QUIC and TCP in Unsafe Networks: A Comparative Analysis

Andrew Simpson[a], Maitha Alshaali[a], Wanqing Tu[a], Muhammad Rizwan Asghar[b]

[a]*Department of Computer Science, Durham University, UK*
[b]*Surrey Centre for Cyber Security, University of Surrey, UK*

Abstract

Secure data transmission and efficient network performance are both key aspects of the modern Internet. Traditionally, TLS/TCP has been used for reliable and secure networking communications. In the past decade, QUIC has been designed and implemented on UDP, attempting to improve security and efficiency of Internet traffic. In this work, we conduct real-world platform investigations to evaluate TLS/TCP and QUIC/UDP in maintaining communication, security and efficiency under three different types of popular cyber-attacks. A set of interesting findings, including delay, loss, server CPU utilisation and server memory usage are presented to provide a comprehensive understanding of the two protocol stacks in performing malicious traffic. More specifically, in terms of the efficiency in achieving short delays and low packet loss rates with limited CPU and memory resources, QUIC/UDP performs better under DoS attacks but TLS/TCP overtakes QUIC/UDP when handling MitM attacks. In terms of security, our implementation of TCP tends to be more secure than QUIC, but QUIC traffic patterns are harder to learn using machine learning methods. We hope that these insights will be informative in protocol selection for future networks and applications, as well as shedding light on the further development of the two protocol stacks.

Keywords: QUIC, TCP, Network Attacks, Denial of Service (DoS), Man-in-the-Middle (MitM), Data Transmission, Network Security

1. Introduction

The Internet traditionally employs the TCP/TLS protocol stack to achieve reliable and secure data transmissions between clients and servers. However, the implementations of TCP and TLS incurs a trade-off between communication efficiency and network reliability or security. This is mainly because that the handshaking processes of TCP and TLS in establishing connections and negotiating security measurements cost additional latency. In order to address this trade-off, Google proposes QUIC (Quick UDP Internet Connections) on top of UDP to support fast and secure networking applications (*e.g.,* web applications). More than half of all connections from the Chrome browser to Google's servers were using QUIC by 2015 [1], and QUIC was standardised by IETF in 2021 [2]. The future of QUIC seems to lead only to more growth as the latest version of HTTP (*e.g.,* HTTP 3) uses QUIC and has been declared by the IETF as the next secure standard for web data transport.

Table 1 presents a comparative anlaysis of QUIC, TCP, and UDP to demonstrate the major features that QUIC has to enhance the efficiency of secured data transport. With QUIC, only one Round Trip Time (RTT) is required to initialise a new connection; whereas, zero RTT is required to resume a connection. This ability is achieved by reusing cached transport and security parameters from a previous connection, then coalescing a new handshake alongside the early application data within the first packets, to provide forward security should these old parameters be compromised. As such, it is commonly believed that QUIC can theoretically establish a connection more quickly than TCP and generate much less overheads when setting up secure connections. However, it is unclear whether QUIC really offers better protection and performance to network users when confronted with various network attack scenarios, or if it can survive through certain attacks at all. In fact, various instances of replay and packet manipulation attacks on QUIC have been identified leading to connection failures and, in certain implementations, necessitating a fallback to TCP (*e.g.,* [3]). Therefore, in this article, we study the efficiency and security of QUIC connections in comparison to TCP connections amidst differ-

Table 1: QUIC enhancements over UDP and TCP.

Functionality	**TCP**	**UDP**	**QUIC**
Connection ID and Migration	✗	✗	✓
1-RTT Handshake	✗	✓	✓
0-RTT Handshake	✗	✓	✓
In-order Delivery Guarantee	✓	✗	✓
Connection Verification	✓	✗	✓
Padding Frames	✗	✗	✓
Anti-amplification Limit	✓	✗	✓

Email address: `wanqing.tu@durham.ac.uk` (Wanqing Tu) ent types of network attacks. More specifically, we look into three types of attacks that are prevalent in the Internet: Denial of Service (DoS) attacks, Man-in-the-Middle (MitM) attacks, and traffic analysis attacks. A real-world platform is developed to analyse factors such as delays, the frequency of interrupted connections, and the utilisation of server resources when QUIC and TCP handle the three types of attacks. From our investigations, we derive the following novel insights:

- **Efficiency.** Our investigations in general indicate that QUIC is more efficient when encountering DoS attacks, while TCP is more efficient when handling MitM attacks. QUIC connections achieve better delays and packet loss rates while using CPU and memory resources less in DoS attacks. TCP achieves short server connection delays and less data loss rates in MitM attacks, but with similar or slightly higher CPU or memory resource usage.

- **Security.** We observe the security of QUIC and TCP connections when traffic analysis attacks in the network. Our observations find that, while our implementation of TCP tends to be more secure than QUIC, QUIC traffic patterns are surprisingly harder to learn using machine learning methods (*e.g.,* Deep Convolutional Neural Networks).

To the best of our knowledge, there is no existing work comparing the efficiency of QUIC and TCP under network attacks covered in our study. Our real-world platform also allows us to evaluate native features, such as network connectivity, CPU and memory use, providing an accurate and comprehensive investigation. Our results will not only provide guidance for selecting the appropriate transport protocol for combating attacks but also offer insights into the limitations of QUIC or TCP protocols in terms of efficient and secure connections.

The rest of this article is organised as follows. Section 2 reviews related work. Section 3 discusses the key features of the QUIC protocol. Section 4 provides the experimental design. Section 4.2 describes the testing platform. Section 4.3 explains the evaluation metrics. Section 5 presents the results. Section 6 this work and offers research directions for future work.

2. Related Work

In this section, we review existing studies on QUIC by focusing on work on security and performance. Many studies explored QUIC security as evident from the most recent study by Murthy, Asghar, and Tu [4] in which they present a data-driven framework for optimising security-efficiency trade-off in QUIC, where use case scenarios have been leveraged to maximise benefits. Chen *et al.* [5] compared the security and availability of TLS and QUIC and found that QUIC implements the reset authentication function because of its stateless reset mechanism. In [6], symbolic model checking is used to analyse the security of QUIC's handshake protocol. It identifies a vulnerability of QUIC allowing an attacker to complete a handshake as a legitimate client with a server. Fischlin *et al.* [7] proposed QUICi that implements a more sophisticated key generation technique to enhance the potential security flaws in QUIC's handshakes. In [3], the quick communication protocol is introduced to regulate the utilisation of the initial session key before the final session key is created during handshakes. In [8], in order to address the forward security issue that the 0-RTT handshake of QUIC experiences, the Belichenbacher attack is designed to quickly guess a server's secret key. In addition to these efforts, quite a few studies (*e.g.,* [9]) investigated different QUIC versions and implementations to verify their key features and security characteristics. Interested readers may find an overview of such efforts in [10]. In general, these experimental or analytical studies attempt to understand QUIC's security properties without taking efficient performance into account.

As for performance, Yu *et al.* investigated QUIC and TCP by analysing production environments [11]. This work finds that QUIC has advantages over TCP but at the cost of high implementation overhead. Biswal *et al.* studied the web page loading latencies of QUIC and HTTP/2 under different network conditions [12]. Their studies show that QUIC is faster than HTTP/2 even when network conditions are poor. In [13], QUIC's congestion response is studied, which concludes that QUIC data takes more link capacity when sharing with TCP in a bottleneck link and hence achieves higher throughput. In [14], the efficiency of QUIC is proved to outperform TCP in 5G network. However, these studies do not consider QUIC's performance under network attacks, which is the focus of our study.

There are a few studies investigated QUIC's performance under network attacks. Lychev *et al.* studied the security and performance of QUIC when replay or manipulation attacks appear, which demonstrates the low security and long latency of QUIC in handling these attacks [3]. Overall, most of existing studies present the insights that QUIC's security and performance may be compromised by network conditions and attacks. This encourages our curiosity in exploring whether QUIC may always outperform its predecessor TCP in achieving efficient yet secure connections under popular network attacks. In the next sections, we will present our study to answer this question.

3. Key Features of QUIC Protocol

In this section, we discuss the key features of QUIC protocol, as compared to TCP protocol, that contribute to faster, secure, and reliable connections.

3.1. Connection IDs and Migration

Connections held by a QUIC endpoint are given a unique source and destination ID pair, connID scID:dcID, which has no linkage to the IP address or port of the remote endpoint. This connID is used to resume a connection if the path, or even the client address, changes (and that change is detected by a probe packet). To determine the connID pair for a connection at a client endpoint, the initial packet is sent with a set scID, provided there is no other client with that ID. The server then registers that scID as its own dcID, and replies with a source scID, which should be used as the client's own destination dcID. To put it simply, this creates a binding on both the client and server that uniquely and robustly identifies this connection for its lifetime.

3.2. 1-RTT Handshakes and 0-RTT Handshakes

QUIC balances the fast connection initialisation provided by UDP with the security of TCP/TLS by combining the cryptographic handshake with the transport parameter negotiation. When a client wishes to initialise a connection over QUIC with a server for the first time, it will be carried out with a 1-RTT handshake. The first packet sent from the client is an initial packet requesting a connection with the server. As this is the first time connecting, the server does not recognise the client as it inevitably provided incomplete transport parameters. The server rejects the incoming request and replies with the handshake information required to start a valid connection, including a connection ID pair and a public encryption key. The client uses this handshake information to reply with a valid initial packet, followed by a valid handshake packet containing the correct transport parameters and encryption token. Application stream data can be sent from this point.

0-RTT handshakes refer to QUIC's ability to resume a connection without going through the handshakes of cryptographic key exchange between a server and a client. QUIC endpoints are able to cache connection information and reuse a cached connection within a certain amount of time, bypassing the delay of a handshake entirely. More specifically, when a client wishes to resume a connection with a cached server, it will carry out a process similar to the 1-RTT handshake. That is, the first packet sent from the client is an initial packet requesting a connection with the server. However, this packet can also contain application data encrypted with the cached key from a previous connection, validated with a cached server token. The handshake is completed as before and application stream data can be sent from this point. New parameters are also negotiated at this point to guarantee secrecy going forward.

3.3. In-order Delivery

Like TCP, QUIC can guarantee in order delivery on top of send and forget approach of UDP, when requested by an application-layer protocol. It does this by numbering packets in the intended order of delivery, transmitting them, and reordering them upon receipt for use in the application. The benefit to taking this approach is that the ordering of packets can be disabled to increase performance in non-critical, lossy connections without violating the transport protocol.

3.4. Connection Verification

In TLS/TCP, all packets pertaining to handshakes are verified by the server. If at any point there is an inconsistency, the entire connection is reset without warning to prevent DoS or spoofing attacks. With the QUIC protocol, the reduced number of round trips in the handshake process means that the client's initial packet cannot be verified by a server. This is because QUIC endpoints encrypt and verify all packets using the keying material established during the handshake process, which has not been completed yet. QUIC is designed to work around this by giving all incoming requests the benefit of the doubt during the handshake, before dropping any illegitimate connections once they are recognised. This recognition delay allows for a variety of handshake spoofing attacks to be attempted, but all are stopped after one round trip – when the server detects and inconsistency in the handshake negotiations.

3.5. Padding Frames

QUIC inserts padding frames into each packet. Padding frames are of arbitrary length and provide endpoints with the ability to obscure the total length of a packet. The obscurity of data length may benefit the mitigation of traffic attacks. However, using length is not the only way to information leakage. Padding frames cannot prevent other attacks (*e.g.,* side-channel attacks) entirely.

3.6. Anti-amplification Limit

With all unverified connections, specially crafted frames within a packet may cause an endpoint to produce a much greater number of packets in response. In the case that an adversary chose to do this maliciously, they could inject these special frames into a packet with a spoofed victim's IP address. Sending these frames within an initial packet as part of an unverified handshake attempt could cause the server to redirect a large volume of traffic to them, severely impacting their network performance. QUIC implements an anti-amplification limit, which mitigates the severity of this type of attack by restricting the maximum amount of data that can be sent to an unverified endpoint to three times the amount of incoming data.

4. Experimental Design

4.1. Selection of Network Attack Scenarios

This subsection describes the network attacks that we select to test efficiency and benchmark security when networks use QUIC as compared to TCP. For the purpose of evaluating efficiency, we look into how quickly clients can

Table 2: Testing platform with hardware specifications.

Device	CPU	No. of Threads	Max. Speed	Memory	Operating System
Server	Celeron N3050	2	2.16GHz	4GB	Ubuntu
Client	Ryzen 7 3700X	16	4.4Ghz	32GB	Arch Linux
Attacker	Ryzen 7 5800H	16	4.4Ghz	16GB	Arch Linux
Router	ARM Cortex-A7	4	900Mhz	1GB	Raspbian

establish reliable connections with a server. As for security, we investigate the core principles, i.e., confidentiality, integrity, and availability. As such, we select Denial of Service (DoS) attacks, Man-in-the-Middle (MitM) attacks, and traffic analysis for our investigations. This is because DoS attacks target the server resources that are important for establishing and managing connections, affecting data transmission availability and efficiency. MitM attacks interfere with the integrity and confidentiality of a connection, degrading communication security. As for traffic analysis, it is selected as an attack type affecting client security and efficiency, allowing complementary investigations to the previous two types. We also choose the above three types of attacks such that both QUIC and TCP are likely to be affected in similar ways, avoiding any that are protocol specific as they would not provide comparable differences between the two.

4.1.1. Denial of Service (DoS) Attacks

DoS attacks are constantly increasing in frequency across all areas of the web. Although there are services such as CloudFlare that aim to detect and mitigate these attacks, the global volume of attacks is still rising due to reasons such as growing online presence, increasing botnets, and malicious activities. We plan to evaluate QUIC and TCP under the two DoS attacks: connection flooding and slowloris.

Connection flooding is performed by sending a high volume of tiny requests to a server over a short period of time. Using this attack, the majority of the processing done on the server will be to open or close connections, thus allowing us to measure the overhead caused by these parts of the connection while neglecting the relatively small amount caused by the requests itself. Slowloris attacks aim to slowly consume servers' resources by opening connections and not closing them. We hope that such attacks will shift the majority of connection overheads to maintaining many active connections, enabling us to perform a detailed analysis and neglect the relatively small overhead caused by other factors.

4.1.2. Man-in-the-Middle (MitM) Attacks

Online MitM attacks are where packets are intercepted or changed by an on-path adversary. While any public fields in a packet can be manipulated in an attempt to interrupt a connection, in order to produce a more meaningful comparison between QUIC and TCP, we select the two attacks: immediate client echoes and random server relays that can impact on shared or similar fields of QUIC and TCP.

Client echoes typically establish a socket to get a connection to a server. Attackers read input from a client on the standard input stream and then forward that text to the server by writing the text to the established socket. The server's replies are also back to the client through the socket. Client echoes can be developed to evaluate the performance of QUIC's connIDs.

A replay attack is to detect and identify a data transmission which will then be delayed or repeated by the attack. With random server relays, when a client sends a request to a server, the server randomly respond with a previous server message to confuse the client the invalid shared connection parameters at the transport layer but valid packet content. This attack is in contrast to the immediate client echo attacks.

4.1.3. Traffic Analysis

Traffic analysis is selected as an attack type affecting the client security rather than the server efficiency, aiming to contrast the previous two categories. We plan to experiment website fingerprinting attacks that analyse traffic patterns based on which such attacks infer confidential information about clients.

4.2. The Testing Platform

Our testing platform consists of a server, a client, a router, and an attacker. Table 2 gives the specifications of these four components. In our experiments, the client repeatedly opens or closes connections to time corresponding responses. To measure security, the client sends encrypted key or perform some unknown activity while the attacker aims to detect and decrypt these. The server employs Nginx [15], which is one of the most popular and widely used web servers that supports both TCP and QUIC in one implementation. Nginx could provide good comparisons for our work. The router is a Raspberry Pi with a generic DHCP server installed to connect the client, the server, and the attacker together.

The attacker implements the network attacks described in Section 3. For connection flooding attacks, we spawn a thread every n seconds, which opens a connection with the server, requests the "/" resource using GET, then closes the connection. Slowloris attacks are carried out using multiple threads, which keep request data from the server periodically without closing the connections. Our tests refer to the aioquic test and slowloris python package [16] for the two DoS attacks.

The MitM attacks are implemented via a python script running the scapy package [17]. With immediate client echoes, a packet that the client sends will be echoed. The action is to confuse the client with valid shared parameters but invalid packet content. With random server replays, the server stores all packets sent to the client, then any time the client sends a packet to the server, the attack randomly responds with one of the previous server messages.

As for website fingerprinting attacks, we use Wireshark [18] to record 50,000 packets from three Google websites, Sheets [19], Docs [20], and Slides [21]. These recorded packets are pre-processed using our method modified based on [22, 23], before training a Deep Convolutional Neural Network to classify the labelled packet sequences. The code developed for the platform is in GitHub [24].

Finally, in order to control networking variables such as bandwidth and delay that may affect the performance that we observe for both protocols, we implemented the following preventative actions. Firstly, the testing setup was within a local wireless network and the locally connected endpoints were kept consistent between experiments. The aim was to ensure that no external factors could influence the bandwidth or latency of the network during each attack. In addition to network traffic, the CPU and memory consumption at endpoints can affect experimental results. To prevent such impact, all endpoints were ensured to have only software running that was relevant to each experiment and the system was restarted between tests to remove the chance that one attack might influence another at a later time. The control measurements for each metric within the experiments show that this was successful, as the data remains consistent throughout the paper.

4.3. Evaluation Metrics

We evaluate the following metrics in our experiments to compare QUIC and TCP.

- **Client delay:** It is the time taken from initialising a handshake to closing the connection after data is received. It is employed to evaluate the efficiency of both protocols in accepting client connections when attacks happen.

- **Loss rate:** It refers to the ratio of the amount of data lost to the amount of data transmitted in total. The loss rate is used to evaluate the efficiency of using resources to transmit data, as retransmissions are required by TCP. Data loss also affects communication integrity.

- **Server CPU utilisation:** It monitors the per-process utilisation of CPUs. We take a reading every 0.1 seconds over 50 seconds to account for any variation. CPU usage is used to evaluate the resource efficiency at endpoints of the two protocols in handling attacks.

- **Server memory utilisation:** It monitors the per-process memory utilisation. We observe this performance over a period of 50 seconds. Memory usage is used to evaluate the resource efficiency at endpoints of the two protocols in handling attacks.

- **F1-score:** The packets that we recorded are equally distributed across classes. For any one class, say a, given the true positive amount (TP_a), the false positive amount (FP_a), and the false negative amount (FN_a), its standard F1-score is

  $$\frac{TP_a}{TP_a + \frac{1}{2}(FP_a + FN_a)}$$

  This score is calculated for all classes, and then a macro, or unweighted, average is taken to produce the final F1-score. F1-scores are evaluated to look into the security protections of the two protocols under different attacks.

- **Protocol conditional model accuracy:** We base this metric on the formula for conditional probability where C is the event where a flow is predicted correctly, and F is the event that a protocol is present in a flow. The probability that a flow is predicted correctly given that said flow contains a specific protocol can be expressed as

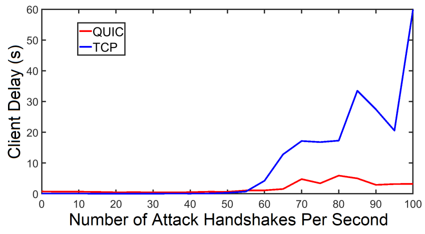
  $$P(C|F) = \frac{P(C \cap F)}{P(F)}$$

  We calculate this probability for each protocol present in each dataset and directly compare the values as a measure of how secure each one is. The protocol conditional model accuracy is used to complement the security scenarios that F1-score may not measure. This is because the packets that leak the most information may have actually been transmitted using a different protocol - although when QUIC is enabled, some of the processes still use TCP and UDP to send data.

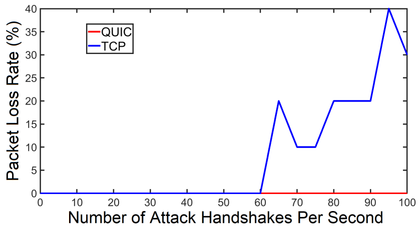
5. Results

5.1. DoS Attacks

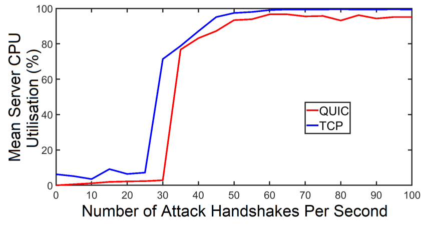
Our first experiment is to evaluate QUIC and TCP when experiencing connection flooding attacks. Figure 1a shows the comparison of client connection delays of the two protocols. QUIC achieves shorter client connection delays than TCP. When there are 100 attacks per second, the client connection delays with TCP is almost 20 times of those with QUIC in our experiment. This shows that QUIC can handle a large amount of connection flooding attacks while maintaining the quick connections between clients and the server, owing to its 1-RTT handshakes to initiate connections. A QUIC server and client complete



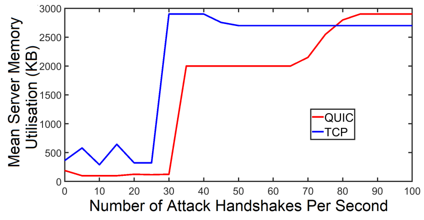
(a) Delay.



(b) Loss.

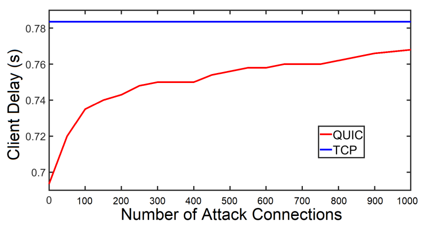


(c) Server CPU utilisation.

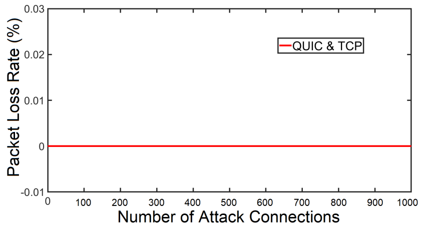


(d) Server memory usage.

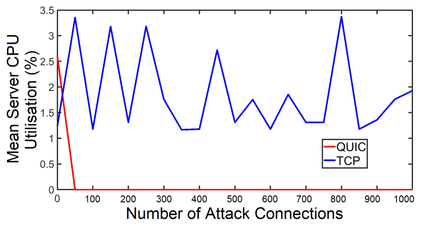
Figure 1: Measurements for QUIC vs TCP during the flooding attack.



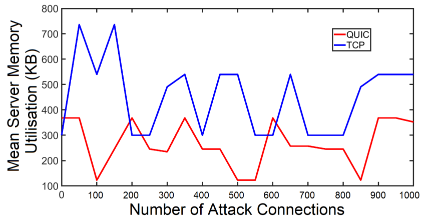
(a) Delay.



(b) Loss.



(c) Server CPU utilisation.



(d) Server memory usage.

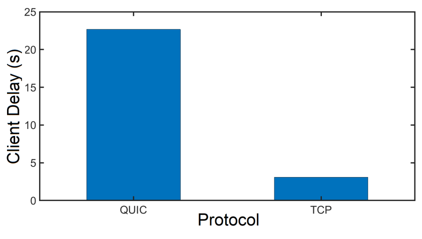
Figure 2: Measurements for QUIC vs TCP during the slowloris attack.

their exchange of setup keys and supported information in the initial handshake process. However, with TLS/TCP, the connection between server and client requires them to process TLS handshakes, in addition to TCP's 3-way handshake. The lengthy TLS/TCP initial procedure incurs delays.

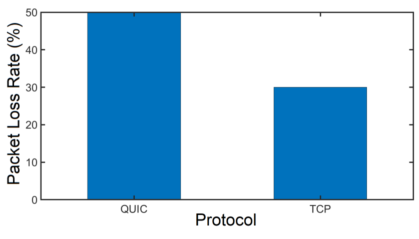
Figure 1b presents the packet losses of QUIC and TCP. Both QUIC and TCP do not lose packets when the number of attacks per second is small (under 60 in our experiment). However, the number of lost TCP packets increases greatly when the number of attacks is greater than 60. This is mainly because TCP and TLS handshakes consume more CPU capacity than QUIC does. When a large number of connection requests arrive (60 attacks per second in our experiment), with TLS/TCP, our CPU cannot handle them all and hence starts dropping data.

Figure 1c illustrates the CPU usage of QUIC and TCP in this experiment. QUIC has a lower CPU usage than TCP. We observe that, with TCP, the server's CPU becomes fully occupied when the number of attacks per second increases (above 50 in our experiment). For QUIC, it does not use up CPU capacity even when there are 100 attacks per second in our experiment. Figure 1d is about the memory usage of both protocols. QUIC has a lower memory usage than TCP in general. This is mainly due to QUIC's requiring less handshake overhead, generating less information to be cached. TCP needs to cache the status information of clients (e.g., sequence numbers, acknowledgement numbers) that have requested connections.

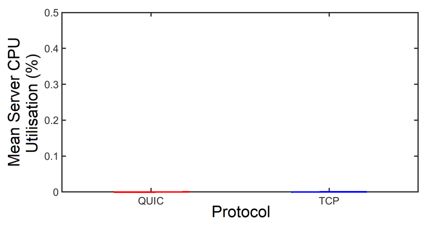
As connection flooding attacks leverage short-term connections, our second experiment observes the efficiency of QUIC and TCP when being attacked with long-term connections, *i.e.,* slowloris attacks. Figure 2a reports the client connection delays of QUIC and TCP when the number of slowloris attacks increases from 0 to 1000. Both protocols do not have greatly increased delays when the num-



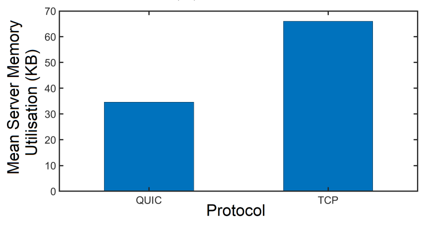
(a) Delay.



(b) Loss.

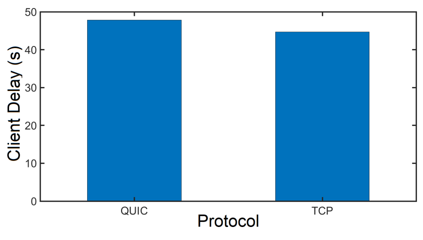


(c) Server CPU utilisation.

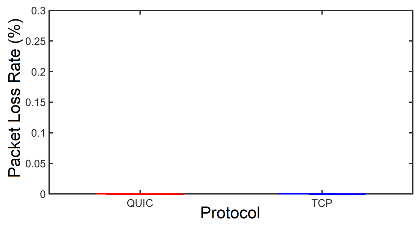


(d) Server memory usage.

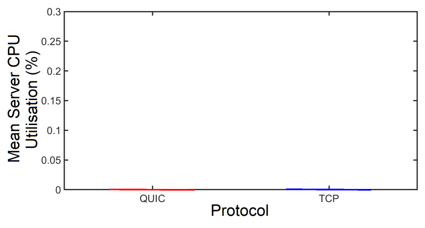
Figure 3: Measurements for QUIC vs TCP during the immediate client echo attack.



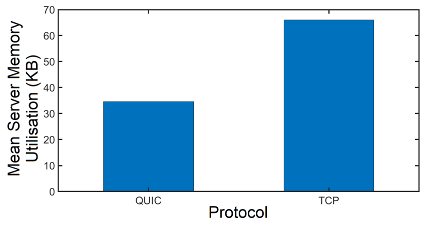
(a) Delay.



(b) Loss.



(c) Server CPU utilisation.



(d) Server memory usage.

Figure 4: Measurements for QUIC vs TCP during the random server replay attack.

ber of attacks increases greatly. This is because slowloris attacks do not disconnect connections once established. As such, both QUIC and TCP do not need to generate as many handshakes as they do in the experiment of client flooding attacks. Therefore, both the server and the network have more capacity to accept connection requsts once these requests are issued, contributing to short client delays.

Figure 2b shows the data loss rate performance. Both protocols have presented a practical performance that avoids packet loss during slowloris attacks in our experiment. Similar to the reason for short client delays, under slowloris attacks, QUIC and TCP implement less handshakes, consuming less CPU and memory capacity. In our experiment, the server's CPU and memory are able to handle these attacks (as in 2c and 2d) without dropping packets.

In our experiment, with 0-1000 slowloris attacks, the CPU utilisation with both QUIC and TCP is rarely registered above 5%, as shown in 2c. The memory utilisation curves in Figure 2d for both protocols fluctuate with the increasing in the number of attacks due to the different processes of the two protocols in handling connections and security. With slowloris attacks, QUIC uses less memory than TCP does as QUIC caches less status information as compared to TCP.

Overall, when handling DoS attacks issuing short-term connections, QUIC outperforms TCP in terms of better delay and packet loss performance. QUIC is able to achieve such performance enhancement by using less computing resources. When handling DoS attacks issuing long-term connections, QUIC achieves similar performance as TCP but with less memory usages.

5.2. Man-In-the-Middle (MitM) Attacks

In order to evaluate the efficiency of QUIC and TCP under MitM attacks, we first experiment the immediate client echo attacks. This attack uses an online MitM mechanism to reflect all packets sent from a client back to that client in an attempt to interrupt the expectation of well-formed incoming traffic at an endpoint. Since this attack does not directly involve the server, our observations of Figures 3c and 3d are to be expected, with only the baseline CPU and memory usage at the server. In terms of delay, Figure 3a shows that the QUIC server requires longer delas to accept a connection request than the TCP server does under immediate client echo attacks. This is because QUIC packets are largely encrypted and always authenticated, requiring longer processing time at endpoints. This prolongs the connection request procedures at the client. Similarly, Figure 3b finds that this attack causes QUIC to lose more data than TCP does due to the additional authentication and encryption processes.

In Figure 4, we report our experimental results when QUIC and TCP experience random server replay attacks. The CPU and memory usage for random server replay attacks is similar to the previous packet manipulation attacks, where we observed no increase in either metric for the duration of the attack, as shown in Figures 4c and 4d. However, the delays and data loss rates present some differences as compared to those in Figure 3. While the delays for both protocols to accept client connection requests are in $[40, 50]$ms, the QUIC server needs a slightly longer delay than the TCP server does, as illustrated in Figure 4a. Compared to client echo attacks, both QUIC and TCP experience longer client delays under random server replay attacks. This is because server replay attacks cause the server to handle more connection requests, which affects TCP's client delays more due to the greater handshake overhead in initiating TLS/TCP connections.

Overall, MitM attacks do not exhaust the CPU and memory resources of QUIC and TCP. In terms of performance under this type of attacks, QUIC connections experience worse performance than TCP connections. The worse performance of QUIC connections is less obvious under the MitM attacks on servers than the poor performance of QUIC connections under the MitM attacks on clients.

5.3. Traffic Analysis Attacks

In this section, we observe the security of QUIC connections and TLS/TCP connections under traffic analysis attacks. We experiment the website fingerprinting attacks which aim to expose side channel leaks in each of the protocols using a Deep Convolutional Neural Network on data captured from three Google sites. We observe F1-score and protocol conditional model accuracy, alongside model loss to evaluate the training process. Model loss represents the mean squared error between the true class for an input and the predicted class given by the model.

Figure 5 demonstrates a downward trend in model losses: Figure 5a for QUIC and Figure 5b for TCP. However, Figure 5b shows a closer alignment between train and test loss. We regard this difference as not significant, as the results for training this model showed a variation of disparities between losses in previous tests, and this was most likely dependent on the random initialisation of the optimiser we used, not the dataset. Generally, after some number of epochs, every training cycle ended in a lower train loss than the test loss, which signifies the model was overfitting to the training dataset, and when consistently occurring is a strong identifier of the dynamic fluctuations of network traffic due to small or highly variant data.

Table 3: QUIC vs TCP CNN model F1-score.

Protocol	**F1-Score**
QUIC	0.63497
TCP	0.67495

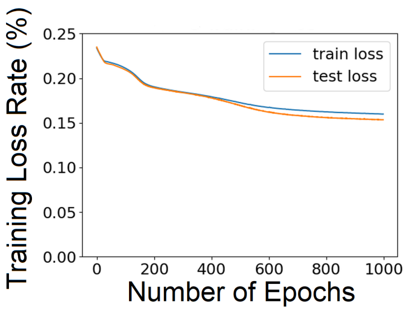
We show the F1-score for each trained model on a 10% test subset in Table 3. QUIC had a slightly lower F1-score, and this was consistent over multiple runs, suggesting the QUIC traffic was harder to learn from than the TCP traffic. Clearly, low F1-scores for a classifier lead to poor model performance overall. We expected this to be the case in such a general attack with no specific packet ordering like that of studies in early traffic [17]. If the model had a high F1-score, this would strongly imply the implementation has no packet obfuscation, and perhaps even a flawed encryption scheme.

We finally present Figure 6, which aims to give insight into which protocols within a flow are actually responsible for leaking data about the nature of a connection to our convolutional neural network. The conditional accuracy for each protocol, or column in the plot, was calculated using the function defined in Section 6, across data from all three websites that we visited in our data collection phase. We observe that UDP is the most informative to an adversary, while QUIC also leaks information. For TCP, we unsurprisingly present how the newer the TLS version, the less data that is leaked unintentionally to an on path adversary. The important finding to take from this attack and our metrics is that the QUIC model performed worse than TCP. On the other hand, the QUIC model mainly got its information from the UDP packets within the dataset. This is strong evidence for this implementation of TCP being more secure than QUIC, but the main culprit for QUIC's weakness is actually the involvement of UDP, so rectifying this would simply require removing the usage of UDP packets in the QUIC implementation.

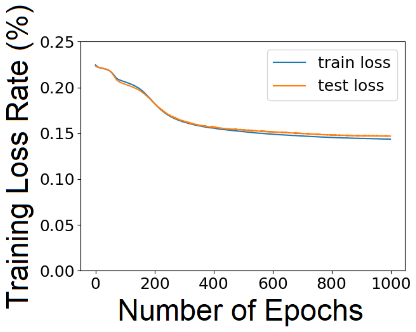
Overall, our implementation of TCP tends to be more secure than QUIC. However, QUIC traffic presents the property of being harder to learn by machine learning methods than TCP traffic.

6. Conclusion

In this work, we investigated the performance of QUIC and TCP in experiencing three types of network attacks: DoS attacks, MitM attacks, and traffic analysis attacks.

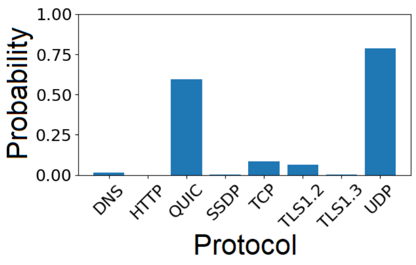


(a) QUIC CNN model loss.

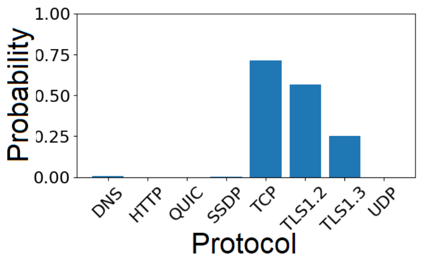


(b) TCP CNN model loss.

Figure 5: CNN model losses for QUIC and TCP.



(a) QUIC



(b) TCP

Figure 6: Protocol conditional model accuracies for QUIC and TCP.

We conducted real-world experiments to look into delays, loss rates, CPU and memory usage, and the resilience of the two protocols to security threats. Our experimental results indicate that, for reducing delay and packet loss rates by using less CPU/memory resources, QUIC is more efficient than TCP under DoS attacks but less efficient than TCP under MitM attacks. In terms of security, our experimental results suggest that TLS/TCP tends to be more secure than QUIC. We discovered that QUIC traffic patterns are harder to learn by machine learning methods. To further enhance the efficiency of the testing platform, instead of controlling these actions manually, another low-powered device (such as a raspberry pi) can be added to the network with the purpose of controlling the simulated attacks and logging systems. Doing so allows tests to be set up much faster than performing these actions manually.

We hope that our results will contribute valuable insights to not only inform protocol selection but also guide future protocol designs toward the best balance between network resilience and communication efficiency when various attacks occur. Particularly, as network threats evolve, schemes that can support QUIC to achieve additional efficiency and reduction in network traffic so as to further mitigate channel saturation and jamming issues is one future research direction. Strategies that can enable TCP to limit its CPU or memory usage at servers without compromising performance under evolving attacks is another future research direction. Additionally, in light of the different performance of QUIC and TCP when experiencing different types of attacks, it will be interesting to design machine learning schemes or artificial intelligence approaches to enable clients and servers to automatically verify and choose between QUIC and TCP for the best communication and threat prevention performance.

References

[1] G. LLC, A QUIC update on Google's experimental transport (Apr 2015).
URL `https://tinyurl.com/2p4ze3zz`

[2] E. J. Iyengar, E. M. Thomson, QUIC: A UDP-based multiplexed and secure transport, RFC 9000, RFC Editor (May 2021).
URL `https://www.rfc-editor.org/rfc/rfc9000`

[3] R. Lychev, S. Jero, A. Boldyreva, C. Nita-Rotaru, How secure and quick is QUIC? provable security and performance analyses 2015 (2015) 214–231. doi:10.1109/SP.2015.21.

[4] A. Murthy, M. R. Asghar, W. Tu, Towards a data-driven framework for optimizing security-efficiency tradeoff in QUIC, Security and Privacy 5 (1) (2022) e184.

[5] S. Chen, S. Jero, M. Jagielski, A. Boldyreva, C. Nita-Rotaru, Secure communication channel establishment: TLS 1.3 (over TCP fast open) vs. QUIC, in: Computer Security – ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part I, Springer-Verlag, Berlin, Heidelberg, 2019, pp. 404–426, `https://doi.org/10.1007/978-3-030-29959-0_20`.

[6] J. Zhang, L. Yang, X. Gao, G. Tang, J. Zhang, Q. Wang, Formal analysis of QUIC handshake protocol using symbolic model checking, IEEE Access 9 (2021) 14836–14848. doi:10.1109/ACCESS.2021.3052578.

[7] M. Fischlin, F. Günther, 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, USA, 2014, p. 1193–1204. doi:10.1145/2660267.2660308.
URL `https://doi.org/10.1145/2660267.2660308`

[8] T. Jager, J. Schwenk, J. Somorovsky, On the security of TLS 1.3 and QUIC against weaknesses in PKCS#1 v1.5 encryption, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 1185–1196. doi:10.1145/2810103.2813657.
URL `https://doi.org/10.1145/2810103.2813657`

[9] R. Marx, W. Lamotte, J. Reynders, K. Pittevils, P. Quax, Towards QUIC debuggability, 2018, pp. 1–7. doi:10.1145/3284850.3284851.

[10] S. Tatschner, S. N. Peters, D. Emeis, J. Morris, T. Newe, A quic(k) security overview: A literature research on implemented security recommendations, in: Proceedings of the 18th International Conference on Availability, Reliability and Security, ARES '23, Association for Computing Machinery, New York, NY, USA, 2023. doi:10.1145/3600160.3605164.
URL `https://doi.org/10.1145/3600160.3605164`

[11] A. Yu, T. A. Benson, Dissecting performance of production QUIC, in: Proceedings of the Web Conference 2021, WWW'21, Association for Computing Machinery, New York, USA, 2021, p. 1157–1168. doi:10.1145/3442381.3450103.
URL `https://doi.org/10.1145/3442381.3450103`

[12] P. Biswal, O. Gnawali, Does QUIC make the web faster?, 2016 IEEE Global Communications Conference (GLOBECOM) (2016) 1–6.
URL `https://api.semanticscholar.org/CorpusID:10245063`

[13] A. Srivastava, Performance evaluation of QUIC protocol under network congestion, Thesis, Worcester Polytechnic Institute, 100 Institute Road, Worcester MA 01609-2280 USA (April 2017).

[14] P. Gärdborn, Is QUIC a better choice than TCP in the 5G core network service based architecture?, Ph.D. thesis (2020).
URL
`https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-289169`

[15] Mercurial, Nginx (Nov 2022).
URL `https://hg.nginx.org//`

[16] G. Yaltirakli, Slowloris (Feb 2023).
URL `https://pypi.org/project/Slowloris/`

[17] P. Biondi, scapy (Dec 2022).
URL `https://pypi.org/project/scapy/`

[18] Wireshark Foundation (Oct 2022). [link].
URL `https://www.wireshark.org`

[19] Google, Google Sheets (2023).
URL `https://docs.google.com/spreadsheets`

[20] Google, Google Docs (2023).
URL `https://docs.google.com/document`

[21] Google, Google Slides (2023).
URL `https://docs.google.com/document`

[22] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, M. Saberian, Deep Packet: A novel approach for encrypted traffic classification using deep learning, Soft Computing 24 (3) (2020) 1999–2012.

[23] V. Tong, H. A. Tran, S. Souihi, A. Mellouk, A novel QUIC traffic classifier based on convolutional neural networks, in: 2018 IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1–6. doi:10.1109/GLOCOM.2018.8647128.

[24] [link].
URL `https://github.com/andrews891/quicvstcpanalysis`