Dytan

ISSTA 2007

James Clause, Wanchun Li, Alessandro Orso

김영철

2016. 4. 19.

Introduction

- Dynamic taint analysis
  : 프로그램 실행 시에 특정한 데이터를 표시하고 추적
  : 보안 분야에서 점점 많이 사용
  : 특정한 목적에 대해서만 정의
  ➔ 확장이나 다른 상황에 적용하는 데 어려움
  : data-flow based tainting only
  : 시간이 많이 소요되고 복잡

Introduction

- Framework in this paper
  (1) flexible and customizable
  (2) data-flow, data-flow & control-flow 수행
  (3) binary 에 대해서도 작동

Introduction

- Dytan 소개
- Dytan을 사용하여 수행되는 예비조사 소개
  (1) Dytan을 이용한 경험
  (2) 구체적인 특징이 taint analysis의 효율과 정확성에 어떤 영향을 미치는지 설명

Background and motivation

- dynamic tainting
  (1) 하나 이상의 marking을 데이터 값과 연관
  (2) marking들을 전달
  : 프로그램 내부의 정보 흐름을 추적

Background and motivation

- explicit information flow
  tainted data가 직접적으로 다른 data에 영향을 줌
  data dependency와 관련

- implicit information flow
  tainted data각 간접적으로 다른 data에 영향을 줌
  control dependency와 관련

Our approach

- General Framework
  : dynamic taint analysis의 특징이 되는 세 가지 측면이 있음
  ➔ taint sources, propagation policy, taint sinks
  : 다양한 분석 방법이 표현될 수 있음

Our approach

- Taint sources
  : taint marking 으로 초기 설정되는 data
  : Memory location
    (1) variable names
    (2) function-return values
    (3) data read from I/O stream

Our approach

- Taint sources
  (1) Variables and memory offsets
  : 사용자는 변수 이름과 범위를 지정하여 taint 되어야 하는 memory area 를 지정할 수 있음

  (2) Data returned from a specific functions
  : 사용자는 함수의 이름을 지정하여 taint 되어야 하는 함수의 반환값을 지정할 수 있음

Our approach

- Taint sources
  (3) Data from a type of I/O stream
  (4) Data from a specific I/O stream
  : 키보드, 네트워크, 파일시스템이 taint 되어야할 때 I/O stream의 타입을 지정할 수 있음

Our approach

- Propagation policies
  : 실행 중에 taint marking을 어떻게 전파할지 기술

  (1) Identifying affecting data
  data-flow only – explicit propagation
  data- and control-flow – implicit propagation
  사용자는 두 방법중 하나를 선택할 수 있음

  (2) Defining a mapping function

Our approach

- Taint sinks
  : ID, memory location, code location, checking operations 네 가지 측면이 특징이 됨
  : memory location 과 code location 을 지정하는 두 가지 방법을 제공함

Our approach

- Taint sinks
  (1) memory location, code location 독립적으로 명시됨
  : 사용자가 변수와 memory location 을 명시할 수 있음
  : source code를 통해 code location 지정
  ➔ source code가 없으면 불가능
  : binary code에 code location 지정

Our approach

- Taint sinks
  (2) 사용자가 instruction의 interest를 명시할 수 있음
  …

Our approach

- Instantiation for x86 Binaries
  : x86의 binary 분석을 위한 포괄적인 framework를 정의
  : 주로 data- and control flow based propagation에 초점

Our approach

- Instantiation for x86 Binaries
  (1) Maintaining Taint Markings
  bit vectors에 marking 저장

Our approach

- Instantiation for x86 Binaries
  (2) Data-flow Based Taint Propagation
  assembly instruction이 주어졌을 때,
  ① source operand와 destination operand를 구별
  ② source operand와 관련 있는 marking들을 combine
  destination operand와 연관 지음

Our approach

- Instantiation for x86 Binaries
  (3) Control-flow Based Taint Propagation
  Background