

BLUEHAT IL

Maor Abutbul

*Why-So-QUIC!? Racing and Fuzzing HTTP/3
using QuicDraw-UI*



BLUEHAT IL

Maor Abutbul

*Why-So-QUIC!? Racing and Fuzzing HTTP/3
using QuicDraw-UI*

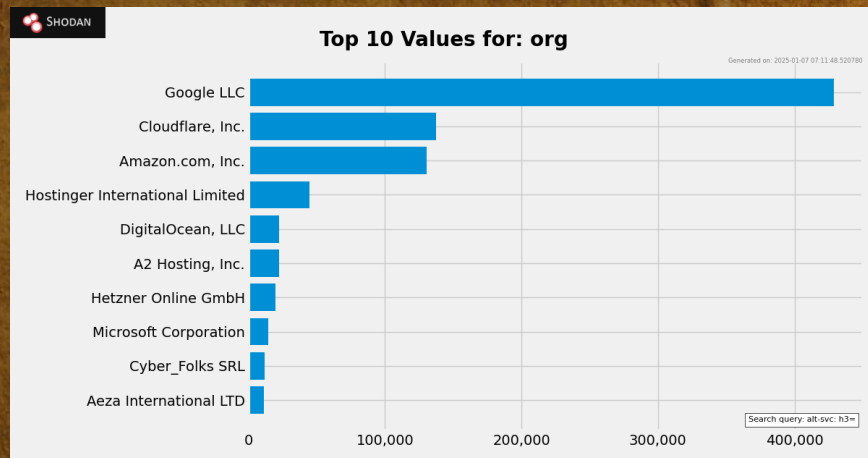


WHY HTTP/3?

UDP

HTTP/3? - Internet-wide support

- 39.8% (w3techs) of Internet Facing
 - ~13M+ HTTP3 Live Websites
- Major Browsers
- Major CDNs & Cloud Providers
- Major Websites



Alt-svc by organization - CDN's



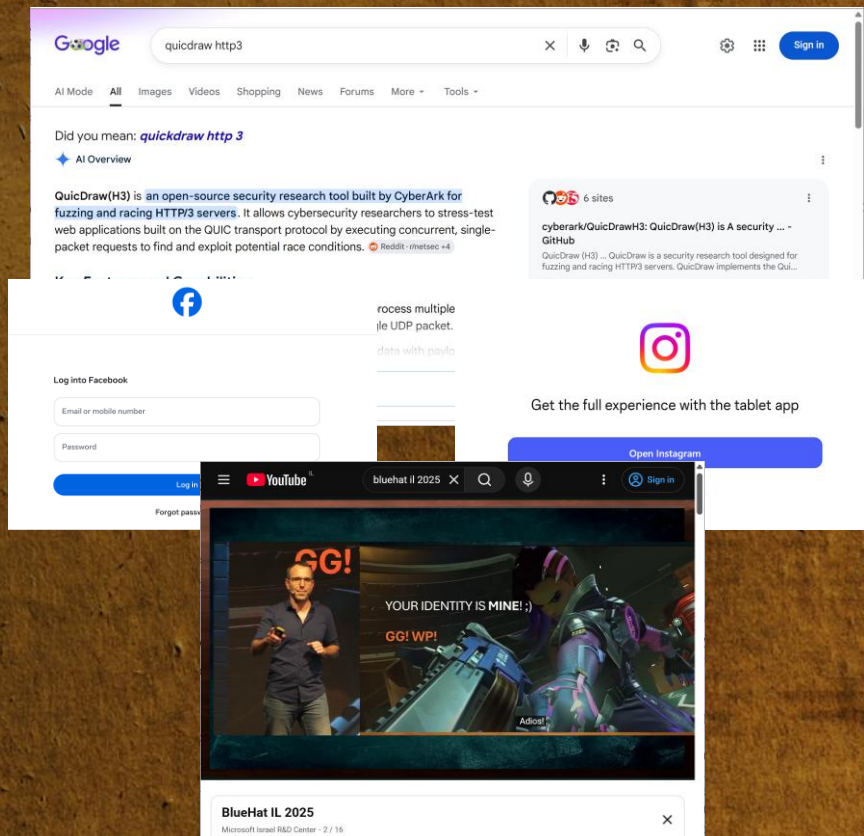
Browsers (caniuse.com/http3)

HTTP/3? - Internet-wide support

- 39.8% (w3techs) of Internet Facing
 - ~13M+ HTTP3 Live Websites
- Major Browsers
- Major CDNs & Cloud Providers
- Major Websites

Name	Method	Status	Protocol	Domain
<u>www.youtube.com</u>	GET	200	h3	www.youtube.com
<u>www.instagram.com</u>	GET	200	h3	www.instagram.com
<u>www.google.com</u>	GET	200	h3	www.google.com
www.facebook.com	GET	200	h3	www.facebook.com
www.cloudflare.com	GET	200	h3	www.cloudflare.com
www.bing.com	GET	200	h3	www.bing.com

Selected HTTP/3 domains



MISSION OBJECTIVE :

List HTTP/3 Features & Find security leads (scenarios)!

GET (:authority:) host :)

Name: Maor Abutbul

Background: Father, Engineer, Researcher, Gamer

Past: ~20 Years in Network & Security,
Engineering, AppSec, R&D, and Research

Current: Vulnerability Researcher @PANW Idira Labs



<https://il.linkedin.com/in/maor-abutbul>

Agenda

#1 Technical Background

#2 Introduction to HTTP/3

- Practical HTTP/3 Tips (&Tools)

#3 Our Research Journey

- 'Quic-Fin-Sync' (algorithm)

#4 QuicDraw & QuicDraw-UI (our open-source)

- QuicDraw Evaluation - Live Demo

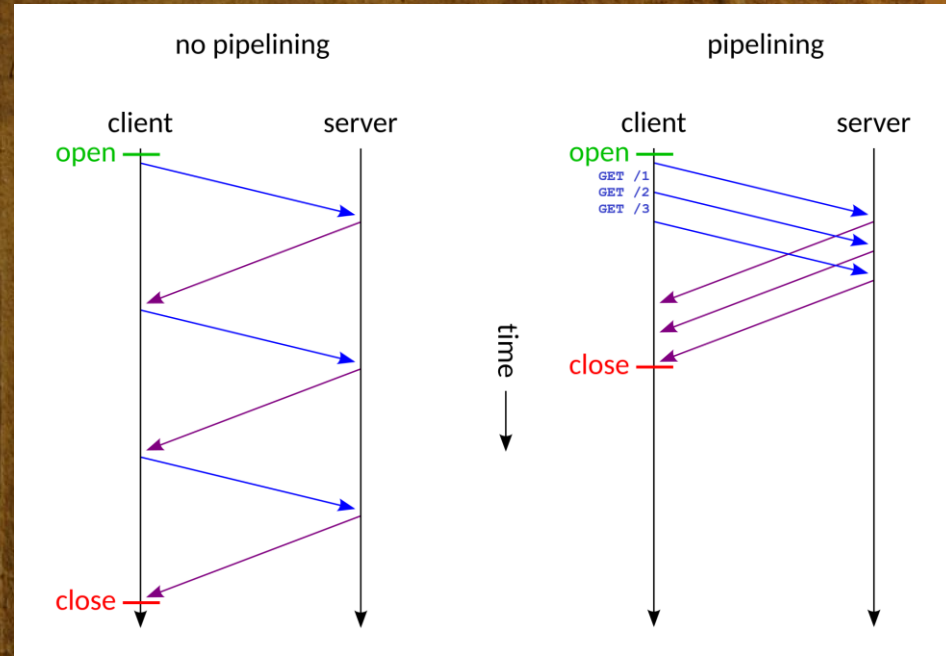
#1 Technical Background

HTTP/1.1, HTTP/2 & HOL blocking

And why do we need **HTTP/3**?

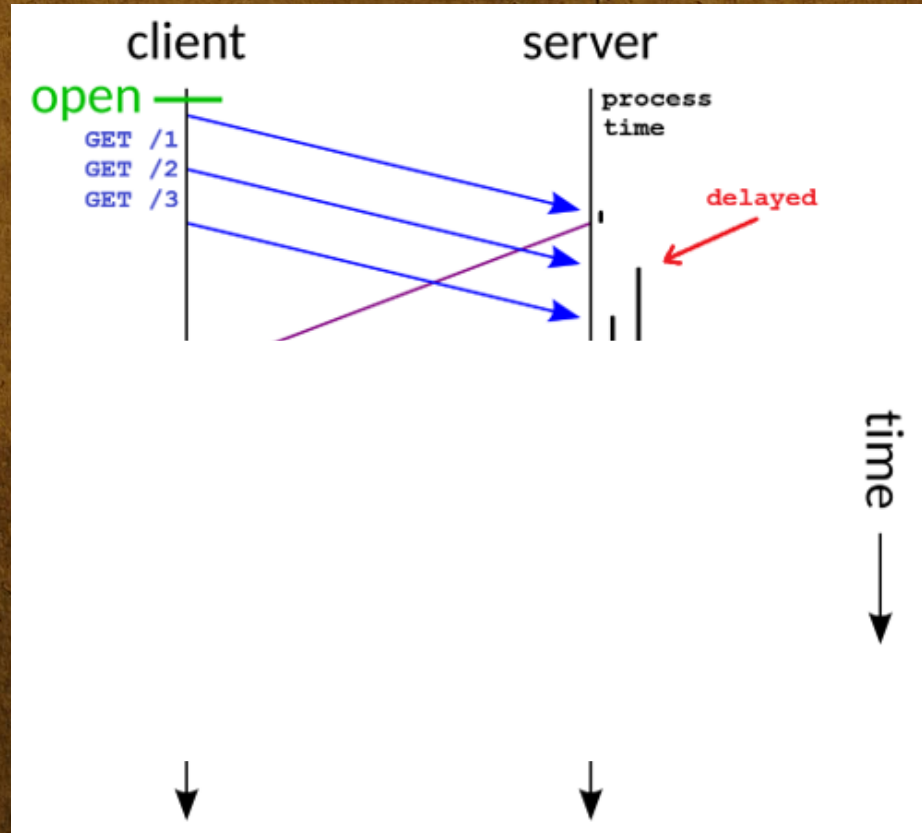
HTTP/1.1 Pipelining

- HTTP/1.0 - response fully received
- HTTP/1.1 - introduced pipelining
- Limitation:
 - Response fully delivered (Ordered) before a new response can be sent.
- But What if a response is delayed?



HTTP/1.1 Pipelining

Pipelining – HTTP Head-Of-Line (HOL) Blocking



HTTP/1.1 HOL Blocking

Brief overview of HTTP2

- Binary
- Encrypted (Browsers - De facto)
- Header compression (HPack)
- Support Multiplexing
 - (using frames and streams)

```

v HyperText Transfer Protocol 2
  > Stream: WINDOW_UPDATE, Stream ID: 0, Length 4
v HyperText Transfer Protocol 2
  > Stream: HEADERS, Stream ID: 1, Length 36, GET /hello?name=m2a_11_00_01
v HyperText Transfer Protocol 2
  > Stream: HEADERS, Stream ID: 3, Length 22, GET /hello?name=m2a_11_00_02
v HyperText Transfer Protocol 2
  > Stream: HEADERS, Stream ID: 5, Length 23, GET /hello?name=m2a_11_00_03

```

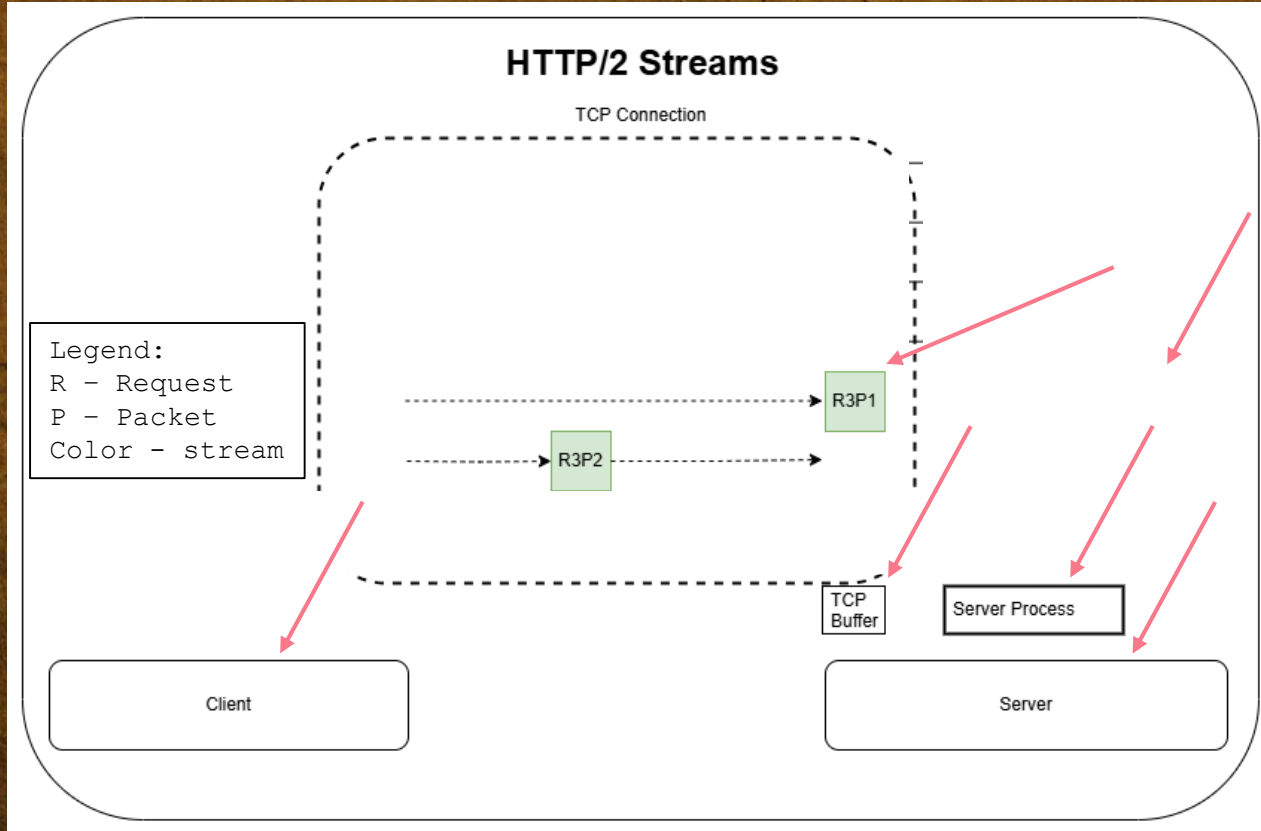


parallel_single-packet_decrypted.pcapng

Brief overview of HTTP2

- Binary
- Encrypted (Browsers - De facto)
- Header compression (HPack)
- Support Multiplexing
 - (using frames and streams)
- Eliminates application-layer (HTTP/1.1) Head-Of-Line blocking

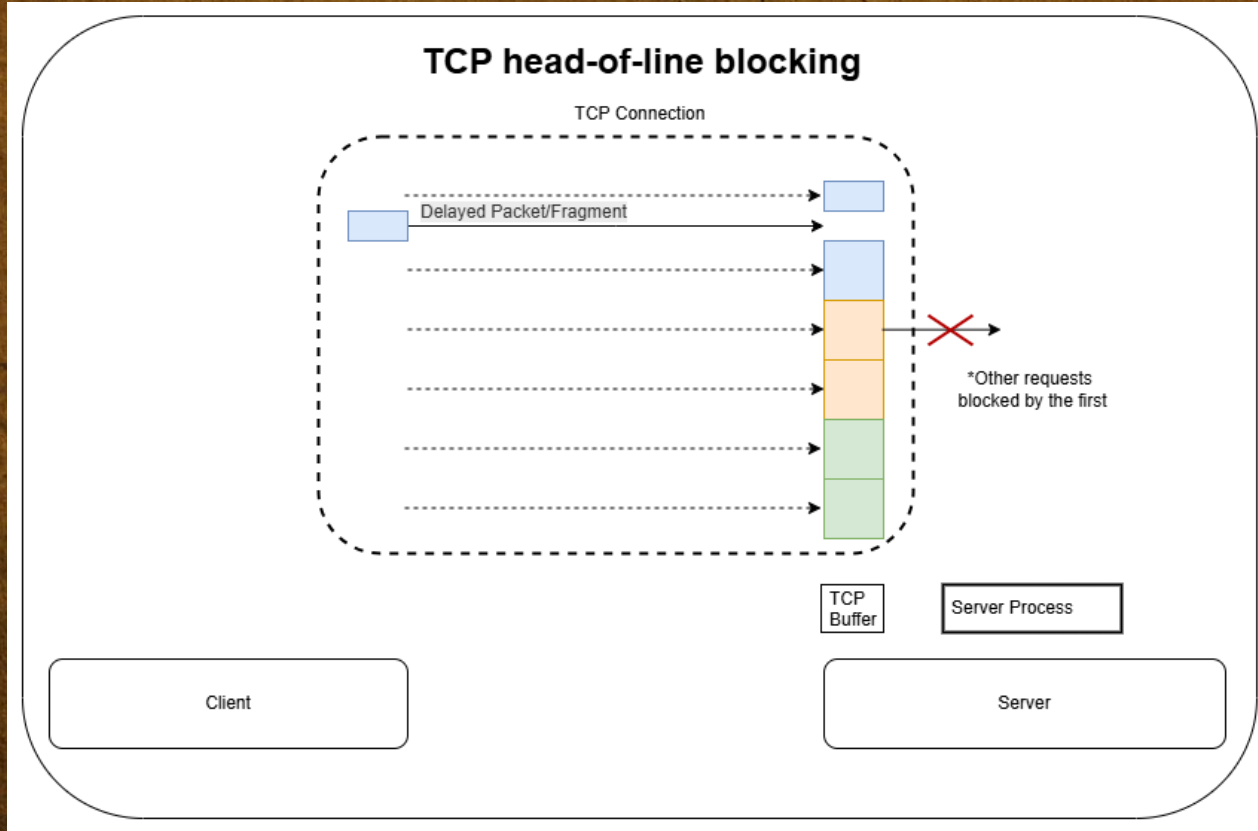
HTTP/2 Streams - HOL blocking Elimination



HTTP/2 Streams - sent immediately

BUT - NETWORK LAYER

TCP Head-Of-Line Blocking



HTTP Stack
Perspective

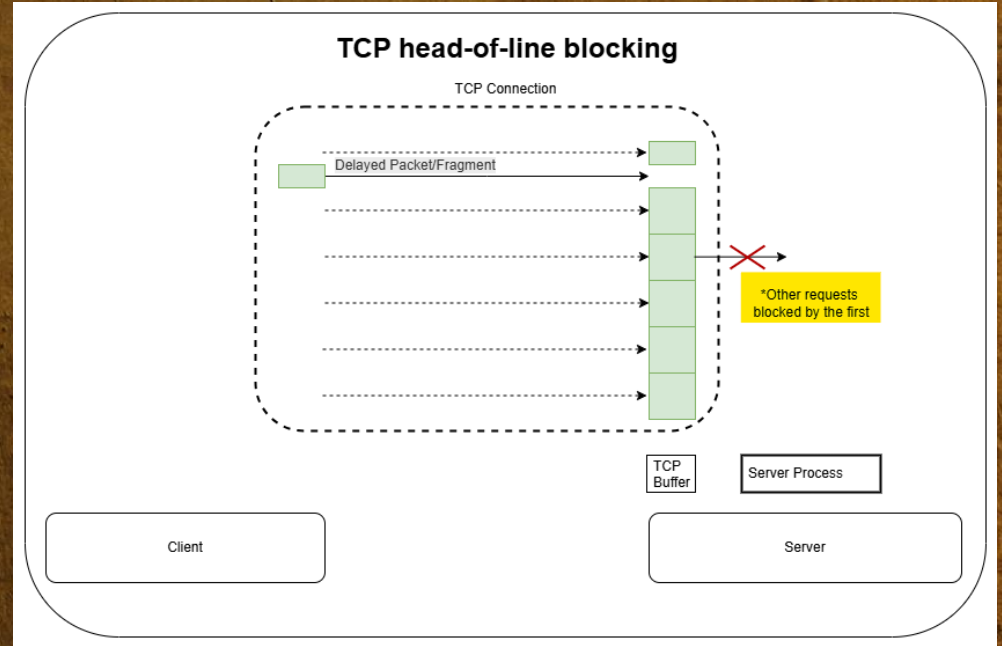
TCP Stack
Perspective

TCP HOL blocking (HTTP / TCP stack)

Brief overview of HTTP2

HTTP/2

- Eliminates HTTP HOL blocking
- Suffers from TCP HOL blocking



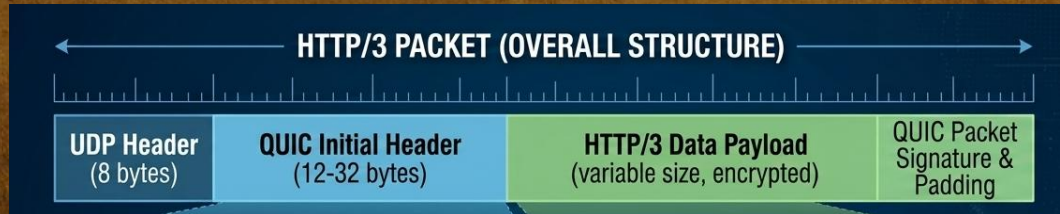
HTTP/2 Streams

SOLVING TCP HOL BLOCKING – PRIMARY
MOTIVATIONS BEHIND HTTP/3

#2 Brief Introduction to HTTP/3

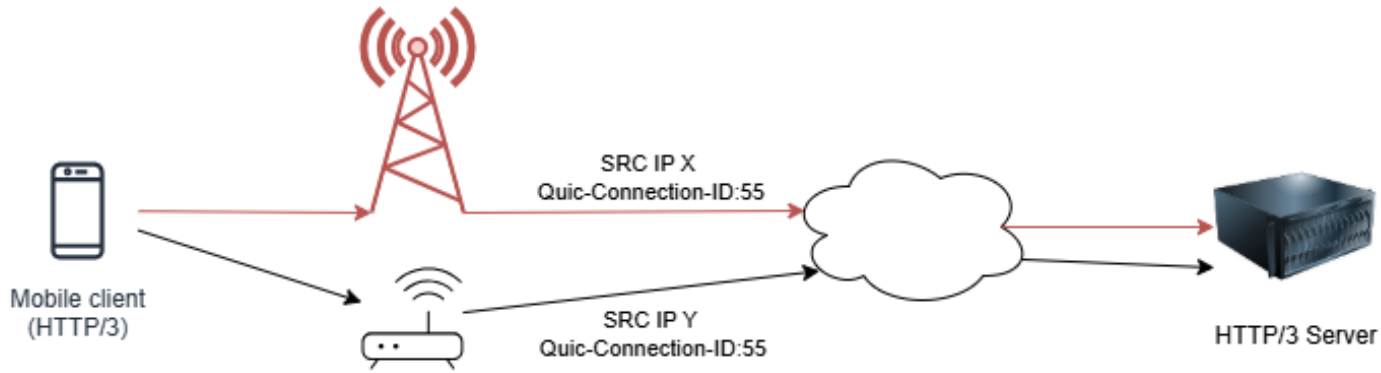
Note

HTTP/3 \approx HTTP/2 OVER QUIC



HTTP/3 (+Quic) Main Features

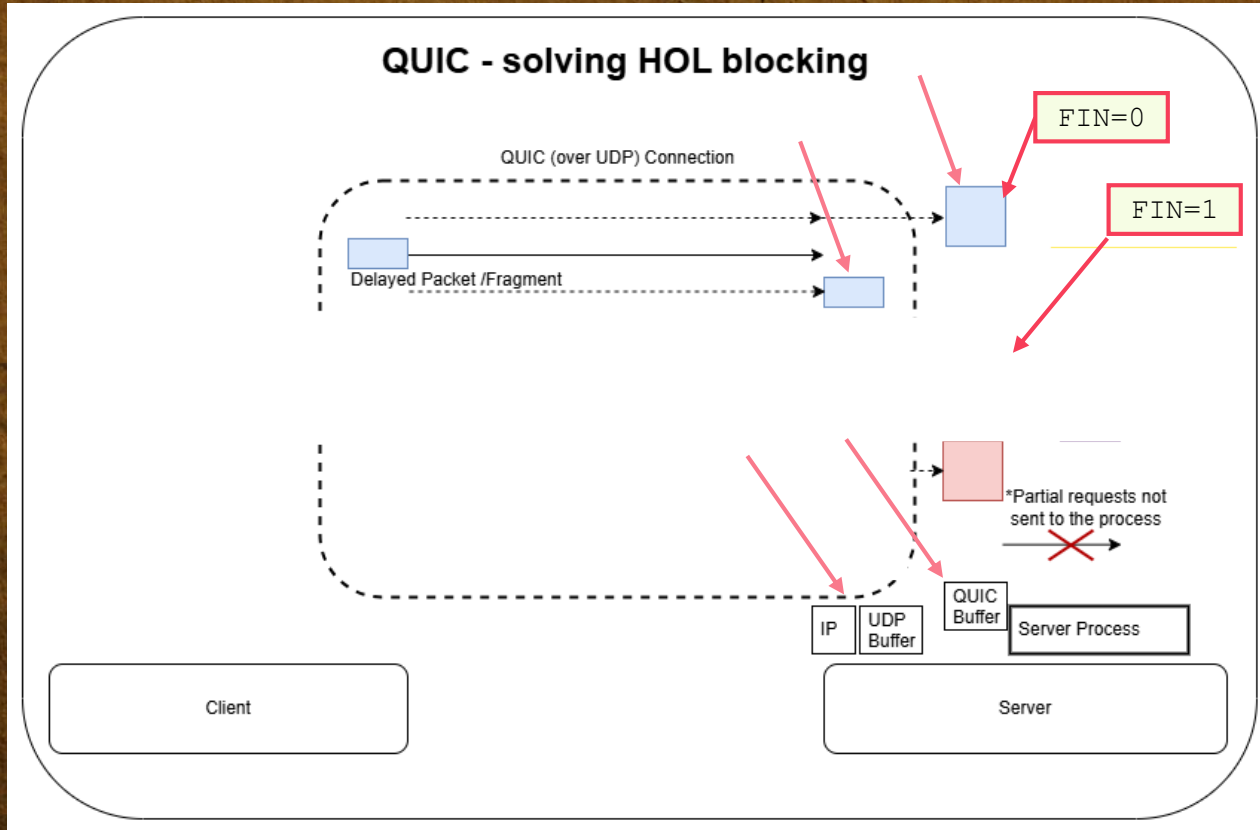
Quic Connection Migration



- Al
- Bi
- He
- Us

- Connection Migration
- Fast connection setup & 0-RTT
- Multiplexing Without Head-of-Line Blocking

QUIC (multiplexing) Without Head-of-Line Blocking



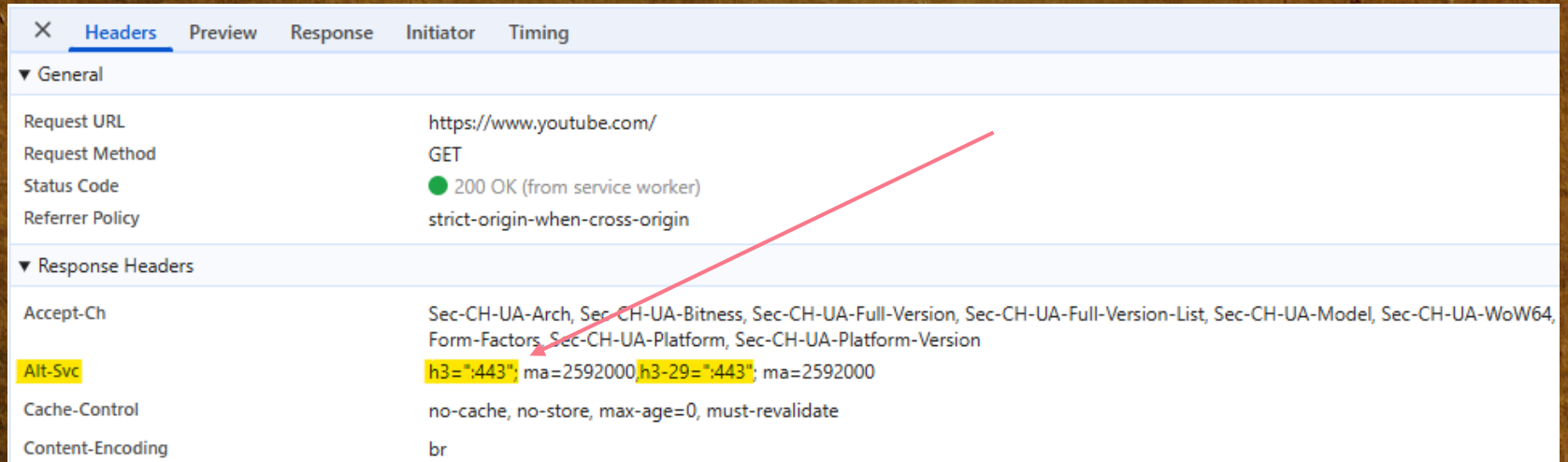
QUIC Solve TCP HOL Blocking

#2.1 HTTP/3 Practical Tips

Server supports HTTP/3?

HTTP/3 Support Discovery

- The Alt-Svc header

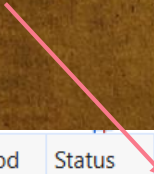


X Headers Preview Response Initiator Timing	
▼ General	
Request URL	https://www.youtube.com/
Request Method	GET
Status Code	● 200 OK (from service worker)
Referrer Policy	strict-origin-when-cross-origin
▼ Response Headers	
Accept-Ch	Sec-CH-UA-Arch, Sec-CH-UA-Bitness, Sec-CH-UA-Full-Version, Sec-CH-UA-Full-Version-List, Sec-CH-UA-Model, Sec-CH-UA-WoW64, Form-Factors, Sec-CH-UA-Platform, Sec-CH-UA-Platform-Version
Alt-Svc	h3=":443"; ma=2592000, h3-29=":443"; ma=2592000
Cache-Control	no-cache, no-store, max-age=0, must-revalidate
Content-Encoding	br

The Alt-Svc header in a response indicates the server supports HTTP/3

HTTP/3 Support Discovery

- All Major web browsers support HTTP/3
- Prefer using HTTP/3



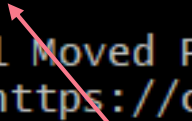
Name	Method	Status	Protocol	Domain
www.youtube.com	GET	200	h3	www.youtube.com
www.instagram.com	GET	200	h3	www.instagram.com
www.google.com	GET	200	h3	www.google.com
www.facebook.com	GET	200	h3	www.facebook.com
www.cloudflare.com	GET	200	h3	www.cloudflare.com
www.bing.com	GET	200	h3	www.bing.com

TOOLS

Tools supporting HTTP3 - browsers & curl

- All Major web browsers support HTTP/3.
 - *local proxy => not using HTTP/3
- Build curl with HTTP/3 Support
 - \$ curl --http3-only https://cyberark.com
- (use "export SSLKEYLOGFILE=" for SSL decryption)

```
$ curl3 --http3-only https://cyberark.com
<html>
<head><title>301 Moved Permanently</title></head>
<body><a href="https://cyberark.com/cdn-cgi/content?"
```



Tools supporting HTTP3 - Wireshark

- All Major web browsers support HTTP/3.
- Build curl with HTTP/3 Support
- Wireshark ! (headers)
 - Edit -> Preferences -> Protocols -> TLS -> "Master Secret log filename"

No.	Time	Source	Destination	Protocol	Leng	strm_count	Stream ID	Info	Frame Type	Value
74	2026-02-04 0...	192.168.1.238	104.16.68.86	HTTP3	255	3	2,6,10	SETTINGS, MAX PUSH ID, STREAM(6), STREAM(10)	ACK,CRYPTO,NEW_...	
75	2026-02-04 0...	104.16.68.86	192.1...	> User Datagram Protocol, Src Port: 58238, Dst Port: 443					STREAM	
76	2026-02-04 0...	104.16.68.86	192.1...	> QUIC IETF					ACK,HANDSHAKE_D...	
77	2026-02-04 0...	104.16.68.86	192.1...	> Hypertext Transfer Protocol Version 3					STREAM	
78	2026-02-04 0...	104.16.68.86	192.1...	> Request Stream					STREAM	
79	2026-02-04 0...	192.168.1.238	104.1...	> HEADERS len=28				ark.com/	ACK,PING,STREAM	GET,https,
81	2026-02-04 0...	104.16.68.86	192.1...					ently	STREAM	301,Wed, 6
82	2026-02-04 0...	104.16.68.86	192.1...						STREAM	
83	2026-02-04 0...	104.16.68.86	192.1...						STREAM	

[Headers Count: 5]

- > Header: **:method:** GET
- > Header: **:scheme:** https
- > Header: **:authority:** cyberark.com
- > Header: **:path:** /
- > Header: **user-agent:** aioquic/1.3.0

Tools supporting HTTP3 - QuicDraw

- All Major web browsers support HTTP/3.
- Build curl with HTTP/3 Support
- Wireshark !
- QuicDraw & QuicDraw-UI 😊
 - \$ pip install quicdraw[ui]
 - \$ quicdraw <https://cyberark.com>



```
$ quicdraw https://cyberark.com  
Response headers received (stream:0) GET : status: (301)  
2026-02-04 11:02:47,076 INFO QuicDraw Response data received (stream:0) GET : 162 bytes
```

Agenda

#1 Technical Background

#2 Introduction to HTTP/3

- Practical HTTP/3 Tips (&Tools)

#3 Our Research Journey

- 'Quic-Fin-Sync' (algorithm)

#4 QuicDraw & QuicDraw-UI (our open-source)

- QuicDraw Evaluation - Live Demo

#3 Our Research Journey

MISSION OBJECTIVE :

List HTTP/3 Features & Find security leads (scenarios)!

HTTP/3 ATTACK SCENARIOS – EASY

Comment

HTTP/3 Attack Scenarios – EZ (selected Scenarios)

- ~~Amplification~~
- ~~Slow Loris Attacks~~
- ~~User Tracking?~~
- ~~Scenarios #4+~~

THIS IS THE END

Or is it?

One more lead


HTTP/3 Attack Scenarios - One more lead

- Keycloak (Race condition finding)
 - Single-Packet Attack (Portswigger)
 - (More details later)
- HTTP/3 - Race conditions !?

Sign in to your account

Username or email

Password

[Forgot Password?](#)

[Sign In](#)

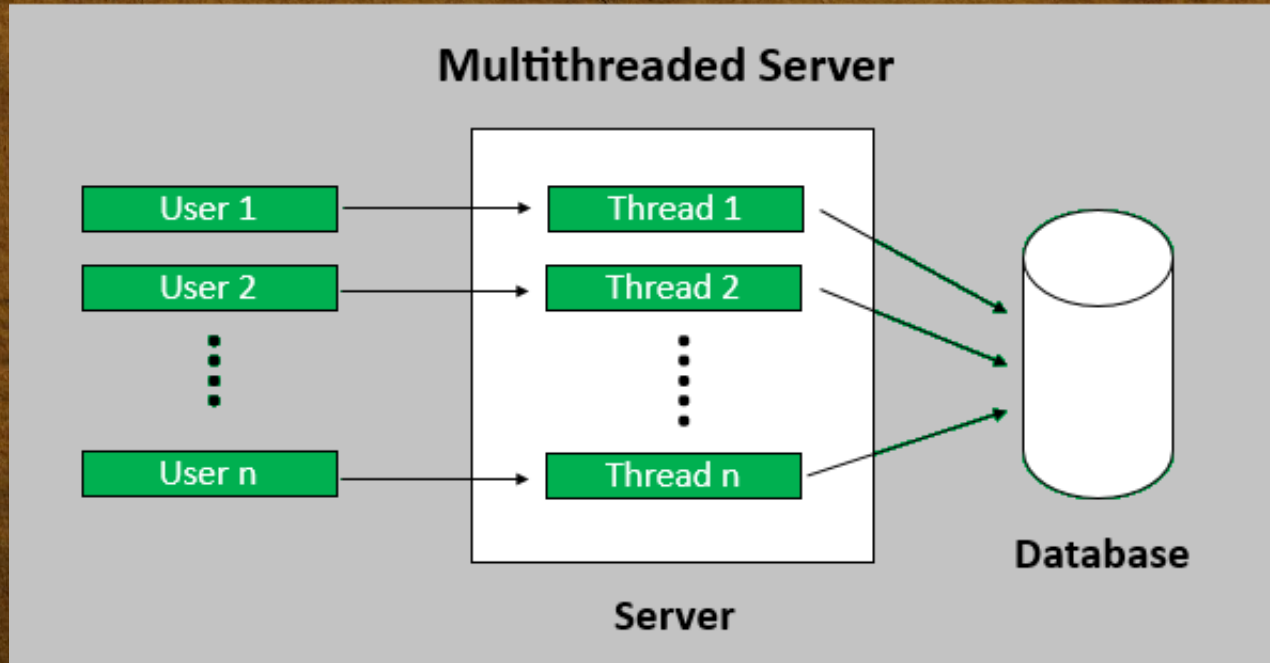
[New user? Register](#)



Is it possible to make race conditions work in HTTP/3?

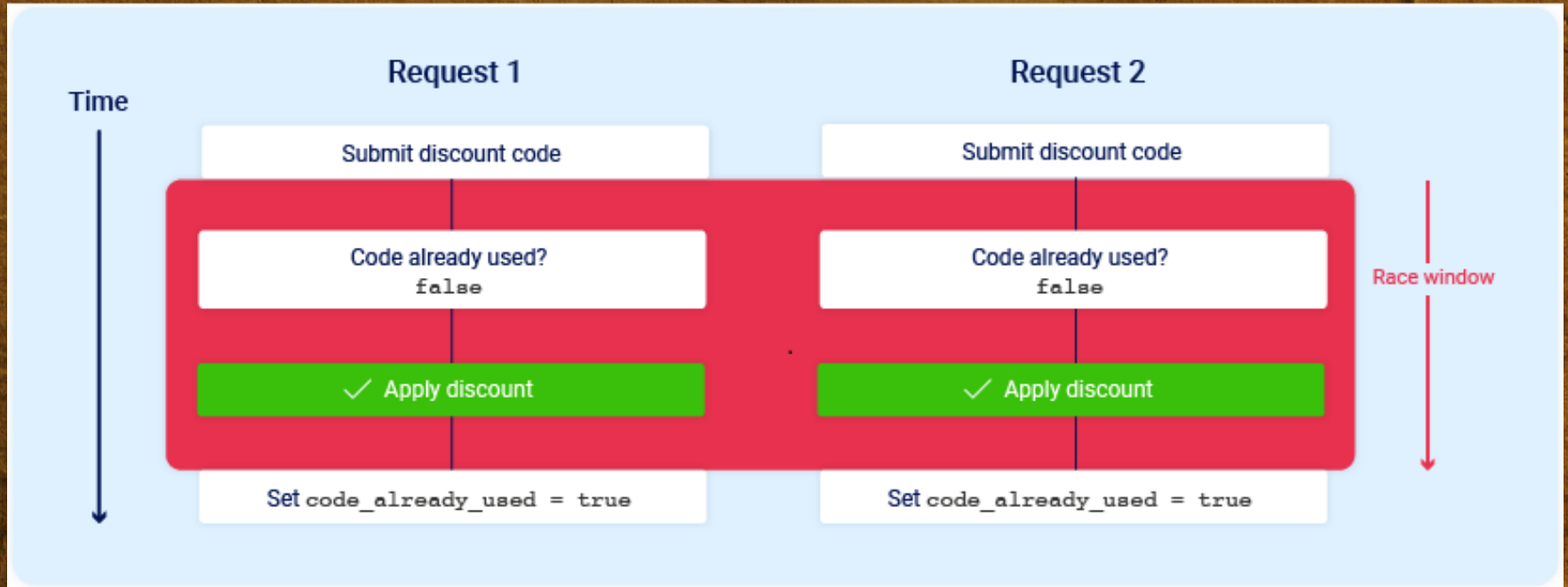
On Race Conditions

Multithreaded Server



"A race condition vulnerability requires a 'collision' of two concurrent operations on a shared resource." ([PortSwigger Research](#))

The classic discount code - Race Flow



Race Conditions - Race Flow

QUIC on the RaceTrack



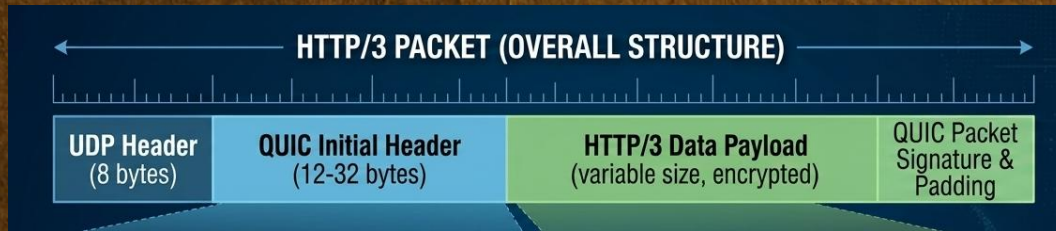
QUIC on the RaceTrack

Is it possible to make race conditions work in HTTP/3?

A way to send / release requests in bulk

QUIC on the RaceTrack – First Attempt: Fragmentation

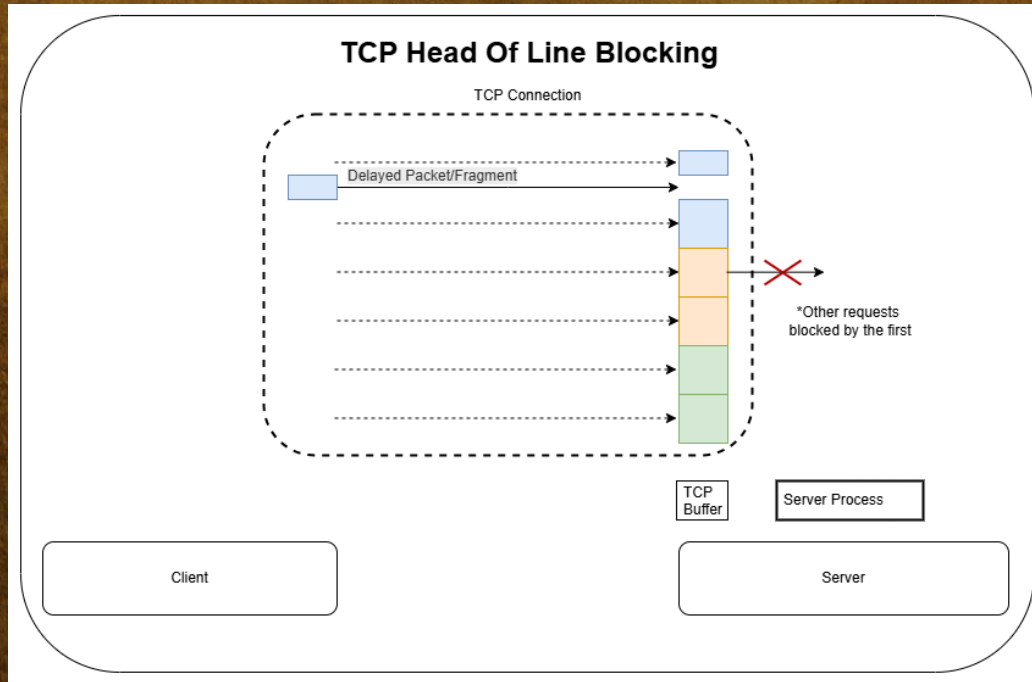
- HTTP/3 is 'just' HTTP2 over QUIC
 - (QUIC implements "the good parts of TCP")
- QUIC runs over UDP
- UDP runs over IP
 - IP has Fragmentation!



“Let’s just Fragment the HTTP/3 traffic”

Racing on HTTP/3 – “Let’s just fragment the HTTP/3 traffic”

Using TCP HOL Blocking (HTTP/2)



TCP HOL Blocking Fragmentation

Fragmentation – EZ ;)

Found a solution

EZ GG

What?

We did! ... Didn't we?

Why?

"Humm..."

"NOP"

"RTFM!"

"RTFM (RTF RFC)"

NOP! Read The RFC

QUIC [RFC \(9000\)](#), "UDP datagrams **MUST NOT** be fragmented at the IP layer."

UDP datagrams **MUST NOT** be fragmented at the IP layer. In IPv4 [[IPv4](#)], the Don't Fragment (DF) bit **MUST** be set if possible, to prevent fragmentation on the path.

RFC9000 - QUIC don't fragment

~_(ツ)_/~

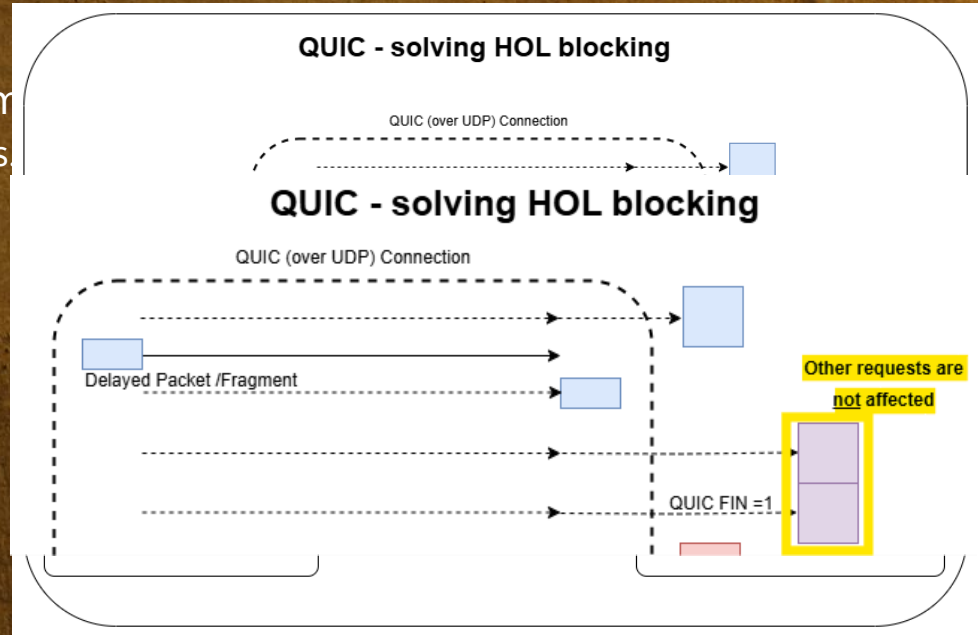
Racing on HTTP/3 - "Let's just fragment the HTTP/3 traffic"

Anyway. Each request has its own stream!

Racing on HTTP/3 - "Let's just fragment the HTTP/3 traffic"

Anyway. Each request has its own stream

- Should not interfere with other streams



QUIC Solve HOL Blocking
(Fragmentation)

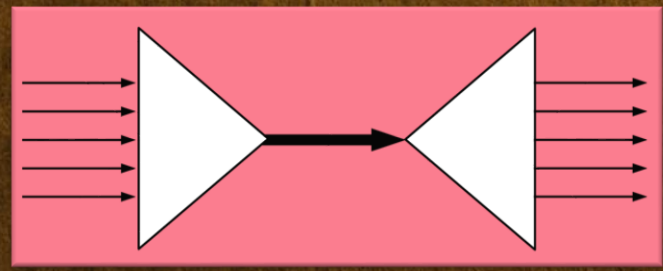
"Yes, But!"

"Streams can be Multiplexed!"

Amazing! But What's Multiplexing?

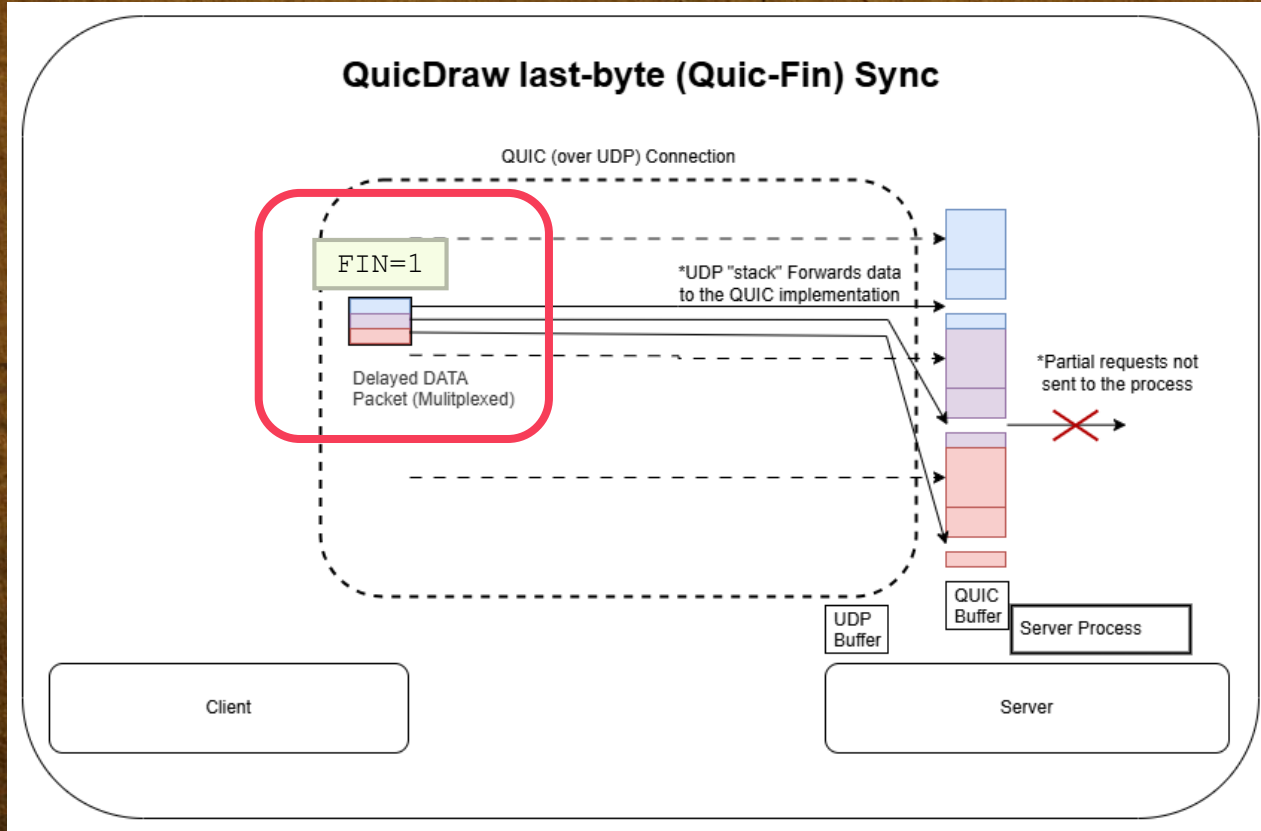
Current filter: http3

Request Versi	Info	Stream ID	Frame Type
Protected Payload (KP0),	DCID=33e641d76a95e0d4, PKN: 15, STREAM(0), HEADERS: 200 OK	0	STREAM
Protected Payload (KP0),	DCID=33e641d76a95e0d4, PKN: 16, STREAM(0), DATA	0	STREAM
Protected Payload (KP0),	DCID=33e641d76a95e0d4, PKN: 17, STREAM(7), STREAM(4), HEADERS: 200 OK	7,4	STREAM,STREAM
Protected Payload (KP0),	DCID=33e641d76a95e0d4, PKN: 18, STREAM(4), DATA	4	STREAM
Protected Payload (KP0),	DCID=33e641d76a95e0d4, PKN: 19, STREAM(8), HEADERS: 200 OK	8	STREAM
Protected Payload (KP0),	DCID=33e641d76a95e0d4, PKN: 20, STREAM(8), DATA	8	STREAM



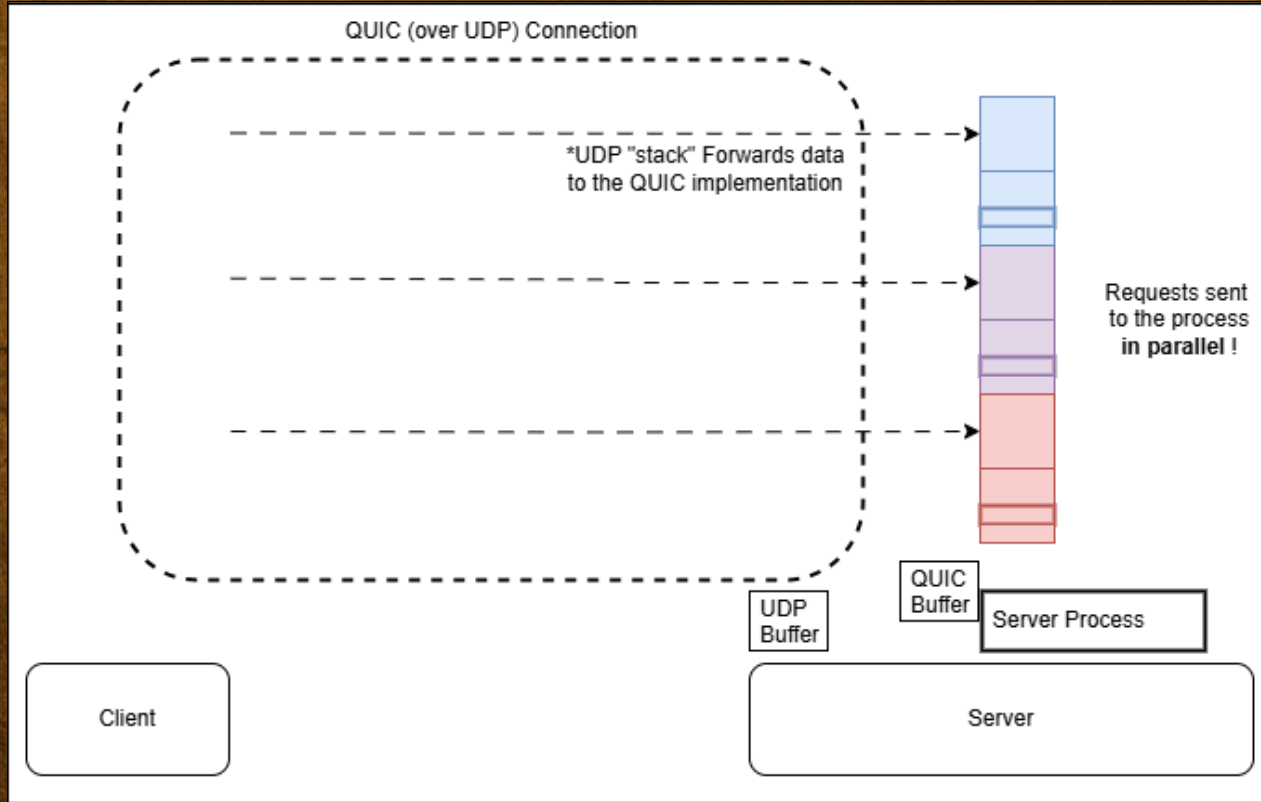
#3.2 Quic-Fin-Sync - Concept

QUIC on the Racetrack - Multiplexing - Concept



QuicDraw Network (Concept)

QUIC on the Racetrack - Quic-Fin-Sync - Concept



QuicDraw Network (Concept)

Decided to build our own tool

Decided to build our own tool

Why reinvent the wheel?

Agenda

#1 Technical Background (HTTP/2)

#2 Introduction to HTTP/3

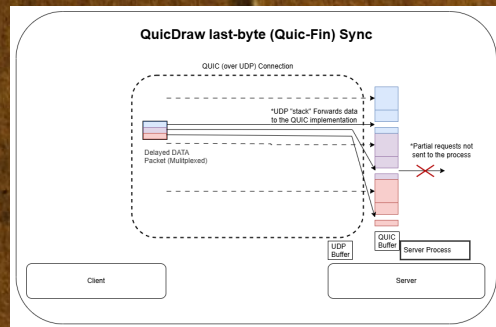
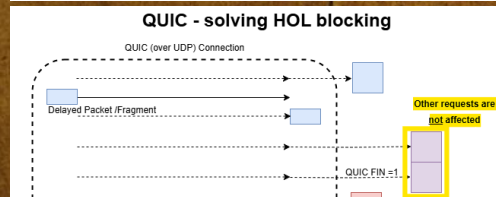
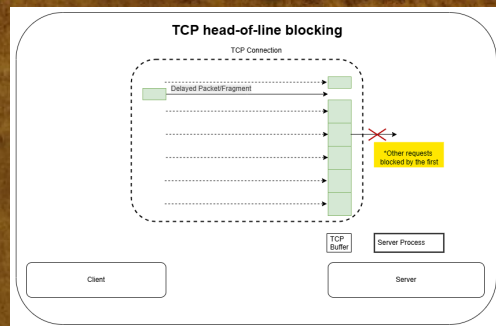
- Practical HTTP/3 Tips (&Tools)

#3 Our Research Journey

- 'Quic-Fin-Sync' (algorithm)

#4 QuicDraw & QuicDraw-UI (open-source)

- QuicDraw Evaluation - Live Demo





#4 QuicDraw & QuicDraw-UI

Our Quic-Fin-Sync Implementation - The Algorithm and Traffic Diagram

For each Request

```
queue Request-Headers
```

```
queue Request-Data[:-1]
```

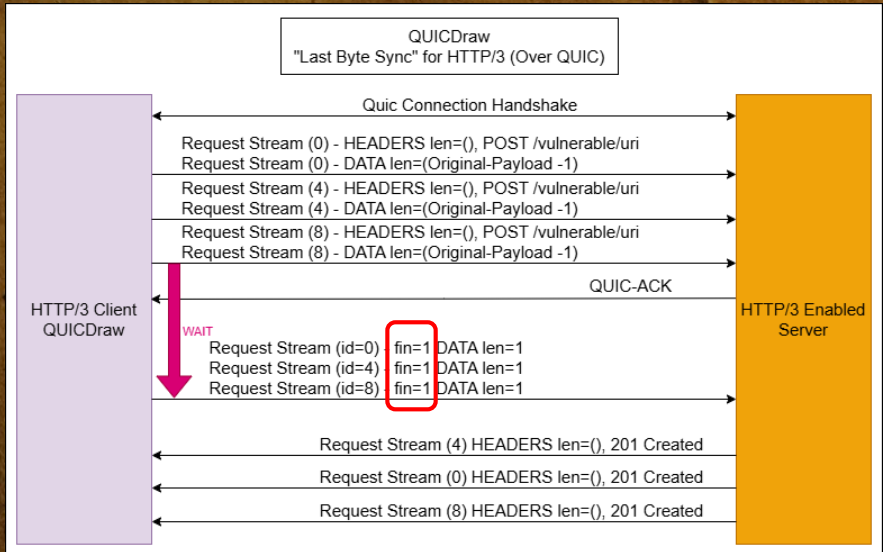
Transmit

wait 1000 ms

For each Request

```
queue RequestData[-1:] & end_stream=1  
                                (QUIC-FIN = 1)
```

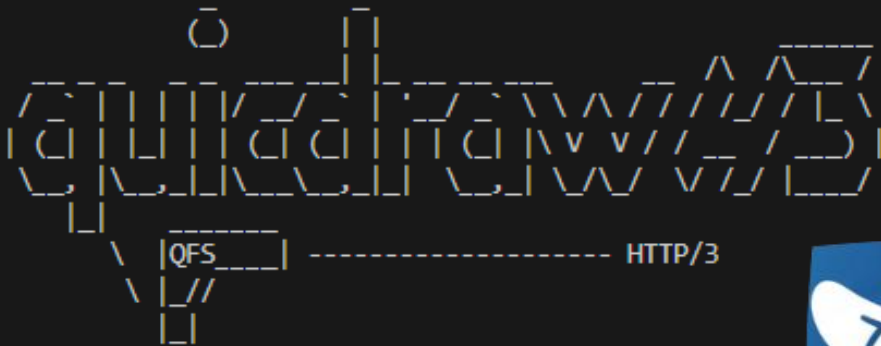
Transmit



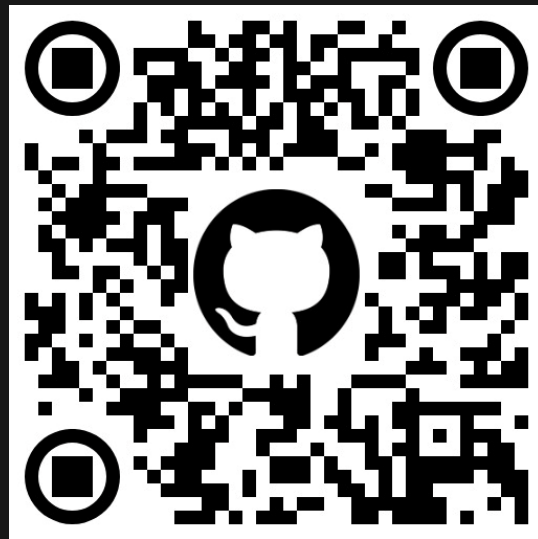
QuicDraw Quic-Fin-Sync
(HTTP3) traffic diagram

QuicDraw & QuicDraw-UI

QuicDraw: HTTP/3 Fuzzing and Racing (Client)



GitHub: <https://github.com/cyberark/QuicDrawH3>
License: Apache-2.0 License
Author: Maor Abutbul <CyberArk Labs>
Version: 0.8.27



QuicDraw(&QuicDraw-UI) GitHub Repo

Racing HTTP/3 applications

```
$ pip install quicdraw[ui]
```

```
quicdraw <url>
```

To use the same request multiple times

- (using the Quic-Fin-Sync)
- Use the `-tr/--total-requests` argument.

```
quicdraw <url> -tr TOTAL_REQUESTS
```

```
quicdraw https://google.com -tr 7
```

Racing HTTP/3 applications – usage demo

The image shows a Wireshark interface capturing traffic on Wi-Fi (udp port 443). The main display area is empty, indicating that the capture is in progress but no packets have been displayed yet. The status bar at the bottom indicates "Wi-Fi: <live capture in progress>" and "Packets: 6 - Displayed: 0 (0.0%)".

In the bottom right corner, a terminal window is open, showing the following commands and output:

```
MINGW64:/c/m2a/Learning_Presentations/2025/QuicDraw_Presentation/Demo
((demo_env) )
RaorABRaorA-Lap-1 MINGW64 /c/m2a/Learning_Presentations/2025/QuicDraw_Presentation/Demo
$
((demo_env) )
RaorABRaorA-Lap-1 MINGW64 /c/m2a/Learning_Presentations/2025/QuicDraw_Presentation/Demo
$ clear
```

The terminal window also shows a cursor (I) on a new line, suggesting that the user is about to enter another command.

QuicDraw (HTTP3) Quic-Fin-Sync traffic

Destination Port	Protocol
50504	HTTP3
50504	HTTP3
443	HTTP3
443	HTTP3
50504	HTTP3
50504	HTTP3
50504	HTTP3

```
> User Datagram Protocol, Src Port: 50504, Dst Port: 443
  > QUIC IETF
    > QUIC Connection information
      [Packet Length: 172]
    > QUIC Short Header DCID=f12f7e04d88b9f7b PKN=13
    > PING
    > STREAM id=6 fin=0 off=5 len=25 dir=Unidirectional origin=Client-initiated
    > STREAM id=0 fin=1 off=0 len=32 dir=Bidirectional origin=Client-initiated
    > STREAM id=4 fin=1 off=0 len=9 dir=Bidirectional origin=Client-initiated
    > STREAM id=8 fin=1 off=0 len=9 dir=Bidirectional origin=Client-initiated
    > STREAM id=12 fin=1 off=0 len=9 dir=Bidirectional origin=Client-initiated
    > STREAM id=16 fin=1 off=0 len=9 dir=Bidirectional origin=Client-initiated
    > STREAM id=20 fin=1 off=0 len=9 dir=Bidirectional origin=Client-initiated
    > STREAM id=24 fin=1 off=0 len=9 dir=Bidirectional origin=Client-initiated
  > Hypertext Transfer Protocol Version 3
  > Hypertext Transfer Protocol Version 3
  > Hypertext Transfer Protocol Version 3
```

```
e8cbfa5038bc, PKN: 9,
e8cbfa5038bc, PKN: 12,
7e04d88b9f7b, PKN: 10,
7e04d88b9f7b, PKN: 13,
e8cbfa5038bc, PKN: 15,
e8cbfa5038bc, PKN: 18,
e8cbfa5038bc, PKN: 20,
```

QuicDraw -tr 7(HTTP3 Quic-Fin-Sync)
(Inspecting TLS traffic)

Fuzzing HTTP/3 applications

- Fuzzing concept, like other web fuzzers: ([Ffuf](#), [Wfuzz](#)).
 - Go over the data section (-d) and
 - Replace any FUZZ with (line in) wordlist (-w)

```
quicdraw <url> -w WORDLIST -d DATA_with_FUZZ_keyword
```

```
quicdraw <https://http3_server.com/path> -w path/to/wordlist  
-d '{"jsonkey":"FUZZ"}'
```

WE HAVE **FUZZING** AND **RACE-CONDITION**
TESTING IN **HTTP/3**



Introducing QuicDraw-UI

A GUI for QuicDraw(H3)

QuicDraw-UI HTTP/3 Request Editor (GET Request)

QuicDrawH3 - HTTP/3 Fuzzing & Racing Tool

HTTP/3 Request Editor Advanced Results

Target URL

https://www.cyberark.com

Custom Headers

Add headers (one per line)

Request Configuration

Data:

{\"key\":\"value\"}

Fuzzing Configuration

Wordlist File:

Wordlist (-w) wordlist.txt

Race Configuration

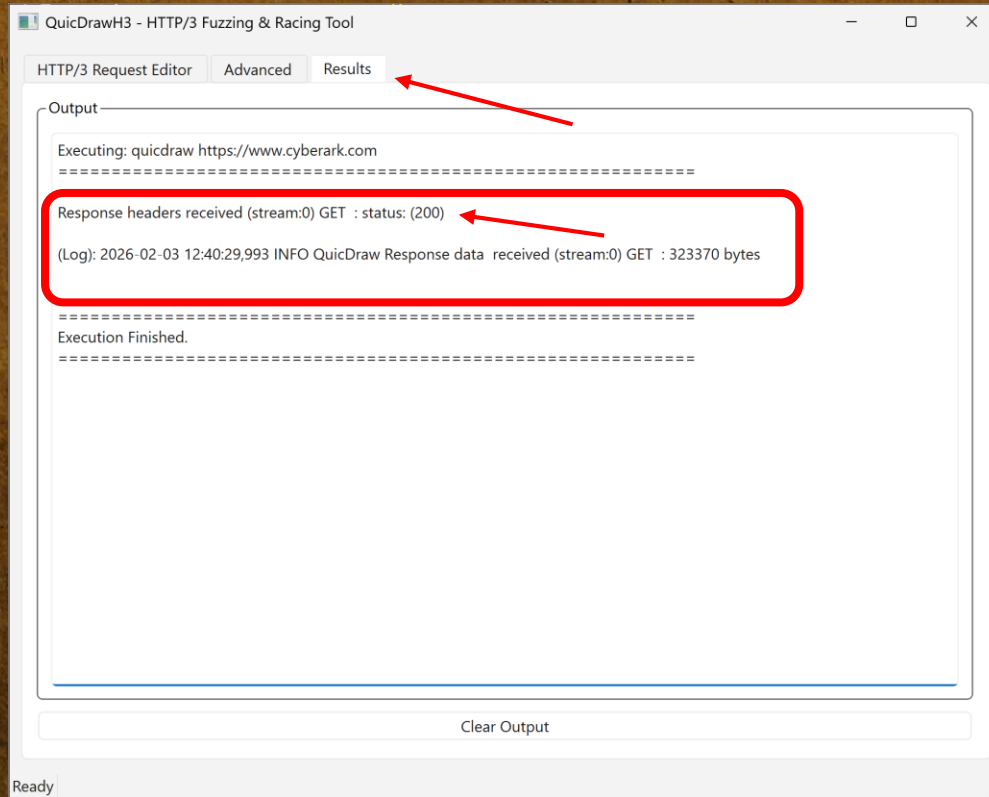
Total Requests:

Repeat Request (-tr) 12 ^ v

Run QuicDraw

Ready

QuicDraw-UI HTTP/3 – (Execution) Results Tab



"It's nice and all but I use other tools.."

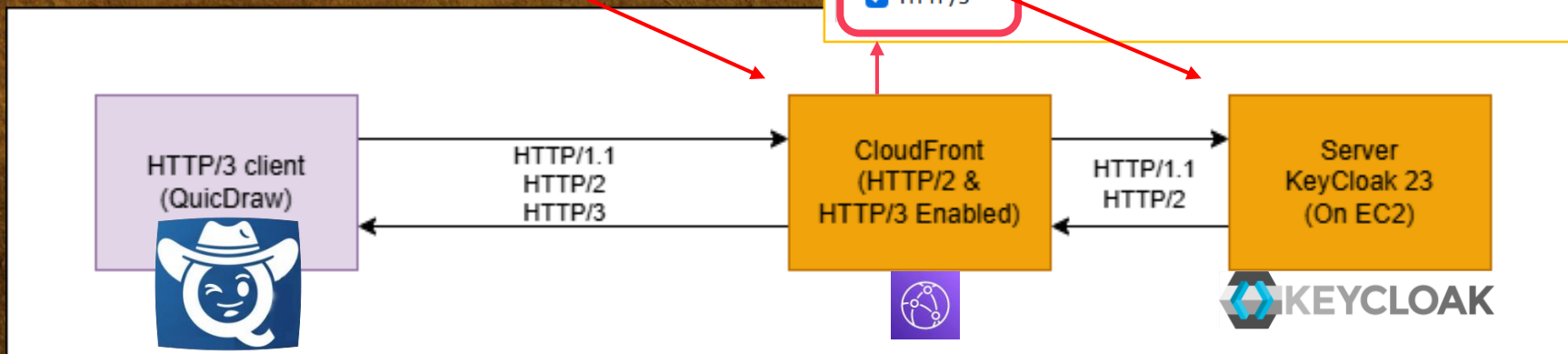
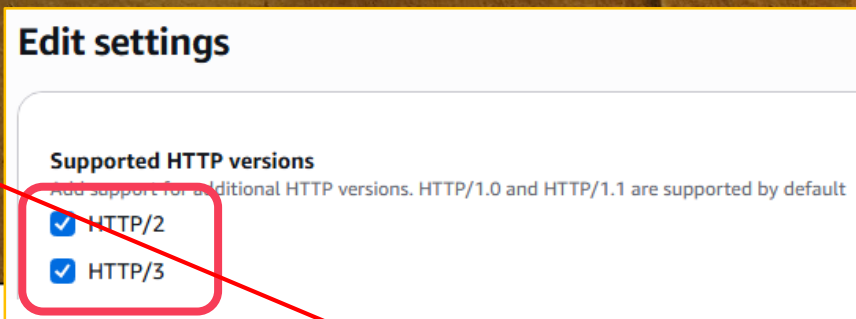
WE HAVE (REQUEST EDITOR) **FUZZING** AND **RACE-**
CONDITION TESTING IN **HTTP/3**

QuicDraw Evaluation

but does it really work?

Our test setup

- Keycloak On AWS EC2
 - Version 23 "Vulnerable"
 - Race condition we reported* (Fixed)
- AWS CloudFront.



Keycloak 23 Setup on AWS

Keycloak Issue Recap

Clients
Clients are applications and services that can request authentication of a user. [Learn more](#)

Clients list | **Initial access token** | Client registration

Search token → [Create](#) 1-1 < >

ID	Created date	Expires	Count	Remaining count
c49fc551-ea4c-498e-96c1-4abb1810afb4	January 30, 2024 at 11:59 AM	January 31, 2024 at 11:59 AM	2	-3

1-1 < >

IAT Token View 5 of 2

A NUMBER **BELOW ZERO** CAUSED A **RACE
CONDITION** ON THE SERVER SIDE.

QuicDraw Performance Demo

PoC Or It Didn't Happen!

Demo Time! (PoC Or It Didn't Happen!)

(Using Mobile Network)

`quicdraw-ui`

`https://d131tng1cpp2km.cloudfront.net/realms/master/clients-registrations/default`

`-l "demo\secrets.log"`


`-H "Host: d131tng1cpp2km.cloudfront.net"`

`-H "Authorization: Bearer eyJhb...gQ"`

`-H "Content-Type: application/json"`

`--data-raw $'{"clientId":"client_quicdrawui_1000_28_02_FUZZ","name":"","description":""}'`

`-w "demo\numbers.txt"`



1
2
.
.
120

Demo Time! – Fuzzing using QuicDraw (Mobile Network)

The screenshot shows the Keycloak Administration UI in a browser window. The address bar displays the URL: `https://d131tng1cpp2km.cloudfront.net/admin/master/console/#/master/clients/initial-access-token`. The page title is "Clients" and the sub-page is "Initial access token". Below the navigation tabs, there is a search bar and a "Create" button. A table lists the initial access tokens.

ID	Created date	Expires	Count	Remaining count
d59ff20-424a-4504-b637-f63cbe02e0d8	October 16, 2025 at 3:06 PM	October 17, 2025 at 3:06 PM	1	1

On the right side of the screen, a mobile network settings overlay is visible. It includes options for Wi-Fi, OnePlus BT, Bluetooth, Airplane mode, Energy saver, Night light, and Accessibility. The battery level is shown as 100%.

Guess?

120 sent using QuicDraw (Quic-Fin-Sync)

No race (1)

2-20

20-50

50-80

80+



Race
Window

Quic-Fin-Sync - Can we do better?

For each Request

```
queue Request-Headers
```

```
queue Request-Data[:-1]
```

Transmit

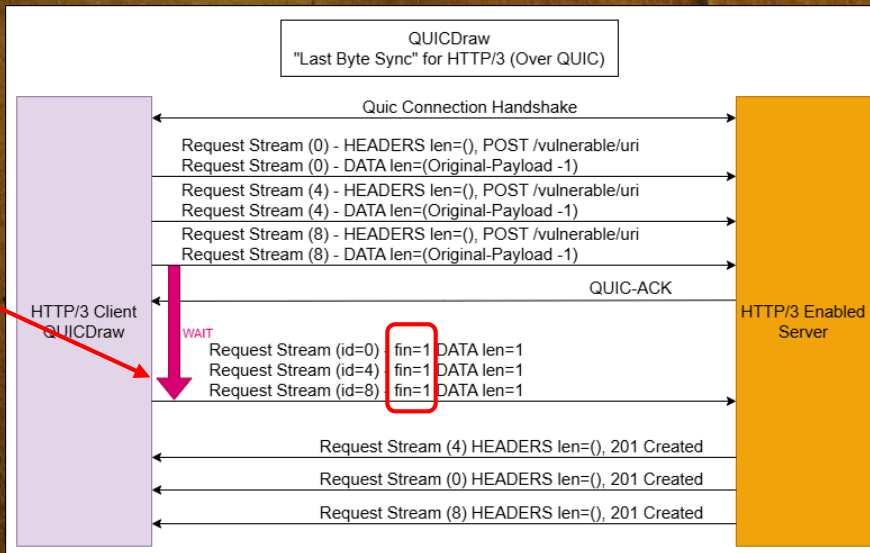
```
wait 1000 ms
```

For each Request

```
queue RequestData[-1:] & end_stream=1  
(QUIC-FIN = 1)
```

Transmit

```
-sd <seconds>  
(sync-delay)  
*version 0.9.1+
```



QuicDraw Quic-Fin-Sync
(HTTP3) traffic diagram

Summary, Takeaways & Further research

References

- <https://webtechsurvey.com/technology/http-3>
- <https://caniuse.com/http3>
- <http://w3techs.com/>
- [RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport](#)
- [RFC 9114: HTTP/3](#)
- [The Ultimate Guide To The HTTP/3 And QUIC Protocols | DebugBear](#)
- [Why HTTP/3 is eating the world | APNIC Blog](#)
- [6 ways HTTP/3 benefits security \(and 7 serious concerns\) | CSO Online](#)
- [README | HTTP/3 explained](#)
- <https://flatt.tech/research/posts/beyond-the-limit-expanding-single-packet-race-condition-with-first-sequence-sync/>
- <https://www.cyberark.com/resources/threat-research-blog/you-cant-always-win-racing-the-keycloak> (our keycloak reaserch)
- <https://aioquic.readthedocs.io/en/latest/h3.html>
- <https://aioquic.readthedocs.io/en/latest/asyncio.html>Http3Session and Streams — Firefox Source Docs documentation
- <https://github.com/m2a2/awesome-http3>
- [HTTP pipelining – Wikipedia](#) (image)
- <https://ideogram.ai/> (cover image)

Takeaways

HTTP/3

- Wide Internet adoption
- Using QUIC
 - Connection Migration
 - Eliminates TCP HOL blocking
- Not supported by security tools / interception proxies

QuicDraw & QuicDraw-UI (Open-source tool)

- Can fuzz and race HTTP/3 web servers (using Quic-Fin-Sync)
- QD-UI HTTP/3 Request Editor
- In our test case - Exploited a race condition 80+? times!?

Multiplexed protocols

- can be susceptible to a last-byte-sync-like attacks

Further Research & Contribution

- QUIC and HTTP/3 research.
 - Many “independent” implementations
 - (<https://github.com/m2a2/awesome-http3#implementations>)
- Racing other multiplexed protocols
- Contribute to QuicDraw & QuicDraw-UI 😊 (We need you!)
 - (<https://github.com/cyberark/QuicDrawH3>)
 - Extend for other attacks over HTTP/3

Questions?

ONE MORE SLIDE...

Read more at our blog posts & Thank you

"Fuzzing and Racing HTTP/3: Open-Sourcing QuicDraw(H3)"

<https://www.cyberark.com/resources/threat-research-blog/racing-and-fuzzing-http-3-open-sourcing-quicdraw>

Contact (LinkedIn):



<https://il.linkedin.com/in/maor-abutbul>



QuicDraw & QuicDraw-UI Repo:

<https://github.com/cyberark/QuicDrawH3>

