Fuzz Testing the ISO 15118 Protocol Stack
About Me

Tobias Schöneberger

- Vector Department: Research & Development for Innovative Tool Applications
- Teams focus is testing of security
- Personal passion for pentesting
  - Just got OSCP certified
Agenda

- Introduction to Fuzz Testing
- Overview ISO 15118
- TLS Testing Challenges
- TLS Testing Example
- Conclusion & Outlook
Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program.

Functional test are mostly defined and written manually
- A limited set of tests is created. Negative tests are sometimes neglected and often do not cover the entire spectrum of inputs
- Example: CANoe Test Package EV with conformance tests

Fuzzing
- Only configuration is needed
- Functional test data can be used as starting point
- Creates lots of tests (positive & negative)
- Fuzz tests go far beyond the typical checks for all reasonably foreseeable misuse
- Good cost-benefit ratio due to automated test case generation

Both should be done, fuzzing does not validate your implemented features
- Fuzzing does enhance the test for robustness and negative invalid input greatly
Overview ISO 15118

- ISO 15118: “Road Vehicles – Vehicle to grid communication interface”
- Negotiate charging parameters
- Authenticate for Plug & Charge

Security View
- Charging communication is an external attack interface
- Easy physical access
- Direct monetary gain by stealing charging credentials
The ISO 15118 Protocol Stack

- V2G Application messages
- Efficient XML Interchange
- V2G Transport Protocol

Test requirements for every protocol:
> Full control over payload one level below
> Example: For TLS tests, control over TCP payload is needed
Complex handshake with optional elements

- Fuzzing support for client and server role
  - ISO 15118: car is client and charging point server

- Support for malformed message sequences
- Predefined automated sequence tests
  - Duplicated messages
  - Skipped messages

- State machine transitions are hidden in library
  - `handshakeUntil("ServerHelloDone")`
  - `handshakeUntil("ServerChangeCipherSpec")`
TLS Testing Challenges

TLS Testing: Complex Data Structures

TLS

- Abstract protocol model where every field can be accessed
  - Special handling for length and control fields
  - Enums for fields with fixed meaning
  - Invalid field values are allowed
    - E.g. TLS version 255.255
- Automated Fuzzing to test all kinds of malformed data

Certificates

- Access to fields with SecurityManager API
  - E.g. set validity date into past
- DER-Format-Fuzzer created to fuzz certificates
  - Fuzzes the X.509v3 data format itself
  - Fuzzes field contents
  - Resigning with SecurityManager
TLS Testing Challenges

TLS Testing: Cryptography

TLS Communication is not readable and difficult to debug
- CANoe support for TLS, set master secret and communication is readable
- TLS Alerts and Application data is readable

Computing of secrets / hashes / nounces / randomness
- Test primitives handle the crypto, the complexity is hidden in the library
- *Never* roll your own crypto
  - API Access to every parameter and field of the TLS handshake
- Support for ECDSA and client authentication

Handling of secret (pre)production certificates
- Security Manager for the storage of secrets and certificates
  - No problems with certificate formats

Vector Security Manager
TLS Testing Example

TLS Fuzzing Test Case

1. Preparation
   ▶ Reset
   ▶ Initiate lower-level connection

2. Stimulation (Fuzzing)
   ▶ Manipulation of message field

3. Observation
   ▶ Check DUT answer
   ▶ Ping to detect fatal error
Conclusion & Outlook

We are creating a test suite for efficient security tests of ISO-15118
- Fuzzing of all protocol layers
- Additional security tests
- Integrated into vTESTstudio and CANoe
- Can be extended with C# or CAPL test code to match test specifications

The TCP, IPv6 and TLS fuzzer and test can be used for all TLS-secured protocols
- TLS encrypted DoIP
- HTTPS connections to a backend
Your questions are welcome!

Author:
Schöneberger, Tobias
Vector Germany