A communication framework for distributed access control in microkernel-based systems

Mohammad Hamad, Johannes Schlatow, Vassilis Prevelakis, Rolf Ernst
Controlling Concurrent Change

http://ccc-project.org/
Overview

Computers on wheels
- 70-100 ECU
- More than 10 millions LOC

Within each ECU
- Multi-levels of security and criticality
  - Different vendors
  - Different security perspective
- Protecting applications from each others
  - Toyota “EnForm” system!

Microkernel is the first step
- Minimum privileged code and TCB
- Inter-process communication (IPC)
Overview

Network on Wheels
- ECUs connected using many bus systems
  - CAN, IP based for on-board
  - Wireless
- Uncontrolled interaction may cause a vulnerabilities
  - Entertainment system attack!

Who should talk to whom?
Contribution

Providing **distributed access control framework**

- Controlling **“who should talk to whom”**
- Providing **security services** (i.e. Integrity, mutual authentication, and confidentiality)
- **Distributed** policy enforcement points
Outline

- Communication scenario
- Framework architecture
- Design and implementation
- Evaluation
- Conclusion
Outline

- Communication scenario
- Framework architecture
- Design and implementation
- Evaluation
- Conclusion
Communication Scenario

ECU1

C3 -> C4

C5

ECU2

C2

C6

network

local
remote
(Security) objectives

<table>
<thead>
<tr>
<th>Objective</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-grained access control</td>
<td>✔️</td>
<td>?</td>
</tr>
<tr>
<td>Integrity, mutual authentication, and confidentiality</td>
<td>❓</td>
<td>❓</td>
</tr>
<tr>
<td>Legacy application support</td>
<td>❓</td>
<td>?</td>
</tr>
<tr>
<td>Composability and migratability</td>
<td>❓</td>
<td>?</td>
</tr>
<tr>
<td>Minimum (application-specific) TCB</td>
<td>❓</td>
<td>?</td>
</tr>
</tbody>
</table>

**Local communication**

- IPC and Capability-based access control enforced by policy engine
- Capability used to identify the application (authentication)

**From IPC towards networked communication**

- Controlling the direct access to the communication module
- Component needs capability and appropriate policy to use network
Outline

- Communication scenario
- Framework architecture
- Design and implementation
- Evaluation
- Conclusion
Framework Architecture

- User-level networking
- Dedicated stack per application
- Threats
  - Spoofing
  - DoS attack
- Layer 2 security

- For each ECU
  - Single Communication Module
  - Shared by all applications
  - Local firewall enforces part of the communication policy
  - Distributed Firewall
  - Layer 3 security
Outline

- Communication scenario
- Framework architecture
- Design and implementation
- Evaluation
- Conclusion
Design and implementation

- Communication Interface
- Policy Decision Module
- Network Stack
- Decision Repository
Design and implementation

Communication Interface

- Implementing the socket API calls as IPC calls
- Managing the shared memory between each application and Communication Module
- Checking the validity of the parameters
- Enforcing the Policy Decision Module results
Design and implementation

Policy Decision Module

- Monitoring the requests based on credentials and connection properties.
- KeyNote Trust management system
- KeyNote Policy definition language.
  - Application independent.
  - Delegation.
Design and implementation

Authorizer: Integrator_public_key
Licensees: Platform_public_key

Conditions: (Vendor_id == “ACME_INSTRUMENTS” && Src_device_name == “headlight_control”
&& Dst_device_name == “ambient_light_sensor” && Src_device_type == CONTROL_PLATFORM
&& Dst_device_type == LIGHT_SENSOR && Security_level >= SL_INTEGRITY) -> “ALLOW”

Signature: Integrator signature
Network Stack

- LwIP stack.
- Embedded IPsec.
  - Mutual authentication, integrity and confidentiality.
  - Rate limiting, queuing priority.
Design and implementation

Decision Repository
- Storing the decision rules (i.e. source IP, destination IP, security level, etc.)
- Improve the efficiency
(Security) objectives

<table>
<thead>
<tr>
<th>Objective</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-grained access control</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Integrity, mutual authentication, and confidentiality</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Legacy application support</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Composability and migratability</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Minimum (application-specific) TCB</td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>
Example

```c
int s = accept (...);

int accept (...){
    return comm_accept(...);
}

int comm_accept (...){
    Rule *rule_r = search_rule (...);
    if (rule_r->empty()) {
        if (verify(...)) {
            add_rule(...);
            return lwip_connect(...);
        } else return -1;
    } else return -1;
    If (rule_r->authorized())
        return lwip_connect(...);
    else return -1;
}
```
Outline

- Communication scenario
- Framework architecture
- Design and implementation
- Evaluation
- Conclusion
Evaluation

- implemented with Genode OS
- Compared to existing Genode OS
  - bridge with proxy-ARP

Source Lines Of Code (SLOC)

- Save 750 SLOC

<table>
<thead>
<tr>
<th>Module Part</th>
<th>SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comunication interface</td>
<td>500</td>
</tr>
<tr>
<td>Policy Decision module Interface</td>
<td>300</td>
</tr>
<tr>
<td>IPsec extension of the Network Stack</td>
<td>2000</td>
</tr>
<tr>
<td>Decision Repository</td>
<td>600</td>
</tr>
</tbody>
</table>
Evaluation

Latency

- Netperf tool
- Genode and Netperf runs on the same Linux machine
- Genode runs on Raspberry Pi, and Netperf runs on remote Linux machine

![Diagram showing network components and communication channels](image-url)
Evaluation

Our Module does not add extra overhead.
Outline

- Communication scenario
- Framework architecture
- Design and implementation
- Evaluation
- Conclusion
## Conclusion

<table>
<thead>
<tr>
<th>Objective</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-grained access control</td>
<td>Capability-based</td>
<td>Communication module</td>
</tr>
<tr>
<td>Integrity, mutual authentication, and confidentiality</td>
<td>Capability-based</td>
<td>IPsec</td>
</tr>
<tr>
<td>Legacy application support</td>
<td></td>
<td>Lib-comm</td>
</tr>
<tr>
<td>Composability and migratability</td>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td>Minimum (application-specific) TCB</td>
<td></td>
<td>?</td>
</tr>
</tbody>
</table>

Performance could be better

- Handle the copy operations.
- Cashing the credentials.