Using Infer to find Bugs

Shane Magrath[1]

[1]Cyber and Electronic Warfare Division
Defence Science and Technology Group

February 2020

Training Outline

- Introduction and Overview
- Labs:
  - Docker and Infer Installation
  - Lab One: Basic Infer Usage
  - Lab Two: **Inferbo**: Buffer Overflow Analysis
  - Lab Three: **Quandary**: Static Source-Sink Analysis
  - Lab Four: **Linting with AL**

Training Notes

- Today is self-paced lab directed learning
- Lunch is 12:30 - 1:30
- All content is available from GitHub from the user "DSTCyber"
- WiFi is available:
- Feedback Welcome: email me

Purpose of Today

What is the goal of today?

- Introduction to static analysis for finding bugs
- Practical experience with a useful tool
- Equip you with one more skill
- Have you develop *realistic expectations* of what Infer can do

We will be auditing three games that were developed in C - Angband, Skynet and BSD Games.

Infer: An Introduction

Infer is

- a **static analysis** tool for **finding bugs** in software
- for C, C++, Objective C, and Java (Servers and Mobile Apps)
- on very large code bases
- using continuous integration processes
- with a reputation for finding *thousands* of bugs a month
- and is used by Facebook, Uber, Spotify, ...

Timeline:

- **2000s**: Queen Mary University: Seperation Logic and Bi-Abduction - Cristiano Calcagno, Dino Distefano and Peter O'Hearn
- **2009**: Monoidics startup
- **2013**: Facebook buys out Monoidics
- **2015**: Facebook open sources Infer

Infer: What makes it useful?

How does it do it?

- Compositional program analysis - analyzes procedures independently to
- Produce logical summaries, using the theories of
- **Abstract Interpretation**: "Inferbo"
  eg intervals domains, bitvector arithmetic domains, etc
- **Separation Logic** using Bi-abduction
- **Source-Sink Analysis** - "Quandary"
- **Linting** - "AL"

Infer: How does it work?

Operations:

- Infer *hooks* the *compilation* process
- Processes the parsing phases:
  Parse Tree, AST, Use-Defs, Control Flow Graphs
  Type Information, Data Flow
- Parse data is analyzed, summarised and stored
- Then a set of **checkers** and **linters** are applied
- produce bug reports

Understanding the Reports

Understanding Infer's bug reports:

- Static analysis is about *all possible* executions
  NOT *actual executions* that are constrained by runtime behaviours.
- **Think: "Could this be a *plausible* execution?"**
- Infer makes mistakes!
  Certain checkers are terrible ...
- Some syntactic constructs in C confuse the checkers
  Incomplete and buggy implementation

How to use Infer

To use Infer productively:

- Learn to triage the bug reports efficiently
- Read the first and last lines of each bug report
  Ignore the middle!
- Look for code smell in functions and translation units
  Bugs patterns are repeated by the same author...
- Think: Can we fuzz this?
  Combine static analysis with dynamic testing ...

Australian Government
Department of Defence
Science and Technology

Getting Started!

Setup the training environment:

- **Source the training:**
  `https://github.com/DSTCyber/infer-training`
- **Create a working directory for the training**
- **Get the training material**
  `https://github.com/DSTCyber/infer-training.git`
- **Open the lab notes:**
  `/infer-training/docs/lab/infer_training_lab.pdf`
- Get started!