ИСП РАН

Магистерская диссертация на тему

Исследование и разработка методов динамического анализа для определения входных данных влияющих на выполнение условных переходов

Дьячков Л.А.

Руководитель: к.ф-м.н, с.н.с. Курмангалеев Ш.Ф.

5 Апреля 2019

ИСП РАН

Введение

Фаззинг

Фаззинг-тестирование – активно развивающийся метод для поиска ошибок в программном обеспечении

Введение

Фаззинг

Фаззинг-тестирование – активно развивающийся метод для поиска ошибок в программном обеспечении

Проблема

Может быть затруднено нахождение входных данных, позволяющих “пройти” условный переход

Введение

Фаззинг

Фаззинг-тестирование – активно развивающийся метод для поиска ошибок в программном обеспечении

Проблема

Может быть затруднено нахождение входных данных, позволяющих “пройти” условный переход

Предлагаемое решение

Использовать методы динамического анализа для определения какие байты из входного файла влияют на конкретный условный переход

Постановка задачи

Цель работы

- Разработка метода динамического анализа, позволяющего определять байты входного файла, влияющего на выполнение инструкции условного перехода.
- Программная реализация метода, работающая на операционной системе Linux с архитектурой процессора x86-64

Постановка задачи

Цель работы

- Разработка метода динамического анализа, позволяющего определять байты входного файла, влияющего на выполнение инструкции условного перехода.
- Программная реализация метода, работающая на операционной системе Linux с архитектурой процессора x86-64

Подзадачи

- Изучение существующих технологий динамического символьного выполнение и динамического анализа потока данных.
- Сравнение технологий на тестовом наборе
- Обзор возможных подходов, реализация прототипов, разработка метода
- Программная реализация на основе выбранной технологии

Динамическое символьное выполнение

Определение

Метод динамического анализа, заключающийся в том в том, что во время выполнения программы некоторым конкретным значениям ставятся в соответствие символьные переменные. Затем, для каждой выполняемой инструкции, генерируются формулы для SMT решателя.

online символьное выполнение

Модуль символьного выполнения используется для того, чтобы генерировать конкретные данные для посещения новых узлов cfg программы.

Offline символьное выполнение

Модуль символьного выполнения используется для анализа конкретной трассы выполнения.

Динамическое символьное выполнение

Определение

Метод динамического анализа, заключающийся в том в том, что во время выполнения программы некоторым конкретным значениям ставятся в соответствие символьные переменные. Затем, для каждой выполняемой инструкции, генерируются формулы для SMT решателя.

online символьное выполнение

Модуль символьного выполнения используется для того, чтобы генерировать конкретные данные для посещения новых узлов cfg программы.

Offline символьное выполнение

Модуль символьного выполнения используется для анализа конкретной трассы выполнения.✓

Динамический анализ помеченных данных

Определение

Динамический анализ помеченных данных (Dynamic Taint Analysis), также известный как динамический анализ потока данных (Dynamic Flow tracking) – это техника анализа програм, позволяющая определить какие состояния программы зависят от входных данных.

Принцип работы

- *Определение источников помеченных данных*. Обычно метками снабжаются данные, получаемые из недоверенного источника (файл, stdin, сеть).
- *Распространение пометок (Taint propogation)*. Для каждой инструкции необходимо принять решение как распротраняются пометки в зависимости от её операндов и факта их помеченности
- *Применение политик безопасности*. Например отслеживание попадания помеченных данных в аргументы “опасных” функций, или факта помеченности счетчика инструкций.

Исследованные технологии

Динамическое символьное выполнение

- Triton
- Angr
- Manticore

Динамический анализ помеченных данных

- Triton
- Taintgrind
- libdft64
- moflow (bap gentrace)

Triton

Фреймворк, поддерживающий архитектуры x86 и x86-64, содержит модули DSE (offline) и анализа помеченных данных, использует *Intel Pin* для динамической бинарной инструментации.

Достоинства

- Поддерживает offline режим динамического символьного выполнения (Нет необходимости предварительно снимать трасу, при помощи pin это делается “на лету”)
- Поддерживает **"ONLY_TAINTED"**, позволяющий генерировать smt формулы только для помеченных инструкций

Недостатки

- Работает крайне медленно (примерно на два порядка медленее других инструментов на базе pin)
- Нет поддержки символьных файлов/пометки данных на основе системных вызовов

Angr

Платфмормонезависимый фреймворк динамического символьного выполнения, использующий трансляцию инструкций в VEX с последующей эмуляцией.

Достоинства

- Поддержка множества архитектур
- Хороший онлайн движок, отлично работающий поиск состояний на небольших примерах.
- Есть множество полезных примитивов (таких как символьные файлы) из коробки

Недостатки

- Нет оффлайн режима (плагины, которые должны его поддерживать не работают)
- На настоящих (например binutils) программах онлайн поиск не может дойти до интересных состояний

Manticore

Аналог Angr, поддерживающий также смарт-контракты. использует эмуляцию инструкций.

Достоинства

- Поддерживает offline режим

Недостатки

- Медленно работает (около 50 секунд эмуляции для того, чтобы дойти до main в программе с glibc)
- Нет поддержки SSE инструкций
- Нет поддержки некоторых системных вызовов при эмуляции