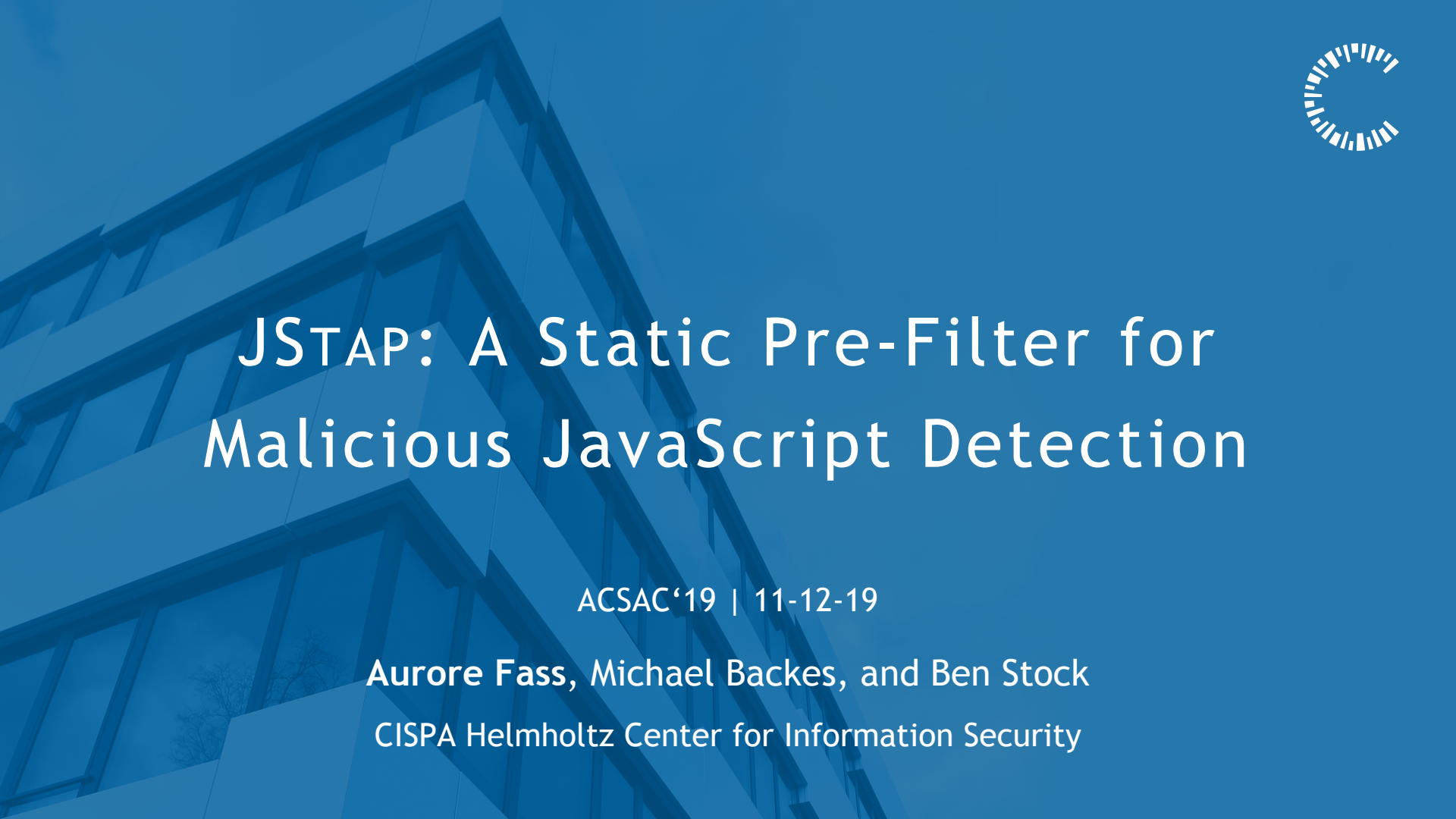


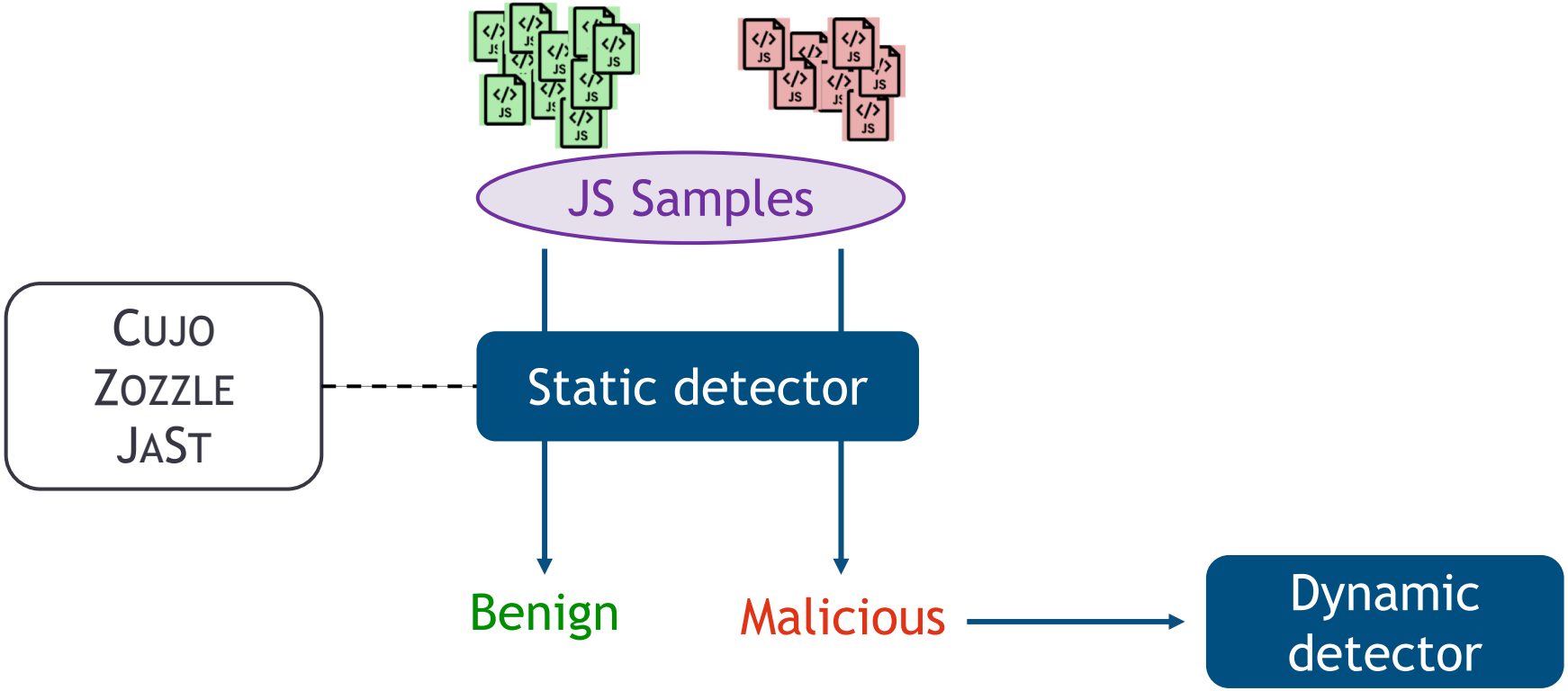
JSTAP: A Static Pre-Filter for Malicious JavaScript Detection

ACSAC'19 | 11-12-19

Aurore Fass, Michael Backes, and Ben Stock

CISPA Helmholtz Center for Information Security

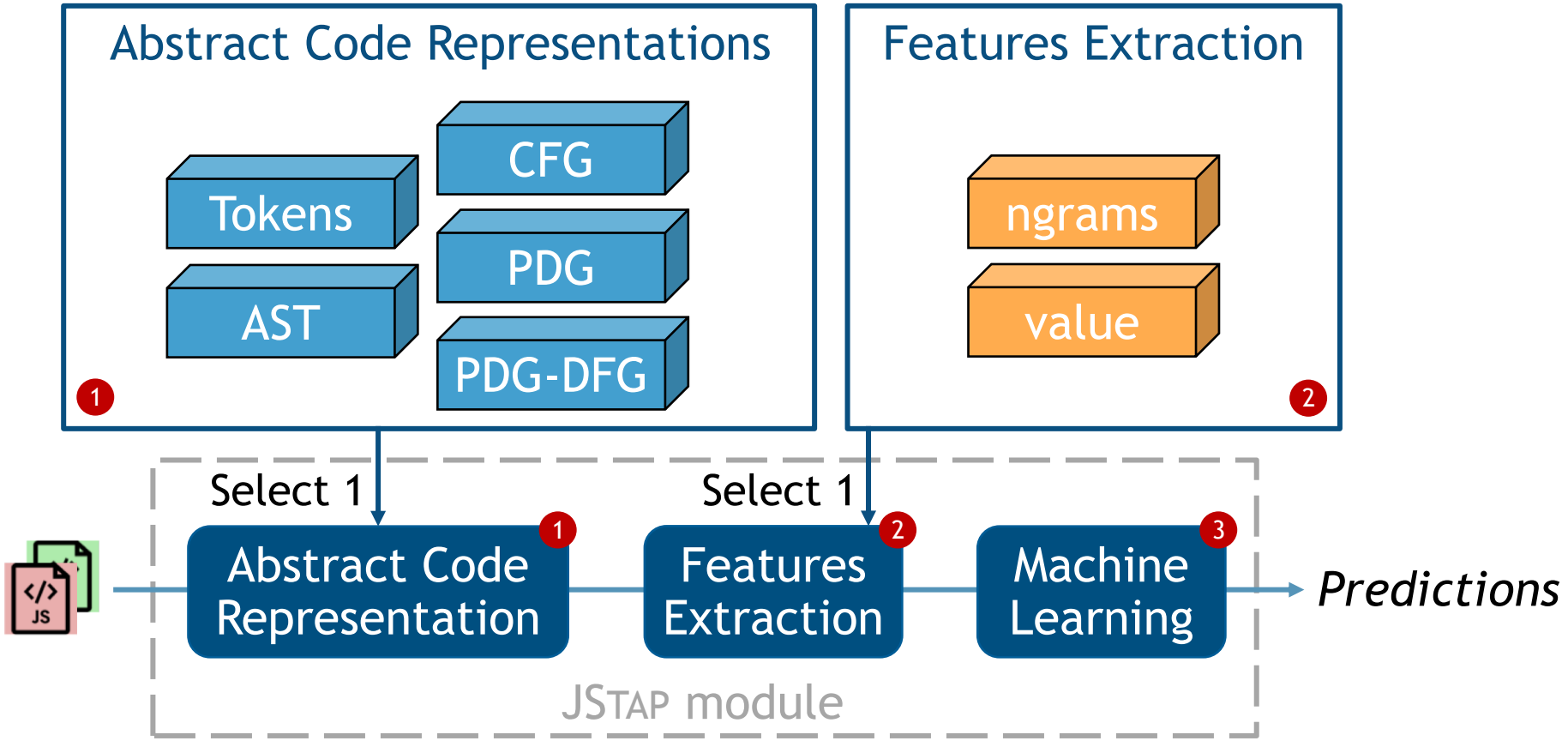
Motivation


JS Samples
CUJO
ZOZZLE
JAST
Static detector
Benign
Malicious
Dynamic detector

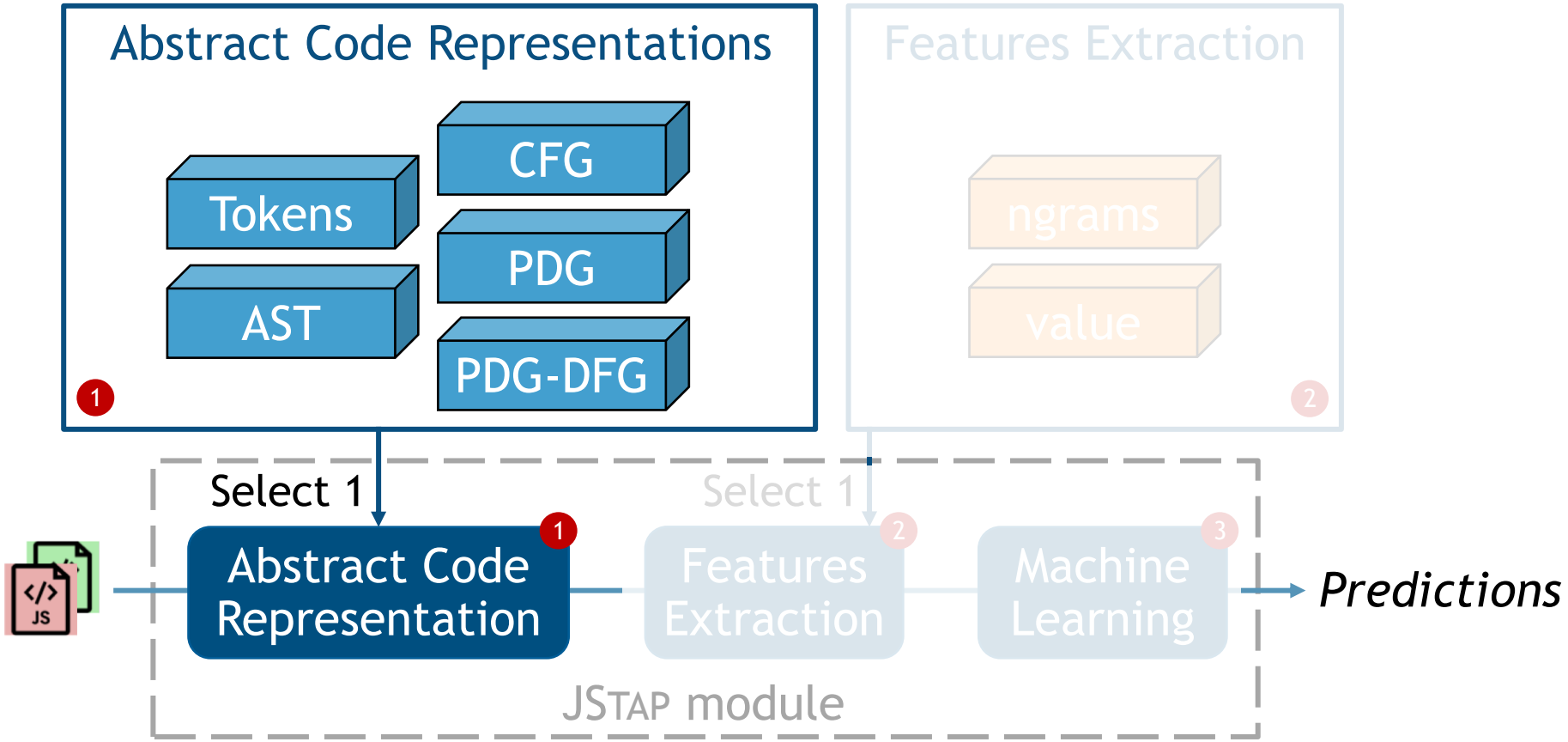
Contributions

- Entirely static analysis with additional semantic information:
  - Control flow: reasons about conditions
  - Data flow: reasons about execution order
  - ➢ Added value?
- Are prior works still able to detect current malicious JS?
- Which static code abstraction performs best?

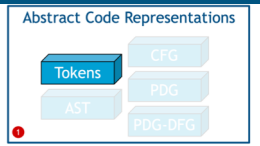
JSTAP Overview



JSTAP Overview



Abstract Code Representation: Tokens

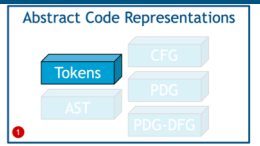


CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

1. `x.if = 1;`
2. `var y = 1;`
3. `if (x.if == 1) {d = y;}`

- Tokens: linear conversion into abstract symbols

Abstract Code Representation: Tokens



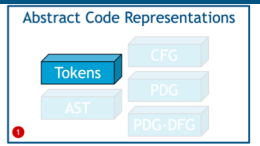
CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

1. `x.if`
2. `var y = 1;`
3. `if (x.if == 1) {d = y;}`

1. `Identifier Punctuator Keyword`

- Tokens: linear conversion into abstract symbols

Abstract Code Representation: Tokens



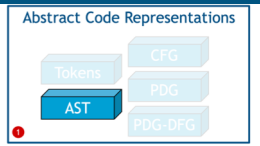
CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

1. `x.if = 1;`
2. `var y = 1;`
3. `if (x.if == 1) {d = y;}`

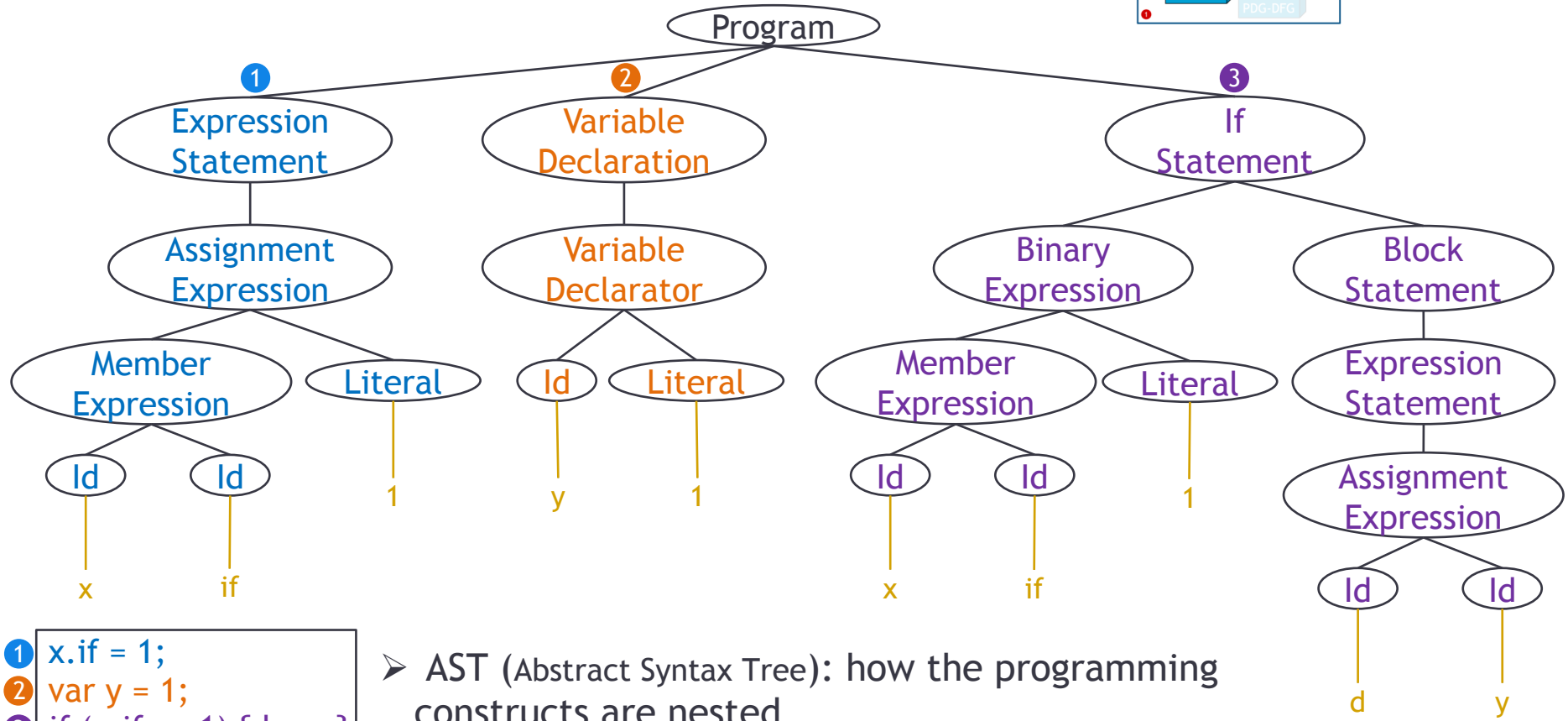
1. `Identifier Punctuator Keyword Punctuator Numeric Punctuator`
2. `Keyword Identifier Punctuator Numeric Punctuator`
3. `Keyword Punctuator Identifier Punctuator Keyword Punctuator Numeric Punctuator Punctuator Identifier Punctuator Identifier Punctuator Punctuator`

➢ Tokens: linear conversion into abstract symbols

Abstract Code Representation: AST



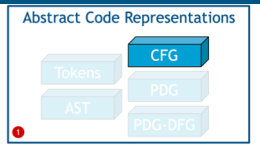
CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY



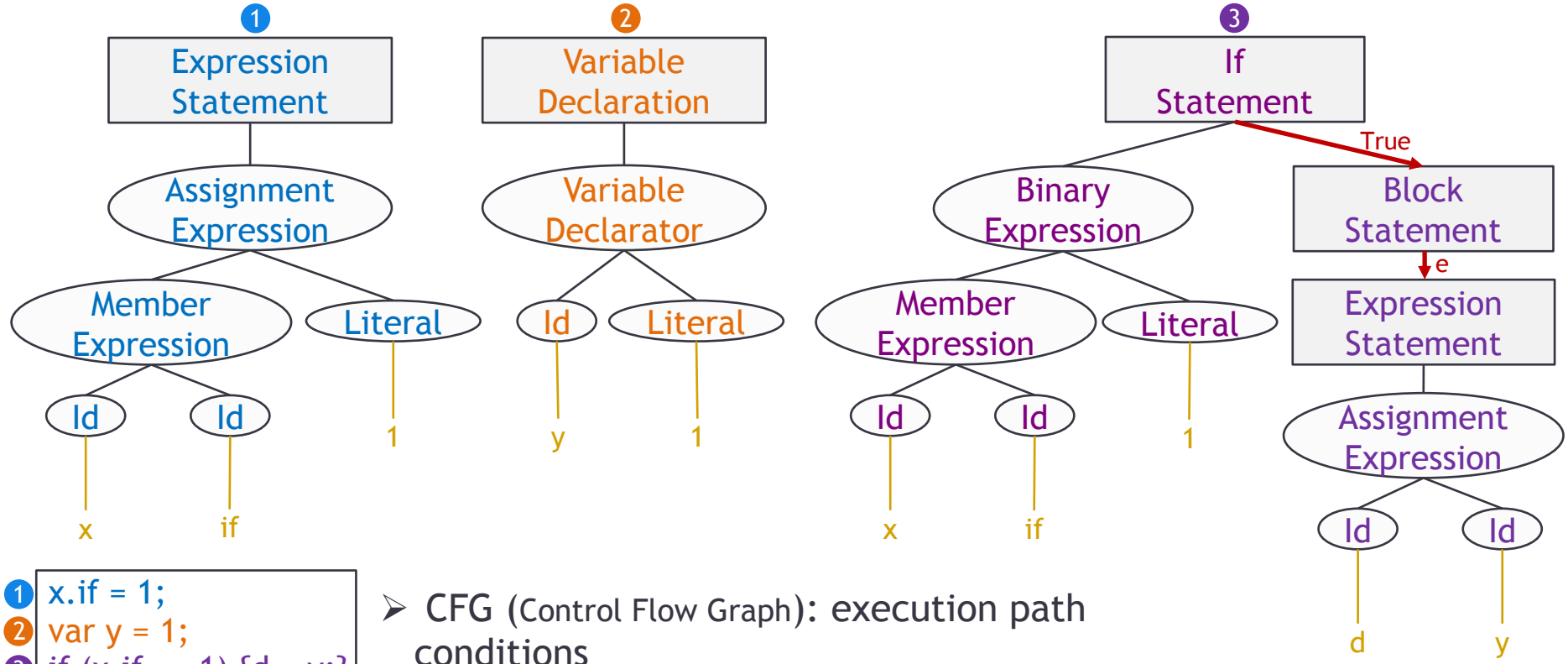
1. `x.if = 1;`
2. `var y = 1;`
3. `if (x.if == 1) {d = y;}`

- AST (Abstract Syntax Tree): how the programming constructs are nested

Abstract Code Representation: CFG



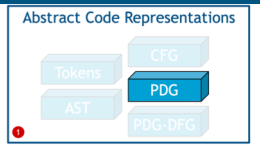
CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY




```
1 x.if = 1;
2 var y = 1;
3 if (x.if == 1) {d = y;}
```

- CFG (Control Flow Graph): execution path conditions

Abstract Code Representation: PDG



CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

1 Expression Statement
Assignment Expression
Member Expression
Literal
Id
Id
1
x
if

2 Variable Declaration
Variable Declarator
Id
Literal
y
1

3 If Statement
True
Binary Expression
Member Expression
Literal
Id
Id
1
x
if

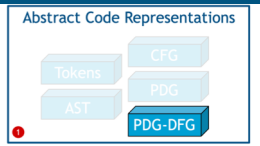
Block Statement
e
Expression Statement
Assignment Expression
Id
Id
d
y

data

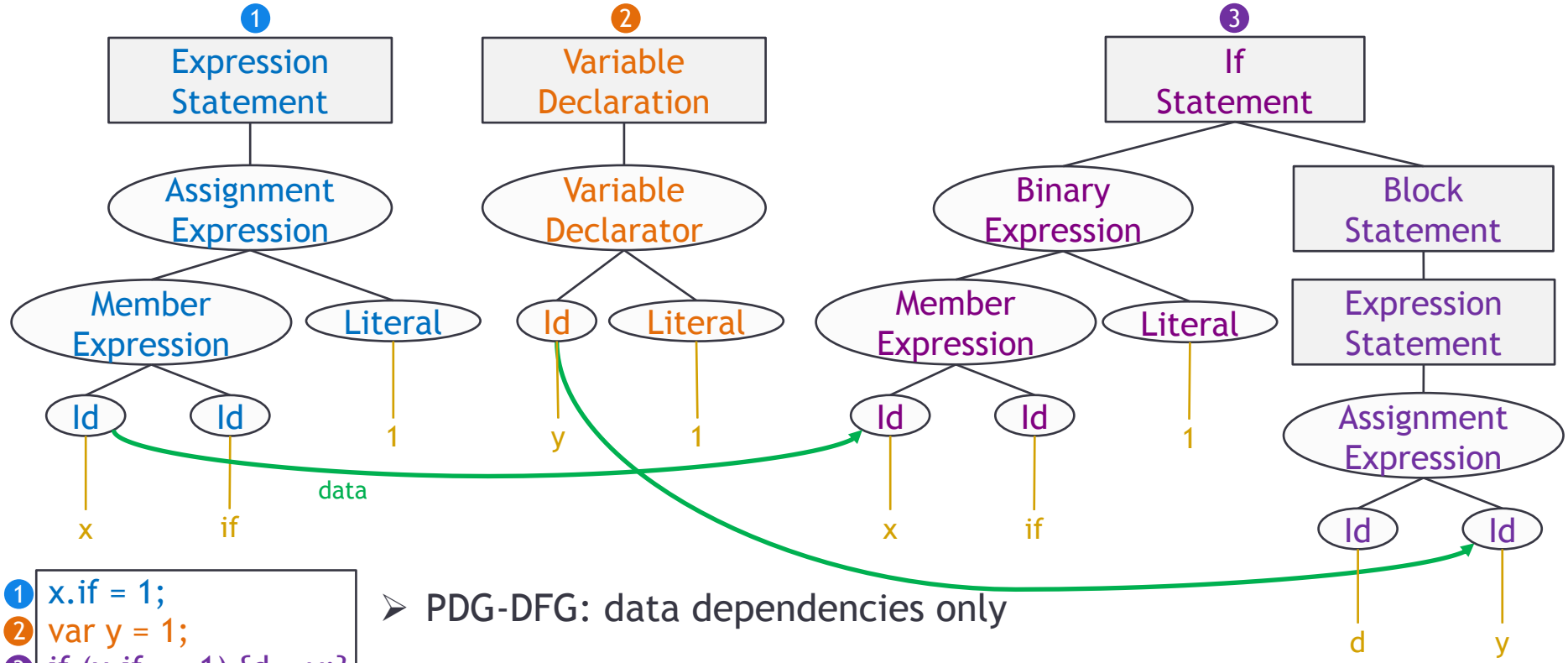
1 x.if = 1;
2 var y = 1;
3 if (x.if == 1) {d = y;}

- PDG (Program Dependency Graph): control flow + data dependencies

Abstract Code Representation: PDG-DFG



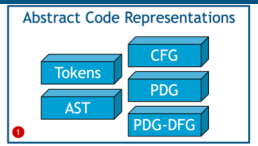
CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY



1 x.if = 1;
2 var y = 1;
3 if (x.if == 1) {d = y;}

- PDG-DFG: data dependencies only

Abstract Code Representations: Summary



CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

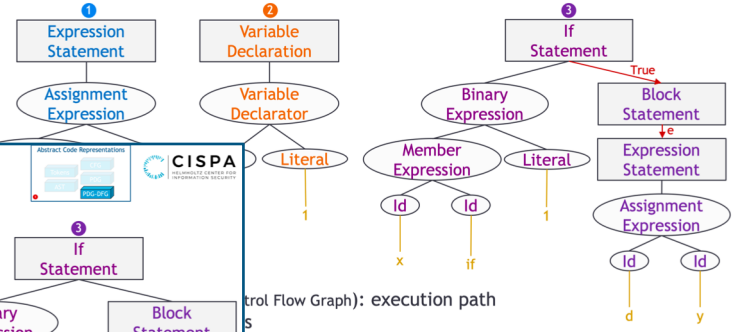
Abstract Code Representation: Tokens

1. x.if = 1;
2. var y = 1;
3. if (x.if == 1) {d = y;}

1 Identifier Punctuator Keyword Punctuator Numeric P
2 Keyword Identifier Punctuator Numeric Punctuator
3 Keyword Punctuator Identifier Punctuator Keyword P
Punctuator Identifier Punctuator Identifier Punctu

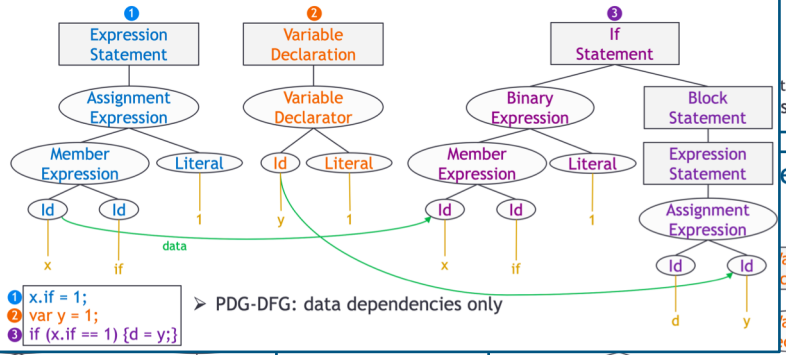
➢ Tokens: linear conversion into abs

Abstract Code Representation: CFG



trol Flow Graph): execution path
s

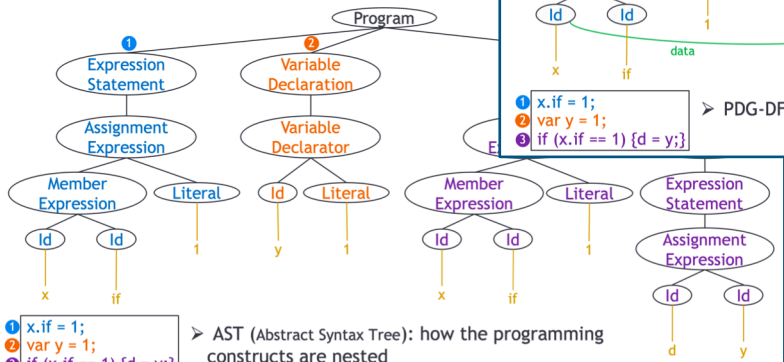
Abstract Code Representation: PDG-DFG



1 x.if = 1;
2 var y = 1;
3 if (x.if == 1) {d = y;}

➢ PDG-DFG: data dependencies only

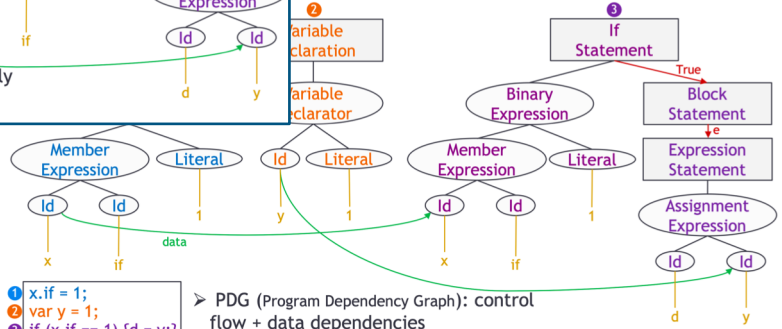
Abstract Code Representation: AST



1 x.if = 1;
2 var y = 1;
3 if (x.if == 1) {d = y;}

➢ AST (Abstract Syntax Tree): how the programming constructs are nested

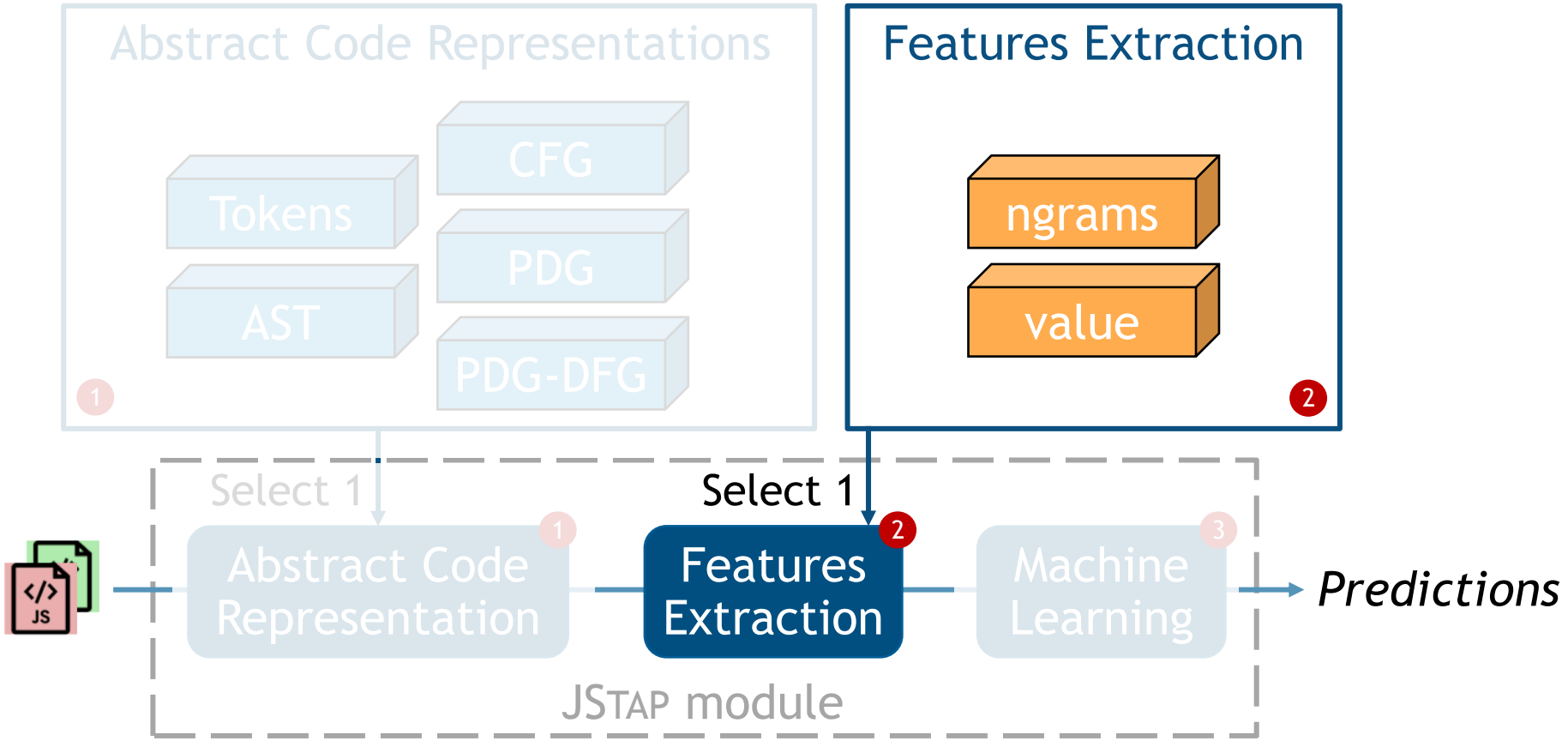
entation: PDG



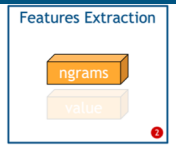
1 x.if = 1;
2 var y = 1;
3 if (x.if == 1) {d = y;}

➢ PDG (Program Dependency Graph): control flow + data dependencies

JSTAP Overview



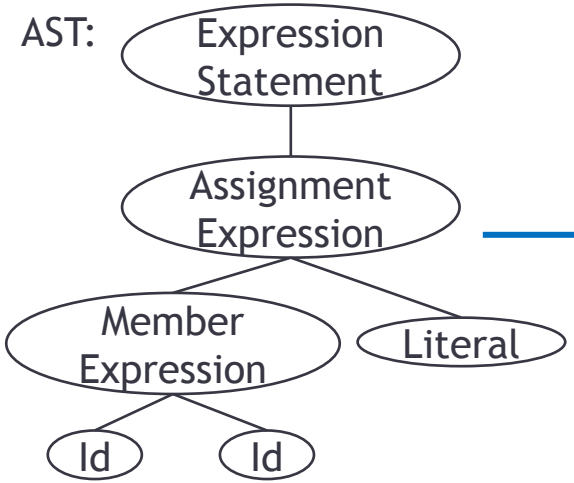
Features Extraction: ngrams



CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

1. Select an abstract code representation
2. Traverse it and extract the corresponding units
3. Combine these units in groups of n = build n-grams

AST:



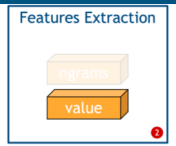
ExpressionStatement
AssignmentExpression
MemberExpression
Id
Id
Literal

ExpressionStatement
AssignmentExpression
MemberExpression
Id

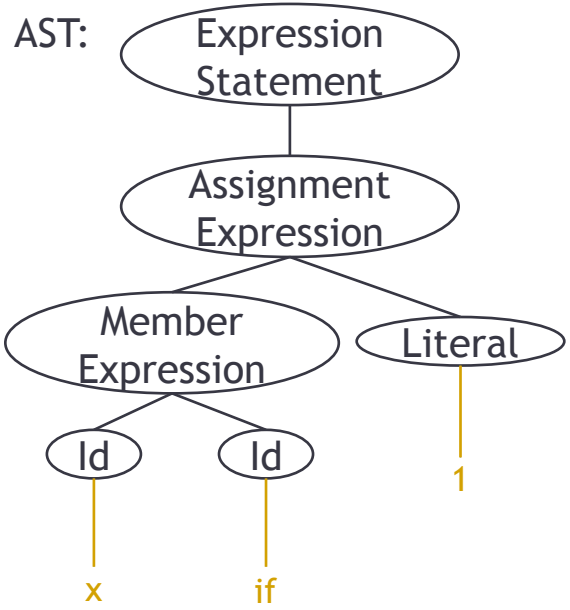
AssignmentExpression
MemberExpression
Id
Id

MemberExpression
Id
Id
Literal

Features Extraction: value

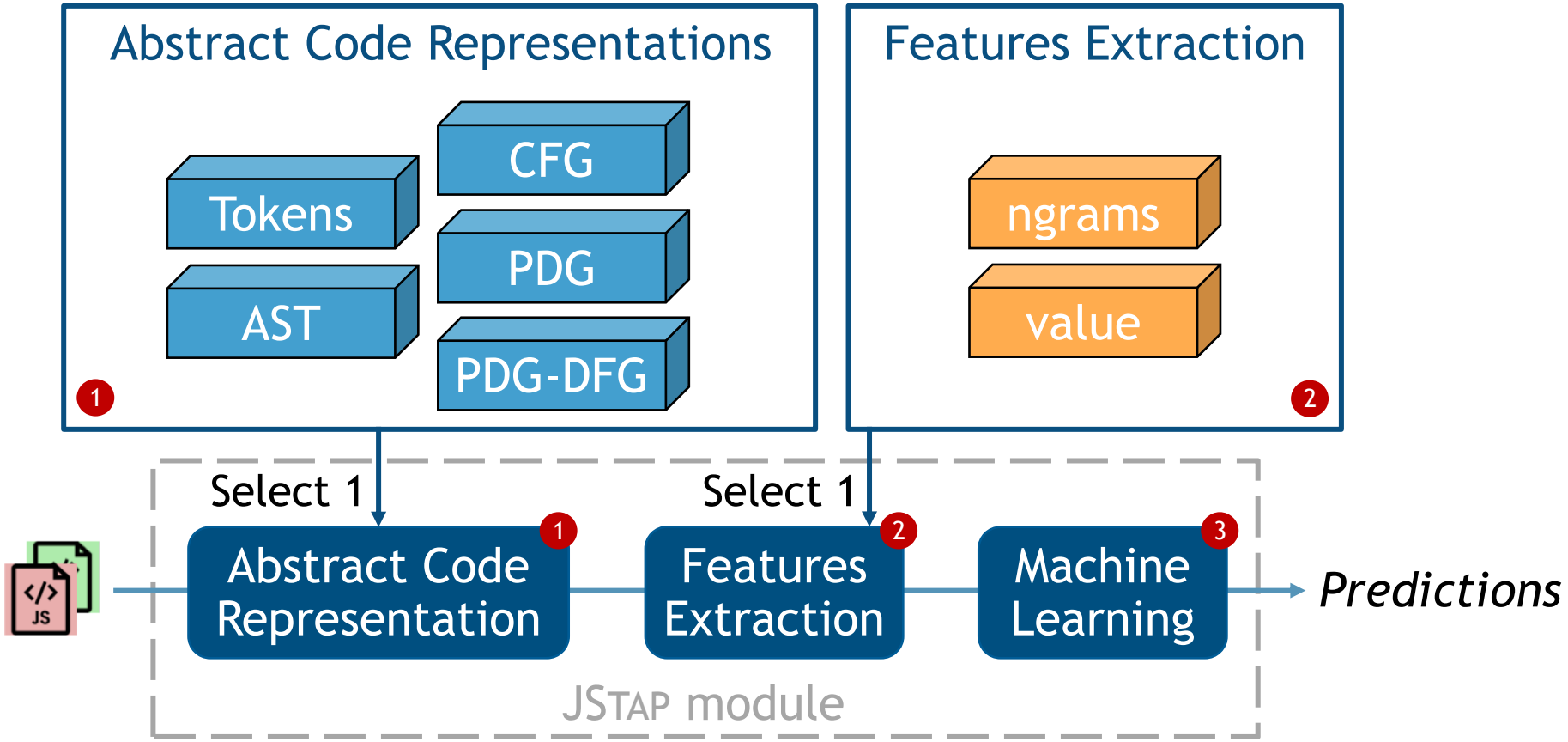


1. Select an abstract code representation
2. Traverse it and extract the corresponding units along with their value



(ExpressionStatement, x)
(AssignmentExpression, x)
(MemberExpression, x)
(Id, x)
(Id, if)
(Int, 1)

JSTAP Overview

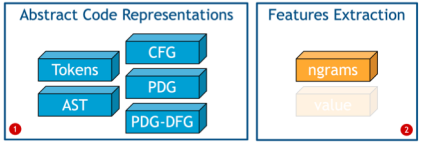


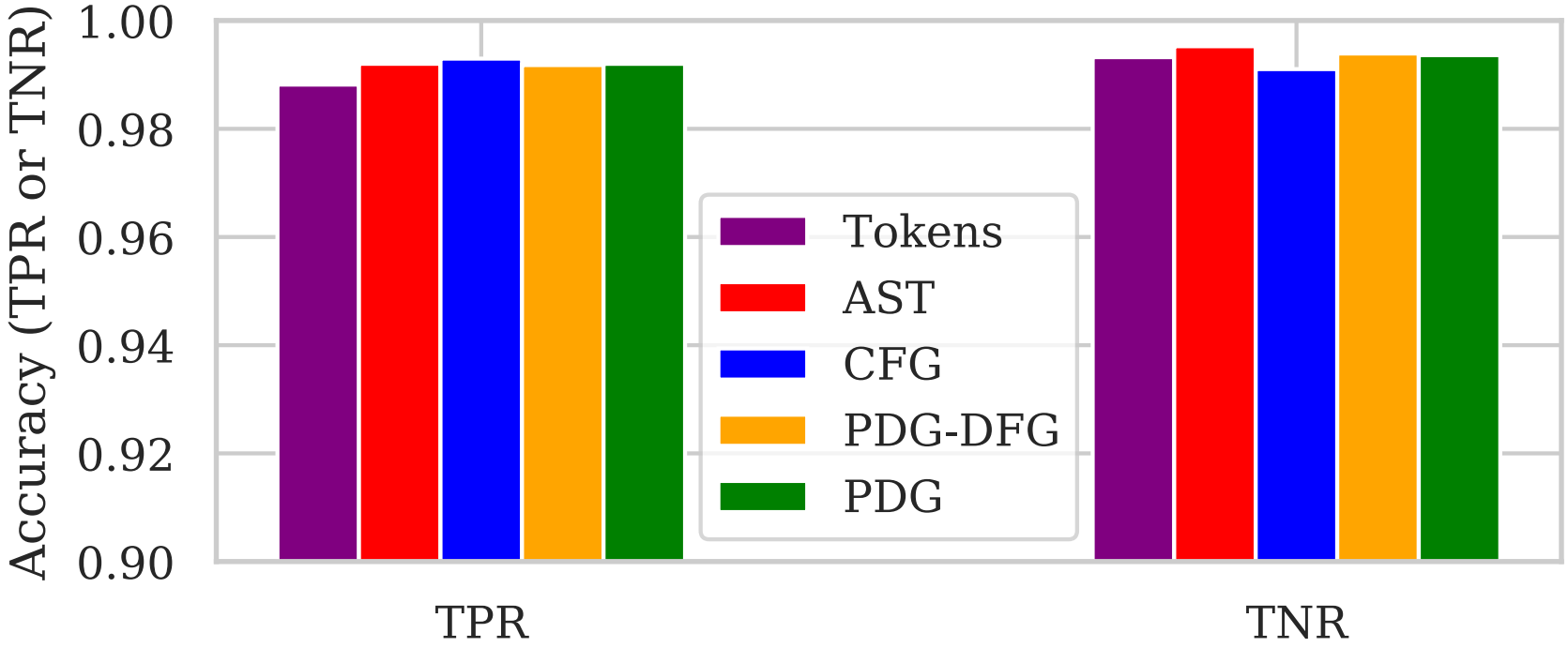
Experimental Setup

Malicious	
Source	#JS
BSI	83,361
Hynek Petrak	29,558
Kafeine DNC	12,982
VirusTotal	3,056
GeeksOnSecurity	2,491
Total	131,448

Benign	
Source	#JS
Tranco-10k	122,910
Microsoft	16,271
Games	1,992
Web Frameworks	427
Atom	168
Total	141,768

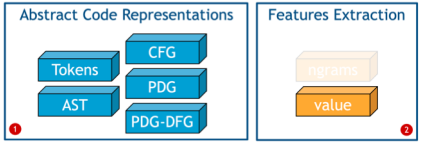
Detection Performance: ngrams

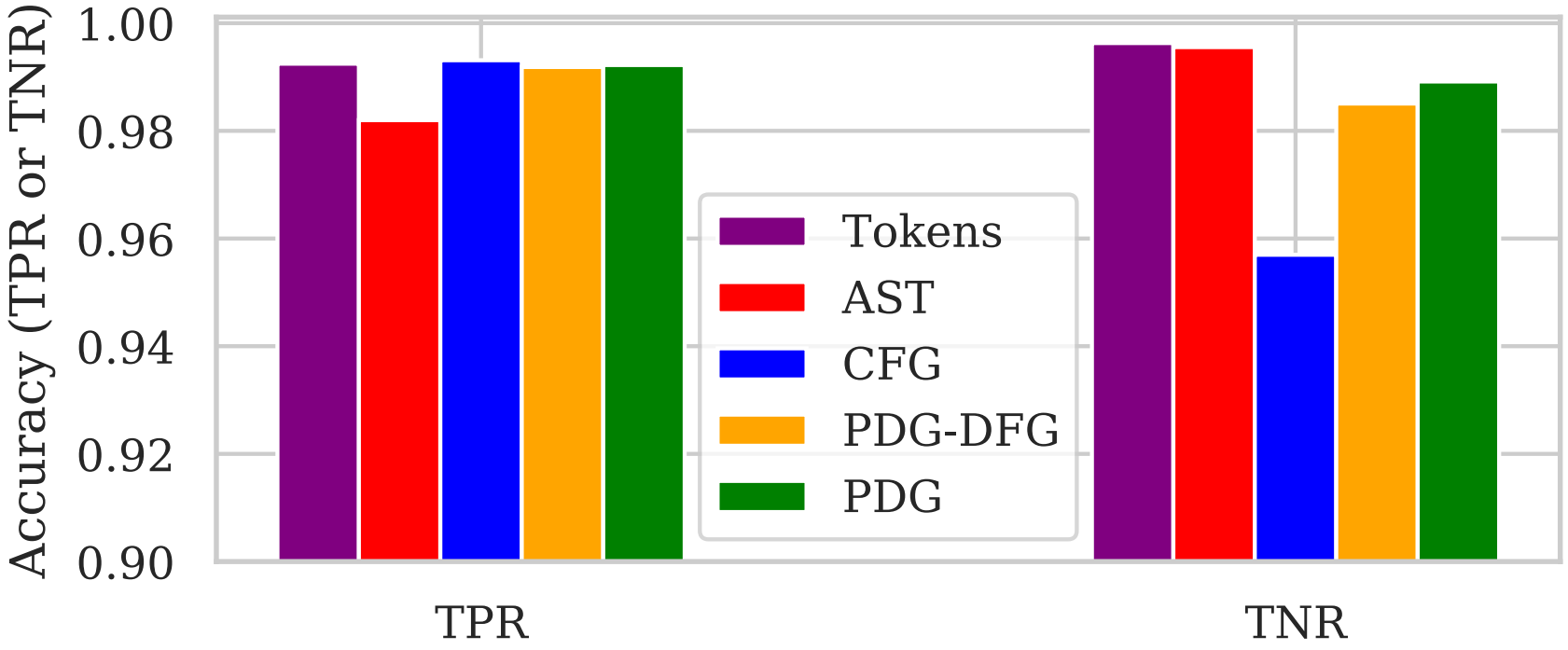




- **Best one: AST with 99.38% correct classifications**

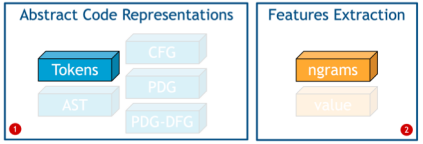
Detection Performance: value

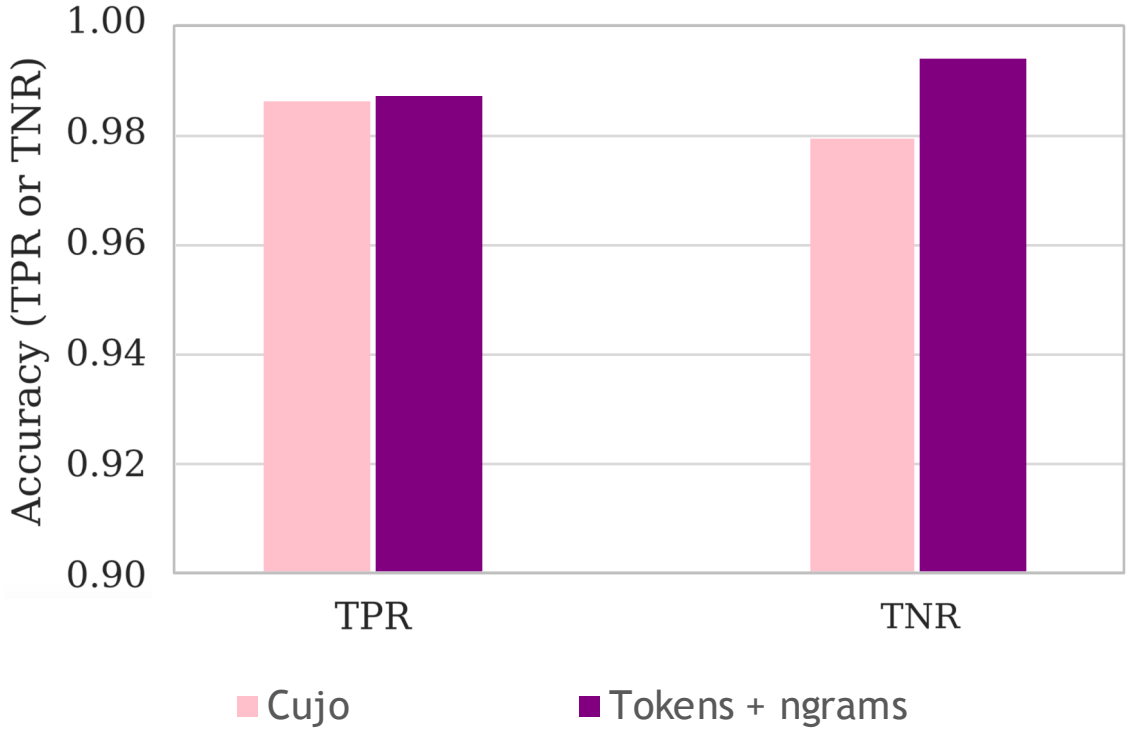




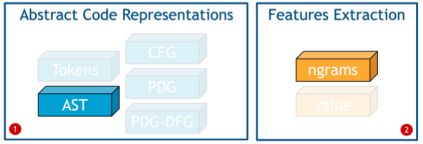
- **Best one: Tokens with 99.44% correct classifications**

Related Work Comparison: CUJO

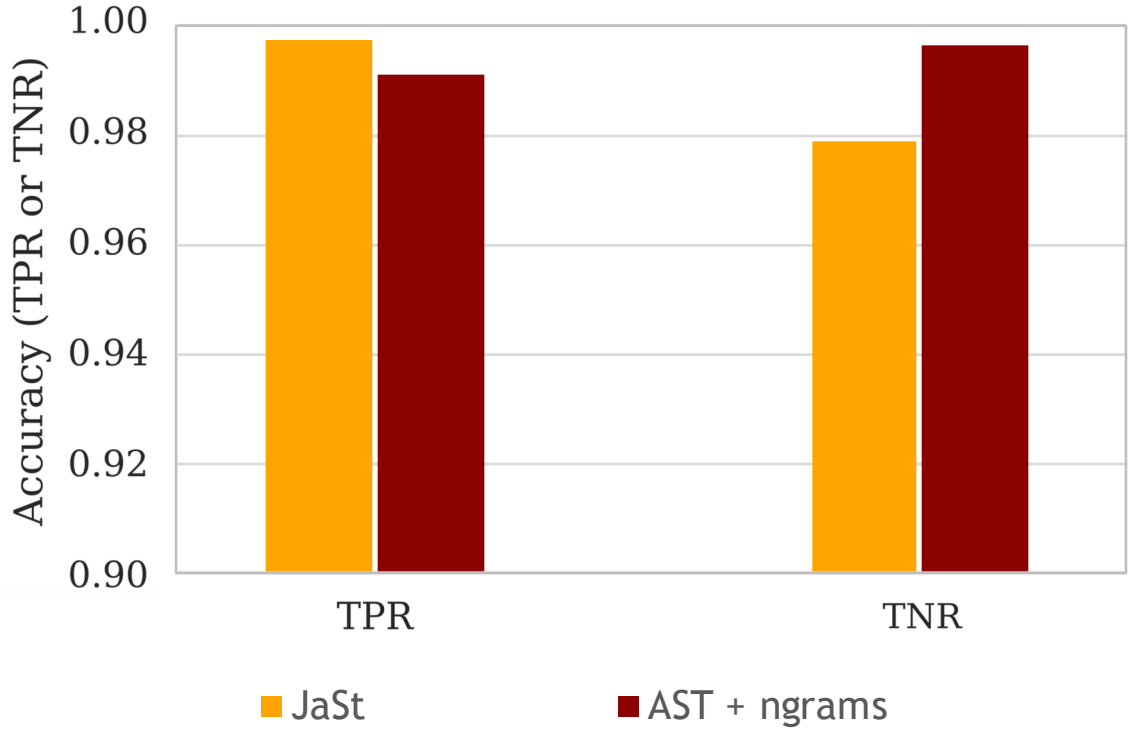




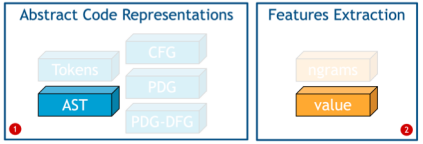
Related Work Comparison: JAST

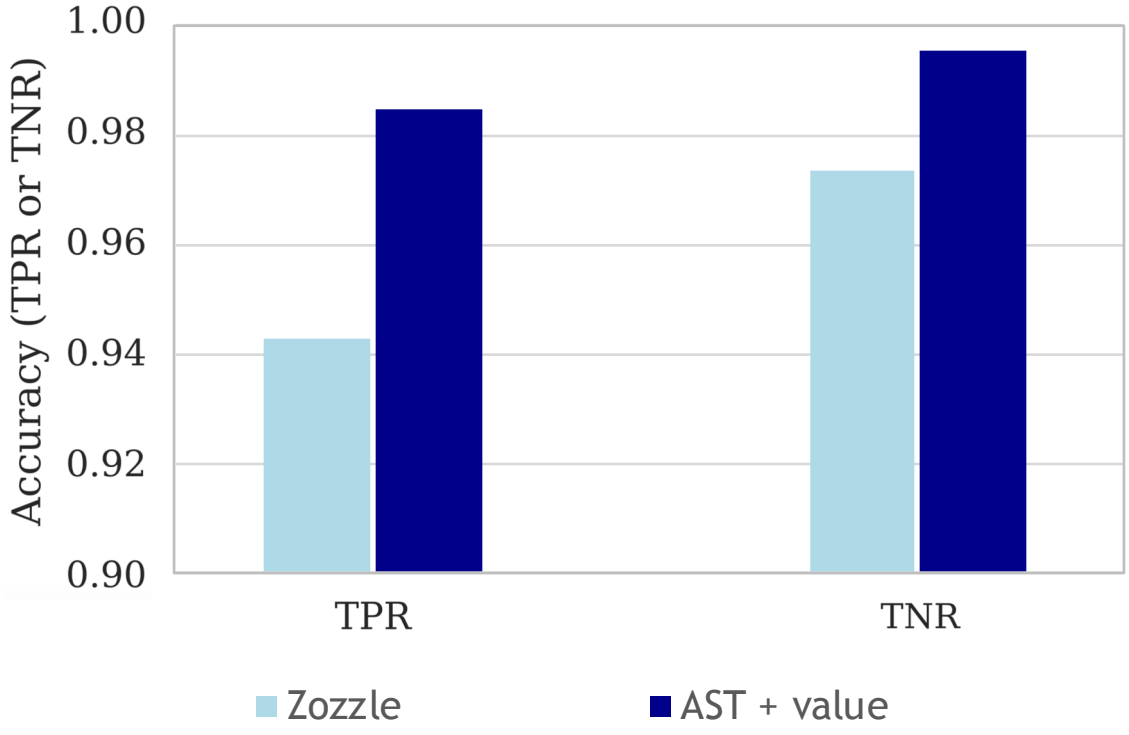


CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

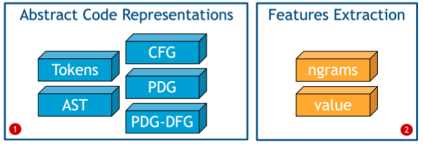


Related Work Comparison: Zozzle





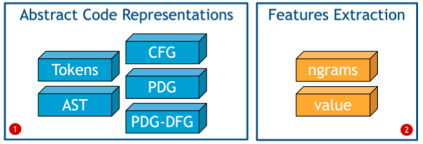
Modules Combination

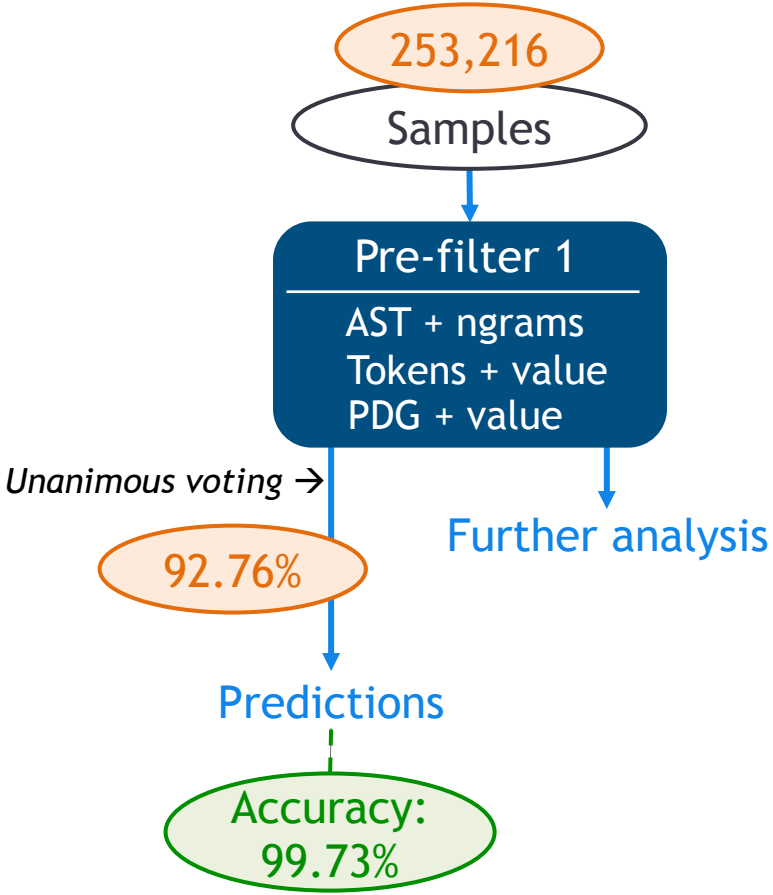


Pre-filter 1

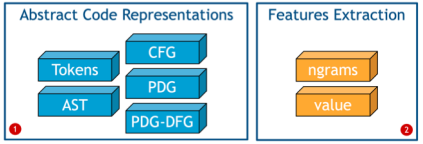
AST + ngrams
Tokens + value
PDG + value

Modules Combination





Modules Combination



CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

253,216
Samples

Pre-filter 1
AST + ngrams
Tokens + value
PDG + value

Unanimous voting →

92.76%

Predictions

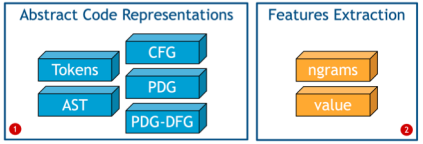
Accuracy:
99.73%

Further analysis

7.24%

Pre-filter 2
Tokens + ngrams
AST + ngrams
AST + value

Modules Combination



CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

253,216
Samples

Pre-filter 1
AST + ngrams
Tokens + value
PDG + value

Unanimous voting →

92.76%

Predictions

Accuracy: 99.73%

Further analysis

7.24%

Pre-filter 2
Tokens + ngrams
AST + ngrams
AST + value

Unanimous voting →

6.5%

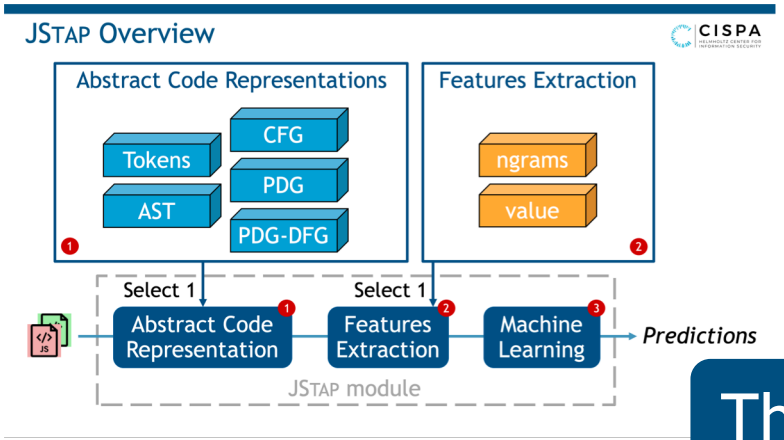
Predictions

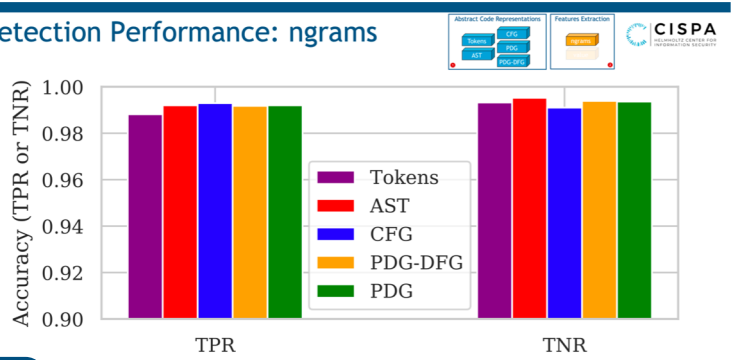
Accuracy: 99%

0.74%

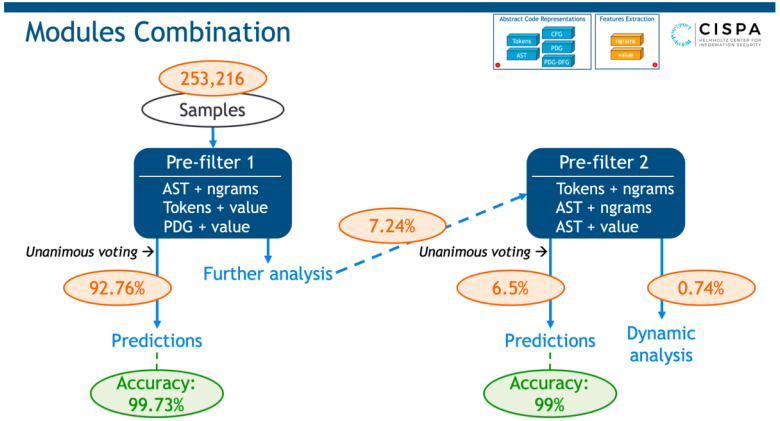
Dynamic analysis

Conclusion





Thank you



JSTAP: A Static Pre-Filter for Malicious JavaScript Detection

Aurore Fass, Michael Backes, and Ben Stock
CISPA Helmholtz Center for Information Security
{aurore.fass,backes,stock}@cispa.saarland

Artifacts Evaluated – acm – Reusable

Aurore54F/JStap

@AuroreFass @kcotsneb