Do you trust your OS? We don’t...

How to protect critical assets from a compromised OS?

Exploitation of an OS vulnerability might leave the system resources completely exposed to attacks. Hence, we need a mechanism (entity) that can be trusted even when the OS is breached.

Current TEE technologies

Intel software guard extension (SGX)
SGX enables to run secure services in an isolated execution environment (called enclave), embedded in the process virtual memory space.

ARM TrustZone
TrustZone provides a technology to run two virtual worlds – a Normal World (NW) and Secure World (SW) – on the same CPU. This mechanism is an infrastructure on which we can create a Trusted Execution Environment.

Our contribution – TROOS

Our goal is to provide a TEE for user services (much like Intel’s SGX). We use Genode as the basis for our trusted OS. On top of the native Core and Init components, which are the system kernel and first user process respectively, we added a few more components to enable trusted user services – trustlets – loading and execution.

Secure world interface

The normal world OS utilizes the secure world interface in order to create and execute trustlets according to its needs via SMCs. As part of our efforts to keep the system attack surface as small as possible, we keep the number of SMCs at the bare minimum. The most significant SMCs are described in the following table:

<table>
<thead>
<tr>
<th>SMC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tcreate</td>
<td>create a new (empty) trustlet</td>
</tr>
<tr>
<td>Tload</td>
<td>load and measure the trustlet code</td>
</tr>
<tr>
<td>Tinit</td>
<td>mark trustlet as ready to run</td>
</tr>
<tr>
<td>Tstart</td>
<td>start executing a trustlet</td>
</tr>
<tr>
<td>Tresume</td>
<td>resume trustlet run after it was stopped</td>
</tr>
<tr>
<td>Tdestroy</td>
<td>stop the trustlet run and free its resources</td>
</tr>
</tbody>
</table>

Future work

Our next step is to complete the system implementation with all of the designed components. We then plan to deliver an elaborated security analysis of the system and a comparison to existing TEEs (TrustZone based and others). The preference impact on the normal world will be tested as well. Due to time constraints, there are aspects that we do not plan to address at this stage, even though they surely are beneficial to TROOS. For example:

- Integrating a secure element.
- Extending system abilities with secure IO.
- Utilizing on chip memory to better protect trustlets code and data against probing.

Contact Information

Notice that in TROOS, a trustlet must not trust the other trustlets in the system. A trustlet doesn’t even need to trust the TROOS services, unless it wishes to use them.