IoT Security: A journey through standardization

ARM

Hannes Tschofenig

Edingburgh
03/07/2017

IoT security is a problem,
according to media.

Top 5 IoT device security vulnerabilities

1. No or limited software update mechanism
2. Missing key management
3. Inappropriate access control
4. Missing communication security
5. Vulnerability to physical attacks

Our approach

- Make embedded development more friendly
- Use off-the-shelf Internet security protocols.
- Developed solutions in
  - Hardware
  - OS
  - Device Management / Communication security protocols

TLS/DTLS

- Most popular communication security protocol
- TLS for connection-oriented transports; DTLS for connection-less transports
- 1.2 is the most recent, finalized version
  - ~25 extensions
  - ~340 ciphersuites

- Has been difficult to "phase-out" old TLS versions and old crypto

TLS 1.3

- 1.3 in development since April 2014

- Supposed to an evolutionary development addressing security problems emerged with earlier specifications.

mbed TLS

- Our implementation of TLS/DTLS for embedded devices
- Open source with an Apache 2 license
- Modular design for optimizations and integration of hardware (e.g., new memory allocator, RNG, AES and ECC hardware acceleration)
- Code: https://github.com/ARMmbed/mbedtls

Standardizing TLS

- There are a few rules in the IETF:
  - Open access to specs and discussions
  - Free participation
  - Running code concept
  - No strict timelines
  - Higher document version != fewer changes to expect
  - No official interops (groups organize themselves)

Participating in TLS Standardization

- Important to learn about potential implications and problems ahead of time.
  - Performance implications of certain design decisions
  - Difficulty of integrating new developments into existing code
- Ability to influence the decision making process (particularly since IoT is not the main use case)

- Started implementation work of TLS 1.3 using existing mbed TLS code

A few years forward…

- Specs keep changing (now at version -20)
- More optimizations & more security reviews
- Implementers updated their specs and regularly met at interop events (actually at the IETF Hackathons)

Code: what looked like a small coding project turned into a re-write of our stack.

What was accomplished?

- TLS 1.3 is a re-design of the popular TLS protocol.
  - Makes the handshake faster (Zero-RTT)
  - Shortened algorithm list
  - Improved privacy protection
  - Harmonized extensions

- More formal analysis
- More external involvement

- Specification now in review by the steering group

Lessons learned

- Getting researchers to pay attention to standardization is really hard.
- It worked with TLS 1.3
- Security reviews/formal method analysis did, however, took a long time.

- Writing code during specification phase provides valuable input but is also frustrating.

Can you help?

- Can we use formal methods in protocol development more aggressively?
- How can we do it in a timely manner?
- Is this the job of researchers or standardization experts?
- What are the best techniques?
- Can we produce code from these formal models/descriptions?
- How can we improve testing (in the style of test-driven development)?

What's next?

- DTLS 1.3 – more optimized version for IoT
- QUIC – a new transport protocol

- We hope to release our mbed TLS 1.3 code to the public soon and do some performance analysis / comparison with earlier versions.

ARM