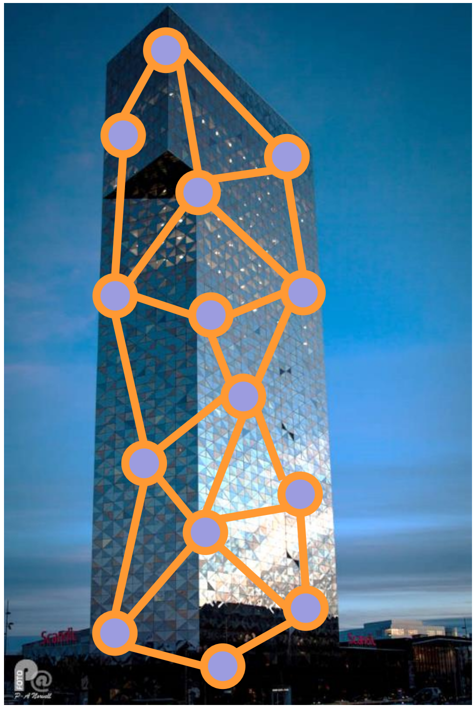
aSSIsT:
Software Security
for the IoT

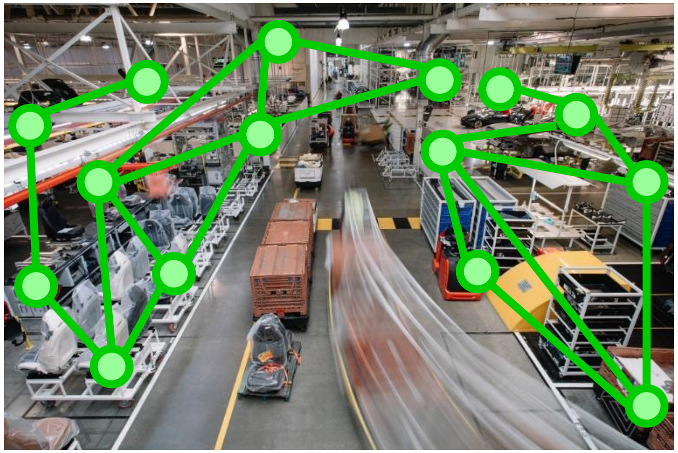
Simon Duquennoy Bengt Jonsson
Luca Mottola Shahid Raza Konstantinos Sagonas

UPPSALA UNIVERSITET
RI. SE

Internet of Things (IoT)



Tens of billions
of connected devices
by 2020?



- Requirements
  - Reliability
  - Long lifetime, low cost
  - Interoperability
  - **Security**

UPPSALA UNIVERSITET

RI. SE

Security in the IoT

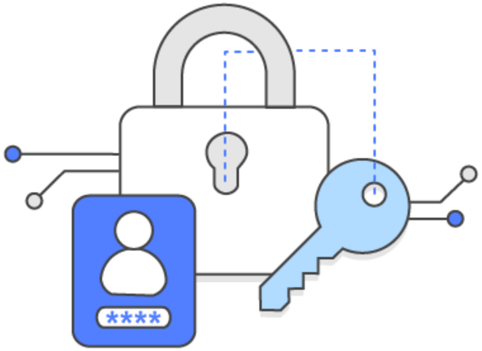
- Recent attacks on IoT networks
  - Takeover of entire IoT networks [Mirai, Linux.Darlloz]
  - Unauthorized control of nodes [Zigbee War-flying]
- Infections spread rapidly and enable coordinated attacks
- Security must be prime concern

Security involves many aspects

- Software

Focus of aSSIsT

- Communication
- Physical Access
- System management
- ....



UPPSALA UNIVERSITET

aSSIsT: Secure Software for IoT

Project duration: 2018-2023, https://assist-project.github.io
Funding: Swedish Foundation for Strategic Research (SSF)

Challenge

Develop techniques for
making IoT software resilient against security attacks

Participating Groups

IT department, Uppsala University

- Verification group [Bengt Jonsson, Parosh Abdulla, Mohammed Faouzi Atig]
- Programming language technology group [Kostis Sagonas, Tobias Wrigstad]

RISE SICS, Kista

- Networked embedded systems groups [Simon Duquennoy, Luca Mottola, Thiemo Voigt]
- Security lab [Shahid Raza, Ludwig Seitz]

Reference group of SMEs with IoT software development

UPPSALA UNIVERSITET

RI. SE

The IoT OS Landscape

RIOT

Apache mynewt

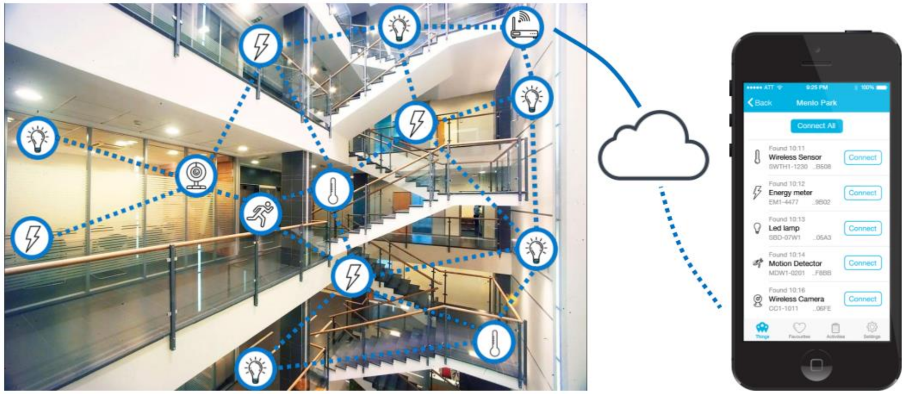
arm MBED

Contiki NG

Zephyr

Introducing Contiki-NG

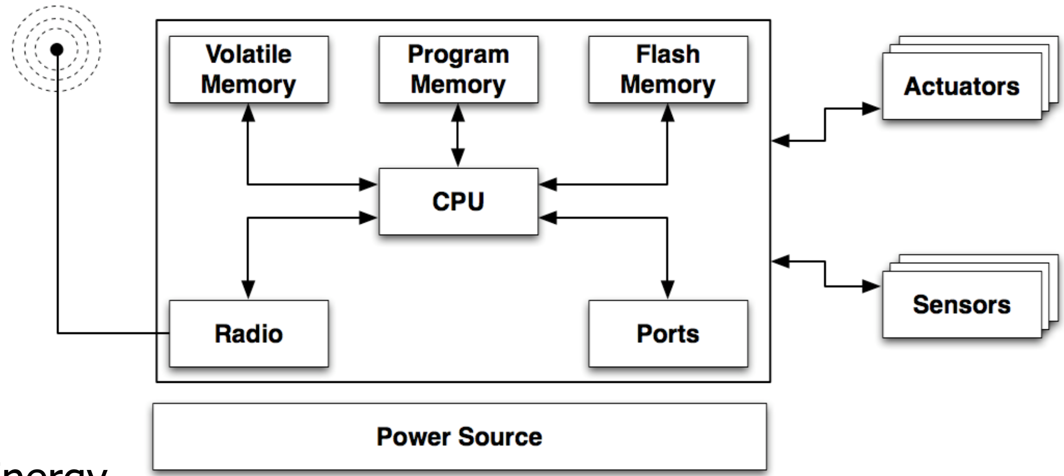
- An Open-source OS for the IoT
- Several Swedish companies build products on top
- Focus on standard-based, interoperable systems
  - 6LoWPAN, 6TiSCH, RPL, CoAP, DTLS, LWM2M etc.
- Focus on **dependable** communication
  - Security: link-layer (IEEE 802.15.4) and application-layer (DTLS)
  - Reliability: brand-new RPL. 99.999% in RPL/TSCH mesh!
- Focus on modern IoT platforms
  - 16 and 32 bits, with low-power radio capabilities
  - Homogenized interfaces for H/W features



IoT Hardware

- CPU
  - ROM for program
  - RAM for variables
  - Flash for long-term storage
  - Ports for I/O
- Radio
  - Low-power, low datarate
  - IEEE 802.15.4, Bluetooth Low Energy, LoRa, etc.
- Power source
  - Battery, or capacitor and harvester
- Sensors / Actuators
  - Application-specific

Picture credit: *Luca Mottola*



Challenges for IoT Software Security

Large attack surface

- Internet
- Wireless
- Physical tampering

Resource-constrained

- Lack much useful support
  - Memory protection and isolation (MCU)
    - Intruders get access to entire system
  - Intrusion detection and mitigation
  - Component replacement
- Highly optimized programming style
  - untyped pointers, limited defensive programming, …

Platform-specific constructs

- For accessing peripherals

aSSIsT: Secure Software for IoT

Goals: Develop techniques and tools for

1. Detecting software vulnerabilities
   - Software analysis, fuzzing
2. Testing and verification of (security) protocol implementations
   - Conformance testing, security testing
3. Run-time protection mechanisms
   - Memory protection, intrusion monitoring

Demonstrators:

- Contiki OS
- IoT protocols, including:
  - DTLS (Datagram TLS)
  - TSCH (Time-Slotted Channel Hopping) MAC-layer protocol for IoT

UPPSALA UNIVERSITET

Software Analysis

Dynamic techniques
- Testing, symbolic execution
- Exercise paths in the code

Static techniques
- Static analysis, formal verification
- Analyze code for (possible) errors

\+ Easy to set up
\+ Scales well
\+ No (few) false positives
\- Do not provide guarantees

\- Can be difficult to set up
\- Tradeoff scalability/false negatives
\+ Do provide guarantees

Software Analysis

Dynamic techniques

- Testing, symbolic execution
- Exercise paths in the code

←→

+ Easy to set up
+ Scale well
+ No (few) false positives
- Do not provide guarantees

HowTo:

1. Instrumentation:
   - Convert properties to be checked into assertions
2. Exploration
   - Exercise as many program paths as possible to search for crash/violation

UPPSALA UNIVERSITET

RISE

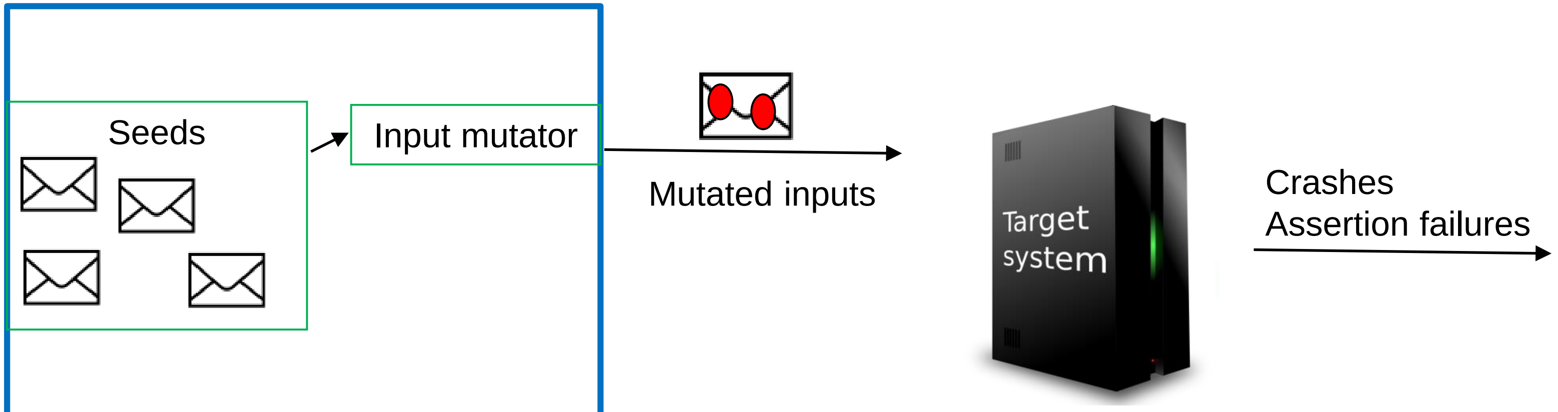
Random Testing

Random inputs →

Crashes
Assertion failures →

- \+ easy to set up
- \+ large number of tests per second
- \- hard to achieve reasonable coverage

Blackbox Fuzzing



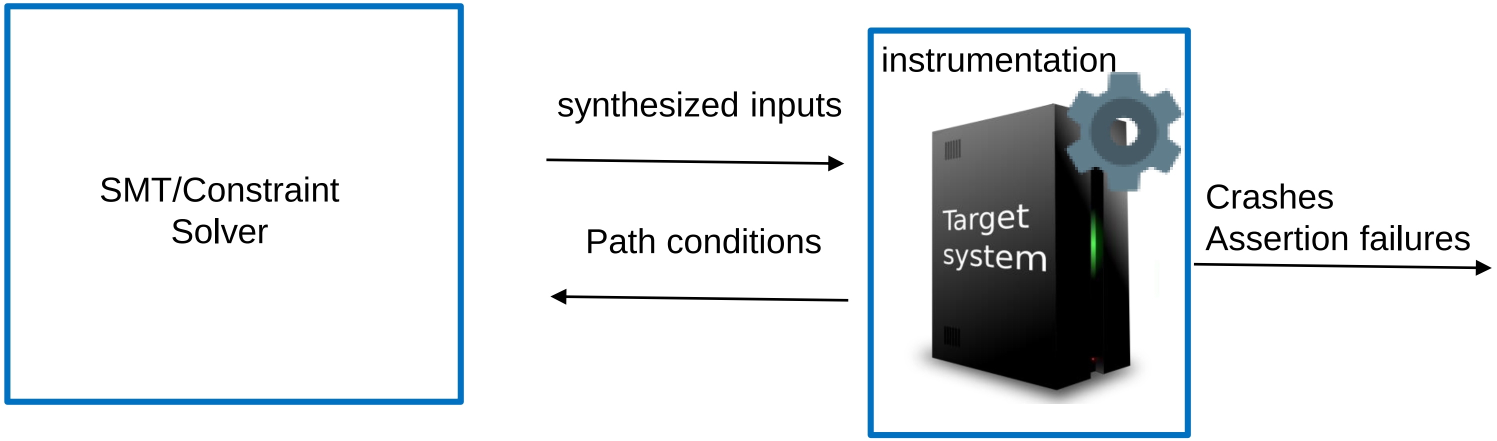
Tools include Radamsa

+ easy to set up
+ large number of tests per second
- hard to achieve reasonable coverage

UPPSALA UNIVERSITET

Whitebox Fuzzing

aka. symbolic/concolic execution



Tools include Klee, CREST

\+ Can search for new paths efficiently

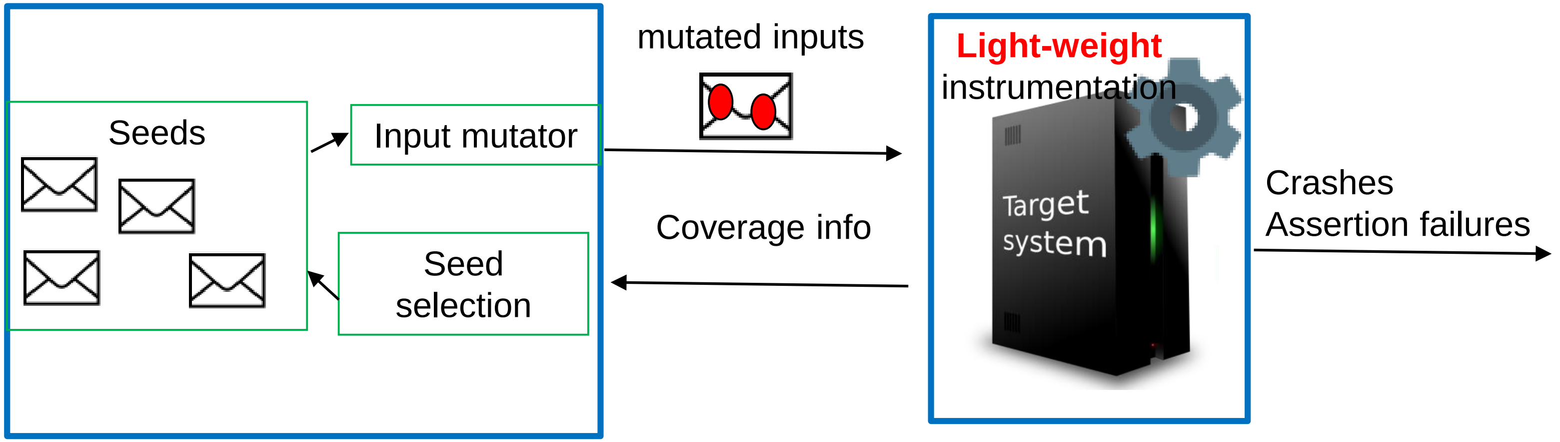
\+ Can find deep bugs / achieve high coverage

\- Non-trival to set up

\- High overhead per test case

\- Some path conditions difficult to analyze

Greybox Fuzzing



Tools include AFL, libFuzzer

\+ easy to set up

\+ (rather) large numbers of tests per second

\+ Can efficiently search for new paths

\- Hard to find "magic numbers"

Challenges for aSSIsT

Goal: Powerful fuzzing techniques for IoT Software

- Device platform characteristics
  - Device-specific ways to access peripherals
  - Proper usage of driver APIs
- Handling interrupts and threading
  - Provoking arbitrary interaction patterns
  - Optimizations to combat state-space explosion
    - Leveraging our previous work for multithreaded software
      e.g., using Nidhugg [Kokologiannakis Sagonas, SPIN 17]
- Approaching complete coverage (as, e.g., [Christakis Godefroid VMCAI 15])
  - Modularization into modest-size components
  - Specification of component interfaces

UPPSALA UNIVERSITET

Contiki-NG Github Issues

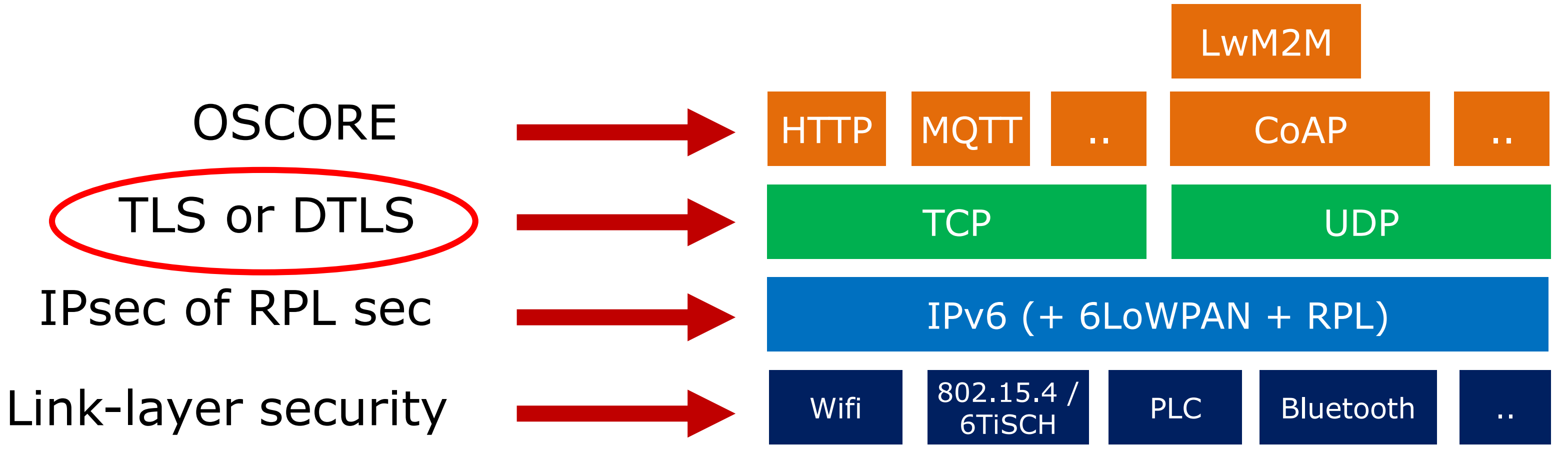
- Vulnerability reports (includes CVEs)

2 Open ✓ 7 Closed — Author ▾ Labels ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

- Stack based buffer overflow while parsing JSON file `bug` `bug/vulnerability`
  #601 opened on Jul 11 by cve-reporting
- Stack based buffer overflow while parsing MQTT messages (parsing PUBLISH message with variable length header) `bug` `bug/vulnerability`
  #600 opened on Jul 11 by cve-reporting
- Stack based buffer overflow while parsing AQL (storage of relations) `bug` `bug/vulnerability`
  #599 by cve-reporting was closed on Aug 28
- Global buffer overflow while parsing AQL (lvm_shift_for_operator) `bug` `bug/vulnerability`
  #598 by cve-reporting was closed on Aug 28
- Global buffer overflow while parsing AQL (lvm_register_variable, lvm_set_variable_value, create_intersection, create_union) `bug` `bug/vulnerability`
  #597 by cve-reporting was closed on Aug 28
- Global buffer overflow while parsing AQL (lvm_set_op, lvm_set_relation, lvm_set_operand) `bug` `bug/vulnerability`
  #596 by cve-reporting was closed on Aug 28
- Stack based buffer overflow while parsing AQL (parsing next string) `bug` `bug/vulnerability`
  #595 by cve-reporting was closed on Aug 28
- Stack based buffer overflow while parsing AQL (parsing next token) `bug` `bug/vulnerability`
  #594 by cve-reporting was closed on Aug 28
- [Potential] Security Vulnerabilities within Erbium `bug` `bug/vulnerability` 💬 7
  #425 by bsmelo was closed on May 4

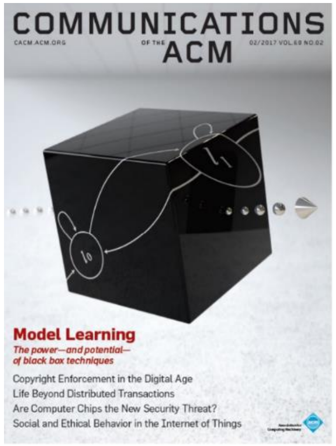
aSSIsT: Communication Security

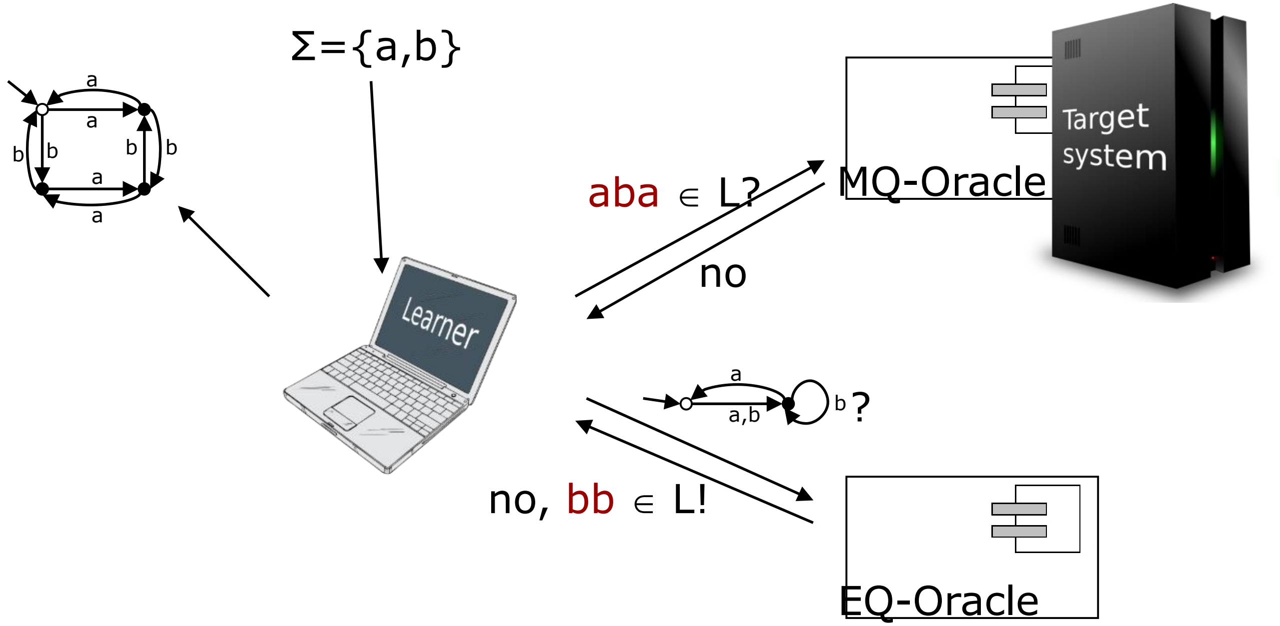
Security in low-power IPv6



- Implementation correctness important
  - Conformance to protocol standard
  - Absence of vulnerabilities
- Weaknesses in TLS implementations discovered in recent years, e.g.,
  - Cryptographic function implementations exposing side channels
    - E.g., Bleichenbacher attack, ..
  - Unexpected input messages may bypass authentication

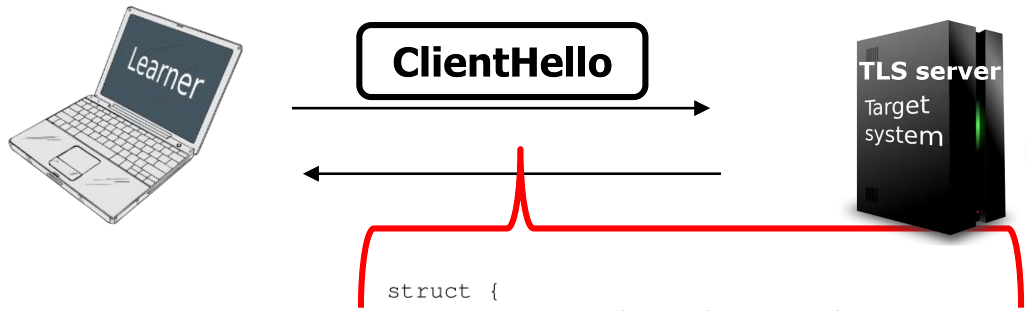
Model Learning





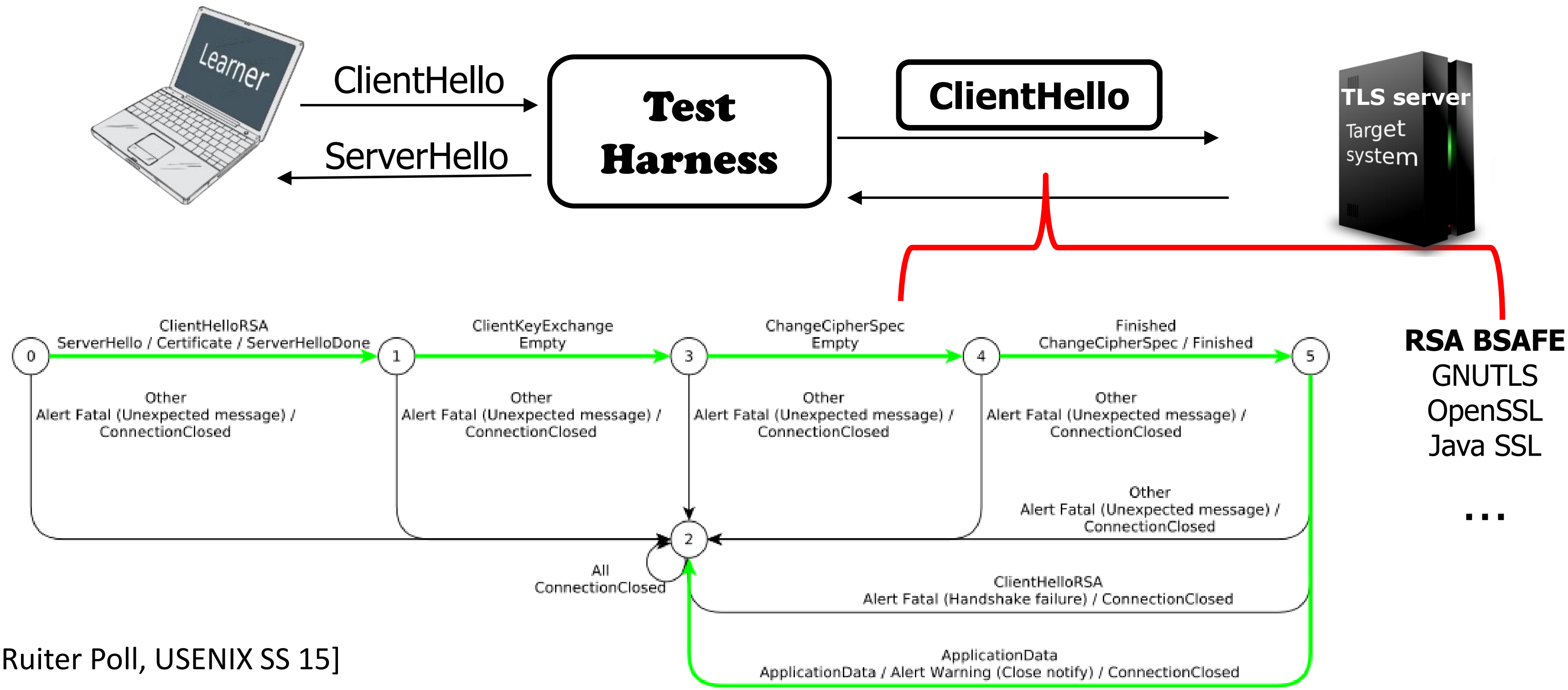
- Learning finite state machines from adaptive test suites
- Starting from only the interface signature, generates
  - Model (finite automaton or Mealy machine), and
  - Conformance test suite
- Techniques well understood for the finite-state case

Applying Model Learning to TLS



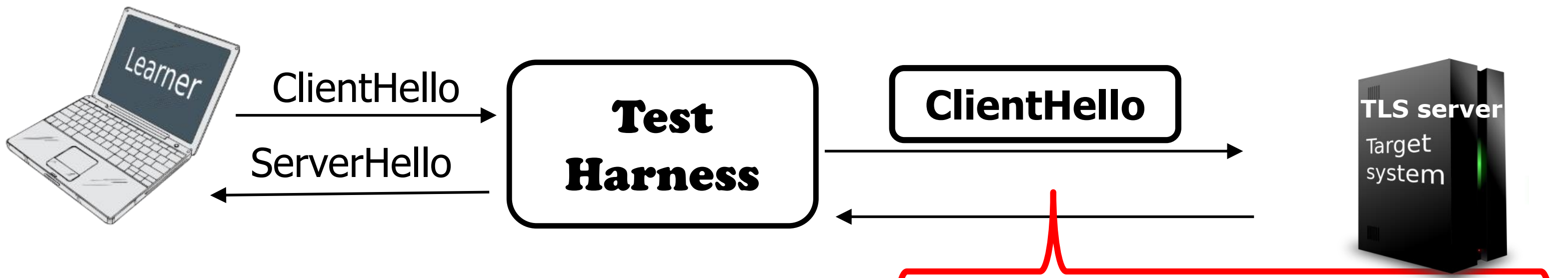

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<2..2^16-2>;
    CompressionMethod compression_methods<1..2^8-1>;
    select (extensions_present) {
        case false:
            struct {};
        case true:
            Extension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

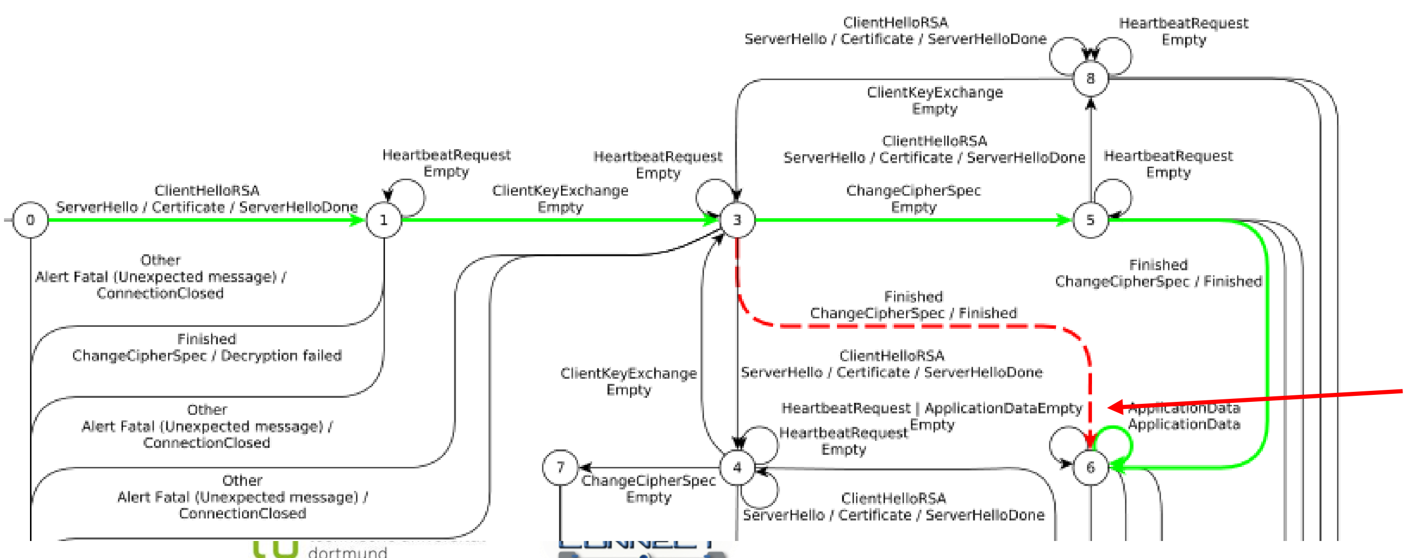
Model Learning: Scenario



[deRuiter Poll, USENIX SS 15]

Model Learning: Scenario





RSA BSAFE
GNUTLS
OpenSSL
Java SSL

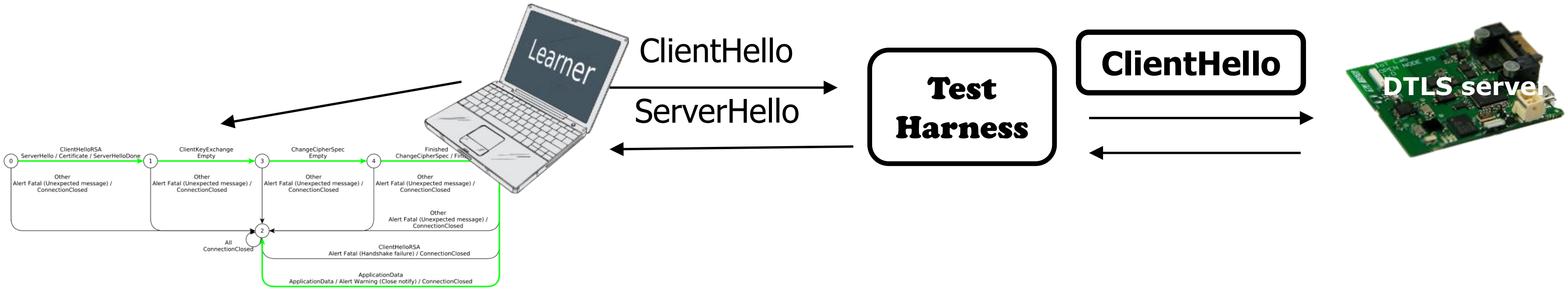
...

unencrypted!

[deRuiter Poll, USENIX SS 15]

UPPSALA UNIVERSITET

aSSIsT: Plans for Testing DTLS



- Build test harness for DTLS,
  - Start from *TLS attacker,* powerful test harness for TLS
    - Developed at Univ. Bochum [Somorovsky, CCS16]
    - Supports many known cryptographic attacks
- Generate state machine model of DTLS implementation
- Use model:
  - Model checking conformance to standard
  - Checking vulnerability to cryptographic attacks
  - As seed for fuzzing

UPPSALA UNIVERSITET

Hardware-based Isolation

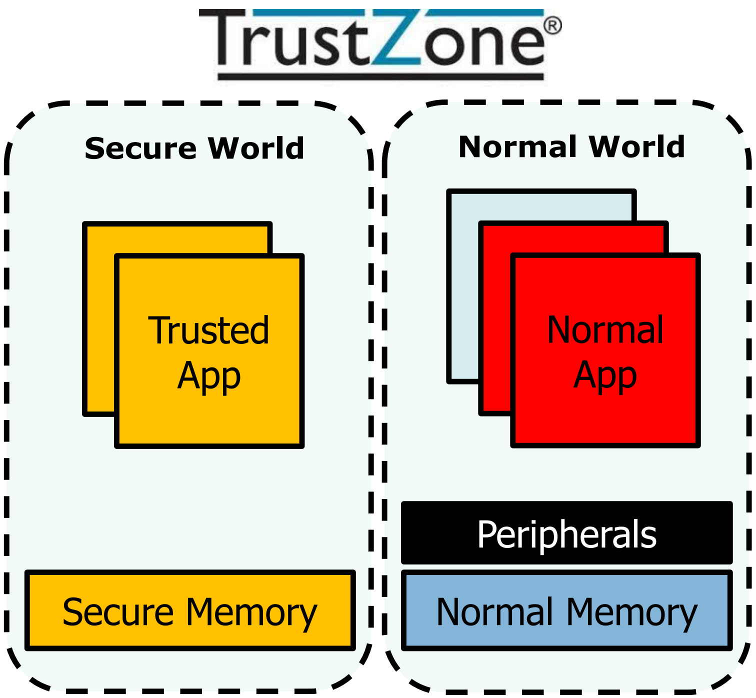
By default, Cortex M0 does not support memory protection/isolation

ARM TrustZone: Hardware-protected area, unaccessible from outside

- Provides memory isolation
- Storage for encryption keys
- …

Challenges

- Best use for Contiki-NG
- Secure communication between trusted and un-trusted zones
- Safe storing of persistent data between transient executions



Enabling (Remote) Updates

SUIT (Software Updates for IoT)

- New protocol for software updates for IoT
- PKI-based

Challenges:

- Implement and evaluate SUIT protocol-set for Contiki
- Formal verification of PKI for IoT (possibly using Tamarin)

Over-The-Air programming (OTA) involving

- Secure bootloader
- Image dissemination (e.g. LWM2M)
- Image verification (signature check)
- Image swapping w/ scratch area

Summary

- Security is a crucial concern for IoT software
- IoT platforms pose new challenges for software security
  - Attack surface, resource constraints, device characteristics
- aSSIsT goal is to develop techniques for
  1. Software analysis for vulnerabilities
  2. V&V for (security) protocol implementations
  3. Platform run-time protection mechanisms
- For use by developers of software for IoT
- For hardening existing IoT platforms
  - Contiki-based, …
  - Low power wireless protocols