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 15, 2022

Table of Contents

Internet Protocols

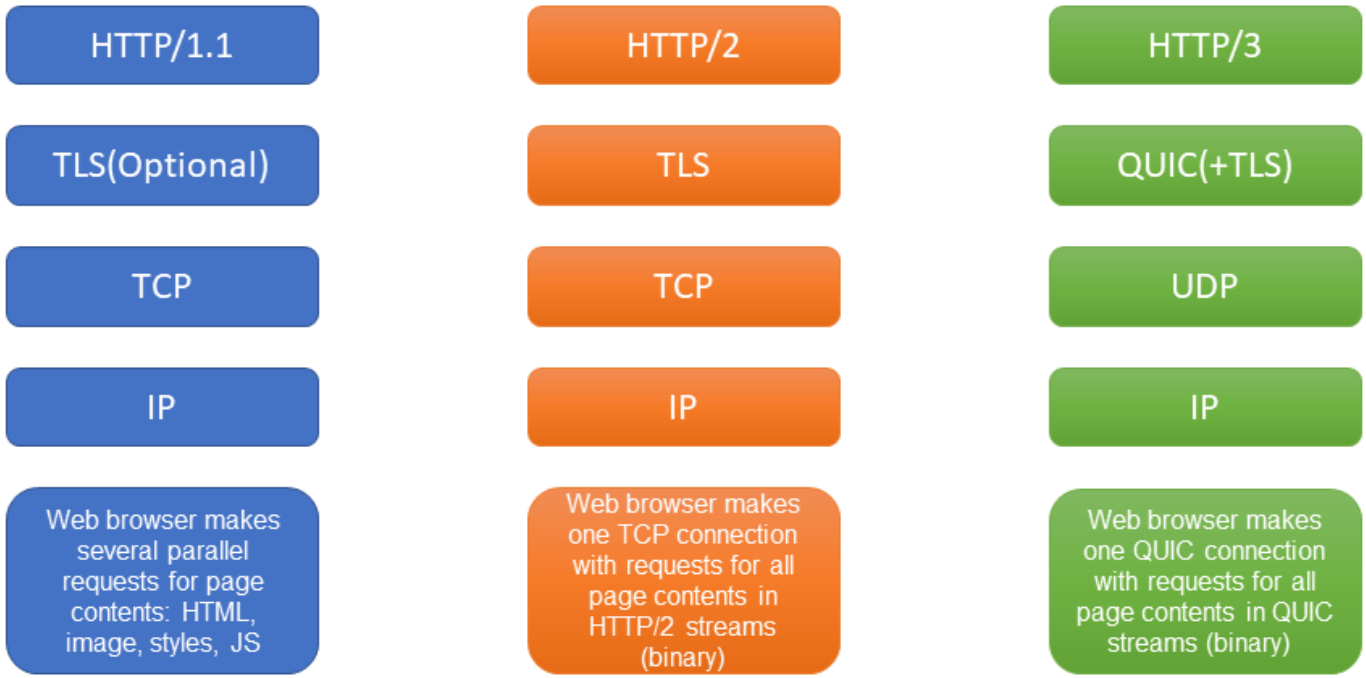

Figure: 1. Internet Versions

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

Literature Survey

Author et al	**Reference Number**	**Proposed Work**
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 [3]	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

Literature Survey

Author/Date	Topic/Purpose	Findings/Synopsis
Efstratios Chatzoglou1, Vasileios Kouliaridis1, Georgios Karopoulos2 Georgios Kambourakisa, 2015 [4]	Revisiting QUIC attacks: A comprehensive review on QUIC security and a hands-on study Quick is QUIC?	1. In presence of atta QUIC maybe unable t attain 0-RTT connect 2. Analysed the pitfal designing performanc driven secure protoco
Robert Lychev, Samuel Jero, Alexandra Boldyreva, 2015 [4]	How Secure and Quick is QUIC? Provable Security and Performance Analyses	1. In presence of atta QUIC maybe unable t attain 0-RTT connect 2. Analysed the pitfal designing performanc driven secure protoco
Sarah Cook, Bertrand Mathieu,	QUIC: Better For What	QUIC outperforms HTTP/2 over TCP/T

Literature Survey

Igor Nogueira de Oliveira, Rafael Roque Aschoff, 2018 [6]	QUIC and TCP: A Performance Evaluation	Influence of RTT in the experiment was noticeable while packet loss ratio influence was inexpressive.
Robin Marx, Joris Herbots Wim Lamotte, Peter Quax, 2020 [5]	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.

Motivation

- Lots of implementations present for HTTP/3. Deciding which implementation to use is hard.
- Adoption of HTTP/3 is increasing so deciding which HTTP/3 implementation is better for use maybe helpful.
- HTTP/2 adjusted to make compatible with QUIC : HTTP/3
- To analyze features of different HTTP/3 servers : faster connection set-up, less HoL blocking, connection migration.
- Learning about potential downside of QUIC like :
  - not allowed by many network
  - has higher encryption overhead
  - makes the web more centralized

Work Done

- Started by learning about the evolution of internet versions available.
  - HTTP/1.1, HTTP/2, HTTP/3
- Learned about the need of evolution of internet protocols over time.
  - **HTTP/1.1**: Web browser makes several parallel requests for page contents: HTML, image, styles, JS.
  - **HTTP/2**: Web browser makes one TCP connection with requests for all page contents in HTTP/2 streams (binary). Head of Line blocking issue exists in HTTP/2.
  - **HTTP/3**: Web browser makes one QUIC connection with requests for all page contents in QUIC streams (binary).
- Read about the Head of Line blocking problem in HTTP/2.
- Studied behaviour of different QUIC supporting servers with packet loss, delay, network change and compared QUIC with earlier versions.
- Pen-tested servers like Apache, Nginx, Openlitespeed, aioquic, Cloudflare.

Work Done

- Usage of **Scapy**: Scapy is a powerful interactive packet manipulation program. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. It can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery.
  We used this to manipulate the data of packets which were sent to the QUIC test servers.
- Usage of **Wireshark**: Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education.
  We used this to track and analyze the packets sent to the QUIC test servers.

Work Done

- Following kind of forgery was performed to QUIC packets which were sent to different QUIC test servers:

Experiment Number	Parameters Changed
1.	Reduce header length of UDP Layer
2.	Changed frag flag of IP layer
3.	Changed the flag of IP layer and dport in UDP layer
4.	Changed IP Version and header length in IP version
5.	Changed ihl variable in IP layer, Reduced header length in UDP layer and changed IP's Id

Work Done

- The steps taken to perform the experiment are as:
  - Started by taking a QUIC packet.
  - Depending on the experiment type we forged the data in packet as per the table above.
  - Tried to flood the test server by sending this QUIC packet.
  - Noted the handshake time and Packet RX time by the help of **http3check.net**

Work Done

- **PACKET RX** - This value represents the time between the first packet sent and the first packet received (measured in milliseconds).

- **HANDSHAKE TIME** - This value represents the time between when the first packet is sent and when the handshake is completed (measured in milliseconds).

Proposed Work

- To try and compare implementations of QUIC supporting servers like aioquic, nginx-quiche, openlitespeed, Cloudflare.
- To study the behaviour of various servers implementing QUIC 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

Penetration Testing of different HTTP/3 testing servers

- Wireshark for capturing and analyzing the packets sent across network.
- Scapy tool for creating and manipulating the packets.
- Python to write script to change packet data using Scapy library.
- http3check.net powered by LiteSpeed to measure the page latency and the handshake time.
- Test servers supporting HTTP/3 namely aioquic, Cloudflare, Openlightspeed, nginx.

Experimental Results

Server name	Normal	No change	Test1	Test2	Test3	Test4	Test5
aioquic	89.9	92.701	92.794	91.548	99.401	95.574	105.384
lsquic-Openlight speed	8.037	10.202	7.776	9.931	9.141	7.934	9.697
Quic-nginx	78.673	81.089	84.344	88.836	81.459	78.548	87.652
Cloudflare -Quiche	3.283	3.205	4.758	4.27	2.557	3.488	2.533

Figure: Handshake Time for servers on different tests

Server name	No change	Test1	Test2	Test3	Test4	Test5
aioquic	3.115684093	3.219132369	1.833147942	10.56840934	6.311457175	17.22358176
lsquic-Openlight speed	26.93791216	3.247480403	23.56600722	13.73646883	1.281572726	20.65447306
Quic-nginx	3.070939204	7.208317974	12.91802779	3.541240324	0.158885514	11.4130642
Cloudflare						

16/21

Experimental Results

Server name	Normal	No change	Test1	Test2	Test3	Test4	Test5
aioquic	177.846	184.673	185.077	182.25	190.506	187.027	193.529
lsquic-Openlight speed	9.118	19.862	8.894	20.163	18.627	9.046	20.023
Quic-nginx	159.575	164.084	171.081	180.58	165.136	159.584	179.861
Cloudflare-Quiche	6.54	8.061	7.431	10.108	6.499	21.65	5.847

Figure: Packet RX Time for servers on different tests

Server name	No change	Test1	Test2	Test3	Test4	Test5
aioquic	3.838714393	4.06587722	2.47629972	7.118518269	5.162331455	8.818303476
lsquic-Openlight speed	117.8328581	2.456679096	121.1340206	104.2882211	0.7896468524	119.5985962
Quic-nginx	2.825630581	7.210402632	13.16308946	3.484881717	0.0056399812	12.71251762
Cloudflare						

Experimental Results

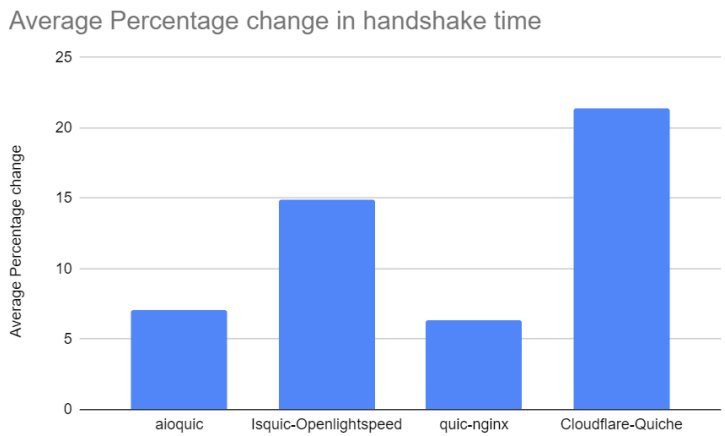

Figure: Average Handshake Time Graph

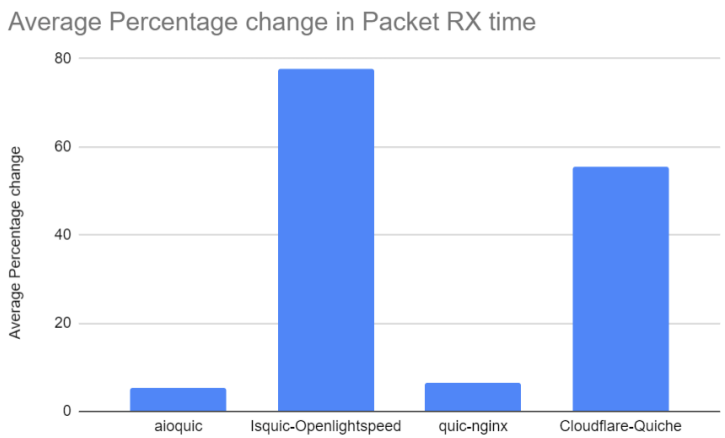

Figure: Average Packet RX Time Graph

Conclusion

- HTTP/3 is basically HTTP/2 over QUIC. This means that it is basically a improvement in terms of performance, security and not a reform.
- The main issue with QUIC lies in the initial phase of handshake because the client hello packet is unencrypted.
- Lot of implementations are present for QUIC. Our experiment of pentesting by altering the packet parameters shows that the order for using these servers is:
  - On the basis of Handshake time: **nginx aioquic openlightspeed cloudflare**
  - On the basis of Packet RX time: **aioquic nginx cloudflare openlightspeed**

References

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

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

[4] 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.

[5] 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.

Thank You!