EPIC-ARES 2023, September 01, 2023

Stefan Tatschner
Sebastian N. Peters
David Emeis
John Morris
Thomas Newe

Fraunhofer AISEC

Fraunhofer Institute for Applied and Integrated Security AISEC

A Quic(k) Security Overview: A Literature Research on Implemented Security Recommendations

Overview

Meta
About Me

Google Search: “Homer Donut”

G92P+P4 Berlin, Germany

- **Stefan Tatschner**
- Security Researcher

Github	`@rumpelsepp`
Matrix	`@rumpelsepp:hackbrettl.de`
Mastodon	`@rumpelsepp@mastodon.social`
LinkedIn	`in/stefan-tatschner`

Meta
About QUIC

QUIC: A UDP-Based Multiplexed and Secure Transport

QUIC provides applications with flow-controlled streams for structured communication, low-latency connection establishment, and network path migration. QUIC includes security measures that ensure confidentiality, integrity, and availability in a range of deployment circumstances.

`https://datatracker.ietf.org/doc/html/rfc9000`

Meta
About RFCs

Request for Comments

A Request for Comments (RFC) is a publication in a series from the principal technical development and standards-setting bodies for the Internet, most prominently the Internet Engineering Task Force (IETF).

`https://en.wikipedia.org/wiki/Request_for_Comments`

Meta
About RFCs

Security Consideration

In RFC documentation, sections called “Security Consideration” are used to discuss conceptional weaknesses in published network protocols according to the attacker model explained in RFC3552. If necessary, follow-up RFC documents are published to keep the older documents up-to-date.

Meta
About RFCs

RFC2119: Key words for use in RFCs to Indicate Requirement Levels

- **Requirement:** *MUST, MUST NOT*
- **Recommendation:** *SHOULD, SHOULD NOT*
- **Optional***: MAY*

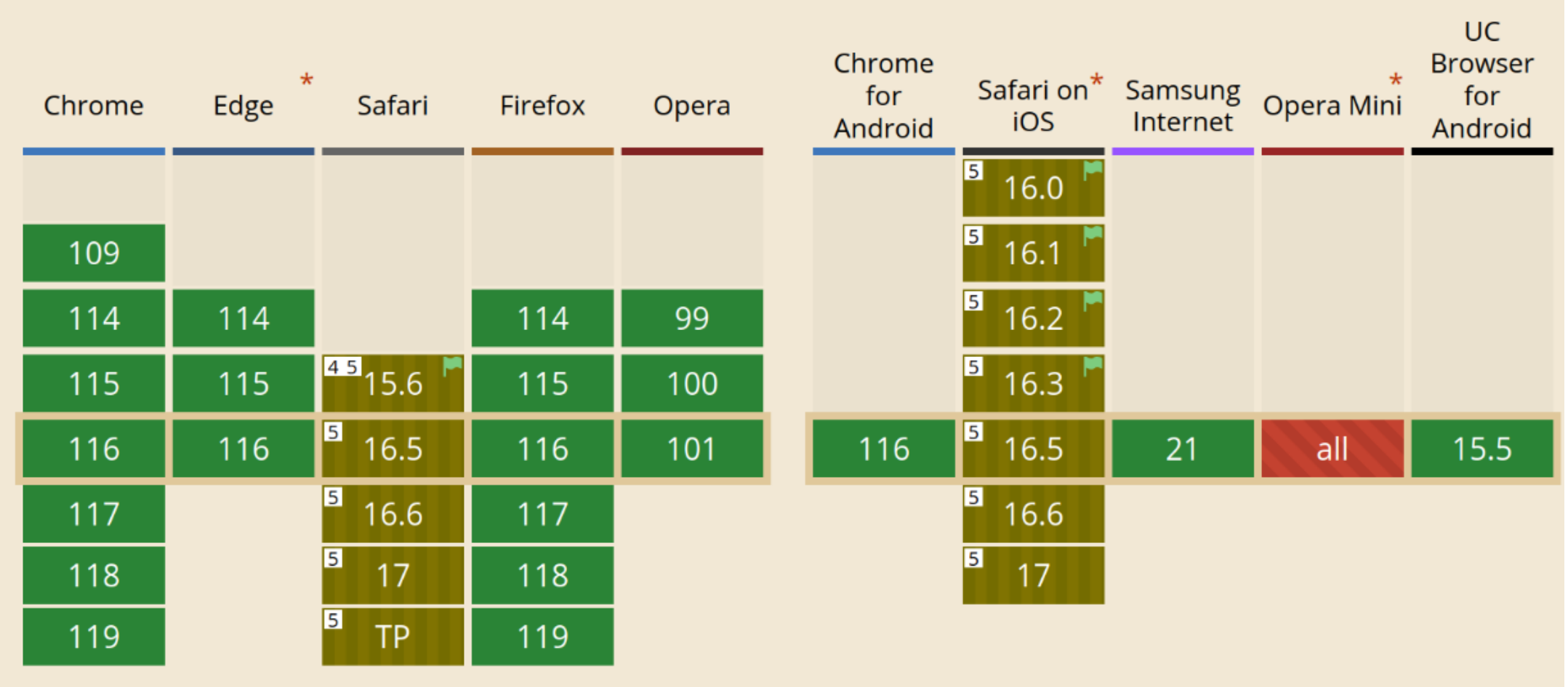
Introduction

Introduction

QUIC: A UDP-Based Multiplexed and Secure Transport

- **designed by Google (2012)**
- **submitted to IETF (2015):** published as RFC9000 (2021)
- **overcomes multiple limitations in TCP+TLS:** nickname TCP/2
- **HTTP/3 is based on QUIC:** highly integrated in the modern web
- **located in user space:** enables rapid iterations of the protocol

Introduction
HTTP/3 and QUIC Support



https://caniuse.com/http3

located in user space:

enables rapid iterations of the protocol

Introduction
Research Question

- **QUIC:** 13 popular production ready implementations*
- **RFC 9000:** 115 pages
- **RFC 9000:** 10 Optional Security Considerations
- **RFC 9000:** half of the RFC2119 keywords: SHOULD or SHOULD NOT

RQ: *Does the complexity of the RFC standard on the one hand and the large number of different QUIC implementations on the other hand lead to an insecure QUIC ecosystem?*

*: We found 29, but we filtered...

Related Work

Related Work

Key Takeaways

analyzed six papers from 2005 — 2023

- misbehaving receivers can cause major impact on infrastructures
- differences in low-level design choices between QUIC implementations → *half of the keywords are SHOULD/SHOULD NOT*
- no setup available to compare those at runtime
- fragmented ecosystem which is not yet mature enough

Methodology

Methodology

1. **Identification of relevant RFC documents:** RFC 9000 ✓
2. **Analysis of “Security Considerations”:** ten sections ✓
3. **Online search for QUIC implementations:** 13 implementations ✓
4. **Evaluation of implemented measures:** next slide…

Results

Security Consideration	*quic-go*	*c/quiche*	*MsQuic*	*Quinn*	*Neqo*	*XQUIC**	*mvfst**	*aioquic*	*LSQUIC*	*ngtcp2*	*s2n-quic*	*quicly*	*g/quiche**
Amplification (cf. Sec. 4.1)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Optimistic ACK (cf. Sec. 4.2)	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗
Slowloris (cf. Sec. 4.3)	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
Stream Fragmentation and Reassembly (cf. Sec. 4.4)	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Stream Commitment (cf. Sec. 4.5)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Peer Denial of Service (cf. Sec. 4.6)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	[✓]	✗	✗
Explicit Congestion Notification (cf. Sec. 4.7)	n/a	n/a	✓	✓	n/a	n/a	n/a	n/a	n/a	✓	✓	n/a	✓
Stateless Reset Oracle (cf. Sec. 4.8)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Version Downgrade (cf. Sec. 4.9)	✓	✗	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✓
Traffic Analysis (cf. Sec. 4.10)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Table 2: Evaluation Results. ✓ means the mitigation is available as defined in Section 4. [✓] means that a special case of this mitigation is implemented. ✗ means the mitigation is not available. The abbreviation n/a indicates that a required underlying feature is not available. Projects with a star * seem to be developed behind closed doors and the results need to be taken with a grain of salt. The full version of this table is available in the supplementary material.

21.6 Slowloris Attacks

The attacks commonly known as Slowloris [SLOWLORIS] try to keep many connections to the target endpoint open and hold them open as long as possible. These attacks can be executed against a QUIC endpoint by generating the minimum amount of activity necessary to avoid being closed for inactivity. This might involve sending small amounts of data, gradually opening flow control windows in order to control the sender rate, or manufacturing ACK frames that simulate a high loss rate.

QUIC deployments SHOULD provide mitigations for the Slowloris attacks, such as increasing the maximum number of clients the server will allow, limiting the number of connections a single IP address is allowed to make, imposing restrictions on the minimum transfer speed a connection is allowed to have, and restricting the length of time an endpoint is allowed to stay connected.

Requirements

Section	Requirement	Status	Text
21.6	SHOULD	Missing test	QUIC deployments SHOULD provide mitigations for the Slowloris attacks, such as increasing the maximum number of clients the server will allow, limiting the number of connections a single IP address is allowed to make, imposing restrictions on the minimum

Discussion

Discussion

- approach is sensitive to **human error** and **not trivial to repeat**
- implementations of certain SCs might overlap
- some SCs are not specific enough
- governance differs strongly (big players maintain their own stack):
  - **very open:** quic-go, quiche, s2n-quic
  - **very closed:** mvfst, google/quiche, XQUIC

Discussion

RQ: *Does the complexity of the RFC standard on the one hand and the large number of different QUIC implementations on the other hand lead to an insecure QUIC ecosystem?*

→ empatic **yes**

Discussion

we propose:

- include QUIC in major, open software stacks (e.g. stdlibs, kernels, …)
- work on a test setup for comparing implementations at runtime
- better track status of RFC2119 keywords

Meta
Contact

G92P+P4 Berlin, Germany

Thank you!

https://github.com/Fraunhofer-AISEC/quic-overview

Stefan Tatschner

Github @rumpelsepp
Matrix @rumpelsepp:hackbrettl.de
Mastodon @rumpelsepp@mastodon.social
LinkedIn in/stefan-tatschner

https://www.aisec.fraunhofer.de/