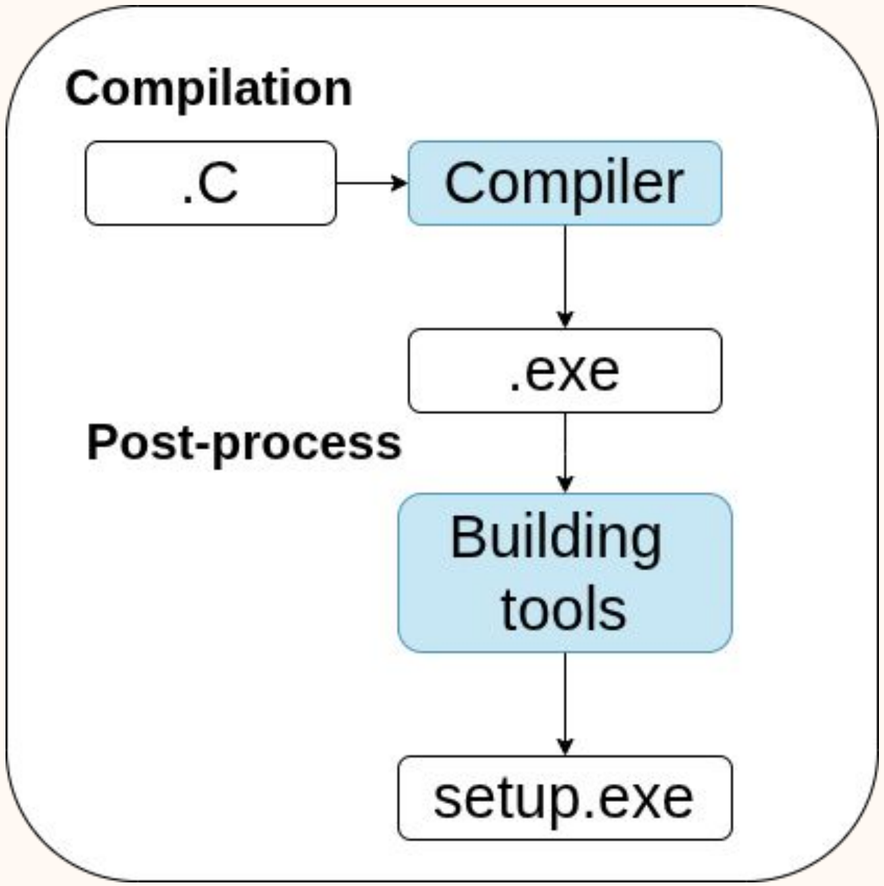
- Encrypt and decrypt at runtime
- Obfuscation oriented
- UPX, PECompact…

Installers

Setup Package

- Installs software
- InnoSetup, NSIS…

Analysis Features - EXE Building Tools



Packer

Code Shield

- Encrypt and decrypt at runtime
- Obfuscation oriented
- UPX, PECompact…

Installers

Setup Package

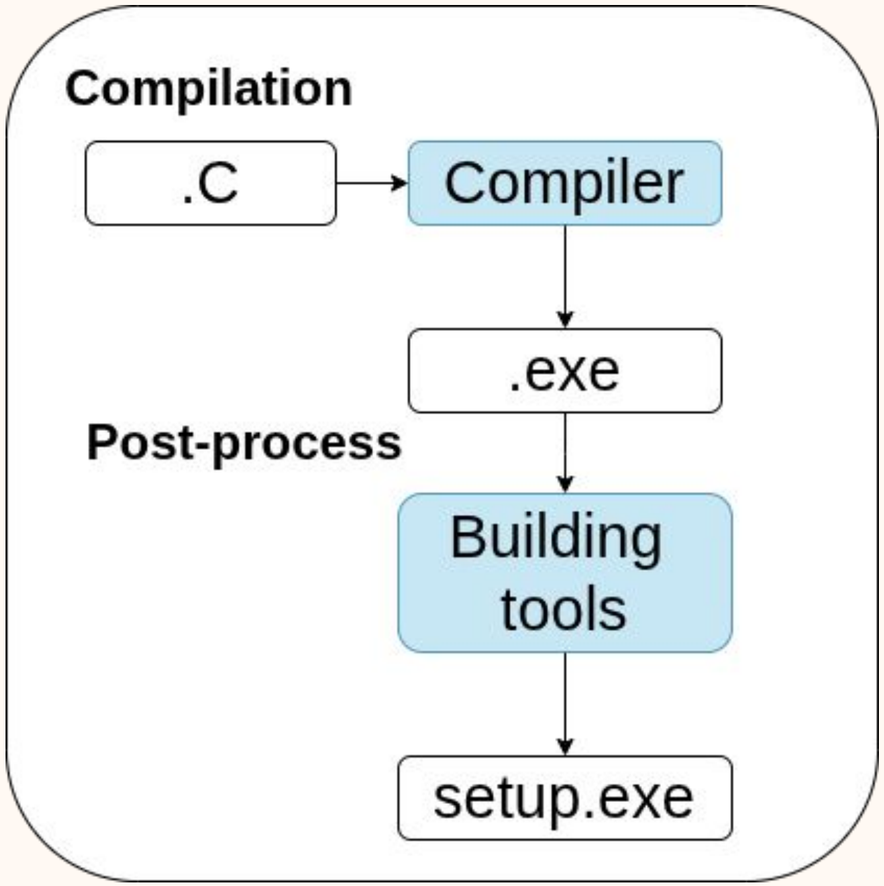
- Installs software
- InnoSetup, NSIS…

Self-Extracting Archive

Auto Extract

- Self-extract compression
- 7-zip, WinRAR…

Analysis Features - EXE Building Tools



Packer

Code Shield

- Encrypt and decrypt at runtime
- Obfuscation oriented
- UPX, PECompact…

Installers

Setup Package

- Installs software
- InnoSetup, NSIS…

Self-Extracting Archive

Auto Extract

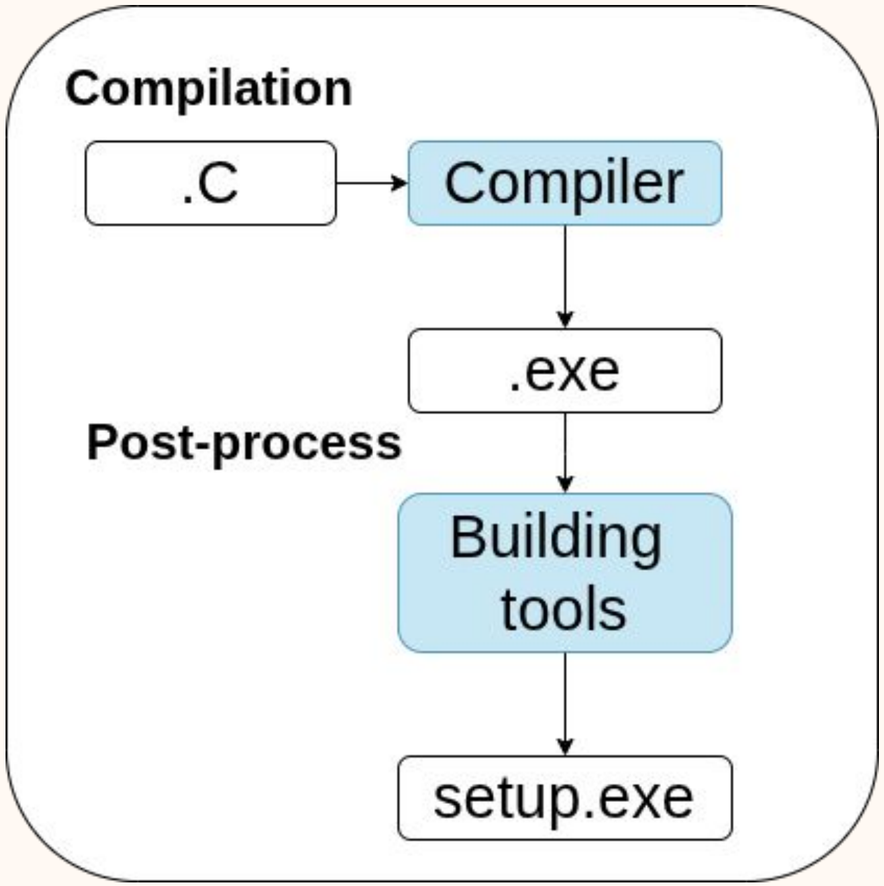
- Self-extract compression
- 7-zip, WinRAR…

Script-Generated Executable

Script Wrapper

- Embed interpreter
- aut2exe, bat2exe

Analysis Features - EXE Building Tools



Packer

Code Shield

- Encrypt and decrypt at runtime
- Obfuscation oriented
- UPX, PECompact…

Installers

Setup Package

- Installs software
- InnoSetup, NSIS…

Self-Extracting Archive

Auto Extract

- Self-extract compression
- 7-zip, WinRAR…

Script-Generated Executable

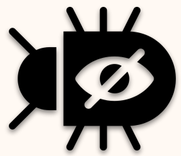
Script Wrapper

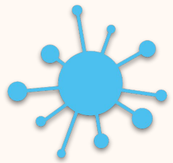
- Embed interpreter
- aut2exe, bat2exe

Identification with Detect-it-Easy and PackGenome

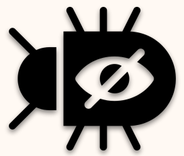
33% of samples

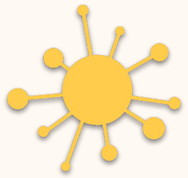
Analysis Approach

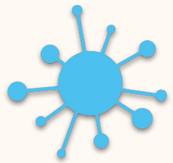




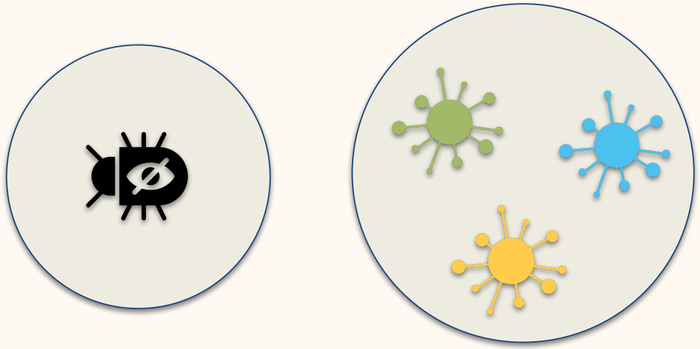
Analysis Approach





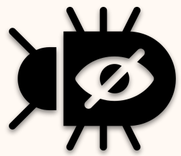


Ground Truth



Analysis Approach

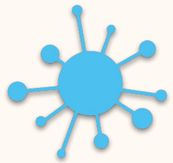
Feature Extraction



a1b2c3

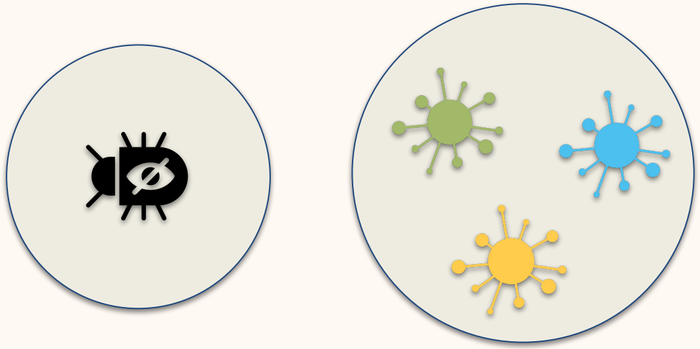
a1b2c3

ffeedd



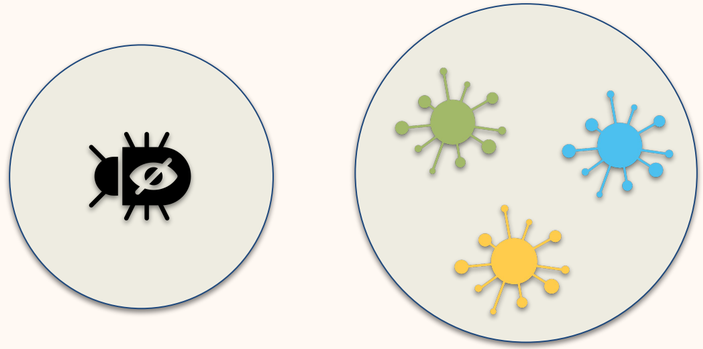
ffeedd

Ground Truth

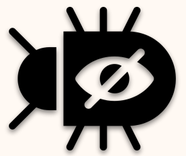


Analysis Approach

Ground Truth



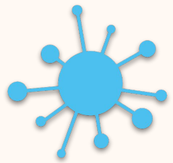
Feature Extraction



a1b2c3

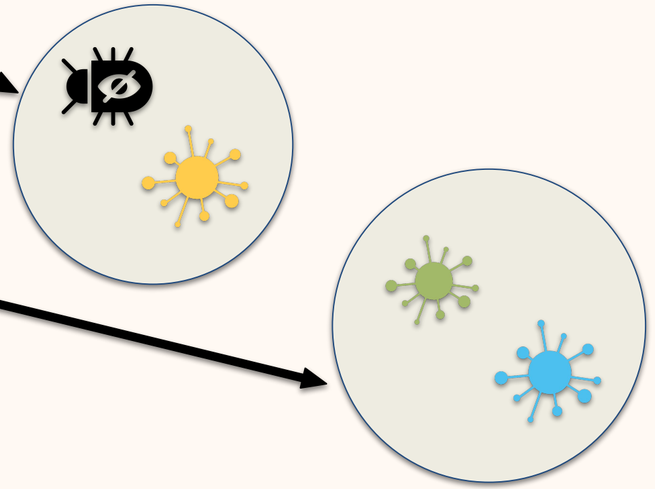
a1b2c3

ffeedd

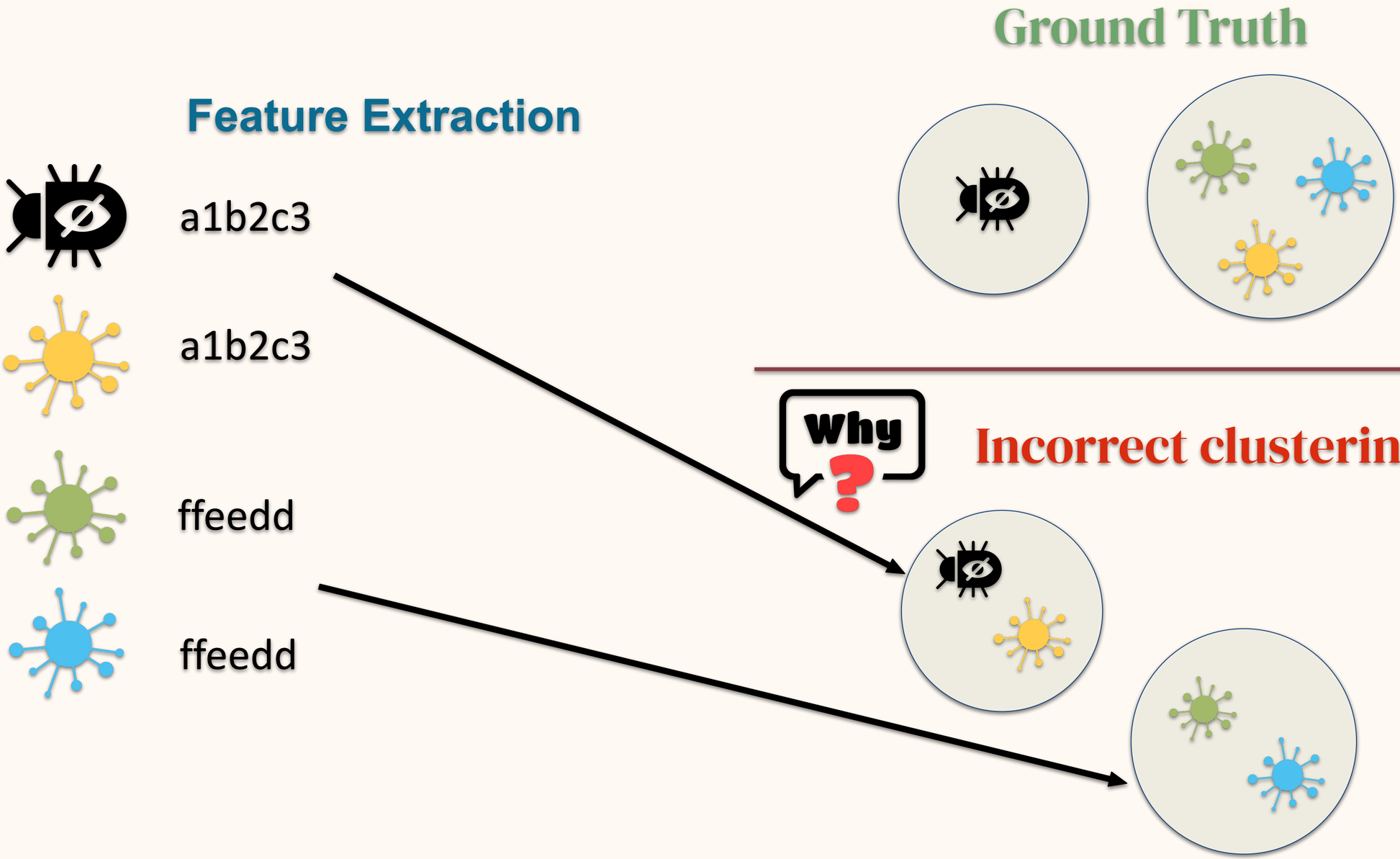


ffeedd

Incorrect clustering



Analysis Approach

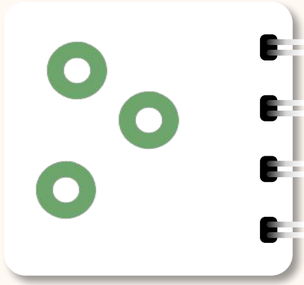


RQ1: Similarity Feature Precision

RQ1: Similarity Feature Precision

99%

Authentihash
Certificate Thumbprint
Certificate Subject



- 99% singletons
- 93% precision no singletons

RQ1: Similarity Feature Precision

- 71-85% singletons
- 94% precision no singletons

Pehash
Ssdeep
Tlsh
Vhash

94-97%

99%

Authentihash
Certificate Thumbprint
Certificate Subject

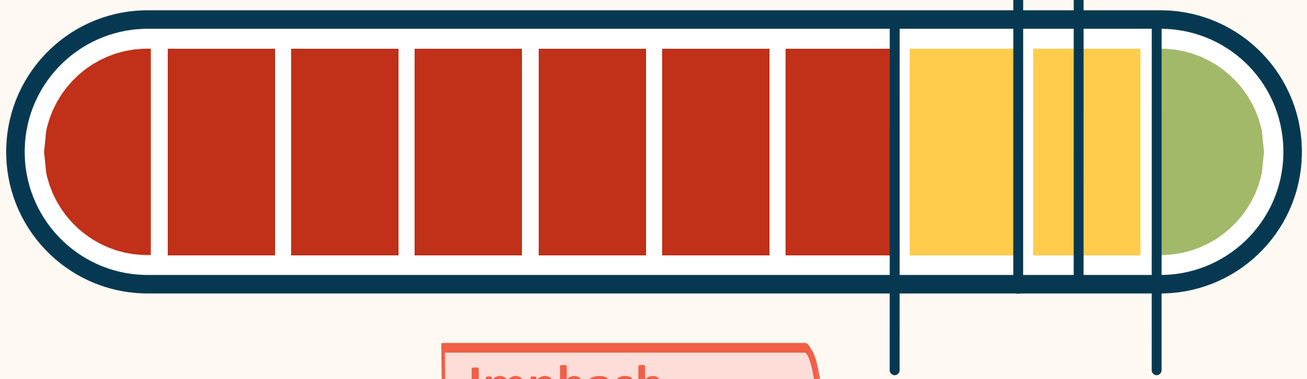
- 99% singletons
- 93% precision no singletons

RQ1: Similarity Feature Precision

- 71-85% singletons
- 94% precision no singletons

Pehash
Ssdeep
Tlsh
Vhash

94-97%



90%

99%

Authentihash
Certificate Thumbprint
Certificate Subject

Imphash
Richpe
Icon hash
Icon dhash

- Precision 85-88%

- 99% singletons
- 93% precision no singletons

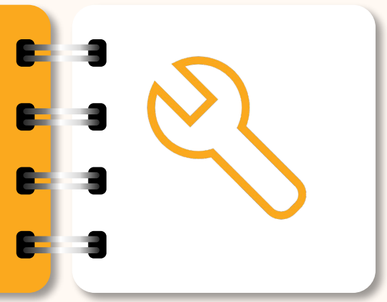
RQ2: Limits of Static Features

Whole-file fuzzy hashes

- Affected by same EXE building tools
- Similarity not from malicious code

Remove building tools

- 2.5% improvement



SSDEEP Top 10 Largest Clusters

# - Reason	Samples	Mixed Clusters	Families
1 - installer:inno	310	✓	5
2 - GT errors	185	✓	4
3 - script:aut2exe	173	✓	19
4 - archive:sfx	167	✓	2
5 -	162	✓	1
6 - No overlay sha256 file	157	✓	2
7 - packer:dxpack	157	✓	2
8 - packer:upx	149	✓	8
9 - GT errors	132	✓	3
10 - archive:sfx	125	✓	3

SSDEEP Top 10 Largest Clusters

# - Reason	Samples	Mixed Clusters	Families
1 - installer:inno	310	✓	5
2 - GT errors	185	✓	4
3 - script:aut2exe	173	✓	19
4 - archive:sfx	167	✓	2
5 -	162	✓	1
6 - No overlay sha256 file	157	✓	2
7 - packer:dxpack	157	✓	2
8 - packer:upx	149	✓	8
9 - GT errors	132	✓	3
10 - archive:sfx	125	✓	3

Large clusters

- All clusters > 100 samples
- Largest cluster > 300 samples

SSDEEP Top 10 Largest Clusters

# - Reason	Samples	Mixed Clusters	Families
1 - installer:inno	310	✓	5
2 - GT errors	185	✓	4
3 - script:aut2exe	173	✓	19
4 - archive:sfx	167	✓	2
5 -	162	✓	1
6 - No overlay sha256 file	157	✓	2
7 - packer:dxpack	157	✓	2
8 - packer:upx	149	✓	8
9 - GT errors	132	✓	3
10 - archive:sfx	125	✓	3

Large clusters

- All clusters > 100 samples
- Largest cluster > 300 samples

9/10 mixed clusters

- 6/9 due to EXE building tools
  - 2 SFXs and 2 packers
  - 1 installer and 1 script
- 2/9 due to dataset errors

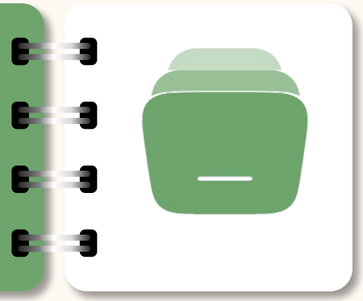
RQ2: Limits of Static Features

Imphash

- Strongly affected by same EXE building tools
- Different packers can produce same import table

Import DLLs filtering

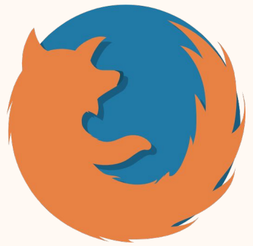
- 10% improvement
- +3% after excluding samples with <10 imports



RQ2: Limits of Static Features

Certificate

- Often due to dataset errors (aliases)
- Few cases of “stolen” benign certificates



Remove invalid certificates

- 2-4% improvement

RQ2: Similarity Feature Limits

Icons

- Collisions due to common icons
- Not specific to family

Ignore generic icons

- Challenge by itself

Conclusions

Feature effectiveness

Similarity features grouped into 3 groups

Impact of EXE-Building Tools

Packers, installers, SFX archives, script-building tools

Feature pre-processing

Improvements to the similarity features

RQ1: Similarity Feature Precision

- 71-85% singletons
- 94% precision no singletons

Pehash
Ssdeep
Tlsh
Vhash

94-97%

Authentihash
Certificate Thumbprint
Certificate Subject

90%

99%

Imphash
Richpe
Icon hash
Icon dhash

- Precision 85-88%

- 99% singletons
- 93% precision no singletons

RQ2: Similarity Feature Limits

Fuzzy hashes

- Affected by same EXE building tools

Certificate

- Often due to dataset errors
- Few cases of “stolen” benign certificates

Imphash

- Affected by same EXE building tools
- Different packers can produce same import table

Richpe

- Affected by different/same EXE building tools
- Collisions without EXE building tools

Icons

- Collisions due to common icons

Conclusions

Feature effectiveness

Similarity features grouped into 3 groups

Impact of EXE-Building Tools

Packers, installers, SFX archives, script-building tools

Feature pre-processing

Improvements to the similarity features

Contact

kevin.liebergen@imdea.org

github.com/kevinLiebergen

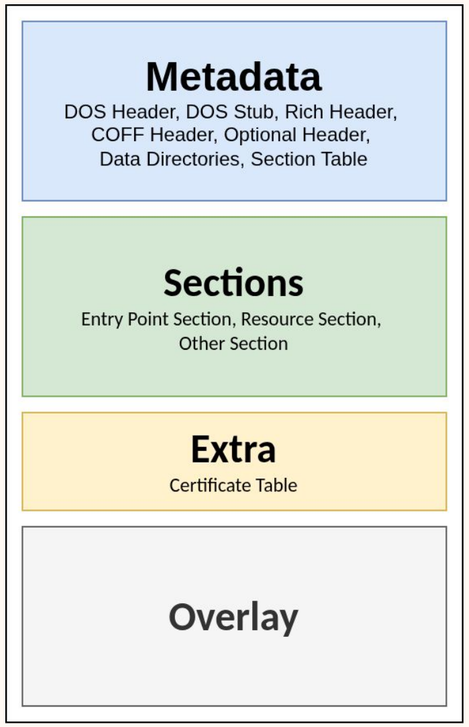
linkedin.com/in/kevin-van-liebergen-avila

Analysis Features - Others

Overlay

Data appended to end

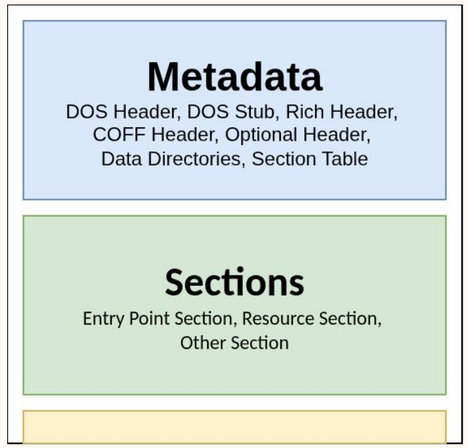
- Data not loaded, but can be accessed
- 36% of samples



Truncation

Corrupted file

- Expected size larger than real file
- Does not run
- 3% of samples



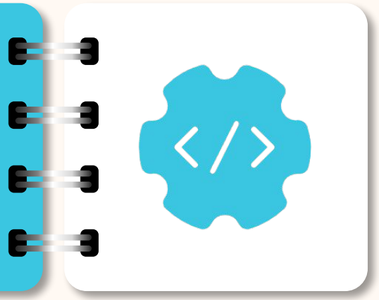
RQ2: Limits of Static Features

Richpe

- Affected by different/same building tools
- Collisions also without building tools

Collisions due to same compiler

- Small number of entries



Backup Slide 1 - Certificates and Precision

What is an invalid certificate?

Certificate with an invalid trust chain (e.g., misconfigured or benign entities)

Q1

Why Authentihash and Thumbprint reach 99% precision?

Authentihash: almost identical binaries (ignores signing fields & checksum).
Certificate Thumbprint: binaries signed by the same certificate.
High precision comes from strict, byte-level or signer equality.

Q2

Are the results computed over singletons?

Yes. Precision includes singletons but also recalculated without them.

Q3

Backup Slide 2 - Overlap ad Feature Differnces

Dataset overlap

Only 10 samples overlap between datasets.

Q4

Authentihash vs. Certificate Thumbprint

Authentihash: Measure Content similarity, only group identical code.
Certificate Thumbprint: Measure signer identity, may group unrelated binaries.

Q5

Why do Imphash and RichPE perform worse?

Imphash: affected by packers/installers creating identical import tables.
RichPE: collisions from same compilers or build environments.

Q6

Backup Slide 3 -EXE Building Tools Detection

What % of samples use EXE-building tools?

≈ 33% (20.6% packed, 6.2% SFX, 4.2% installers, 2.7% scripts).

Q7

Are script wrappers or SFX archives detected automatically?

Yes, via DiE/PackGenome signatures (no manual labeling).

Q8

How were these detected?

Detect-It-Easy (DiE): signature-based.
PackGenome: YARA rules for 20 packers.
PEiD excluded (outdated).

Q9

Backup Slide 4 - Applications

How can results apply to threat intelligence?

Q12

Helps distinguish true family links vs. artificial similarities.
Avoids misinterpreting packer-based clusters.