Hardware Enforcement of Application Security Policies using Tagged Memory

Nickolai Zeldovich^, Hari Kannan*, Michael Dalton*, Christos Kozyrakis*

^MIT  *Stanford University
Motivation

- Systems do not provide good abstractions for security
  - Applications forced to build own security mechanisms
  - Software managing security bloats TCB greatly

- TCB for large SW systems is in the millions of LOC
  - Written by multiple developers in different companies
  - Difficult to eliminate bugs or verify correctness
VMs don’t solve the problem

- Virtualization used for minimizing OS TCB
- Good for partitioning resources, applications
- But apps that require sharing must run in same VM
  - Virtualization provides no benefit
Current HW also inadequate

- Kernel data structures require fine-grained protection
  - struct proc {
    struct proc *next;
    pid_t pid;
    char proctitle[64];
    struct inode *cwd;
  }

- Page-aligning data structures is complex and expensive

- Paging does not associate policy with **physical** resources
  - Translation mechanisms subject to **aliasing**
Our proposal: Tagged Memory

- Tagged memory:
  - Each word of physical memory associated with a 32-bit tag
  - Tags map to access permissions (R/W/X) for prot. domain
  - Fine-grained access control

- Simplifies security enforcement
  - SW manages tags, but HW enforces security policies
  - Helps maintain security in face of compromised OS

- Ties security policies to physical resources
  - Physical resource policies avoid ambiguity

- Allows for a smaller TCB (need to trust less software)
Outline

- Motivation & Tagged Memory Overview

- Software Architecture
  - Enforcing and Managing Security Policies with tags

- Full-system FPGA Prototype
  - Hardware Design
  - Experiments

- Conclusion
High-level system structure

- Complex kernel enforces security and implements software abstractions
- Monitor labels resources
- Hardware enforces security
- Kernel provides abstractions, but no longer fully trusted
Prototype: LoStar (Loki + HiStar)

- **Based on HiStar OS**
  - Worst-case scenario: kernel already very small, $O(10,000 \text{ LOC})$
  - Key feature: OS interface consists of few types of **objects**
  - Applications specify security policies in terms of **labels** on objects
  - Kernel uses labels to control **read/write access**

- **Security monitor**: translates HiStar labels into hardware tags
  - Maintains separate tag permissions for every executing context
  - Controls access to shared memory using tags
  - Protects important kernel data structures

- **Loki**: tags for fine-grained permissions on physical memory
  - Tag permission cache in hardware populated by monitor
LoStar enforces application policies

Application security policy enforced in hardware
Controlled sharing in LoStar

Alice's thread

Alice private

Alice public_html

Alice & Bob Shared

Bob's thread
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Kernel

Monitor

Alice private

Alice public_html

Alice & Bob Shared
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>--</td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Kernel

Monitor

Alice private

Alice public_html

Alice & Bob Shared
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>RW</td>
</tr>
</tbody>
</table>

Kernel

Monitor

Alice private

Alice public_html

Alice & Bob Shared

TRAP
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Alice private

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>--</td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Alice & Bob Shared

Monitor

Kernel

Kernel
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Kernel

Monitor

Alice private

Alice public_html

Alice & Bob Shared
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Kernel

Monitor

Alice private

Alice public_html

Alice & Bob Shared

Id alice_data
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Kernel

Monitor

Alice private

Alice & Bob Shared

Id alice_data

OK

Alice public_html
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Kernel

Monitor

Alice private

Alice public_html

Alice & Bob Shared
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>RW</td>
</tr>
</tbody>
</table>

Kernel

Monitor

Alice private

Alice public_html

Alice & Bob Shared
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

Kernel
Monitor
Alice private
Public html
TRAP
Alice & Bob
Shared
Controlled sharing in LoStar

<table>
<thead>
<tr>
<th>Tags</th>
<th>Perm</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

```
st alice_thread
```

Kernel

Monitor

Alice private

Alice public_html

Alice & Bob Shared
Controlled sharing in LoStar

Monitor provides “yield-to” primitive for thread collaboration
Controlled sharing in LoStar

Create super-privileged thread

Kernel

Monitor

Kernel
Controlled sharing in LoStar

Create super-privileged thread

Kernel

Monitor

Kernel
Controlled sharing in LoStar

Create super-privileged thread

Kernel

Monitor

Yield to

Kernel
Controlled sharing in LoStar

Create super-privileged thread

Kernel

TRAP

Yield to

Monitor

Monitor provides operations to
- Create protection domain
- Change privileges
Controlled sharing in LoStar

Kernel

Monitor

Kernel
Controlled sharing in LoStar

Kernel

Give mem page to Alice

Monitor

Kernel

PAGE 1
Controlled sharing in LoStar

Accept mem page

Kernel

Kernel

Monitor

PAGE 1
Controlled sharing in LoStar

Kernel

Update mem_page

Monitor

PAGE 1
Controlled sharing in LoStar

Kernel

Monitor

Update mem_page

TRAP

PAGE 1
Controlled sharing in LoStar

Update mem_page

Kernel

Kernel

Monitor

TRAP

PAGE 1

Alice’s memory allocator sets tags on physical memory
Controlled sharing in LoStar

Kernel

ID 123

Reference

Monitor

ID 123
RefCnt: 1

Alice file
Controlled sharing in LoStar

Kernel

ID 123

Reference

Monitor

ID 123

RefCnt: 1

Alice file

Change Ref-count
Controlled sharing in LoStar

Special ref-count scheme using fine-grained tags

Kernel

Reference

ID 123

Monitor

Alice file

ID 123

Change Ref-count

TRAP
Controlled sharing in LoStar

Special ref-count scheme using fine-grained tags

Create Reference

Kernel

Reference

ID 123

Monitor

ID 123
RefCnt: 2

Alice file

ID 123
Controlled sharing in LoStar

Special ref-count scheme using fine-grained tags

Delete Reference

Kernel

ID 123
Reference

Monitor

ID 123
RefCnt: 2
Alice file

ID 123

Kernel
Controlled sharing in LoStar

Kernel

ID 123
Reference

Monitor

ID 123
RefCnt: 1
Alice file

Delete Reference
Controlled sharing in LoStar

Kernel

TRAP

Reference

Monitor

ID 123
RefCnt: 1
Alice file

Delete Reference
Controlled sharing in LoStar

References protected using fine-grained tags

Garbage collection:
- Separate idle “GC” protection domain for each object
- GC code thus runs outside of the monitor (TCB)
Controlled sharing in LoStar

H = 23
Object ID hash table

ID 223
Bob’s file

ID 123
Alice file

Monitor

Kernel

Kernel
Controlled sharing in LoStar

St ID123 in hashmap

Kernel

Monitor

ID 223
Bob’s file

ID 123
Alice file

H = 23

Object ID hash table
Controlled sharing in LoStar

Monitor sets read-only tags on objectID, and linked list pointers. Need fine-grained protection.
Controlled sharing in LoStar

Monitor sets read-only tags on objectID, and linked list pointers need **fine-grained** protection.
Monitor vs. Kernel

- **Monitor operations:**
  - Uses tags for enforcing security
  - Manipulating protection domains
  - Context-switching between protection domains
  - Manipulating memory

- Kernel still in charge of scheduling, translation etc.
Kernel security mechanisms

- So what does the kernel do?
  - Monitor provides read-write memory protection
  - However, monitor cannot ensure liveness
  - Monitor cannot prevent covert channels
    - In LoStar, HiStar’s kernel can provide stronger security
    - Uses information flow to prevent covert channels
LoStar Prototype System

- HW: modified SPARC processor (Leon3)
  - In-order, 7 stage pipeline
  - Mapped to FPGA board

- Loki logic overhead: 19%
  - Much lower fraction in more complex CPUs

- Loki clock frequency overhead: none
- Instruction/data caches extended to hold tag bits

- Permissions cache accessed in two cases
  - On instruction fetch using the instruction tag
  - On loads/stores using the data tag

- Exceptions invoke security monitor
Loki Tag Storage

- Fine-grained permission for physical memory
- Simple approach: +32 bits/word in caches and memory
  - 100% storage overhead
- Multi-granular tag storage scheme [Suh’04]
  - Exploit tag similarity to reduce storage overhead
  - Page-level tags $\Rightarrow$ word-level tags

<table>
<thead>
<tr>
<th>Physical Memory</th>
<th>Page-Tag Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Frame 1</td>
<td>Tag for Page 1</td>
</tr>
<tr>
<td>Page Frame 2</td>
<td>Tag for Page 2</td>
</tr>
<tr>
<td></td>
<td>Tag for Page 3</td>
</tr>
<tr>
<td></td>
<td>Tag for Page 4</td>
</tr>
</tbody>
</table>
Loki Tag Storage

- Fine-grained permission for physical memory
- Simple approach: +32 bits/word in caches and memory
  - 100% storage overhead
- Multi-granular tag storage scheme [Suh’04]
  - Exploit tag similarity to reduce storage overhead
  - Page-level tags ⇒ word-level tags

![Diagram of Loki Tag Storage]

- Physical Memory
  - Page Frame 1
  - Page Frame 2
  - Tag Page Frame

- Page-Tag Array
  - Tag for Page 1
  - Tags for Page 2
  - Tag for Page 3
  - Tag for Page 4
Monitor Mode

- Updates of tags and permission cache entries
  - Requires new operating mode

<table>
<thead>
<tr>
<th>Role</th>
<th>Access Details</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Limited instructions; limited address ranges; VM</td>
<td>VM</td>
</tr>
<tr>
<td>Kernel</td>
<td>Access to all instructions &amp; address ranges; VM/PM</td>
<td>VM/PM</td>
</tr>
<tr>
<td>Monitor</td>
<td>Access to all instructions &amp; address ranges; PM</td>
<td>PM</td>
</tr>
</tbody>
</table>
Monitor Mode

- Updates of tags and permission cache entries
  - Requires new operating mode

<table>
<thead>
<tr>
<th>Role</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Tags are transparent</td>
</tr>
<tr>
<td>Kernel</td>
<td>Can read tags</td>
</tr>
<tr>
<td>Monitor</td>
<td>Can read/write tags</td>
</tr>
</tbody>
</table>

- On permission check failure/permission cache miss
  - Switch to monitor mode & jump to security monitor
  - Disable processor’s MMU
    - This helps remove MMU handling code from the OS’ TCB
## TCB Reduction

- **TCB reduction**

<table>
<thead>
<tr>
<th>LOC</th>
<th>HiStar</th>
<th>LoStar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>11,600 (trusted)</td>
<td>12,700 (untrusted)</td>
</tr>
<tr>
<td>Bootstrapping</td>
<td>1,300</td>
<td>1,300</td>
</tr>
<tr>
<td>Security Monitor</td>
<td>N/A</td>
<td>5,200 (trusted)</td>
</tr>
<tr>
<td><strong>TCB size: Trusted code</strong></td>
<td>11,600</td>
<td>5,200</td>
</tr>
</tbody>
</table>

- **TCB reduction: from 12K LOC to 5K LOC**
- **Currently RAMDisk system**
  - Could use flash to store tags for disk sectors
LoStar Performance Evaluation

- Negligible performance impact
Need for fine-grained tags

- Fine-grained tags are necessary
- Need many tags to protect users, processes, file descriptors
Conclusions

- **Loki hardware architecture using tagged memory**
  - Enforces application security policies on data

- **LoStar = Loki + HiStar**
  - Provides fine-grained access control in hardware
    - Many independent protection domains
  - Protects user data from other malicious kernels
  - Reduces the TCB of HiStar by over a factor of 2
  - Provides strong security guarantees and good performance
Questions?

☐ Want to use LoStar?
  • Let us know …
  • Loki port to Xilinx XUP board
    ▪ $300 for academics
    ▪ $1500 for industry
  • Full RTL + HiStar-SPARC distribution