aSSIsT

Software Security for the IoT

Bengt Jonsson   Luca Mottola
Shahid Raza   Konstantinos Sagonas

UPPSALA UNIVERSITET

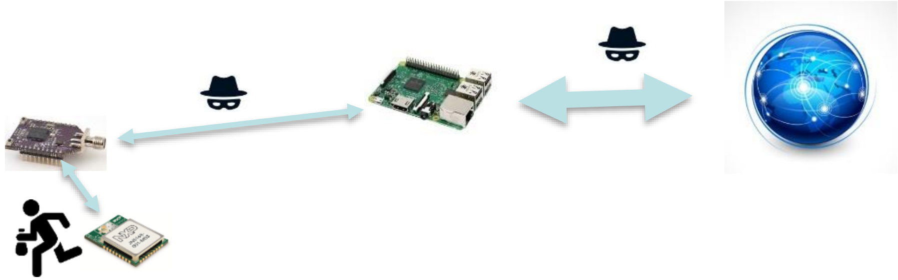
Background and Motivation

Internet of Things (IoT):

- Primary concern: **Security**

Focus 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

Goals and Approach

Overall Goal:

Develop techniques to make IoT software resilient against security attacks, for use by developers of Software for IoT

Approach:

Advance state-of-the-art in

1. Testing and verification of security protocol implementations
2. Testing and security analysis of IoT software
3. Run-time protection mechanisms
   - Trusted execution environments
   - Low-power intermittent computing

UPPSALA UNIVERSITET

Consortium

Uppsala University, Dept. IT

Senior: Bengt Jonsson, Kostis Sagonas, Mohammed Faouzi Atig
PostDocs: Paul Fiterau-Brostean, Clement Poncelet
PhD: Hooman Asadian, Sarbojit Das, Magnus Lång, Fredrik Tåkvist

RISE CS, Kista

Senior: Luca Mottola, Shahid Raza, Nicolas Tsiftes, Thiemo Voigt
PostDocs: Navid Bhatti, Sileshi Demesie Yalew, Carlos Penichet
Ph.D: Anum Khurshid

Reference Group

ASSA ABLOY, Intel Sweden, LumenRadio, Upwis, Wittra

UPPSALA UNIVERSITET

Testing of Security Protocols Implementations

Challenge:

Cover all possible sequences of attacker inputs

Challenge 1:

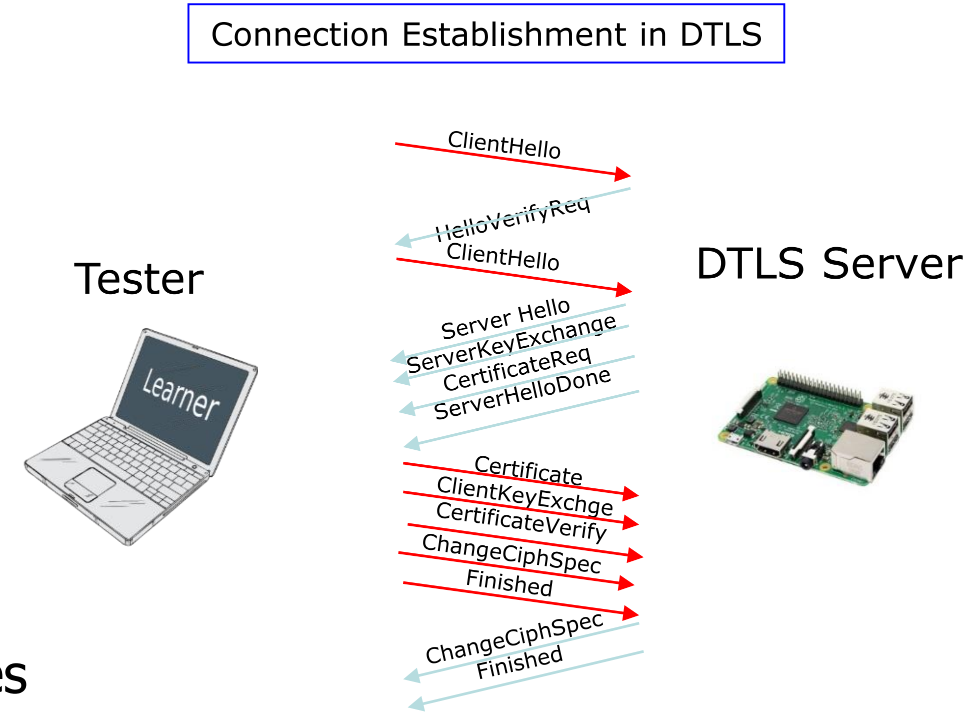
Correct ordering of packets received and sent

- E.g., can authentication be bypassed?

Solution:

State Fuzzing

- Systematic application of constructed input sequences
- Automated detection of packet ordering errors
- Applied to DTLS, SSH, TCP



Testing of Security Protocols Implementations

Challenge:

Cover all possible sequences of attacker inputs

Challenge 2:

Correctness of packet data

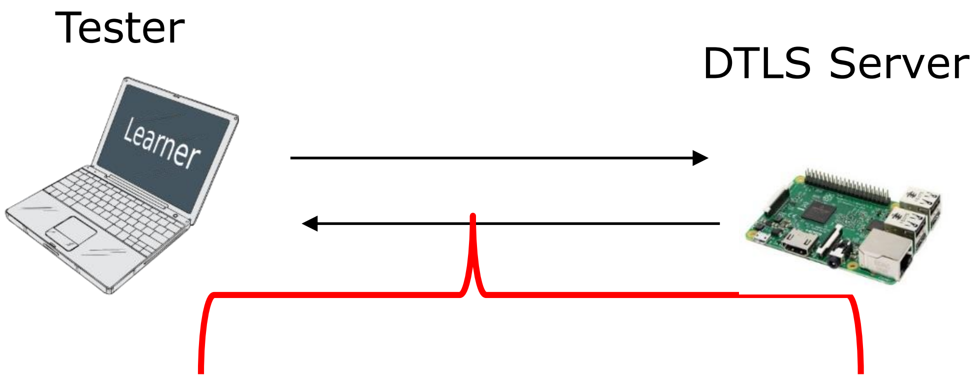
- E.g., is correctness of size fields in input packets checked?
  - Insufficient checks cause overreads/overwrites (cf. Heartbleed)

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

Packet structure in DTLS




```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<2..2^16-2>;
    CompressionMethod compression_methods<1..2^8-1>;
    select (extensions_present) {
        case false:
            struct {};
        case true:
            Extension extensions<0..2^16-1>;
    };
} ClientHello;
```

UPPSALA UNIVERSITET

Software Analysis for IoT Software

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**
  - technique that combines the two above

Our target: **Contiki-NG**

"The OS for Next Generation of IoT Devices"

NG

UPPSALA UNIVERSITET

RISE

Fuzzing the Contiki-NG Network Stack

Created infrastructure to fuzz at different network stack layers

Detected and fixed:

- 17 vulnerabilities (in IPv6, 6LoWPAN, ICMPv6, and RPL)

Using 8 state-of-the-art fuzzing tools

- Mutation-based: AFL, AFL-cf, Mopt
- Hybrid: Angora, QSym, Intriguer, SAVIOR, SymCC

Impact on Existing IoT Software

Fixes of bugs and vulnerabilities found in fuzzing research:

- For Contiki-NG:
  - 17 bug fixes and 6 CVEs
  - First continuous integration test suite for Contiki-NG which directly targets security
- For DTLS implementations:
  - 17 bug fixes and 3 CVEs
  - In GnuTLS, Java SSE, OpenSSL, PionDTLS, Scandium, TinyDTLS, WolfSSL

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

Trusted Execution Environments (TEE)

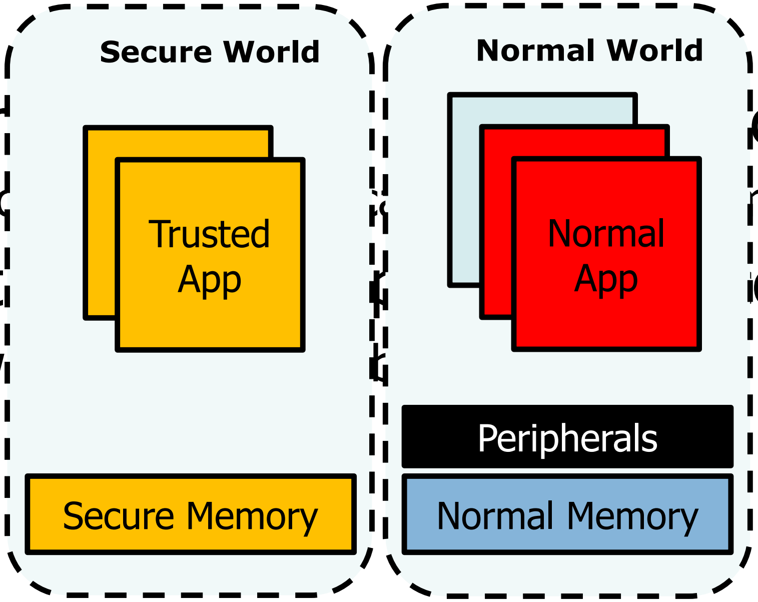
TEEs provide efficient mechanisms to isolate critical software components

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

TrustZone®

Problems:

1. Communication char … ormal world is vulnerable
   - No way to authenticate in … n the normal to secure world
2. Impossible to detect … ompromised
   - unauthorized activities w…



Trusted Execution Environments (TEE)

Solutions:

1. ShieLD: Lightweight message protection scheme ensuring confidentiality and integrity
2. TEE-watchdog: Mitigation of unauthorized activity of applications in TEE

Proof-of-concept implementations on IoT hardware w. TrustZone-M

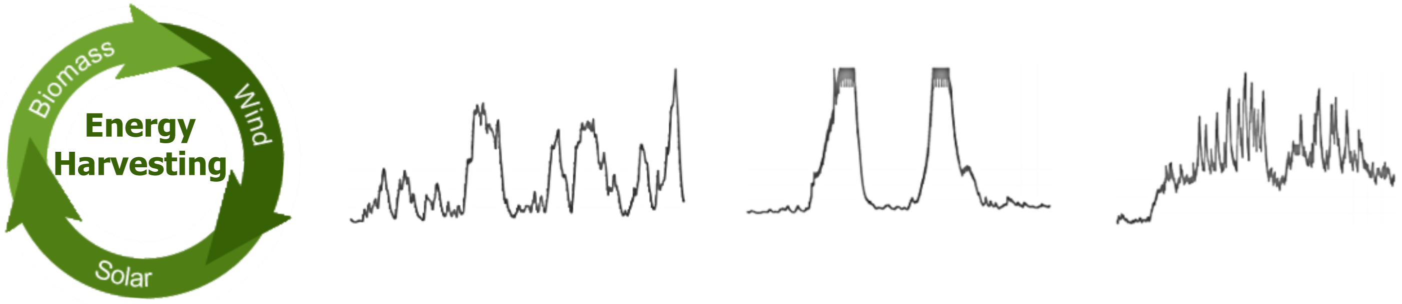
- Minimal execution overhead

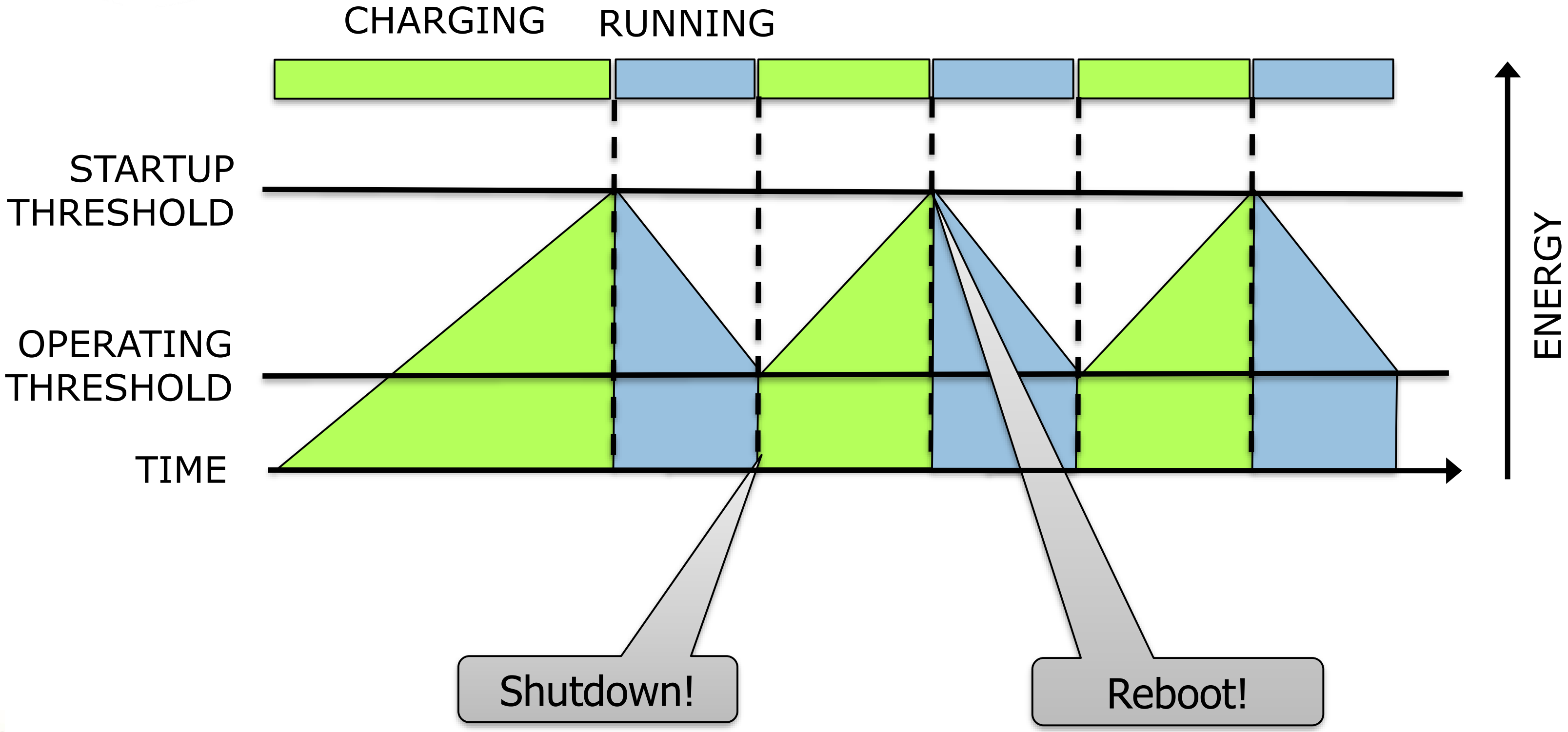
Publications under submission.

Future Work:

- Remote attestation of IoT devices
- Software-state certification of IoT devices

Securing Intermittent Computing



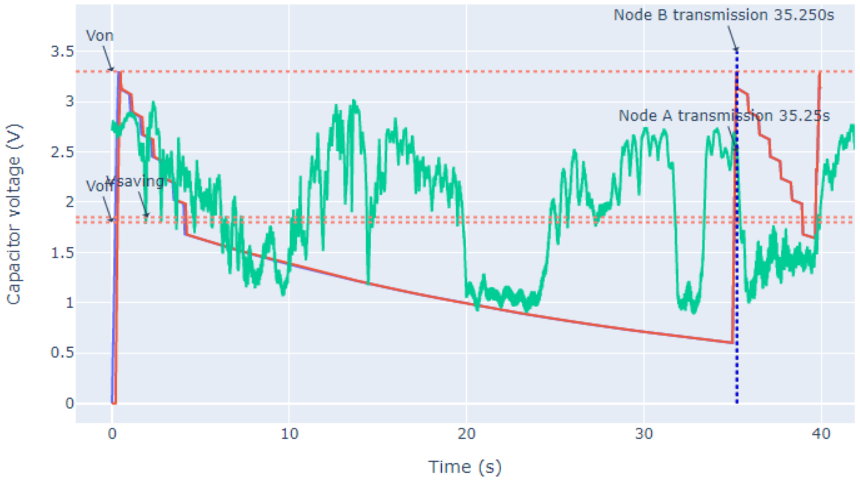


UPPSALA UNIVERSITET

RISE

Intermittent Computing: Results

- **Problem:** securing persistent state
  - **Results**: paper at ENSSYS20
- **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:**
  - A monitoring system with 95%+ accuracy and little overhead
  - A mitigation architecture to let programmers deal with it



Opportunities for Future Work and Collaboration

Testing of protocol implementations

- Applying test techniques to other IoT protocols (e.g., EDHOC, OSCORE, …)

Software analysis

- Test effectiveness of our techniques on other IoT software

TEEs

- Remote attestation and software-state certification of IoT devices
- Realization on open-source hardware

Intermittent computing

- Low-power reconfigurable hardware
- Energy-harvesting technology