Symbolic Execution for Simple R

HARVARD UNIVERSITY

Laura Zharmukhametova Gian Pietro Farina Marco Gaboardi Stephen Chong

Simple R

R is a language for statistical analysis, machine learning, and data visualization. Despite its stable and rapid growth over the years, we still don't have an instrument for rigorous inspection and analysis of R programs.

Since the behavior of R can be fairly ambiguous, it's difficult to identify and fix vulnerabilities of a given program.

Simple R is a subset of R. It supports the core R operations, but excludes programs that use unsafe and hard-to-analyze operations such as super-assignments.

Approach

The symbolic execution engine executes a given Simple R program symbolically, finding and reporting errors of a specified kind. It also produces a concrete test case for a discovered error.

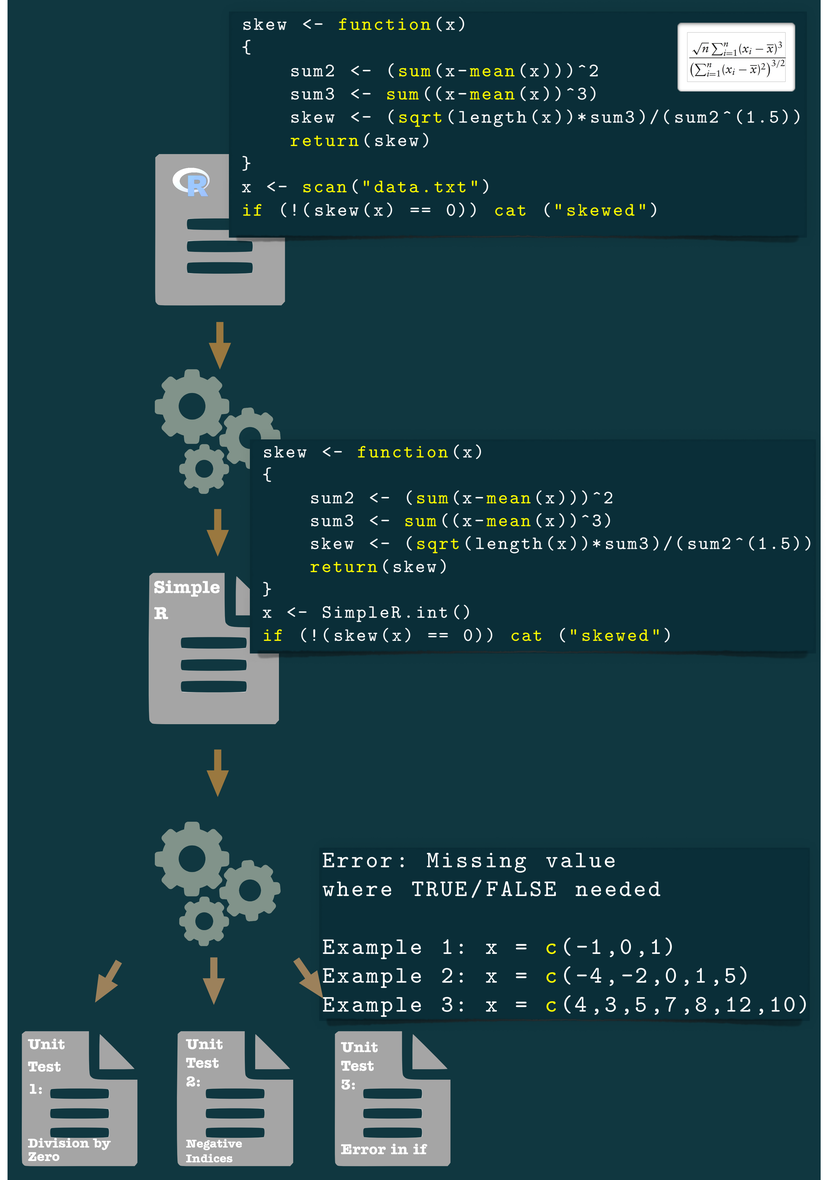
$$\textsc{IfTrue}\ \frac{H(m_v) = (\mathsf{true}, A_v)}{\langle(\mathbb{C}[\mathsf{if}\ m_v\ \mathsf{then}\ e_t\ \mathsf{else}\ e_f], m_\Gamma) : S; H; C\rangle \Longrightarrow \langle(\mathbb{C}[e_t], m_\Gamma) : S; H; C\rangle}$$

$$\textsc{IfFalse}\ \frac{H(m_v) = (\mathsf{false}, A_v)}{\langle(\mathbb{C}[\mathsf{if}\ m_v\ \mathsf{then}\ e_t\ \mathsf{else}\ e_f], m_\Gamma) : S; H; C\rangle \Longrightarrow \langle(\mathbb{C}[e_f], m_\Gamma) : S; H; C\rangle}$$

$$\textsc{IfSymbolic}\ \frac{H(m_v) = (X_{\mathrm{bool}}, A_v) \quad C' = C \cup \{X_{\mathrm{bool}} = \mathsf{true}\} \quad C'' = C \cup \{X_{\mathrm{bool}} = \mathsf{false}\}}{\langle(\mathbb{C}[\mathsf{if}\ m_v\ \mathsf{then}\ e_t\ \mathsf{else}\ e_f], m_\Gamma) : S; H; C\rangle \Longrightarrow \{\langle(\mathbb{C}[e_t], m_\Gamma) : S; H; C'\rangle, \langle(\mathbb{C}[e_f], m_\Gamma) : S; H; C''\rangle\}}$$

$$\textsc{DivSymbolic}\ \frac{\begin{array}{c} m_v\ \text{fresh} \quad Z_{\mathrm{int}}\ \text{fresh} \quad H(m_{v_1}) = (v_1, A_{v_1}) \quad H(m_{v_2}) = (v_2, A_{v_2}) \\ C' = C \cup \{Z_{\mathrm{int}} = v_1/v_2\} \cup \{v_2 \neq 0\} \quad H' = H[m_v \mapsto (Z_{\mathrm{int}}, A)] \\ C'' = C \cup \{v_2 = 0\} \quad H'' = H[m_v \mapsto (\mathsf{DivError}, \emptyset)] \end{array}}{\langle(m_{v_1}/m_{v_2}, m_\Gamma); H; C\rangle \hookrightarrow \{\langle m_v; H'; C'\rangle, \langle m_v; H''; C''\rangle\}}$$

The Symbolic Execution Engine



Challenges

We used SMT-LIB to represent symbolic vectors and path constraints. Symbolic vectors are placeholders for usual R vectors, which are the basic objects of R.

Finding a certain kind of error in a unit of R code boils down to solving Satisfiability Modulo Theories formulae using a theorem prover.

```
- (forall ((a (Array s1 s2)) (i s1) (e s2))
    (= (select (store a i e) i) e))

- (forall ((a (Array s1 s2)) (i s1) (j s1) (e s2))
    (=> (distinct i j)
            (= (select (store a i e) j) (select a j))))

- (forall ((a (Array s1 s2)) (b (Array s1 s2)))
    (=> (forall ((i s1)) (= (select a i) (select b i)))
            (= a b)))
```

Boolean, Integers and Real theories are directly supported in SMT-LIB v.2, while symbolic vectors were implemented using the theory of arrays. Thus, an X Y Array in SMT-LIB is a map from X to Y, where X,Y can be Int, Real or Boolean.

Future Work

Make a user-friendly tool

Relational Symbolic Execution

Contact

lzharmukhametova@college.harvard.edu

The 4th NSF Secure and Trustworthy Cyberspace Principal Investigator Meeting
October 28-29, 2019 I Alexandria, Virginia