Other state-of-the-art fuzzers do not find those, we do thanks to

- TLSPU
- Term-Level Mutations
- DY Mutations

Example:

- IEEE
- TLSPU
- Built on
- Replace-Reuse
- Replace-Match
- Generate
- Repeat

No structural message/message flow modification, e.g. negligible probability of computing crypto and other structural modifications through bit-level mutations

Test cases = function symbols
- ✗ Fuzzing Limitation 1: Reachability
- ✗ String-level mutations only

Harness: Mapper + Executor
- To each function symbol, we build an interpretation \[ J : \alpha \xrightarrow{} \alpha \]
- Mapper can interpret any term by recursively applying interpretations: \( \text{Terms} \rightarrow \alpha \)
- Mapper is protocol-dependent but PUT-independent and can be built once-for-all on top of a reference implementation or any PUT

DY Fuzzer = DY attacker $$\Psi$$ in a fuzzing loop

DY Objective Oracle
- Memory-related objective oracle
  - Classical with bit-level fuzzing: code instrumentation with AddressSanitizer (ASan)
  - DY security properties checking
    - Introduce \( \text{claims} \) triggered by roles executing the PUT
    - E.g. agreement claims: Agr(client, pk, m)\( \Rightarrow \) means “client believes to have agreed with a server with public key \( pk \) on m in action \( i \)"
    - As in DY models: security properties expressed as 1st-order formula
      - E.g. auth. vs. m: Agr(client, pk, m)\( \Rightarrow \) Run(server, pk, m, R) \( \wedge \) j
- Objective Oracle always checks those properties by first applying \( \cdot \rightarrow \)

Future Work
- Code-coverage is a poor metric prone to exhaustion, we plan to design a domain-specific DY-based notion of coverage
- Explore differential fuzzing + extend objective oracle (with more properties and compromise scenarios)
- Combine DY fuzzing with bit-level fuzzing: reach deep states and then smash PUTs with bit-level mutations
- Apply DY fuzzing to more protocols (e.g. WPA*, TelCo, etc.) and PUTs
- Partially automate the Mapper (and Harness) → PUT/Protocol-agnostic
- Connect further with DY verification tools (ProVerif/Tamarin)

Check our website: https://tlspu.github.io

IEEE Security and Privacy, May 20, 2024