Introduction

Verification of large data structures is beyond the current state of the art for finite state model checking.

We present a new technique, S\textsuperscript{2}W, for the verification of safety properties on large or unbounded data structures.

Motivation

Hypervisors and CPU emulators are used in a variety of security-critical applications:

- Cloud computing
- Malware analysis
- Hosting dangerous applications

Verifying the isolation properties provided by virtualization often involves reasoning about large data structures:

- Page tables
- Translation lookaside buffers (TLB)
- Caches

Heuristics

In general, computing a bound \(k\) on the reachability diameter is undecidable for our class of systems. We develop two heuristics for computing \(k\).

Evaluation

We use case studies to evaluate the practicality of using S\textsuperscript{2}W for real-world applications.

We successfully verify safety properties for six systems:

- **Bochs’ TLB:** Address translation optimization is correct
- **Content Addressable Memory-based Cache:** Cache optimization is correct
- **Shadow Page Tables:** Guest/host isolation
- **SecVisor:** Only approved code executes in kernel mode
- **sHype:** Chinese Wall policy is enforced
- **ShadowVisor:** Guest/host isolation

<table>
<thead>
<tr>
<th>Case Study</th>
<th>(k)</th>
<th>Computing (k)</th>
<th>BMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bochs’ TLB</td>
<td>8 steps</td>
<td>12 hours</td>
<td>23 hours</td>
</tr>
<tr>
<td>CAM-based Cache</td>
<td>2 steps</td>
<td>1 second</td>
<td>1 second</td>
</tr>
<tr>
<td>Shadow Page Tables</td>
<td>4 steps</td>
<td>1 minute</td>
<td>5 seconds</td>
</tr>
<tr>
<td>SecVisor</td>
<td>Verification completed with induction proof</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sHype</td>
<td>Verification completed with induction proof</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShadowVisor</td>
<td>Verification completed with induction proof</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S\textsuperscript{2}W

Our contribution: a semi-decision procedure for verifying property \(\Phi\) is an invariant of system \(S\): \(S \models \Leftrightarrow \Phi\).

S\textsuperscript{2}W is a three-step procedure:

1. **Standard Mathematical Induction**
   - If \(\Phi\) is an invariant, return True.
   - Else, continue to step 2.

2. **Small World**
   - Create \(\hat{S}\), a scaled down model with most data structure members abstracted away.
   - Abstraction makes state space manageable.

3. **Short World**
   - Compute a bound \(k\) on the reachability diameter of \(\hat{S}\).
   - Use bounded model checking to prove the invariant on the abstracted system: \(\hat{S} \models \Leftrightarrow \Phi\).
   - Using \(k\) as the bound for BMC makes the verification sound.

Evaluation

We use case studies to evaluate the practicality of using S\textsuperscript{2}W for real-world applications.

We successfully verify safety properties for six systems:

- **Bochs’ TLB:** Address translation optimization is correct
- **Content Addressable Memory-based Cache:** Cache optimization is correct
- **Shadow Page Tables:** Guest/host isolation
- **SecVisor:** Only approved code executes in kernel mode
- **sHype:** Chinese Wall policy is enforced
- **ShadowVisor:** Guest/host isolation