Security Challenges in Network Communication Caused by the Quic Protocol

K. Josić, S. Papić
Algebra University Zagreb, Croatia
karlo.josic@algebra.hr, silvio.papic@algebra.hr

***Abstract* - *Quic* (*Quic*k UDP Internet Connections) emerges as an innovative protocol aimed at enhancing the speed and security of HTTP (Hypertext Transfer Protocol) traffic, potentially superseding TCP (Transmission Control Protocol) based TLS (Transport Layer Security) in web applications. While some web browsers have incorporated it as a default setting, its adoption by various websites is on the rise. A significant challenge arises as network security devices categorize *Quic* traffic merely as generic UDP (User Datagram Protocol) rather than distinct web traffic. Consequently, conventional web filter mechanisms fail to scrutinize or log *Quic* traffic, leading to potential vulnerabilities by allowing access to prohibited or malicious sites. This study delves into the functionality of *Quic*, and its implications for network security, and offers insights from leading firewall vendors on addressing the nuances of *Quic* protocol inspection. Furthermore, our evaluation of a FortiGate virtual appliance underscores that the existing web filter engine remains ineffectual in inspecting, logging, or reporting *Quic* web traffic.**

Keywords – Quic Protocol, Web Traffic Filtering, Firewall Inspection, Network Security, HTTPv3

I. Introduction

Many application protocols that have made the Internet popular, such as HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), etc. use TCP (Transmission Control Protocol) on the transport layer but not every application or service benefits from using TCP, especially nowadays when many users are using wireless networks [1]. Wireless networks are unreliable and unpredictable, and this is what is affecting user experience today. For that reason, Google designed a new protocol *Google Quic* (short *gQuic*), that provides several improvements to make the web more efficient and secure. After years of experimentation, the *Quic* protocol was adopted by the IETF (Internet Engineering Task Force) in 2018. for standardization. The *Quic* protocol is used by 7.5% of all websites and usage is growing every day [2]. While the HTTP protocol traditionally utilizes TCP at the transport layer in versions 1.x and 2, the latest iteration, HTTP version 3, adopts the *Quic* protocol, which is constructed on top of UDP. The existing security infrastructure, including network intrusion detection systems and firewalls, is predominantly designed to counter security threats targeting the TCP protocol, such as *SYN flood* attacks, *TCP session hijacking* attacks, and *TCP reset* attacks [3,4]. Introducing the UDP protocol at the transport layer for reliable services brings forth a previously unexplored realm of potential attacks. In this paper, we will elucidate some of the key attributes of the *Quic* protocol and its implications for the web filtering engine of firewalls. In this paper, the authors present the *Quic* protocol and the idea behind using *Quic* for HTTP traffic and compare HTTP over *Quic* with HTTP over TCP/TLS protocols. Also, it will be shown how Quic is causing some security issues in web filtering on the firewall which could be of critical importance in certain scenarios, and after a short reference to what vendors say about *Quic*, the conclusion will be given.

II. About *Quic*

Quic was originally designed by Google engineers in 2012. known as *Google Quic* or *gQuic*. After extensive experimentation over the years, Google has invested significant efforts in enhancing web efficiency, leading to its adoption for standardization by the IETF. IETF *Quic* or just *Quic* has improved the original *gQuic* design, so today we can consider it an almost separate protocol. The *Quic* protocol enhances web communications by integrating security directly into HTTPv3, offering significant improvements in speed and efficiency. By streamlining connection setups and supporting seamless IP transitions, *Quic* optimizes performance, especially on mobile networks. Its role as the foundation of HTTPv3 eliminates common data transmission bottlenecks, promising a future of faster and more secure Internet browsing. As adoption increases, the *Quic* protocol is going to transform the landscape of digital communication. Because of its potential significance in the development of Internet communication *Quic* protocol is important in the context of network security and potential misuse by attackers, which is the focus of this paper. The IETF *Quic* working group outlined five essential objectives that *Quic* should achieve [5]:

- Ensure the security of the payload using TLS (Transport Layer Security) 1.3.
- Facilitate deployment without necessitating alterations to network equipment.
- Implement multiplexing with significantly reduced *HoL* (head-of-line) blocking normally caused when multiple objects are requested, and a small object gets stuck because a preceding large object got delayed.
- Reduce the connection establishment and transport latency.

- Support extensions for forward error correction and multipath connections.

Built upon UDP, *Quic* can be seamlessly incorporated into end-host applications. Presently, there is a client-side implementation embedded in Chromium and Android, employed when accessing server-side applications that support it, such as YouTube, Google Drive, and Facebook. The most important difference between *Quic* and TCP is that the *Quic* protocol provides authentication and encryption by itself. TCP, on the other hand, is a connection-oriented protocol without any security mechanism. Applications that use the TCP protocol on the transport layer should use a higher-layer protocol, such as TLS (Secure Sockets Layer), to achieve authentication and encryption. *Quic* aims to encrypt a comprehensive range of data, encompassing signaling information, to obscure it from network equipment. This strategy aims to thwart firewall vendors from making assumptions that could impede or obstruct future modifications to the protocol [5]. The IETF *Quic* protocol is a connection-oriented protocol, like TCP, using a three-way handshake to establish an end-to-end connection. To ensure authentication and negotiation of cryptographic parameters, *Quic* uses TLS 1.3 handshake messages but replaces the TLS record layer with its framing format. As a result, the process of establishing an initial *Quic* connection is much faster compared to the process of establishing a TCP/TLS connection. As shown in Fig. 1 the standard *Quic* handshake completes in just one round-trip between the client and server, as opposed to the two round-trips needed for the combined TCP and TLS 1.3 handshakes [6].

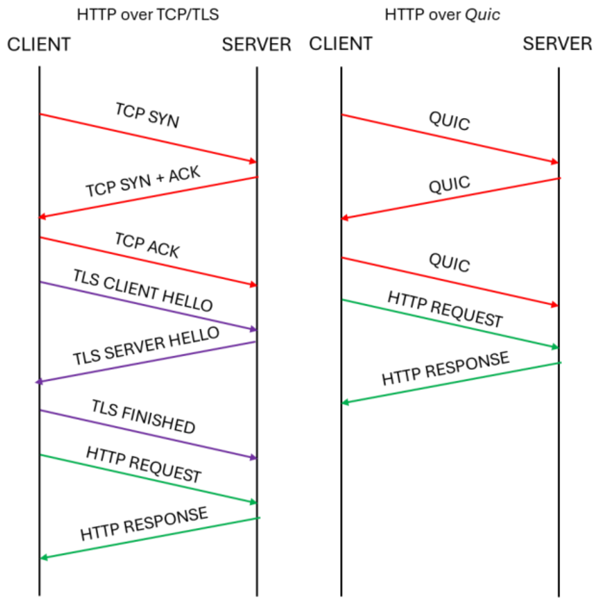


Fig. 1. HTTP over TCP/TLS vs. HTTP over *Quic*

III. TLS Inspection and Web Filtering

A web filter is a network security feature in firewalls used for preventing the viewing of inappropriate material, preserving employee productivity, preventing network congestion, reducing exposure to web-based threats, etc. Web filter allows a network administrator to either monitor or restrict access to the websites and may include report functionality. Web filters are built to analyze HTTP traffic on TCP ports 80 and 443 on the transport layer. Most of today's web traffic is encrypted by TLS protocol, therefore the firewall needs to decrypt it so that traffic can be analyzed. For this purpose, the firewall performs TLS inspection acting like a man-in-the-middle attack (MITM). When the client starts the session, the firewall intercepts the TCP connection, generating SYN-ACK for the client, and completes a TCP/TLS handshake process with the client. The firewall then creates another TCP connection and completes a TCP/TLS handshake process with the Web server, Fig 2. When a client starts a session, the firewall doing TLS inspection is not the destination IP address, so the client must trust the firewall's certificate for this communication to take place. To make TLS inspection transparent to the client, it is necessary to install a firewall certificate in the trusted authorities list of the client's web browser.

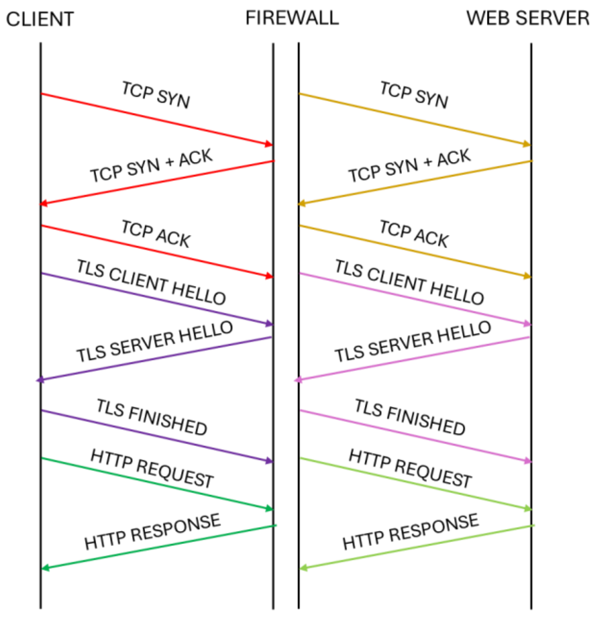


Fig. 2. TLS inspection

Today's modern next-generation firewalls and web filters support several inspection features, such as static URL filtering or website category filtering. Before continuing transmission, the firewall will buffer transmitted files for inspection and parsing HTTP headers. If we enable both filters, static URL filtering will be performed first, followed by web page category filtering. If a web filter blocks the connection, the firewall will send a notification to the client browser. Also, the network security administrator can configure it. a URL exception to bypass HTTP inspection, Fig. 2. It is also possible to use the exception for trusted websites, bank payments, etc.

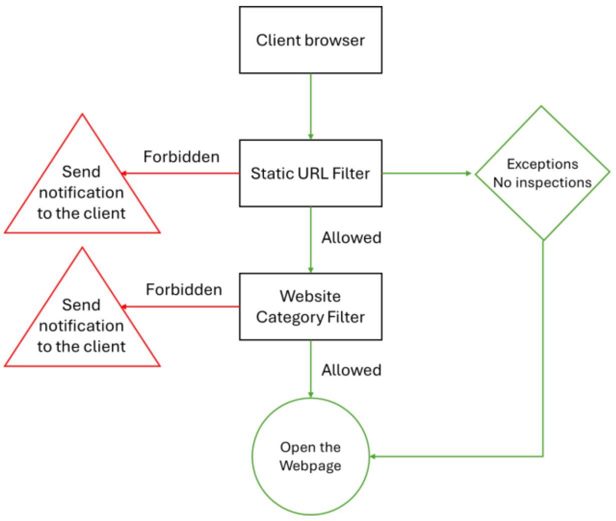
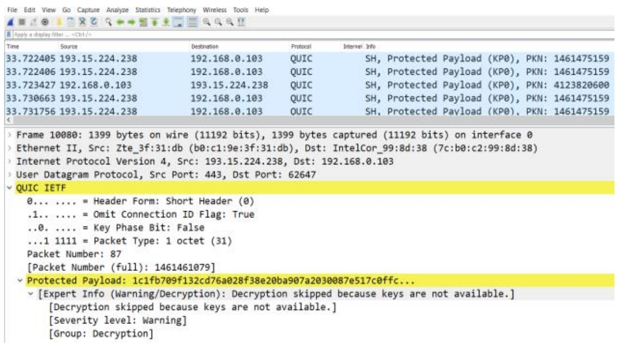


Fig. 2. HTTP traffic inspection flow

IV. WEB FILTERING ISSUES CAUSED BY THE *QUIC* PROTOCOL

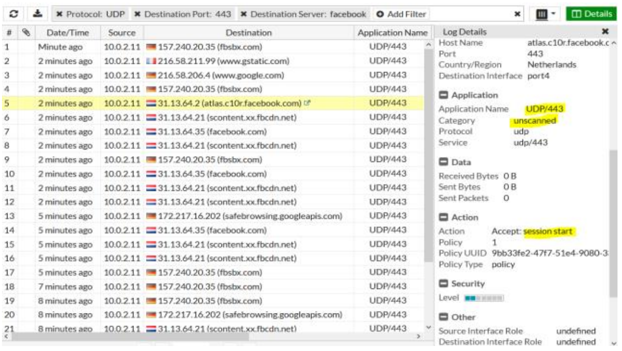
The main security issue with the *Quic* protocol is that it is not supported by firewalls yet. The *Quic* uses UDP port 443 for transport web traffic instead of traditional TCP port 443. A web filtering engine is built to intercept TCP connections and interpret web traffic from the transport layer up to the application layer. Browsers and web servers that support the *Quic* protocol can process it as web traffic, but the firewalls between them cannot inspect the *Quic* protocol and they are treating *Quic* as a generic layer 4 UDP traffic [7]. Therefore, *Quic* traffic bypasses the network filter and opens huge security issues for network security administrators. The opening of banned or malicious websites is not prevented anymore. For this paper, we did *Quic* protocol tests on Fortinet's virtual device Fortigate. On the client side, we used two different web browsers that support *Quic* protocol, Chrome Canary and Firefox Nightly, and we generated web traffic to servers that also support *Quic*, like YouTube, Gmail, Facebook, etc. We analyzed *Quic* web traffic using Wireshark on the client and logs on the FortiGate firewall. From Wireshark's output, we can see that *Quic* is on top of UDP, *Quic* traffic is encrypted, and Wireshark cannot see the *Quic* protocol deeper to detect information from the upper layers, Fig. 3.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
Time      Source          Destination     Protocol  Internet Info
33.722405 193.15.224.238  192.168.0.103   QUIC      SH, Protected Payload (KP0), PKN: 1461475159
33.722406 193.15.224.238  192.168.0.103   QUIC      SH, Protected Payload (KP0), PKN: 1461475159
33.723427 192.168.0.103   193.15.224.238  QUIC      SH, Protected Payload (KP0), PKN: 4123820600
33.730663 193.15.224.238  192.168.0.103   QUIC      SH, Protected Payload (KP0), PKN: 1461475159
33.731756 193.15.224.238  192.168.0.103   QUIC      SH, Protected Payload (KP0), PKN: 1461475159

> Frame 10080: 1399 bytes on wire (11192 bits), 1399 bytes captured (11192 bits) on interface 0
> Ethernet II, Src: Zte_3f:31:db (b0:c1:9e:3f:31:db), Dst: IntelCor_99:8d:38 (7c:b0:c2:99:8d:38)
> Internet Protocol Version 4, Src: 193.15.224.238, Dst: 192.168.0.103
> User Datagram Protocol, Src Port: 443, Dst Port: 62647
v QUIC IETF
    0... .... = Header Form: Short Header (0)
    .1.. .... = Omit Connection ID Flag: True
    ..0. .... = Key Phase Bit: False
    ...1 1111 = Packet Type: 1 octet (31)
    Packet Number: 87
    [Packet Number (full): 1461461079]
  v Protected Payload: 1c1fb709f132cd76a028f38e20ba907a2030087e517c0ffc...
    v [Expert Info (Warning/Decryption): Decryption skipped because keys are not available.]
        [Decryption skipped because keys are not available.]
        [Severity level: Warning]
        [Group: Decryption]
```

Fig. 3. FortiGate *Quic* log output

On the firewall side, for this paper, a web filter was configured that blocks the social networking category. As we can see from FortiGate's logs, social networking traffic passed through the firewall security inspection engine as unscanned and uncategorized UDP traffic on port 443. Therefore, we do not have all the detailed data in the log that we would get with HTTP traffic, such as a web category or URL description, Fig. 4.

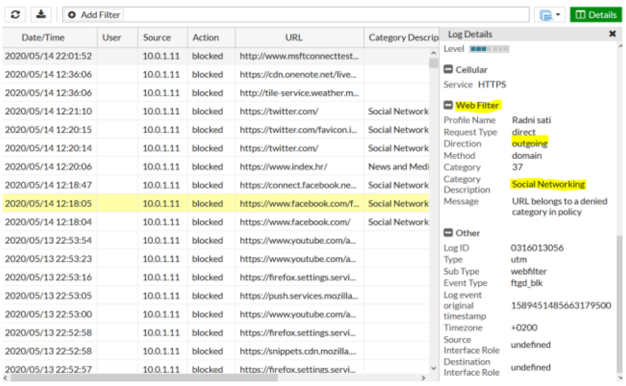


#	Date/Time	Source	Destination	Application Name
1	Minute ago	10.0.2.11	157.240.20.35 (fbsbx.com)	UDP/443
2	2 minutes ago	10.0.2.11	216.58.211.99 (www.gstatic.com)	UDP/443
3	2 minutes ago	10.0.2.11	216.58.206.4 (www.google.com)	UDP/443
4	2 minutes ago	10.0.2.11	157.240.20.35 (fbsbx.com)	UDP/443
5	2 minutes ago	10.0.2.11	31.13.64.2 (atlas.c10r.facebook.com)	UDP/443
6	2 minutes ago	10.0.2.11	31.13.64.21 (scontent.xx.fbcdn.net)	UDP/443
7	2 minutes ago	10.0.2.11	31.13.64.35 (facebook.com)	UDP/443
8	2 minutes ago	10.0.2.11	31.13.64.21 (scontent.xx.fbcdn.net)	UDP/443
9	2 minutes ago	10.0.2.11	157.240.20.35 (fbsbx.com)	UDP/443
10	2 minutes ago	10.0.2.11	31.13.64.35 (facebook.com)	UDP/443
11	2 minutes ago	10.0.2.11	31.13.64.21 (scontent.xx.fbcdn.net)	UDP/443
12	2 minutes ago	10.0.2.11	31.13.64.21 (scontent.xx.fbcdn.net)	UDP/443
13	5 minutes ago	10.0.2.11	172.217.16.202 (safebrowsing.googleapis.com)	UDP/443
14	5 minutes ago	10.0.2.11	31.13.64.35 (facebook.com)	UDP/443
15	5 minutes ago	10.0.2.11	31.13.64.21 (scontent.xx.fbcdn.net)	UDP/443
16	5 minutes ago	10.0.2.11	31.13.64.21 (scontent.xx.fbcdn.net)	UDP/443
17	5 minutes ago	10.0.2.11	157.240.20.35 (fbsbx.com)	UDP/443
18	7 minutes ago	10.0.2.11	157.240.20.35 (fbsbx.com)	UDP/443
19	8 minutes ago	10.0.2.11	157.240.20.35 (fbsbx.com)	UDP/443
20	8 minutes ago	10.0.2.11	172.217.16.202 (safebrowsing.googleapis.com)	UDP/443
21	8 minutes ago	10.0.2.11	31.13.64.21 (scontent.xx.fbcdn.net)	UDP/443

```
Log Details
Host Name                    atlas.c10r.facebook.c
Port                         443
Country/Region               Netherlands
Destination Interface        port4

Application
Application Name        UDP/443
Category             unscanned
Protocol             udp
Service              udp/443

Data
Received Bytes  0 B
Sent Bytes      0 B
Sent Packets    0

Action
Action          Accept: session start
Policy          1
Policy UUID     9bb33fe2-47f7-51e4-9080-3
Policy Type     policy

Security
Level

Other
Source Interface Role           undefined
Destination Interface Role      undefined
```

Fig. 4. FortiGate *Quic* log output

If the web browser and web server cannot connect via the *Quic* protocol, they will try to establish a connection using HTTP/HTTPS over TCP. Network security administrators can use this functionality to block UDP port 443 and force end devices to use traditional HTTP/HTTPS over TCP methods. In this way, the web filter engine can inspect traffic, record logs correctly, and categorize websites, Fig. 5.



Date/Time	User	Source	Action	URL	Category Descrip
2020/05/14 22:01:52		10.0.1.11	blocked	http://www.msftconnecttest...	
2020/05/14 12:36:06		10.0.1.11	blocked	https://cdn.onenote.net/live...	
2020/05/14 12:36:06		10.0.1.11	blocked	http://tile-service.weather.m...	
2020/05/14 12:21:10		10.0.1.11	blocked	https://twitter.com/	Social Network
2020/05/14 12:20:15		10.0.1.11	blocked	https://twitter.com/favicon.i...	Social Network
2020/05/14 12:20:14		10.0.1.11	blocked	https://twitter.com/	Social Network
2020/05/14 12:20:06		10.0.1.11	blocked	https://www.index.hr/	News and Medi
2020/05/14 12:18:47		10.0.1.11	blocked	https://connect.facebook.ne...	Social Network
2020/05/14 12:18:05		10.0.1.11	blocked	https://www.facebook.com/f...	Social Network
2020/05/14 12:18:04		10.0.1.11	blocked	https://www.facebook.com/	Social Network
2020/05/13 22:53:54		10.0.1.11	blocked	https://www.youtube.com/a...	
2020/05/13 22:53:23		10.0.1.11	blocked	https://www.youtube.com/a...	
2020/05/13 22:53:16		10.0.1.11	blocked	https://firefox.settings.servi...	
2020/05/13 22:53:05		10.0.1.11	blocked	https://push.services.mozilla...	
2020/05/13 22:53:00		10.0.1.11	blocked	https://www.youtube.com/a...	
2020/05/13 22:52:58		10.0.1.11	blocked	https://firefox.settings.servi...	
2020/05/13 22:52:58		10.0.1.11	blocked	https://snippets.cdn.mozilla....	
2020/05/13 22:52:57		10.0.1.11	blocked	https://firefox.settings.servi...	

```
Log Details
Level

Cellular
Service  HTTPS

Web Filter
Profile Name        Radni sati
Request Type        direct
Direction           outgoing
Method              domain
Category            37
Category
Description         Social Networking
Message             URL belongs to a denied
                    category in policy

Other
Log ID              0316013056
Type                utm
Sub Type            webfilter
Event Type          ftgd_blk
Log event
original            1589451485663179500
timestamp
Timezone            +0200
Source
Interface Role      undefined
Destination
Interface Role      undefined
```

Fig. 5. WebFilter blocked *HTTP over TCP* traffic by forbidden web category

V. WHAT THE FIREWALL VENDORS SAY ABOUT *QUIC*?

Most firewall vendors suggest blocking *Quic* because the protocol is not supported yet. We can do this in two ways, the first way is to block all UDP traffic on port 443 on the firewall, or otherwise, we can disable the *Quic* protocol in the client browser. There is another issue with advice to block UDP 443. Some protocols like OpenVPN can use either UDP or TCP as the transport, therefore it is important to verify that blocking of all UDP traffic on port 443 will not create unwanted communication issues. Here are some recommendations about *Quic* of the leading firewall vendors:

- Fortinet notes that the *Quic* protocol is currently in an experimental stage, leading to non-support and encountering problems when using TLS inspection to block certain websites hosted by Google. [8].

- Palo Alto suggests establishing a security policy to restrict the *Quic* protocol, compelling web browsers to employ traditional TLS/TLS. This is advised to maintain visibility and control over Google applications, as whitelisting the *Quic* protocol results in a loss of such capabilities. [9].

- Sophos proposes analogous approaches to prevent the *Quic* protocol from evading the firewall's web filters. These methods include disabling *Quic* in the browser on the client machine, using application control engine-based blocking, or firewall rule-based blocking to halt UDP port 443.[10].

VI. Conclusion

Quic protocol provides several improvements designed to accelerate HTTP traffic and make it more secure, replacing TCP and TLS on the web. It changes the communication pattern on the Internet using UDP instead of traditional TCP at the transport layer. *Quic* is still in the development phase and the leading firewall manufacturers are still not fully supporting it. As we showed in this paper, TLS inspection, and web filter engines cannot inspect and log *Quic* protocol, so banned and malicious websites traffic pass through the firewall without any restrictions. The only solution given to us, by firewall manufacturers, is recommendations on how to block the *Quic* protocol. However, some of these recommendations could lead to another security issue, such as making a rule to block UDP port 443, which can also block legitimate traffic of another protocol like *OpenVPN*. Using UDP protocol at the transport layer for reliable services could open numerous unexplored attacks and that is why more attention should be given to this topic. To expand upon this conclusion there are also other important reasons to consider *Quic* protocol in the context of security. Primarily the fact that security is a fundamental aspect of *Quic* protocol, not an optional one like with TCP/TLS, also with built-in security *Quic* offers improved performance with reduced time for establishing a connection in comparison to TCP/TLS which could be especially valuable characteristics, especially for mobile networks and devices where latency is a significant concern. Furthermore, one important aspect of *Quic* is that because of its design, it can alleviate symptoms of common web attacks such as TCP SYN floods and other TCP attacks which is an argument for wide usage of *Quic* rather than TCP/TLS. Also, a very important characteristic of *Quic* is that it is designed to be extensible, allowing future upgrades and features to be added without requiring changes to network infrastructure. In summary, the *Quic* protocol represents a significant step forward in the quest for a faster and more secure internet. By addressing several limitations of the traditional TCP/TLS stack and incorporating advanced security features directly into the protocol, *Quic* not only enhances web performance but also strengthens the security of web communications against modern threats. Because of all the above, it is crucial to recognize that *Quic* is here to stay and that security issues like the one described in this paper need to be taken seriously so that we can have overall more secure Internet communication.

References

[1] K. Edeline, M. Kühlewind, B. Trammell, and B. Donnet, “copycat,” in *Proceedings of the Applied Networking Research Workshop on - ANRW ’17*, 2017, vol. 7, pp. 13–19, doi: 10.1145/3106328.3106330.

[2] “Usage Statistics of *QUIC* for Websites", Sep 2023. [Online]. Available: https://w3techs.com/technologies/details/ce-*Quic*.

[3] U. Chandravadan Shah, “Flow-based Analysis of *QUIC* Protocol.”, 2018. [Online]. Available: https://is.muni.cz/th/q69ug/.

[4] D. Wenilang, *Computer Security: A Hands-on Approach*. CreateSpace Independent Publishing Platform, 2017. ISBN: 9781733003957

[5] M. Burghard and B. Jaeger, “How Good Is *QUIC* Actually?,” doi: 10.2313/NET-2019-10-1_06.

[6] “The Road to *QUIC*.” [Online]. Available: https://blog.cloudflare.com/the-road-to-*Quic*/.

[7] “How Google’s *QUIC* Protocol Impacts Network Security and Reporting.” [Online]. Available: https://www.fastvue.co/fastvue/blog/googles-*Quic*-protocols-security-and-reporting-implications/.

[8] “Technical Note: Disabling / Blocking *QUIC* Protocol to force Google Chrome to use TLS.” [Online]. Available: https://kb.fortinet.com/kb/viewContent.do?externalId=FD36529.

[9] “How to Block *QUIC* Protocol - Knowledge Base - Palo Alto Networks.” [Online]. Available: https://knowledgebase.paloaltonetworks.com/KCSArticleDetail?id=kA10g000000ClarCAC.

[10] “Advisory: Google’s *QUIC* protocol bypasses scanning on Sophos XG Firewall - Sophos Community.” [Online]. Available: https://community.sophos.com/kb/en-us/127719#How to block *QUIC* Protocol with Firewall Rules.