aSSIsT

Software Security for the IoT

Bengt Jonsson   Luca Mottola
Shahid Raza   Konstantinos Sagonas

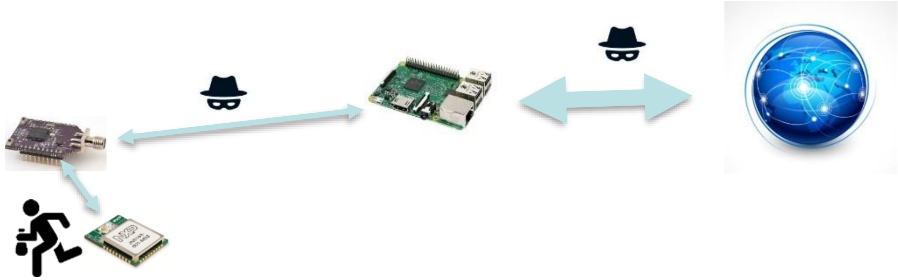
IoT Software Security: Challenges

Internet of Things (IoT):

- Primary concern: **Security**

Scope of aSSIsT:

- Security of **IoT Software**
  - in platforms, communications, applications.



Challenges:

- Large attack surface
  - Internet, Wireless, Physical
- Resource-constrained platforms

  ⇒ Lack of support (memory protection, intrusion detection, …)

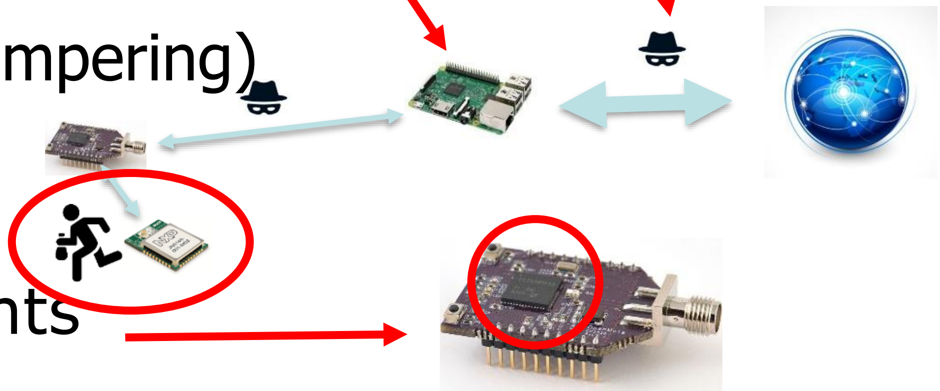
UPPSALA UNIVERSITET

RISE

aSSIsT Focus Directions

- Software Testing and Fuzzing

- Testing and verification of security protocol implementations
- Battery-Free Devices, (Physical Tampering)
- Trusted Execution Environments



Targets:
- IoT OSes: Contiki-NG, Zephyr
- IoT protocols: DTLS, QUIC, EDHOC

UPPSALA UNIVERSITET

RI. SE

aSSIsT: Secure Software for IoT

Project duration: 2018-2024, https://assist-project.github.io
Funding: Swedish Foundation for Strategic Research (SSF)

Participating Groups

Uppsala University, Dept. IT
Senior: Bengt Jonsson, Kostis Sagonas, Mohammed Faouzi Atig
PostDocs: Paul Fiterau-Brostean, Sandip Ghosal, Rémi Parrot
PhD: Hooman Asadian, Sarbojit Das, Magnus Lång, Fredrik Tåkvist

RISE CS, Kista
Senior: Luca Mottola, Shahid Raza, Nicolas Tsiftes, Thiemo Voigt
PostDocs: Chetna Singhal
Ph.D: Anum Khurshid (just defended)

Reference Group
ASSA ABLOY, Intel Sweden, LumenRadio, Upwis, Wittra

UPPSALA UNIVERSITET

Software Testing and Fuzzing

Detect bugs and vulnerabilities using

- **Fuzzing** (or **Fuzz Testing**)
  - fast software testing based on random inputs
- **Symbolic Execution**
  - slow but effective in exploring most/all program paths
- **Hybrid Fuzzing**
  - combines the two above

One of our targets: **Contiki-NG**

"The OS for Next Generation of IoT Devices"

Fuzzing the Contiki-NG Network Stack

Created infrastructure to fuzz at different network stack layers

Detected and fixed:

- 18 vulnerabilities (in IPv6, 6LoWPAN, ICMPv6, and RPL)
- 11 of which come with CVEs

Evaluated the effectiveness of eight state-of-the-art fuzzing tools

- Mutation-based: AFL-gcc, AFL-clang-fast, Honggfuzz, Mopt-AFL
- Hybrid: Angora, QSYM, Intriguer, SymCC

with and without sanitizer support

C. Poncelet, K. Sagonas, N. Tsiftes. **So Many Fuzzers, So Little Time - Experience from Evaluating Fuzzers on the Contiki-NG Network (Hay)Stack.** *ASE 2022*.

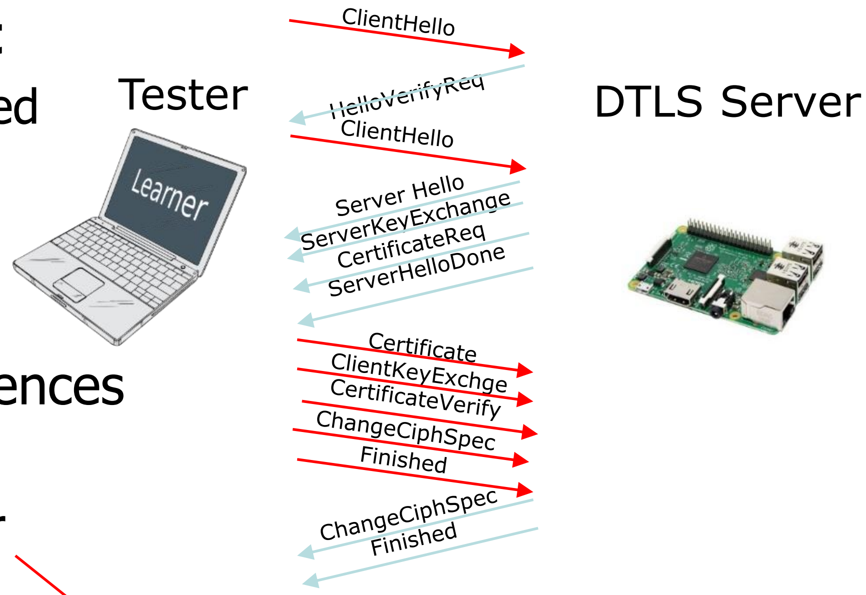
Testing of Security Protocols Implementations

Challenge 1: Test that

Only correctly ordered packets are received and sent

- E.g., Input with missing authentication packet should be rejected

Connection Establishment in DTLS



Solution:

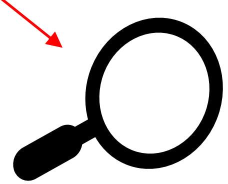
State Fuzzing

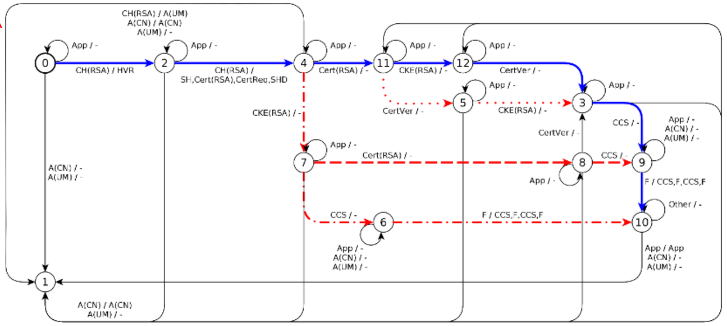
1. Test reaction to systematically constructed packet sequences

2. Learn **model** of implementation's input-output behavior

3. Check packet ordering requirements on **model**

automatic





Applied to DTLS, SSH, TCP, EDHOC

P. Fiterau-Brostean, B. Jonsson, K. Sagonas, F. Tåkvist. **Automata-Based Automated Detection of State Machine Bugs in Protocol Implementations**. *NDSS 2023*

UPPSALA UNIVERSITET

RI. SE

Testing of Security Protocols Implementations

Challenge 2: Test that

Only correct packet data is received and sent

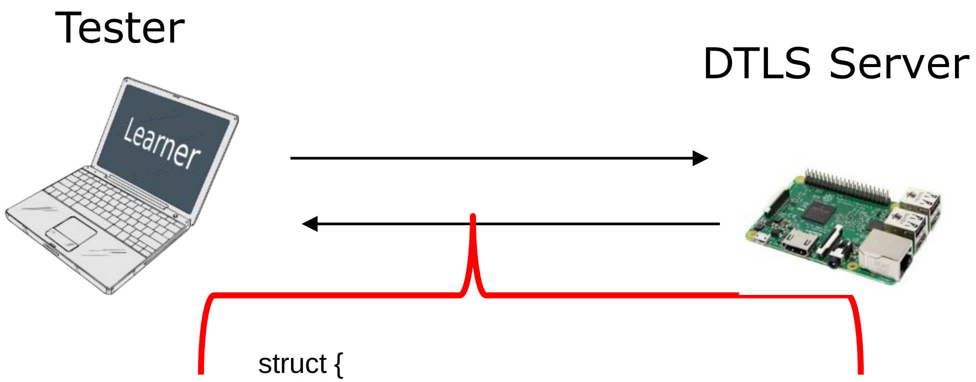
- E.g., is correctness of size fields in input packets checked?
  - Insufficient checks cause overreads/overwrites (cf. Heartbleed)

Solution:

Symbolic Execution

- Covers all values of data fields in input packets
- Detects insufficient checking of packet contents, and incorrect data in output
- Applied to DTLS, QUIC

Packet structure in DTLS




```
struct {
    ContentType type;
    ProtocolVersion version;
    uint16 epoch;
    uint48 sequence_number;
    uint16 length;
    opaque fragment[DTLSPlaintext.length];
} DTLSPlaintext;
```

H. Asadian, P. Fiterau-Brostean, B. Jonsson, K. Sagonas. **Applying Symbolic Execution to Test Implementations of a Network Protocol Against its Specification.** *ICST 2022*

UPPSALA UNIVERSITET

Impact on Existing IoT Software

Fixes of bugs and vulnerabilities found in fuzzing research:

- For Contiki-NG:
  - 18 bug fixes and 11 CVEs
  - First continuous integration test suite for Contiki-NG which directly targets security
- For DTLS implementations:
  - 30+ bug fixes and 3 CVEs
  - In GnuTLS, Java SSE, OpenSSL, PionDTLS, Scandium, TinyDTLS, WolfSSL
- For QUIC implementations: 3 bug fixes

Open-source software tools:

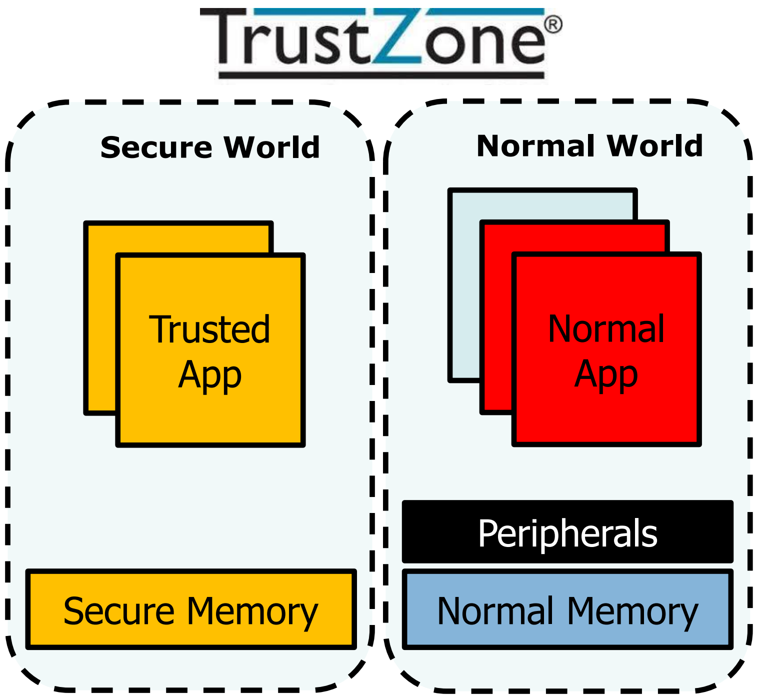
- *DTLS-Fuzzer:* Framework for state fuzzing of DTLS implementations
- *PropEr:* Property-based testing, now also for network protocols
- *Nidhugg:* Finding concurrency errors in concurrent C code

UPPSALA UNIVERSITET

Trusted Execution Environments (TEE)

TEEs provide efficient mechanisms to isolate critical software functionality

- Secure boot, digital signatures, authentication, firmware update
- Memory and peripherals partitioned into **secure** and **normal** world
- ARM supports TEE security extension in microcontrollers: TrustZone-M



Trusted Execution Environments (TEE)

Challenges for TrustZone-M on resource-constrained devices:

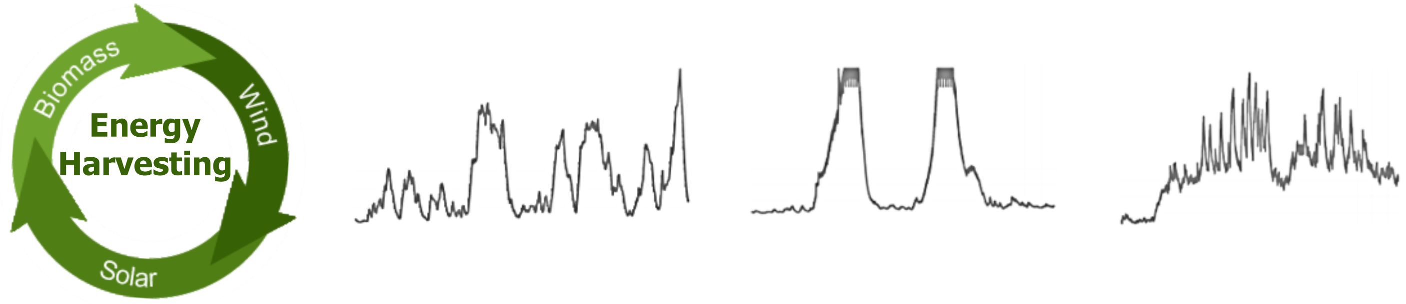
1. Authenticating communication requests from normal to secure world
   - ShieLD: Lightweight message protection scheme ensuring confidentiality and integrity, does not rely on encryption
2. Detecting if a secure application is compromised
   - TEE-watchdog: Mitigation of unauthorized activity in TEE
3. Remote attestation and Software-state certification of IoT devices
   - AutoCert: Combines Software-state certification and PKI
4. Supporting TEEs in Contiki-NG
   - Work in progress

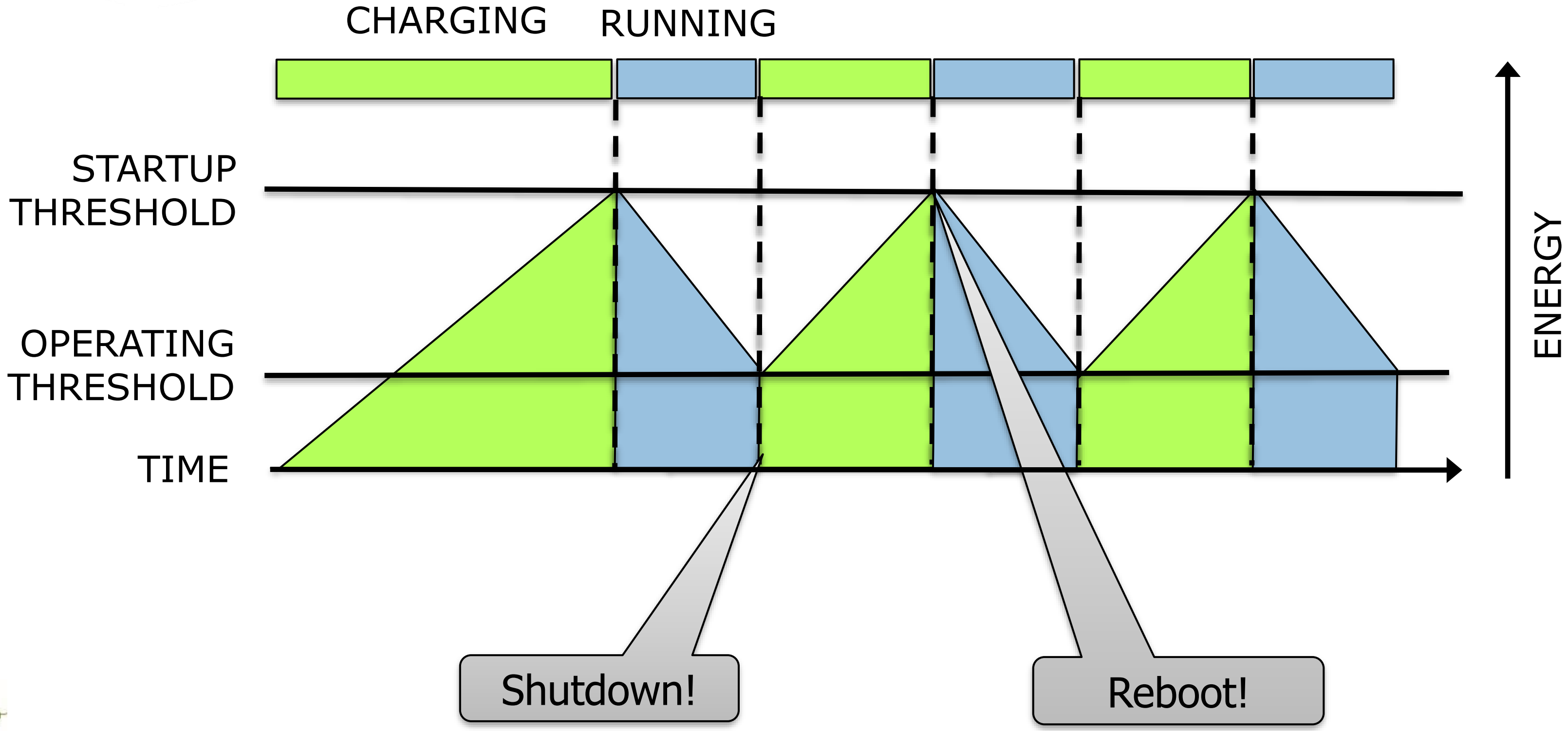
Anum Khurshid, S.D. Yalew, M. Aslam, S. Raza. **ShieLD: Shielding Cross-zone Communication within Limited-resourced IoT Devices running Vulnerable Software Stack**. *IEEE Transactions on Dependable and Secure Computing*.

Anum Khurshid, S.D. Yalew, M. Aslam, S. Raza. **TEE-Watchdog: Mitigating Unauthorized Activities within Trusted Execution Environments in ARM-Based Low-Power IoT Devices**. *Security and Communication Networks*.

A. Khurshid, S. Raza. **AutoCert: Automated TOCTOU-secure digital certification for IoT with combined authentication and assurance**. *Elsevier Computers and Security*.

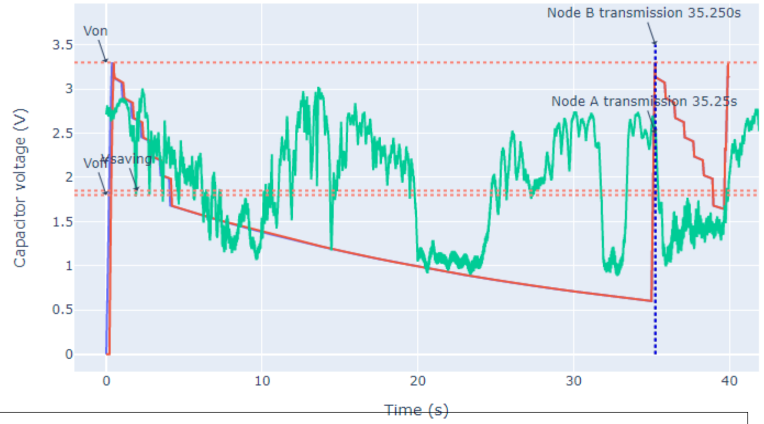
Securing Intermittent Computing





Intermittent Computing: Results

- **Problem:** Securing persistent state
  - **Results**: Comparing different schemes
- **Problem:** Energy attacks
  - How to detect the attacker is messing with the source?
  - How to mitigate the effects?
- **Findings:**
  - Energy attacks may cause priority inversion, livelocks, and unwanted synchronization
- **Outcomes:**
  - Monitoring system with 95%+ accuracy and little overhead
  - Mitigation architecture to deal with it
  - Multi-capacitor attack-aware energy management
  - Open-source release soon!



H. Asad, E. Wouters, N. Bhatti, L. Mottola, T. Voigt. **On Securing Persistent State in Intermittent Computing.** *ENSSYS* 2020.

A. Maioli, L. Mottola, J. Siddiqui, H. Alizai. **Discovering the Hidden Anomalies of Intermittent Computing.** *EWSN* 2021.

UPPSALA UNIVERSITET

Opportunities for Future Work and Collaboration

Software fuzzing and testing

- Test effectiveness of fuzzing techniques on other IoT software
- Infrastructure for Fuzzing in new target environments
  - ***In progress:*** *fuzzing infrastructure on emulation platforms*

(Infrastructure for) Testing protocol implementations

- Application to other IoT protocols: OSCORE, QUIC
- ***In progress:*** *Testing EDHOC*

TEEs

- ***In progress:*** *Supporting TrustZone-M in Contiki*

Intermittent computing

- Low-power reconfigurable hardware