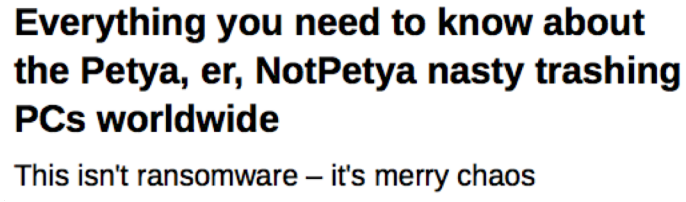
JAST: Fully Syntactic Detection of Malicious (Obfuscated) JavaScript

Blackhoodie‘18 | 16/11/18

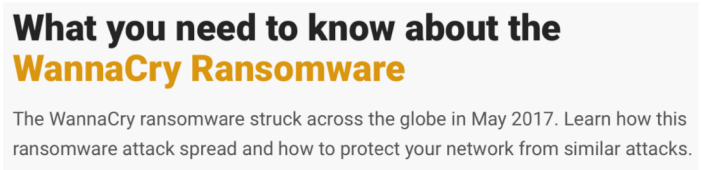
A. Fass, R. Krawczyk, M. Backes, B. Stock

Context

- Increase in the number / impact of cyber attacks



- Ex: crypto-trojans using JavaScript as payload



Why JavaScript?

CISPA HELMHOLTZ-ZENTRUM i. G.

- Enhance the interactivity of websites
- Can provide a basis for attacks
  - Offloads the work to the user's browser
    - Drive-by download attacks
    - Redirections
    - Crypto mining
  - Windows hosts: execution after double-click
    - Malicious email attachments

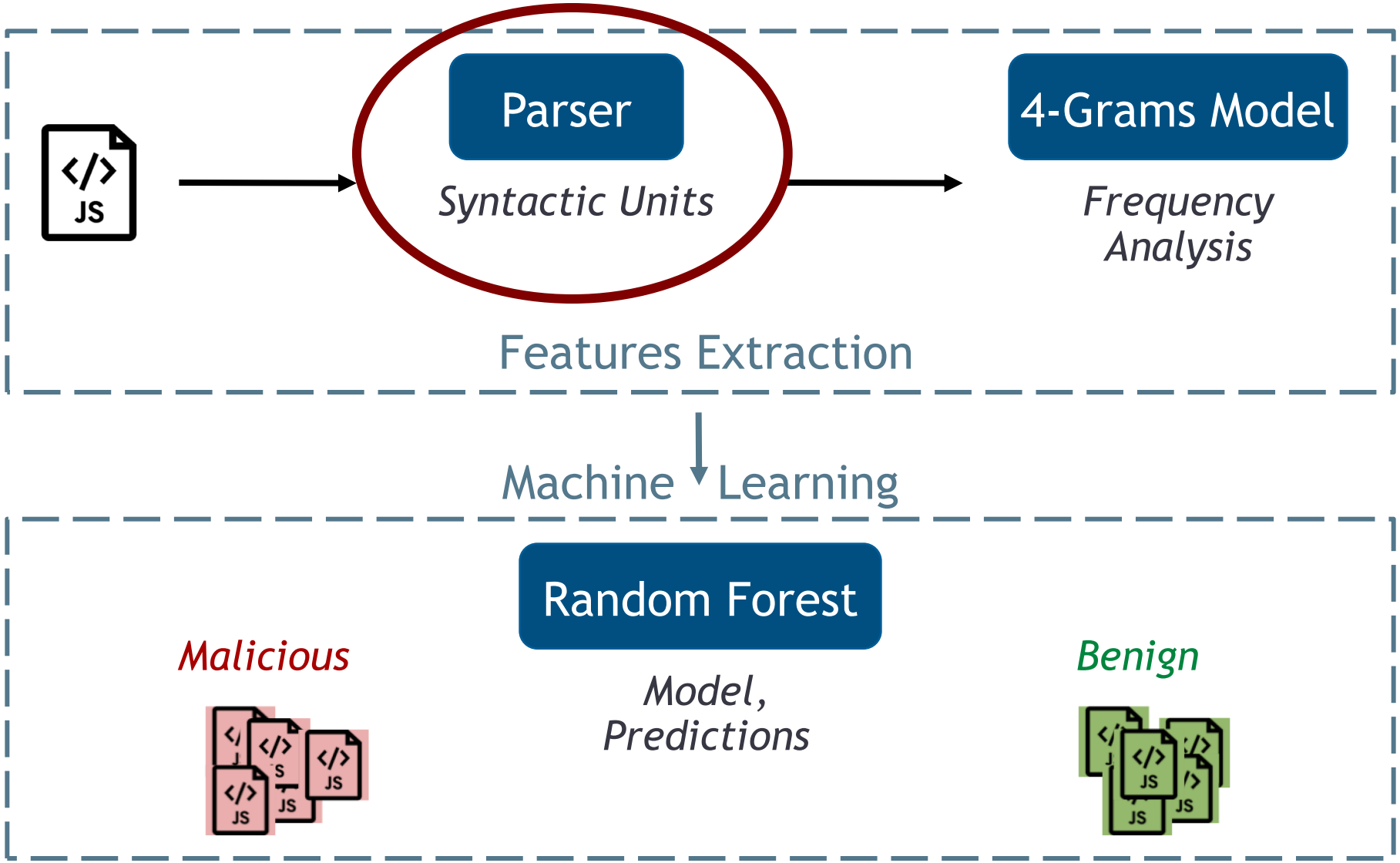
Motivations

- Plethora of attacks
  -  Manual analysis
  -  Machine learning
- Obfuscation techniques
  -  Static analysis
  -  AST-based analysis

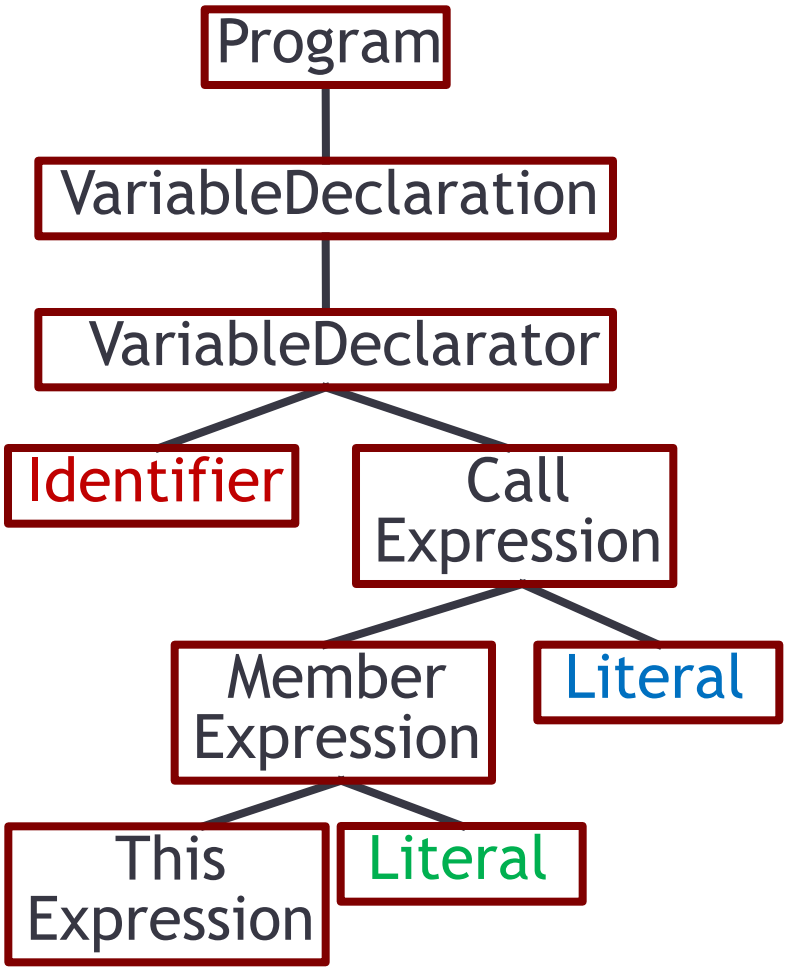
Why an AST-Based Analysis?

- Identifies programmatic and structural constructs
- Provides code abstraction
- Identifies specific and recurrent malicious patterns
- Bypasses obfuscation
- Detects unknown malicious JS variants

Methodology



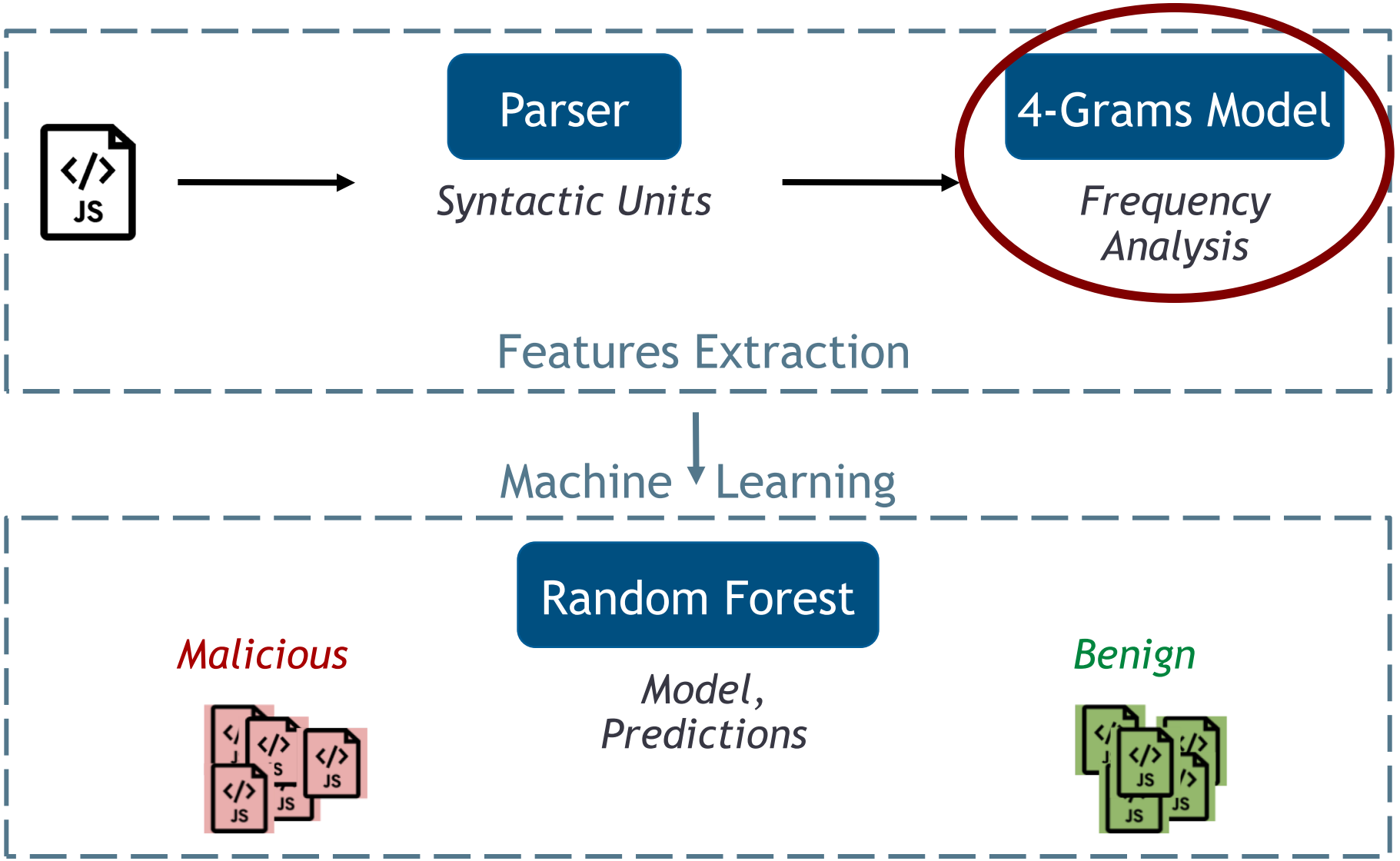
Extraction of Syntactic Units



Identifier
ThisExpression
Literal
MemberExpression
Literal
CallExpression
VariableDeclarator
VariableDeclaration
Program

Source: esprima parser
https://github.com/jquery/esprima

Methodology



Frequency Analysis of 4-Grams

Serialisation

[15 , 0 , 3 , 0 , 3 , 0 , 12 , 4 , 8]

4-grams production

(15 , 0 , 3 , 0)	1/6	2,000
(0 , 3 , 0 , 3)	1/6	143
(3 , 0 , 3 , 0)	1/6	677
(0 , 3 , 0 , 12)	1/6	152
(3 , 0 , 12 , 4)	1/6	806
(0 , 12 , 4 , 8)	1/6	257

Frequency

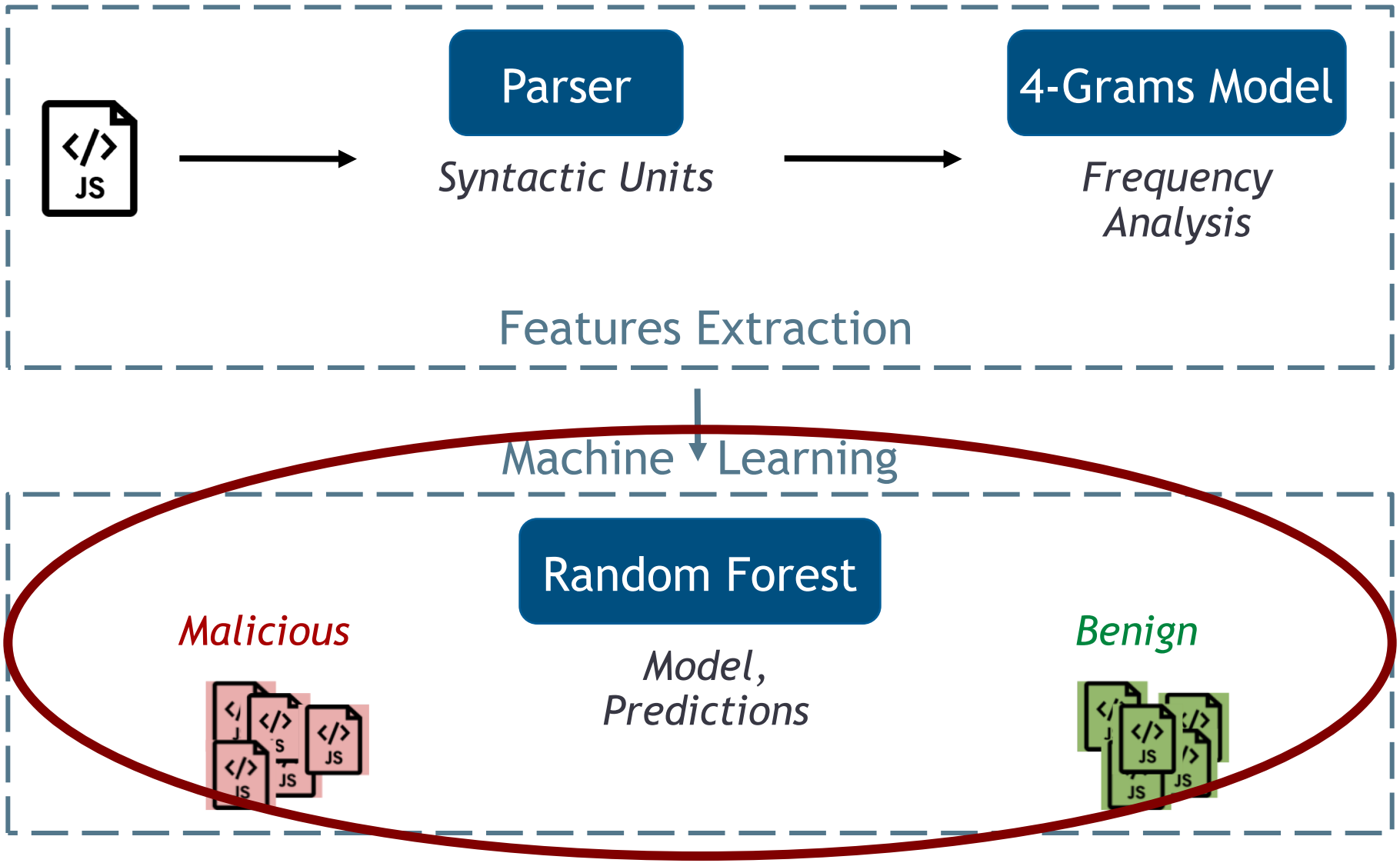
Position in vector

Storage in a vector

- Set of 4-grams: reflect short patterns

[…, 1/6, …, 1/6, … , 1/6, …,1/6, …,1/6, …,1/6, …]

Methodology



Learning and Classification

- Classifier: Random Forest, 500 trees
- Build a generic model to distinguish benign from malicious JS samples
- Classification of unknown JS instances

Experimental Data Sets

JS type	Creation	#JS	Label	Obfuscated
Emails	2017-2018	85,059	Malicious	y
Microsoft	2015-2018	17,668	Benign	y
Games	N/A	2,007	Benign	n
Frameworks	N/A	434	Benign	N/A
Atom	2011-2018	137	Benign	y

Experimental Data Sets

JS type	Creation	#JS	Label	Obfuscated
Emails				
Microsoft				
Games				
Frameworks				
Atom				

Why email attachments?

➢Common and effective way to spread JScript-Loader

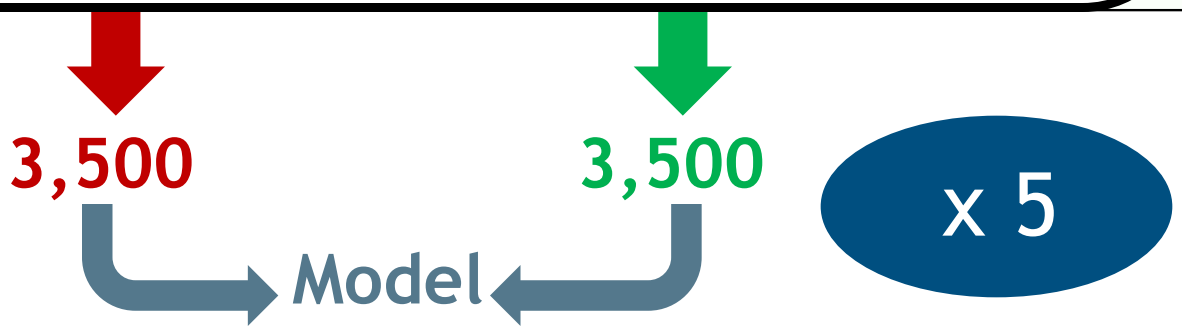
➢Unique copy of the attachment for each recipient

Experimental Data Sets

JS type	Creation	#JS	Label	Obfuscated
Emails				
Microsoft				
Games				
Frameworks				
Atom				

Why these benign samples?

- Does not confound obfuscation with maliciousness
- Nor unseen / unusual syntactic structures with maliciousness



Detection Rate

$$\text{Detection rate} = \frac{TP + TN}{TP + FP + TN + FN}$$

TP: True Positives
TN: True Negatives

FP: False Positives
FN: False Negatives

Detection Rate

JS type	Detection rate
Emails	99.46%
Microsoft	99.50%
Games	99.52%
Frameworks	99.03%
Atom	98.98%
Average benign	99.48%

Detection Rate

JS type	Detection rate
Emails	99.46%
Average benign	99.48%

JaSt shows that:

- Obfuscation ⇏ malicious
- Plain text ⇏ benign

JaSt leverages the differences in the obfuscation process for an accurate detection

Related Work

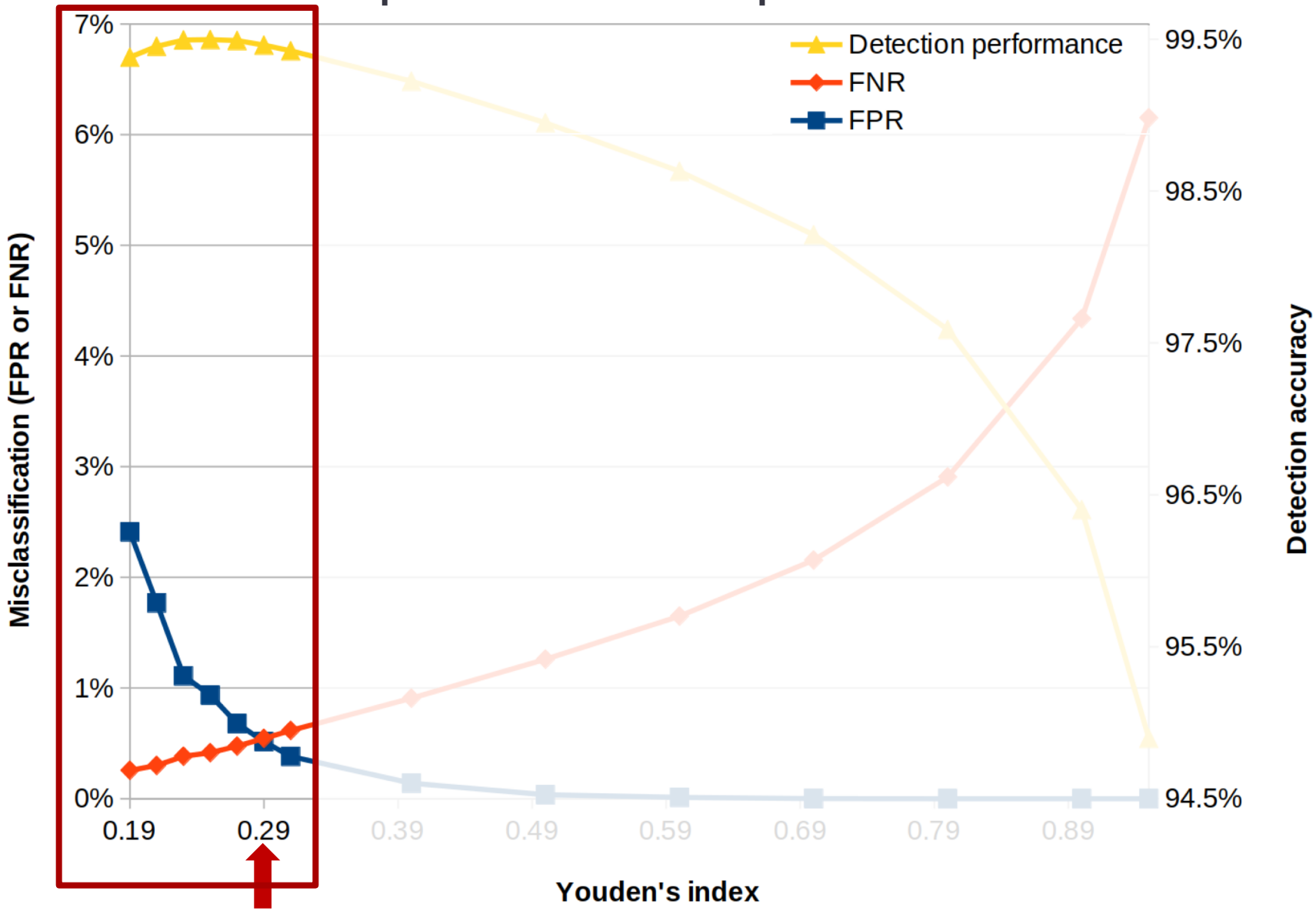
Project	Lexical	AST	Static	Dynamic
JaSt	-	✓	✓	-
Cujo *[Rieck et al.]*	✓	-	✓	✓
PJScan *[Laskov and Šrndić]*	✓	-	✓	-
Zozzle *[Curtsinger et al.]*	-	✓	✓	✓

Comparison with Related Work

Project	FPR	FNR	Static	Dynamic
JaSt	0.52%	0.54%	✓	-
Cujo *[Rieck et al.]*	0.002%	5.6%	✓	✓
PJScan *[Laskov and Šrndić]*	16%	15%	✓	-
Zozzle *[Curtsinger et al.]*	0.00031%	9.2%	✓	✓

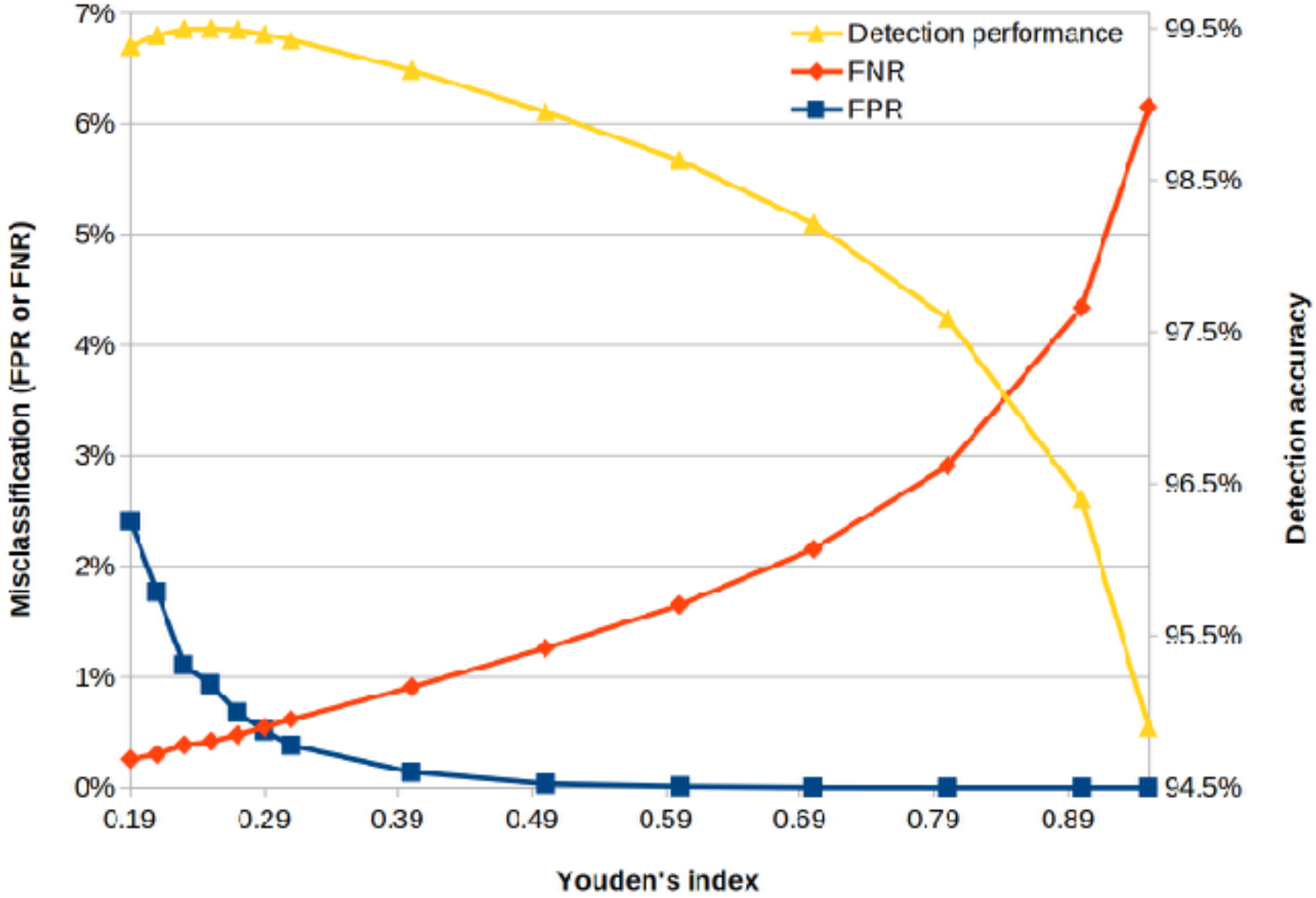
Youden's J Statistic: J = TPR - FPR

- Max value: optimal cut-off point between FP & FN



Youden's J Statistic: J = TPR - FPR

- Max value: optimal cut-off point between FP & FN



Comparison with Related Work

Project	FPR	FNR
JaSt	0.52%	0.54%
Cujo *[Rieck et al.]*	0.002%	5.6%
Zozzle *[Curtsinger et al.]*	0.00031%	9.2%

Index	FPR	FNR
0.7	0.00119%	2.16%
0.8	0	2.91%
0.9	0	4.34%

Comparison with Related Work

Project	FPR	FNR	Classifier	Approach
JaSt	0.52%	0.54%	RF	AST
Cujo *[Rieck et al.]*	0.002%	5.6%	SVM	Lexical
Zozzle *[Curtsinger et al.]*	0.00031%	9.2%	B-NB	AST

Index	FPR	FNR	FPR-Alexa 10k
0.7	0.00119%	2.16%	0.0126%
0.8	0	2.91%	0.00168%
0.9	0	4.34%	0.000671%

Limitations

- Malicious instances with features not in the training set

- Could be impacted by pollution attacks

Further Defenses

- Combining several classifiers
- Number of (different) AST nodes / 4-grams
- 4-grams presence/absence in addition to frequencies

Outlook: Email-Based Model for Web JS

CISPA

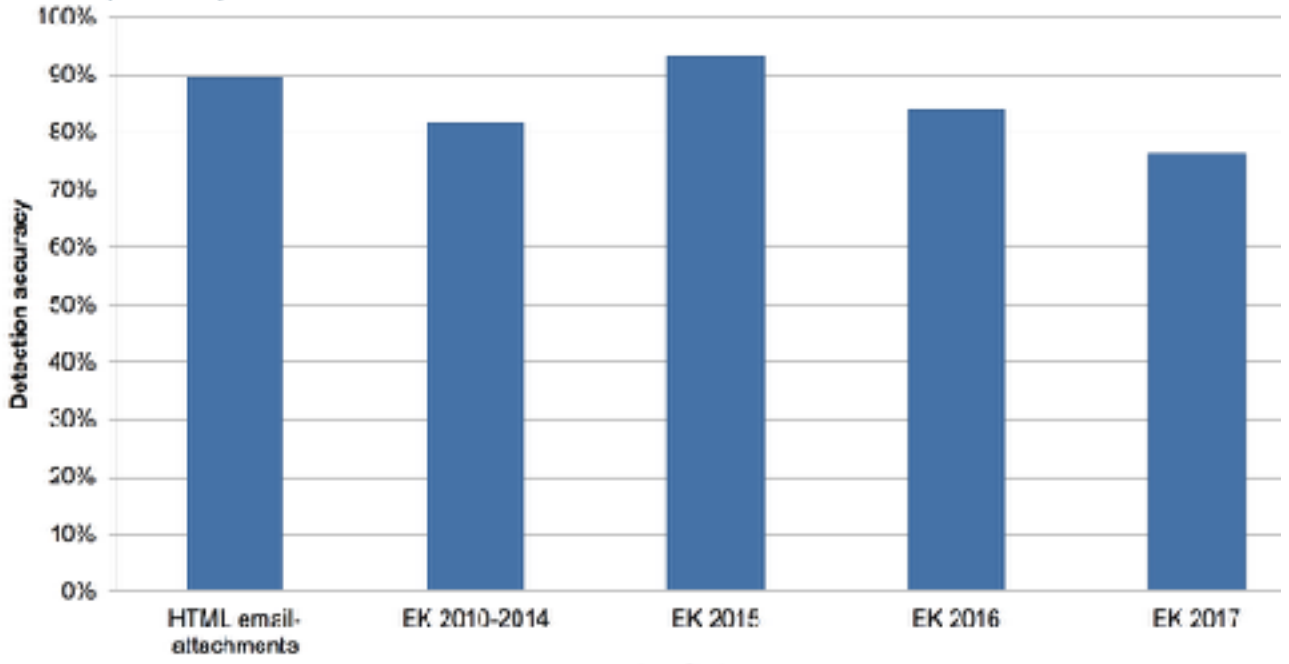
Experimental Data Sets

JS type	Creation	#JS	Label	Obfuscated
Emails	2017-2[illegible]	81,559	Malicious	y
Microsoft	2015-2018	14,168	Benign	y
Games	N/A	2,007	Benign	n
Frameworks	N/A	434	Benign	N/A
Atom	2011-2018	137	Benign	y

3,500 → Model ← 3,500 (x 5)

Detection Rate

JS type	Detection rate
Emails	99.46%
Microsoft	99.50%
Games	99.52%
Frameworks	99.03%
Atom	98.98%
Average benign	99.48%



Outlook: Email-Based Model for Web JS

CISPA

Experimental Data Sets

JS type	Creation	#JS	Label
Emails	2017-2[illegible]	81,559	Malicious
Microsoft	2015-2[illegible]18	14,168	[illegible]ign
Games	N/A	2,007	Ben[illegible]gn
Frameworks	N/A	434	Ben[illegible]gn
Atom	2011-2[illegible]18	137	Ben[illegible]gn

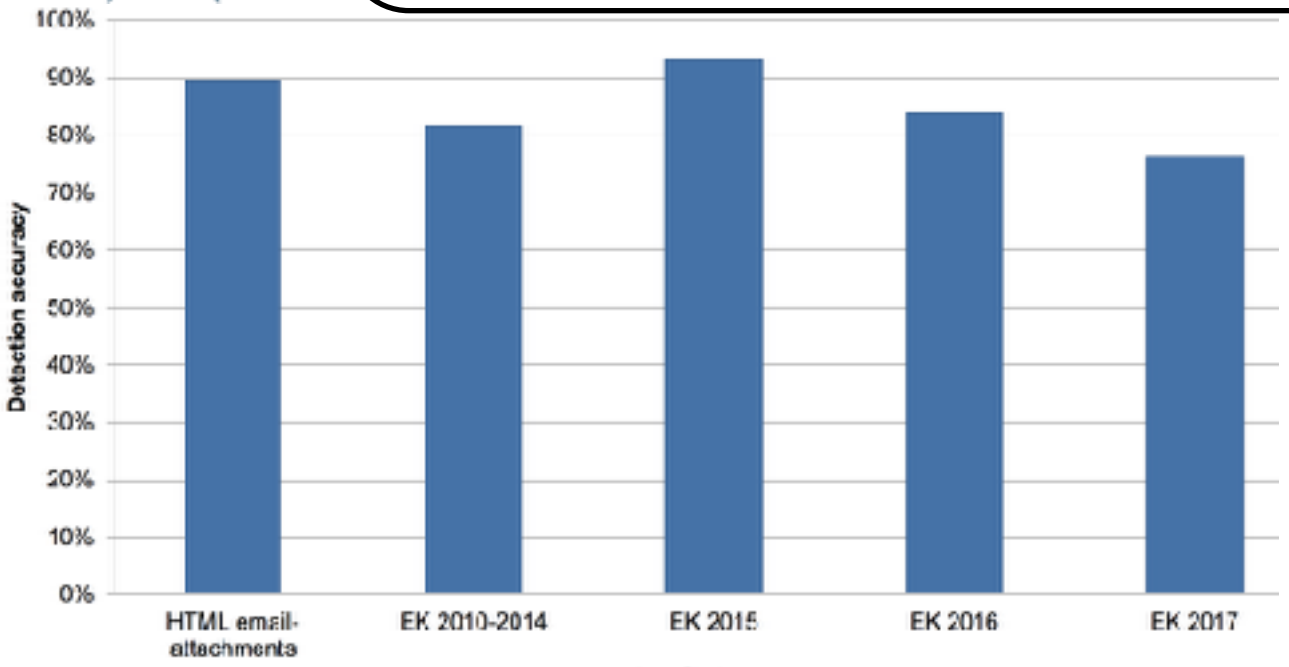
3,500 3,500

Model

HTML document / EK:

➢malicious iff at least one inline JS is classified as malicious

➢benign iff all inline JS are classified as benign



Outlook: Email-Based Model for Web JS

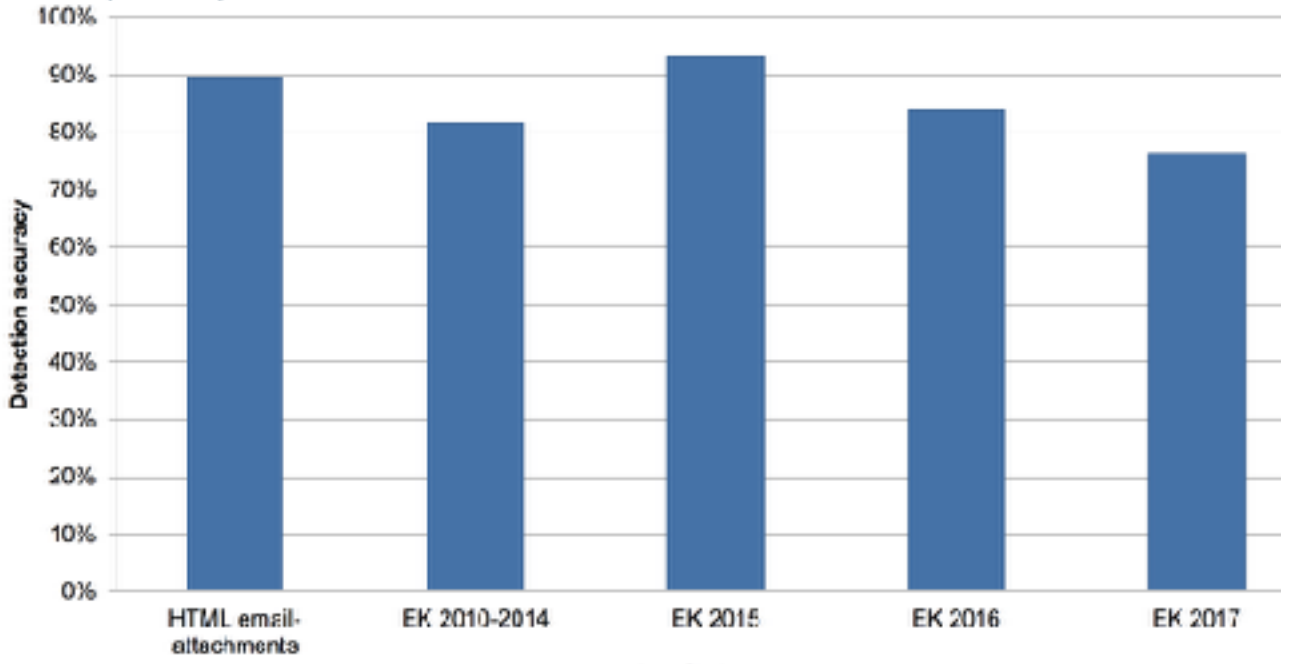
Experimental Data Sets

JS type	Creation	#JS	Label	Obfuscated
Emails	2017-2[illegible]	81,559	Malicious	y
Microsoft	2015-2[illegible]18	14,168	[illegible]ign	y
Games	N/A	2,007	Ben[illegible]gn	n
Frameworks	N/A	434	Ben[illegible]gn	N/A
Atom	2011-2[illegible]18	137	Ben[illegible]gn	y

3,500 → Model ← 3,500 x 5

Detection Rate

JS type	Detection rate
Emails	99.46%
Microsoft	99.50%
Games	99.52%
Frameworks	99.03%
Atom	98.98%
Average benign	99.48%



Outlook: Email-Based Model for Web JS

Email-JS	Web-JS
Unique copy to each recipient	Identify/exploit software vulnerabilities on the client's side
- Variable/function name randomization - Data obfuscation - Encoding obfuscation	- Variable/function name randomization

➢ Syntax features: core in classifying JS

Conclusion

- AST-based analysis to detect malicious JS
  - ➢ Extraction of syntactic units
  - ➢ Frequency analysis of 4-grams
  - ➢ Learning and classification with random forest
  - ➢ Performance evaluation
    - ✓ Accuracy of the predictions: 99.46%
    - ✓ Low false negative rate: 0.54%
    - ✓ Resilience to common obfuscation transformations
    - ✓ Resilience to environment-dependent malware

Thank you

Fass et al. JaSt: Fully Syntactic Detection of Malicious (Obfuscated) JavaScript. In *DIMVA* 2018

https://github.com/Aurore54F/JaSt @AuroreFass