IT Security Practicals
Task 1 - Constructive TLS

Sebastian Ramacher, Philipp Harb, Martin Mandl, Samuel Sprung, Stefan Steinegger

October 10, 2016

Task 0

- Has everyone registered a group?
- Deadline is at 2016.10.10 23:59.

Reminder

- Use your Git repository!
- Git repositories set up after registration deadline.
- Document your method and success in a short README file.
- Add the test result files to your repository.
- Do not forget to tag and `git push`.

Reference VMs

- Now available from the download page.
- Installed packages:
  - cmake ($\geq$ 2.8), pkg-config
  - g++ ($\geq$ 4.9), check
  - git, doxygen
  - eclipse-cdt, gdb, valgrind

Assignment Document

- Goal: reinforce your knowledge about the task.
- For each task answer all questions.
- We do not check the correctness . . . but they will be part of the interviews.

Task 1

- Research and implement some way to get random seeds.
- Implement HMAC-SHA256.
- Implement HMAC based PRF.
- Implement partially implicit nonce counter.
- Implement application data part of TLS using Ascon as cipher.

Task 1: The Bigger Picture

- `TLS_ECHDE_ECDSA_WITH_AES_128_GCM_SHA256`
- `ECHDE_ECDSA`: key exchange and authentication
- `AES_128_GCM`: cipher and potential mode of operation
- `SHA256`: hash function used for pseudorandom function families and message authentication codes
- We look at primitives for encryption, pseudorandom function families and message authentication codes.

Task 1: MACs and PRFs

- Message authentication codes (MAC) provides data integrity and authentication.
- HMAC: MAC from cryptographic hash functions:

$$H(K, m) = h\left((K \oplus O) \| h\left((K \oplus I) \| m\right)\right)$$

- Pseudorandom function (family, PRF): efficiently computable functions which are indistinguishable from a random function.
- TLS PRF: construction based on a HMAC taking a secret, and (public) label and seed. Used to derive various secrets in TLS.

Task 1: PRFs and the Secret

- TLS PRF still requires a secret as input.
- What possibilities are there to fetch random data suitable as cryptographic keys using only the C++ standard library and functionality from the Linux kernel?

Task 1: Authenticated Encryption

- Ciphers (like AES-CBC) used in TLS provide confidentiality.
- . . . but what about integrity and authenticity?
- $\Rightarrow$ TLS combines ciphers with HMACs.
- Implementation of the combination proves to be difficult (see Task 2).

Task 1: Authenticated Encryption

- New scheme: authenticated encryption (AE or AEAD) providing confidentiality, integrity and authenticity.
- AES-GCM or AES-CCM are ciphermodes providing AE.
- TLS 1.2 has been extended to support generic AE ciphers.
- CAESAR competition similar to SHA3 competition to standardize new AE ciphers.
- IAIK submitted ASCON.

Task 1: Nonces

- AE schemes require nonces to provide indistinguishability.
- Needs to be unique, but not required to be random.
- $\Rightarrow$ use a (big endian) counter with fixed high bits.
- The fixed bits are split into an implicit (derived from a common secret) and explicit (chosen at random) part.

Further Reading

- RFC 5116 (An Interface and Algorithms for Authenticated Encryption)
- RFC 5226 (The Transport Layer Security (TLS) Protocol, Version 1.2)
- Dobraunig, Eichlseder, Mendel, Schläffer. Ascon v1.2, Submission to the CAESAR Competition. `http://ascon.iaik.tugraz.at/`

Questions