Security model for embedded systems using Smack*

Simple but secure

* Simplified Mandatory Access Control Kernel
Context

- José Bollo
- Eurogiciel
- Intel
- Tizen
- Smack
- Linux
Smack overview

- The author of Smack is mainly Casey Schaufler.
- In Linux since kernel 2.6.25 – 17 April 2008 – as a LSM (Linux Security Module)
- Evoluting since this first days.
- Inside Tizen since the first days (2012).
- Use extended file attributes to store data relating to files.
- Controlled via a filesystem interface: smackfs.
- Controls accesses of processes to files, IPC, sockets and processes (ptrace, signals, ...).
The Smack rules

- Smack's rules have 3 items:
  - the **subject's label**
  - the **object's label**
  - the **access**

This rule tells to allow **read**, **write** and **execute** access to objects labelled **User** for the processes labelled **System**.

What are labels?  What are subjects?  What are objects?  How to set?
The Smack vocabulary

- **Labels**: are just text (of valid ASCII characters) without any special meaning: they are compared to equality (case sensitive: a≠A).

- **Subjects**: are running processes: any running process has a smack label.

- **Objects**: are files, IPC, sockets, processes.

- The label of a running process is called its **context**.
  
  - The commands `id`, `ps` (option -Z or -M), `ls` (option -Z) are prompting the contexts of the current process, the running processes, the files.

- The grantables accesses are: **read** (r), **write** (w), **execute** (x), **append** (a), **lock** (l), **transmute** (t).
Setting Smack

• How to set context? **You can't!** Except if you have the capability **CAP_MAC_ADMIN**.

```
# chsmack --access label file
# echo -n label > /proc/$$/attr/current
```

• How to set rules? **You can only reduce accesses** for the current thread (inherited by cloning). But if you have the capability **CAP_MAC_ADMIN**, you can change all rules.

```
# echo "subject object rwt" > /sys/fs/smackfs/load-self2
# echo "subject object rwt" > /sys/fs/smackfs/load2
# echo "subject object rwt" > smackload
```
Targets devices

- mobile handsets
- In-vehicle infotainment (IVI)
- NUCs and boxes
- television
# Targets usages

<table>
<thead>
<tr>
<th></th>
<th>Single seat</th>
<th>Multi seats</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single user</strong></td>
<td>- handsets boxes</td>
<td></td>
</tr>
<tr>
<td><strong>Multi users</strong></td>
<td>- tablets - laptops NUC</td>
<td>IVI</td>
</tr>
</tbody>
</table>

Multi seats is meaning that several users are using the same system through several interfaces.
The installer enable the application and configure the system according to the manifest.
Installer + launcher model

Installed Applications (untrusted)

The installer enable the application and configure it according to the manifest.

The launcher prepare the environment in agreement with the manifest and launch the application.

Trusted System (installed, signed)

Installed Application with manifest

Trusted environment

Smack rules for authorised services

process
Tizen offers the possibility to install applications that are either natives or widgets (W3C compliant) or a mix of both.

Each application has potentially access to a wide variety of services.

The accessed services MUST be conform to what the manifest of the application is claiming for. That is the condition to have a trusted system, a secure system.
Implementations

• The problem is difficult due to its power characteristic: controlling $N$ resources for $M$ kinds of accesses brings to $M^N$ cases!

• For Tizen 2.0 there was many smack rules (for a basic mobile handset, not less than 33232 rules!)
  - Each application have a own context label
  - The rules are the spare matrix of all the authorised accesses

• For tizen 3.0 IVI the three-domains model will be used.
  - Basically, three subject labels exist: __, System and User
  - There few more object labels
  - The rules are restricted to the minimum
  - It requires a launcher to achieve the full control of accesses
Three-domains model overview

Base system: _

The floor domain provides the foundation upon which the system is built.

Services: System

- System::Run
- System::Shared
- System::Log

The System domain is comprised of the basic system services and the data they maintain.

Applications: User

The User domain is comprised of the services that interact directly with the person using the Tizen system and the data those services maintain.
Links

- LSM Smack
  - http://schaufler-ca.com/

- Smack utilities
  - https://github.com/smack-team/smack

- Tizen
  - https://www.tizen.org/
Summary

- It works well and is really simple to learn.
- You can activate it on any Linux kernel.
- The embedded linux distribution TIZEN implements it and its community can help you.
- You can contribute to improve the smack tools and models.
Questions

Thanks
EUROGICIEL

- Open source development and integration:
- Maintainers for tizen.org (Base, Test, Web Framework,... domains)
- Embedded systems for real-time multimédia:
  - Widi/Miracast stack,
  - Wayland/Weston,
  - Webkit2 browser with HW acceleration,
- Application: HTML5/CSS3, jquery, jqmobi, Cordova
- Location : Brittany – France
- http://www.eurogiciel.fr/
Evolutions of Smack

- The author of Smack is mainly **Casey Schaufler**.
- In Linux **since kernel 2.6.25** – 17 April 2008 – as a **LSM** (Linux Security Module)
- Evoluting since this first days.
  - Lock access mode (kernel 3.13)
  - Support for multi-rule write to load2 and change-rule (kernel 3.12)
  - Maximum value for CIPSO category change from 63 to 184 (kernel 3.12)
  - Longer Smack labels (24->255) and recursive transmute (kernel 3.5)
  - Transmute access mode (kernel 2.6.38)
Three-domains model rules

### Explicit rules 1/2

<table>
<thead>
<tr>
<th>Subject</th>
<th>Object</th>
<th>Rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>System::Run</td>
<td>rwxat</td>
</tr>
<tr>
<td>System</td>
<td>System::Shared</td>
<td>rwxat</td>
</tr>
<tr>
<td>System</td>
<td>User</td>
<td>rwx</td>
</tr>
<tr>
<td>System</td>
<td>^</td>
<td>rwxa</td>
</tr>
<tr>
<td>System</td>
<td>_</td>
<td>l</td>
</tr>
<tr>
<td>User</td>
<td>System</td>
<td>wx</td>
</tr>
<tr>
<td>User</td>
<td>System::Run</td>
<td>rwxat</td>
</tr>
<tr>
<td>User</td>
<td>System::Shared</td>
<td>rx</td>
</tr>
<tr>
<td>User</td>
<td>_</td>
<td>l</td>
</tr>
</tbody>
</table>

### Explicit rules 2/2

<table>
<thead>
<tr>
<th>Subject</th>
<th>Object</th>
<th>Rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>System</td>
<td>rwxa</td>
</tr>
<tr>
<td>^</td>
<td>System::Run</td>
<td>rwxat</td>
</tr>
<tr>
<td>_</td>
<td>System</td>
<td>wx</td>
</tr>
<tr>
<td>_</td>
<td>System::Run</td>
<td>rwxat</td>
</tr>
</tbody>
</table>

### Some implicit rules

<table>
<thead>
<tr>
<th>Subject</th>
<th>_</th>
<th>^</th>
<th>*</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>rwxatl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>rx</td>
<td>rwxatl</td>
<td>rwxatl</td>
<td>rx</td>
</tr>
<tr>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>rx</td>
<td>rwxatl</td>
<td>rwxatl</td>
<td>rwxatl if X=Y</td>
</tr>
</tbody>
</table>

Subject | Object | Rights
---|--------|--------

Explicit rules 1/2

Explicit rules 2/2

Some implicit rules
Security server

1. get cookie
2. cookie
3. request service with cookie
4. ask privilege of cookie
5. result (granted or not)
6. response of service

But there is no Smack here?!