HTTP/3 Attacks

Yann Chicheportiche
Ariel University
Netanya, Israel
yanoukaxbox@gmail.com

Tal Malka
Ariel University
Ramat Gan, Israel
tal.coder@gmail.com

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

I. Introduction

HTTP, or Hypertext Transfer Protocol, is the main communication protocol used on the internet, allowing information to be exchanged between web browsers and servers. First created in the early 1990s, HTTP has improved over time to handle the growing needs of the web. It started with the basic HTTP/1.0, advanced to the faster and more efficient HTTP/2, and now has reached HTTP/3, which brings even more improvements. Each version has aimed to make the web faster, more secure, and better able to support modern websites and apps. As a key part of how the internet works, HTTP continues to evolve, playing an important role in improving online experiences.

HTTP/1, introduced in the early 1990s, laid the foundation for how web browsers and servers communicate. It was simple and effective for its time, using a request-response model to transfer data. However, it faced limitations as the web grew, such as inefficiency in handling multiple requests. Each file on a webpage—images, stylesheets, or scripts—required a separate connection, which led to delays and increased load times. While HTTP/1.1, an improved version, introduced features like persistent connections to address these issues, the protocol still struggled with modern demands for speed and scalability.

HTTP was originally designed without focusing on security and reliability; this is one of the main motivations behind the development of HTTP/2. HTTP/2, introduced in 2015, was a major step forward in addressing the limitations of HTTP/1. Built on the same basic request-response model, it introduced features like multiplexing, which allowed multiple requests to be handled simultaneously over a single connection. This significantly reduced latency and improved website loading times. HTTP/2 also introduced header compression (Header compression is a technique used to reduce the size of HTTP headers transmitted in requests and responses. HTTP headers carry metadata like content type, language preferences, and cookies, which can be repetitive and bulky, especially in frequent communications. HTTP/2 uses an algorithm called HPACK, which compresses headers by identifying repeated values and replacing them with references to previously sent data. HTTP/3 employs an upgraded version called QPACK, designed to handle the unique characteristics of QUIC, such as packet loss, without compromising efficiency. This reduces bandwidth usage and speeds up data transfers, making it especially beneficial for networks with high latency or limited capacity.) and prioritized requests, making it much more efficient for modern web applications. However, despite its improvements, HTTP/2 still relied on TCP (Transmission Control Protocol), which brought challenges like head-of-line blocking (Head-of-Line Blocking: Head-of-line blocking occurs when the processing of a request or response is delayed because another item at the front of the queue must be handled first. In HTTP/2, even though multiple requests can be multiplexed over a single TCP connection, a problem (like packet loss) affects all streams on that connection, slowing down the delivery of other requests. HTTP/3 solves this issue with QUIC, where each stream operates independently. A packet loss in one stream does not delay the others.) , where delays in one request could impact others.

HTTP/3 represents the latest evolution of the protocol, designed to address the limitations of HTTP/2. Built on QUIC, a transport protocol that integrates reliable data transfer and encryption, HTTP/3 enables faster connection setup and improved performance, particularly in high-latency or packet loss-prone networks. Its features, such as connection migration, multiplexing without head-of-line blocking, and enhanced security, make it a suitable solution for the modern web. As it gains adoption across browsers and servers, HTTP/3 is set to become the new standard, ensuring better user experiences and greater efficiency for web applications.

QUIC (Quick UDP Internet Connections) is a modern transport protocol developed by Google to overcome the limitations of traditional protocols like TCP. Based on UDP, QUIC combines reliable data transfer and encryption into its core, optimizing communication while enhancing security. Key features include faster connection establishment with 0-RTT (The 0-RTT feature in QUIC allows a client to send application data before the handshake is complete. This is made possible by reusing negotiated parameters from a previous connection. To enable this, 0-RTT depends on the client remembering critical parameters and providing the server with a TLS session ticket that allows the server to recover the same information.) and 1-RTT handshakes, multiplexing without head-of-line blocking, and seamless connection migration between network interfaces, such as switching from Wi-Fi to mobile data. By reducing latency and improving resilience

to packet loss, QUIC meets the demands of modern internet applications. Adopted as the foundation for HTTP/3, QUIC is becoming a crucial component of the web, enabling faster and more reliable online experiences.

As an entirely new protocol for the web developed on top of UDP, It can be seen that the QUIC protocol has a unique performance advantage, but whenever it is mentioned, we have to think of its underlying layer. UDP only provides a simple and unreliable messaging service, so it has many serious security issues. New attempts at the architecture of the QUIC protocol are bound to lead to some new challenges. Therefore, before putting QUIC into practical applications [5], we must consider its security issues.

Common threats include:

Attack Name	**Description**
Denial of Service (DoS)	Attackers overwhelm a server with excessive requests, causing the server to become unresponsive or crash, disrupting service for legitimate users.
Slow-rate DoS	Involves sending a steady stream of slow or low-volume requests to gradually deplete the server's resources over time, without triggering immediate traffic spikes, leading to resource exhaustion.
Smuggling	Exploits desynchronization between the server and client to inject malicious data into the communication, bypassing security controls and potentially compromising the system.
Privacy Concerns	May arise if attackers break multiplexing and isolate or track individual user sessions, thereby violating user privacy and enabling unauthorized data collection.
Downgrade	Forces connections to revert to older, less secure protocol versions, exposing sensitive data to potential interception or manipulation.
Replay	Occurs when an attacker intercepts and replays previously valid data packets to trick the server or client into executing actions that were not originally intended, such as unauthorized access or transactions.
Man-in-the-Middle (MITM)	Involves an attacker secretly intercepting, monitoring, or altering communication between the client and server, allowing them to steal, modify, or inject malicious content into the data being transmitted.
DNS Spoofing / Cache Poisoning	Involves manipulating DNS records or poisoning the DNS cache to redirect traffic to malicious websites, potentially exposing users to phishing or malware attacks.

TABLE I
DESCRIPTIONS OF HTTP/3 ATTACKS

II. RELATED WORK

add content