1. Introduzione al protocollo HTTPS

La maggior parte dei siti web utilizzano il protocollo HTTPS, ovvero il risultato ottenuto fra la congiunzione del protocollo TLS e HTTP. Il protocollo HTTPS verifica l'identità di un sito web e crittografa le informazioni inviate tra il sito web e il client.
La comunicazione con client e server avviene tramite l'utilizzo di una **chiave di sessione**. Durante la prima fase di connessione fra client e server, il browser crea una (nuova) **chiave di sessione** che verrà utilizzata per codificare e decodificare i dati scambiati fra questi ultimi. Una volta creata la chiave, il browser ne invia una copia al server. Prima di essere inviata, la chiave viene codificata tramite l'utilizzo della chiave pubblica del server stesso. In questo modo solo il server potrà, tramite l'utilizzo della chiave privata, decodificare il messaggio e quindi ottenere la chiave di sessione.

2. Analizzare il traffico con WireShark

WireShark è un tool multipiattaforma che consente di intercettare il traffico dati che transita su una specifica interfaccia di rete, fornendo informazioni riguardanti lo scambio di pacchetti dati.
In questo contesto WireShark è stato utilizzato per analizzare il traffico HTTP e quindi analizzare quali sono le versioni HTTP utilizzate dai vari siti. Come già detto in precedenza, molti siti utilizzano il protocollo HTTPS per permettere lo scambio di informazioni in maniera sicura. Questo potrebbe risultare un ostacolo al raggiungimento dell'obiettivo, in quanto è sempre possibile in WireShark analizzare i pacchetti scambiati in una comunicazione HTTPS ma quest'ultimi non risultano essere leggibili in quanto cifrati. Esiste però una soluzione a questo problema. Durante una comunicazione HTTPS abbiamo detto che il browser genera la chiave di sessione che viene utilizzata per cifrare e decifrare i dati. Tutte le chiavi di sessione generate vengono inserite all'interno di un file di log specificato all'interno della variabile globale **SSLKEYLOGFILE.** All'interno di WireShark è possibile impostare questo file di log in modo da decifrare i pacchetti TLS e ottenere le informazioni che a noi interessano.

3. Analizzare il traffico con Mitmproxy

Mitmproxy è un server proxy interattivo open source, dotato di interfaccia CLI (mitmproxy) e GUI (mitmweb), oltre che di un set di API Python. Nell'ambito di questo assignment, mitmproxy ha consentito di analizzare il traffico dati su tutte le interfacce di rete disponibili sulle macchine impiegate, attraverso poche semplici modifiche alle impostazioni di rete di queste ultime. Similmente a Wireshark, Mitmproxy consente di recuperare anche i pacchetti HTTPS, riuscendo tuttavia a leggerne il contenuto senza richiedere l'utilizzo di chiavi per la decrittazione. Un server proxy, infatti, funge da intermediario tra l'utente ed il web, inoltrando le richieste dell'utente ai server e le risposte di questi ultimi all'utente. Durante questo processo, richieste e risposte vengono dapprima decifrate dal proxy server, che quindi può leggerne il contenuto, e successivamente cifrate prima dell'inoltro all'effettivo destinatario.

4. Descrizione dell'obiettivo

L'obiettivo di questo assignment è studiare, tramite l'analisi del traffico, la versione di protocollo HTTP utilizzato dai 50 siti web più popolari secondo Alexa.

5. Definizione del workflow

5.1. Soluzione implementata con Selenium e Wireshark

La fase iniziale comprende la cattura dei pacchetti che vengono scambiati fra il client e il server durante la connessione ad ogni singolo sito web. Questa fase è stata del tutto automatizzata tramite l'utilizzo di Selenium. Tramite Selenium, una volta avviata la cattura dei pacchetti su WireShark, vengono visitati tutti i 50 siti riportati sul sito di Alexa. Al termine di questa operazione tutti i pacchetti scambiati saranno catturati all'interno di un file .pcapng. Esso verrà analizzato tramite l'utilizzo della libreria **pyshark** per il linguaggio **python**, in modo da automatizzare il processo di analisi e quindi di determinazione della versione HTTP adottata dai diversi siti Web. Per ogni sito web analizzato l'algoritmo controlla:

1. La lista di indirizzi IP che corrispondo al sito web;
2. Se all'interno del file .pcapng sono stati catturati pacchetti QUIC (protocollo utilizzato dalla versione HTTP/3) che hanno come destinazione uno degli indirizzi IP corrispondente al sito web l'algoritmo stabilisce che il sito web utilizza il protocollo HTTP/3 e procede con l'analisi del sito web successivo, altrimenti passa alla fase 3;
3. Se all'interno del file .pcapng sono stati catturati pacchetti HTTP2 che hanno come destinazione uno degli indirizzi IP corrispondente al sito web l'algoritmo stabilisce che il sito web utilizza il protocollo HTTP/2, altrimenti, per esclusione, stabilisce che il sito web utilizza il protocollo HTTP/1.1. In entrambi i casi l'algoritmo procede con l'analisi del successivo sito web.

Una volta terminata l'analisi dei diversi siti web l'algoritmo restituisce l'output sotto forma di file CSV.

5.2. Soluzione implementata con Selenium e Mitmproxy

Similmente alla precedente soluzione, la fase iniziale ha previsto la cattura dei pacchetti da analizzare. Il processo di connessione ai 50 siti web è stato completamente automatizzato tramite Selenium. Per l'intercettazione del traffico è stata utilizzata la versione GUI di Mitmproxy, avviabile utilizzando il comando "mitmweb" nella shell del sistema operativo. Una volta avviato il proxy, sulla shell compare l'indirizzo IPv4 da visitare per accedere alla GUI e l'indirizzo IPv4 sul quale viene effettuato l'ascolto. A questo punto è necessario impostare l'indirizzo del server proxy tramite le impostazioni proxy di sistema o del browser web utilizzati, in modo da ridirezionare il traffico uscente.

Una volta avviato il programma Java, grazie al quale è possibile sfruttare le API di Selenium, tutto il traffico verrà intercettato ed analizzato dal Mitmproxy e, grazie alla GUI, sarà anche possibile utilizzare numerosi filtri per evidenziare o ricercare i dati di nostro interesse.

Uno dei limiti di Mitmproxy è la mancanza di controlli per discriminare l'utilizzo del protocollo QUIC. Effettuando un controllo incrociato con Wireshark, infatti, si è notato che i pacchetti QUIC vengono correttamente ricevuti, ma non correttamente identificati dal server proxy, che li identifica invece come pacchetti HTTP/2.