Taint Mechanisms

Advanced topics in Computer Security

Matteo Di Pirro

BSc in Computer Science
Department of Mathematics

University of Padova

December 7, 2016

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Outline

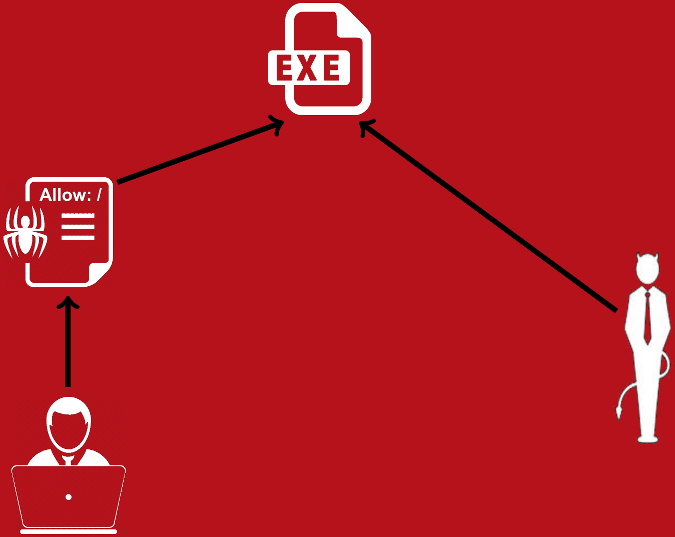
Introduction

Dynamic Taint Analysis

Forward Symbolic Execution

Conclusions

Of who or what do we trust?



Input Analysis

There are two essential questions about the input analysis:

Input Analysis

There are two essential questions about the input analysis:

1. **Is the final value affected by user input?**
    - **Dynamic Taint Analysis**!
    - Tracks information flow between sources and sinks

Input Analysis

There are two essential questions about the input analysis:

1. **Is the final value affected by user input?**
   - **Dynamic Taint Analysis**!
   - Tracks information flow between sources and sinks
2. **What input will make execution reach this line of code?**
   - **Forward Symbolic Execution**
   - Allows us to reason about the behavior of a program on many different inputs

Dynamic Taint Analysis

```
x = get_input(😈)
z = 42
y = x + z
goto y
```

Dynamic Taint Analysis

Tainted

```
x = get_input(😈)
z = 42
y = x + z
goto y
```

x is derived from a tainted source

Dynamic Taint Analysis

```
x = get_input(😈)
z = 42
y = x + z
goto y
```

Untainted

z is a ”static” constant

Dynamic Taint Analysis

```
x = get_input(😈)
z = 42
y = x + z
goto y
```

Is y taited?

Dynamic Taint Analysis

```
x = get_input(😈)
z = 42
y = x + z
goto y
```

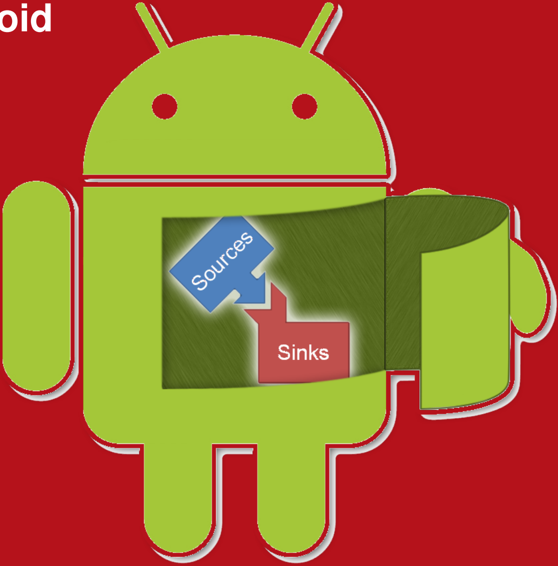
Is y taited?

It depends on the selected policy

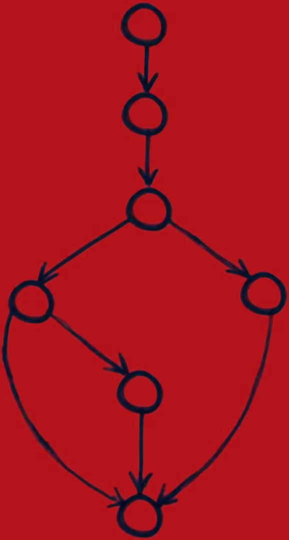
What’s a policy?

- A taint policy specifies three properties:
  - **Taint Introduction**
    - How is taint introduced into a system?
  - **Taint Propagation**
    - How does taint propagate into a system?
  - **Taint Checking**
    - Is the current operation secure?
- **Undertainting** vs **Overtainting**

Can we go further?
TaintDroid

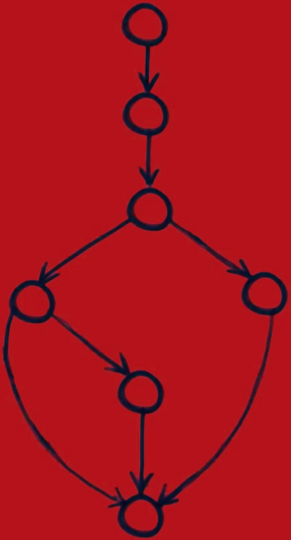


Limitations



- **Sanitization problem**

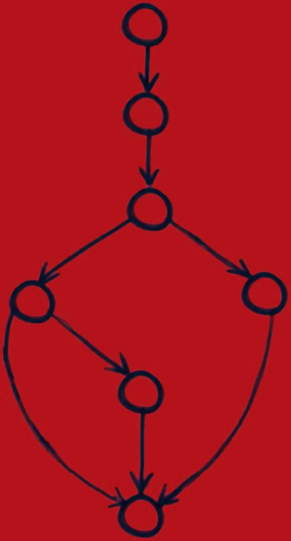
Limitations



- **Sanitization problem**

$$b := a \oplus a$$

Limitations



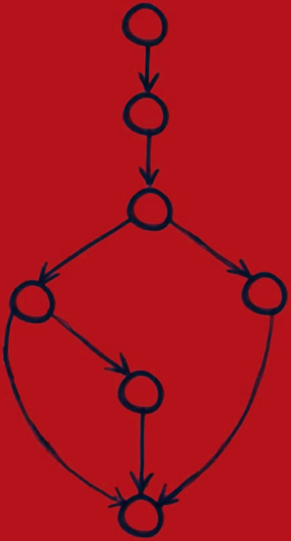
- **Sanitization problem**

$$b := a \oplus a$$

- Pure dynamic taint analysis considers **data flows**...
  ...but it ignores **control-flows**

```
x = get_input(src)
if x == 1 then
    y = 1
z = 42
```

Limitations



- **Sanitization problem**

  $$b := a \oplus a$$

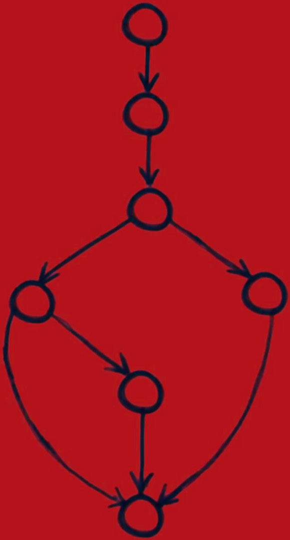
- Pure dynamic taint analysis considers **data flows**...

  ...but it ignores **control-flows**

```
x = get_input(src)
if x == 1 then
    y = 1
z = 42
```

- What about different security policies for different I/O channels?

Limitations



- **Sanitization problem**

$$b := a \oplus a$$

- Pure dynamic taint analysis considers **data flows**...
  ...but it ignores **control-flows**

```
x = get_input(src)
if x == 1 then
    y = 1
z = 42
```

- What about different security policies for different I/O channels?
  → **Static analysis**

Forward Symbolic Execution

- We can reason about the behavior of a program using the logic...
- ... and it is **conceptually** a very simple process

```
x = 2 * get_input(src)
if x - 5 == 14 then goto 3 else goto 4
// line 3:  catastrophic failure
// line 4:  normal behavior
```

Forward Symbolic Execution

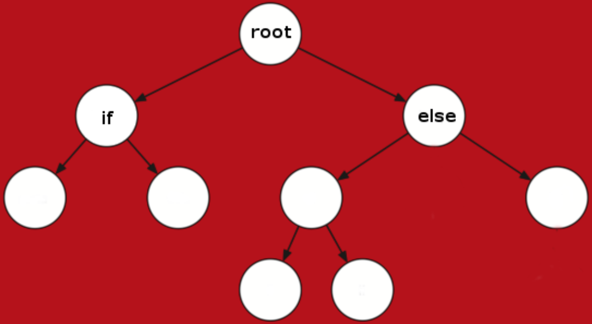
- We can reason about the behavior of a program using the logic...
- ... and it is **conceptually** a very simple process

```
x = 2 * get_input(src)
if x - 5 == 14 then goto 3 else goto 4
// line 3:  catastrophic failure
// line 4:  normal behavior
```

- `get_input(src)` now returns a **symbol** instead of a concrete value
- But now expressions **cannot** be fully evaluated to a concrete value

Path Selection and Performance

- For each conditional jump we must decide what path to follow first
  - But some path may never terminate
- Exponential blowup due to branches



Solutions

- **Path Selection Heuristics**
  - Concolic Testing
  - Depth-First or Random Search
- More and faster **hardware**
- Identify **redundancies** between formulas
- Identify **independent subformulas**

Solutions

UNIVERSITÀ DEGLI STUDI DI PADOVA

- **Path Selection Heuristics**
  - Concolic Testing
  - Depth-First or Random Search
- More and faster **hardware**
- Identify **redundancies** between formulas
- Identify **independent subformulas**

...but solving a logical formula is a $\mathcal{NP}$**-Complete** problem!

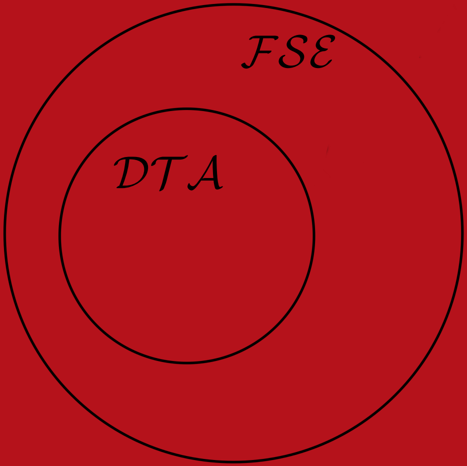
A small comparison

Dynamic
Taint
Analysis

$\overset{?}{\substack{\subset \\ = \\ \supset}}$

Forward
Symbolic
Execution

A small comparison



Dynamic taint analysis analyzes only ***feasible*** paths

Conclusions

v Conceptually simple methods of analysis

v There are a lot of possible use cases

- Malware detection and analysis
- Automatic testing
- Automatic programs understanding

x Usable with some care

x The effectiveness depends on the application

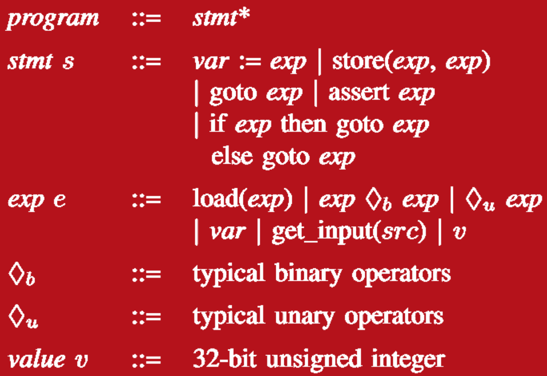
UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Thank you for allowing me to taint your time!

Questions?

DIPARTIMENTO
MATEMATICA

SimpIL Grammar



$$
\begin{array}{lcl}
\textit{program} & ::= & \textit{stmt}^* \\
\textit{stmt}\ s & ::= & \textit{var} := \textit{exp} \mid \text{store}(\textit{exp}, \textit{exp}) \\
 & & \mid \text{goto } \textit{exp} \mid \text{assert } \textit{exp} \\
 & & \mid \text{if } \textit{exp} \text{ then goto } \textit{exp} \\
 & & \quad \text{else goto } \textit{exp} \\
\textit{exp}\ e & ::= & \text{load}(\textit{exp}) \mid \textit{exp} \lozenge_b \textit{exp} \mid \lozenge_u \textit{exp} \\
 & & \mid \textit{var} \mid \text{get\_input}(\textit{src}) \mid v \\
\lozenge_b & ::= & \text{typical binary operators} \\
\lozenge_u & ::= & \text{typical unary operators} \\
\textit{value}\ v & ::= & \text{32-bit unsigned integer}
\end{array}
$$

SimplL Operational Semantic

- **Each** statement rule of the operational semantic is like:

$$\frac{\text{computation}}{<\text{current state}>,\ \text{stmt} \rightarrow <\text{end state}>,\ \text{stmt}}$$

- The `state` is composed of:
  - Program statements ($\sum$)
  - Current memory state (μ)
  - Current values for variables (Δ)
  - Program counter (***pc***)
  - Current statement (***i***)

Memory Address Problems
Forward Symbolic Execution

- What are we supposed to do if a referenced address is derived from user input?
  - `LOAD`, `STORE` → **Symbolic Memory Address**
  - `GOTO` → **Symbolic Jumps**
- Solutions
  - Concolic testing
  - SMT (**S**atisfiability **M**odulo **T**heories) solvers
    - $\mathcal{NP}$**-Complete** problem!
  - Static and alias analysis