PISTIS: Trusted Computing Architecture for Low-end Embedded Systems

Michele Grisafi
University of Trento, Italy
michele.grisafi@unitn.it

Mahmoud Ammar
Huawei Research, Germany
mahmoud.ammar@huawei.com

Marco Roveri
University of Trento, Italy
marco.roveri@unitn.it

Bruno Crispo
University of Trento, Italy
bruno.crispo@unitn.it
The issue at hand - embedded systems

The issue:

Embedded systems are at risk

The solution:

Security Services (e.g., Remote Attestation)

enabled by...

Trusted Execution Environments (TEEs)
The state of the art

Remote Attestation and TEEs

Features and Security guarantees

- PISTIS
- SpuV
- SBAP, SWATT, SAKE, SCUBA

HW resources and HW modifications

- SGX
- ARM TrustZone-M
- VRASED
- TyTAN
- TrustLite
- SANCUS
- SMART

- Hardware-based
- Minimal Hardware
- Software based

3
A TEE to fill the gap

- No hardware modifications
- No security hardware (e.g., MPU)
- DMA support
- Interrupts support
- Trusted Applications support
- Untrusted Toolchain

PISTIS
When to use Pistis

Trusted Execution Environment (TEE) with a set of Trusted Applications (TAs)

Low-end MCU in an embedded system, CPS, OT or IoT environment

Software-based remote adversary. DoS out of scope

Untrusted programmer with the source code of an application
PISTIS toolchain

Untrusted source files

Untrusted Application

Trusted Application

Runtime Library

Remote Attestation

Secure Code Update

MCU

Malware

Memory Isolation

Run-time attack

MitM

Corrupt toolchain

Inject malware

1

2

3

PISTIS

Untrusted Application

Runtime Library

Untrusted source files
Memory isolation - our policy

How?
1. Divide memory in regions
2. Deploy PISTIS and the application in different regions
3. Enforce Access Control Policy at runtime

Software instrumentation and virtualization

Make sure all of the instructions of the application are compliant with our Access Control Policy
A software-based approach

Custom untrusted toolchain

Replaces unsafe instructions with virtual calls to the TCB

Binary verification

Rejects applications with unsafe instructions

Run-time checks

Check safety of virtual calls

Sample code

CALL appFun
CALL R10

CALL appFun
MOVE R10, R4
CALL vrtCall

CALL appFun
MOVE R10, R4
CALL vrtCall

CALL appFun
MOVE R10, R4
CALL vrtCall

vrtCall: is *(R4) safe?

Compile bypassing instrumentation

Deploy

Run

Deploy

Run

Deploy

Run

STOP

CALL
## Performance evaluation

Evaluated on a TI MSP430F5529 MCU with a set of 13 embedded applications, including CPU, I/O and memory intensive operations, and an official TI benchmark.

<table>
<thead>
<tr>
<th>App</th>
<th>Memory Footprint</th>
<th>Runtime Overhead</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Orig.</td>
<td>Mod.</td>
<td>Orig.</td>
</tr>
<tr>
<td>SerialMSP</td>
<td>302 B</td>
<td>356 B (+17.88%)</td>
<td>334.1976 ms</td>
</tr>
<tr>
<td>CopyDMA</td>
<td>444 B</td>
<td>628 B (+41.44%)</td>
<td>118.4960 ms</td>
</tr>
<tr>
<td>XorCypher</td>
<td>247 B</td>
<td>475 B (+92.31%)</td>
<td>245.6500 ms</td>
</tr>
<tr>
<td>Bitcount</td>
<td>3684 B</td>
<td>5462 B (+48.26%)</td>
<td>5.7520 ms</td>
</tr>
<tr>
<td>SHA-256</td>
<td>1376 B</td>
<td>1546 B (+12.35%)</td>
<td>49.1888 ms</td>
</tr>
<tr>
<td>ML-acc</td>
<td>6174 B</td>
<td>9452 B (+53.09%)</td>
<td>1456.9092 ms</td>
</tr>
<tr>
<td>PrimeFactor</td>
<td>2192 B</td>
<td>3286 B (+49.91%)</td>
<td>4.0810 ms</td>
</tr>
<tr>
<td>32bitMath</td>
<td>522 B</td>
<td>766 B (+46.74%)</td>
<td>0.9310 ms</td>
</tr>
<tr>
<td>16bitSwitch</td>
<td>102 B</td>
<td>126 B (+23.53%)</td>
<td>0.0050 ms</td>
</tr>
<tr>
<td>8bitMatrix</td>
<td>844 B</td>
<td>860 B (+1.90%)</td>
<td>0.5760 ms</td>
</tr>
<tr>
<td>MatrixMul</td>
<td>500 B</td>
<td>516 B (+3.20%)</td>
<td>0.3430 ms</td>
</tr>
<tr>
<td>firFilter</td>
<td>3312 B</td>
<td>5430 B (+63.95%)</td>
<td>1093.5059 ms</td>
</tr>
<tr>
<td>dhrystone</td>
<td>1335 B</td>
<td>2411 B (+80.60%)</td>
<td>102.9200 ms</td>
</tr>
</tbody>
</table>

**Average**

- Memory Footprint: +41.17%
- Runtime Overhead: +52.72%
- Deployment: -73.63%
To recap PISTIS

Why do we need it?
- To bridge the security gap for low-end embedded systems
- Need for feature-rich and strong security solutions
- PISTIS might be the cheapest available option

What is it?
- Trusted Execution Environment (TEE)
- Support for TAs (e.g., Remote Attestation)
- Support for secure DMA and Interrupts operations

How does it work?
- Policy-based Memory Isolation
- Software-based Trusted Computing Base
- Software instrumentation and virtualisation

Q&A
- michele.grisafi@unitn.it
- github.com/CybersecurityUnitn/PISTIS

Check it out!