WebTransport Protocol Fuzzing

Bakalárska práca

Autor: Vladyslav Havriuk

Školiteľ: doc. RNDr. Martin Stanek, PhD.

https://github.com/qbitroot/webtransport-fuzzer

Čo je WebTransport?

- Moderný protokol pre obojsmernú komunikáciu klient-server.
- Postavený na **HTTP/3** a **QUIC**.
- Kombinuje nízku latenciu (ako UDP) s bezpečnosťou a spoľahlivosťou (TLS 1.3, Congestion Control).
- **Cieľ práce:** Testovanie bezpečnosti tohto protokolu.

Čo je Fuzzing?

- **Definition:** Automated software testing method.
- **Principle:** Injecting unexpected, random, or invalid inputs (malformed data).
- **Goal:** Discover crashes, memory leaks, and security vulnerabilities.
- *"Throwing bricks at windows to see if the house collapses."*

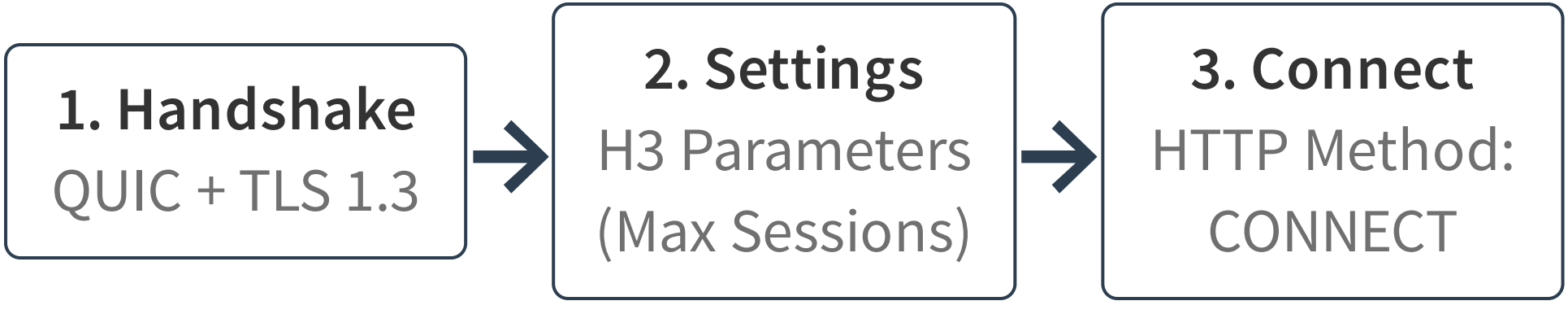
Architektúra protokolov

Protokol	Rola	Vrstva
QUIC	Cesta (The Road)	Transport (L4)
HTTP/3	Dopravné predpisy	Application (L7)
WebTransport	Náklad (Cargo)	Application (L7+)

WebTransport využíva HTTP/3 na "setup", ale potom posiela dáta priamo cez QUIC streamy.

Fáza 1: Nadviazanie spojenia

Pred fuzzingom musíme prejsť striktným bezpečnostným procesom.



Až po úspešnom kroku 3 sme v stave "OPEN" a môžeme fuzzovať.

Fáza 2: Fuzzing Loop

BOOFUZZ

Generuje invalidné VarInts, Payloady...

STATE: OPEN

- Inject: Malformed Capsules
- Inject: Invalid Stream IDs
- Inject: Overflowed Datagrams

Fuzzing Instrumentation

What is Instrumentation?

Technika, ktorá nám umožňuje "vidieť dovnútra" bežiaceho programu.

- Code Coverage tracking (execution paths).
- Crash and Error detection mechanisms.
- Feedback loop to guide the fuzzer.

Graybox Fuzzing via Logging

Language Agnostic Approach

The Problem

Tradičný graybox (napr. AFL++) vyžaduje kompiláciu so špeciálnou inštrumentáciou, čo je ťažké pre rôzne jazyky (Rust, Go, Python).

The Solution: Logs

Echo Server writes execution logs in a standardized format that our Fuzzer understands.

Fuzzer číta tieto logy a "vidí", v akom stave je server, bez priameho prístupu k pamäti.

Ciele testovania (Targets)

Testujeme rôzne implementácie WebTransport Echo Serverov:

Rust (wtransport)

Go (webtransport-go)

Python (aioquic)

Node.js (Socket.IO)

Implementácia

Technológia: Python + BooFuzz + aioquic

- **Shim Architektúra:** Bridge medzi fuzzerom a šifrovaným spojením.
- **State Machine:** BooFuzz kontroluje logickú postupnosť (Connect -> Open Stream -> Send Data).

Záver: Vytvárame univerzálny nástroj na overenie robustnosti WebTransport implementácií.

Current Findings

wtransport (Rust)

Reachable Assertion Bug

Objavený v štandardnom echo serveri z
github.com/BiagioFesta/wtransport

- **Trigger:** Odoslanie DRAIN capsule bez korektného HTTP/3 frame.
- **Impact:** Worker thread panic (nie crash celého servera).
- **Problém:** V Rust-e by knižnica nemala nikdy panikovať.

Future Plan (1/2)

11.02: One-Shot Fuzzer

Odoslanie jednej malformed permutovanej požiadavky + monitoring. Po každej fuzzed session vykonáme validnú session na detekciu memory corruption.

14.02: Log Monitor

Fuzzer spustí server proces a zachytí jeho output. Logy sa uložia do SQLite DB pre kategorizáciu unique failure modes.

18.02: Cross-Language Testing

Spustenie fuzzera na Rust, Go, Python, Node.js. Dokumentácia findings a analýza príčin v library kóde.

Future Plan (2/2)

28.02: Multistep Fuzzing

Testovanie hlbších scenárov. Permutatácia Boofuzz state machine - duplicates, valid messages in invalid states.

01.03: Final Testing & Refinement

Opakované spustenie fuzzera, dokumentácia a finálne vylepšenia.

Ďakujem za pozornosť