SECLAB

When Malware is Packin’ Heat;

Limits of Machine Learning Classifiers Based on Static Analysis Features

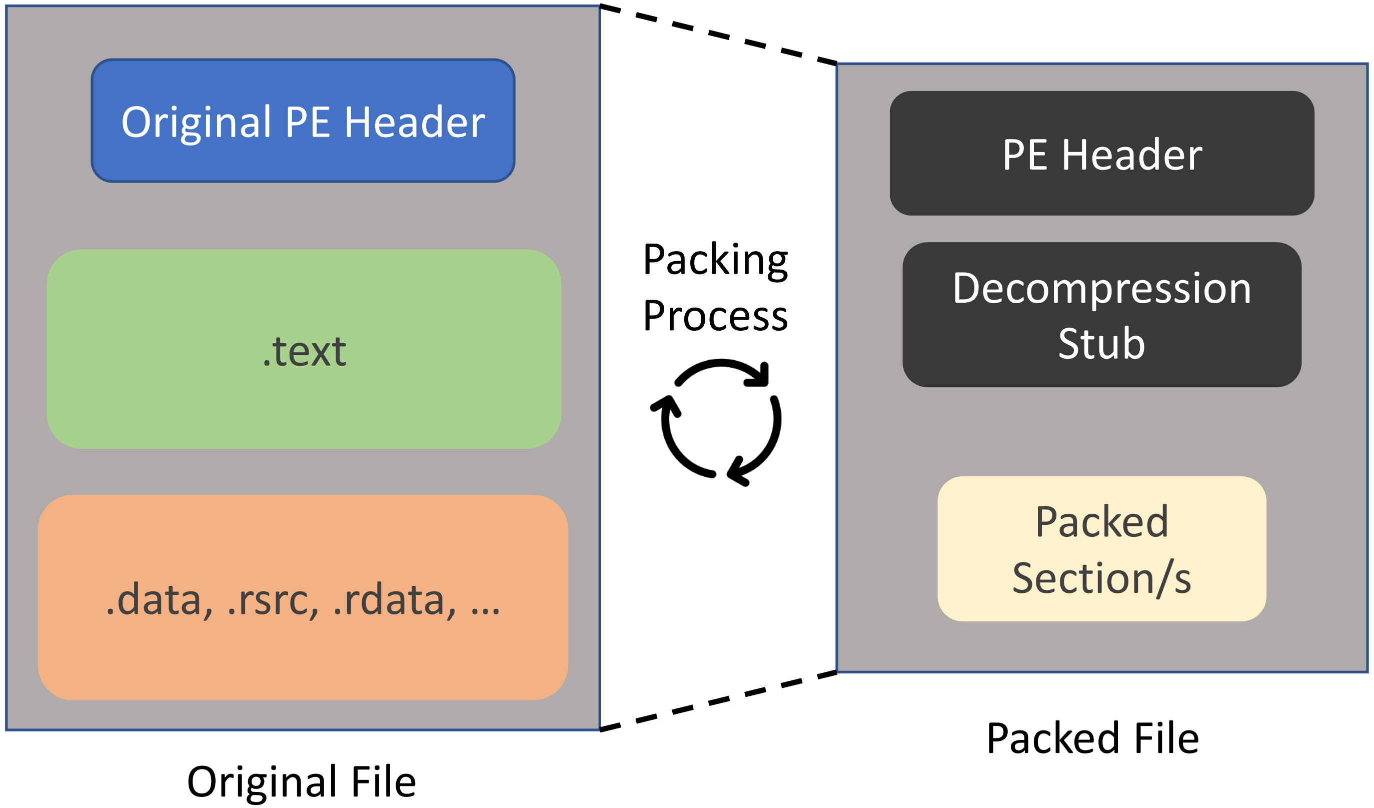
Hojjat Aghakhani, *Fabio Gritti, Francesco Mecca, Martina Lindorfer,*
Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, Christopher Kruegel

UNIVERSITÀ
DEGLI STUDI
DI TORINO

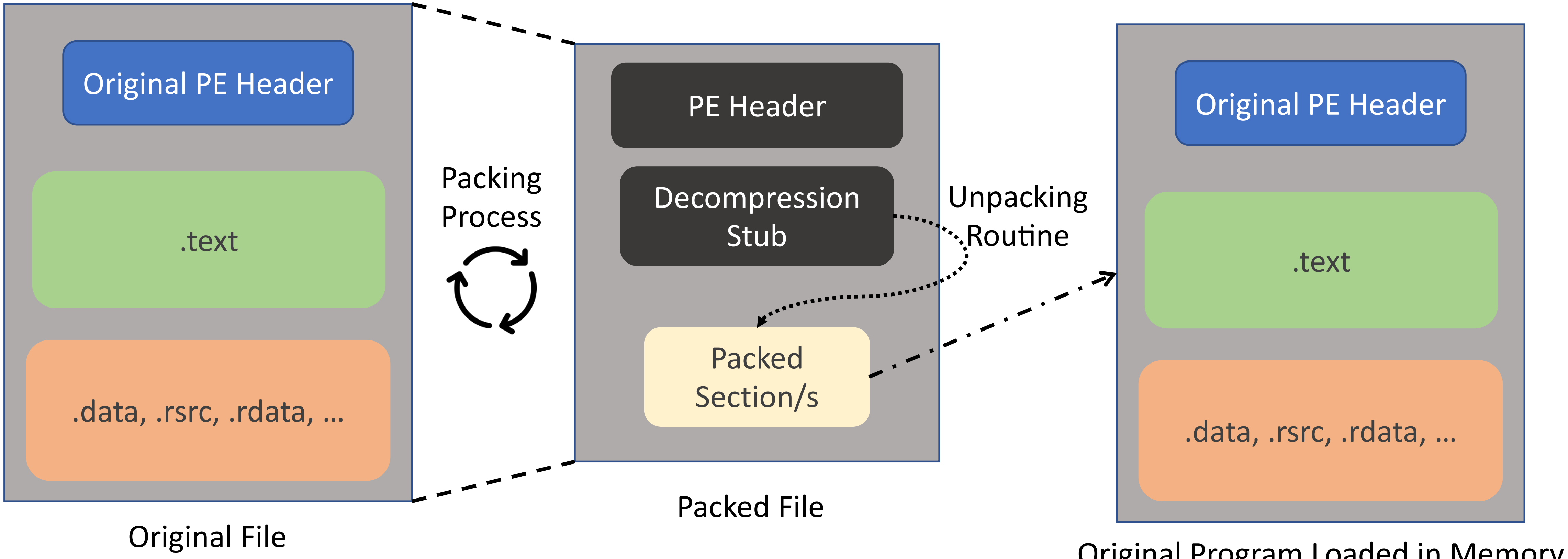
TU WIEN
TECHNISCHE
UNIVERSITÄT
WIEN

EURECOM
Sophia Antipolis

Packing


Original PE Header
.text
.data, .rsrc, .rdata, …
Original File
Packing Process
PE Header
Decompression Stub
Packed Section/s
Packed File

Packing


Original PE Header
.text
.data, .rsrc, .rdata, …
Original File
Packing Process
PE Header
Decompression Stub
Packed Section/s
Packed File
Unpacking Routine
Original PE Header
.text
.data, .rsrc, .rdata, …
Original Program Loaded in Memory

Packing Employed By Malware Authors

Packing Evolution

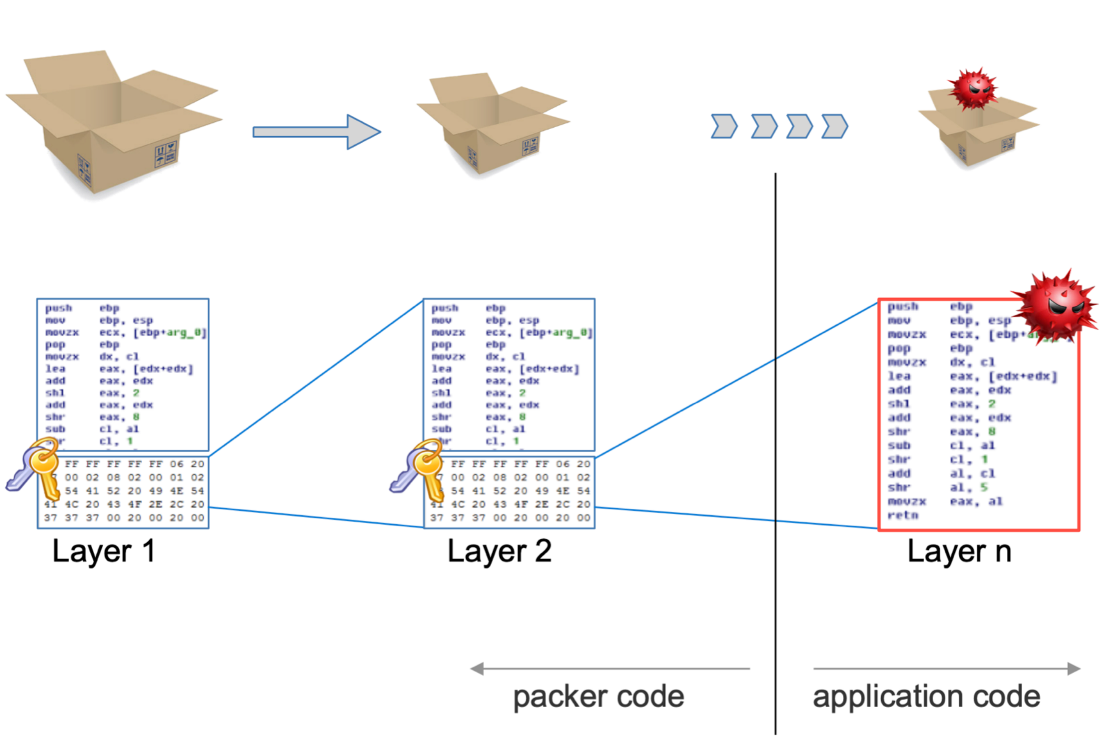
- Most packers are not this simple anymore...

Packing Evolution

- Most packers are not this simple anymore...
  - Different methods of obfuscation or encryption are being used

Packing Evolution

- Most packers are not this simple anymore...
  - Different methods of obfuscation or
  - Packing happens at multiple layers



Layer 1

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
   FF FF FF FF FF 06 20
   00 02 08 02 00 01 02
   54 41 52 20 49 4E 54
41 4C 20 43 4F 2E 2C 20
37 37 37 00 20 00 20 00
```

Layer 2

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

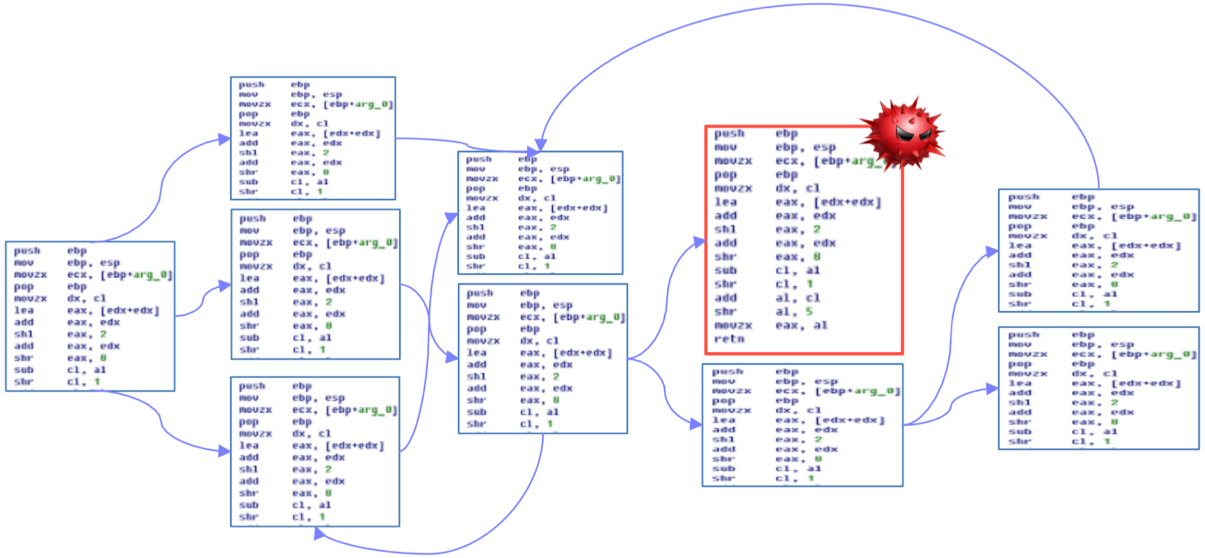
```
   FF FF FF FF FF 06 20
   00 02 08 02 00 01 02
   54 41 52 20 49 4E 54
41 4C 20 43 4F 2E 2C 20
37 37 37 00 20 00 20 00
```

Layer n

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
add     al, cl
shr     al, 5
movzx   eax, al
retn
```

Packing Evolution

- Most packers are not this simple anymore...
  - Different methods of obfuscation or encryption are being used
  - Packing happens at multiple layers
  - Unpacking routines are not necessarily executed in a straight line



```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
add     al, cl
shr     al, 5
movzx   eax, al
retn
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
```

Packing Evolution

- Most packers are not this simple anymore...
  - Different methods of obfuscation or encryption are being used
  - Packing happens at multiple layers
  - Unpacking routines are not necessarily executed in a straight line
  - Only a single fragment of the original code at any given time

Packing Evolution

- Most packers are not this simple anymore...
  - Different methods of obfuscation or encryption are being used
  - Packing happens at multiple layers
  - Unpacking routines are not necessarily executed in a straight line
  - Only a single fragment of the original code at any given time
  - Usually anti-debugging or anti-reverse-engineering techniques are employed

Why Does Packing Matter?

- It hampers the analysis of the code

Why Does Packing Matter?

- It hampers the analysis of the code
- Makes malware classification more challenging!

Why Does Packing Matter?

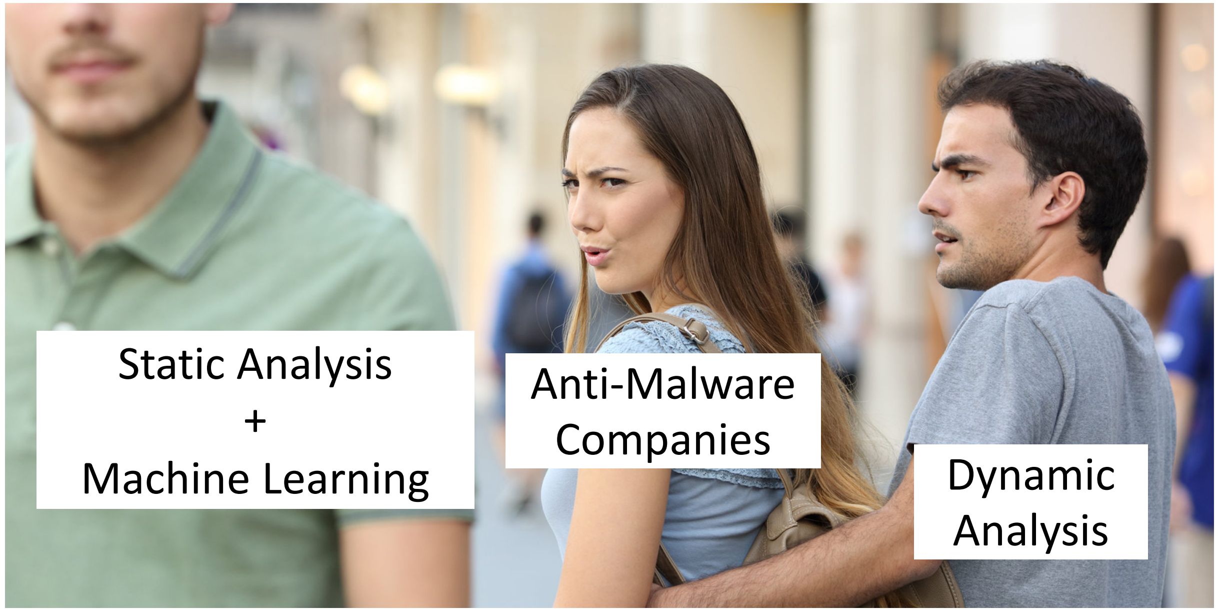
- It hampers the analysis of the code
- Makes malware classification more challenging!
  - Especially, when using only static analysis

Malware Classification Using Static Analysis

ENDGAME.

CYLANCE

cybereason



SentinelOne

Acronis

TRAPMINE

Malware Classification Using Static Analysis

ENDGAME.

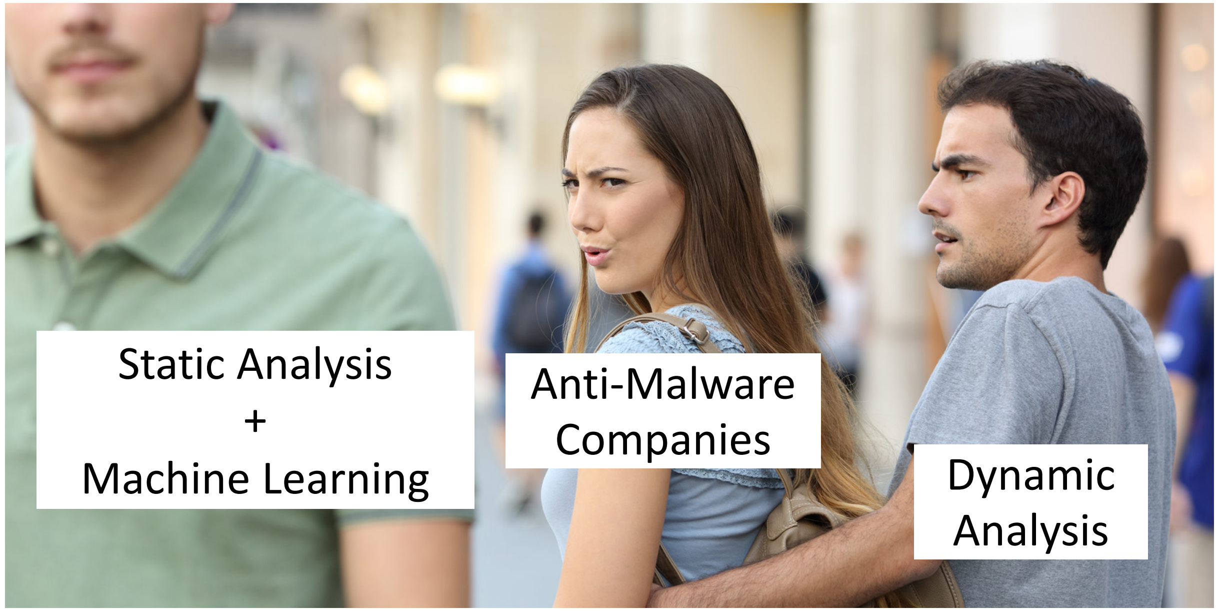
CYLANCE

cybereason

SentinelOne

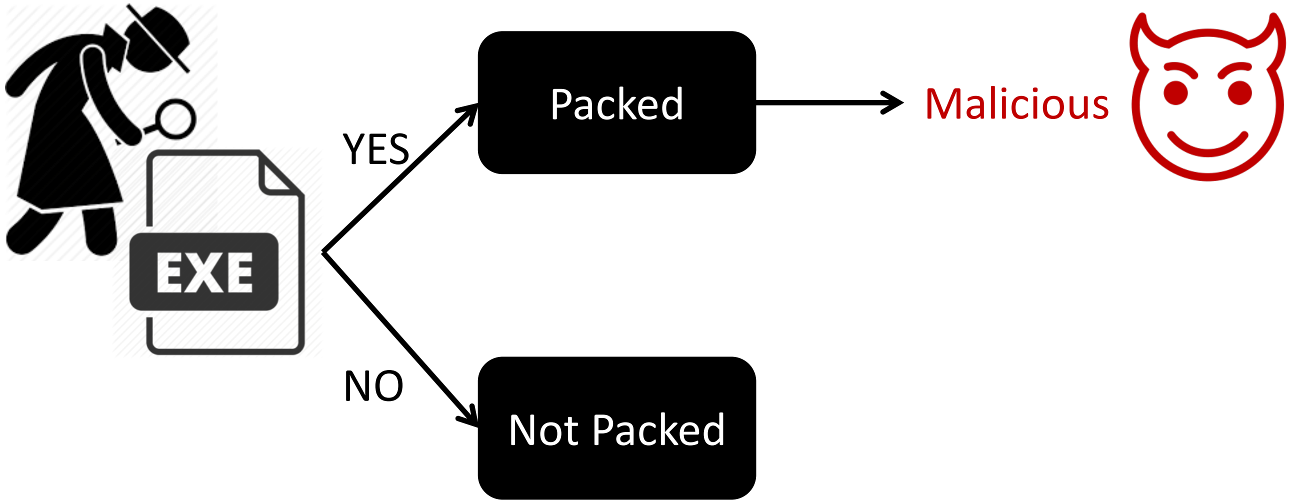
Acronis

TRAPMINE

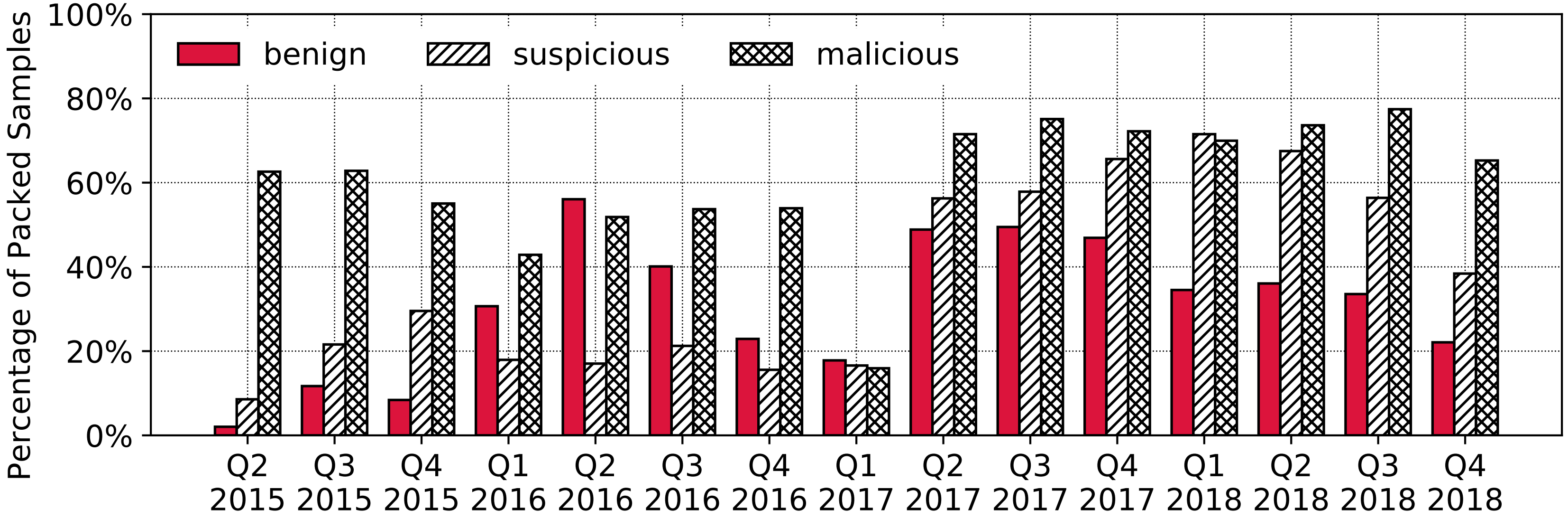


- What happens if the program is packed, i.e., the features are obfuscated?

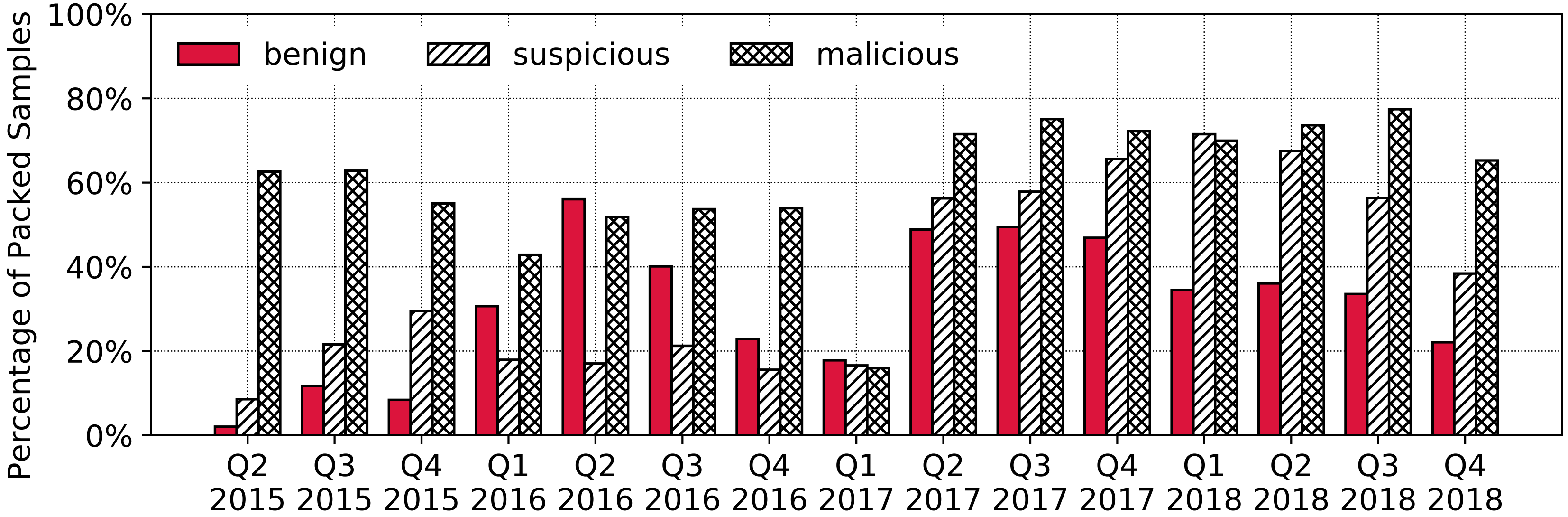
Do Benign Software Programs Use Packing?



Packing Is Common in Benign Programs



Packing Is Common in Benign Programs

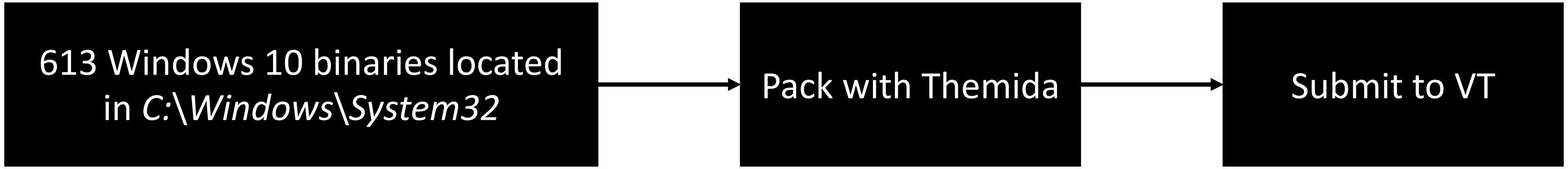


- Rahbarinia et al. [84], who studied 3 million web-based software downloads over 7 months in 2014, found that *both* malicious and benign files use known packers (58% and 54%, respectively)

B. Rahbarinia, M. Balduzzi, and R. Perdisci, “Exploring the Long Tail of (Malicious) Software Downloads,” in *Proc. of the International Conference on Dependable Systems and Networks (DSN)*, 2017.

“Packing == Malicious” → FALSE POSITIVE

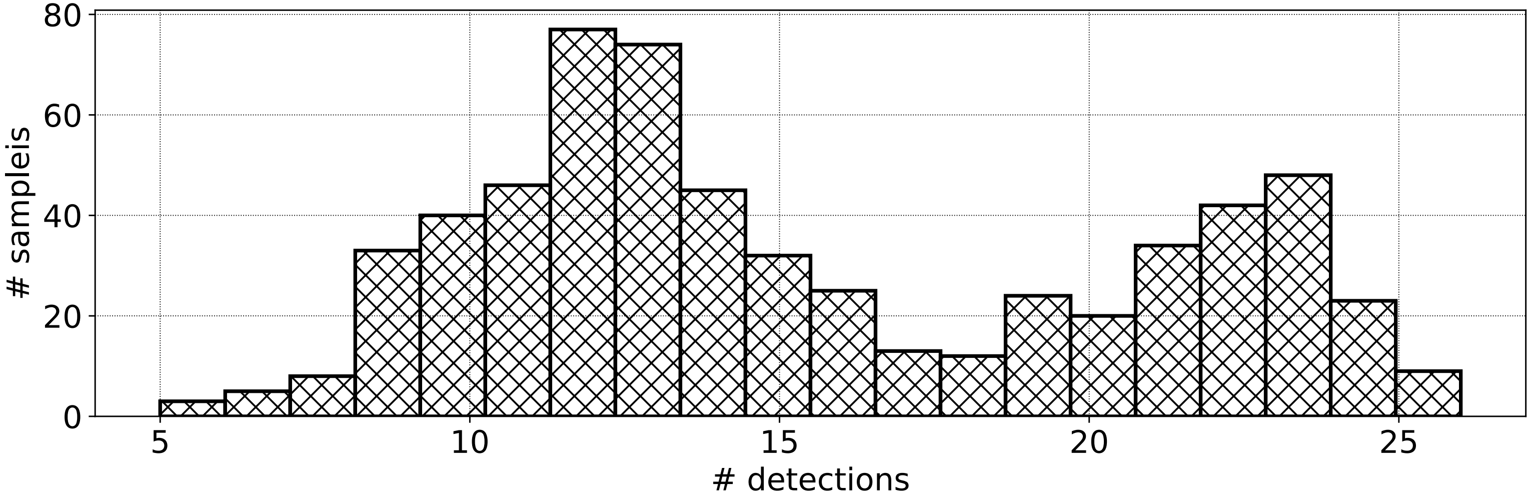
“Packing == Malicious” on VirusTotal?



613 Windows 10 binaries located in *C:\Windows\System32*

Pack with Themida

Submit to VT



Dataset Pollution

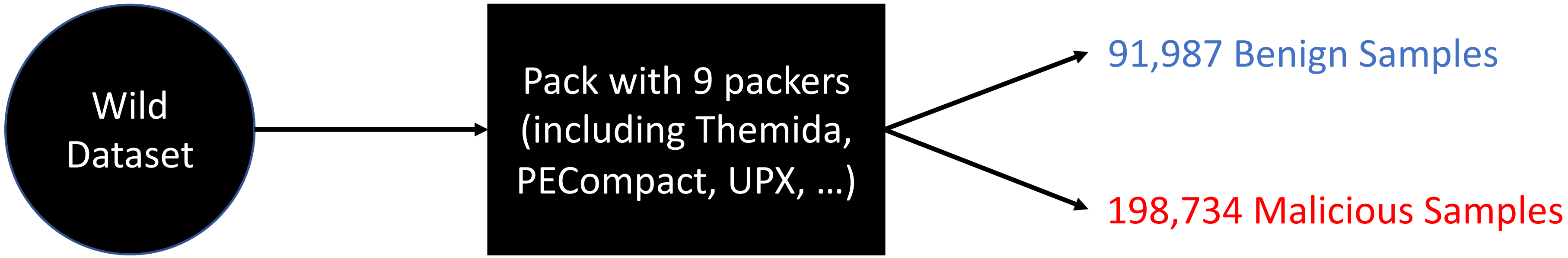
Does static analysis on packed binaries provide *rich enough* features to a malware classifier?

Datasets

1. Wild Dataset (50,724 executables):
   - 4,396 unpacked benign
   - 12,647 packed benign
   - 33,681 packed malicious

Datasets

1. Wild Dataset (50,724 executables):
   - 4,396 unpacked benign
   - 12,647 packed benign
   - 33,681 packed malicious
2. Lab Dataset:


Wild Dataset
Pack with 9 packers (including Themida, PECompact, UPX, …)
91,987 Benign Samples
198,734 Malicious Samples

Nine Feature Categories

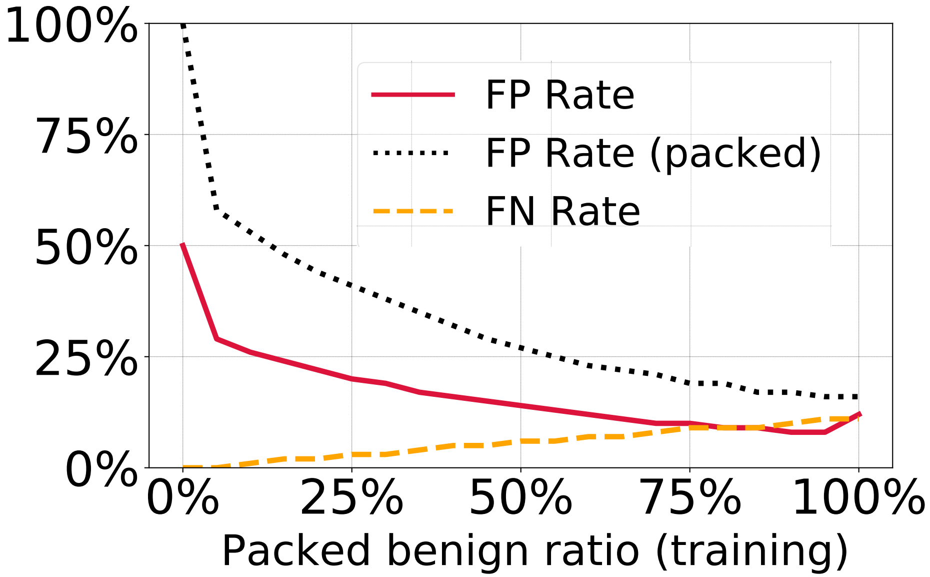
Category	# Features
PE headers	28
PE sections	570
DLL imports	4,305
API imports	19,168
Rich Header	66
Byte n-grams	13,000
Opcode n-grams	2,500
Strings	16,900
File generic	2

Our Research Questions

1. Do packers preserve static analysis features that are useful for malware classification?

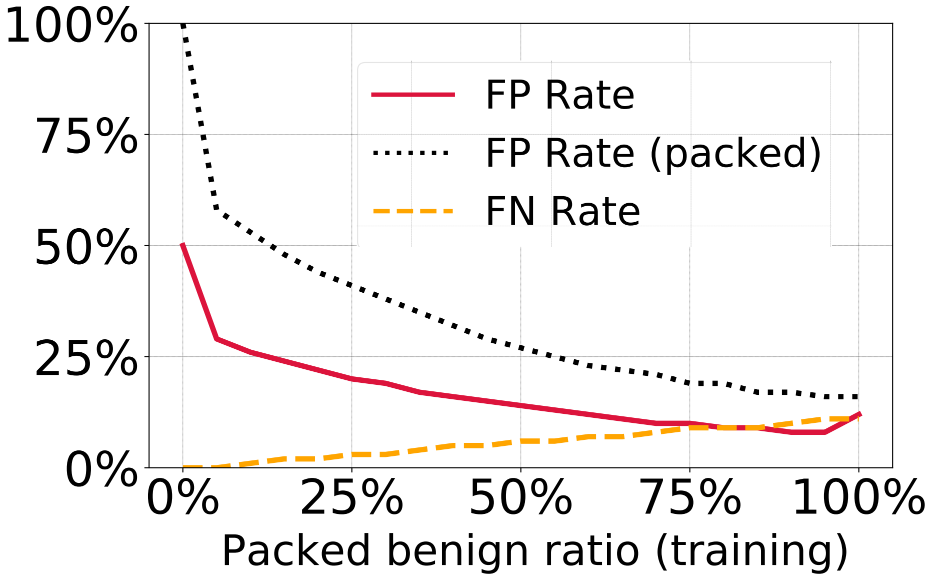
Experiment “Different Packed Ratios (lab)”

1. We exclude packed benign samples from the training set
2. Then, we keep adding more packed benign samples to the training set



Experiment “Different Packed Ratios (lab)”

1. We exclude packed benign samples from the training set.
2. Then, we keep adding more packed benign samples to the training set



- Surprisingly, the classifier is doing ok!

But, How??

- We focused on one packer at a time to identify ***useful features*** for each packer!

1. Some packers (e.g., Themida) often keep the Rich Header.
2. Packers often keep .CAB file headers in the resource sections of the executables.
3. UPX keeps one API for each DLL.

Our Research Questions

1. Do packers preserve static analysis features that are useful for malware classification?

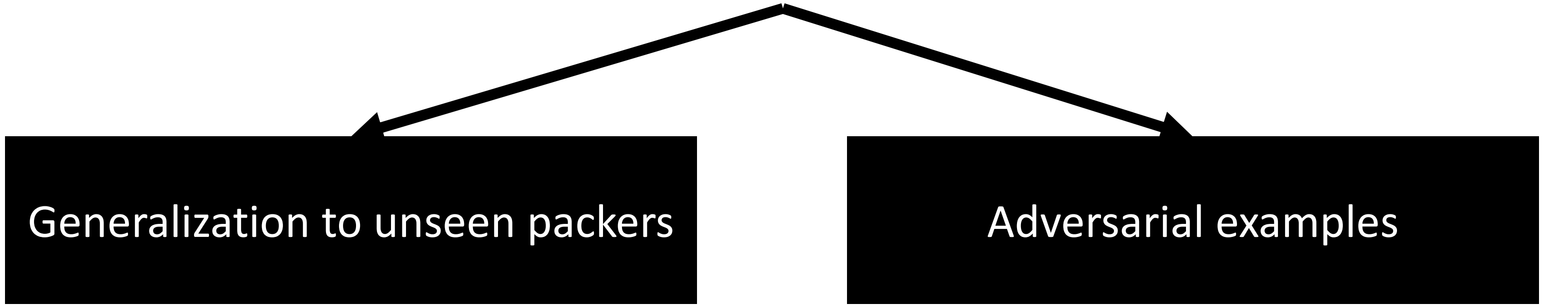
Packers preserve some information when packing programs that may be “useful” for malware classification, however, such information does not necessarily represent the real nature of samples

Our Research Questions

1. Do packers preserve static analysis features that are useful for malware classification?
2. Can such a classifier perform well in real-world scenarios?

Our Research Questions

1. Do packers preserve static analysis features that are useful for malware classification?
2. Can such a classifier perform well in real-world scenarios?

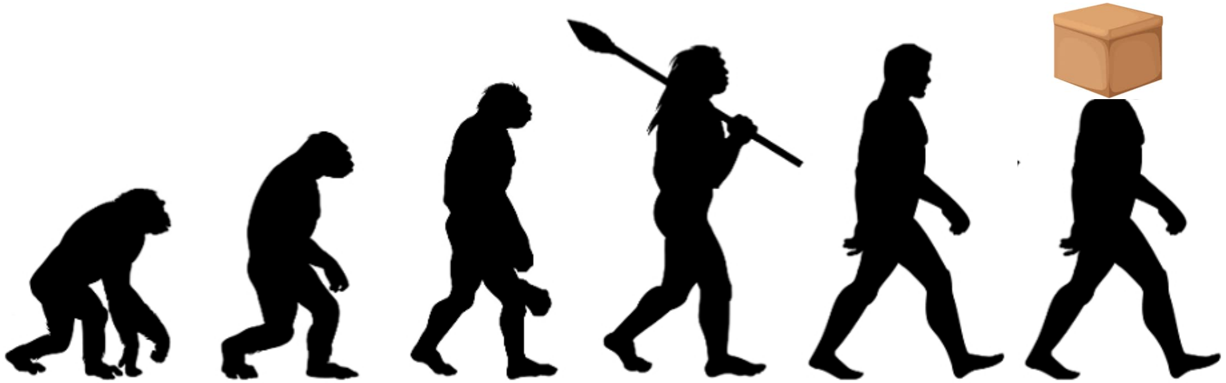


Generalization to unseen packers

Adversarial examples

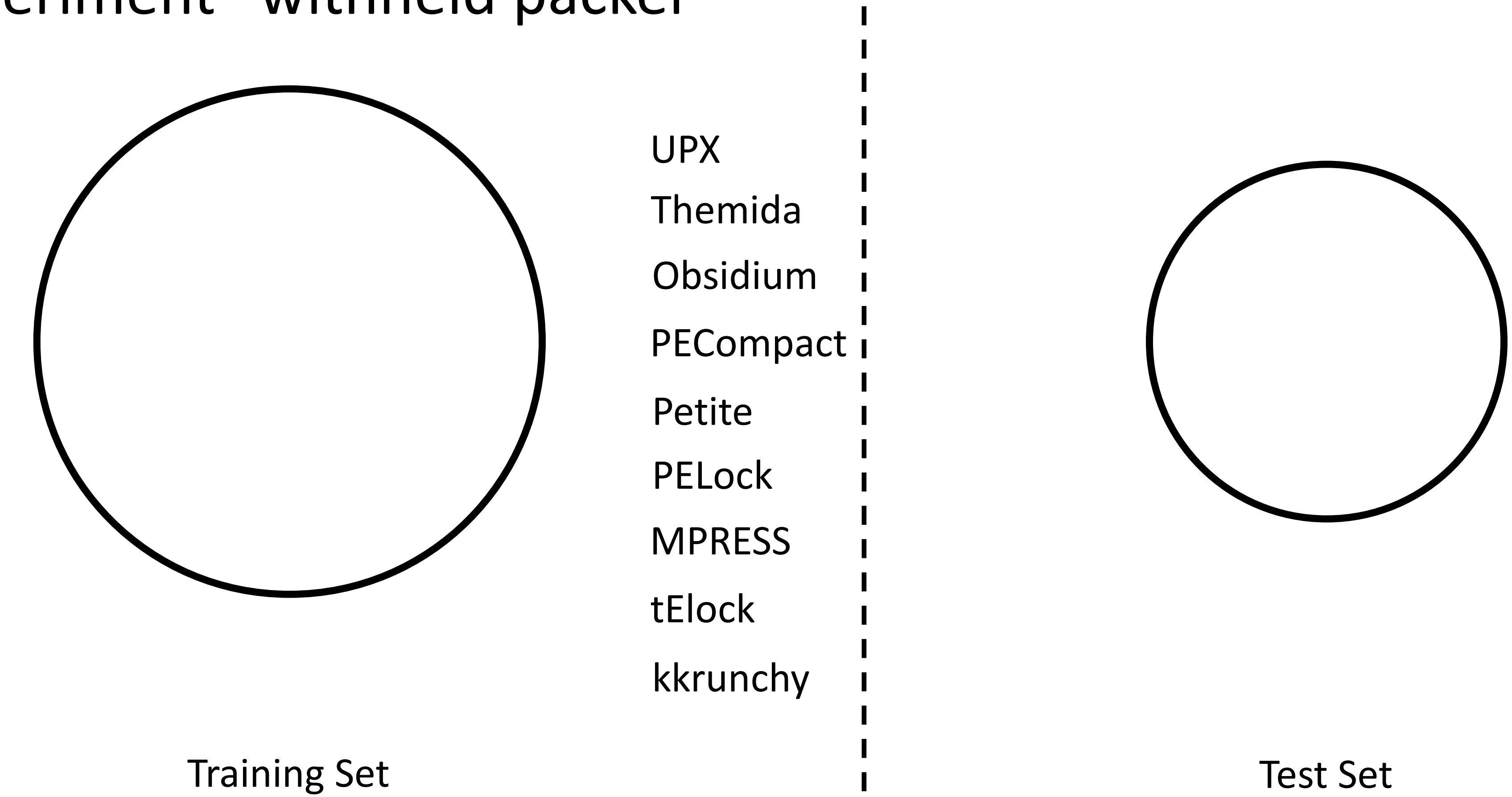
Generalization To Unseen Packers

- Runtime packers are evolving, and malware authors often tend to use their own custom packers



Generalization To Unseen Packers

1. Experiment “withheld packer”



UPX

Themida

Obsidium

PECompact

Petite

PELock

MPRESS

tElock

kkrunchy

Training Set

Test Set

Generalization To Unseen Packers

1. Experiment “withheld packer”

Withheld Packer	FPR (%)	FNR (%)
PELock	7.30	3.74
PECompact	47.49	2.14
Obsidium	17.42	3.32
Petite	5.16	4.47
tElock	43.65	2.02
Themida	6.21	3.29
MPRESS	5.43	4.53
kkrunchy	83.06	2.50
UPX	11.21	4.34

Generalization To Unseen Packers

2. Experiment “lab against wild”

- We train the classifier on Lab Dataset
- And evaluate it on packed executables in Wild Dataset

Generalization To Unseen Packers

2. Experiment “lab against wild”

- We train the classifier on Lab Dataset
- And evaluate it on packed executables in Wild Dataset

- We observed the false negative rate of 41.84%, and false positive rate of 7.27%

Poor Generalization To Unseen Packers

Adversarial Examples

- Machine-learning-based malware detectors shown to be vulnerable to adversarial samples

Adversarial Examples

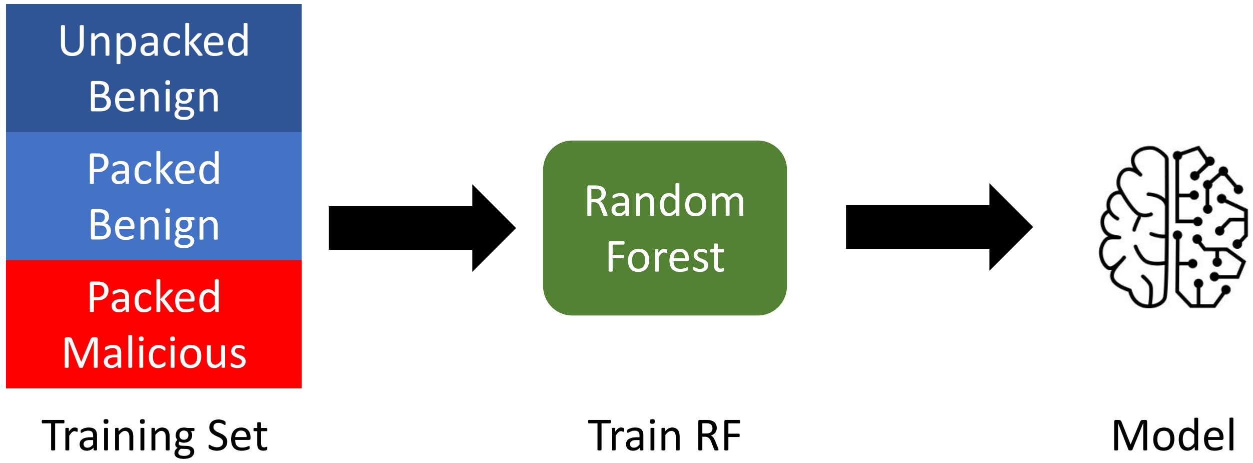
- Machine-learning-based malware detectors shown to be vulnerable to adversarial samples
- Packing produces features not directly deriving from the actual (unpacked) program

Adversarial Examples

- Machine-learning-based malware detectors shown to be vulnerable to adversarial samples
- Packing produces features not directly deriving from the actual (unpacked) program
- Generating such adversarial samples would be easier for an adversary

Adversarial Examples

Training:



Unpacked Benign

Packed Benign

Packed Malicious

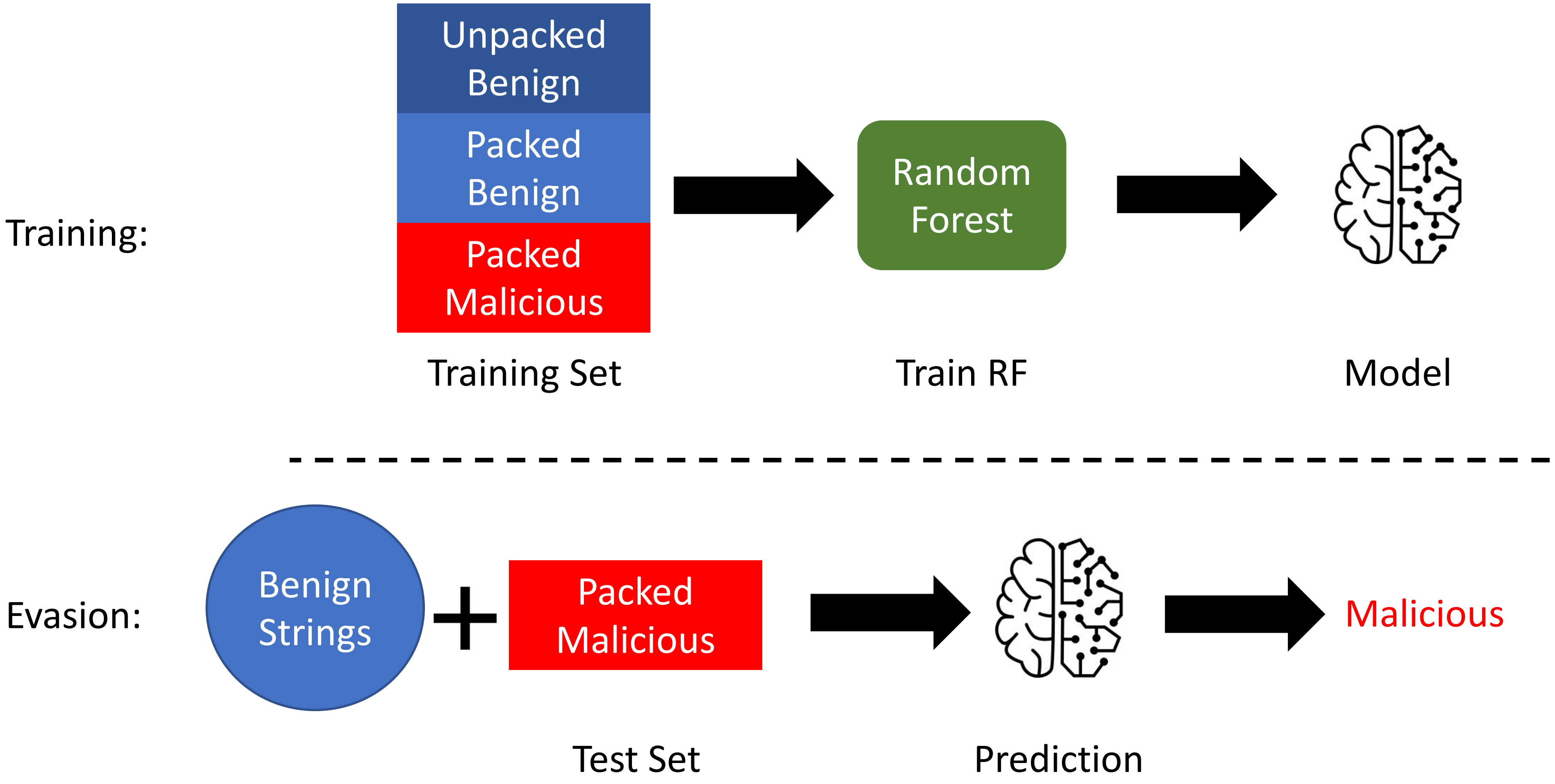
Training Set

Random Forest

Train RF

Model

Adversarial Examples



Training:

Unpacked Benign

Packed Benign

Packed Malicious

Training Set

Random Forest

Train RF

Model

Evasion:

Benign Strings

+

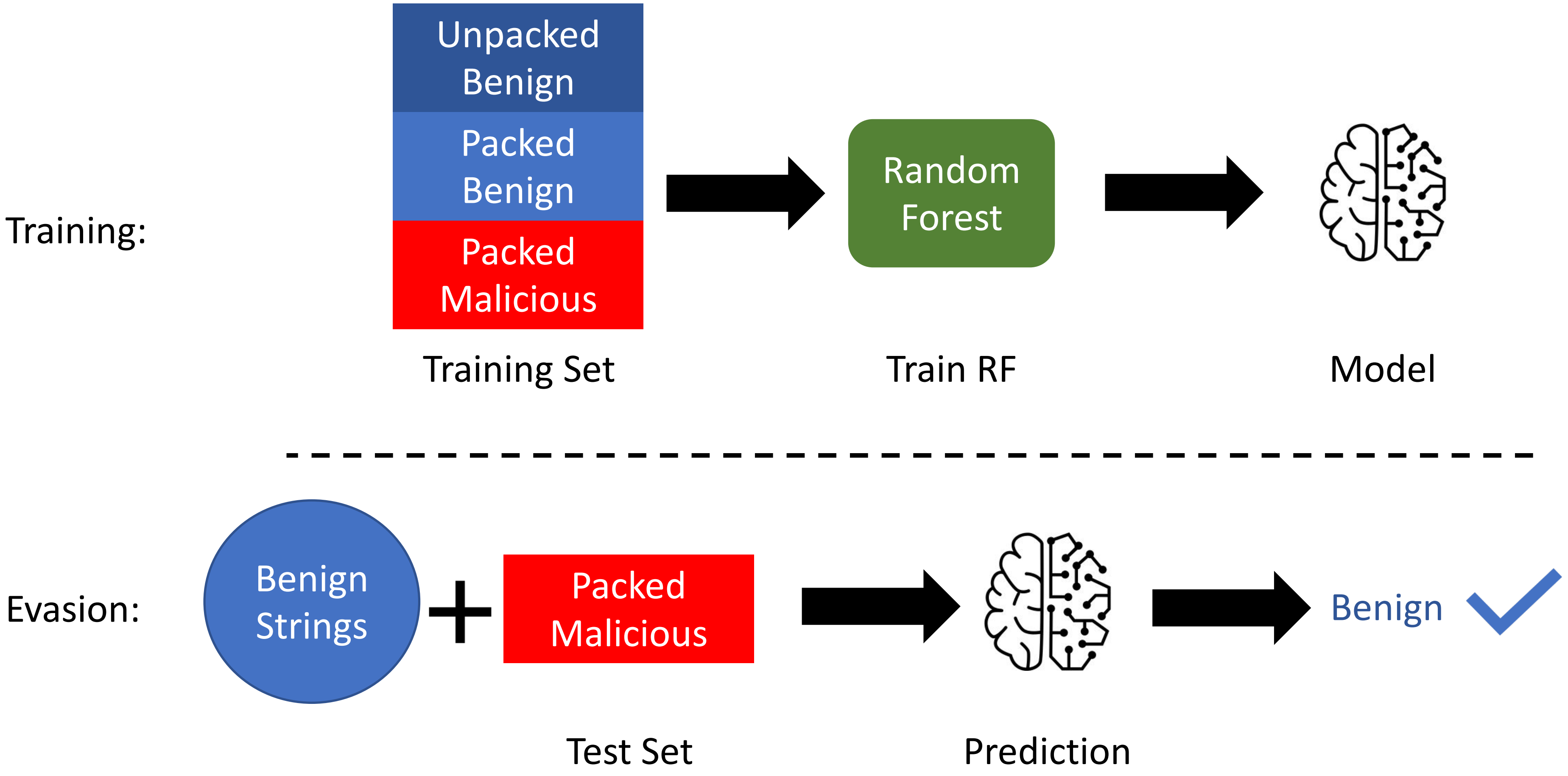
Packed Malicious

Test Set

Prediction

Malicious

Adversarial Examples



Training:

Unpacked Benign

Packed Benign

Packed Malicious

Training Set

Random Forest

Train RF

Model

Evasion:

Benign Strings

+

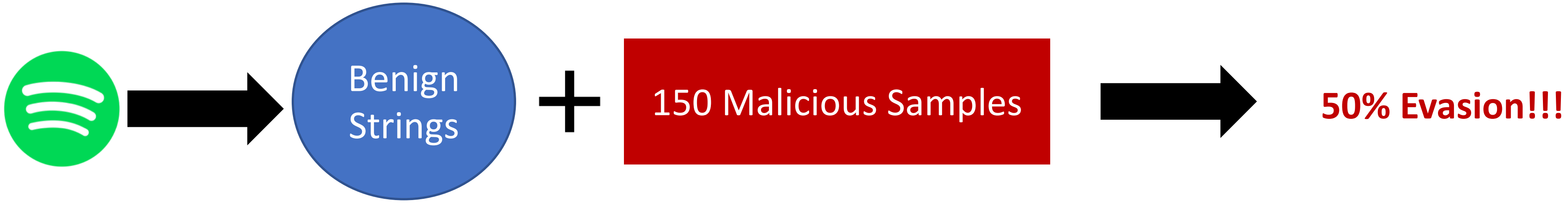
Packed Malicious

Test Set

Prediction

Benign ✓

Machine Learning Static Evasion Competition



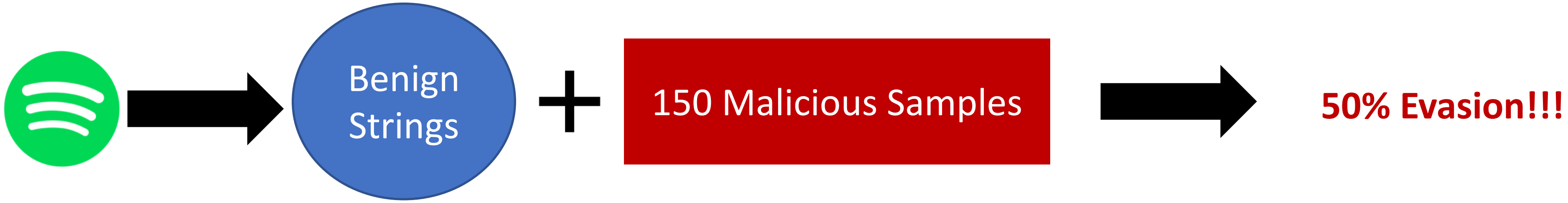
Benign Strings

+

150 Malicious Samples

50% Evasion!!!

Machine Learning Static Evasion Competition



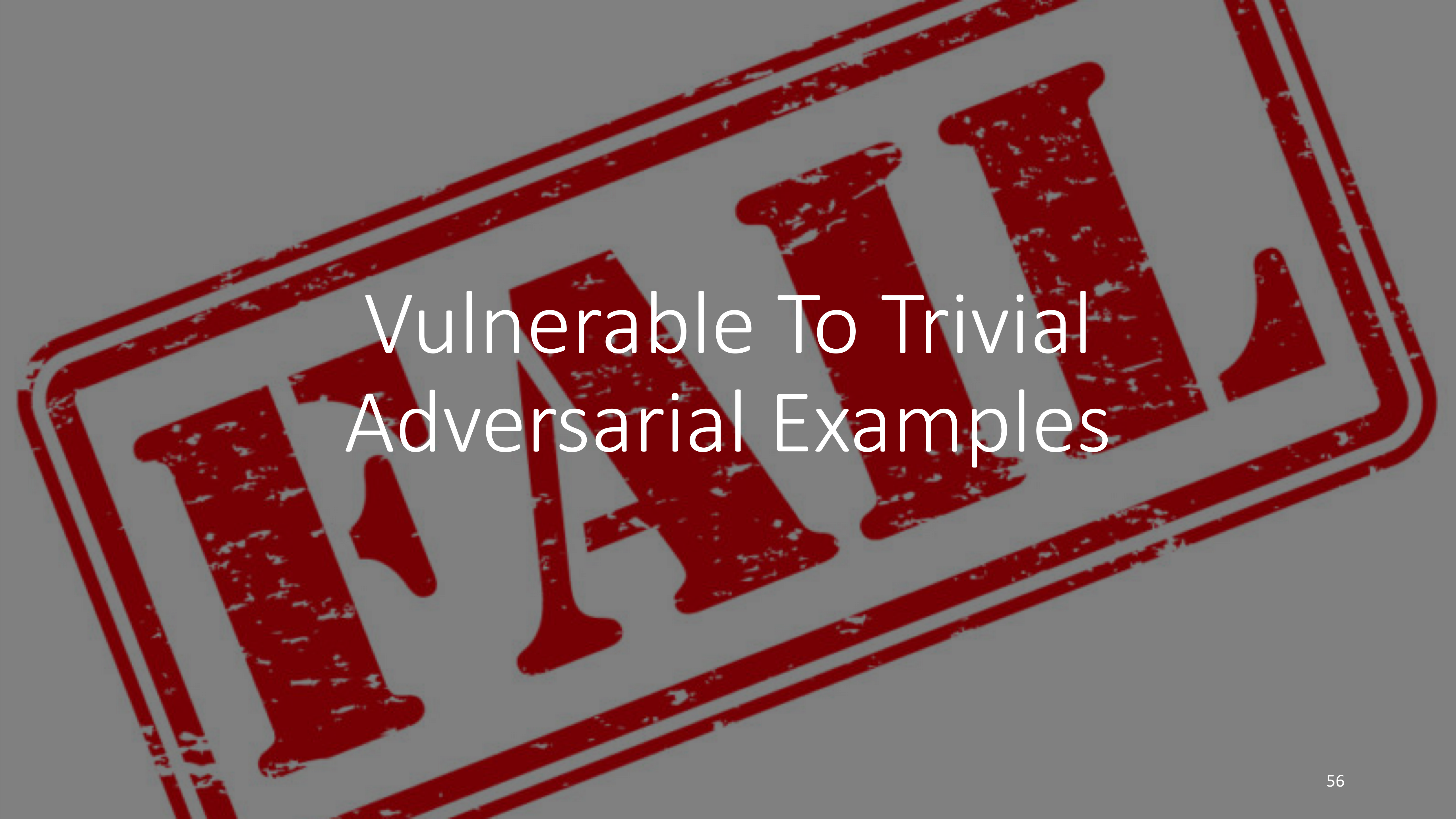
Benign Strings

+

150 Malicious Samples

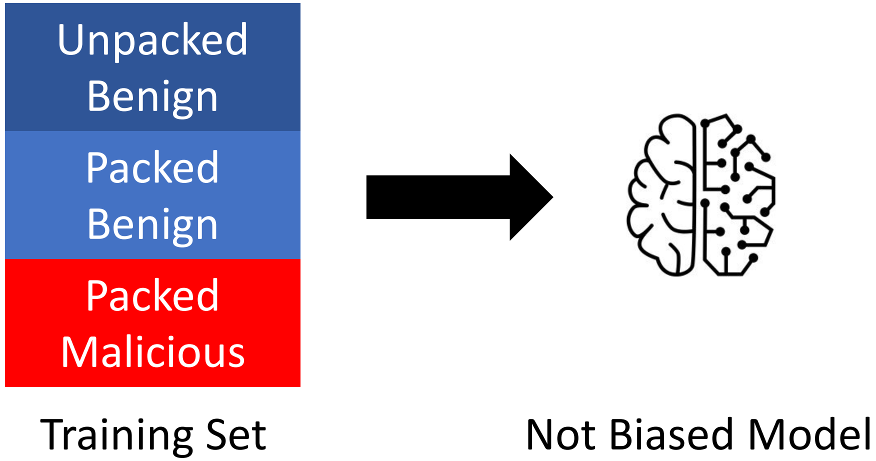
50% Evasion!!!

- Recently, a group of researchers found a very similar way to subvert an AI-based anti-malware engine
- By simply taking strings from an online gaming program and appending them to known malware, like WannaCry



Vulnerable To Trivial Adversarial Examples

Conclusion



Unpacked Benign

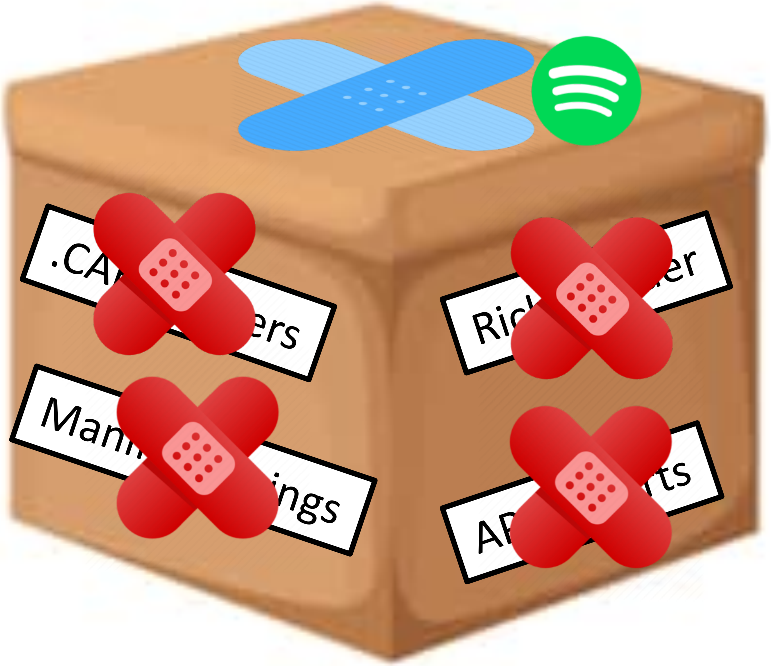
Packed Benign

Packed Malicious

Training Set

Not Biased Model

Conclusion

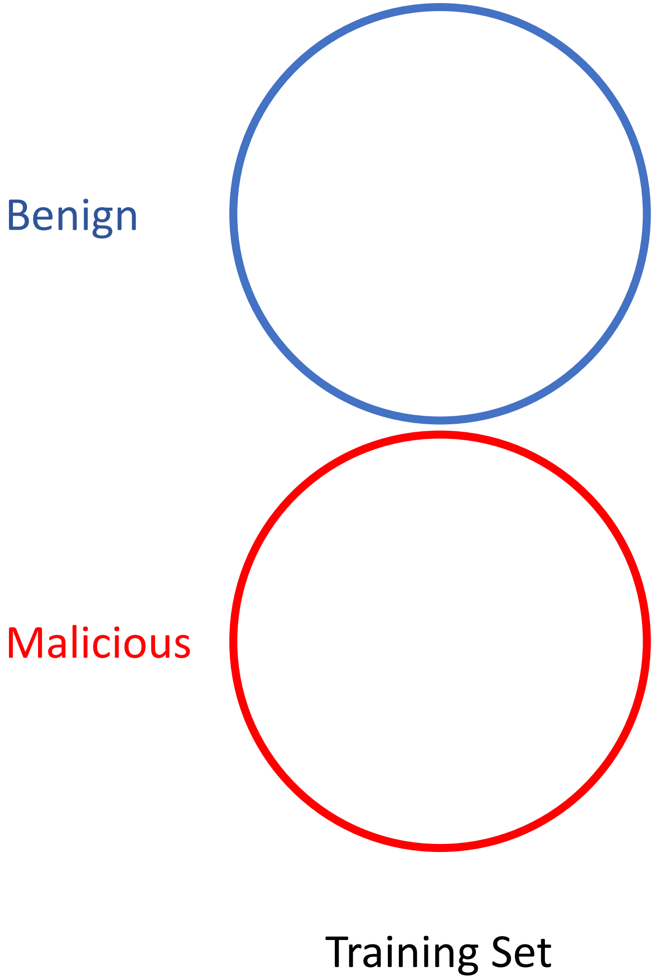


Reproducibility

- The source code and our datasets of 392,168 executables are available at https://github.com/ucsb-seclab/packware
- All experiments can be simply executed in the provided Docker image

Any Questions?

Experiment “Good-Bad Packers”



Benign

Malicious

Training Set

UPX

Themida

Obsidium

PECompact

Petite

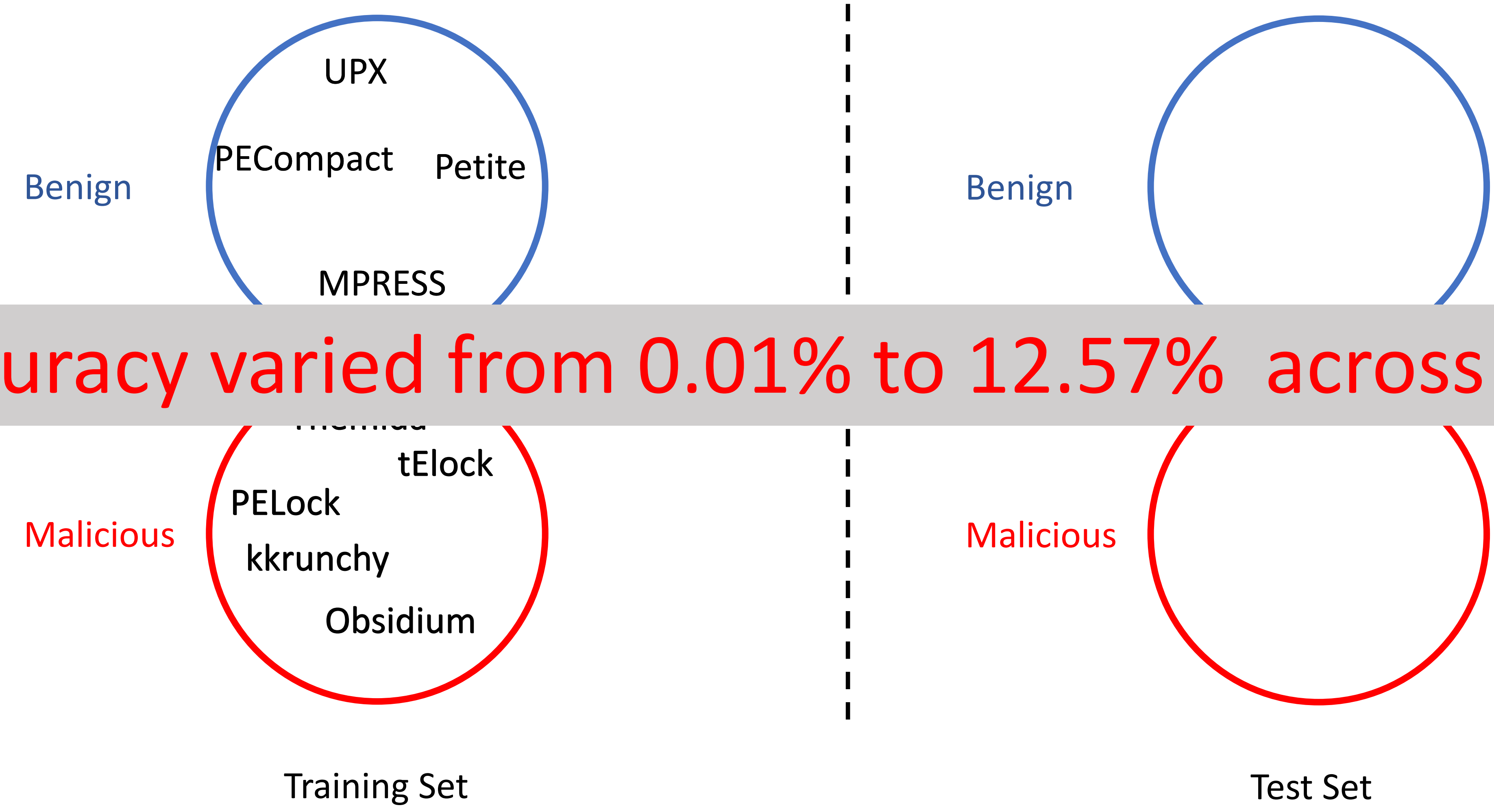
PELock

MPRESS

tElock

kkrunchy

Experiment “Good-Bad Packers”



Accuracy varied from 0.01% to 12.57% across all splits