Fuzz-Testing of Proprietary SCADA/Control Network Protocols

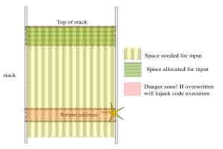
TCIPG

Rebecca Shapiro, Sergey Bratus, and Sean W. Smith

Motivation

Software bugs are more than just a nuisance. They often result from poorly implemented input validation or memory management. These **vulnerabilities** can be exploited by attackers and have historically been a **major source of security problems**.

Memory corruption



Code Injection

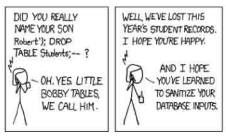


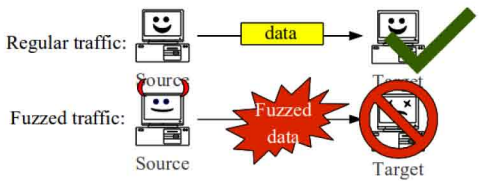
Image source: XKCD

Power Grid Applications

- Be proactive in finding bugs in power applications and equipment.
- If industry is aware of bugs in their applications and equipment, they can protect themselves better.
  - Can a compromised machine on a control network crash the EMS?
  - Are the SCADA protocols brittle and susceptible to injection of malicious data?

Research Plan

- The purpose of a fuzzer is to **craft input that will trigger bugs.**



Challenges of Fuzzing

- Create inputs that are well-formed enough to **pass sanity checks** but malformed enough to **trigger bugs**.
- Craft a variety of inputs that **maximize code coverage** on target.

Fuzzing Proprietary Protocols

- Modern fuzzing tools require a **domain expert's knowledge** of a protocol.
- Domain experts need a **fuzzing expert's help** to fuzz their equipment.
- Domain experts are **reluctant to share protocol details** with fuzzing experts.
- To understand undocumented protocols, we need to use a **debugger** [1] [2].
- Domain experts **may not have the option of installing (or time to install) a debugger on their equipment** to reverse-engineer protocols.

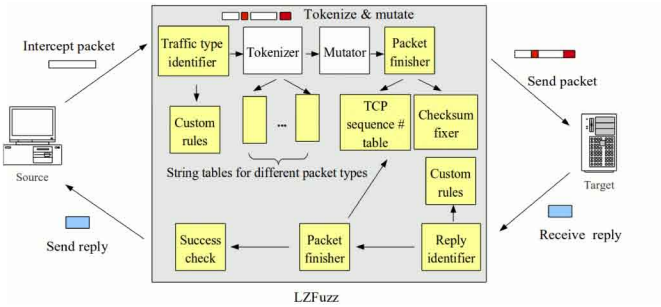
Goals

- Build a fuzzing appliance that a domain expert can use.
- Flexibly work with no, partial, or full protocol knowledge.
- Transparently capture packets in transit to and from target.
- Require no configuration changes to source or target.
- **Empower the domain expert** to fuzz proprietary equipment.

Previous Results

- LZFuzz 1.0, the first generation of LZFuzz, was built as a prototype [3].
- LZFuzz 1.0 successfully fuzzed SCADA equipment.
- LZFuzz 1.0 lacked the usability and flexibility needed in order to be useful for a domain expert.

LZFuzz 2.0 Design



Module part of original LZFuzz

Module new to LZFuzz 2.0

Future Efforts

- Run experiments to **test effectiveness** of fuzzer in comparison to other state-of-the-art fuzzers.
- Work with industry partners to test fuzzer in **real-world settings**.
- Refine interface to make it more **usable** by a non-fuzzing-expert.
- **Enhance logging** capabilities so that successful fuzzing sessions can be studied and replayed.
- **Design and build traffic analyzer** to help users identify different types of traffic used by protocol.
- Build module that **searches for and fixes traffic checksums**.

References

1. C. Cadar, V. Ganesh, P. Pawlowski, D. Dill, D. Engler. "EXE: Automatically generating inputs of death." *ACM Transactions on Information and System Security (TISSEC)*.
2. P. Godefroid, A. Kiezun, M.Y. Levin. "Grammar-based whitebox fuzzing." *ACM SIGPLAN Notices.* 2008.
3. S. Bratus, A. Hansen, A. Shubina. "LZfuzz: A fast compression-based fuzzer for poorly documented protocols." Dartmouth Computer Science Technical Report TR2008-634. 2008.