M.Sc. thesis project(s):
Testing and Fuzzing of Security Protocols

Background  Within the project aSSIst (https://assist-project.github.io/), we are developing techniques for ensuring that implementations of security protocols (as well as other security-critical software) conform to their specifications, and are free from security vulnerabilities. In particular, we care about software for security of IoT devices. A motivation is that the number of Internet of Things (IoT) devices is projected to reach 11.6 billion by 2021, thus constituting half of all devices connected to the Internet.

A key building block in IoT software consist of implementations of security protocol. As an example, let us consider Datagram Transport Layer Security (DTLS) [1, 2], which is one of the primary protocols for securing IoT applications [3]. DTLS is based on TLS, a widely used security protocol responsible for securing communication over a reliable data transfer protocol. Unlike TLS, DTLS is meant to operate over UDP, an unreliable transfer protocol commonly used in IoT systems, but also increasingly being used in Voice over IP, tunneling technologies, and new Web protocols. Whereas significant effort has been invested into ensuring security of TLS implementations, those based on DTLS have so far received considerably less scrutiny.

Within aSSIst and our research group, we are developing techniques to ensure that implementations of DTLS and other security protocols conform to their specifications, and are free from security vulnerabilities. Two major techniques that we consider are: — model based testing, in which an implementation is tested against a specification in the form of a model of its functionality, and — fuzzing, in which large numbers of well-formed and non-wellformed inputs are systematically input to an implementation in order to search for security vulnerabilities, including buffer overflows, memory safety errors, run-time errors, etc. As part of our effort, we welcome contributions from M.Sc. projects for combining, developing, or applying such techniques to IoT security protocols, e.g., DTLS.

Project Basis  As a basis for the project, we have (in a collaboration with the Univ. of Bochum) developed a testing infrastructure for testing DTLS implementations, based on a testing infrastructure for TLS [4]. This infrastructure contains support for composing and parsing protocol packets under strategies which can be easily configured. In a recently completed project, we have used this infrastructure to generate abstract models, in the form of state machines, which describe how a DTLS implementation reacts to a sequence of well-formed protocol messages. We have used these models to search for a particular form of bugs, where a protocol implementation reacts incorrectly to sequences of well-formed messages, which are possible sent in an order that was not foreseen by the implementor. We detected several bugs in widely used implementations, that allow to bypass authentication steps or establish insecure connections.

Possible M.Sc. Project(s)  We regard it as highly interesting to reinforce our current efforts by developing strategies for testing other aspects of the protocol than just how it handles sequences of well-formed messages. We mention here two aspects that can be considered in the project, but others are also possible.

- **fragmentation**: Many protocols should support fragmentation of messages into records, due to limits on packet size in protocols like UDP. According to protocol specifications, such fragmentation should be handled transparently, i.e., it should not affect protocol
functionality nor be noticeable by clients. Testing of other protocols have sometimes revealed sever security bugs that can be exposed by particular ways to inject fragmented messages. An important goal is to develop strategies for “near-exhaustive” testing that an implementation correctly handles all possible ways in which messages can be fragmented.

- injection of non-well-formed messages: Fuzzing in various forms has proven a very effective technique for finding buffer overflows, memory management errors, run-time errors and the like. It has primarily been applied to programs that read a single input, e.g., an image file, and less to protocols that receive a sequence of interdependent input messages. An important goal is to develop strategies for how to create non-well-formed sequences of non-well-formed messages that are good at exposing such bugs, which can sometimes be security-critical.

The protocol can consider DTLS as a primary testing target, using our existing test infrastructure as a basis, but techniques should be generally applicable also to other similar protocols. There are several candidates for protocols that can be the target for an M.Sc. project. One obvious candidate is to extend our previous testing effort, which concerned DTLS servers, to target instead DTLS clients. Previous testing efforts on TLS and other protocols have shown that client implementations tend to be less well-tested than server implementations, thus potentially having at least as many vulnerabilities.

Supervisors and Contact persons  Bengt Jonsson (bengt@it.uu.se), Kostis Sagonas (kostis@it.uu.se), Paul Fiterau-Brostean (paul.fiterau_brostean@it.uu.se).

References