New Static Analysis Techniques to Detect Entropy Failure Vulnerabilities

Andrew Russell & Rushi Shah

December 12, 2018

The University of Texas at Austin

Entropy Failures In the Wild

- OpenSSL
- FreeBSD

How Could This Have Been Avoided?

- Audit code once, then use relational verification to prove you haven't introduced bugs with small changes to program

- Differential approach works well with way software is written (CI, etc.)

Proposed Static Analysis Approach

Problem Statement

Given two versions V_1, V_2 of a program, prove that if legitimate sources of entropy in V_1 flow into their sinks properly, then the same is true of V_2.

If v_1 is the taint set of variable v passed to sink in program one and v_2 is the taint set of v in program two, then we would like

$$assert(v_1 == v_2)$$

2-safety property: making an assertion based on two runs of programs.

Static Analysis

- Static-analysis technique called predicate abstraction can prove 1-safety properties.
- Existing techniques can transform 2-safety properties into 1-safety properties.

Our Approach

Language Semantics

$$
\begin{array}{llll}
\textit{Statement} & S & := & A \\
 & & & \mid S_1 \ ; \ S_2 \\
 & & & \mid \text{if } p \text{ then } S_1 \text{ else } S_2 \\
 & & & \mid \text{while } p \ S \\
\textit{Predicate} & p & := & \top \mid \bot \mid A \mid \neg p \mid p \odot p \\
\textit{Operator} & \odot & := & \wedge \mid \vee
\end{array}
$$

Predicate Abstraction

- Off-the-shelf state-of-the-art: CPAChecker

High level overview

1. Instrumentation
2. Product Program
3. Assertions + CPAChecker

Product Program Construction

- Sequential Composition

$$S_1 \ ; \ S_2$$

Product Program Construction

- Sequential Composition

$$S_1 \ ; \ S_2$$

- Synchronized Composition

$$S_1 \otimes S_2$$

Product Program Construction

- Sequential Composition

$$S_1 \ ; \ S_2$$

- Synchronized Composition

$$S_1 \otimes S_2$$

- Hybrid?

Technical Details

Instrumentation

- Replace sources with labelled constants.

- For values that are tainted by more than one source (for example $S_1 + S_2$) replace with one of two uninterpreted functions over the sources:

1. $\mathit{preserving}(s_1, s_2, \ldots, s_n)$. For example $+$, XOR, etc.
2. $\mathit{nonPreserving}(s_1, s_2, \ldots, s_n)$. For example left or right shift, etc.

- Perform taint analysis on sources to generate environment Γ which marks statements involving tainted variables.

Sequential Product Program

- Most straightforward
- Hard for CPAChecker to reason about

Synchronized Product Program

- Easiest for CPAChecker to reason about

- Exponential blowup

$$\frac{}{A_1 \otimes A_2 \rightsquigarrow A_1 \ ; \ A_2}$$

$$\frac{S_2 \otimes S_1 \rightsquigarrow P}{S_1 \otimes S_2 \rightsquigarrow P}$$

$$\frac{S_1 \otimes S \rightsquigarrow S_1' \quad S_2 \otimes S \rightsquigarrow S_2' \quad P = \mathit{if}\,(p)\ \mathit{then}\ S_1'\ \mathit{else}\ S_2'}{\mathit{if}\,(p)\ \mathit{then}\ S_1\ \mathit{else}\ S_2 \otimes S \rightsquigarrow P}$$

$$\frac{P_0 = \mathit{while}(p_1 \wedge p_2)\ S_1\ ;\ S_2 \quad P_1 = \mathit{while}(p_1)\ S_1 \quad P_2 = \mathit{while}(p_2)\ S_2}{\mathit{while}(p_1)\ S_1 \otimes \mathit{while}(p_2)\ S_2 \rightsquigarrow P_0\ ;\ P_1\ ;\ P_2}$$

Hybrid Product Program

- Based on key insight: don't reason precisely about unrelated parts of the program

- "Unrelated" if not tainted. Use environement Γ and add following inference rule:

$$\frac{\begin{array}{c}\Gamma \nvdash S_1 \\ \Gamma \nvdash S_2\end{array}}{\Gamma \vdash S_1 \otimes S_2 \rightsquigarrow S_1 \ ; \ S_2}$$

Assertions + CPAChecker

For every variable v that is tainted in a statement s that is marked as a sink, insert an assertion:

$$assert(v_1 == v_2)$$

Recall we replaced sources with labelled constants and propagated them, so this will be asserting the taintsets of the two variables are equivalent

CPAChecker returning TRUE means that V_2 is correct modulo V_1

Future Work

Push Button Implementation

Evaluation

Evaluate conjectures about differences between three constructions of the product programs.

An expirement comparing the three approaches will determine which one works best in practice.

Acknowledgements

Prof. Hovav Shacham and Prof. Isil Dillig

Thank you!