RedLeaf: Isolation and Communication in a Safe Operating System

By Narayanan et al.
Presented by Alex Appel

Note: Slides Adapted from OSDI ‘20 Presentation
Agenda

- Introduction/Background
- RedLeaf Implementation
- RedLeaf Evaluation
- Conclusion
- Questions
Isolation in Operating Systems

- Isolation of kernel subsystems is a critical mechanism
- Systems remained monolithic
  - Isolation was expensive
- Hardware Isolation
- Language-based Isolation
Traditional Safe Languages vs. Rust
Language-based Isolation: Rust

- Other projects mostly use Rust as a drop-in replacement for C
- Numerous possibilities
  - Fault Isolation
  - Transparent Device-Driver Recovery
  - Safe Kernel Extensions
  - Fine-grained capability-based access control
Fault-Isolation in Language-Based Systems

- Unfortunately, built-in language mechanisms alone are not sufficient
  - Crash of a single component could leave the entire system in a corrupted state
- Need a mechanism that *isolates faults*
- Provide a way to terminate a faulting or misbehaving computation leaving system in a clean state
  - 1) Deallocate all resources that were in use by the subsystem
  - 2) Preserve the objects that were allocated by the subsystem but then passed to another subsystem
  - 3) Ensure that all future invocations do not violate safety or block the caller
Redleaf Fault Isolation

- Crash occurs when one of the threads that enters the domain panics
- Fault is isolated if
  - 1) We can unwind all threads running inside the crashing domain to the domain entry point and return an error to the caller
  - 2) Subsequent attempts to invoke the domain return errors but do not violate safety guarantees or result in panics
  - 3) All resources of the crashed domain can be safely deallocated, and we can reclaim all resources owned by the domain without leaks
  - 4) Threads in other domains continue execution, and can continue accessing objects that were allocated by the crashed domain, but were moved to other domains before the crash
RedLeaf Key Principles

- **Heap Isolation**
  - Since no other domains hold pointers into the private heap of a crashing domain, it’s safe to deallocate the entire heap

- **Exchangeable Types**
  - Objects allocated on the shared heap cannot have pointers into private domain heaps, but can have references to other objects on the shared heap

- **Ownership Tracking**
  - Keep track of ownership for all objects on the shared heap

- **Interface Validation**
  - Allow domain authors to define custom interfaces while retaining isolation

- **Cross-Domain Call Proxying**
  - All cross-domain invocations are mediated with invocation proxies
Evaluation: System Setup

- 2 x Intel E5-2660 v3 10-core CPUs at 2.60 GHz (Haswell EP)
- Disabled: Hyper-Threading, Turbo Boost, CPU Idle states
- Linux and DPDK benchmarks run on version 4.8.4
- RedLeaf benchmarks run on baremetal
Evaluation: Language-Based Isolation vs Hardware Mechanisms

<table>
<thead>
<tr>
<th>Operation</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>seL4</td>
<td>834</td>
</tr>
<tr>
<td>VMFUNC</td>
<td>169</td>
</tr>
<tr>
<td>VMFUNC-based call/reply invocation</td>
<td>396</td>
</tr>
<tr>
<td>RedLeaf cross-domain invocation</td>
<td>124</td>
</tr>
<tr>
<td>RedLeaf cross-domain invocation (passing an <code>RRef&lt;T&gt;</code>)</td>
<td>141</td>
</tr>
<tr>
<td>RedLeaf cross-domain invocation via shadow</td>
<td>279</td>
</tr>
<tr>
<td>RedLeaf cross-domain via shadow (passing an <code>RRef&lt;T&gt;</code>)</td>
<td>297</td>
</tr>
</tbody>
</table>

Table 1: Language-based cross-domain invocation vs hardware isolation mechanisms.
Evaluation: Language Overheads (C vs Rust)

Figure 5: C vs Rust performance comparison
Evaluation: Ixgbe Driver Comparison
Evaluation: Application Benchmark (Maglev Load Balancer)
Conclusion

- Heap Isolation, Exchangeable Types, Ownership Tracking, Interface Validation, Cross-Domain Call Proxying
- Provides a collection of mechanisms for enabling isolation
- A step forward in enabling future system architectures
  - Secure kernel extensions, fine-grained access control, transparent recovery etc.
Resources

Paper: RedLeaf: Isolation and Communication in a Safe Operating System

RedLeaf Site: https://mars-research.github.io/redleaf

RedLeaf Source Code: https://github.com/mars-research/redleaf