Semantics-Aware Malware Detection

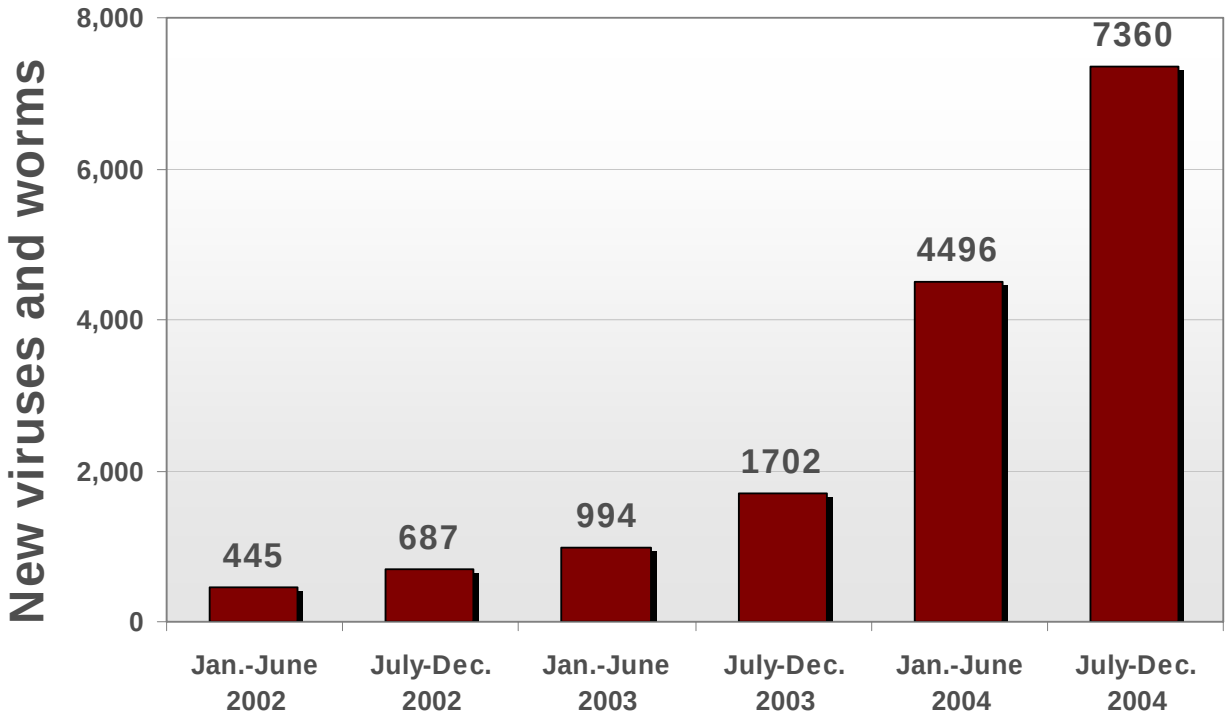
Mihai Christodorescu, Somesh Jha
University of Wisconsin, Madison

Sanjit Seshia, Dawn Song, Randal Bryant
Carnegie Mellon University

Malicious Code Problem

Malware is everywhere.

Source: Symantec Internet Security Threat Report (vol. VII)



- Large malware families.

Evasion Techniques

- Obfuscations applied to malicious code make evasion very easy.

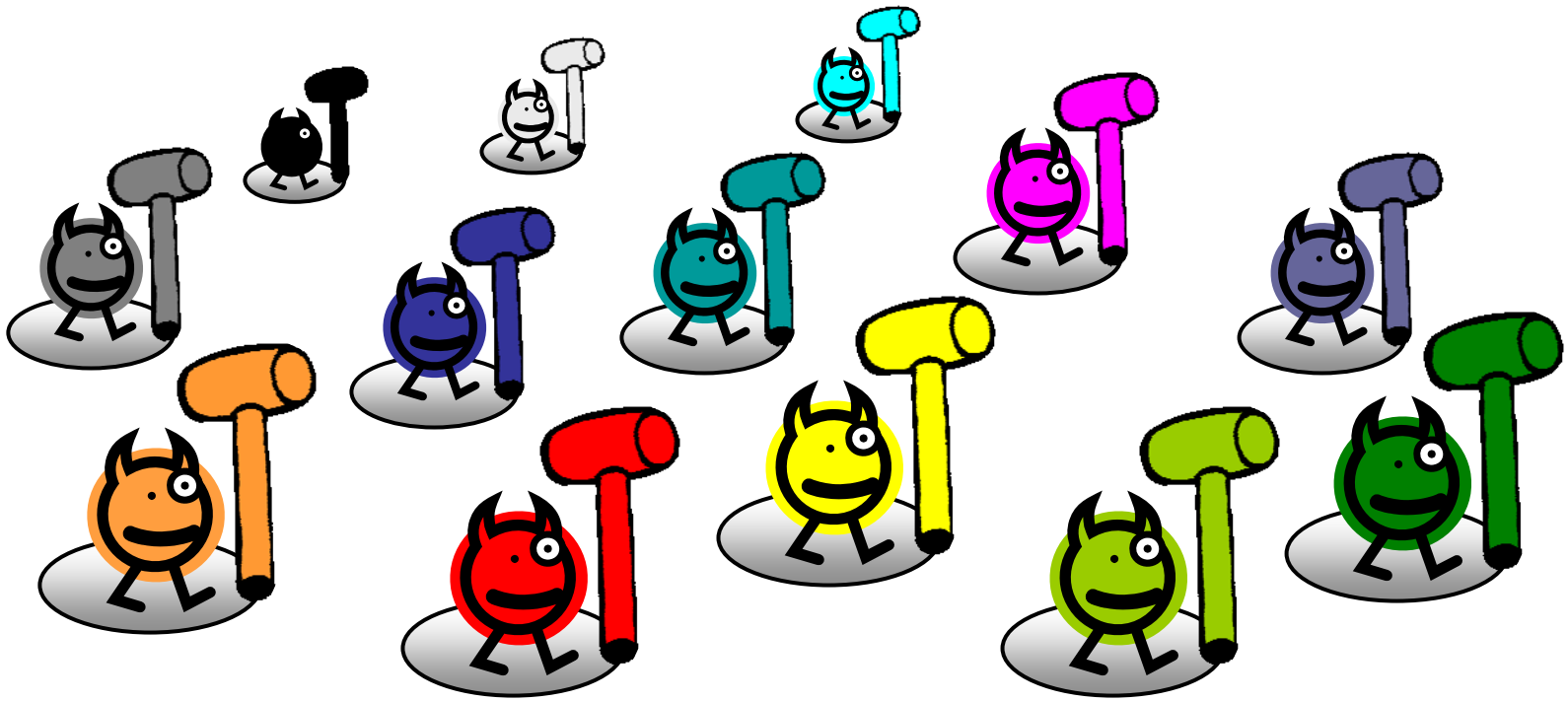
- Attacker's goal:

  Preserve (subset of) behavior.
  - Transformations of code and data.
  - Addition of new code and data.

- Easy to create variants in large numbers.

The Current Solution

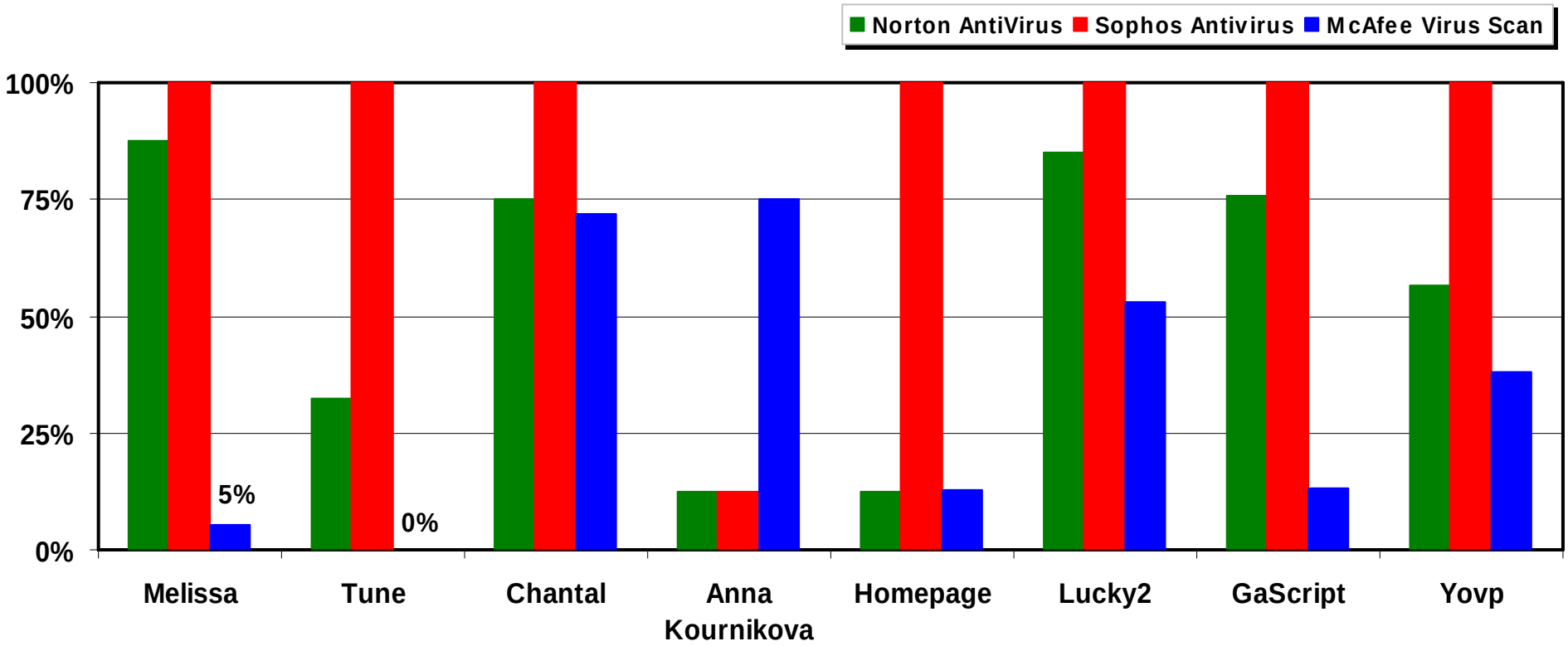
- Syntactic signatures are insufficient.



- Overfitting, easy to evade.

No Resilience to Obfuscations

False Negative Rate for Obfuscated Worms



Source: "Testing Malware Detectors" (ISSTA 2004)

Our Contributions

- Introduce semantic signatures that combine syntactic and semantic information.
- Develop a prototype for malware detection using semantic signatures.
- Show, empirically, that one semantic signature can detect a malware family.

An Example

Goal: detect any mass-mailing virus.

1. Detect email capabilities.
2. Detect self-propagation.

```
s = socket( ... );
connect( s );
...
sprintf( buf, “EHLO %s”, dnsname );
send( s, buf );
```

Knkz~on#pyw #W ¥N yyw 8[

Possible syntactic signature:

```
socket()
connect()


“EHLO”
send()
```

Variant 1: String Manipulation

- Hide known constants from virus scanner.
- Syntactic signature does not match.

```
s = socket( ... );
connect( s );
...
sprintf( buf, “EHLO %s”, dnsname );
send( s, buf );
```

Knkz-on#pyw #W ¥N yyw 8V

Variant 1: String Manipulation

- Hide known constants from virus scanner.
- Syntactic signature does not match.

```
s = socket( ... );
connect( s );
...
sprintf( buf, “E%s %s”, “HLO”, dnsname );
send( s, buf );
```

Knkz~on#plyw #W ¥N yyw 8V

Variant 1: String Manipulation

- Hide known constants from virus scanner.
- Syntactic signature does not match.

```
s = socket( ... );
connect( s );
...
sprintf( buf, “E%s %s”, “HLO”, dnsname
send( s, buf );
```

Syntactic signature:

```
socket()
connect()
“EHLO”
send()
```

Knkz~on#plyw #W ¥N yyw 8V

Variant 2: String Obfuscation

- Hide known constants using simple encryption techniques (e.g. ROT13, XOR).
- Syntactic signature does not match.

```
s = socket( ... );
connect( s );
...
sprintf( buf, “EHLO %s”, dnsname );
send( s, buf );
```

Variant 2: String Obfuscation

- Hide known constants using simple encryption techniques (e.g. ROT13, XOR).
- Syntactic signature does not match.

```
s = socket( ... );
connect( s );
...
cmdbuf = rot13( “URYB %f” );
sprintf( buf, cmdbuf, dnsname );
send( s, buf );
```

Knkz-on#pyw #W ¥N yyw 8Q

Variant 2: String Obfuscation

- Hide known constants using simple encryption techniques (e.g. ROT13, XOR).
- Syntactic signature does not match.

```
s = socket( ... );
connect( s );
...
cmdbuf = rot13( “URYB %f” );
sprintf( buf, cmdbuf, dnsname );
send( s, buf );
```

Syntactic signature:

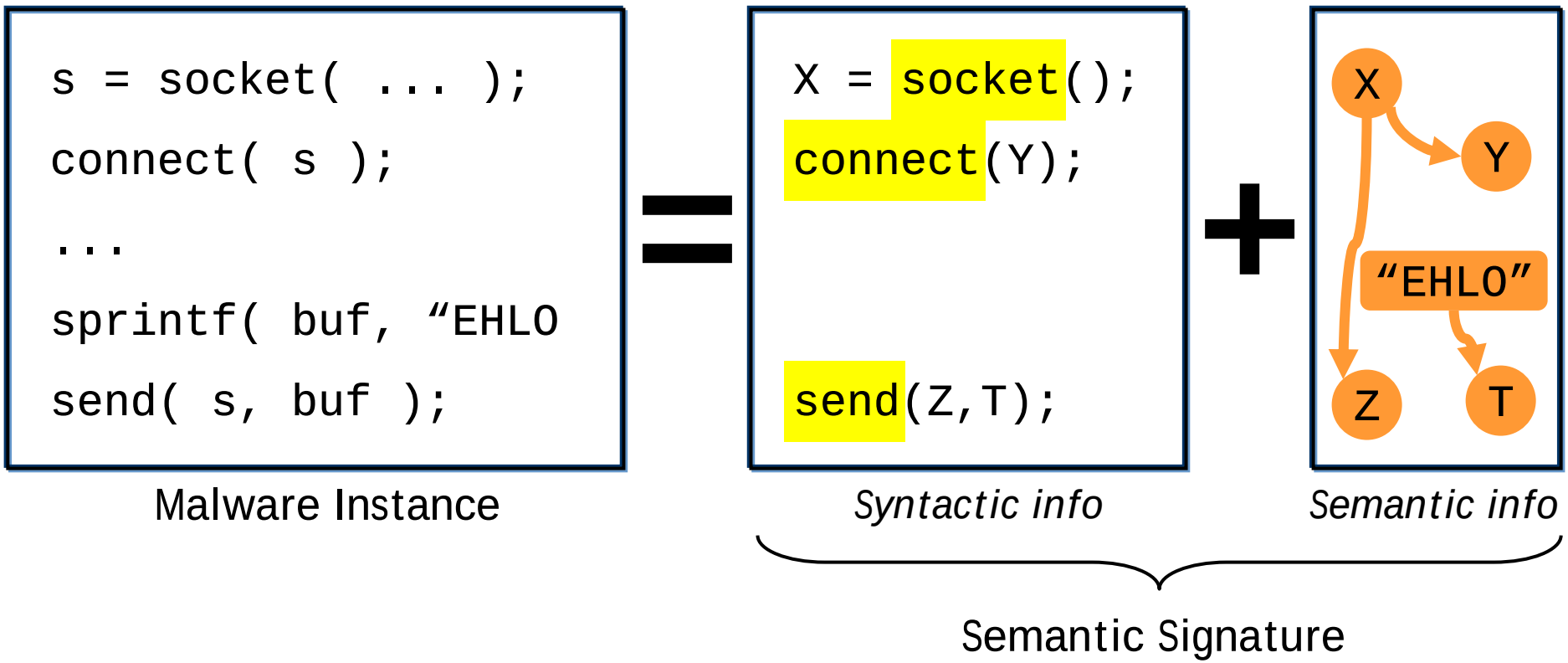
```
socket()
connect()

“EHLO”
send()
```

Knkz-on#pyw #W ¥N yyw 8Q

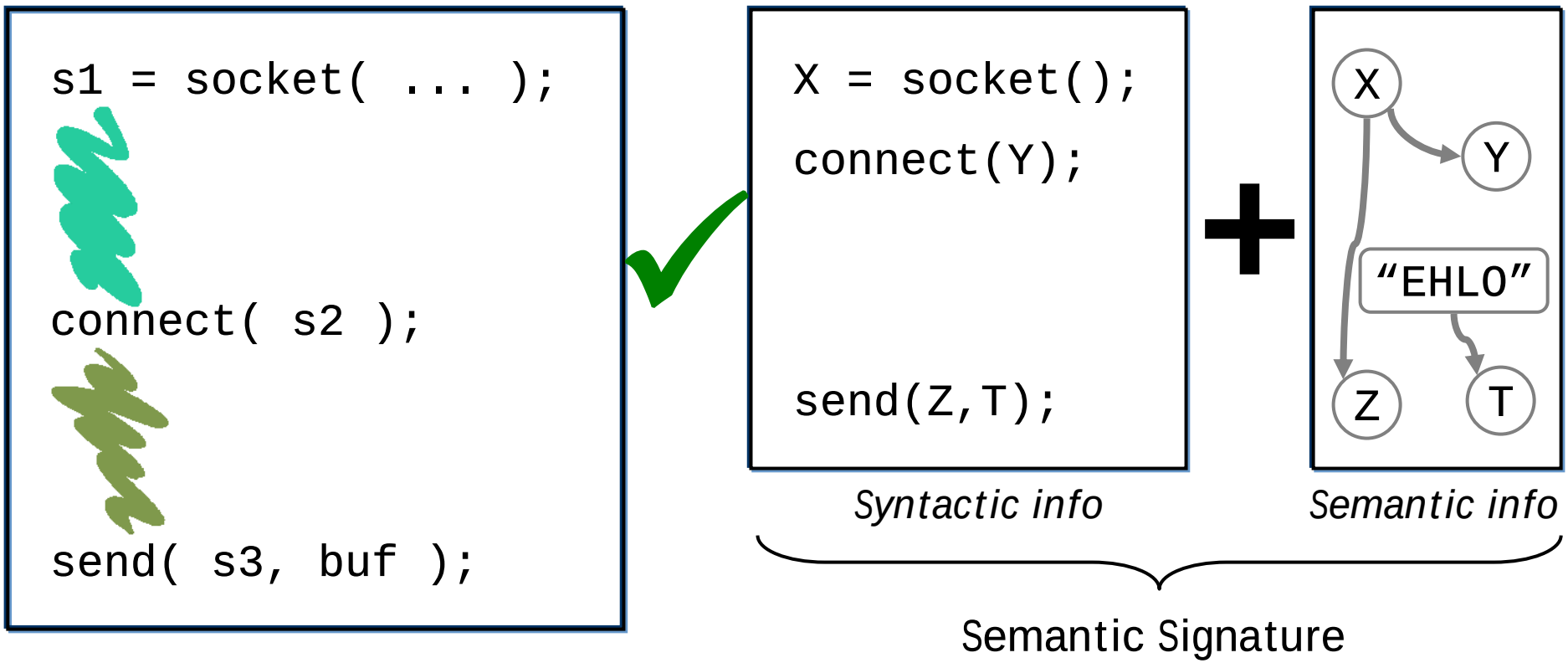
Semantic Signatures

- Goal of attacker: same behavior in different form.



Power of Semantic Signatures

- Detect any variant that uses the same sequence of instructions.



Semantics-Aware Detection

- We have a new detection model:

  Semantic signatures combine syntactic and semantic information.

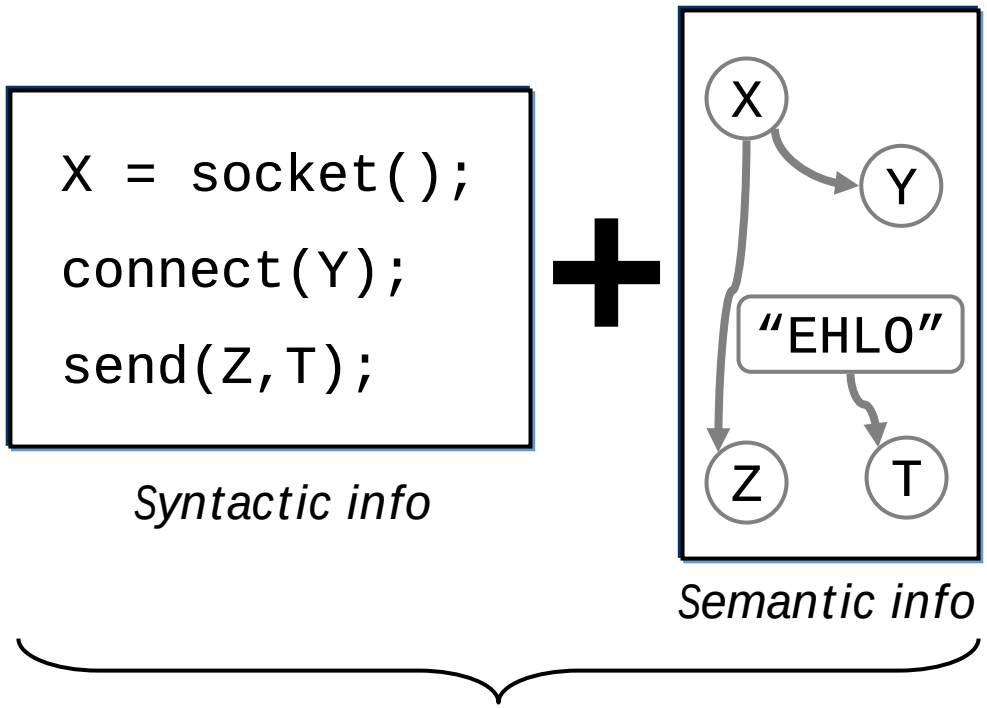
1. Can we build it?

2. Does it work?

3. Where do signatures come from?

A Semantics-Aware Detector

- Match the syntactic constructs, then check for the semantic information.

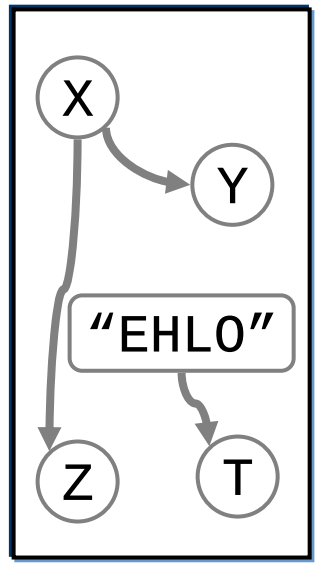


Checking for Semantic Info

Program:

```
1  s1 = socket( ... );
2  a[ i++ ] = s1;
3  s2 = a[ i – 1 ];
4  connect( s2 );
   ...
```

```
X = socket();
connect(Y);
...
```



- Check that "s2 has the same value as s1"
  or check the value predicate:

  value(s1 after line 1) == value(s2 before line 4)

Checking a Value Predicate

Program:

```
1  s1 = socket( ... );
2  a[ i++ ] = s1;
3  s2 = a[ i – 1 ];
4  connect( s2 );
   ...
```

Value predicate:

value(s1 after line 1)
==
value(s2 before line 4)

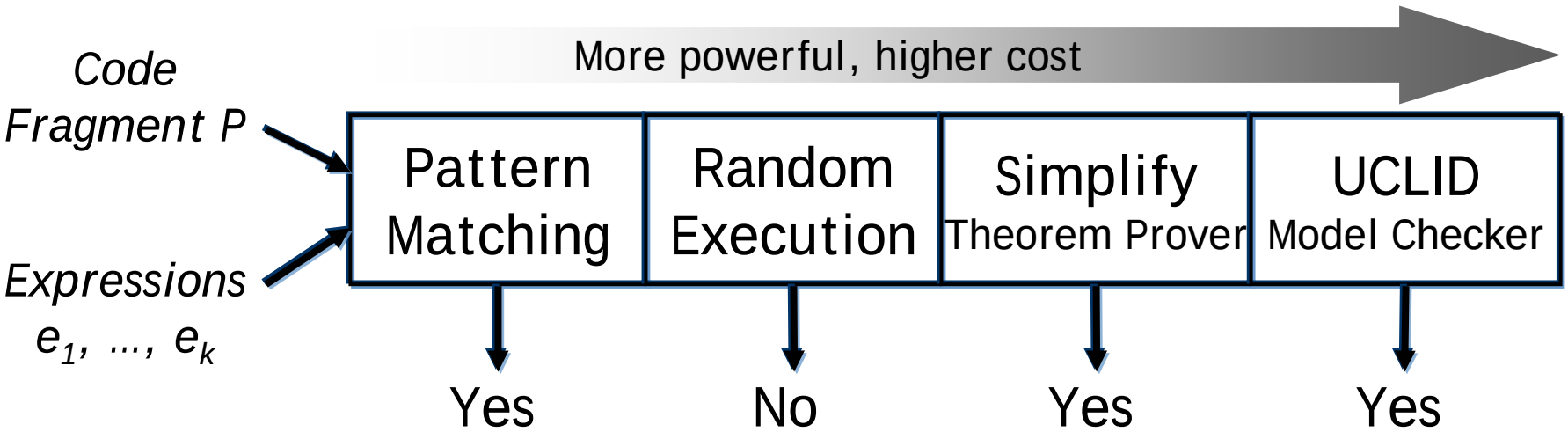
Equivalent condition:

Lines 2 and 3 are a semantic nop with respect to the value predicate.

Tools for Checking Value Preds.

- Instance of program verification problem:

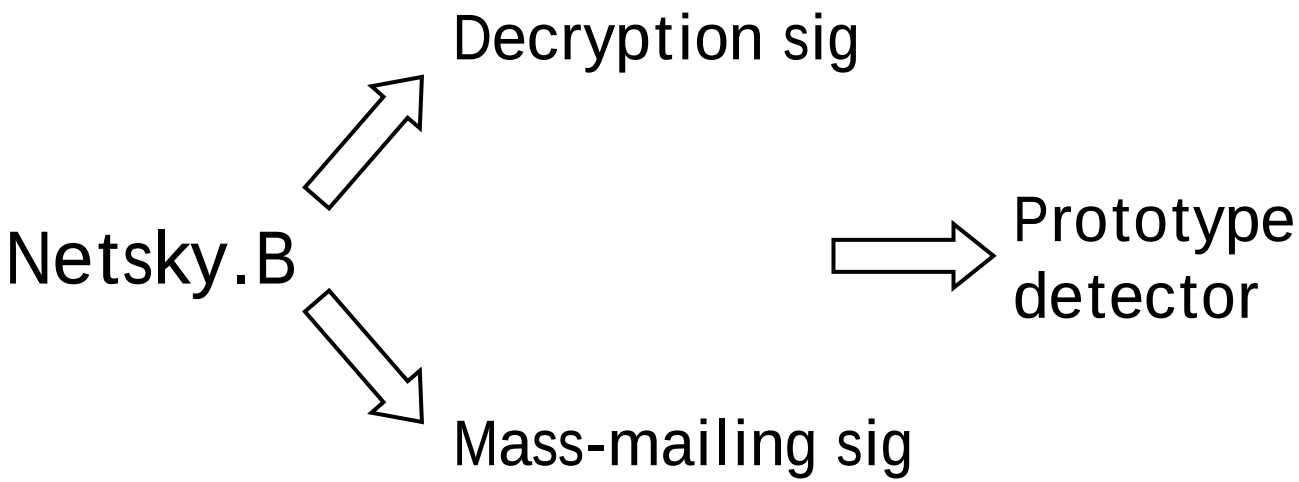
  Does program P respect property ϕ ?



Evaluation of Our Prototype

- Developed signatures for several families of worms.
- No false positives.
- Improved resilience to common obfuscations.

Evaluation of Semantic Signatures



Netsky.C	✓
Netsky.D	✓
Netsky.O	✓
Netsky.P	✓
Netsky.T	✓
Netsky.W	✓

McAfee uses individual signatures for each worm.

Semantic signatures provide forward detection.

Performance

- Prototype is slower than commercial anti-virus tools.

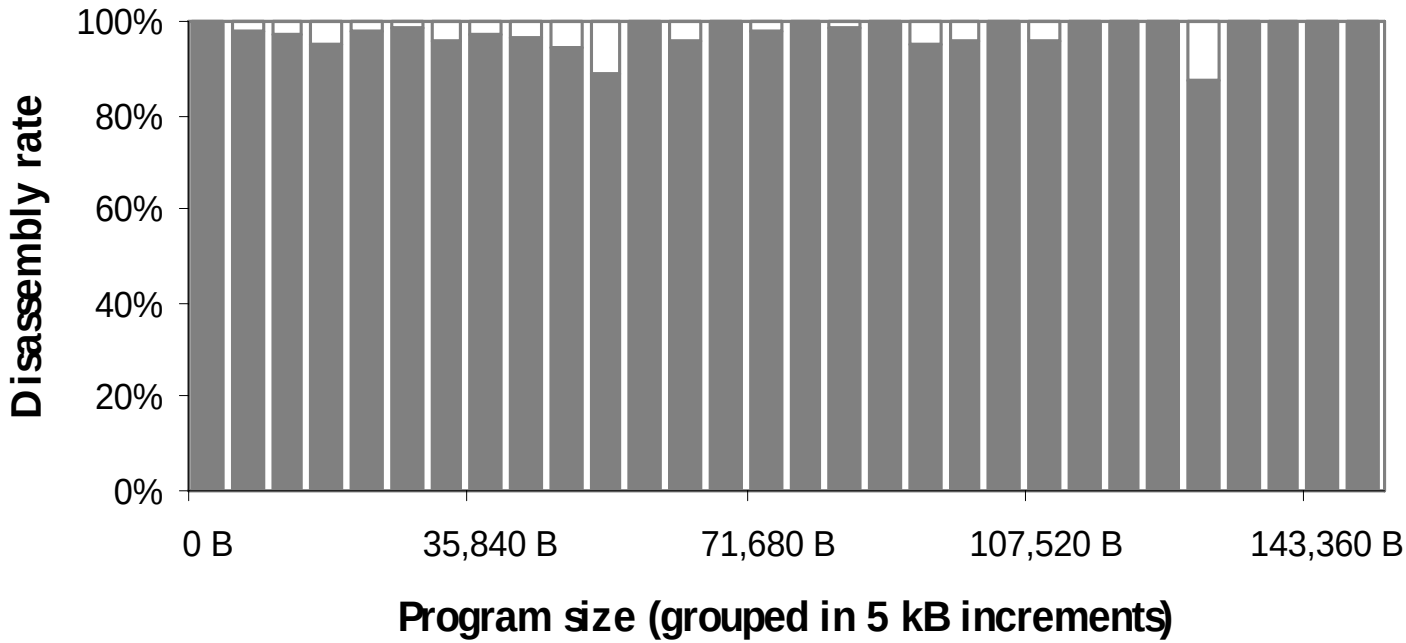
Malware Family	*Running Time*	
	Average	*Std. Deviation*
Netsky	99.57 s	41.01 s
Beagle	56.41 s	40.72 s

- Plenty of room for improvement.

  e.g. disassembler: 25% of time.

Evaluation: False Positive Rate

- Tested the semantic signatures on 2,000 benign Windows binaries.
- False positive rate: 0%



Evaluation: Obfuscation Resilience

- Different types garbage insertion applied to Beagle.Y to obtain more variants.

Obfuscation Type	*Semantics-Aware Detection*		*McAfee*
	Average Time	*Detection Rate*	
Nop insertion	74.81 s	**100%**	75%
Stack op. insertion	159.10 s	**100%**	25%
Math op. insertion	186.50 s	**95%**	5%

Limitations

- Limited support for equivalent code sequences.

```
a = b * 2
```

X

```
a = b << 1
```

- In the semantic signature, order of instructions is significant.

```
a = b + 1
c = c + 1
```

X

```
c = c + 1
a = b + 1
```

Where do we go from here?

Up to now: Syntactic signature detection

This work: Semantics-aware detection

Future: Equivalent code sequences
Detection of self-replication
Better performance

Semantics-Aware Malware Detection

Mihai Christodorescu
mihai@cs.wisc.edu

Somesh Jha
jha@cs.wisc.edu

University of Wisconsin, Madison

Sanjit Seshia
sanjit@cs.cmu.edu

Dawn Song
dawnsong@cmu.edu

Randal Bryant
bryant@cs.cmu.edu

Carnegie Mellon University