Dolev-Yao-Model-Guided Fuzzing of Cryptographic Protocols

City and country Nancy, France.

Team or project in the lab
Team PESTO at LORIA lab (Inria Nancy, CNRS, and Université de Lorraine).

Name and email address of the advisors
Lucca Hirschi, `lucca.hirschi@inria.fr` & Steve Kremer, `steve.kremer@inria.fr`

Name and email of the head of the laboratory
Yannick Toussaint, `yannick.toussaint@loria.fr`

Indemnisation
The internship is supported by the ProtoFuzz ANR (program managed by the French National Research Agency under grant agreement No. ANR-22-CE48-0017).

TL;DR. The goal is to propose, implement, and evaluate new design ideas for the tlspuffin fuzzer [1] (part of our ProtoFuzz project).

Context. Critical and widely used cryptographic protocols have repeatedly been found to contain flaws in their design and their implementation. A prominent class of such vulnerabilities is logical attacks, e.g., attacks that exploit flawed protocol logic. Automated formal verification methods, based on the Dolev-Yao (DY) attacker, formally define and excel at finding such flaws, but operate only on abstract specification models. Fully automated verification of existing protocol implementations is today still out of reach. This leaves open whether such implementations are secure.

On the opposite side of the spectrum, *fuzz testing*, developed since the 90s, is now the gold standard for testing security software and is used at scale by the largest software companies. However, even if a protocol implementation is tested against memory-related vulnerabilities using state-of-the-art fuzzers, the whole class of *implementation-level logical attacks* remains out of scope. Unfortunately, this blind spot hides numerous attacks, notably recent logical attacks on widely used TLS implementations introduced by implementation bugs.

We have recently answered by proposing a novel and effective technique that we called DY model-guided fuzzing [1], which precludes logical attacks against protocol implementations. The main idea is to consider as possible test cases the set of abstract DY executions of the formal DY attacker, and use a novel *mutation-based fuzzer* to explore this set. This approach enables reasoning at a more structural and security-related level of messages represented as *formal terms* (e.g., decrypt a message and re-encrypt it with a different key) as opposed to random bit-level modifications that are much less likely to produce relevant logical adversarial behaviors. We have implemented a full-fledged and modular DY protocol fuzzer tlspuffin. We have demonstrated its effectiveness by fuzzing three popular TLS implementations, resulting in the discovery of four novel vulnerabilities[1].

This recent work has opened up various exciting new research questions we would like to explore in this internship. In particular, by building on this prior work, the main objectives are to design and implement a DY fuzzer feedback metric and to reflect the varieties of DY security properties in the fuzzer engine.

[1] CVE-2022-42905 (critical severity), CVE-2022-42905 and CVE-2022-42905 (medium severity), and CVE-2022-38153 (low severity).

Intern's tasks. First of all, the intern will get familiar with formal verification in DY models, fuzzing, as well as with the existing Rust code base.

Next, depending on the intern's affinity for and knowledge of the different involved aspects of this project, we will be able to adapt the project goals and choose one among several research directions, such as:

1. tighten the link between our fuzzing approach and formal methods and symbolic verifiers which are able to reason about protocols using formal logic,
2. design domain-specific feedback metric to incentivize the fuzzer to seek for new symbolic traces. The underlying fundamental question is: what is a good "symbolic feedback" that promotes semantically different symbolic traces?
3. define scoring metrics that can be effectively computed by symbolic verifiers and that can help the fuzzer promoting test cases that are close to attack traces.
4. design new fuzzing mutations and benchmark them with our test-bed,
5. design an efficient grammar-based fuzzing engine and evaluate it with our test-bed.
6. Design and implement a semi-automatic way to specify the DY model of a protocol (in particular the message model), given as input to tlspuffin. For instance, this could leverage static analysis or code introspection of protocol implementations.

The precise direction this project will take shall be agreed upon with the intern at the beginning of the project. It can be more theory-oriented (1 and 2) or more practical-oriented (3-6). Should we find any vulnerability, we would follow standard and ethical responsible disclosure practices.

Expected ability of the student. We expect mathematical maturity and basic knowledge in logic and theoretical computer science. Knowledge in security and cryptography is not mandatory but is definitely a plus. For the implementation, a good command of Rust is a plus.

If the candidate is interested, continuation towards a PhD, for which we already have funding, is possible.

References

[1] Max Ammann, Lucca Hirschi, and Steve Kremer. DY fuzzing: formal dolev-yao models meet cryptographic protocol fuzz testing. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1481–1499. IEEE, 2024.