Evaluating QUIC for Privacy Improvements Over Its Predecessors

CSE 227 Research Project

Allison Turner
Rukshani Athapathu
Chitra Kharbanda
aturner@ucsd.edu
dathapathu@ucsd.edu
ckharbanda@ucsd.edu

ABSTRACT

QUIC is a new transport protocol that is slowly gaining popularity in the Internet ecosystem. For example QUIC carries one third of the Google traffic and at Facebook 75% of its traffic is responsible for QUIC. Considering QUIC's influence, in this paper we evaluate QUIC's privacy properties using two main streams. First, we analyse third party traffic in the wild using Chrome and Firefox and then we evaluate DNS over QUIC against other encrypted DNS protocols. Analyzing third party traffic reveals that address validation token pose a serious treat against privacy of users specially with third party tracking. We are also able to fingerprint Chrome and Firefox browsers using the values of transport parameters.

ACM Reference Format:
Allison Turner, Rukshani Athapathu, and Chitra Kharbanda. 2022. Evaluating QUIC for Privacy Improvements Over Its Predecessors: CSE 227 Research Project. In *Proceedings of UCSD CSE 227 SP22.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

QUIC is new transport protocol designed by Google and now standardized by the Internet Engineering Task Force (IETF) [15]. Hypertext Transfer Protocol (HTTP) Version 3 is built on top of QUIC and used by many sites as of now. For example it is now known that one third of the Google traffic is carried out by QUIC [1]. However, only 8% of websites still uses QUIC in overall [2].

Due to the gaining popularity of QUIC among the community as well as the numerous implementations that are out there, it is imperative that a thorough analysis of its weaknesses should be investigated in timely manner. We divide our work into two separate sections. 1) We evaluate the QUIC behavior on third-party websites in the wild. 2) We compare encrypted DNS protocols with a specific focus on DNS-over-QUIC which is called DOQ.

1.1 QUIC Design

QUIC is designed with the idea to provide applications with low latency connection establishment and flow controlled streams for the communication. QUIC is integrated with TLS 1.3 to provide confidentiality and integrity of data. Figure shows how QUIC is integrated into the protocol stack compared to TLS over TCP. As you can see, that QUIC is a user space protocol making use of UDP protocol in the kernel networking stack.

1.1.1 Connection Setup. QUIC supports both 0-RTT and 1-RTT connection setups. First, client sends an 'Initial' packet containing a CRYPTO frame carrying TLS 1.3 client hello and transport parameters. Then the server replies with an initial packet carrying server hello and further packets with necessary handshake frames. Once the TLS handshake is finished client and server starts the data transmission with encrypted packets. 0-RTT works only if a previous connection has been established with a host and the secret values are cached. Figure shows a simplified handshake of 1-RTT and 0-RTT as well as how QUIC connection setup differs from TLS over TCP.

1.1.2 Connection ID Selection. Connection identifiers are used to identify a connection. Connection IDs can be selected by endpoints independently and each endpoint selects the connection ID for its peer. A zero length connection IDs are permitted when a connection ID is not necessary to route to the correct endpoint. Connection IDs are valid for the duration of the connection or until its peer invalidates the connection ID.

1.1.3 Address Validation Token. Server can provide clients with an address validation token during the connection setup and clients can use this token in a future connection by including it in the initial packet so that the address validation can take place. However, RFC 9000 specification mentions that the attackers could replay tokens as amplifiers in DDoS

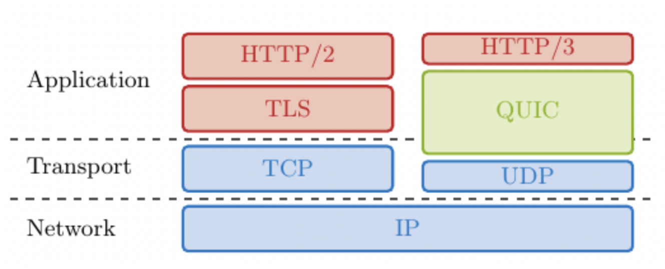

Figure 1: QUIC stack in comparison to TCP [3]

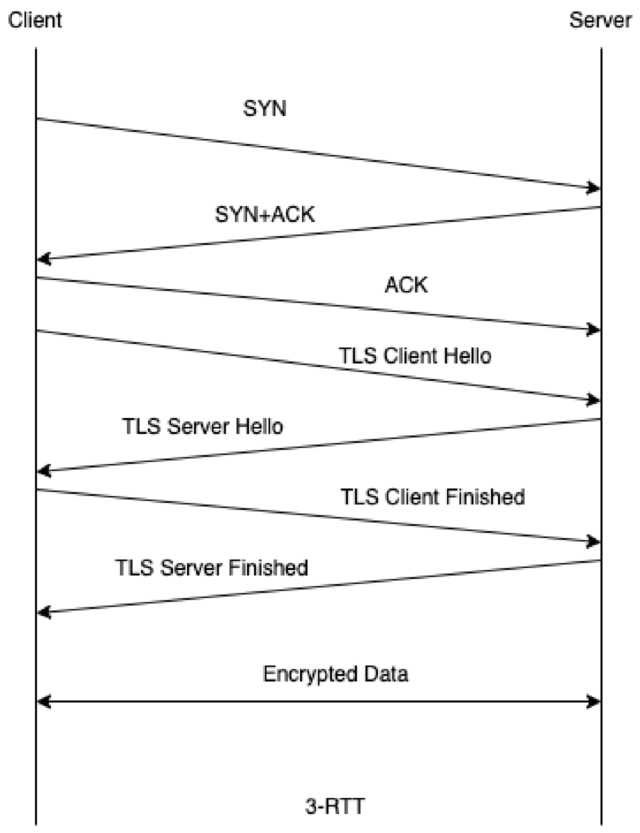

Figure 2: TLS over TCP

attacks and these tokens should only be accepted for a short period of time.

1.1.4 Transport Parameters. During the connection setup each endpoint can declare their transport parameters so that the other endpoint has to comply with those values when the data transmission starts. These parameters are sent in a TLS extension during the handshake. In our study we use these parameter values to see whether those can be used for browser fingerprinting.

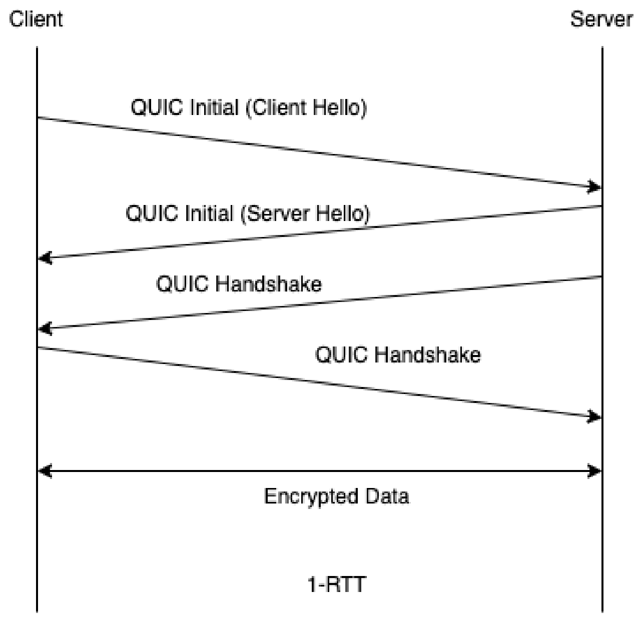

Figure 3: QUIC 1-RTT Connection

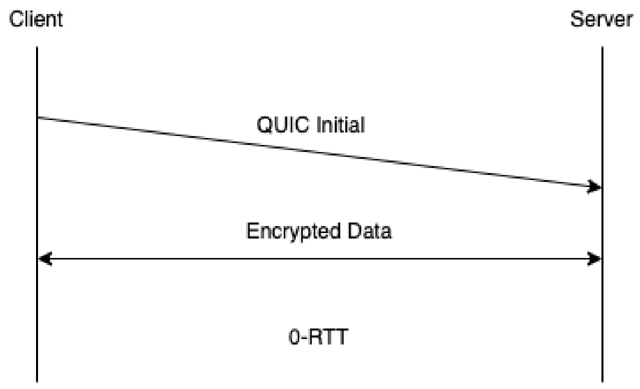

Figure 4: QUIC 0-RTT Connection

1.2 DNS-over-QUIC

The Domain Name Service (DNS) is one of the most important parts of making internetworking possible. To find the global IP address of any public host, a user need only query their nearest recursive resolver, and will have several answers in less than seconds. However, DNS also represents many complex security issues; we will consider only the concerns of the individual user. Such a user would likely be most concerned about the privacy of their DNS traffic and whether it will be subject to censorship filters.

Traditional DNS queries and responses are sent in plain text over UDP. While this makes name resolution services relatively simple to access, it leaves individual users vulnerable. An observer could easily read any DNS queries to learn about what sites the user is visiting, or even tamper with the responses to prevent that user from accessing those sites. The reality of these violations is a well-documented and persistent problem.

One approach to this problem is encryption between the user and the DNS resolver, or between the user and a proxy server. Proposals for encrypted DNS protocols include DNS-over-TLS (DOT), DNS-over-HTTPS (DOH), and, of course, DNS-over-QUIC (DOQ). QUIC is an attractive option for secure DNS transport, as its authors built in mechanisms to, ideally, make it robust against tampering, spoofing, and monitoring.

2 PROBLEM STATEMENT

2.1 In Common Web Applications

The authors of [15] state that QUIC's general security goals are "confidentiality, integrity, and availability in a range of deployment circumstances". Confidentiality and integrity are the goals most pertinent to our analysis. We'll first investigate which parts of QUIC are meant to achieve these goals, and then extrapolate the specific privacy metrics on which QUIC's authors aimed to out-perform comparable protocol stacks.

QUIC is supposed to be difficult for intermediators to tamper with the traffic since it eliminates the attack surface provided by TCP. But one could also argue that it makes it harder to analyze network traffic making it difficult to identify malicious activity. In this study we investigate whether it is possible to track users by passively analysing third party QUIC traffic in the wild and then whether we can use the information in the initial packet for browser fingerprinting.

2.2 As Compared to Other Solutions in an Internet Security Problem Space

In addition to evaluating QUIC as a standalone secure transport mechanism, we consider a second perspective: QUIC as a tool that may help solve an internet security problem. The problem space we chose is per-user DNS privacy and integrity.

[12] outlines how Mozilla has already made moves to utilize DOH via Cloudflare in the Firefox web browser, and how these moves raise concern over, for example, the centralization of DNS. As DOQ is the newest in this growing family of protocols, encrypted DNS is increasingly embraced, and DOT, DOH, and DOQ have a lot in common with each other, it seems important to evaluate their common design invariants, unique strengths, and net efficacy. In doing so, we have another case study on QUIC to consider as the successor to HTTPS.

3 RELATED WORK

3.1 QUIC Security Evaluations

QUIC went through 34 revisions before IETF standardized it in 2021 [15]. There are many previous works which analyzes the security aspect of QUIC [8], [19]. However, these take into consideration, the old versions which differs vastly in terms of fields, etc from the current standardized version.

[20] analyzes the possibility of tracking clients across several connections. It analyzes user tracking and third-party tracking over QUIC. However, it takes into consideration, using field like 'server-config', which has been deprecated in the RFC version of QUIC.

TLS fingerprinting [23] has been used for client fingerprinting by creating signature, such as JA3 [17]. TLS version, list of accepted Ciphers, List of extension, Elliptic Curve and Elliptic Curve Format are the fields used for constructing the JA3 signature. Other techniques have been used for fingerprinting client using QUIC traffic. Using fields in the initial packet of the QUIC, such as user-agent information, creation of signature such as CYU [22] has been tried. However, JA3 is not a standardized field of QUIC and with standardization of QUIC some of the fields such as user-agent have been deprecated.

3.2 DNS Security and Mechanisms for Protection

[18] is the first study to measure DNS-over-QUIC at Internet-wide scale. Kosek et al. scanned for DOQ servers to evaluate rates of adoption and assembled statistics comparing DOQ performance to DOT and DOH. They found 1,217 "DoQ-verified" resolvers, 45% of which are in Asia and 32% of which are in Europe, with many operated by AdGuard and nextDNS. Interestingly, none of the DOQ resolvers they found enabled 0-RTT functionality - Kosek et al. hypothesize that this decision was made to alleviate privacy risks. [6] performed a similar study, focused on DOH, while [11] found DOT, DOH, and traditional DNS to have reliability tradeoffs in a variety of network conditions.

[4], [5], [7], [13] and [21] all show that the 3 encrypted DNS mechanisms we consider cannot protect users from simple, off-the-shelf machine learning analyses to identify the contents of an encrypted DNS query.

4 METHODOLOGY

4.1 Evaluating QUIC Behavior on Websites in the Wild

We have randomly selected twenty websites including both popular as well as not so popular sites to observe the behavior of QUIC traffic. We have initially selected five browsers for our study namely, Chrome, Firefox, Safari, Brave and

Opera. Unfortunately Safari does not support QUIC by default and the website we have visited in Safari ended up using TLS over TCP. With Brave and Opera browsers we see an unusual behavior where even the initial packet information is encrypted making it difficult to uncover values of the parameters we were interested in. Since Opera and Brave both use the chromium source we wonder somehow there is a connection between the chrome browser and Opera and Brave since only if we use the sites we visited in Chrome in Opera we see this unusual behaviour. However, with brave even the new sites have encrypted client hello packet in the initial packet. If we choose a website that we have not visited using Chrome then we were able to see the parameters in clear text in the initial packet in Opera. Therefore, we have restricted some part of our study just to Chrome and Firefox considering the time limit we have.

The experiments were conducted as follows. We intercepted the traffic between the browser and the website and got the pcaps with the help of Wireshark. A single website was tested three times. First capture is when we visited the site for the first time, second capture was done without closing the browser and the final capture is done by closing and reopening the browser and visiting the website again. Wireshark was running on the same computer as the tested browser. To analyse the pcaps, we then wrote a simple script based on pyshark. For the fingerprinting we use MD5 hash generator. (Appendices: B)

4.2 Comparing Encrypted DNS Protocols

To compare DNS-Over-QUIC to other encrypted DNS schemes, we reviewed existing work on the privacy and security properties, and built a small network artifact capture pipeline to confirm and build on those findings. The pipeline is structured as follows:

- A unique test is made up of a combination of target domain, DNS record type, and encrypted DNS method
  - an 'A' record query for google.com using DNS-Over-HTTPS constitutes a separate test from an 'A' record query for google.com using DNS-Over-QUIC
  - a 'AAAA' record query for bing.com using DNS-Over-TLS constitues a separate test from a 'A' record query for bing.com using DNS-Over-TLS
  - an 'A' record query for cloudflare.com using DNS-Over-HTTPS constitutes a separate test from an 'A' record query for developers.cloudflare.com using DNS-Over-HTTPS
- We run 2 batteries of tests:
  - All tests are run from within their own new Docker containers
  - All unique combinations of target domain and DNS record type are queried from within the same Docker container

  The purpose of these 2 suites is to look for recurring identifiers across the second set of queries that could be used for associating multiple queries to a single user. Unfortunately, these tests will not fully simulate how web browser peripherals, such as unique user-agent strings or managed caches, may enhance the risk of partial or full de-anonymization.
- no other containers are running. This allows us to isolate the encrypted DNS packet capture activity from other network activity on the testing machine, by recording on the 'docker0' interface with dumpcap. This also simulates the perspective of a generalized passive observer between client and encrypted DNS resolver.
- Target domains include a variety of TLDs, such as .gov, .edu, .com, .org, .mil, and .fr. We run queries on A and AAAA records for all target domains
- For each unique test or test set, we start a new Docker container, start the packet capture file, run the query, save the packet capture, STDOUT, and STDERR for analysis, and finally shut down and remove the container.
- We use a combination of Cloudflare's public DNS resolver services via curl and Adguard's dnslookup tool as the encrypted DNS implementations for examination. DOQ testing tools are rare at this point in time, given how recent standardization was, and so Adguard's libraries are one of the few third party options.

When the generated artifacts show unexpected errors, we re-test the command in a different environment.

We evaluate each mechanism's behavior on several axes:

- Obfuscation
  - Does this mechanism obfuscate the user's DNS activity? To what degree, in comparison to traditional DNS? Other encrypted DNS mechanisms?
  - Does this mechanisms obfuscate the user's identity, or make the task of assembling a user profile more difficult? To what degree, in comparison to traditional DNS? Other encrypted DNS mechanisms?
- Integrity
  - Does this mechanism make tampering with DNS queries and/or replies more difficult? To what degree, in comparison to traditional DNS? Other encrypted DNS mechanisms?

After evaluating each encrypted DNS mechanism on these axes, we will return to a general evaluation of QUIC's utility in this problem space.

5 EVALUATION

5.1 In Common Web Applications

1) Can a user be identified across several connections?

Each QUIC connection starts with a handshake. However, if you have made a connection with the server before, QUIC protocol allows sending data with 0-RTT functionality. Each connection has a connection identifier and these IDs are selected by the peer. The selection of this connection ID is implementation specific.

Analysing the connection IDs in wireshark logs for the websites we have selected showed us that this connection ID is always visible in the protected QUIC packets. Because of this it seems that a passive observer can easily track the packets that are destined for a certain connection.

For the 'semrush' website if we reopen the website after closing the tab, we see that the connection ID selected by the source for the destination is different each time. 'uber.com', 'm.indiamart.com' and few other sites we have tried all follow the same behaviour. However with 'cloudflare', 'facebook', 'getexperience.acs.org', 'overthecap.com' and 'cirebonats.com' we see that the server also selects a connection ID for the client. The connection ID selected by the server is being used in the subsequent connections. That seems to be the only difference between 'cloudflare', 'facebook', 'getexperience.acs.org', 'overthecap.com', 'cirebonats.com' and other websites.

For the 'google.ru' website even though we have not browsed that website ever before, it does a 0-RTT packet. However, for the connections even with 0-RTT, client chooses a different destination connection ID each time making it difficult for a passive observer to track a user across connections.

For the 'gumtree.com.au' website, 0-RTT seems to be not implemented as while trying multiple connections without closing the browser, the full handshake process occurs before any payload is exchanged.

So the conclusion is that it is difficult to track a user across several connections.

2) Is this common to Firefox?

We carried out the experiments with the same set of websites on Firefox and compared it with the pcap data for the Chrome browser to observe the behaviour in different browsers. Among the differences we found, a major one was that all of the sites in Firefox, both the source and the destination IDs are being selected by the client. In chrome, the source connection ID selected by the server was always 0 for the websites that we have analyzed. Another general trend noticed in the Firefox pcap was that more information is present in its initial packets and the fields are consistent with almost all the websites tried. In chrome, the behavior observed varied with different websites. Some of the websites had multiple 'Crypto' frames in their initial packet, and in general, only one of those frames had the hello protocol information present. For the website, 'gumtree.com.au' initial packet for chrome had malformed tls indicated showing the dubious nature of the chrome implementation of quic client. Apart from this, all of the Firefox pcap had the information stored in cleartext. In contrast, chrome's behavior was not consistent for some of the websites, and the initial packets implemented the 'encrypted handshake' protocol which hid all of the information. The second connection, where we connected to the website after just closing the tab, for websites like 'facebook.com', 'cloudflare' and 'semrush.com' did not capture any quic traffic. Although, some 'udp' traffic was visible in the Wireshark. Another behavior in the chrome pcap file was that the second connection had encrypted data in the initial packet for most of the websites analyzed.

3) What are the security implications of zero round-trip time (0- RTT) handshake?

Based on our observation across multiple connections with the same server, below is the behaviour which was observed related to 0-RTT session resumption. When a client makes a connection with the server, the server sends a token on the first connection which the client caches and uses for the subsequent connection (0-RTT). However, we were not able to capture the token being sent from the server to the client but in 0-RTT connections, we were able to see the client using the token. Another thing observed was that the token being used from the client in the 0-RTT connection was different every time. This behaviour can only be seen when the websites are visited using Chrome. The Firefox browser does not present these tokens at all with 0-RTT connection making us believe that either Firefox does not cache this token or that the address validation token feature is not implemented.

The main purpose of the address validation is to ensure that the connection request originates at the IP address claimed. Since Chrome always present this token to the server in the initial packet in clear text we wanted to figure out whether an attacker can sniff the network traffic and collect this token and spoof the connection request to make a new connection with the server. The scapy software we tried out did not support QUIC and to get it to work to perform a man in the middle attack requires us to write a custom protocol in scapy which would require more time and effort. Considering the limited time we have for this project we had to let go of that ambition. But we do feel that this token replay attack can be a legitimate concern for security of the server.

Address validation token is also vulnerable to third party tracking. Consider the following scenario. Let's say website

A has third party called ***Z*** and the same third party ***Z*** is also present in another website ***B***. Since it is the server that decides the address validation token and its validity period, third party ***Z*** can reuse the token when the same user visits the website ***B***. That way the third party can track the websites that this particular user visits and create a profile for that user. Also, the third party does not necessarily have to reuse the token, it can simply keep a record of tokens it issues to a particular user so that it can build a profile of the said user.

4) Can we use QUIC info in initial packets for browser fingerprinting?

Fingerprinting involves collecting information about browsers, operating systems, network etc... and using these characteristics to uniquely identify a particular object be it operating systems, unique browsers or users. Our goal in this section is to analyze the third party pcaps that we have gathered to see what information in the initial packet can be used for browser fingerprinting.

Initial packet that the client sends to the server has transport parameters in the client hello section. According to the RFC 9000, each endpoint can choose values for transport parameters independent of the values chosen by its peer. Since each browser can choose to use any values for these parameters, we first try to see whether we can identify the clients based on these values. We choose five browsers, Chrome, Firefox, Safari, Brave and Opera. Safari by default does not support QUIC and the sites we have checked ended up using TCP. With the Brave and Opera browsers we ran into a unique issue where all the transport parameters including the other info in client hello packet was encrypted by default even when we were visiting those sites for the first time with a fresh installation of these browsers. We could not figure out how it manages to encrypt information in the initial packet even before establishing the TLS connection. With Opera browser this behaviour is only visible for the sites we have visited using Chrome. But with Brave all the information in client hello is encrypted even if we have not visited those websites in Chrome. It worked as if somehow a previous connection has been established just like the behaviour we see in chrome when we visit a particular site multiple times. Considering these facts our observations are limited to only Chrome, Firefox and Opera.

Table 1 shows the transport parameters chosen by Chrome, Firefox and Opera browsers.

From the Table 1 you can see that Chrome and Opera share the same values while Firefox is different. These values are consistent across website within a single browser. Therefore, if we create a hash out of these values, chrome and opera always result in '86cc808155f0f6cf4a5694246e7d5832' while Firefox result in '2c35b99f97e416f627765886238c560f' value.

Transport Parameter	Chrome/Opera	Firefox
initial_max_stream_data_uni	6291456	1048576
max_datagram_frame_size	65536	0
initial_max_streams_uni	103	16
initial_max_stream_data_bidi_local	6291456	12582912
initial_max_data	15728640	25165824
initial_max_stream_data_bidi_remote	6291456	1048576
max_idle_timeout	30000	30000
initial_max_streams_bidi	100	16
active_connection_id_limit	MISSING	8

Table 1: Transport parameter values chosen by Chrome, Opera and Firefox

A passive observer can inspect these unprotected values in the initial packet and create a hash and match with these values to identify which browser the client is communicating with. Even though you cannot distinguish between Opera and Chrome since Opera uses the same chromium source as chrome we do feel that if a vulnerability present in the chromium source both of these browsers will be affected and a passive observer can use this information for possible future attacks.

Next we wanted to see whether we can uniquely identify the browser version using some of the cert parameters in QUIC initial client hello packet. We limit this part of our study to just Firefox since it always gives us consistent result. We experimented with different fields of the initial client hello packet of 90.0, 92.0, 94.0, 97.0, 98.0, 99.0, 100.0 and 101.0 versions of Firefox. The observed JA3 for all of these versions was same with its hash value being consistent as 'b719940c5ab9a3373cb4475d8143ff88' for the first connection and '7faeb63993918104466311 4099ee6e23' for the second connection without exiting the browser. Apart from this, almost all the fields across all versions had the same value making it difficult to track the browser version. Although, for versions 90 and 92 had 'tls.quic.parameter.initial_max_stream_data_bidi_local', and 'tls.quic.parameter.initial_max_data' different from the rest of the newer versions of the Firefox. The hash value of the 'transport_parameters' (Appendices: A Pt. 2) for Firefox version 90.0 and 92.0 was observed to be '339e4551d2466db4bed4ba54297b5960' and for rest of the versions was observed to be '1f0fe5a8b7d6e4418ca7a2fbd38a723b '.

We also tried to see whether we could uniquely identify the same browser from different machines using the cert parameters in QUIC initial client hello packet. We tried analysing pcap data from two different machines running mac OS v12.0.1 and a Ubuntu machine running on a EC2 VM. For all of the combinations of fields analyzed in the initial client hello packet, we found out that the values and the fields were identical across the machines and hence tracking different users is not possible.

5.2 DNS-Over-QUIC Compared to Other Encrypted DNS Mechanisms

All 3 encrypted DNS (EDNS for brevity) mechanisms have the following in common:

- IP
- TLS
- DNS

Thus, any attacks made viable by the structure of these protocols will affect EDNS. For example, in [9], Hoang et al. show that the benefits toward per-user obfuscation conferred by EDNS and improvements upon TLS Client Hello are undermined by the unique set of server IPs from which contemporary websites source different types of content. Hoang et al. show this notion to be a viable website fingerprinting approach.

A plethora of other such flaws can be found in TLS. Our packet captures showed that all 3 EDNS mechanisms' Client Hellos contained Server Name Indicators (SNI) in cleartext, broadcasting the intent to communicate with a public EDNS service; or, once the DNS query has been answered, in TLS connections with services that are even more sensitive for the user. While Trevisan et al. show in [21] that using the encrypted SNI (eSNI) extension doesn't make much difference against machine-learning-based traffic analysis methods for revealing domains in EDNS, [10] finds that using eSNI can help users access sites censored by DNS-filtering, to the tune of 55% in China and 95% in other censoring countries. Separately, there exists the threat of man-in-the-middle attacks on TLS, and the concerns regarding the Certificate Authority ecosystem when regarding the server certificate private key solution to TLS MITM.

Closer to the core of TLS' efficacy, [4], [5], [7], and [21] all show that relatively simple machine learning techniques can reveal the targets of EDNS queries, even with padding included. Hu and Fukuda [13] present similar preliminary findings for DNS-over-QUIC. Bushart and Rossow further show that DNS queries, whether encrypted or not, are subject to a phenomenon in DNS reminiscient of [9]'s findings on IP-level fingerprinting [4].

To account for the aforementioned complexities, we focus on the strengths and vulnerabilities offered to EDNS by each transport mechanism.

5.2.1 *DNS-over-TLS and DNS-over-HTTPS.* ***Obfuscation***

As observed in our packet captures, DOT uses port 853, the port publicly assigned by IANA to DNS-over-TLS, DNS-over-DTLS, and DNS-over-QUIC. DOT's Client Hello also indicates "dns" in TLS' Application Layer Protocol Negotiation field. These two features make DOT connections easy to identify and isolate, falling behind DOH in obfuscating application-layer purpose

DOH uses port 443. DOH also indicates "http/1.1" in TLS' Application Layer Protocol Negotiation field, in contrast with DOT and DOQ, which indicate "dns" and "doq-i##" respectively. These choices result in DOH connections blending effectively with other HTTPS applications.

DOT and DOH are, besides the above-noted differences, functionally very similar. They both perform a TCP handshake, then a TLS handshake, and finally transmit the DNS operation in question in TLS application data. In DOH, the query is wrapped in an extra layer of HTTP within that application data payload. On the one hand, this makes isolating the query slightly harder; on the other hand, fingerprinting based on HTTP header fields, values, and orderings is a well-known technique. Even so, as [16], [14], and [23] show, in addition to [4], [5], [7], and [21] - contemporary encryption does not perform well on the axis of obfuscation.

Integrity

The presence of encryption significantly increases the barrier to tampering with DNS queries. Minor concerns exist regarding the ability of middleboxes to alter unencrypted header fields, but improvement to per-user integrity is only as good as TLS, for the most part.

5.2.2 *DNS-over-QUIC.* ***Obfuscation***

DOQ uses port 853, the port publicly assigned by IANA to DNS-over-TLS, DNS-over-DTLS, and DNS-over-QUIC. DOT's Client Hello also indicates "doq-i##" in TLS' Application Layer Protocol Negotiation field. These two features make DOQ connections easy to identify and isolate, falling behind DOH in obfuscating application-layer purpose.

As discussed in our analysis of regular QUIC behavior from third-party sites and applications, QUIC's source connection ID and address validation token features pose major fingerprinting risks. This risk remains in DOQ, with DOQ queries in the persistent environment using SCIDs and tokens across connections. Given the brevity of interactions between devices and DNS resolvers, we question the value of the SCID when weighed against the described risk.

When executing DOQ queries in a persistent environment, we noticed the incrementation of the connection number with each new query. This creates an easy mapping between QUIC connections and queries when analyzed by a passive observer, another variable that could be used to identify query contents. It is thus our recommendation that any DOQ query should package several domains into a packet.

Integrity

After the TLS handshake, QUIC encrypts not only the payload but many of the header fields as well, resulting in a packet that would be difficult to edit without alerting the recipient.

5.2.3 QUIC's Value in the Encrypted DNS Space. Much of the value that QUIC brings to the EDNS space is in DOQ's performance improvements on DOT and DOH. QUIC's handshake process is much faster, and is attractive in an application to DNS, where devices have a multitude of short exchanges with resolvers. QUIC's connection migration features are mostly irrelevant to EDNS, and thus features related to this function simply represent unnecessary attack surface.

6 CONCLUSION

Our study shows that the privacy concerns, several other studies have found with regards to QUIC has mostly been fixed with the standardized version of QUIC. For example it previously was shown that the server config identifier can be used to recognize a user across multiple sessions. However, with our limited study from both Chrome and Firefox showed us that this connection ID changes everytime a user goes to a webpage even with 0-RTT sessions making it difficult to track a user from different sessions. Also since this connection ID is determined by the peer, a malicious third party tracker would find it very difficult to track users across sites. However, address validation token is still vulnerable to both token replay attacks and web tracking. Our analysis of the field values present in the initial packet of the quic connection depicts that the transport parameter fields are different for different browsers and also for older versus newer versions of the browser. Therefore, the fields present in the transport parameter can be used for browser fingerprinting.

Additionally, we identify points of improvement in DNS-over-QUIC. DOQ clients should implement eSNI and use a broader ALPN value. Port 853 is standardized in RFC 9250, however, allowing the client to migrate queries to a "regular QUIC traffic" port may better blend DNS activity with other traffic, protecting both in doing so. QUIC clients should avoid a 1:1 relationship between query target domain and connection number, to further obfuscate query contents. Finally, features for connection migration, and possibly 0-RTT connections, are arguably unnecessary for DNS, and may only serve to increase attack surface.

REFERENCES

[1] [n. d.]. deploying HTTP/3 and IETF QUIC. ([n. d.]). https://blog.chromium.org/2020/10/chrome-is-deploying-http3-and-ietf-quic.html

[2] [n. d.]. Usage Statistics of QUIC for Websites, May 2022. ([n. d.]). https://w3techs.com/technologies/details/ce-quic

[3] 2022. 2.6 million addresses support QUIC. (Jan. 2022). https://blog.apnic.net/2022/01/27/2-6-million-addresses-support-quic/

[4] Jonas Bushart and Christian Rossow. 2020. Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association. https://www.usenix.org/conference/foci20/presentation/bushart

[5] Zihan Chen, Guang Cheng, Ziheng Xu, Shuyi Guo, Yuyang Zhou, and Yuyu Zhao. 2021. Length matters: Scalable fast encrypted internet traffic service classification based on multiple protocol data unit length sequence with composite deep learning. *Digital Communications and Networks* (2021). https://doi.org/10.1016/j.dcan.2021.09.009

[6] Rishabh Chhabra, Paul Murley, Deepak Kumar, Michael Bailey, and Gang Wang. 2021. Measuring DNS-over-HTTPS performance around the world. In *Proceedings of the 21st ACM Internet Measurement Conference*. ACM, Virtual Event, 351–365. https://doi.org/10.1145/3487552.3487849

[7] Levente Csikor, Himanshu Singh, Min Suk Kang, and Dinil Mon Divakaran. 2021. Privacy of DNS-over-HTTPS: Requiem for a Dream?. In *2021 IEEE European Symposium on Security and Privacy (EuroS P)*. 252–271. https://doi.org/10.1109/EuroSP51992.2021.00026

[8] Marc Fischlin and Felix Günther. 2014. Multi-Stage Key Exchange and the Case of Google's QUIC Protocol. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1193–1204. https://doi.org/10.1145/2660267.2660308

[9] Nguyen Phong Hoang, Arian Akhavan Niaki, Phillipa Gill, and Michalis Polychronakis. 2021. Domain name encryption is not enough: privacy leakage via IP-based website fingerprinting. *Proceedings on Privacy Enhancing Technologies* 2021, 4 (2021), 420–440. https://doi.org/doi:10.2478/popets-2021-0078

[10] Nguyen Phong Hoang, Michalis Polychronakis, and Phillipa Gill. 2022. Measuring the Accessibility of Domain Name Encryption and Its Impact on Internet Filtering. In *Passive and Active Measurement*, Oliver Hohlfeld, Giovane Moura, and Cristel Pelsser (Eds.). Springer International Publishing, Cham, 518–536.

[11] Austin Hounsel, Kevin Borgolte, Paul Schmitt, Jordan Holland, and Nick Feamster. 2019. Analyzing the Costs (and Benefits) of DNS, DoT, and DoH for the Modern Web. In *Proceedings of the Applied Networking Research Workshop (ANRW '19)*. Association for Computing Machinery, New York, NY, USA, 20–22. https://doi.org/10.1145/3340301.3341129 event-place: Montreal, Quebec, Canada.

[12] Austin Hounsel, Paul Schmitt, Kevin Borgolte, and Nick Feamster. 2021. Designing for Tussle in Encrypted DNS. In *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks (HotNets '21)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3484266.3487383

[13] Guannan Hu and Kensuke Fukuda. 2021. An Analysis of Privacy Leakage in DoQ Traffic. In *Proceedings of the CoNEXT Student Workshop*. Association for Computing Machinery, New York, NY, USA, 7–8. https://doi.org/10.1145/3488658.3493782

[14] Martin Husák, Milan Cermák, Tomá Jirsík, and Pavel Celeda. 2015. Network-Based HTTPS Client Identification Using SSL/TLS Fingerprinting. In *2015 10th International Conference on Availability, Reliability and Security*. 389–396. https://doi.org/10.1109/ARES.2015.35

[15] Jana Iyengar and Martin Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport*. Request for Comments 9000. Internet Engineering Task Force (IETF). https://doi.org/10.17487/RFC9000

[16] Yu Jiang, Yuying Li, and Yanbin Sun. 2021. Networked Device Identification: A Survey. In *2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*. 543–548. https://doi.org/10.1109/DSC53577.2021.00086

[17] Josh Atkins John Althouse, Jeff Atkinson. 2018. JA3 - A method for profiling SSL/TLS Clients. (July 2018). https://github.com/salesforce/ja3

[18] Mike Kosek, Trinh Viet Doan, Malte Granderath, and Vaibhav Bajpai. 2022. One to Rule Them All? A First Look at DNS over QUIC. In *Passive and Active Measurement*, Oliver Hohlfeld, Giovane Moura, and Cristel Pelsser (Eds.). Springer International Publishing, 537–551.

[19] Robert Lychev, Samuel Jero, Alexandra Boldyreva, and Cristina Nita-Rotaru. 2015. How Secure and Quick is QUIC? Provable Security and Performance Analyses. In *2015 IEEE Symposium on Security and Privacy*. 214–231. https://doi.org/10.1109/SP.2015.21

[20] Erik Sy, Christian Burkert, Hannes Federrath, and Mathias Fischer. 2019. A QUIC Look at Web Tracking. *Proceedings on Privacy Enhancing Technologies* 2019 (07 2019), 255–266. https://doi.org/10.2478/popets-2019-0046

[21] Martino Trevisan, Francesca Soro, Marco Mellia, Idilio Drago, and Ricardo Morla. 2020. Does Domain Name Encryption Increase Users' Privacy? *SIGCOMM Comput. Commun. Rev.* 50, 3 (July 2020), 16–22. https://doi.org/10.1145/3411740.3411743 Place: New York, NY, USA Publisher: Association for Computing Machinery.

[22] Caleb Yu. 2018. GQUIC Protocol Analysis and Fingerprinting in Zeek. (Aug. 2018). https://engineering.salesforce.com/gquic-protocol-analysis-and-fingerprinting-in-zeek-a4178855d75f/

[23] Lars Wüstrich Zlatina Gancheva, Patrick Sattler. [n. d.]. TLS Fingerprinting Techniques. ([n. d.]). https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2020-04-1/NET-2020-04-1_04.pdf

Appendices

A. Combination of handshake fields used for browser fingerprinting

1. **"JA3_fields"**:
"tls.handshake.ja3_full",
"tls.handshake.ja3"

2. **"transport_parameters"**:
"tls.quic.parameter.initial_max_stream_data_bidi_remote",
"tls.quic.parameter.initial_max_stream_data_uni",
"tls.quic.parameter.active_connection_id_limit",
"tls.quic.parameter.initial_max_streams_uni",
"tls.quic.parameter.max_idle_timeout" ,
"tls.quic.parameter.initial_max_streams_bidi",
"tls.quic.parameter.initial_max_data" ,
"tls.quic.parameter.max_datagram_frame_size",
"tls.quic.parameter.max_ack_delay",
"tls.quic.parameter.initial_max_stream_data_bidi_local"

3. **"quic_data"**:
"quic.header_form", "quic.fixed_bit",
"quic.packet_number_length",
"quic.version",
"quic.frame_type",
"quic.crypto.offset"

4. **"ext_field_1"**:
"tls.handshake.extension.type",
"tls.handshake.extensions_reneg_info_len",
"tls.handshake.extensions_supported_groups",
"tls.handshake.extensions_supported_group"

5. **"ext_field_2"**:
"tls.handshake.extensions_alpn_list",
"tls.handshake.extensions_alpn_str",
"tls.handshake.extensions_status_request_type",
"tls.handshake.extensions_status_request_exts_len",
"tls.handshake.sig_hash_algs",
"tls.handshake.sig_hash_alg",
"tls.handshake.sig_hash_hash",
"tls.handshake.sig_hash_sig",
"tls.handshake.extensions_key_share_group",
"tls.handshake.extensions.supported_versions_len",
"tls.handshake.extensions.supported_version",
"tls.extension.psk_ke_mode",
"tls.record_size_limit"

6. **"remaining_field"**:
"tls.handshake.type",
"tls.handshake.version",
"tls.handshake.cipher_suites_length",
"tls.handshake.ciphersuites",
"tls.handshake.ciphersuite",
"tls.handshake.comp_methods_length",
"tls.handshake.comp_methods",
"tls.handshake.comp_method"

B. Analyzing pcap for fingerprinting

https://github.com/ckh1698/quic.git