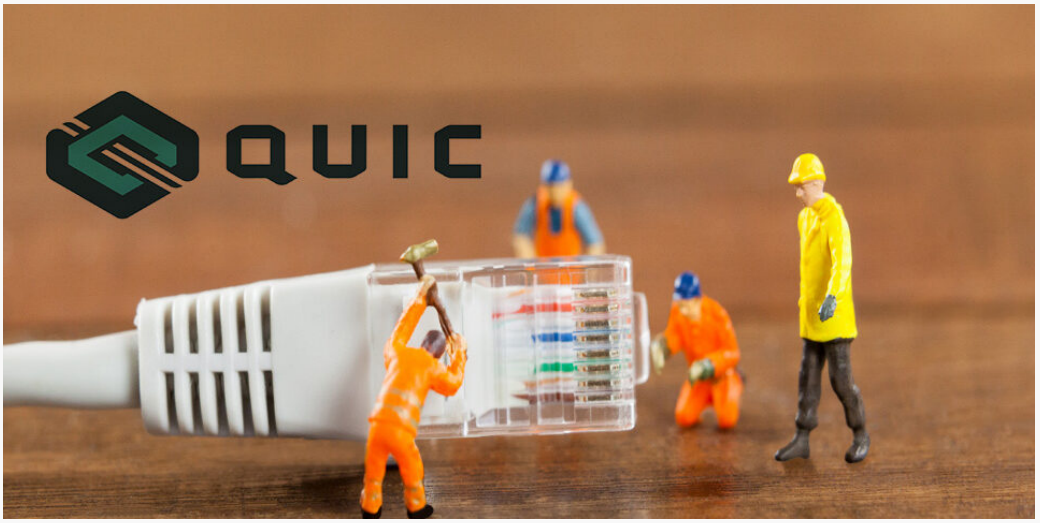
redazione@ictsecuritymagazine.com

QUIC: un nuovo protocollo Internet e la sicurezza IT

A cura di: Andrea Pasquinucci - Pubblicato il 3 Marzo 2021



E' possibile che tra non molto cambi buona parte del traffico Internet, a partire da quella generata dai Browser Web. Questo è dovuto al futuro arrivo di un nuovo protocollo di comunicazione chiamato QUIC (inizialmente era l'acronimo di "Quick UDP Internet Connections", ma ora è solo un nome, si vedano Rif. 1 e 2) che sarà inizialmente utilizzato dalla terza versione di HTTP, HTTP/3 (Rif. 1).

In questo breve articolo si cerca di descrivere ancor più brevemente QUIC, la sua collocazione tra i protocolli di rete, i principali motivi che hanno portato alla sua creazione, le principali novità che introduce e le ripercussioni che potrebbe avere per la sicurezza delle comunicazioni.

HTTPS e lo stack di rete

Per chiarire dove si colloca QUIC e quale è il suo scopo, è meglio partire da un breve ripasso di come è organizzata una connessione HTTPS secondo la pila ISO/OSI [1]:

Livello	Protocollo	Note
7. Application	HTTP	L'applicazione Web che utilizza HTTP spesso stabilisce una sessione ad esempio a seguito di un accesso e utilizzando Cookies
ion	TLS	Viene stabilita una sessione

Livello	Protocollo	Note
4. Transport	TCP	Viene stabilita una sessione
3. Network	IP	

Tabella 1: HTTPS nella pila ISO/OSI

Una vista forse più interessante è quella che considera il processo di incapsulamento dei dati quando passano dall'applicazione sino al livello fisico di trasmissione come descritto sommariamente in Fig. 1:

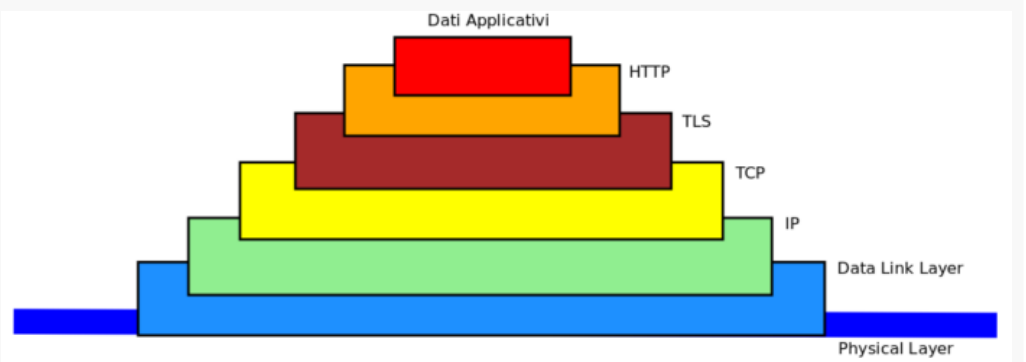


Figura 1. Incapsulamento dei dati per HTTPS.

L'incapsulamento nei vari protocolli ai differenti livelli è sicuramente il motivo della fantastica duttilità delle comunicazioni IT ove ad ogni livello, eccetto al più al livello di Network IP, può essere utilizzato un protocollo diverso a seconda dell'applicazione, del tipo di comunicazione o di rete fisica. Ad esempio, in Fig.1 si è riportato il caso di HTTPS, ma un diagramma simile per HTTP è identico a parte l'esclusione di TLS.

Da HTTP/1.1 a QUIC

Non è detto però che l'incapsulamento, a cui non si vuole e non si può rinunciare, costituisca sempre la soluzione più efficace, efficiente e semplice da implementare. Questo è maggiormente possibile quando sono utilizzati dei protocolli generici, ovvero non orientati e adattati specificatamente all'applicazione che li utilizza.

Già in Tabella 1 è riportato un primo segnale di inefficienza: una comunicazione Web verso un'applicazione tipicamente richiede la gestione di sessioni, ovvero la gestione dello stato della comunicazione, a livello Applicativo, TLS e TCP. Visto che la sessione applicativa deve essere gestita direttamente dall'applicazione, una possibile idea per semplificare la comunicazione potrebbe essere di unificare la gestione delle sessioni TLS e TCP.

In realtà il problema pratico che forse ha contribuito di più alla nascita di QUIC nel 2012 da parte Jim Roskind (Google) è quello chiamato tecnicamente come "*multiplexing without head-of-line blocking*", ovvero un problema di Disponibilità.

Una descrizione semplificata del problema è la seguente. Una pagina Web tipicamente contiene molti elementi, ad esempio immagini, codice Javascript, Frame HTML ecc. ognuno dei quali ha un proprio indirizzo (URL). La soluzione più semplice è di far richiedere dal Client Web e far inviare dall'applicazione i componenti in serie, uno dopo l'altro, utilizzando la stessa sessione TCP (e TLS). Questo però sicuramente non è efficiente. Si possono stabilire sessioni TCP (e TLS) parallele per mponenti della pagina, ma queste comportano un ulteriore carico sui server Web, che anzi alleggerire.

modalità di inserimento dati nel Fascicolo sanitario elettronico
1 Marzo 2021


La de-anonimizzazione dei dati personali. Il caso del dataset Netflix
24 Febbraio 2021


Sicurezza Web e Web Application Firewall
17 Febbraio 2021

In HTTP/1.1 è possibile utilizzare il metodo *Pipelining*, ovvero l'invio dal client di molteplici richieste GET in sequenza senza attendere i dati di risposta di ognuna. L'uso di questo metodo dovrebbe velocizzare la trasmissione dei dati ma in pratica l'implementazione nei Browser è risultata problematica e poco efficiente per cui la quasi totalità dei Browser non lo utilizza.

HTTP/2 ha introdotto la possibilità di gestire molteplici *Stream* di richieste e di trasferimenti dati all'interno della stessa connessione TCP (e TLS) utilizzando *Multiplexing* ed altre ottimizzazioni quali la compressione degli *Header* con HPACK (Rif. 5) [2]. HTTP/2 permette quindi che si instaurino tra server e client sullo stesso canale di comunicazione molteplici e concorrenti sessioni di comunicazione [3]. Tipicamente l'utilizzo di HTTP/2 permette di velocizzare il caricamento di una pagina Web dal 10% al 50%.

Perché QUIC

HTTP/2 non risolve però il problema del "*head-of-line blocking*" o più precisamente del "TCP *head-of-line blocking*". Come indicato, HTTP/2 utilizza una sola connessione TCP e nel caso si perda un pacchetto TCP durante la trasmissione, tutti gli Stream HTTP/2 sono bloccati e devono essere ritrasmessi almeno in parte. La resilienza di TCP e la capacità di recuperare una sessione interrotta, in questo caso generano un problema di efficienza.

QUIC si prefigge i seguenti cinque obiettivi principali:

1. Implementare il "*multiplexing without head-of-line blocking*";
2. Minimizzare i tempi di creazione di una connessione e di trasporto dei dati per qualunque applicazione, prendendo HTTP come applicazione d'esempio;
3. Potersi implementare solo a livello applicativo, ovvero senza cambiare protocolli di rete e sistemi operativi;
4. Permettere la comunicazione con percorsi diversi su rete (es. il cambio dinamico di indirizzi IP del client) e la gestione degli errori di comunicazione;
5. Garantire sempre la sicurezza delle comunicazioni, per Confidenzialità e Integrità, tramite l'adozione ed inclusione di TLS (specificatamente al momento TLS-1.3).

QUIC non è una nuova versione di HTTP, e la sua collocazione nella pila dei protocolli di rete è descritta sommariamente in Fig. 2:

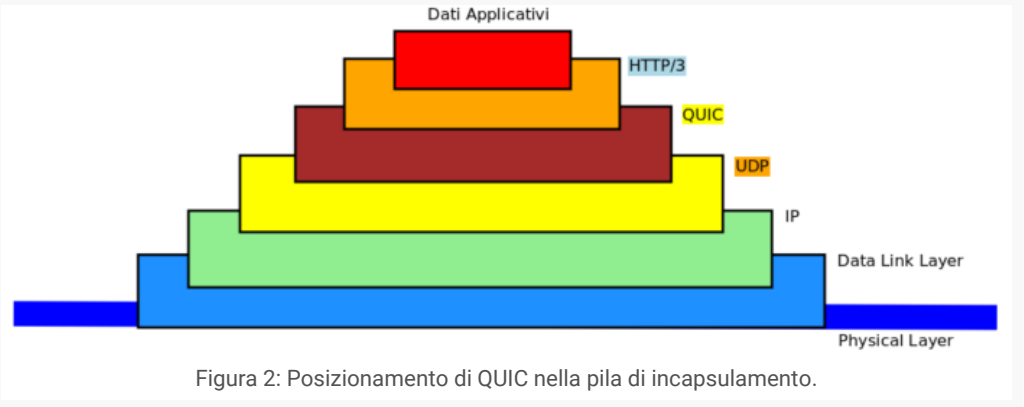


Figura 2: Posizionamento di QUIC nella pila di incapsulamento.

QUIC è posizionato a livello "6. Presentation" al posto di **TLS**, utilizza **UDP** a livello "4. Transport" ed in questo esempio è richiamato da **HTTP/3** a livello "7. Application". HTTP/3 è l'evoluzione di HTTP/2 in grado di utilizzare tutte le funzionalità di QUIC: in particolare alcune funzionalità di HTTP/2 sono ora implementate da QUIC. HTTP/3 sarà solo il primo, e probabilmente il più importante, protocollo ad

C, ma altri protocolli o applicazioni potranno implementarlo e utilizzarlo.

a osservazione è che QUIC, come dal terzo requisito, è implementabile a livello applicativo, anche tramite librerie dedicate. Anzi ci si aspetta che librerie che oggi forniscono TLS (es. Openssl)

implementino anche QUIC con TLS (oggi TLS-1.3).

La seconda osservazione è che QUIC utilizza UDP e non TCP, quindi un protocollo di trasporto che non gestisce la connessione, ogni pacchetto è inviato e non viene tracciato il suo stato, ovvero se è ricevuto correttamente ed in quale ordine. La gestione della connessione è fatta da QUIC che integra la gestione di TLS e della sicurezza del trasferimento dei dati con la gestione di *Stream* concomitanti (come in HTTP/2).

La struttura di QUIC

La struttura di un pacchetto dati trasferito da QUIC è abbastanza complessa:

- 1 Datagramma UDP contiene 1 o più QUIC Packet;
- Ogni QUIC Packet contiene 1 o più Frame;
- Ogni Frame contiene dati di 1 Stream;
- Ogni Connessione ha 1 o più Stream.

Tutti i dati sono protetti con TLS, i dati applicativi sono sempre cifrati mentre i dati necessari per la comunicazione sono verificati solo per l'Autenticità e l'Integrità [4]. La Figura 3 cerca di descrivere questa organizzazione dei dati di QUIC:

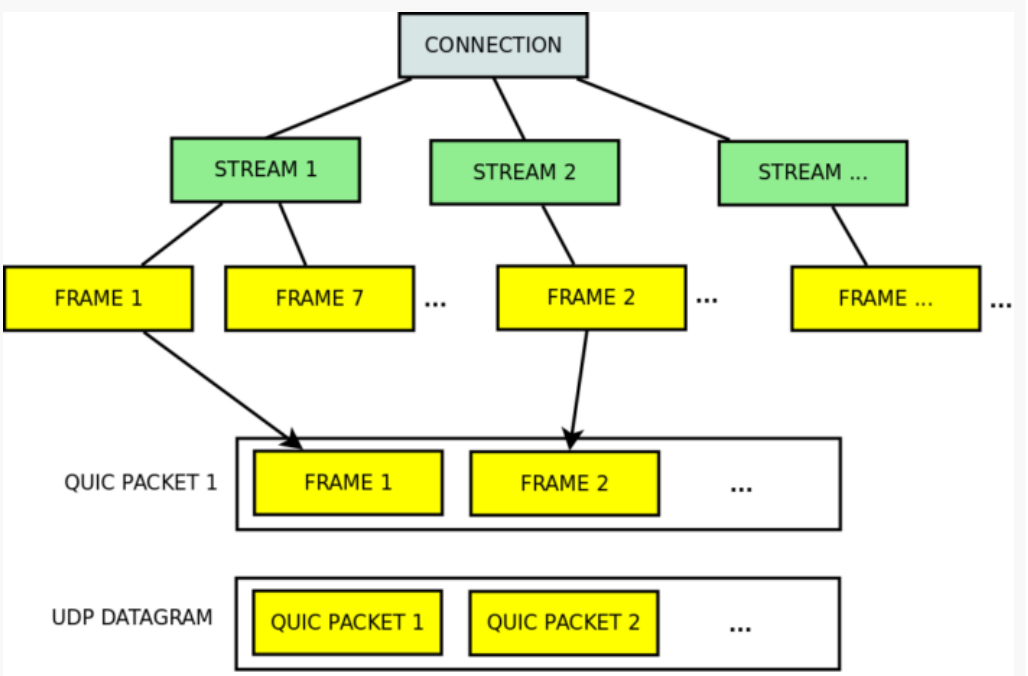


Figura 3: organizzazione dei dati in un pacchetto di rete secondo il protocollo QUIC

QUIC suggerisce che le applicazioni inseriscano in un Datagramma UDP sono QUIC Packet contenenti Frame che appartengono allo stesso Stream, questo per risolvere il problema del "*multiplexing with head-of-line blocking*". QUIC lascia però la possibilità a chi utilizza il protocollo di mischiare Frame e QUIC Packet di Stream diversi (vedi Fig. 3) nel caso in cui sia ben chiaro allo sviluppatore che se si perde il pacchetto UDP due o più Stream saranno impattati e dovranno inviare di nuovo i dati. Vi possono essere casi, soprattutto se i Frame sono piccoli, in cui può risultare più conveniente inviare un solo Datagramma UDP con dati di diversi Stream piuttosto che molteplici piccoli Datagrammi UDP ognuno con i dati di un solo Stream.

Ovviamente la macchina a stati di QUIC mantiene lo stato di invio di ogni Frame di ogni Stream

lo un proprio meccanismo di conferma della ricevuta dei dati (ACK o *acknowledgment*) ed nvio, visto che UDP a differenza di TCP non gestisce le connessioni.

La sicurezza dei dati, garantita da TLS (oggi TLS-1.3), è integrata in QUIC in modo che non interferisca con le caratteristiche di Disponibilità ed efficienza nella gestione delle connessioni. Ad esempio, ogni QUIC Packet è cifrato indipendentemente in modo da garantire che ogni Datagramma UDP possa essere decifrato indipendentemente. Questa garanzia non c'è per HTTPS/1.1 e HTTPS/2 perché TLS e TCP agiscono a livelli diversi ed indipendentemente.

Oltre a *Multiplexing*, per velocizzare il trasferimento di dati, QUIC, come già HTTP/2, offre anche la possibilità per l'applicazione (server) di fare *Push* dei dati verso il Browser (client). Il caso della navigazione Web è molto semplice da descrivere. Come già indicato, un Browser richiede ad un server Web una pagina, ma questa è composta da molti elementi, ad esempio immagini e codice Javascript, ognuno dei quali ha un proprio indirizzo (URL). L'applicazione sa che per rendere la pagina, il Browser avrà bisogno non solo del codice HTML richiesto ma anche di tutti questi ulteriori elementi. Per velocizzare il trasferimento, l'applicazione può inviare al Browser tutti gli elementi necessari senza che il Browser li richieda esplicitamente, riducendo quindi il tempo necessario a trasferire tutti i dati.

La complessità di QUIC

Come abbiamo brevemente descritto, QUIC deve fare molte cose contemporaneamente. Ed è proprio il fare tutte queste cose contemporaneamente che gli permetterà di rendere più veloce e, da un certo punto di vista, più semplice, trasferire i dati tra applicazione e dispositivo utente. Un breve riassunto (molto incompleto) delle attività contemporanee di QUIC è:

1. Garantire sempre la sicurezza delle comunicazioni, per Confidenzialità, Integrità e Autenticità, tramite la protezione di tutto il traffico con TLS (TLS-1.3);

1. Gestire un'unica sessione tra client e server di tutti gli Stream paralleli attivi (ad eccezione di eventuali sessioni applicative a seguito di autenticazione e accesso dell'utente al servizio); questo include la gestione degli errori, la ritrasmissione dei dati persi e l'integrazione di TLS;
2. Permettere la comunicazione con percorsi diversi su rete tracciando e gestendo eventuali cambi di indirizzo IP del client (cambio di NAT, cambio di cella telefonica per uno smartphone ecc.) e ri-autenticando automaticamente e molto velocemente il client con il supporto di TLS [5];
3. Gestire la congestione delle comunicazioni (*Congestion Control*): la perdita di molti pacchetti UDP è di norma dovuta ad una linea lenta e congestionata; QUIC quindi, soprattutto lato server, può rallentare l'invio di dati per diminuire la perdita di pacchetti;
4. Gestire tutte le componenti di trasmissione dati che possono essere impattate dall'adozione di QUIC, ad esempio Proxy Web e Load Balancer in caso di cambio dinamico di indirizzi IP del client.

In pratica questo richiede di re-implementare parte di HTTP/2, tutto TLS e buona parte di TCP in un unico standard.

Questa complessità si riflette direttamente nel lavoro del QUIC Working Group IETF (Rif. 1) che sta preparando ben nove standard e l'ultima versione, al momento di scrivere queste note, numero 34, consiste di 510 pagine in totale.

Vista la complessità, lo sviluppo dei protocolli va di pari passo con l'implementazione ed i test da parte di un consorzio che comprende praticamente tutti i grandi produttori di Browser e HTTP Server (o, più precisamente, delle librerie utilizzate da Browser e HTTP Server). Vengono verificati sia i protocolli sia l'interoperabilità delle implementazioni (Rif. 2). Lo sviluppo dei protocolli è quindi incrementale con un approccio che si può definire "*Agile*" e che ha lo scopo di produrre alla fine un protocollo efficace, efficiente e corretto insieme alle relative implementazioni.

Il lavoro del QUIC Working Group IETF è incominciato nel 2016, dopo i primi sviluppi di Google a partire dal 2012, e si prevede che termini tra la fine del 2021 e l'inizio del 2022. Quindi, a 10 anni dalla prima ossibile che questo protocollo diventi standard e cominci ad essere utilizzato nte da tutti noi. E' probabile che i tempi di adozione saranno molto veloci visto che sia i Server Web già lo implementano e lo utilizzano nelle versioni di sviluppo e produzione (ma non abilitato per default).

Implementazioni e prestazioni

Come già indicato, QUIC viene al momento utilizzato per le connessioni Web, ovvero HTTP. Browser e Server Web saranno però per un lungo periodo retro-compatibili con HTTP/1.1 e HTTP/2. Il motivo più semplice è che QUIC utilizza la porta UDP/443 mentre HTTP/1.1 e HTTP/2 utilizzano TCP/80 e TCP/443. Nel caso la porta UDP/443 sia chiusa, o vi sia un disallineamento tra client e server, o uno dei due non supporti QUIC, Browser e Server Web ritornano ad utilizzare i protocolli esistenti. Questo è anche il motivo per cui facendo oggi delle prove, come descritto di seguito, si vede che spesso componenti di pagine Web sono scaricati con QUIC, altri con HTTP/1.1 o HTTP/2.

Per quanto riguarda le prestazioni, una rapida ricerca in Internet riporta risultati molto contraddittori. In alcuni casi sono state riscontrate prestazioni nettamente migliori, ad esempio Rif. 6 riporta prestazioni migliori di circa 10% di QUIC rispetto a HTTP/2 e Rif. 7 prestazioni migliori dal 10% al 30% di QUIC rispetto a HTTP/1.1 (mentre in questo caso HTTP/2 si comporta peggio di entrambi). Google, Rif. 8, riporta un miglioramento rispetto a HTTP/1.1 del 2% in Google Search, 9% in Youtube, 3% sui Client Desktop e 7% sui Client Mobile. In altri casi sono stati rilevate prestazioni praticamente uguali o leggermente peggiori di QUIC rispetto a HTTP/1.1 o HTTP/2.

Vi sono vari motivi per queste discrepanze, innanzitutto ci sono molte diverse versioni di sviluppo di QUIC, sia quelle standard IETF sia quelle Google, ed essendo appunto versioni di sviluppo contengono funzionalità diverse o implementate diversamente proprio per verificarne le prestazioni ed il comportamento. Inoltre lo standard non è ancora completato e uno dei principali scopi dei lavori in corso è proprio quello di eliminare le inefficienze degli algoritmi e protocolli. Infine l'aspettativa è che QUIC sia molto più performante dei protocolli attuali non in tutte le situazioni ma in quelle più critiche e problematiche; che riesca come prima cosa a ridurre il carico dei Server Web e di conseguenza possa migliorare anche le prestazioni dei Browser sui Client. Va però segnalato che in condizioni di utilizzo normali l'utente non noterà praticamente alcuna differenza essendo il guadagno troppo piccolo per potersi notare. Quello che dovrebbe invece succedere è che, in situazioni non ottimali di connessione, la navigazione sarà più fluida, con meno blocchi o rallentamenti.

QUIC e Sicurezza IT

Vi sono almeno quattro aspetti da considerare dal punto di vista della sicurezza IT per QUIC:

1. La sicurezza del protocollo;
2. La sicurezza delle implementazioni;
3. La sicurezza nelle interazioni con gli altri componenti della rete IT;
4. La sicurezza fornita nelle comunicazioni IT.

Molti studi sono stati fatti anche di recente (si vedano ad esempio Rif. 9 e 10) sulla sicurezza del protocollo e della sua implementazione di TLS, questo a garanzia delle caratteristiche di Confidenzialità, Integrità e Autenticità dei dati. Si può dire che almeno a livello teorico, il protocollo è nato ed è stato sviluppato mantenendo la sicurezza come uno dei principali requisiti. Ci si aspetta quindi che non vi saranno problemi di sicurezza dovuti al protocollo stesso.

Diverso è il problema delle implementazioni. Come abbiamo descritto il protocollo è molto ampio e complesso, comprendendo allo stesso tempo parte di HTTP/2, tutto TLS e buona parte di TCP. Tenendo conto che tutti gli aspetti di sicurezza delle comunicazioni sono gestite da QUIC, quanti e quali saranno i fraintendimenti o gli errori di implementazione che avranno ricadute dirette sulla sicurezza? Speriamo nessuno, ma la storia dello stack TCP/IP, nato 47 anni fa, e le vulnerabilità che appaiono ancora, per fortuna raramente, non ci induce ad essere del tutto ottimisti (si vedano ad esempio le recenti vulnerabilità descritte in Rif. 11). D'altra parte visto che QUIC è implementato in librerie applicative o direttamente nelle applicazioni e non nei sistemi operativi, fa sperare che il processo di patching e

essere semplice e veloce. Purtroppo vi saranno molte implementazioni diverse, il che chio che qualcuna possa avere delle vulnerabilità che impattano anche pochi sistemi, e casi il processo di patching e update possa essere complesso o lento o non applicato. Come ben noto, il problema principale di patching e update di sicurezza non sono le applicazioni Server,

i Browser Web o le applicazioni Desktop, ma gli infiniti sistemi IoT che sono difficilmente se non impossibili da aggiornare.

QUIC è un nuovo protocollo di rete, e quindi deve essere recepito in tutti i dispositivi di rete che interagiscono dal livello 4 "Transport" in su. La prima osservazione è che QUIC utilizza UDP/443 mentre HTTP ha sempre utilizzato TCP/80 e TCP/443. Quindi i Firewall dovranno essere configurati per permettere il traffico sulla porta UDP/443. Questo fornisce anche un modo semplice per bloccare QUIC, basta bloccare la porta UDP/443 (facendo però attenzione a che non sia utilizzata da altre applicazioni, come ad esempio OpenVPN).

In generale comunque tutti i dispositivi di rete che agiscono sul traffico dal livello 4 "Transport" in su, come Firewall, IPS, ProxyWeb, WAF, Bilanciatori ecc., dovranno essere aggiornati in modo da riconoscere il traffico QUIC e gestirlo di conseguenza. Ovviamente questa è una problematica solo transitoria e che se la porta UDP/443 è chiusa, non porterà alcun cambiamento, cioè nessun utilizzo di QUIC, sino all'aggiornamento e riconfigurazione dei sistemi di rete.

Infine ci si aspetta che adottando QUIC ci sarà anche un qualche impatto positivo sulla sicurezza delle comunicazioni IT rispetto alla situazione attuale. L'utilizzo sempre di TLS garantisce che tutte le comunicazioni siano sempre protette per Confidenzialità, Integrità ed Autenticità mentre il protocollo stesso è stato disegnato per migliorare le caratteristiche di Disponibilità delle comunicazioni soprattutto in situazioni non ottimali o critiche. Inoltre QUIC potrà essere utilizzato da qualunque applicazione, non solamente per il traffico Web e HTTP, possibilmente ampliando il numero di applicazioni e i tipi di informazione che saranno protetti.

Parafrasando una nota affermazione, possiamo dire che QUIC mira a implementare la sicurezza nelle comunicazioni IT "*by design and by default*". D'altra parte va anche evidenziato che rispetto ad un attuale corretto uso di HTTP+TLS+TCP, i benefici saranno ottenuti più dai fornitori di servizi IT che dagli utenti.

Provare QUIC

Per concludere, è possibile già oggi utilizzare QUIC in via sperimentale. Tutti i grandi fornitori di servizi Web hanno già abilitato in via sperimentale QUIC sui propri servizi. Anche i principali Browser Web supportano QUIC, ma non è abilitato per default.

Ad esempio rifacendosi a Google, che comunque è sempre il padre del progetto, si può procedere come segue:

- accertarsi che la porta UDP/443 sia aperta dal proprio dispositivo verso Internet
- aprire il browser Chrome (o Chromium)
- aprire la pagina "*chrome://flags/*", cercare "QUIC" e metterlo "Enabled" (il "Default" è "Disabled")
- dal menu aprire i "Developer Tools" (o CTRL-SHIFT-I), selezionare "Network" ed assicurarsi che la colonna "Protocol" sia presente nel frame in basso (altrimenti cliccare con il tasto destro sul nome di una delle colonne presenti ed aggiungerla)
- infine visitare una pagina di Google, come nell'esempio in Fig. 4 in cui la colonna "Protocol" riporta l'indicazione h3-Q50 che secondo Rif. 8 vuol dire che è stato utilizzato HTTP/3 su Google-QUIC-50 (una versione sperimentale di QUIC di Google).

Una procedura simile può essere utilizzata con un altro Browser e su altri siti già abilitati a QUIC.

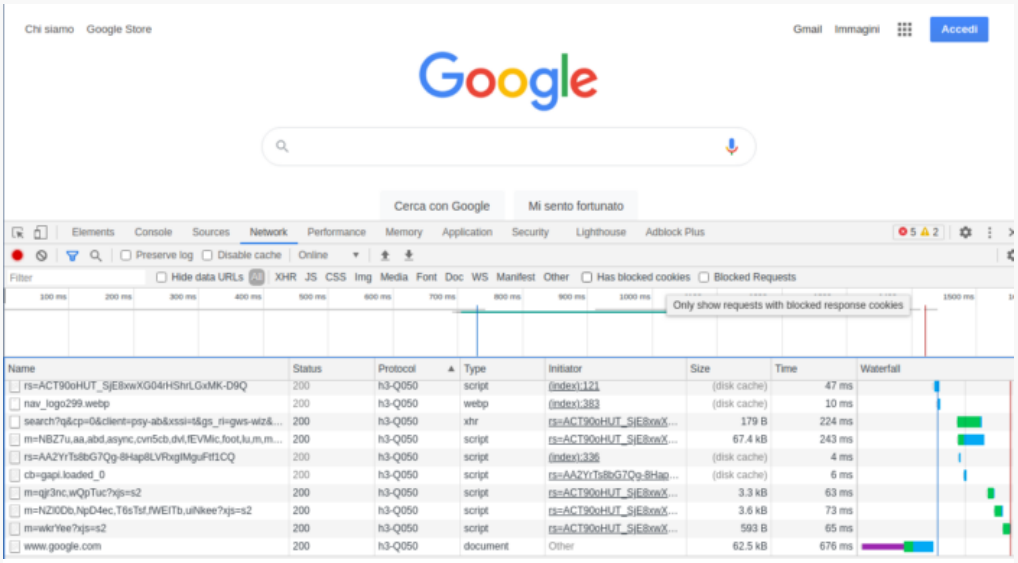


Fig. 4 Esempio di navigazione Web con QUIC sulla pagina di Google Search

Riferimenti Bibliografici

Rif. 1: IETF QUIC Working Group, https://quicwg.org/ e https://datatracker.ietf.org/wg/quic/about/

Rif. 2: Implementazioni di QUIC disponibili a https://github.com/quicwg/base-drafts/wiki/Implementations , l'interoperabilità è descritta in https://interop.seemann.io/

Rif. 3: HTTP/2 RFC 7540, https://tools.ietf.org/html/rfc7540 (ed anche https://en.wikipedia.org/wiki/HTTP/2)

Rif. 4: per ISO/OSI si veda anche https://en.wikipedia.org/wiki/OSI_model e https://en.wikipedia.org/wiki/List_of_network_protocols_(OSI_model)

Rif. 5: HPACK RFC 7541, https://tools.ietf.org/html/rfc7541

Rif. 6: Cloudflare "Comparing HTTP/3 vs. HTTP/2 Performance", https://blog.cloudflare.com/http-3-vs-http-2/

Rif. 7: Uber Engineering "Employing QUIC Protocol to Optimize Uber's App Performance", https://eng.uber.com/employing-quic-protocol/

Rif. 8: Chromium Blog "Chrome is deploying HTTP/3 and IETF QUIC". https://blog.chromium.org/2020/10/chrome-is-deploying-http3-and-ietf-quic.html

Rif. 9: Samuel Jero "QUIC: Performance and Security at the Transport Layer", https://www.ietfjournal.org/quic-performance-and-security-at-the-transport-layer/

Rif. 10: QUIC Privacy and Security Workshop 2020, https://www.ndss-symposium.org/ndss2020/quic-privacy-and-security-workshop/

Rif. 11: Forescout "Number Jack – Weak ISN Generation in Embedded TCP/IP Stacks", https://www.forescout.com/company/resources/numberjack-weak-isn-generation-in-embedded-tcpip-stacks/

Note

[1] Il livello "5. Session" non è utilizzato in questa descrizione di HTTPS, si veda anche Rif. 4., e non esplicitamente i livelli inferiori "1. Physical" e "2. Data Link".

ientazione di HPACK in QUIC è chiamata QPACK ed è uno degli standard RFC QUIC.

[3] HTTP/2 non richiede l'utilizzo di TLS, ma la pratica comune dei Browser è di utilizzarlo solo insieme a TLS.

[4] I dati nei pacchetti sono cifrati utilizzando il protocollo "Authenticated Encryption with Associated Data" (AEAD) che protegge non solo i dati trasportati ma anche gli *Header* dei pacchetti.

[5] QUIC utilizza un protocollo di rinegoziazione molto veloce chiamato 0-RTT.

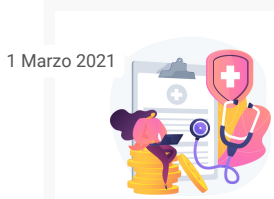
Articolo a cura di **Andrea Pasquinucci**

Autore | Bio

Andrea Pasquinucci

PhD CISA CISSP

Ti potrebbe interessare



1 Marzo 2021

Intervento del Garante Privacy e chiarimenti sulle modalità di inserimento dati nel Fascicolo sanitario elettronico

24 Febbraio 2021

La de-anonimizzazione dei dati personali. Il caso del dataset Netflix

17 Febbraio 2021

Sicurezza Web e Web Application Firewall

Segreteria: Humana Srls
C.F e P.IVA: 13642431004
redazione@ictsecuritymagazine.com

ENISA 5G

Intelligenza Artificiale

edge computing awareness

tutela dei dati personali