Prevalence and Impact of Low-Entropy Packing Schemes in the Malware Ecosystem

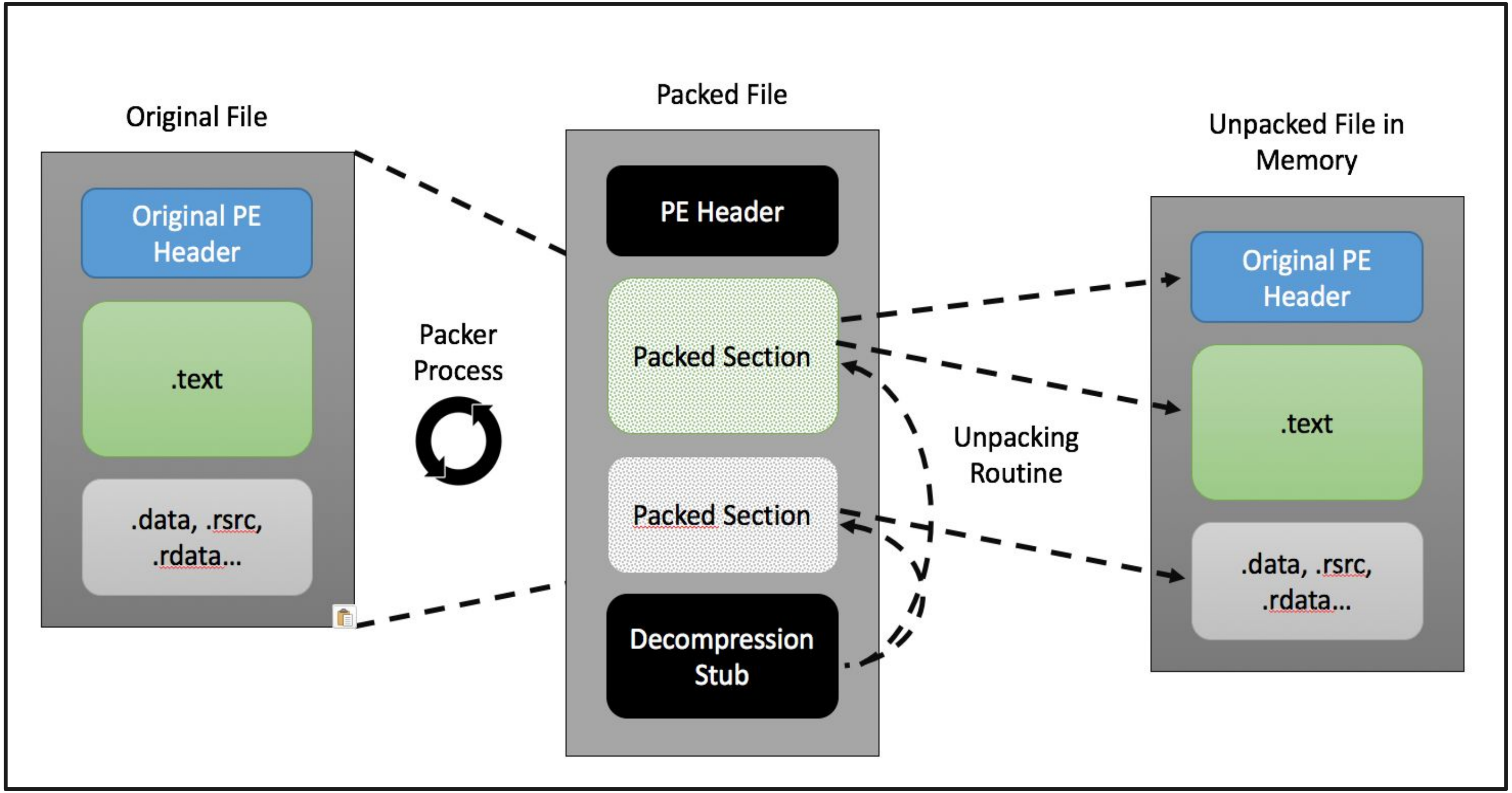
Alessandro Mantovani (EURECOM), Simone Aonzo (UniGe), Xabier Ugarte Pedrero (CISCO),
Alessio Merlo (UniGe), Davide Balzarotti (EURECOM)

UNIVERSITÀ DEGLI STUDI
DI GENOVA

EURECOM

CISCO

Packing


Original File
Original PE Header
.text
.data, .rsrc, .rdata...
Packer Process
Packed File
PE Header
Packed Section
Packed Section
Decompression Stub
Unpacking Routine
Unpacked File in Memory
Original PE Header
.text
.data, .rsrc, .rdata...

Scope / Packing Definition

(Our definition of) packing implies

- Original code present, but NOT in an executable form
- Real code recovered at run-time

(Our definition of) packing does NOT include

- JIT compilers
- Droppers
- Emulators (Themida)
- Shellcode

Packed or not packed: that is the question

- Fundamental in malware analysis
- Wrong classification ⇒
  - costly and time-consuming dynamic analysis trying to unpack the sample
  - pollute the datasets used in many malware analysis studies
  - even worse, EVASION
- Our (false) friend: the entropy
  - compressed/encrypted data has high entropy levels

Our Agenda

1. The propagation of low-entropy packed samples
2. The adopted schemes
3. Current tools/approaches vs. low-entropy packed malware

Dataset

VirusTotal

Do malware authors use low-entropy schemes to evade entropy checks?

- 50.000 Portable Executable files (excluding libraries and .Net applications)
- 2013 - 2019
- Classified as malicious by more than 20 antivirus engines
- Entropy $H < 7.0$
  - entire file [1]
  - each section [2]
  - overlay data

Ugarte-Pedrero, Balzarotti, Santos, Bringas.
Deep packer inspection: A longitudinal study of the complexity of run-time (2015)

pefile -- Python module

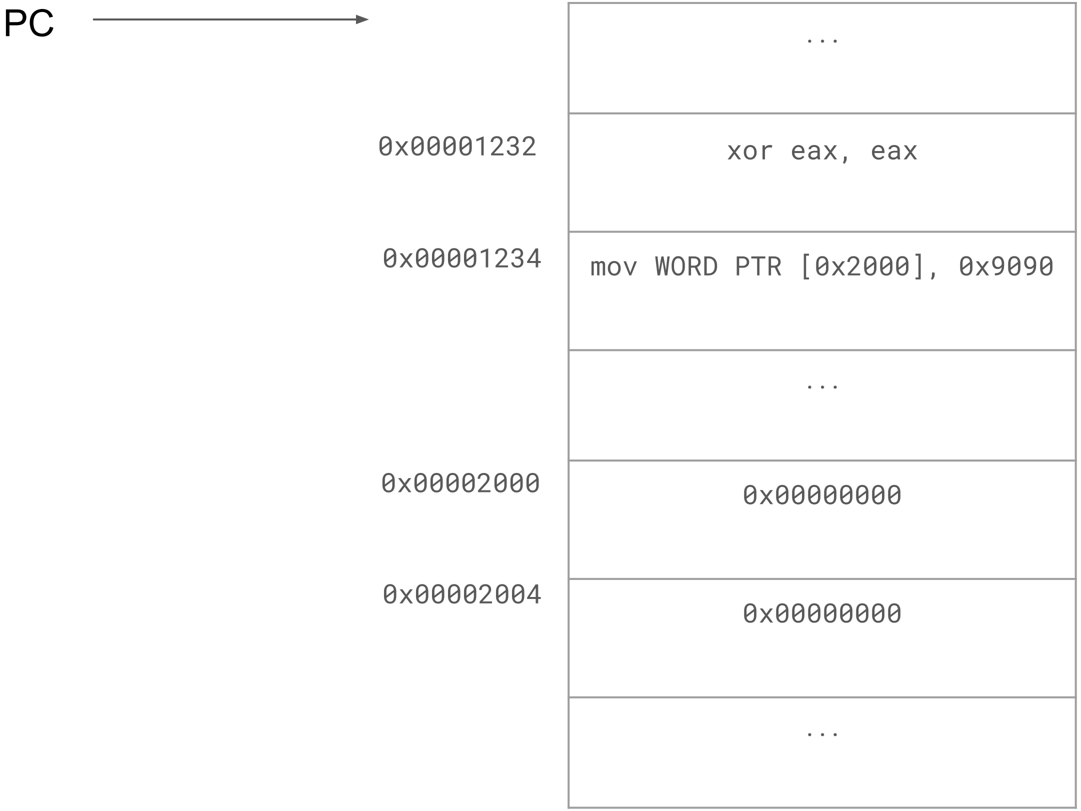
Manalyze -- static analyzer for PE executables

---

[1] Lyda and Hamrock. Using entropy analysis to find encrypted and packed malware (2007).

[2] Han and Lee. Packed PE file detection for malware forensics (2009).

Packer Detector (⅕)



PC

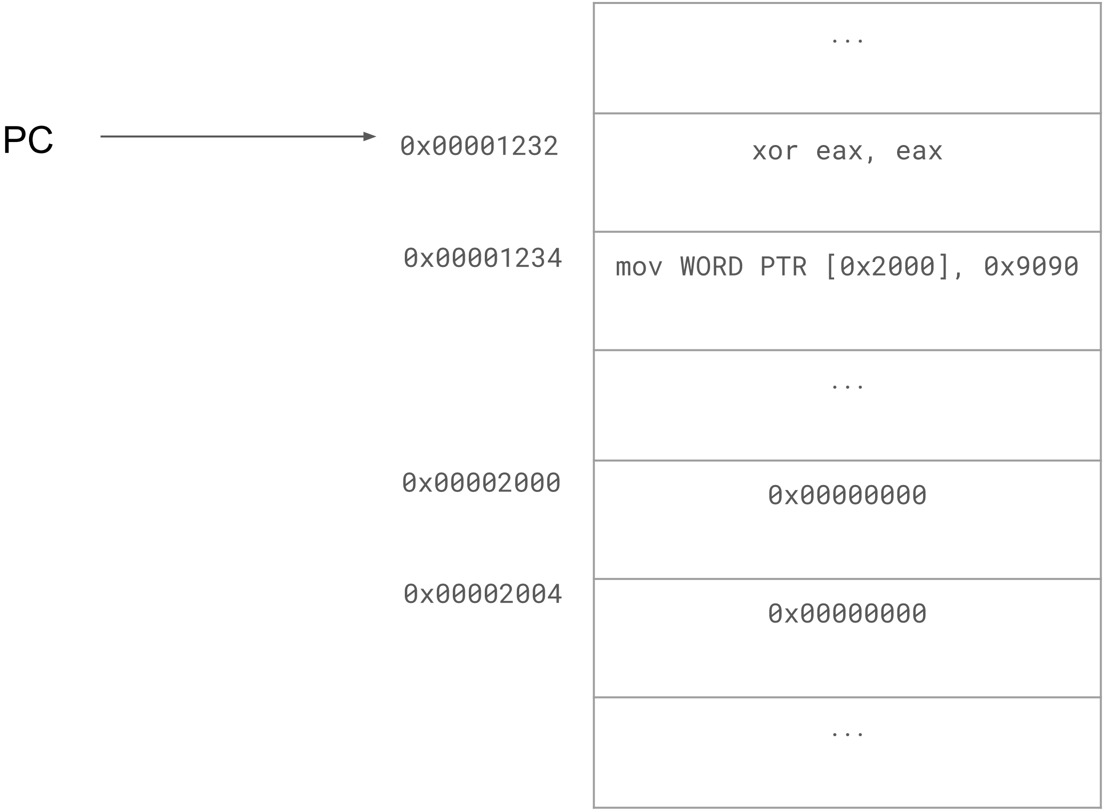
	...
0x00001232	xor eax, eax
0x00001234	mov WORD PTR [0x2000], 0x9090
	...
0x00002000	0x00000000
0x00002004	0x00000000
	...

Lists status

WL = [ ]

WXL = [ ]

Packer Detector (⅖)



PC → 0x00001232

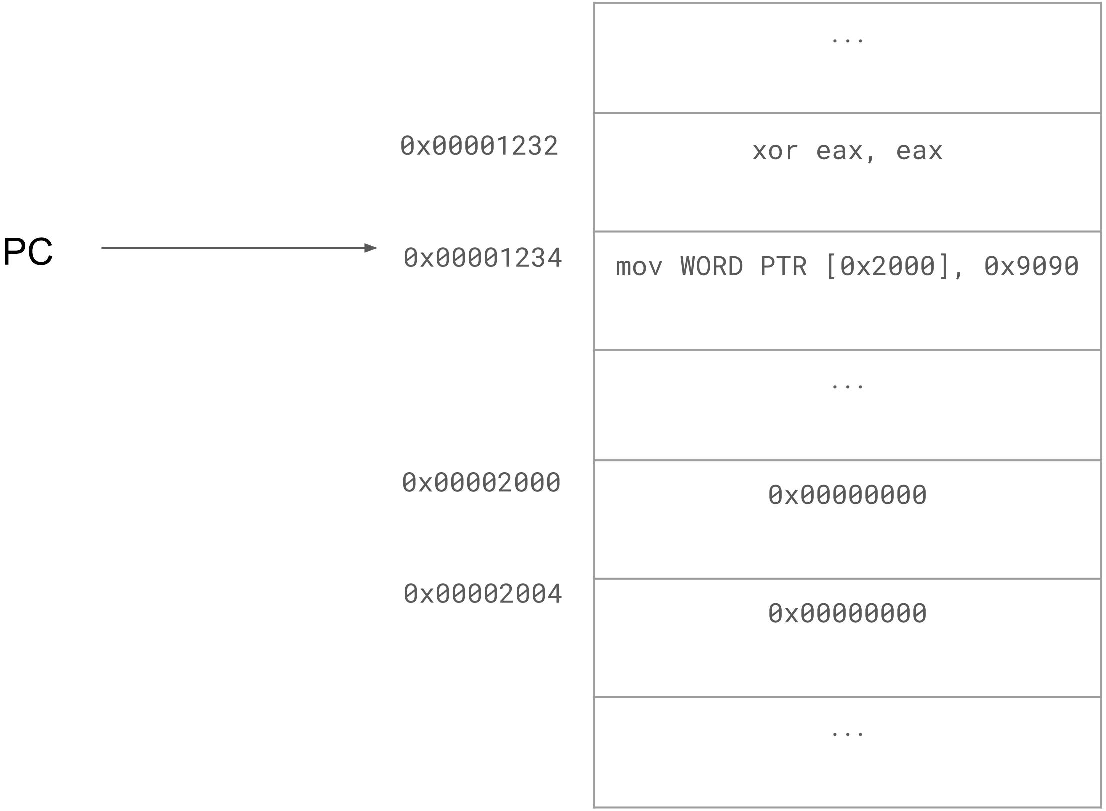
	...
0x00001232	xor eax, eax
0x00001234	mov WORD PTR [0x2000], 0x9090
	...
0x00002000	0x00000000
0x00002004	0x00000000
	...

Lists status

WL = [ ]

WXL = [ ]

Packer Detector (⅗)



PC

0x00001232

0x00001234

0x00002000

0x00002004

...

`xor eax, eax`

`mov WORD PTR [0x2000], 0x9090`

...

0x00000000

0x00000000

...

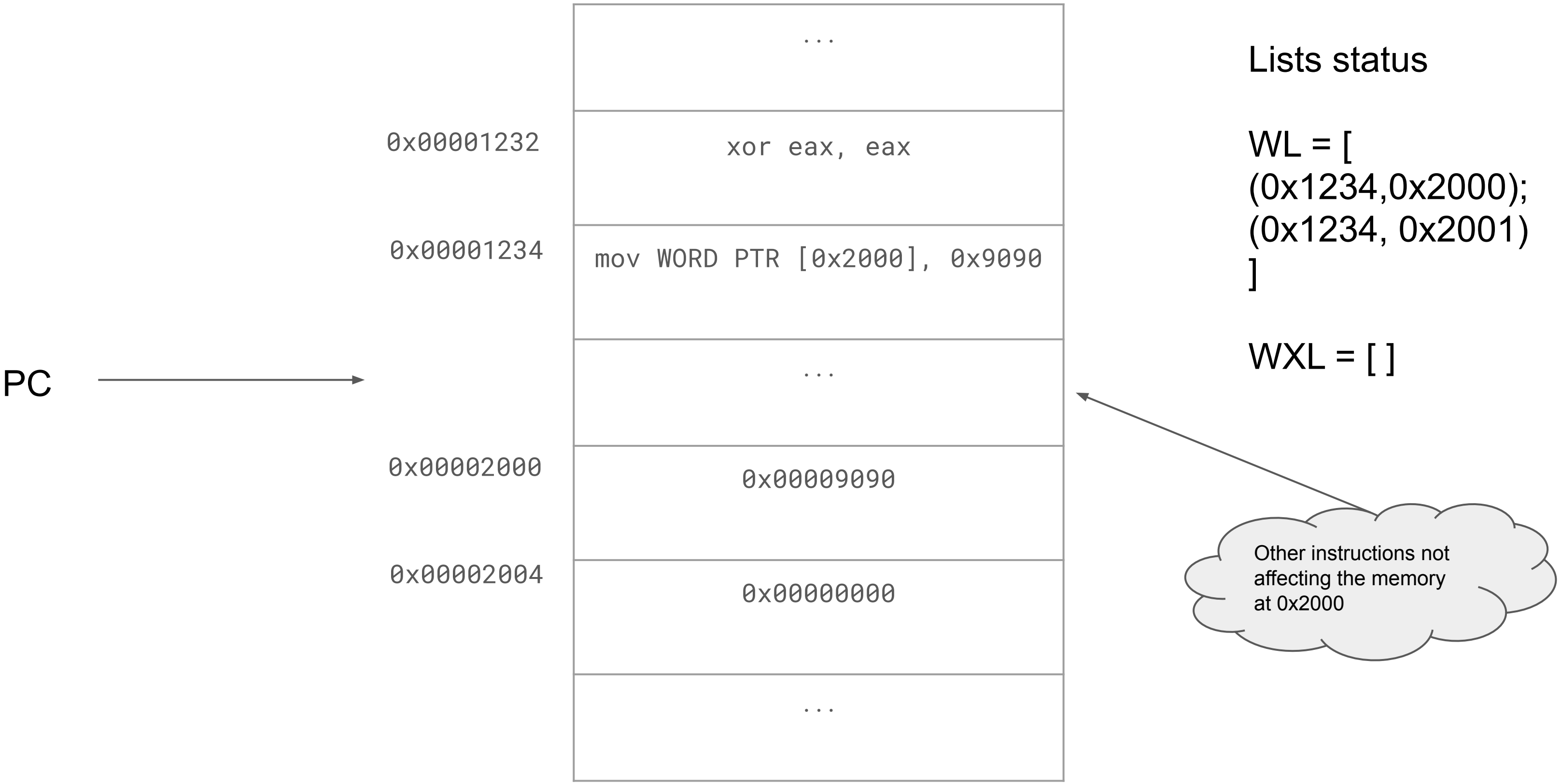
Lists status

WL = [
(0x1234,0x2000);
(0x1234, 0x2001)
]

WXL = [ ]

After executing the current instruction the memory at 0x2000 will be written

Packer Detector (⅘)

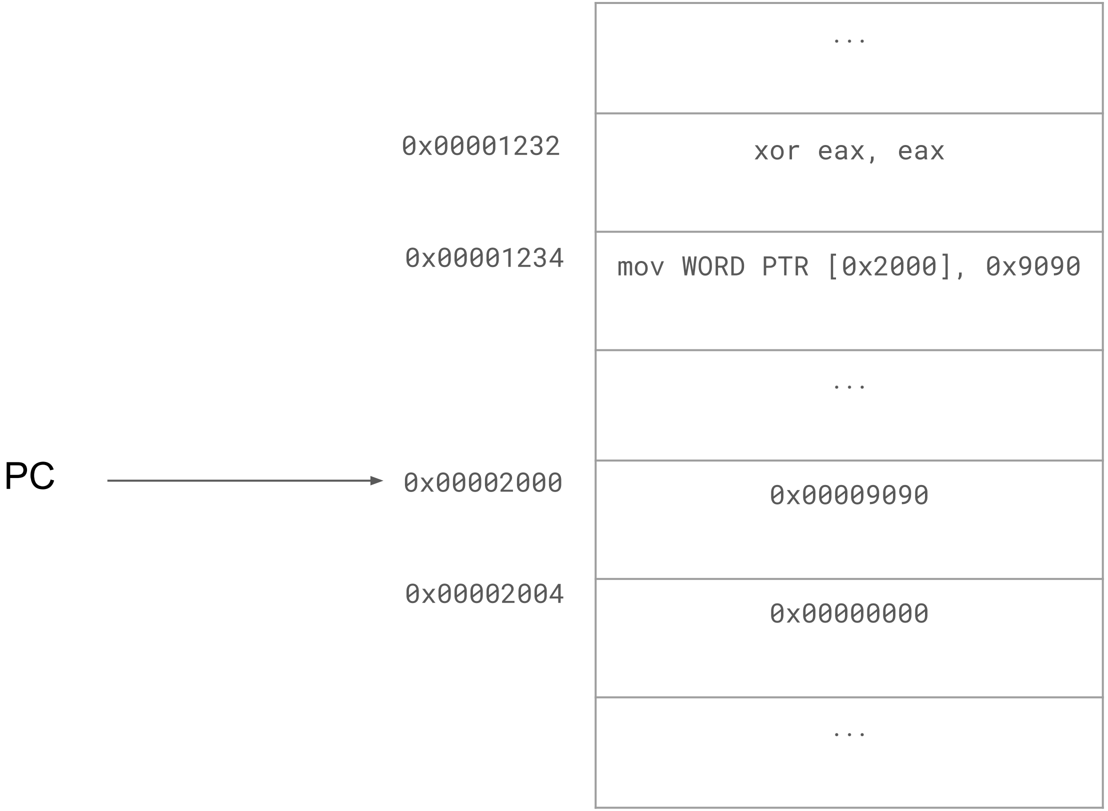


Lists status

WL = [
(0x1234,0x2000);
(0x1234, 0x2001)
]

WXL = [ ]

Packer Detector (5/5)



PC

	...
0x00001232	xor eax, eax
0x00001234	mov WORD PTR [0x2000], 0x9090
	...
0x00002000	0x00009090
0x00002004	0x00000000
	...

Lists status

WL = [
(0x1234,0x2000);
(0x1234, 0x2001)
]

WXL = [(0x1234, 0x2000)]

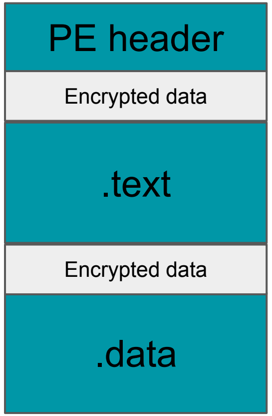
Packer Detector - False Negatives

- False Negatives -- packed samples detected as not packed
    - unexpected crash
    - virtual environment detection
    - missing dependencies
    - incorrect command line arguments
- We discarded the samples that did not exhibit a sufficient runtime behavior
    - did not invoke at least 10 disk or network-related syscalls
    - samples whose executed instructions did not span at least five memory pages
- 50.000 - 3.705 = 46.295

Hidden high-entropy data

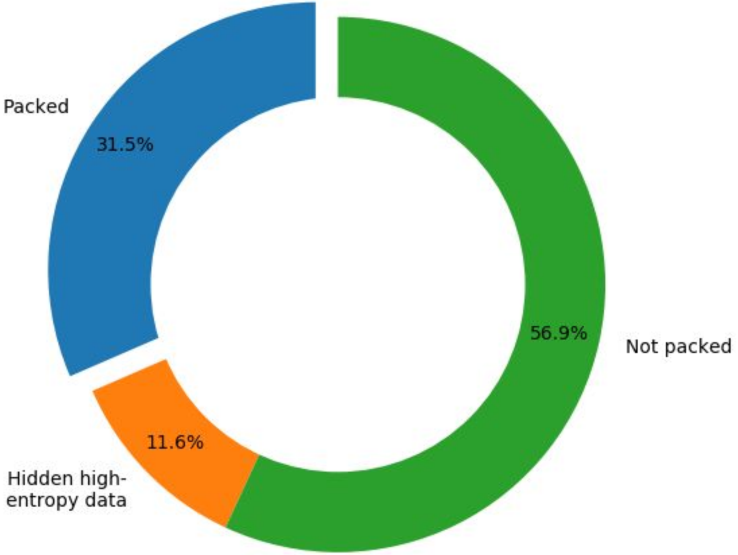
While packed with a high-entropy scheme, these samples evaded our set of filters

- Encrypted data, but the data was
  - not stored in any of the section
  - nor in the overlay area
- **11.6%** (5.386/46.295)
  - dominated by two families: *hematite* and *hworld*
- E.g., *hematite*
  - file infector
  - area created between the PE header and the first section



Packer Detector - Results

31.5% (14.583/46.295) ⇒ entropy alone is a very poor metric to select packed samples



Schemes Taxonomy w.r.t. Entropy

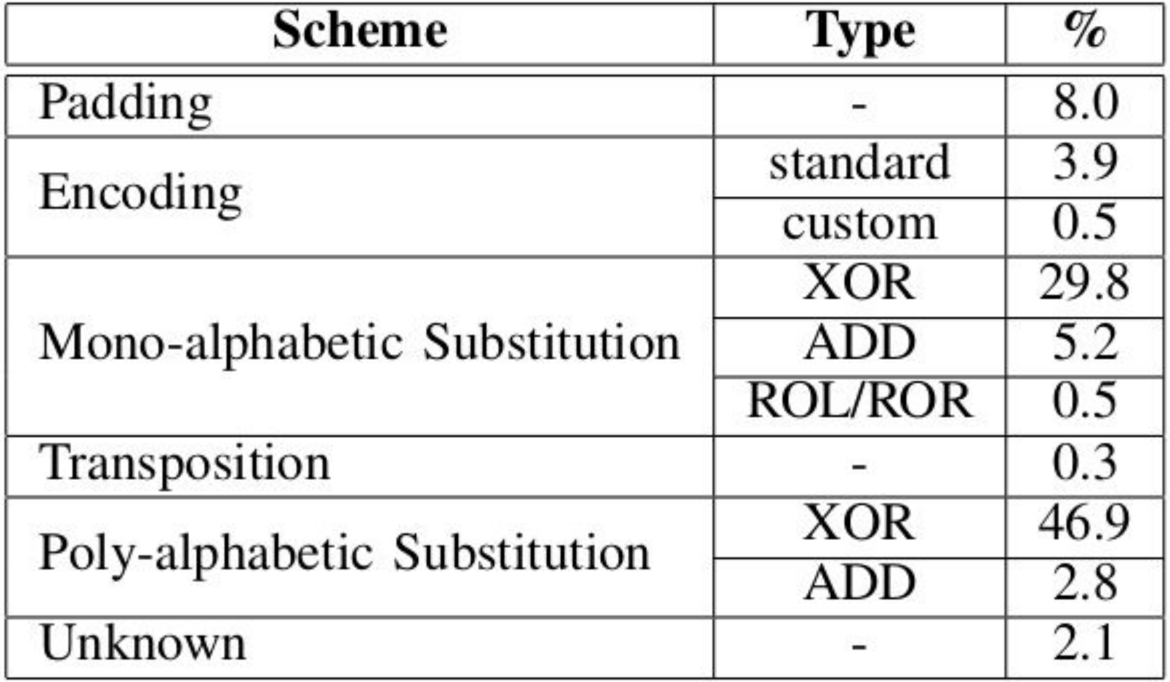
1. **Decreasing**
    - Byte Padding
    - Encoding
2. **Unchanged**
    - Transposition
    - Monoalphabetic Substitution
3. **Slightly Increasing**
    - Polyalphabetic Substitution

Scheme Classifier

Relies on the output of Packer Detector ⇒ Written and eXecuted List [**WXL**]

- Every packing scheme needs to follow the same steps while unpacking
  - locate and access the source buffer that contains the packed data
  - perform operations on such data
  - write the unpacked data in the destination buffer
- We use PANDA to perform deterministic record and replay of a sample
  - *⟨**PCx** , AWy⟩ ∈ [**WXL**]*
  - backward data-flow analysis to locate the source buffer
- Decision making based on the byte distribution of source and destination buffers

Scheme Classifier - Results



Scheme	**Type**	**%**
Padding	-	8.0
Encoding	standard	3.9
	custom	0.5
Mono-alphabetic Substitution	XOR	29.8
	ADD	5.2
	ROL/ROR	0.5
Transposition	-	0.3
Poly-alphabetic Substitution	XOR	46.9
	ADD	2.8
Unknown	-	2.1

Case Study: Custom Encoding (*Emotet*)

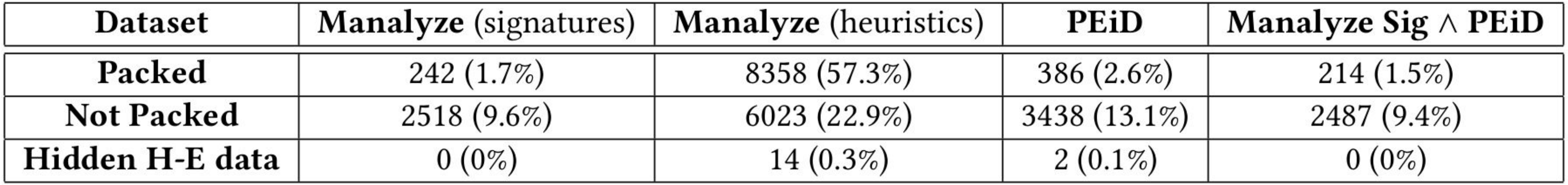
Two layers of packing

- The second layer uses a custom high-entropy encryption with an 8-bytes long key
- The first layer reduces the entropy from 7.63 to 6.57
- Custom encoding + byte padding
- Packed data and keys stored in the sections: “`.rsrc`” and “`.rdata`”

Signature and Rule-Based Packing Detection

- Detect It Easy (DIE)
  - signatures based on a scripting language
- PEiD
  - signatures only contain low-level byte patterns
- Manalyze
  - signatures
  - PE structure heuristics
    - unusual section names
    - sections WX
    - low number of imported functions
    - resources bigger than the file itself
    - sections with H > 7.0

Signature and Rule-Based Packing Detection - Results



Dataset	**Manalyze** (signatures)	**Manalyze** (heuristics)	**PEiD**	**Manalyze Sig ∧ PEiD**
Packed	242 (1.7%)	8358 (57.3%)	386 (2.6%)	214 (1.5%)
Not Packed	2518 (9.6%)	6023 (22.9%)	3438 (13.1%)	2487 (9.4%)
Hidden H-E data	0 (0%)	14 (0.3%)	2 (0.1%)	0 (0%)

- DIE detects no well-known packer in our entire dataset
- PEiD and Manalyze generated a large number of false positives
  - detected the presence of packing more often in unpacked samples than in the packed group
- Manalyze alerts are based on sections names used by some off-the-shelf packers
  - why the malware authors used those names?
  - they could be fake clues used on purpose to deceive automated tools

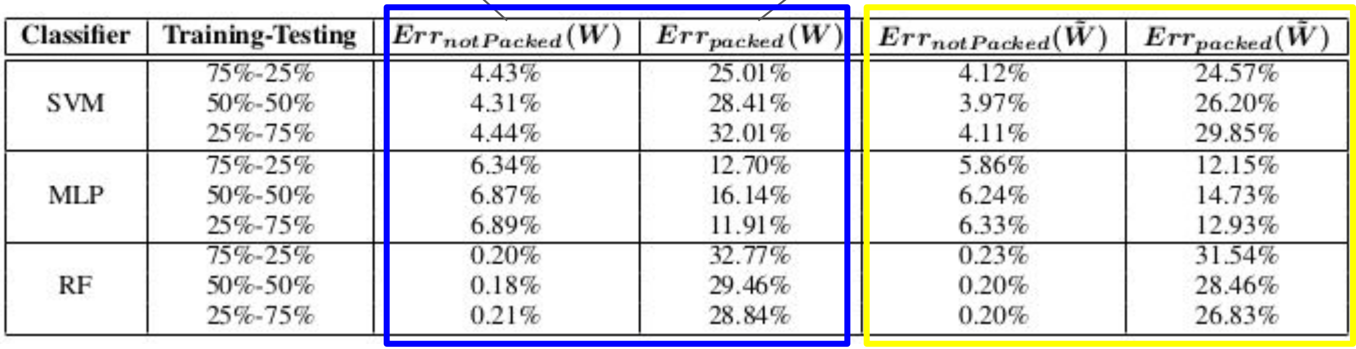
ML Packing Detection

- 15 approaches deal with this problem (SOTA)
- Several features categories
    - PE structure, heuristics, opcodes, n-grams, statistics, entropy
- Features vector ($\mathbf{W}$): union of all features from previous studies
    - A separate features vector excluding the entropy ($\tilde{\mathbf{W}}$) 😉
- The most popular classifiers: SVM, RF, MLP
- Dataset: low entropy packed + not packed (~40K)

ML Packing Detection - Results

$$Err_{notPacked} = \frac{|FP|}{|TeS_{notPacked}|}$$

$$Err_{packed} = \frac{|FN|}{|TeS_{packed}|}$$



Classifier	Training-Testing	$Err_{notPacked}(W)$	$Err_{packed}(W)$	$Err_{notPacked}(\bar{W})$	$Err_{packed}(\bar{W})$
SVM	75%-25%	4.43%	25.01%	4.12%	24.57%
	50%-50%	4.31%	28.41%	3.97%	26.20%
	25%-75%	4.44%	32.01%	4.11%	29.85%
MLP	75%-25%	6.34%	12.70%	5.86%	12.15%
	50%-50%	6.87%	16.14%	6.24%	14.73%
	25%-75%	6.89%	11.91%	6.33%	12.93%
RF	75%-25%	0.20%	32.77%	0.23%	31.54%
	50%-50%	0.18%	29.46%	0.20%	28.46%
	25%-75%	0.21%	28.84%	0.20%	26.83%

Considering H

Not Considering H

NO classifier was able to identify accurately low-entropy packed malware!

Key Takeaways

Conclusions

- Low-entropy packing schemes are a real and widespread problem
- Existing static analysis techniques are unsuccessful against them
    - Entropy ❌
    - Signature and Rule-Based ❌
    - Machine Learning ❌
- There is need for new solutions
- Low-entropy packing schemes must be considered in future experiments

-- Thank you for your attention --