How Secure and Quick is QUIC?

Provable Security and Performance Analyses

Jakob Roberts

In this presentation

Quick what is QUIC?

The importance of Security.

What is a security related proof?

About the paper & Comments.

Comparisons & Feedback.

Quickly: What is QUIC?

QUIC = Quick UDP Internet Connections

Sits on top of UDP

Implements its own form of encryption

Uses HTTP/2’s API as a top end

Reduced connection and transport latency

Congestion avoidance similar to TCP

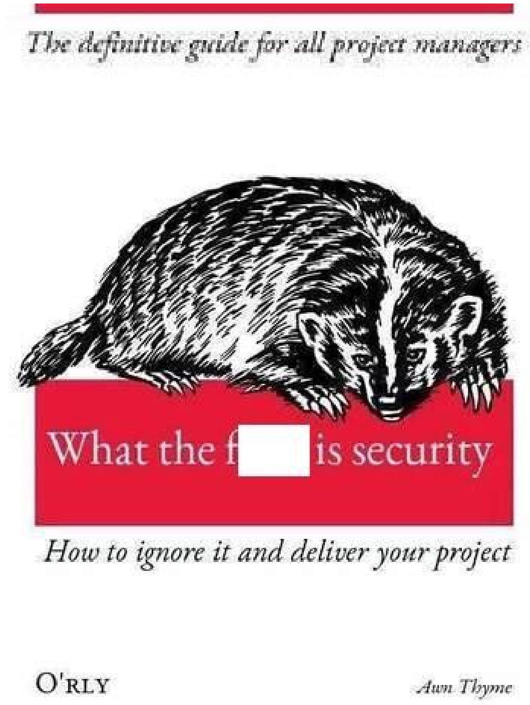
Developed by Google and currently being used by Chrome Browser

Why is security important?

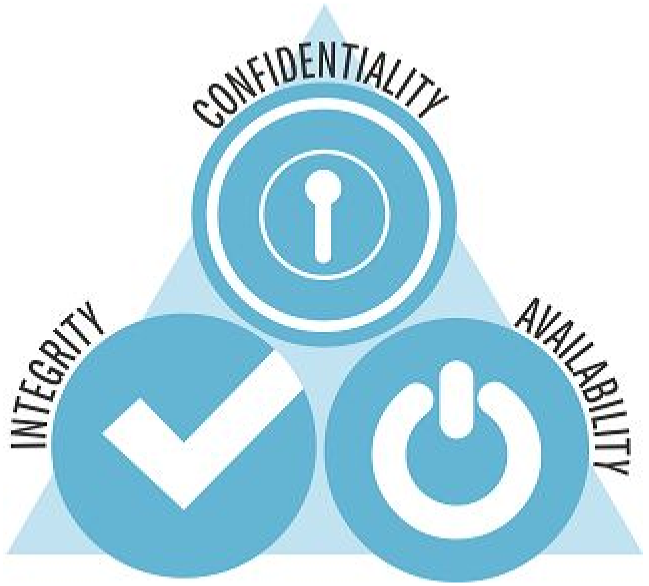
Overlooked most of the time.

Can make or break an application.

Without it, would people notice?


The definitive guide for all project managers
What the f is security
How to ignore it and deliver your project
O'RLY
Awn Thyme

Why is security important?



What is a security proof?

In general:

> It is a mathematical proof/approach to show that the code is secure by design based on the given attacker model and the system model. If all potential attacker threats can be accounted for, the system is considered to be proven secure.

The idea is sound in concept, but lacks a concrete model and is still not widely accepted as a standard as no formal guarantees can be made on the presumptions made within each proof.

Feedback on Writeup

The Problem
The problem is that current encrypted connections are usually associated with high overheads and latency. QUIC hopes to decrease the latency while maintaining similar encryption standards. This paper goes into extreme depth(proofs) as to how QUIC stacks up in the security front.

Main Idea
The idea isn’t much different, they are trying to compare security between protocols by providing a provable security analysis. The proofs associated with the analysis will give validity to the claims in security. At the time of paper publication, there have been other internet protocol alternatives, but in the paper they compare to TCP with TLS.

Feedback on Writeup

Major Strengths
They provide a great summary of what is listed in the paper in the introduction sections so you can get an understanding of what types of analysis were done. Despite all the complicated terminology, the authors spend a good deal of important time setting up information(sections 2, 3, 4, 5) relevant to the actual research sections(6, 7, 8) in order to help understand the relevance of the research done. In section 9, they actually talk about the results of well known attack types and how they would relate to QUIC, instead of doing a mathematical proof. They speak in the conclusion that QUIC has pitfalls, as would any performance based system.

Feedback on Writeup

Major Weaknesses

Unfortunately the paper is not very friendly to people who are not already familiar with security related topics. In section 3 they describe naming conventions and set up mathematical assumptions, this means that whoever is reading the paper also needs to be well versed in the understanding of proof theory. Despite nicely having the sections separated. I feel like this paper could have been separated into multiple topics: one describing the mathematical proofs behind QUIC’s security, and another more qualitative talking about the known attack vectors. I don’t expect that these things to be improved upon.

Feedback on Writeup

Possible Improvement

I would look at comparing QUIC to other up-and-coming internet protocol architectures and see how their performance compares. I don’t think I could do anything better as it is out of my scope of knowledge.

Security basics of QUIC

QUIC implements two keys instead of a single as in TLS

Protection against IP Spoofing and packet reordering

Instead of using the ACCE model from TLS, they have created QACCE where the Q stands for Quick and is designed for Quick Communication protocols.

Overview of paper

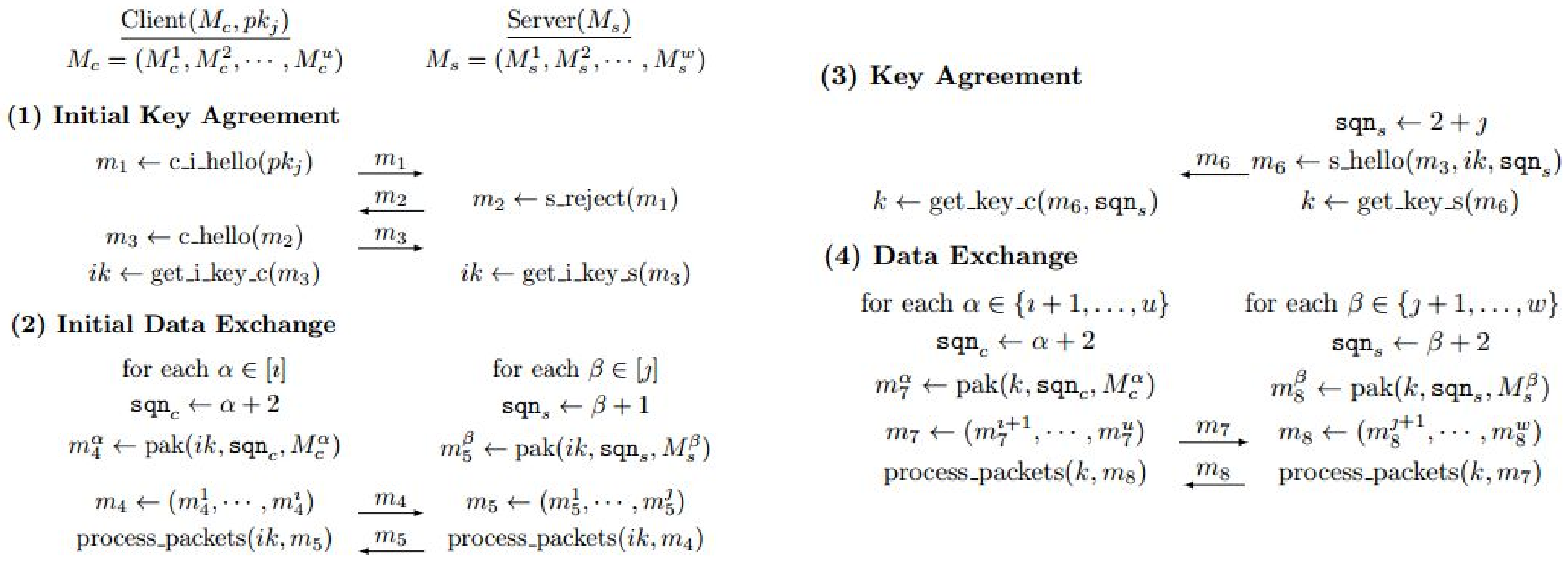
The paper is able to shed light on the guarantees and weaknesses of QUIC that would be useful for anyone developing with the protocol.

They were able to detail the exact level of security that QUIC can provide.

A Quick Connection Protocol(QC) can be defined where there is a communication between a client and a server. An initial key is exchanged and then a final key is set to further continue information exchange. Both QUIC and TLS1.3 fit this definition.

QUIC does not stack up against some TLS modes such as TLS-DHE as the forward secrecy is lost.

RTT Connection Establishment



$$\begin{array}{ccc} \underline{\text{Client}(M_c, pk_\jmath)} & & \underline{\text{Server}(M_s)} \\ M_c = (M_c^1, M_c^2, \cdots, M_c^u) & & M_s = (M_s^1, M_s^2, \cdots, M_s^w) \end{array}$$

(1) Initial Key Agreement

$$\begin{array}{ccc} m_1 \leftarrow \text{c\_i\_hello}(pk_\jmath) & \xrightarrow{m_1} & \\ & \xleftarrow{m_2} & m_2 \leftarrow \text{s\_reject}(m_1) \\ m_3 \leftarrow \text{c\_hello}(m_2) & \xrightarrow{m_3} & \\ ik \leftarrow \text{get\_i\_key\_c}(m_3) & & ik \leftarrow \text{get\_i\_key\_s}(m_3) \end{array}$$

(2) Initial Data Exchange

$$\begin{array}{ccc} \text{for each } \alpha \in [\imath] & & \text{for each } \beta \in [\jmath] \\ \mathtt{sqn}_c \leftarrow \alpha + 2 & & \mathtt{sqn}_s \leftarrow \beta + 1 \\ m_4^\alpha \leftarrow \text{pak}(ik, \mathtt{sqn}_c, M_c^\alpha) & & m_5^\beta \leftarrow \text{pak}(ik, \mathtt{sqn}_s, M_s^\beta) \\ m_4 \leftarrow (m_4^1, \cdots, m_4^\imath) & \xrightarrow{m_4} & m_5 \leftarrow (m_5^1, \cdots, m_5^\jmath) \\ \text{process\_packets}(ik, m_5) & \xleftarrow{m_5} & \text{process\_packets}(ik, m_4) \end{array}$$

(3) Key Agreement

$$\begin{array}{ccc} & & \mathtt{sqn}_s \leftarrow 2 + \jmath \\ & \xleftarrow{m_6} & m_6 \leftarrow \text{s\_hello}(m_3, ik, \mathtt{sqn}_s) \\ k \leftarrow \text{get\_key\_c}(m_6, \mathtt{sqn}_s) & & k \leftarrow \text{get\_key\_s}(m_6) \end{array}$$

(4) Data Exchange

$$\begin{array}{ccc} \text{for each } \alpha \in \{\imath + 1, \ldots, u\} & & \text{for each } \beta \in \{\jmath + 1, \ldots, w\} \\ \mathtt{sqn}_c \leftarrow \alpha + 2 & & \mathtt{sqn}_s \leftarrow \beta + 2 \\ m_7^\alpha \leftarrow \text{pak}(k, \mathtt{sqn}_c, M_c^\alpha) & & m_8^\beta \leftarrow \text{pak}(k, \mathtt{sqn}_s, M_s^\beta) \\ m_7 \leftarrow (m_7^{\imath+1}, \cdots, m_7^u) & \xrightarrow{m_7} & m_8 \leftarrow (m_8^{\jmath+1}, \cdots, m_8^w) \\ \text{process\_packets}(k, m_8) & \xleftarrow{m_8} & \text{process\_packets}(k, m_7) \end{array}$$

More about the paper

Proofs, lots of them.

- Replay Attacks
- Packet Manipulation Attacks
- Other...

Replay attacks and QUIC

There are two kinds considered: Server Config Replay Attacks, and Source-Address Token Replay Attacks.

Using a Server Config Replay Attack, the paper explains how they were able to completely prevent QUIC connection establishment as the connection would terminate after half a second, or a timeout of 10 seconds. In order for it to work, the attack had to have a lower transmission delay to the server than the client.

The source-Address Token Replay Attacks, they were able to create an established connection to the server with a spoofed client by highjacking the s_reject messages and sending them off to attempt to establish a connection.

Packet Manipulation attacks and QUIC

The attacker in this case could flip bits and other information in unprotected parameters during a connection that could cause the QUIC connection between the server and the client to fail.

The initial key can’t be known to the adversary so the information being transmitted is safe and satisfies confidentiality, unfortunately integrity is broken as the information could still be tampered with.

Any attempts to sign each packet’s contents for integrity could be then used as an opportunity to launch a DoS attack on the server.

Packet Manipulation attacks and QUIC

Using a connection ID manipulation attack, they were able to successfully lock out connections between a client and server by corrupting the initial key sent so that any attempts to establish a connection failed due to decryption problems.

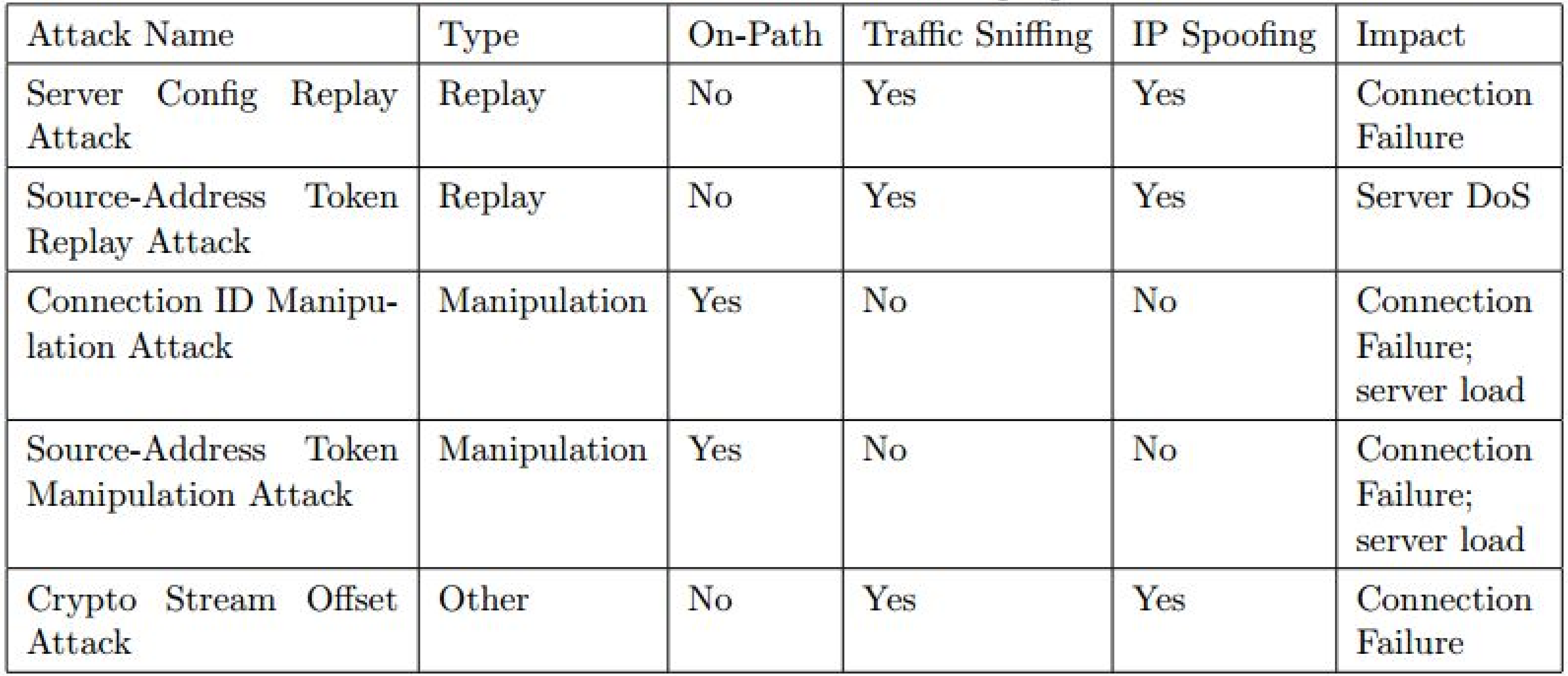
Using a source-address token manipulation attack, they were able to effectively prevent all targeted QUIC connections and all attempts had a 10 second delay before timing out. This would force all connections to resort to TCP/TLS.

Crypto Stream Offset Attack

The initial handshake consists of a byte stream between the server and the client. By injecting erroneous data, the connection can fail to establish using QUIC. The attacker requires very little information to establish this attack as all they need would be the knowledge of when a connection attempt would/is beginning to occur.

This attack could potentially be done completely blind with an optimized C implementation, making it very capable at forcing clients to resort back to TCP/TLS connections instead of QUIC.

Summary of Attack Types



Attack Name	Type	On-Path	Traffic Sniffing	IP Spoofing	Impact
Server Config Replay Attack	Replay	No	Yes	Yes	Connection Failure
Source-Address Token Replay Attack	Replay	No	Yes	Yes	Server DoS
Connection ID Manipulation Attack	Manipulation	Yes	No	No	Connection Failure; server load
Source-Address Token Manipulation Attack	Manipulation	Yes	No	No	Connection Failure; server load
Crypto Stream Offset Attack	Other	No	Yes	Yes	Connection Failure

Are QUIC's tradeoffs worth it?

The tradeoffs happen from how QUIC tries to reduce latency by trading out for security weaknesses.

We can take that at this point in time, you can't have good security and a fast connection.

Regardless of Google's efforts, it seems that there may be fundamental limitations to the QUIC protocol that prevent it from effectively mitigating built-in weaknesses associated with its speed tradeoffs.

Are QUIC’s tradeoffs worth it?

From what was outlined in the paper:

> QUIC may be able to establish a 0-RTT connection (connection re-establishment state), but it would have a difficult time to maintain it due to the presence of attackers. Such attacks could also be used to easily mount a DoS attack and make servers unuseable.
>
> QUIC is fast, but at a certain cost. As long as the host system can resort to using TCP/TLS instead of QUIC in the circumstances it is attacked, at least the user shouldn’t notice any significant disruptions.

Similar up and coming protocol

TLS 1.3 is supposed to be considered a Quick Communication (QC) protocol like QUIC.

Another study would be good to compare these two as so far, TLS 1.3 seems to be implementing a lot of the same features as QUIC!

A newer paper talking about TLS in QUIC can be found here: https://tools.ietf.org/html/draft-thomson-quic-tls-01

Thank You