Motivation

- Goal: Protect sensitive code and data
Motivation

- Goal: Protect sensitive code and data
Motivation

- Goal: Protect sensitive code and data
Motivation

- Goal: Protect sensitive code and data from malicious software
Motivation

- Goal: Protect sensitive code and data from malicious software
- Intel SGX is cool, but
Motivation

- **Goal:** Protect sensitive code and data from malicious software
- Intel SGX is cool, but
  - Closed ISA, microarchitecture
  - Hardware backdoors? [8]
Motivation

- **Goal:** Protect sensitive code and data from malicious software
- Intel SGX is cool, but
  - Closed ISA, microarchitecture
  - Hardware backdoors? [8]
- RISC-V is completely open
Motivation

- Goal: Protect sensitive code and data from malicious software
- Intel SGX is cool, but
  - Closed ISA, microarchitecture
  - Hardware backdoors? [8]
- RISC-V is completely open
  - Bring SGX features to embedded RISC-V
Motivation

- **Goal**: Protect sensitive code and data from malicious software

- Intel SGX is cool, but
  - Closed ISA, microarchitecture
  - Hardware backdoors? [8]

- RISC-V is completely open
  - **Bring SGX features to embedded RISC-V**
    (embedded = ARM Cortex-M)
Background: Enclaves

- Secure execution
Background: Enclaves

- Secure execution
- Protect against all other software
  - Malicious app
Background: Enclaves

- Secure execution
- Protect against all other software
  - Malicious app
  - Malicious OS
Background: Enclaves

- Secure execution
- Protect against all other software
  - Malicious app
  - Malicious OS
  - Malicious hypervisor
Background: Enclaves

- Secure execution
- Protect against all other software
  - Malicious app
  - Malicious OS
  - Malicious hypervisor
  - Malicious enclaves
Background: Enclaves

- Secure execution
- Protect against all other software
  - Malicious app
  - Malicious OS
  - Malicious hypervisor
  - Malicious enclaves
- Minimal trust (enclave + HW)
Related Work - Secure Execution

- "Large": Sanctum (RISC-V) [2]
- "Embedded": [3, 4, 7, 1, 9]
  - RISC-V: MultiZone [5], Keystone [6]
Related Work - Secure Execution

- "Large": Sanctum (RISC-V) [2]
- "Embedded": [3, 4, 7, 1, 9]
  - RISC-V: MultiZone [5], Keystone [6]

Problems

- Inflexible isolation boundaries
- Memory fragmentation
Problem: Memory Fragmentation
Problem: Memory Fragmentation

Physical memory

Process A

App

Enclave

Data

Stack

We want to interleave memory → Stack sharing
Problem: Memory Fragmentation

Physical memory

Process A

App

Enclave

Data

Stack

Data

Stack

We want to interleave memory → Stack sharing

Samuel Weiser
NDSS’19, San Diego
Problem: Memory Fragmentation

We want to interleave memory

→ Stack sharing

Samuel Weiser
NDSS’19, San Diego
Problem: Memory Fragmentation

Physical memory

Process A
App  Enclave
Data  Stack  Data  Stack

Process B
App  Enclave
Data  Stack  Data  Stack

We want to interleave memory → Stack sharing
Problem: Memory Fragmentation

We want to interleave memory → Stack sharing

Samuel Weiser
NDSS'19, San Diego

www.iaik.tugraz.at
Problem: Memory Fragmentation

- Bad memory utilization
Problem: Memory Fragmentation

- Bad memory utilization
- We want to interleave memory → Stack sharing
Problem: Memory Fragmentation

- Bad memory utilization
- We want to interleave memory → Stack sharing
Problem: Memory Fragmentation

- Bad memory utilization
- We want to interleave memory → Stack sharing
Contributions

- **TIMBER-V**: Enclaves from tagged memory for embedded RISC-V
Contributions

- **TIMBER-V**: Enclaves from tagged memory for embedded RISC-V
- Novel stack sharing
Contributions

- **TIMBER-V**: Enclaves from tagged memory for embedded RISC-V
- Novel stack sharing
- Fast shared enclave memory
- Proof-of-concept
TIMBER-V Overview

U-mode

App A

App B

S-mode

Operating System

Machine M-mode
TIMBER-V Overview

- Normal N-domains
  - App A
  - App B

- Trusted T-domains
  - Enclave A
  - Enclave B

- U-mode
- S-mode
  - Operating System
  - Tag Root

- TU-mode
- TS-mode
  - Machine M-mode

Attacker-controlled

Samuel Weiser
NDSS'19, San Diego

www.iaik.tugraz.at
Enclave building blocks:
Enclave building blocks:

- Memory isolation
Enclave building blocks:

- Memory isolation
- Entry points
Enclave building blocks:

- Memory isolation
- Entry points
- Attestation, sealing
- Inter-enclave communication
Enclave building blocks:

- Memory isolation
- Entry points
- Attestation, sealing
- Inter-enclave communication
Traditional Memory Protection Unit (MPU)

- MPU regions define application
- Problem: inflexible
Tagged Memory

Memory tags → Physical Memory

TIMBER-V

Samuel Weiser
NDSS’19, San Diego

www.iaik.tugraz.at
Tagged Memory

- Memory tags define applications
Tagged Memory

- Memory tags define applications
- Problem: high memory overhead
TIMBER-V: MPU + Tagged Memory

- MPU region defines application
TIMBER-V: MPU + Tagged Memory

- MPU region defines application
- Application cannot escape
TIMBER-V: MPU + Tagged Memory

- Memory tag defines enclave
TIMBER-V: MPU + Tagged Memory

- Memory tag defines enclave
- Memory tag defines TagRoot
TIMBER-V Tag Isolation

- Normal memory

Process A (User)  OS (Supervisor)  Process B (User)  TagRoot + OS interleaved (Supervisor)

Normal
TIMBER-V Tag Isolation

- Normal memory
- Cannot access others
TIMBER-V Tag Isolation

- Normal memory
- Cannot access others
TIMBER-V Tag Isolation

- Trusted User memory (enclaves)

Process A (User)
OS (Supervisor)
Process B (User)
TagRoot + OS interleaved (Supervisor)

N Normal
TU Trusted User
TIMBER-V Tag Isolation

- Trusted User memory (enclaves)
- Can access and update normal memory
TIMBER-V Tag Isolation

- **TU** Trusted User memory (enclaves)
- Can access and update normal memory
TIMBER-V Tag Isolation

- Trusted User memory (enclaves)
- Can access and update normal memory
TIMBER-V Tag Isolation

- Trusted User memory (enclaves)
- Can access and update normal memory
TIMBER-V Tag Isolation

- Trusted Supervisor memory (TagRoot)
TIMBER-V Tag Isolation

- **TS** Trusted Supervisor memory (TagRoot)
- Can access and update any tag
TIMBER-V Tag Isolation

- Trusted Supervisor memory (TagRoot)
- Can access and update any tag
TIMBER-V Tag Isolation

- **TS** Trusted Supervisor memory (TagRoot)
- Can access and update any tag

Trusted Supervisor (TagRoot)

Can access and update any tag
TIMBER-V Tag Isolation

- Trusted Supervisor memory (TagRoot)
- Can access and update any tag
TIMBER-V Memory Isolation
TIMBER-V

Enclave building blocks:

✓ Memory isolation
  - Entry points
  - Attestation, sealing
  - Inter-enclave communication
Enclave building blocks:

- Memory isolation
- Entry points
- Attestation, sealing
- Inter-enclave communication
TIMBER-V Entry Points

- Enter only at **TC** Trusted Callable
TIMBER-V Entry Points

- Enter only at **TC** Trusted Callable
- Zero runtime overhead (ordinary \texttt{jmp})
TIMBER-V Entry Points

- Enter only at Trusted Callable
- Zero runtime overhead (ordinary jmp)
TIMBER-V Entry Points

- Four tags → two tag bits only
TIMBER-V Entry Points

- Four tags → two tag bits only
- For 32-bit system +6.25% memory overhead
TIMBER-V

Enclave building blocks:

- Memory isolation
- Entry points
  - Attestation, sealing
  - Inter-enclave communication

Samuel Weiser
NDSS ’19, San Diego

www.iaik.tugraz.at
TIMBER-V

Enclave building blocks:

- Memory isolation
- Entry points
  - Attestation, sealing
  - Inter-enclave communication

Samuel Weiser
NDSS’19, San Diego
www.iaik.tugraz.at
TIMBER-V TagRoot

- Root of trust in privileged software
TIMBER-V TagRoot

- Root of trust in privileged software
- Supports SGX and TrustZone model
TIMBER-V TagRoot

- Root of trust in privileged software
- Supports SGX and TrustZone model
- Enclave management
TIMBER-V TagRoot

- Root of trust in privileged software
- Supports SGX and TrustZone model
- Enclave management
- Inter-enclave communication
  - Fast shared memory
  - Mutual authentication
  - Implicit local attestation
TIMBER-V TagRoot

- Root of trust in privileged software
- Supports SGX and TrustZone model
- Enclave management
- Inter-enclave communication
  - Fast shared memory
  - Mutual authentication
  - Implicit local attestation
- Sealing (like SGX)
TIMBER-V

Enclave building blocks:

✓ Memory isolation
✓ Entry points
✓ Attestation, sealing
✓ Inter-enclave communication
Novel Stack Sharing

- Single stack shared between application and enclave...
Novel Stack Sharing

- Single stack shared between application and enclave ...
Novel Stack Sharing

- Single stack shared between application and enclave ...
Novel Stack Sharing

- Single stack shared between application and enclave ...
Novel Stack Sharing

- Single stack shared between application and enclave...
Novel Stack Sharing

- Single stack shared between application and enclave ...

Application

store XXX
jmp e
...
a: store YYY
store ZZZ

Stack

XXX
KEY

Enclave

e: store KEY
store KEY
jmp a
...

Single stack shared between application and enclave ...
Novel Stack Sharing

- Single stack shared between application and enclave ...
Novel Stack Sharing

- Single stack shared between application and enclave ...

Application

store XXX
jmp e
...

a: store YYY
store ZZZ

Stack

XXX
KEY
KEY

Enclave

e: store KEY
store KEY
jmp a
...

Single stack shared between application and enclave ...
Novel Stack Sharing

- Single stack shared between application and enclave...
Novel Stack Sharing

- Single stack shared between application and enclave ...
Novel Stack Sharing

- Single stack shared between application and enclave...
Novel Stack Sharing

- Single stack shared between application and enclave ...
Novel Stack Sharing

- Single stack shared between application and enclave...
Novel Stack Sharing

- Single stack shared between application and enclave ...
  ... and between TagRoot!
Novel Stack Sharing

- Single stack shared between application and enclave ...
- ... and between TagRoot!
- Heap sharing quite similar
Key Insights

- Build enclaves with tagged memory
  - Fine granularity and high flexibility
- Combination with MPU allows tiny 2-bit tags
- Reduced memory fragmentation
  - Shared stacks, heaps ...
Proof-of-Concept

- Integration in ISA simulator (Spike)
- Full TagRoot implementation
- FreeRTOS integration
- Gnu GCC support
- Benchmarks (Coremark, Beebs)
- Open source: github.com/IAIK/timber-v
TIMBER-V

Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V

Samuel Weiser* Mario Werner*
Ferdinand Brasser† Maja Malenko*
Stefan Mangard* Ahmad Sadeghi†
*Graz University of Technology
†TU Darmstadt
### Bonus: New Instructions

<table>
<thead>
<tr>
<th>RISC-V</th>
<th>TIMBER-V</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>lb, lbu</td>
<td>lbct, lbuct</td>
<td>etag ← check for expected memory tag</td>
</tr>
<tr>
<td>lh, lhu</td>
<td>lhct, lhuct</td>
<td>etag  (fault on mismatch)</td>
</tr>
<tr>
<td>lw</td>
<td>lwct</td>
<td>etag</td>
</tr>
<tr>
<td>ltt</td>
<td>etag</td>
<td>← load and test tag w.o. fault</td>
</tr>
<tr>
<td>sb</td>
<td>sbct</td>
<td>etag, ntag ← also store new memory tag</td>
</tr>
<tr>
<td>sh</td>
<td>shct</td>
<td>etag, ntag</td>
</tr>
<tr>
<td>sw</td>
<td>swct</td>
<td>etag, ntag</td>
</tr>
</tbody>
</table>
Bonus: TIMBER-V Overhead Estimate

Average overhead: 25.2% vs 2.6% (with caching of tags)
References


