Penetration Testing of HTTP/3.0 Servers

Shaan Kumar, *19074015*,
Naresh Kumar, *19075047*,
Nikita, *19075048*

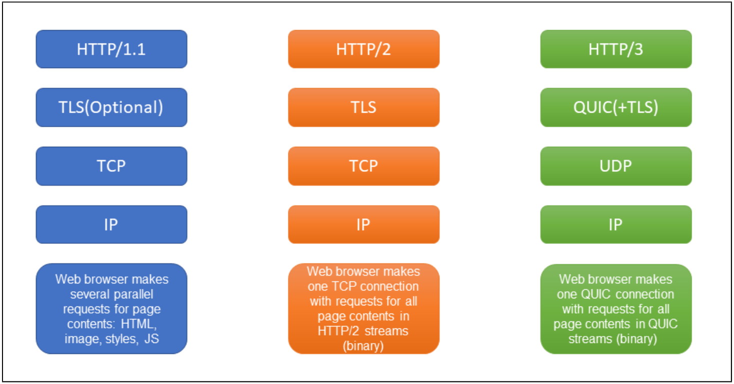
Under the guidance of
Dr. Mayank Swarnkar, Assistant Professor,
Department of Computer Science and Engineering
Indian Institute of Technology (BHU), Varanasi

November 29, 2022

Table of Contents

1. Introduction
2. Problem Statement
3. Literature Survey
4. Motivation
5. Proposed Work
6. Experimental Setup
7. Study
8. Experimental Results
9. Conclusion
10. References

HTTP Protocols



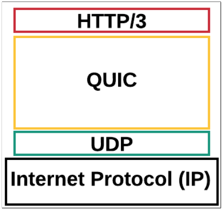
HTTP/1.1	HTTP/2	HTTP/3
TLS(Optional)	TLS	QUIC(+TLS)
TCP	TCP	UDP
IP	IP	IP
Web browser makes several parallel requests for page contents: HTML, image, styles, JS	Web browser makes one TCP connection with requests for all page contents in HTTP/2 streams (binary)	Web browser makes one QUIC connection with requests for all page contents in QUIC streams (binary)

Figure: 1. HTTP Versions

Introduction

QUIC Transport Protocol

- fast, secure, evolvable generic transport protocol
- integrates with TLS, uses connection IDs frames
- reduced connection establishment time
- supports multiplexing
- solves head-of-line blocking problem in HTTP/2



Problem Statement

- Penetration testing of HTTP/3 servers for detection of new vulnerabilities.

Literature Survey

Author et al	**Proposed Work**	**Findings / Synopsis**
Xudong Cao, Shangru Zhao, Yuqing Zhang, 2019 [1]	0-RTT Attack and Defense of QUIC Protocol	Tests the security mechanism of QUIC, & proposes a new attack against the protocol, proves feasibility of attack through experiments.
Adam Langley, Alistair Riddoch, Alyssa Wilk, 2017 [4]	The QUIC Transport Protocol: Design and Internet-Scale Deployment	Layering enables modularity but often at the cost of performance. Squashing the layers of HTTPS in QUIC allows to weed out inefficiencies in the HTTPS stack.
Efstratios Chatzoglou1, Vasileios Kouliaridis1, Georgios Karopoulos2 Georgios Kambourakisa, 2015 [2]	Revisiting QUIC attacks: A comprehensive review on QUIC security and a hands-on study Quick is QUIC?	1. A hands-on security evaluation performed against the six most popular QUIC and HTTP/3 enabled servers. 2. Identifying attacks against both IETF QUIC and gQUIC components.
Robert Lychev, Samuel Jero, Alexandra Boldyreva, 2015 [5]	How Secure and Quick is QUIC? Provable Security and Performance Analyses	1. In presence of attackers, QUIC maybe unable to attain 0-RTT connections. 2. Analysed the pitfalls of designing performance-driven secure protocols.
Igor Nogueira de Oliveira, Rafael Roque Aschoff, 2018 [7]	QUIC and TCP: A Performance Evaluation	Influence of RTT in the experiment was noticeable while packet loss ratio influence was inexpressive.

Literature Survey

Author et al	**Proposed Work**	**Shortcoming of work**
Robin Marx, Joris Herbots Wim Lamotte, Peter Quax, 2020 [6]	Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity	Analysed behaviour of 15 different QUIC implementations based on features such as Flow Control, Congestion Control, Prioritization and 0-RTT etc
Mehdi Yosofie, Benedikt Jaeger, 2019	Recent Progress on the QUIC Protocol	Discussed testing QUIC in production mode within Chrome/ Chromium on YouTube and other Google services by Google.
Sarah Cook, Bertrand Mathieu, Patrick Truong, 2017 [3]	QUIC: Better For What And For Whom?	QUIC outperforms HTTP/2 over TCP/TLS in unstable networks such as wireless mobile networks.

Motivation

- HTTP/3 a very recent HTTP protocol, eyecandy for hackers.
- HTTP/3 not allowed by many networks.
- Adoption of HTTP/3 is increasing so deciding which HTTP/3 implementation is better for use maybe helpful.
- To analyze features of HTTP/3 servers : faster connection set-up, less Head of Line blocking, connection migration.

Proposed Work

- To try and compare implementations of HTTP/3 supporting servers like **aioquic, nginx-quiche, openlitespeed, Cloudflare**.
- Study the behaviour of various servers implementing HTTP/3 protocol by sending forgery packets.
- To compare different servers on the basis of performance i.e. HANDSHAKE TIME and PACKET RX time in handling forgery packets.

Experimental Setup

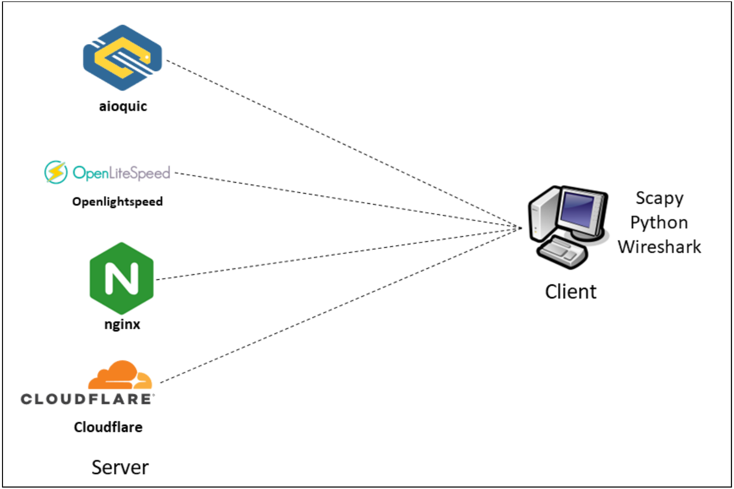


Figure: 2. Setup for Penetration Testing

Study

- Learned about HTTP protocols:.
    - **HTTP/1.1**: Several parallel requests for page contents: HTML, image, styles, JS.
    - **HTTP/2**: One TCP connection with requests for all page contents in HTTP/2 streams (binary).
    - **HTTP/3**: One QUIC connection with requests for all page contents in QUIC streams (binary).
- **Head of Line blocking** problem in HTTP/2.

Work Done

- Results of penetration testing performed on HTTP/3 servers are as:

Experiment Number	Parameters Changed
1	Set Version number to zero
2	Set Version number to a positive value
3	Changing fixed bit in the public flag
4	Changing packet number length in the public flag
5	Buffer Overflow

Tools Used: Scapy, Wireshark, http3check.net

Work Done

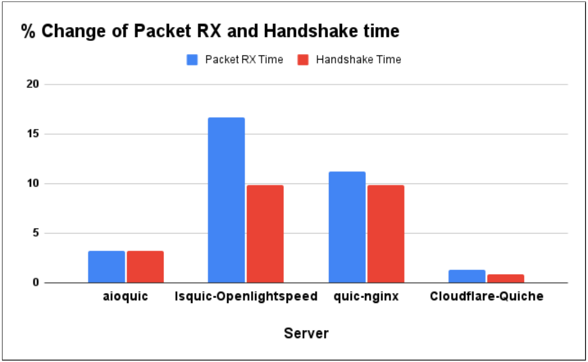
Steps for performing experiment:

- Start by taking a QUIC(HTTP/3) packet.
- Modify the data in packet using Scapy.
- Flood the test server by sending this QUIC packet.
- Measure PACKET RX time and HANDSHAKE time with the help of `http3check.net`
    - **PACKET RX** - Time between the first packet sent and the first packet received (measured in milliseconds).
    - **HANDSHAKE TIME** - Time between when the first packet is sent and when the handshake is completed (measured in milliseconds).

Work Done

Experiment 1: Set Version number to zero.

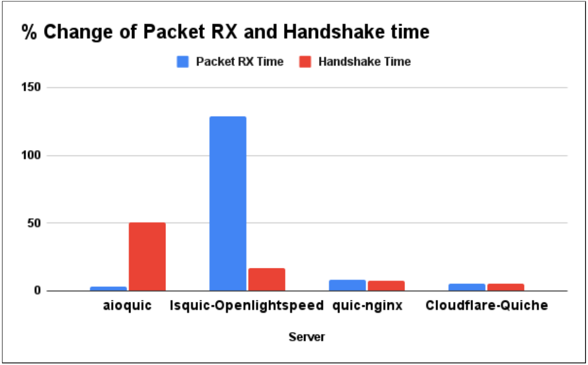
Server	aioquic		lsquic-Openlightspeed		quic-nginx		Cloudflare-Quiche	
	Before	After	Before	After	Before	After	Before	After
PACKET RX	91.67	94.64	9.64	11.24	85.74	76.11	155.66	153.62
HANDSHAKE	181.94	187.77	20.42	22.43	175.22	157.99	157.40	156.11



Work Done

Experiment 2: Set Version number to a positive value.

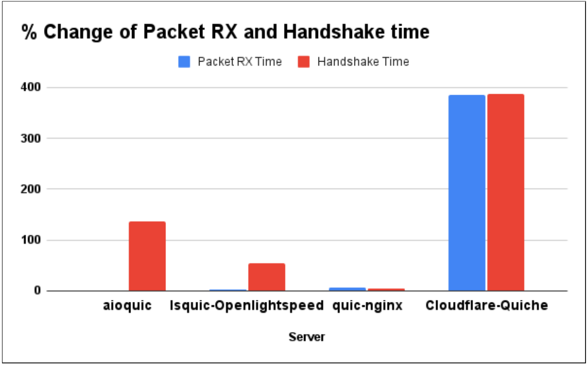
Server	aioquic		lsquic-Openlightspeed		quic-nginx		Cloudflare-Quiche	
	Before	After	Before	After	Before	After	Before	After
PACKET RX	91.67	94.85	9.64	22.07	85.74	78.87	155.66	164.19
HANDSHAKE	181.94	273.57	20.42	23.87	175.22	162.01	157.40	165.95



Work Done

Experiment 3: Changing fixed bit in the public flag.

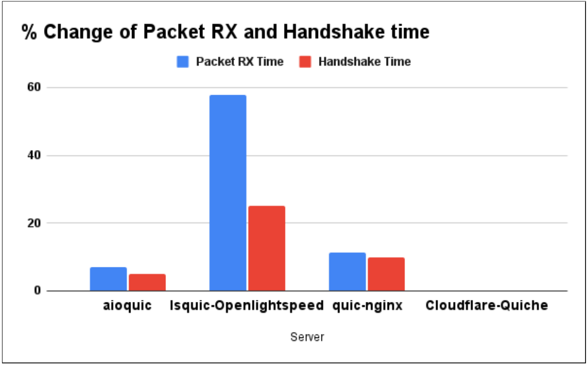
Server	aioquic		lsquic-Openlightspeed		quic-nginx		Cloudflare-Quiche	
	Before	After	Before	After	Before	After	Before	After
PACKET RX	91.67	91.17	9.64	9.335	85.74	79.96	155.66	755.77
HANDSHAKE	181.94	431.86	20.42	31.70	175.22	167.91	157.40	767.24



Work Done

Experiment 4: Changing packet number length in the public flag.

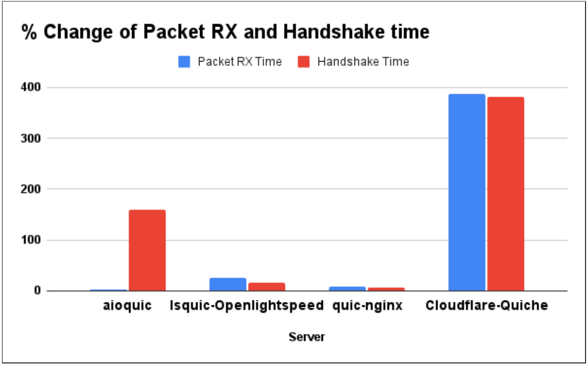
Server	aioquic		lsquic-Openlightspeed		quic-nginx		Cloudflare-Quiche	
	Before	After	Before	After	Before	After	Before	After
PACKET RX	91.67	98.19	9.64	15.21	85.74	75.98	155.66	155.35
HANDSHAKE	181.94	190.81	20.42	25.56	175.22	157.75	157.40	157.09



Work Done

Experiment 5: Buffer Overflow.

Server	aioquic		lsquic-Openlightspeed		quic-nginx		Cloudflare-Quiche	
	Before	After	Before	After	Before	After	Before	After
PACKET RX	91.67	88.68	9.64	12.18	85.74	78.96	155.66	757.20
HANDSHAKE	181.94	472.26	20.42	23.53	175.22	164.06	157.40	758.94



Experimental Results

HANDSHAKE Time:

Server name	Normal(Avg)	Exp1	Exp2	Exp3	Exp4	Exp5
aioquic	181.94	187.77	273.57	431.86	190.81	472.26
lsquic-Openlightspeed	20.42	22.43	23.87	31.67	25.56	23.53
Quic-nginx	175.22	157.99	162.01	167.91	157.75	164.06
Cloudflare-Quiche	157.40	156.11	165.95	767.24	157.09	758.94

Table: Handshake Time for servers on different tests

Server name	Exp1	Exp2	Exp3	Exp4	Exp5
aioquic	3.20	50.36	137.36	4.87	159.56
lsquic-Openlightspeed	9.82	16.92	55.23	25.19	15.22
Quic-nginx	9.83	7.54	4.17	9.97	6.37
Cloudflare-Quiche	0.82	5.43	387.44	0.20	382.17

Table: Percentage change in Handshake time

Experimental Results

PACKET RX Time:

Server name	Normal(Avg)	Exp1	Exp2	Exp3	Exp4	Exp5
aioquic	91.67	94.64	94.85	91.17	98.120	88.68
lsquic-Openlightspeed	9.64	11.24	22.07	9.34	15.21	12.18
Quic-nginx	85.74	76.11	78.87	79.96	75.98	78.96
Cloudflare-Quiche	155.66	153.62	164.19	755.77	155.35	757.20

Table: Packet RX Time for servers on different tests

Server name	Exp1	Exp2	Exp3	Exp4	Exp5
aioquic	3.23	3.46	0.55	7.12	3.26
lsquic-Openlightspeed	16.63	128.87	3.17	57.75	26.30
Quic-nginx	11.23	8.01	6.74	11.38	7.91
Cloudflare-Quiche	1.31	5.48	385.53	0.20	386.44

Table: Percentage change in Packet RX Time

Experimental Results

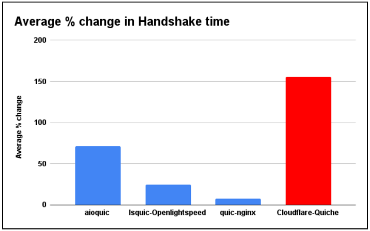


Figure: Average Handshake Time Graph

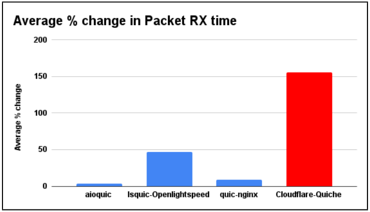


Figure: Average Packet RX Time Graph

Conclusion

- Our experiment of pentesting shows that the order for using these servers is:
  - On the basis of Handshake time: **nginx** > **openlightspeed** > **aioquic** > **cloudflare**
  - On the basis of Packet RX time: **aioquic** > **nginx** > **openlightspeed** > **cloudflare**

References I

[1] Xudong Cao, Shangru Zhao, and Yuqing Zhang. “0-RTT Attack and Defense of QUIC Protocol”. In: *2019 IEEE Globecom Workshops (GC Wkshps)*. Dec. 2019, pp. 1–6. DOI: 10.1109/GCWkshps45667.2019.9024637.

[2] Efstratios Chatzoglou et al. “Revisiting QUIC attacks: A comprehensive review on QUIC security and a hands-on study”. In: July 2022. DOI: 10.21203/rs.3.rs-1676730/v1.

[3] Sarah Cook et al. “QUIC: Better for what and for whom?” In: May 2017, pp. 1–6. DOI: 10.1109/ICC.2017.7997281.

[4] Adam Langley et al. “The QUIC Transport Protocol: Design and Internet-Scale Deployment”. In: Aug. 2017, pp. 183–196. DOI: 10.1145/3098822.3098842.

References II

[5] Robert Lychev et al. “How Secure and Quick is QUIC? Provable Security and Performance Analyses”. In: *2015 IEEE Symposium on Security and Privacy*. 2015, pp. 214–231. DOI: 10.1109/SP.2015.21.

[6] Robin Marx et al. “Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity”. In: Aug. 2020. DOI: 10.1145/3405796.3405828.

[7] Késsia Nepomuceno et al. “QUIC and TCP: A Performance Evaluation”. In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. June 2018, pp. 00045–00051. DOI: 10.1109/ISCC.2018.8538687.

Thank You!