L4.sec Implementation
Kernel Memory Management

Bernhard Kauer
kauer@os.inf.tu-dresden.de

Diplomverteidigung 2005-05-20
Motivation
Why L4.sec?
Motivation
Why L4.sec?

L4 does not have a security model and hence needs one.

Hermann Härting

Lack of security

- kernel-resource management
- IPC control
New Concepts
L4.sec - a new L4 API?

aim
Build an improved L4 that allows to
  • implement access control
  • manage kernel memory
  • flexible communication
New Concepts
L4.sec - a new L4 API?

aim
Build an improved L4 that allows to
- implement access control
- manage kernel memory
- flexible communication

Capabilities
Kernel-Memory objects
Endpoints
New Concepts
L4.sec - a new L4 API?

**aim**

Build an improved L4 that allows to

- implement access control
- manage kernel memory
- flexible communication

**Capabilities**

**Kernel-Memory objects**

**Endpoints**

**previous state**

- L4.sec ideas
- but: Implementation missing
Capability
What are capabilities?
Capability
What are capabilities?

**definition**

\[ \text{Capability} = \text{Object} + \text{Permissions} \]

- **Objects**  thread, task, endpoint, kernel-memory object, ... 
- **Permissions**  object dependent rights like read, write or send, recv
Capability
What are capabilities?

definition

Capability = Object + Permissions

Objects  thread, task, endpoint, kernel-memory object, ... 
Permissions object dependent rights like read, write or send, recv

capability space

- translate task local number to capabilities
- extension of the recursive address space model
  - map
  - unmap
Kernel-Memory Objects
How to handle kernel memory?
Kernel-Memory Objects
How to handle kernel memory?

requirements

- managing the used kernel-memory
- no predefined amount of kernel memory
- map and unmap kernel-memory
Kernel-Memory Objects
How to handle kernel memory?

requirements
- managing the used kernel-memory
- no predefined amount of kernel memory
- map and unmap kernel-memory

solution
- creating kernel-memory objects *converts* user-pages
- converted pages are inaccessible
- objects are *created* with kernel-memory (objects)
- reconvert at *unmap* of user-page or kmem object
- delete all objects within it at reconvert
Delete Objects

Dependencies
Delete Objects

Dependencies

**delete named objects**

- last capability removed
- memory revoked
Delete Objects

Dependencies

delete named objects

- last capability removed
- memory revoked

⇒ have to delete all dependent objects
Delete Objects

Dependencies

delete named objects
- last capability removed
- memory revoked

⇒ have to delete all dependent objects

Dependencies
- Resource: Who provides the resources for this object?
- Functional: What needs this object to work?
- Map: Where it is mapped?
Dependencies

Example: Task
Avoid cycles

- directed dependency graph
- only a tree walk needed
- simplifies locking and implementation
Avoid cycles
- directed dependency graph
- only a tree walk needed
- simplifies locking and implementation

Solutions
- **Bind** thread to task
- **Collocate** named object and mapping node
- **Redirection** change the pointers
The W5 binding
A syscall binding for L4.sec
The W5 binding
A syscall binding for L4.sec

**idea**

- parameters in virtual registers
- param words as abstraction for in- and output-parameters
- up to 255 param words could be used
- transfer at least 5 param words in registers
The W5 binding  
A syscall binding for L4.sec

### Idea

- parameters in virtual registers
- param words as abstraction for in- and output-parameters
- up to 255 param words could be used
- transfer at least 5 param words in registers

### Properties

- used with iret, sysenter, syscall
- architecture independently
- fast and simple
Current state
What is working?
Current state
What is working?

- Long-IPC
- fine-grained locking
- native ia32 and interrupts
Current state
What is working?

**missing**
- Long-IPC
- fine-grained locking
- native ia32 and interrupts

**working**
- send messages and page a task
- convert user-pages and create objects
- mapping capabilities and unmapping objects
Evaluation
Code complexity
## Evaluation

### Code complexity

Number of SLOC for the *UX*-versions:

<table>
<thead>
<tr>
<th>name</th>
<th>L4.V2</th>
<th>L4.sec</th>
<th>difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel</td>
<td>17805</td>
<td>14421</td>
<td>−3384</td>
</tr>
<tr>
<td>jdb</td>
<td>14656</td>
<td>3950</td>
<td>−9626</td>
</tr>
<tr>
<td>unit tests</td>
<td>400</td>
<td>1762</td>
<td>+1362</td>
</tr>
<tr>
<td>kernel memory and allocators</td>
<td>1521</td>
<td>972</td>
<td>−549</td>
</tr>
<tr>
<td>mapping database and spaces</td>
<td>1994</td>
<td>2614</td>
<td>+620</td>
</tr>
<tr>
<td>thread and syscalls</td>
<td>4347</td>
<td>2265</td>
<td>−2082</td>
</tr>
<tr>
<td>api</td>
<td>1067</td>
<td>889</td>
<td>−178</td>
</tr>
<tr>
<td>reused code</td>
<td>7287</td>
<td>6985</td>
<td>−302</td>
</tr>
<tr>
<td>others</td>
<td>1589</td>
<td>696</td>
<td>−893</td>
</tr>
</tbody>
</table>
Conclusion
The important thinks
The important thinks

experiences

Three quarter of the code is present
  • similar code complexity
  • fundamental change of Fiasco
  • it is feasible to build L4.sec
## Conclusion

The important thinks

### Experiences

- Three quarter of the code is present
  - similar code complexity
  - fundamental change of Fiasco
  - it is feasible to build L4.sec

### Conclusion

- problem
- idea
- implementation
- application?
Conclusion
The important thinks

**Experiences**

- Three quarter of the code is present
- Similar code complexity
- Fundamental change of Fiasco
- It is feasible to build L4.sec

**Conclusion**

- Problem
- Idea
- Implementation
- Application?
Appendix

Endpoint

What is an endpoint?

**communication channel**
- replaces direct IPC to a given thread
- many threads could send and receive from an endpoint
- works over task boundaries

**differences**
- sender or receiver wait at an endpoint
- only open wait at a single endpoint possible
- a call uses two different endpoints