IsaJafarov / **Q3Fuzz** Public

Code | Issues | Pull requests | Actions | Projects | Security and quality | Insights

main | 1 Branch | 0 Tags | Go to file | Go to file | Code

IsaJafarov delete unnecessary output keylog file 04dd7d6 · 3 days ago

evaluation	Updated evaluation instructions	4 days ago
exploits	added all exploits	2 months ago
main	delete unnecessary output keylog file	3 days ago
result	Regenerated SMs. Restructured SM gener…	2 months ago
sample_traffics	Regenerated SMs. Restructured SM gener…	2 months ago
servers_setup	small changes to server installation	2 months ago
.gitignore	libs folder	2 months ago
README.md	Add Proxygen and Kestrel vulnerabilities to…	4 days ago

README

Q3Fuzz

Q3Fuzz is a novel black-box framework to model two network protocols (QUIC and HTTP/3) together and conduct multi-layered stateful fuzzing. It constructs the model based on observed network traces by treating the underlying layer and its encapsulated payload as co-occurring events. It then uses the model to direct both mutation-based and generation-based fuzzing.

The work is published at the IEEE DSN 2026 conference.

#	Vendor	CWE	Vulnerability Impact	Zero-day	# Attack Vectors
1	Proxygen	CWE-119	Server crash: `Segmentation fault`	Yes	3+
2	Xquic	CWE-119	Server crash: `Segmentation fault`	Yes	1
3	Quicly	CWE-617	Server crash: `Assertion 'quicly_num_streams(conn) == 0' failed`	Yes (CVE-2025-61684)	3+
4	Quicly	CWE-617	Server crash: `Assertion 'v <= 4611686018427387903' failed`	Yes (CVE-2025-61684)	2
5	Quicly	CWE-617	Server crash: `Assertion 'iter->p->acked == quicly_sentmap__type_packet' failed`	Yes (CVE-2025-61684)	3+
6	Quicly	CWE-617	Server crash: `Assertion '!"invalid CID sequence number"' failed`	Yes	1
7	Neqo	CWE-617	Server crash: `internal error: entered unreachable code`	No	2
8	Quiche	CWE-248	Server crash: `called Result::unwrap() on an Err value: Done`	Yes	1

#	Vendor	CWE	Vulnerability Impact	Zero-day	# Attack Vectors
9	Neqo	CWE-248	Server crash: `called Result::unwrap() on an Err value: InvalidStreamId`	Yes	1
10	Neqo	CWE-248	Server crash: `called Result::unwrap() on an Err value: TransportError(InvalidStreamId)`	Yes	3+
11	Neqo	CWE-248	Server crash: `called Result::unwrap() on an Err value: Transport(InvalidStreamId)`	Yes	3+
12	Neqo	CWE-190	Server crash: `Varint value too large`	No	2
13	Kestrel	CWE-400	Low-rate DoS due to high CPU usage	Yes (CVE-2026-25667))	1
14	Quiche	CWE-400	Low-rate DoS due to high CPU and memory usage	Yes	1
15	Xquic	CWE-400	Low-rate DoS due to high disk and memory usage	Yes	1
16	Aioquic	CWE-401	Low-rate DoS due to memory leak	Yes	1
17	Neqo	CWE-401	Out-of-Memory: Kernel kills server process	Yes	1

Prerequisites

Install the dependencies

```
$ sudo add-apt-repository ppa:wireshark-dev/stable
$ sudo apt update
$ sudo apt install -y tshark=4.6.3-1~ubuntu22.04.0~ppa1 graphviz graphviz-dev
$ pip3 install pyshark==0.6 aioquic==1.2.0 networkx==3.1 hypothesis==6.113.0 rich transitions[diagrams] paramiko
```

The methodology consists of three main stages:

1. Generating QUIC-HTTP/3 Traffic
2. Generating a state machine based on the recorded traffic
3. Fuzzing using the State Machine

Step 1: Generating Traffic

Sample traffic is provided in the sample_traffics folder. You can also generate a new one following the steps below.

autoinstall.py and autorun.py allows to easily build and run a wide range of QUIC-HTTP/3 servers.

Once the installation process completes, the server automatically runs. You can run the server later without reinstalling using the autorun.py script with the same command-line flags.

By default, the servers use `/usr/local/nginx/html/` folder as their web root. Make sure the folder exists with a small `index.html` file.

After running the server, generate a sample QUIC-HTTP/3 traffic by communicating with the server using a client. We generated traffic using Firefox 132.0. First, instruct Firefox to use QUIC connection. On Firefox, update `about:config` preferences:

- `network.http.http3.alt-svc-mapping-for-testing`: `localhost;h3=":12345"`
- `network.http.http3.disable_when_third_party_roots_found`: `false`

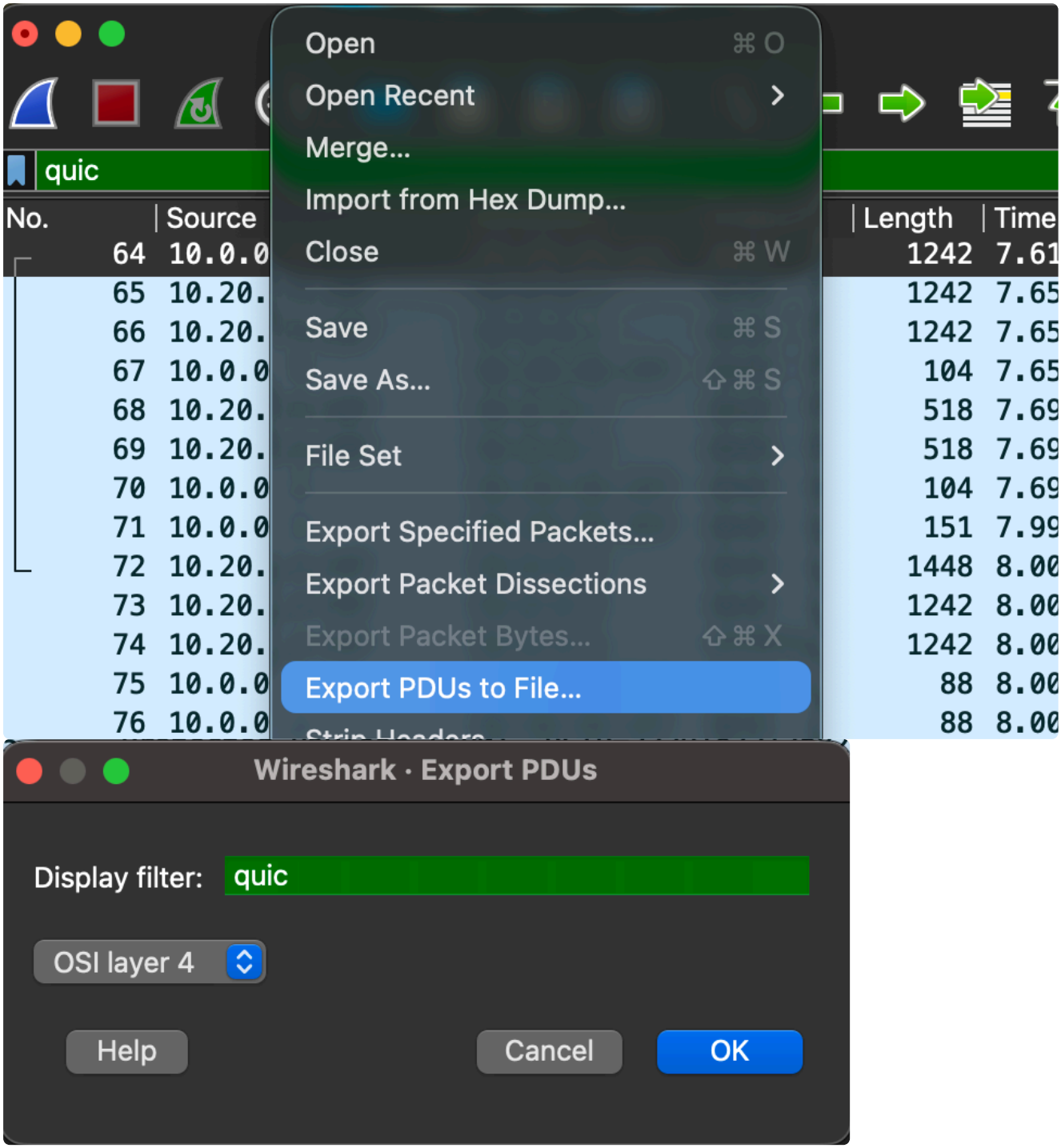
With Wireshark listen for the QUIC communication.

Set `SSLKEYLOGFILE` environment variable and start the client.

```
$ export SSLKEYLOGFILE=./secrets.keylog
$ ./firefox --private-window https://<server-address>/
```

Once the page is loaded, stop capturing packets on wireshark. Using the secrets from the `./secrets.keylog` file, decrypt the captured traffic

We are interested only in QUIC and HTTP/3 layers. Therefore, export PDU.



Step 2: State machine generation

Sample state machines are provided in the result folder.

You can also generate a new state machine given a target server is running and a .pcapng file.

```
$ python3 generate_sm.py http3://q3fuzz.com ../sample_traffics/ff_quicgo.pcapng -dk ../sample_traffics/secrets.keylog
```

The generated state machine will be saved in the result folder.

Step 3: Fuzzing

Once the given traffic model is defined, you can start fuzzing the target server. The fuzzer needs the generated state machine for guidance, as well as the traffic extract/mutate/replay messages.

Two types of fuzzers are provided:

1. Mutation-based fuzzer

Sample command to run the mutation-based fuzzer to target `https://q3fuzz.com` server using the `ff_quicgo.pcapng` traffic (decoded with `secrets.keylog`) and `level_3.json` model.

```
$ python3 mutation_fuzzer.py https://q3fuzz.com ./sample_traffics/ff_quicgo.pcapng ./result/ff_quicgo/level_3.json
-dk ./sample_traffics/secrets.keylog -p5 -i1 -d10 -m1000 -v
```

The fuzzer applies `1000` mutations to each message, sends to the server over 5 connections every second (`1`) for the duration of 10 seconds. Sending a message multiple times for some time is useful to detect slow-rate DoS vulnerabilities. If you want to detect vulnerabilities triggered by a single message, set `-p1 -i1 -d1`.

2. Generation-based fuzzer

Sample command to run the mutation-based fuzzer to target `https://q3fuzz.com` server using the `ff_quicgo.pcapng` traffic (decoded with `secrets.keylog`) and `level_3.json` model.

Releases

No releases published

Packages

No packages published

Contributors 2


IsaJafarov Isa Jafarov


choonginlee

Languages

Python 99.9% **C#** 0.1%