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

Matteo Di Pirro

BSc in Computer Science
Department of Mathematics

University of Padova

December 7, 2016

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Università degli Studi di Padova

Outline

Static and Dynamic Analysis

- **Static Analysis**
  - Examines a program's text to derive properties that hold for all executions
  - Program-centric analysis
- **Dynamic Analysis**
  - Examines the running program to derive properties hold for one or more executions
  - Detect violations of stated properties
  - Provide useful information about the behavior of the program
  - Input-centric analysis

Dynamic Analysis

There are two essential questions about the input analysis:

Dynamic 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

Dynamic 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

Use cases

The number of security applications utilizing these two techniques is enormous:

Use cases

The number of security applications utilizing these two techniques is enormous:

1. **Unknown Vulnerability Detection**: monitor whether user input is executed

Use cases

The number of security applications utilizing these two techniques is enormous:

1. **Unknown Vulnerability Detection**: monitor whether user input is executed
2. **Automatic Input Filter Generation**: detect and remove exploits from the input stream

Use cases

The number of security applications utilizing these two techniques is enormous:

1. **Unknown Vulnerability Detection**: monitor whether user input is executed
2. **Automatic Input Filter Generation**: detect and remove exploits from the input stream
3. **Forward Symbolic Execution**: analyze how information flows through a malware binary

Use cases

The number of security applications utilizing these two techniques is enormous:

1. **Unknown Vulnerability Detection**: monitor whether user input is executed
2. **Automatic Input Filter Generation**: detect and remove exploits from the input stream
3. **Forward Symbolic Execution**: analyze how information flows through a malware binary
4. **Test Case Generation**: automatically generate inputs to test programs

SimpIL

Designed to demonstrate the critical aspects of this analysis.

program	::=	*stmt**
stmt s	::=	*var* := *exp* | store(*exp*, *exp*)
		| goto *exp* | assert *exp*
		| if *exp* then goto *exp*
		else goto *exp*
exp e	::=	load(*exp*) | *exp* $\Diamond_b$ *exp* | $\Diamond_u$ *exp*
		| *var* | get_input(*src*) | *v*
$\Diamond_b$	::=	typical binary operators
$\Diamond_u$	::=	typical unary operators
value v	::=	32-bit unsigned integer

SimpIL Grammar

SimpIL

UNIVERSITÀ DEGLI STUDI DI PADOVA

Designed to demonstrate the critical aspects of this analysis.

- **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***)

Dynamic Taint Analysis

UNIVERSITÀ DEGLI STUDI DI PADOVA

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()

Untainted

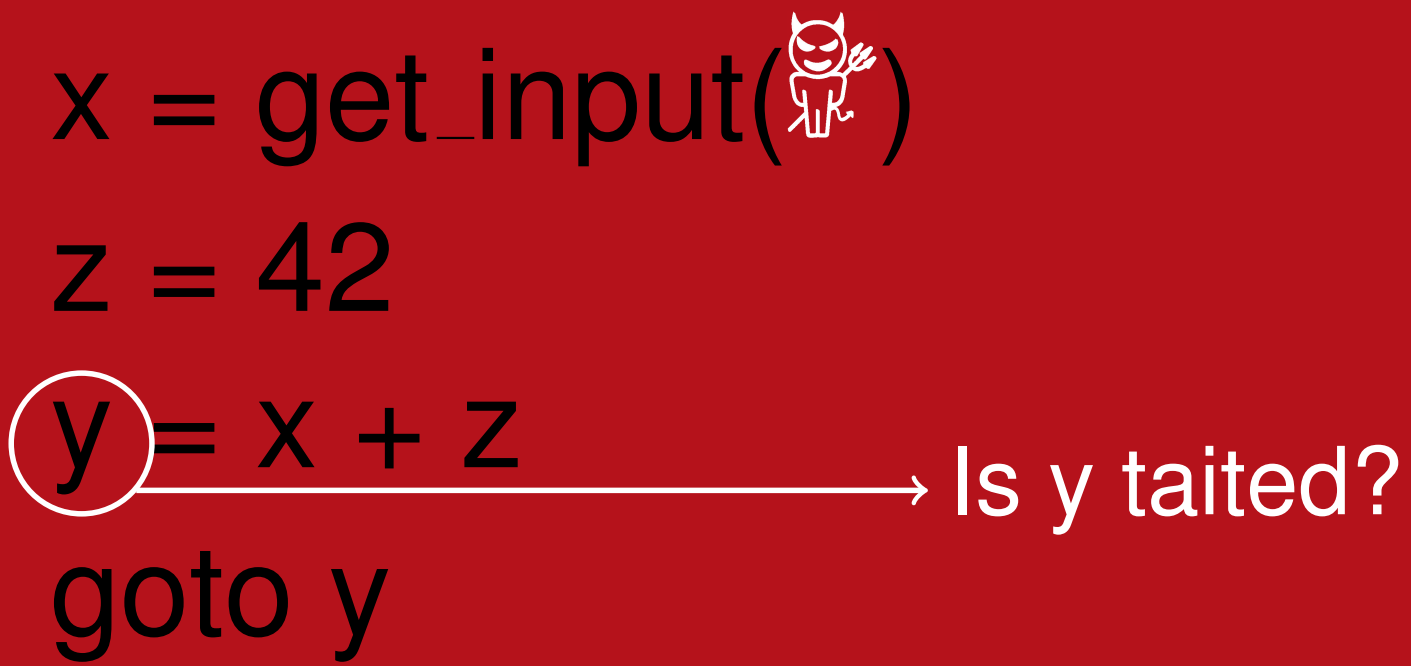
z = 42

y = x + z

goto y

z is a ”static” constant

Dynamic Taint Analysis



Dynamic Taint Analysis

x = get_input()

z = 42

y = x + z → Is y taited?

goto y

It depends on the selected policy

What's a policy?

UNIVERSITÀ DEGLI STUDI DI PADOVA

- A taint policy specifies three properties:
  - **Taint Introduction**
    - specifies how taint is introduced into a system
    - typically distinguishes between different input sources
  - **Taint Propagation**
    - specifies the taint status for data derived from tainted or untainted operands
  - **Taint Checking**
    - is used to determine the runtime behavior of a program
- **Undertainting** vs **Overtainting**

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

THANK YOU FOR ALLOWING ME TO TAINT YOUR PRECIOUS TIME!

Questions?

DIPARTIMENTO
MATEMATICA