

Progetto di Gestione di Reti

Luca Cordisco
l.cordisco@studenti.unipi.it

A.A. 2023/2024

1 Introduzione

Lo scopo di questo progetto è quello di realizzare un plugin per Wireshark che implementi il fingerprinting QUIC, ispirandosi a QUIC Hunter [2]. Tuttavia, mentre QUIC Hunter [2] si focalizza sulle librerie server QUIC, questo plugin è specificamente progettato per produrre un fingerprint per i client QUIC.

La validità del fingerprint è stata testata catturando l'handshake con diversi client QUIC.

2 Background

QUIC è un protocollo di trasporto sicuro e affidabile orientato alla connessione. L'handshake di QUIC integra al suo interno un handshake TLS, in quanto usa TLS per garantire la confidenzialità e l'integrità dei pacchetti.

I transport parameters sono impostazioni che permettono di configurare e ottimizzare la connessione per tutti gli endpoint, un po' come le TCP Options. I transport parameters vengono scambiati durante l'handshake e sono inclusi all'interno di una estensione TLS chiamata `quic_transport_parameters`: facendo ciò viene garantita l'integrità di questi valori.

Il plugin analizza il `ClientHello` di ogni handshake QUIC e calcola un fingerprint per ciascuno, considerando l'insieme e l'ordine dei transport parameters, oltre all'ordine di alcune estensioni TLS che verranno discusse più avanti

3 Funzionamento

Il plugin per Wireshark è stato realizzato come post-dissector con il linguaggio Lua. Un post-dissector è un dissector che viene eseguito dopo che vengono eseguiti tutti gli altri dissector. Possono aggiungere ulteriori elementi al dissection tree.

Esso, oltre a mostrare i fingerprint, mostra per ognuno il nome della possibile libreria client, sulla base dei risultati discussi in seguito.

Lo sviluppo e testing è stato fatto con WireShark v4.2.5-0-g4aa814ac25a1 e Lua 5.4.6.

3.1 Installazione

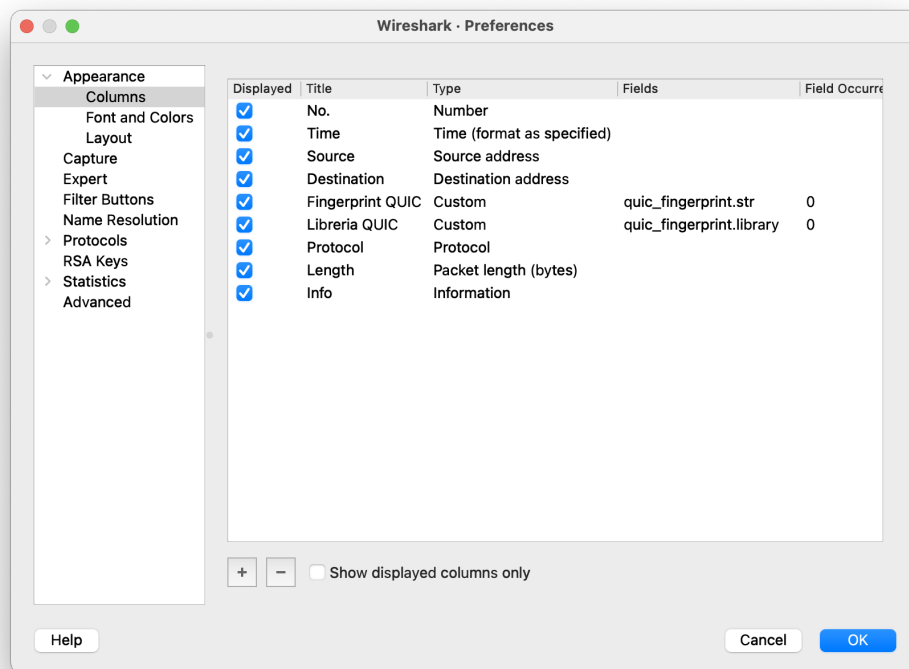
Il plugin è un file con estensione `.lua`¹ e viene caricato ad ogni avvio di Wireshark se viene posizionato nella personal plugin folder o nella global plugin folder.

Il path della personal plugin folder è `%APPDATA%\Wireshark\plugins` su sistemi Windows e `/.local/lib/wireshark/plugins` su sistemi Unix-like.

3.2 Modalità d'uso

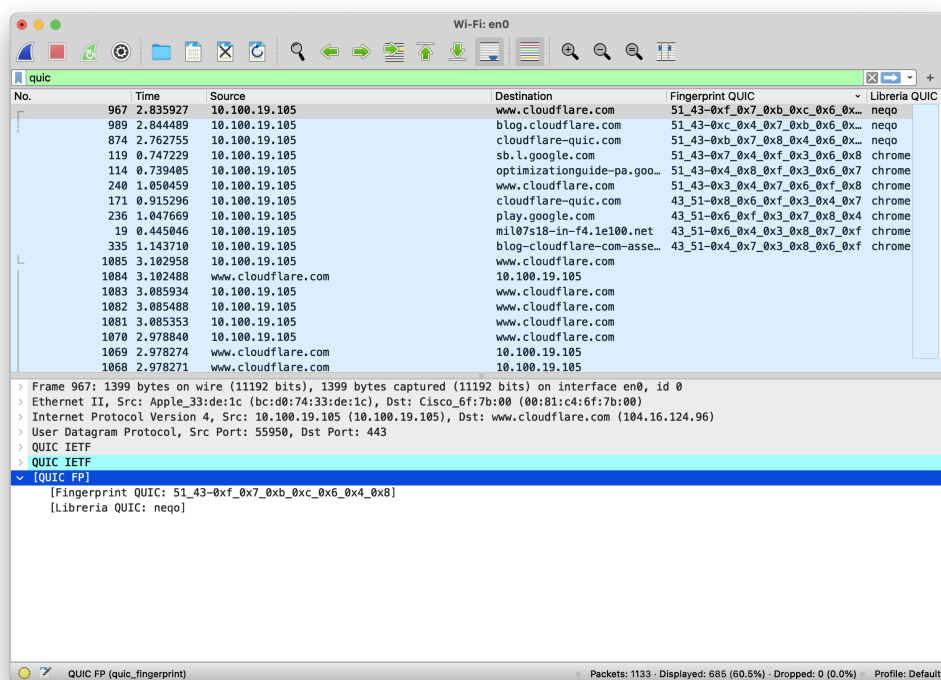
Il campo che contiene il fingerprint è `quic_fingerprint.str`, mentre `quic_fingerprint.library` il nome della libreria.

È consigliato aggiungerli nelle colonne di Wireshark per visualizzare e filtrare con facilità, andando su "Preferences..." e facendo quanto segue:

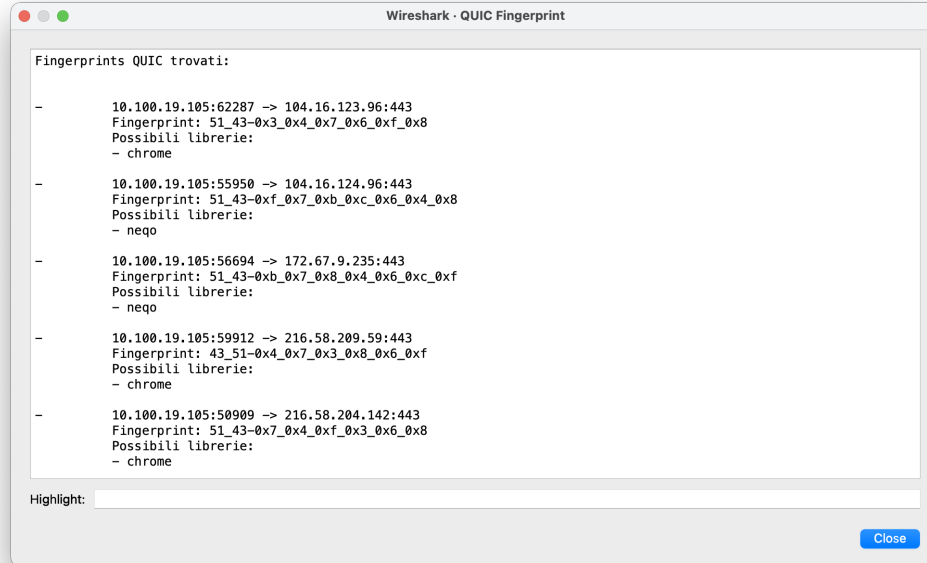


¹`quic_fingerprint.lua`, nella cartella del progetto

Di seguito uno screenshot di come vengono visualizzate le colonne. I campi vengono anche aggiunti nel dissection tree, selezionando il primo pacchetto dell'handshake:



Inoltre andando su Tools > QUIC Fingerprint > Mostra fingerprints è possibile visualizzare tutti i fingerprint acquisiti dal plugin:



3.3 Fingerprint

Per il fingerprint sono considerate l'ordine di alcune estensioni TLS e l'ordine e l'insieme dei transport parameters, nel formato [Estensioni TLS]-[Transport parameters]. Ecco un esempio:

$$\underbrace{43_51}_{\text{Estensioni TLS}} - \underbrace{0x3_0x4_0x6_0x7_0x8_0xa_0xb_0xf}_{\text{Transport parameters}}$$

Per quanto riguarda le estensioni TLS, viene considerato l'ordine di **supported_versions** (43) e **key_share** (51), in quanto sono le uniche estensioni che devono essere sempre presenti in TLS 1.3.

Per i transport parameters, viene considerato solo l'ordine delle estensioni utilizzate e inviate dai client QUIC, escludendo quelle del server.

4 Testing

La validità del fingerprint è stata testata catturando l'handshake con diversi client QUIC, molte volte, per controllare se qualche client rende causale l'ordine di uno dei campi del fingerprint.

Per facilitare tutto ciò è stato molto utile QUIC-Interop-Runner (QIR)² [1]: esso viene utilizzato dall'IETF QUIC Working Group per i test di interoperabilità delle implementazioni QUIC. Esso supporta numerose implementazioni client/server.

Il runner testa ogni implementazione client con ogni implementazione server diverse volte al giorno eseguendo diversi check, salvando per ognuno la cattura del traffico in un file pcap. Essi possono essere quindi utilizzati con Wireshark per testare il fingerprinting.

4.1 Script

Per poter testare in modo automatico i vari client, è stato creato uno script³ bash che per ogni implementazione client QUIC scarica un pcap con un handshake, lo passa a tshark e mostra la riga del dissection tree con il fingerprint.

4.2 Risultati

Nella Tabella 1 sono mostrati i risultati ottenuti con l'esecuzione dello script.

Almeno per quanto visto da questi client, il fingerprint è consistente per tutti tranne Chrome e Neqo, in quanto randomizzano l'ordine dei transport parameters, e anche l'ordine delle estensioni TLS per quanto riguarda Chrome. Si può ovviare a questo problema ordinando i transport parameters.

Anche ordinando i transport parameters il fingerprint è univoco, tranne per una permutazione dei transport parameters di Chrome, che va in collisione con Quinn. Per distinguere i due client basterebbe catturare altri handshake e controllare se muta l'ordine dei transport parameters.

²<https://interop.seemann.io>

³test.sh, nella cartella del progetto

Implementazione	Estensioni	Transport parameters
quic-go	43-51	0x6-0x7-0x4-0x8-0x3-0xb-0xc-0xf
ngtcp2	51-43	0xf-0x7-0x4
mvfst	43-51	0x6-0x7-0x4-0x8-0xa-0x3-0xf
quiche	51-43	0x3-0x4-0x6-0x7-0x8-0xa-0xb-0xc-0xf
kwik	43-51	0x3-0x4-0x6-0x7-0x8-0xa-0xb-0xf
picoquic	51-43	0x4-0x8-0x3-0x6-0x7-0xb-0xf
aioquic	51-43	0x4-0x6-0x7-0x8-0xa-0xb-0xf
msquic	43-51	0x3-0x4-0x6-0x7-0xa-0xb-0xf
xquic	43-51	0x3-0x4-0x6-0x7-0x8-0xc-0xf
lsquic	51-43	0x4-0x7-0x8-0xf
quinn	43-51	0x3-0x4-0x6-0x7-0x8-0xf
s2n-quic	43-51	0x4-0x6-0x7-0x8-0xf
go-x-net	43-51	0x3-0x4-0x6-0x7-0xc-0xf
chrome	43-51	set(0x3, 0x4, 0x6, 0x7, 0x8, 0xf)
	51-43	
neqo	51-43	set(0x4, 0x6, 0x7, 0x8, 0xb, 0xc, 0xf)

Tabella 1: Fingerprint di diversi client QUIC. Gli ultimi 2 randomizzano l'ordine dei transport parameters

Riferimenti bibliografici

- [1] Marten Seemann e Jana Iyengar. «Automating QUIC Interoperability Testing». In: *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*. EPIQ '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 8–13. ISBN: 9781450380478. DOI: 10.1145/3405796.3405826. URL: <https://doi.org/10.1145/3405796.3405826>.
- [2] Johannes Zirngibl et al. «QUIC Hunter: Finding QUIC Deployments and Identifying Server Libraries Across the Internet». In: *Passive and Active Measurement Conference (PAM)*. Mar. 2024.