Symbolic Execution in Selfie -a first step
Clement Poncelet, Christoph Kirsch

To cite this version:
Clement Poncelet, Christoph Kirsch. Symbolic Execution in Selfie -a first step. Klee Workshop 2018, Apr 2018, Londres, United Kingdom. hal-01851706

HAL Id: hal-01851706
https://hal.science/hal-01851706
Submitted on 30 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Symbolic Execution in Selfie  
- a first step -

Clément Poncelet, Christoph Kirsch  
cponcelet@cs.uni-salzburg.at, ck@cs.uni-salzburg.at

**INTRODUCTION**

Our project consists in designing and implementing a low-level symbolic emulator following selfie’s philosophy: simplicity. More precisely, we tackle the question of how to implement a fully, sound and complete symbolic emulator capable of deducing properties for any machine code sequence given in input. Our first goal being to have no loss of information for at least the selfie code and define clearly the subset of RISC-U language it imply. This particular challenge is made possible thank to the simple languages handled by this independent software, alleging and preventing us from the usual assumptions in numerical analysis. All in all, we aim at using selfie’s self-property to symbolically verify the selfie compilation and emulation.

**LANGUAGES**

<table>
<thead>
<tr>
<th>High-level</th>
<th>RISC-U: Low-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>C*</td>
<td>RISC-U</td>
</tr>
<tr>
<td>a tiny subset of C, supports only two data types uint64_t and uint64_t*</td>
<td>a tiny subset of 64-bit RISC-V, contains 14 instructions for unsigned arithmetic only.</td>
</tr>
</tbody>
</table>

**SELFIE**

Selfie® is an independent 64-bit implementation of (1) a self-compiling compiler written in C* targeting RISC-U, (2) a self-executing RISC-U emulator, and (3) a self-hosting hypervisor that virtualizes the emulated RISC-U machine. Selfie is implemented in a single 8k-line file and can compile, execute, and virtualize itself any number of times. Selfie has originally been developed for educational purposes but has recently become a research platform as well.

"http://selfie.cs.uni-salzburg.at

**SYMBOLIC EXECUTION**

The symbolic execution emulates a given RISC-U binary, analyzing numerically the instructions over abstract domains and operations. A current version implements w-intervals and initializes read system calls’ return value as every possible value. The implementation logs each emulated instruction in a trace recording a triple ⟨inst, prev, value⟩ of the instruction binaries, the previous destination register value and the new abstract one. Moreover, each register and memory records the trace index of its last store. This data-structure allows us to apply constraints and back-propagation algorithms when the branch on equal (beq) instruction is emulated by simply moving backward or forward through the trace. An abstract domain for a complete and sound numerical analysis is still in investigation for the selfie binary, its simplicity allows us to avoid the usual over-approximations and fit exactly the possible values of each variable for all paths.

**LOGGING TRACE**

<table>
<thead>
<tr>
<th>tc</th>
<th>pc</th>
<th>prev</th>
<th>v_start</th>
<th>v_end</th>
<th>def</th>
<th>use</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>n</td>
<td>n+1</td>
<td>n+2</td>
<td>n+3</td>
<td>n+4</td>
<td>n+5</td>
</tr>
<tr>
<td>...</td>
<td>n+2</td>
<td>n+3</td>
<td>n+4</td>
<td>n+5</td>
<td>n+6</td>
<td>n+7</td>
</tr>
<tr>
<td>...</td>
<td>MAX</td>
<td>2</td>
<td>MAX</td>
<td>MAX</td>
<td>MAX</td>
<td>10</td>
</tr>
</tbody>
</table>

[Diagram of symbolic execution and logging trace]