BenchIoT: A Security Benchmark for The Internet of Things

Naif Almakhdhub, Abraham Clements, Mathias Payer, and Saurabh Bagchi
The number of IoT devices is expected to exceed 20 billion by 2020.

Many will be microcontroller based systems (IoT-μCs).
- Run single static binary image directly on the hardware.
- Can be with/without an OS (bare-metal).
- Direct access to peripherals and processor.
- Small memory.

Examples:
- WiFi System on Chip
- Cyber-physical systems
- UAVs
Internet of Things Security

• In 2016, one of the largest DDoS attack to date was caused by IoT devices[1].

• In 2017, Google’s Project Zero used a vulnerable WiFi SoC to gain control of the application processor on smart phones[2].

Evaluation in Current IoT Defenses

• Multiple defenses have been proposed.
  • TyTan [DAC15], TrustLite [EurSys14], C-FLAT [CCS16], nesCheck [AsiaCCS17], SCFP [EuroS&P18], LiteHAX [ICCAD18], CFI CaRE [RAID17], ACES [SEC18], MINION [NDSS18], EPOXY [S&P17]

• How are they evaluated?
  • Ad-hoc evaluation.

<table>
<thead>
<tr>
<th>Defense</th>
<th>Evaluation Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Benchmark</td>
</tr>
<tr>
<td>TyTan</td>
<td></td>
</tr>
<tr>
<td>TrustLite</td>
<td></td>
</tr>
<tr>
<td>C-FLAT</td>
<td></td>
</tr>
<tr>
<td>nesCheck</td>
<td></td>
</tr>
<tr>
<td>SCFP</td>
<td>Dhrystone[1]</td>
</tr>
<tr>
<td>LiteHAX</td>
<td>CoreMark[2]</td>
</tr>
<tr>
<td>CFI CaRE</td>
<td>Dhrystone[1]</td>
</tr>
<tr>
<td>ACES</td>
<td></td>
</tr>
<tr>
<td>Minion</td>
<td></td>
</tr>
<tr>
<td>EPOXY</td>
<td>BEEBS[3]</td>
</tr>
</tbody>
</table>

IoT-μCs Evaluation (Ideally)

1. Defense Mechanism A

2. Benchmark foo

3. Evaluation Metrics

A standardized software application
IoT-μCs Evaluation (Reality)

1. Defense Mechanism A

2. Benchmark foo
   - Different benchmarks
   - Different Metrics

3. A’s Evaluation Metrics
   - Comparison is not feasible
   - Evaluation is limited and tedious

Defense Mechanism B

Benchmark bar

B’s Evaluation Metrics
Why not use Existing Benchmark?

- Current benchmarks are rigid and simplistic.
  - Many are just one file with simple application.
  - Metrics are limited and cumbersome to collect.
  - Hardware dependent.
- Do not use peripherals.
- No network connectivity.
Proposed Solution: BenchIoT

- BenchIoT provides a suite of benchmark applications and an evaluation framework.

- A realistic set of *IoT* benchmarks.
  - Mimics common IoT characteristics, e.g., tight coupling with sensors and actuators.
  - Works for both with/without an OS.

- Our evaluation framework is versatile and portable.
  - A software based approach.
  - Can collect metrics related to security and resource usage.

- Targeted Architecture: ARMv7-M (Cortex-M3,4, and 7 processors).
## Comparison Between BenchIoT and Other Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Task Type</th>
<th>Network Connectivity</th>
<th>Peripherals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sense</td>
<td>Compute</td>
<td>Actuate</td>
</tr>
<tr>
<td>BEEBS [2]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dhrystone [1]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoreMark [3]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IoTMark [4]</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SecureMark [5]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BenchIoT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Benchmark: Overview

Evaluation Framework

- Run benchmark on board
- Collect dynamic metrics
- Parse the benchmark binary
- Collect static metrics
- Metric collector runtime library

Results file

Can use a different benchmark
BenchIoT Design Feature: (1) Hardware agnostic

- Applications often depend on the underlying vendor & board.
  - Memory is mapped differently on each board.
  - Peripherals are different across boards.

- For Operating systems:
  - Mbed OS(C++)

Diagram:
- Application
- Portable
- Vendor & board dependent
- Hardware
  - Mbed
  - HAL Library (Hardware Abstraction Layer)
  - CMSIS (Cortex Microcontroller Software Interface Standard)
  - MCU Registers
BenchIoT Design Feature: (2) Reproducibility

• Applications are event driven.
  • Example: User enters a pin.
  • Problem: This is inconsistent (e.g., variable timing).

• Solution: Trigger interrupt from software.
  • Creates deterministic timing.
  • Allows controlling the benchmarking execution.
BenchIoT Design Feature: (2) Reproducibility

Normal application

```c
/* Pseudocode */
1. void benchmark(void){
2.  do_some_computation();
3.  ...
4.  ...
5.  wait_for_user_input();
6.  read_user_input();
7.  ...
8.  ...
9. }
```

This is not deterministic

BenchIoT

```c
/* Pseudocode */
1. void benchmark(void){
2.  do_some_computation();
3.  ...
4.  ...
5.  trigger_interrupt();
6.  ...
7.  read_user_input();
8.  ...
9.  ...
10. }
```

Deterministic
BenchIoT Design Feature: (3) Metrics

• Allows for measurement of 4 classes of metrics: Security, performance, energy, and memory.
BenchIoT Design Feature: (3) Metrics

- **Static metric**
  - Total privileged cycles
  - Privileged Thread cycles
  - SVC cycles
  - Max Data region ratio
  - Max Code region ratio
  - DEP
  - ROP resiliency
  - # of indirect calls

- **Dynamic metric**
  - Total execution cycles
  - CPU sleep cycles
  - Total energy

- **Security**
- **Performance & Energy**
- **Memory**
  - Stack+Heap usage
  - Total RAM usage
  - Total Flash usage
## Set of Benchmark Applications

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Task Type</th>
<th>Peripheral</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sense</td>
<td>Compute</td>
</tr>
<tr>
<td>Smart Light</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Smart Thermostat</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Smart Locker</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Firmware Updater</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Connected Display</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- Boards without non-common peripherals can still run the benchmark.
BenchIoT Evaluation: Defense Mechanisms

**ARM’s Mbed-µVisor**

- **Unprivileged**
  - Application code

- **Privileged**
  - µVisor + OS
  - A hypervisor that enforces the principle of least privilege.

**Remote Attestation (RA)**

- Hashed code block
- 25ms

- Verifies the integrity of the code present on the device.
- Uses a real-time task that runs in a separate thread.
- Isolates its code in a secure privileged region.

**Data Integrity (DI)**

- Sensitive Data
  - Privileged
- Isolates sensitive data to a secure privileged region.
- Disables the secure region after the data is accessed.
BenchIoT Evaluation: Defense Mechanisms

• The goal is to demonstrate BenchIoT effectiveness in evaluation.
  • **Non-goal**: To propose a new defense mechanism.

• ARM’s Mbed-µVisor and Remote Attestation (RA) **require an OS**.

• Data Integrity (DI) is applicable to **Bare-Metal (BM) and OS benchmarks**.
BenchIoT Evaluation: Defense Mechanisms

- Comparable
- Evaluation is automated and extensible.

BenchIoT Evaluation Framework

- ARM’s Mbed-µVisor Evaluation
- Remote Attestation (RA) Evaluation
- Data Integrity (DI) Evaluation

BenchIoT Benchmarks

ARM’s Mbed-µVisor

Remote Attestation (RA)

Data Integrity (DI)
Performance Results

Number of cycles in (Billions/Millions)

Evaluated without the display peripheral
Privileged Execution Minimization Results

- Overhead as % of the insecure baseline application

Almost the entire application runs as privileged for all defenses Except uVisor

uVisor is the most effective defense in reducing privileged execution

Lower privileged execution → Better Security
## Code Injection Evaluation

<table>
<thead>
<tr>
<th>Defense</th>
<th>Data Execution Prevention (DEP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mbed-uVisor</td>
<td>✗ (Heap)</td>
</tr>
<tr>
<td>Remote Attestation (OS)</td>
<td>✓</td>
</tr>
<tr>
<td>Data Integrity (OS)</td>
<td>✗</td>
</tr>
<tr>
<td>Data Integrity (Bare-metal)</td>
<td>✗</td>
</tr>
</tbody>
</table>
Energy Consumption Results

All defenses had modest runtime overhead.

uVisor had no sleep cycles
≈ 20% energy overhead

Overhead as % over baseline
Measurement Overhead

Average Overhead → 1.2%

Percentage of total execution cycles
BenchIoT: Summary

• Benchmark suite of five realistic IoT applications.
  • Demonstrates network connectivity, sense, compute, and actuate characteristics.
  • Applies to systems with/without an OS.

• Evaluation framework:
  • Covers security, performance, memory usage, and energy consumption.
  • Automated and extensible.

• Evaluation insights:
  • Defenses can have similar runtime overhead, but a large difference in energy consumption.

• Open source:
  • https://github.com/embedded-sec/BenchIoT