All You Ever Wanted to Know About Dynamic Taint Analysis & Forward Symbolic Execution (but might have been afraid to ask)

(Yes, we were trying to overflow the title length field on the submission server)

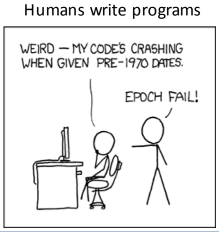
Edward J. Schwartz, Thanassis Avgerinos, David Brumley

A ***Few Things*** You Need to Know About Dynamic Taint Analysis & Forward Symbolic Execution (but might have been afraid to ask)

Edward J. Schwartz, Thanassis Avgerinos, David Brumley

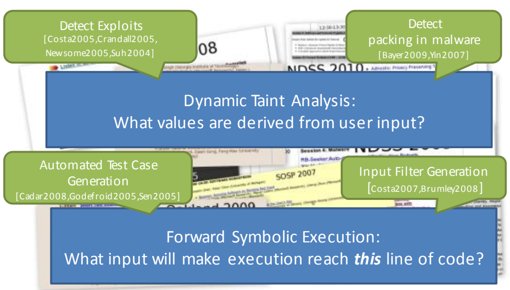
The Root of All Evil

Humans write programs



This Talk:
Computers Analyzing Programs Dynamically at Runtime

Two Essential Runtime Analyses



Detect Exploits [Costa2005,Crandall2005, Newsome2005,Suh2004]

Detect packing in malware [Bayer2009,Yin2007]

Dynamic Taint Analysis:
What values are derived from user input?

Automated Test Case Generation [Cadar2008,Godefroid2005,Sen2005]

Input Filter Generation [Costa2007,Brumley2008]

Forward Symbolic Execution:
What input will make execution reach ***this*** line of code?

Our Contributions

Computers Analyzing Programs Dynamically at Runtime

Dynamic Taint Analysis:
Is this value affected by user input?

Forward Symbolic Execution:
What input will make execution reach ***this*** line of code?

1: Turn English descriptions into an ***algorithm***
- Operational Semantics

2: Algorithm highlights caveats, issues, and unsolved problems that are deceptively hard

Our Contributions (cont'd)

3: Systematize recurring themes in a wealth of previous work

Dynamic Taint Analysis: What values are derived from user input?

1. How it works – example
2. Desired properties
3. Example issue. Paper has many more.

tainted untainted

```
x = get_input( )
y = x + 42
...
goto y
```

Input is tainted

Taint Introduction

$$\text{Input}\ \frac{t = \text{IsUntrusted}(src)}{\text{get\_input}(src) \downarrow t}$$

Δ

Var	Val
x	7

τ

Var	Tainted?
x	T

tainted untainted

```
x = get_input( )
y = x + 42
...
goto y
```

Data derived from user input is tainted

Taint Propagation

$$\text{BinOp}\ \frac{t_1 = \tau[x_1]\ ,\ t_2 = \tau[x_2]}{x_1 + x_2 \downarrow t_1 \vee t_2}$$

Δ

Var	Val
x	7
y	49

τ

Var	Tainted?
x	T
y	T

tainted untainted

```
x = get_input( )
y = x + 42
...
goto y
```

Policy Violation Detected

Taint Checking

$P_{goto}(t_a) = \neg\ t_a$

(Must be true to execute)

Δ

Var	Val
x	7
y	49

τ

Var	Tainted?
x	T
y	T


```
x = get_input( )
y = ...
...
goto y
```

Jumping to overwritten return address

Real Use: Exploit Detection

```
...
strcpy(buffer,argv[1]) ;
...
return ;
```


Memory Load

Variables

Δ

Var	Val
x	7

τ

Var	Tainted?
x	T

Memory

μ

Addr	Val
7	42

τ_μ

Addr	Tainted?
7	F

Problem: Memory Addresses

x = get_input( )
y = load(x)
...
goto y

All values derived from user input are tainted??

Δ

Var	Val
x	7

μ

Addr	Val
7	42

τ_μ

Addr	Tainted?
7	F

Policy 1: Taint depends only on the memory cell

x = get_i...
y = load(...
...
goto y

Undertainting

Failing to identify tainted values
- e.g., missing exploits

Δ

Var	Val
x	7

μ

Addr	Val
7	42

Taint Propagation

$$\text{Load}\ \frac{v = \Delta[x]\ ,\ t = \tau_\mu[v]}{\text{load}(x) \downarrow t}$$

τ_μ

Addr	Tainted?
7	F

Policy 2: If either the address or the memory cell is tainted, then the value is tainted

x = get_i...
y = load(...
...
goto y

Overtainting

Unaffected values are tainted
- e.g., exploits on safe inputs

Memory

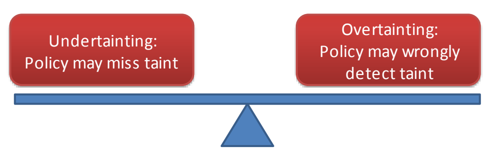
Address expression is tainted

printa
printb

Taint Propagation

$$\text{Load}\ \frac{v = \Delta[x]\ ,\ t = \tau_\mu[v],\ t_a = \tau[x]}{\text{load}(x) \downarrow t \lor t_a}$$

Research Challenge
State-of-the-Art is not perfect for all programs



Forward Symbolic Execution:
What input will make execution reach ***this*** line of code?

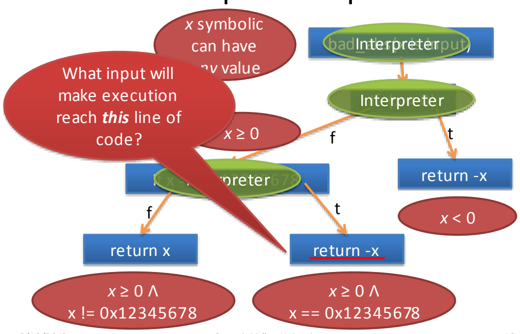
- How it works – example
- Inherent problems of symbolic execution
- Proposed solutions

The Challenge

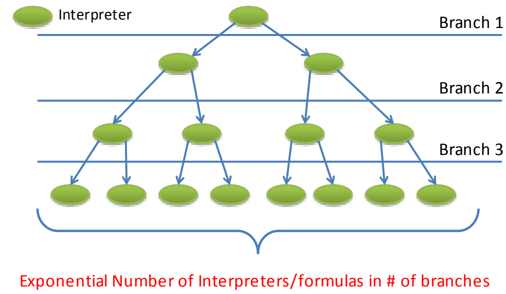

```
bad_abs(x is input)
  if (x < 0) then
      return -x
  if (x = 0x12345678) then
      return -x
  return x
```

Forward Symbolic Execution:
What input will make execution reach ***this*** line of code?

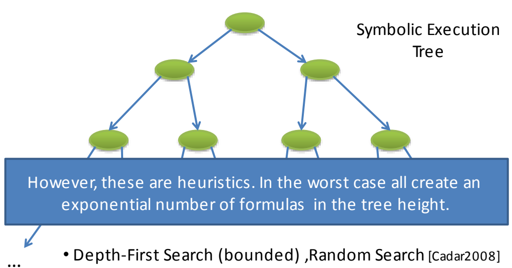
A Simple Example



One Problem:
Exponential Blowup Due to Branches



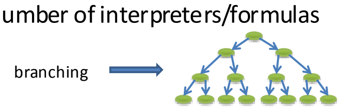
Path Selection Heuristics



- Depth-First Search (bounded) ,Random Search [Cadar2008]
- Concolic Testing [Sen2005,Godefroid2008]

Symbolic Execution is *not* Easy

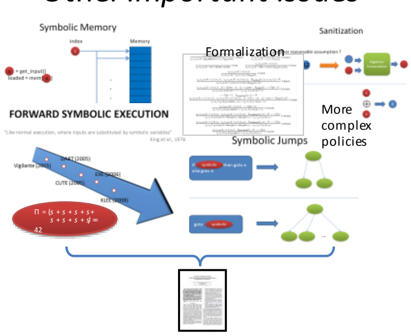
- Exponential number of interpreters/formulas



- Exponentially-sized formulas

- Solving a formula is NP-Complete!

Other *Important* Issues



Conclusion

- Dynamic taint analysis and forward symbolic execution used extensively in literature
  - Formal algorithm and what is done for each possible step of execution often not emphasized
- We provided a formal definition and summarized
  - Critical issues
  - State-of-the-art solutions
  - Common tradeoffs

Thank You!
thanassis@cmu.edu

Questions?