Alma Mater Studiorum-Università di Bologna
Campus di Cesena

Corso di Laurea in Ingegneria e Scienze Informatiche

Scuola di Scienze

Filtraggio e censura dei servizi Internet
Un'analisi sul protocollo SSL/TLS

Relazione finale in
Programmazione di Reti

Relatore
Andrea Omicini

Presentata da
Andrea Vignudelli

Sessione III, Marzo 2017
Anno accademico 2015/2016

Abstract

Obiettivo ultimo di questa tesi è fornire le competenze necessarie per poter capire come effettuare politiche di filtraggio dei contenuti fruibili su Internet in contesti prevalentemente di network locali come LAN aziendali ma anche su scala maggiore, effettuando politiche di blocco al livello logico 7 della pila protocollare ISO/OSI.
Nello specifico verranno approfondite le specifiche tecniche del protocollo SSL/TLS e su queste basi teoriche verranno implementate e testate regole di blocco di specifici servizi Internet simulando il contesto di una piccola rete domestica con un Firewall di frontiera posto davanti al gateway della rete locale che funge da filtro per la comunicazione con la rete Internet.
Verranno inoltre esposte delle problematiche direttamente correlate a quella che è l'infrastruttura giuridica fisicamente costruita intorno all'accertamento dell'identità digitale degli enti/soggetti all'interno di comunicazioni che sfruttano le funzionalità del protocollo SSL/TLS come strumento per criptare i dati ed avere garanzia sull'identità che si cela sul lato Server della comunicazione, trattando nello specifico alcuni casi storici di brecce informatiche all'interno delle infrastrutture fisiche dei Certificate Authorities, enti predisposti al ruolo di certificatori delle identità digitali sopracitate, o dell'uso improprio da parte loro del potere conferitogli, saranno inoltre esposti e replicati su scala minore esempi di come alcuni enti governativi nazionali sfruttano alcune caratteristiche intrinseche di questo protocollo per bloccare determinati servizi all'interno della sottorete Internet da loro gestita e di come in un contesto storico nel quale venisse a mancare il caposaldo della Network Neutrality, proprio quelle specifiche protocollari potrebbero essere utilizzate per effettuare politiche di gestione prioritaria nell'instradamento dei pacchetti all'interno della rete Internet.

Indice

1 Identificazione della tipologia di traffico, il Transport Layer Security 5

1.1 Introduzione 5

1.2 Metodologie di identificazione, Deep Packet Inspection e disassembly dei dati 6

1.2.1 Wireshark, packet sniffer 7

1.3 Il protocollo SSL/TLS, specifiche 8

1.3.1 Handshake, Cipher suit list, Tunneled stream 8

1.3.2 Stream Cipher e Block Cipher 9

1.3.3 PKI, Certificato digitale, X.509 e Certificate Authority 11

1.4 Exploit e problematiche del protocollo SSL/TLS 13

1.4.1 Timing attacks, RC4 attacks, Cipher attacks, Downgrade attacks, Protocol Bugs, il caso di HeartBleed e Violazioni ai CA 13

1.4.1.1 CRIME 13

1.4.1.2 BREACH 14

1.4.1.3 Cipher RC4 14

1.4.1.4 HeartBleed e l'NSA 14

1.4.1.5 CA compromessi, i casi di DigiNotar, VeriSign e Trustwave 15

1.5 Casi di studio 16

1.5.1 Man In The Middle e Proxy Server 17

2 Dati reperibili da un handshake TLS utili al filtraggio dei servizi ed implicazioni correlate 20

2.1 Fingerprinting Browser Agent/Applicazione 20

2.2 Network Neutrality e fingerprinting sul Certificato 21

2.2.1 Dumb Network, discriminazione per protocollo e Traffic Shaping . 22

3 Filtraggio e blocco, un esempio completo di attacco e difesa, TOR 25

3.1 Infrastruttura Rete 26

3.2 Circuito TOR 27

3.2.1 Onion Routing e Handshake Diffie-Hellman 28

3.3 Debolezze e problemi noti, Great Firewall Of China VS TOR 29

3.3.1 Contromisure adottate, Bridges e Pluggable Transports 31

4 Filtraggio del traffico, architettura Firewall Analisi, implementazione e risultati **33**
4.1 Architettura di rete e software 34
4.1.1 RouterOS 34
4.1.2 Iptables 35
4.1.3 ntopng 36
4.1.4 tc: Traffic Control 36
4.2 Analisi e collezione dei dati 37
4.3 Test 38
4.3.1 Blocco 38
4.3.2 Traffic Shaping 39
4.4 Risultati 39

5 Conclusioni **43**
5.1 Chi custodirà i custodi? PGP come alternativa alla PKI 43
5.2 Network Neutrality 44
5.3 Lavori futuri 46

Riferimenti **47**

CAPITOLO 1

Identificazione della tipologia di traffico, il Transport Layer Security

1.1 Introduzione

All'interno di questa dissertazione vi sarà la descrizione dettagliata della struttura, del funzionamento, dei pro e dei contro sull'impiego del protocollo SSL/TLS, utilizzato per l'instauramento di un canale di comunicazione end-to-end criptato e sicuro tra due end-point (solitamente un client che inizia l'handshake ed un server che risponde) ad un livello direttamente superiore rispetto al livello di trasporto.
La ricerca sarà orientata a fornire gli strumenti necessari a comprendere come in molti casi il filtraggio od il blocco di contenuti e servizi fruibili attraverso la rete Internet sia possibile o addirittura già realizzato in passato o peggio ancora in atto mediante lo sfruttamento di alcune caratteristiche intrinseche del protocollo. Nei capitoli finali di questa tesi invece verranno descritte e testate su base pratica le nozioni apprese attraverso i primi tre capitoli, tutto questo mediante la simulazione di scenari in cui alcuni servizi Internet vengono censurati attraverso router/firewall appositamente configurati per lo scopo, analizzando i componenti software ed hardware impiegati per i vari test, e descrivendo in maniera dettagliata come sono state effettuate le regole per il blocco di specifici servizi Internet e del perché del loro funzionamento.
Alla luce di questo verranno inoltre toccati aspetti come la Network Neutrality, nello specifico proprio di come nel caso venisse a mancare questo principio all'interno della rete Internet, gli stessi strumenti e le stesse nozioni apprese nel corso di questa dissertazione sarebbero sufficienti a permettere di essere in grado di effettuare politiche di instradamento prioritario dei pacchetti differenziando e discriminando in base al tipo di servizio offerto (...e quindi le necessità di reattività nell'invio/ricezione dei pacchetti, esempio applicazione di streaming vs applicazione di messaggistica) per garantire una maggiore QoS (Quality of Service, lett. qualità del servizio) generale della rete Internet.

Nel capitolo iniziale vengono illustrati e descritti nel dettaglio:

- La pratica della Deep Packet Inspection ed il software Wireshark, un packet sniffer che, tra le varie funzionalità che offre, nel nostro caso di studio permette di isolarele interazioni tra client e server mediante il filtraggio in maniera esclusiva dei pacchetti SSL/TLS per avere un focus su quel genere di interazioni.
- un'analisi dettagliata sulle varie specifiche, funzionalità, caratteristiche identificative e contesti d'uso del protocollo SSL/TLS.
- la presentazione ed il confronto di due tecniche usate come casi di studio per intromettersi all'interno di una connessione SSL/TLS, potendo visualizzare in chiaro i dati che i due host agli estremi della comunicazione si scambiano, mediante forzatura illegittima nel caso dell'attacco Man in the Middle, concesso dal Client o comunque utilizzato in numerosi contesti aziendali il caso del Proxy Server.

1.2 Metodologie di identificazione, Deep Packet Inspection e disassembly dei dati

La Deep Packet Inspection[1] (abbr. DPI) è una pratica di analisi dei pacchetti implementabile mediante software/scrpit appositi su interfacce di rete che possono essere associate ad un qualsiasi dispositivo con una scheda di rete (PC domestici, Server, smartphone, routers etc...) che focalizza la propria attenzione sulla ricerca di dati specifici e ricorrenti (stringhe statiche che possono identificare l'azione di un virus, parole chiave all'interno del testo in chiaro di un pacchetto con dati HTML etc...) e che effettua appunto una scansione completa dei pacchetti transitanti per l'interfaccia di rete sulla quale un software orientato alla packet inspection rimane in ascolto intercettando questi ultimi, non fermandosi solo alle varie intestazioni IP incapsulate come nel modello di riferimento ISO/OSI fa un comune router secondo le specifiche del protocollo IP per l'instradamento dei pacchetti, bensì scandagliando interamente il contenuto del traffico in esame, intestazione e soprattutto parte relativa ai dati. Con l'uso dei vari filtri di cattura e di visualizzazione mediante regular expression presenti all'interno del tool Wireshark è possibile isolare ed in un secondo momento analizzare in maniera esclusiva dettagli implementativi e comportamento del protocollo TLS. Nello specifico tutta la parte di analisi da me effettuata concentra il proprio interesse nella ricerca di pattern ricorrenti o di metodi efficaci per il fingerprinting delle applicazioni in uso lato Client che instaurano un tunnel TLS con i Server associati a quel servizio, attraverso l'individuazione e la successiva implementazione di regole firewall per il testing sul blocco del tentativo di instauramento del canale criptato tra le due controparti interagenti, come possono essere ad esempio l'uso di un Browser Agent custom che si interfaccia ad un nodo d'ingresso della rete TOR o di un'applicazione Client VPN che si connette al desiderato Server VPN come ponte di connessione alla rete Internet.

1.2.1 Wireshark, packet sniffer

Wireshark[2] è un software open source multipiattaforma, con molteplici funzionalità di monitoraggio, analisi del traffico e dissezione dei protocolli.
Viene utilizzato perlopiù in ambiente didattico come strumento per l'analisi dei protocolli, delle intestazioni dei pacchetti e per un concreto riscontro visivo del flusso di rete in ingresso, in uscita ed in transito nei confronti l'interfaccia di rete monitorata.
In informatica forense può essere utilizzato sia come strumento di attacco che di difesa, ad esempio all'interno di una rete compromessa per intercettare tentativi di ARP Spoofing da parte di un host maligno che si camuffa da gateway della rete sfruttando la mancanza di autenticazione del protocollo ARP andando a controllare il MAC addresses della trama Ethernet associata al pacchetto di risposta di una ARP request e confrontarlo con quello del gateway fisico, oppure nello stesso scenario ma come sturmento di attacco Man in the Middle da parte dell'host maligno precedentemente citato il quale attraverso la funzionalità di collezionatore che Wireshark offre, può visualizzare, salvare ed analizzare in un secondo momento tutto il traffico sia in chiaro che criptato che transita per il proprio dispositivo/gateway camuffato prima di essere reinstradato verso la legittima via d'uscita della rete locale, fornendo potenziali informazioni utili per pianificare altri tipi di attacco (SSL Highjacking, HeartBleed bug, specifico per alcune versioni del protocollo SSL etc...).

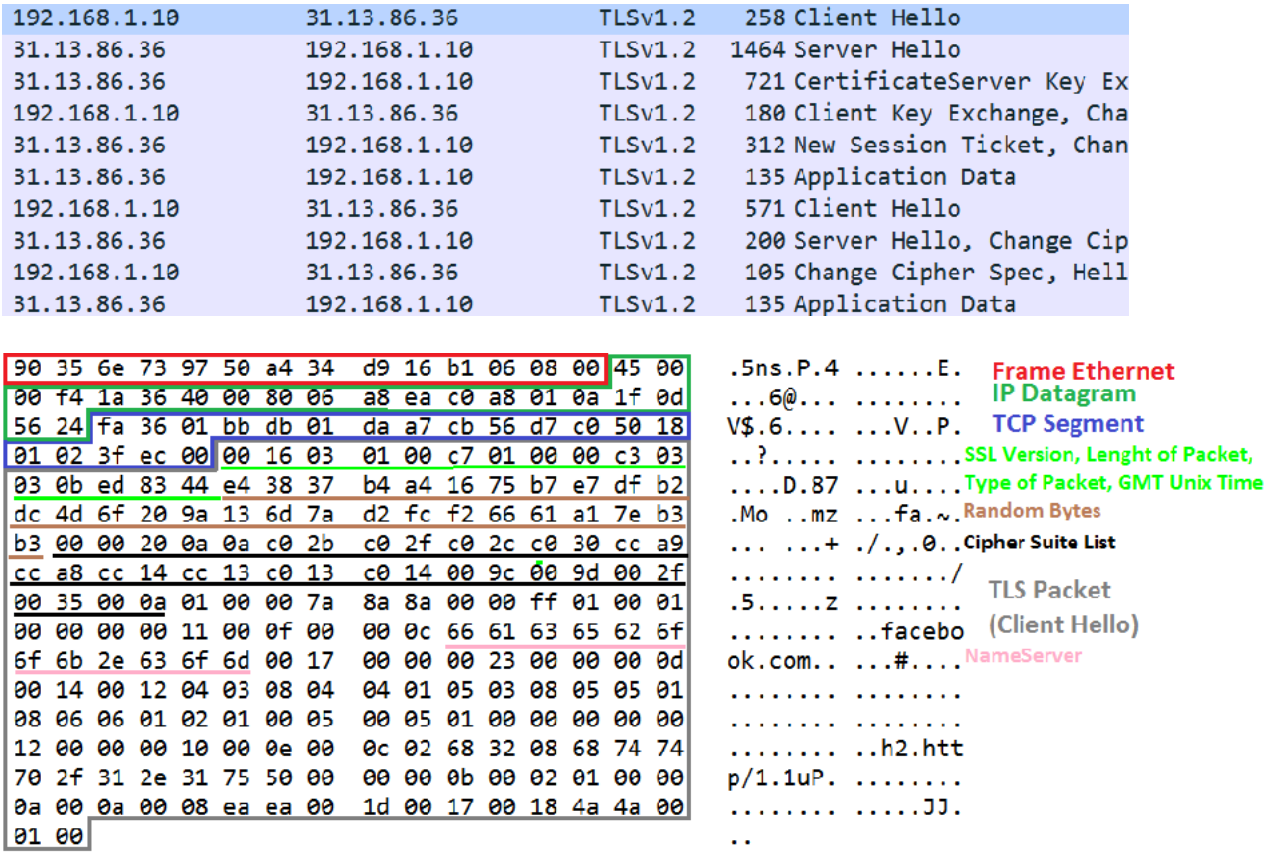


Figura 1.1: ***Immagine di cattura pacchetti mediante Wireshark, in alto il flusso di dati in ingresso ed in uscita nei confronti dell'IP 192.168.1.10 durante un handshake TLS, in basso la struttura del pacchetto Client Hello di presentazione del client in questione.***

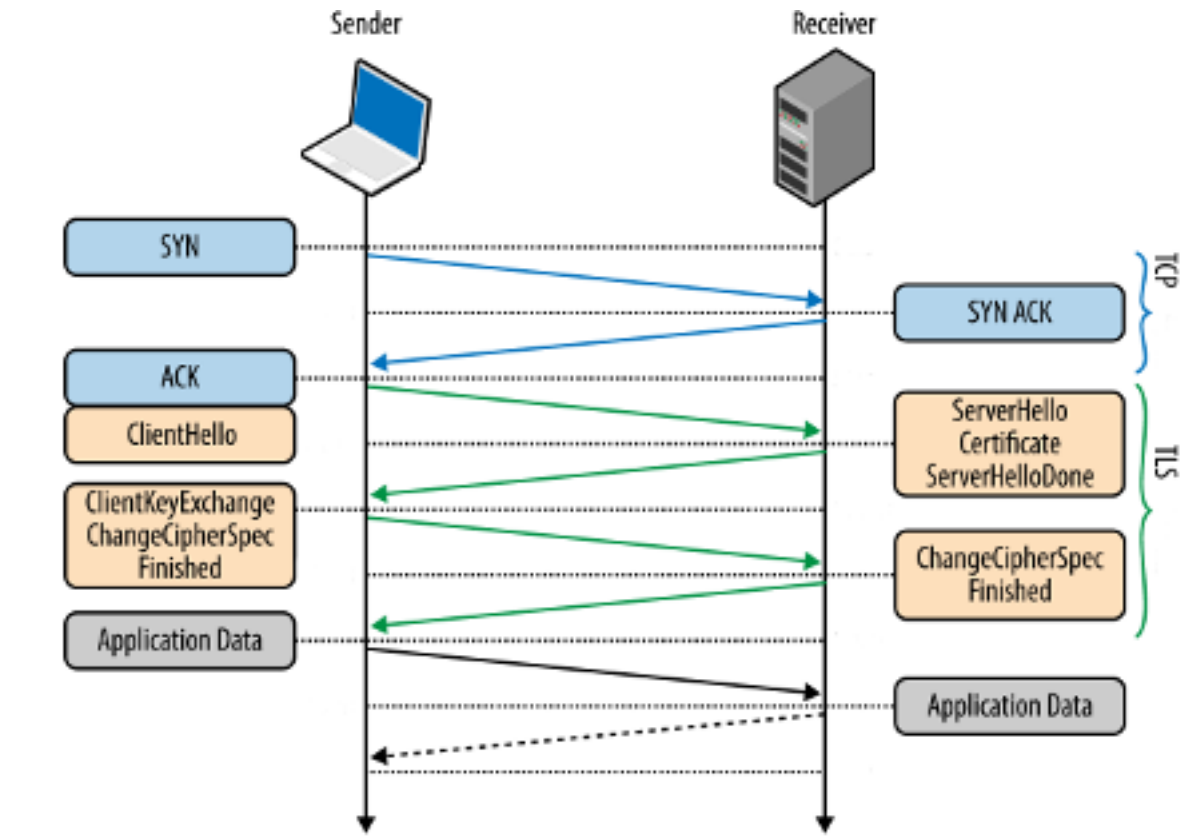


Figura 1.2: ***Regolare interazione tra Client e Server durante un handshake TLS.***

1.3 Il protocollo SSL/TLS, specifiche

Il protocollo[3] si colloca al quinto livello (Session layer) della pila protocollare ISO/OSI, è studiato appositamente per fornire un canale di comunicazione bidirezionale client/server criptato e offre funzionalità di autenticazione del server mediante certificato digitale firmato elettronicamente, si pone un gradino superiore al livello di trasporto TCP, sul quale viene incapsulato per l'instradamento e si pone come intermediario tra per l'appunto il livello di trasporto e quello di applicazione (es. HTTP). Fornisce uno strumento che permette di ovviare o comunque mitigare il furto di dati mediante packet sniffing criptando il traffico e di attacchi di tipo spoofing (camuffamento) garantendo un sistema di autenticazione gerarchizzato, composto dai vari Certificate Authorities, enti certificatori che rilasciano per l'appunto certificati digitali confermando l'identità degli enti che richiedono un certificato di autenticazione, il tutto in maniera gerarchicamente ricorsiva con uno schema ad albero/piramide (io Root certificate certifico te, ente certificatore di primo livello, chi verrà certificato da te potrà svolgere il ruolo di certificatore di secondo livello etc...).

1.3.1 Handshake, Cipher suit list, Tunneled stream

Come già detto il protocollo ha lo scopo di fornire un canale di comunicazione bidirezionale criptato, caratteristica che si rende necessaria per garantire la sicurezza dello scambio di dati in transazioni sensibili passanti per la rete Internet (acquisto online con carta di credito...)

la quale non è assolutamente presente nel layer sottostante della pila (TCP si preoccupa dell'affidabilità della connessione, garantendo il servizio di invio e ricezione confermati dei pacchetti, non della sicurezza) e che non è sempre presente a livello di applicazione a causa della grande etereogeneità dei servizi che i protocolli offrono a quel livello.
L'handshake tra client e server rappresenta l'unica parte dell'interazione TLS tra i due host a viaggiare in chiaro: il client instrada un pacchetto denominato "Hello Client" con il quale si presenta al Server inviandogli le informazioni necessarie per la creazione del tunnel di comunicazione. Di grande interesse come aspetto trattato nel seguito di questa ricerca, all'interno del pacchetto Hello Client è presente la lista dei Cipher[4] supportati dall'applicazione in uso da quest'ultimo (es. Browser Agent), ogni Cipher (Cifrario) rappresenta una combinazione di: tipologia di algoritmo di crittografia (autenticazione unilaterale, asimmetrica mediante scambio di chiavi pubbliche etc...), tipologia delle chiavi di cifratura e consequenzialmente dell'algoritmo utilizzato per criptare il data stream ed il MAC algorythm (Message Authentication Code) il quale determina la lunghezza e il tipo di algoritmo utilizzati per creare il digest hash utilizzato poi dal relativo host ricevitore del pacchetto per verificarne l'integrità e l'assenza di manipolazione da parte di un attaccante esterno. Il messaggio di risposta e di presentazione del server, denominato appunto "Hello Server" contiene due parametri fondamentali per la buona riuscita dell'handshake, il Cipher preferito supportato dal server in base alla lista di Ciphers fornita nell'Hello Client ed il certificato di autenticazione del server, con il quale quest'ultimo fornisce la propria identità digitale al client per l'autenticazione.
Eventuali ulteriori pacchetti possono essere scambiati tra client e server nel caso ci sia la necessità o la scelta arbitraria da parte di uno dei due di cambiare Cipher per l'encryption del canale di comunicazione, questo espediente viene per l'appunto utilizzato come metodo per rinegoziare le chiavi senza dover reinizializzare l'handshake.
Una volta designato ed accettato il Cipher da entrambe le parti chiamate in causa, lo scambio di dati vero e proprio può avvenire (es. richiesta di risorsa web mediante protocollo HTTPS...).

1.3.2 Stream Cipher e Block Cipher

Degna di nota è la differenza sul come agiscono le due tipologie di Cipher che possono essere utilizzate per crittografare il tunnel TLS, gli stream Cipher ed i Block Cipher, confronto necessario in quanto alcuni tentativi di exploit del protocollo sfruttano o hanno comportamenti differenti in funzione di un tipo di Cipher o dell'altro.[5]
Gli stream Cipher sono chiavi di cifratura simmetrica nelle quali la parte di dato in chiaro viene combinata con una chiave di cifratura pseudorandom (keystream), dove ogni byte del flusso viene crittografato con il relativo byte del keystream (es: keystream di 10 bytes, flusso di 100 bytes, i bytes 1,11,21... verranno crittografati con il primo byte del keystream, i bytes 2,12,22... con il secondo byte del keystream etc...), la crittografia avviene tramite semplici operazioni logiche booleane (generalmente XOR). Il valore del keystream viene ottenuto mediante l'uso di un numero variabile di operazioni di SHIFT L o SHIFT R sui registri della CPU. Il carico computazionale associato per l'esecuzione di algoritmi che si basano su stream cipher è piuttosto basso, in quanto il numero ed il tipo di operazioni che vengono

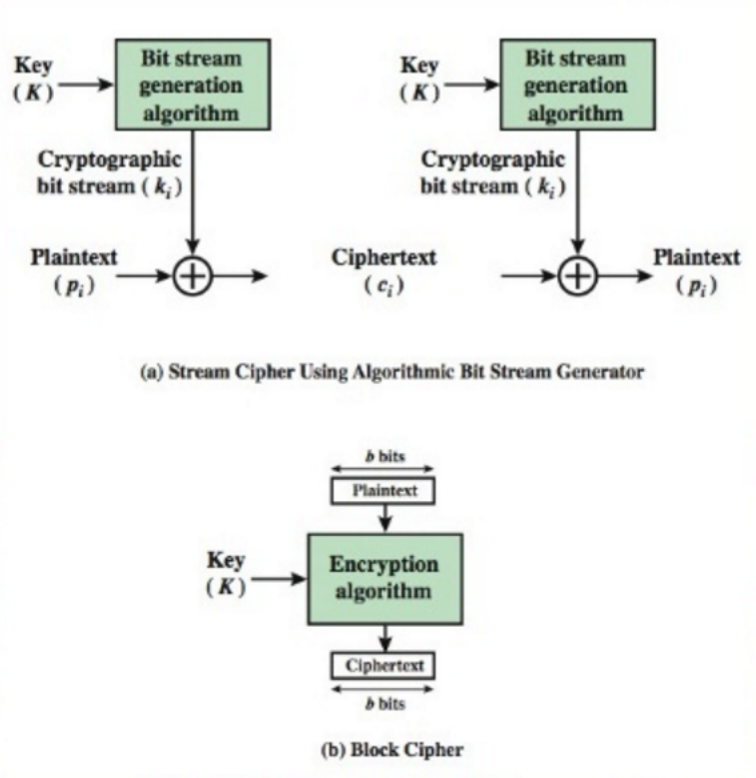


Figura 1.3: ***Flusso differenziato del processo di crittografia di Stream Cipher e di Block Cipher.***

effettuato è alquanto semplice, soprattutto se paragonato con quello dei Block Ciphers.
La crittografia usata dai Block Ciphers si differenzia da quella degli Stream in quanto nel secondo caso la chiave generata di k bytes opera indipendentemente da quella che è la lunghezza dello stream (essendo appunto uno stream, la dimensione del flusso di dati da crittografare non è deducibile a priori) e generalmente parti dello stream in cui un dato viene ripetuto il risultato dopo il termine dell'algoritmo su quelle porzioni di flusso generalmente varia in quanto la posizione del dato all'interno dello stream influisce su quale parte del keystream viene utilizzata per oscurare quella porzione di stream. Nei Block Cipher, come il nome suggerisce, l'offuscamento avviene per blocchi, quindi nel caso la stessa porzione di bytes da crittografare sia contenuta più volte all'interno di uno stesso blocco, ogni istanza di quella porzione criptata risultante risulterà comunque uguale all'interno di quel blocco.
I block Cipher sono chiavi di cifratura di k bytes che operano su blocchi di dati di lunghezza di n bytes dove k ed n sono dei numeri fissi dipendenti dalla tipologia di Block Cipher utilizzato. Gli algoritmi usati per i block cipher prevedono inizialmente la divisione logica del flusso per l'appunto in blocchi, i quali possono essere codificati sempre con la medesima chiave oppure con un vettore circolare (initialization vector) nel quale vengono inserite le chiavi che si susseguiranno nella codifica di ogni blocco. Utilizzando blocchi di dimensione fissa spesso si rende necessario aggiungere una componente di padding al dato da cifrare, in modo da raggiungere la dimensione fissata del blocco, aspetto che per alcuni block ciphers è risultato cruciale nello sfruttamento delle vulnerabilità di quest'ultimi.

1.3.3 PKI, Certificato digitale, X.509 e Certificate Authority

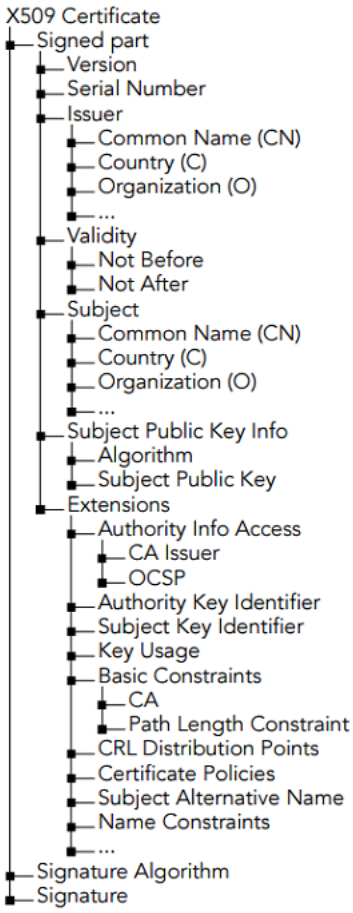
Un certificato digitale è un documento elettronico che lega un'identità ad una chiave pubblica facendo affidamento sulle firme digitali[6]. Il certificato in questione, specifico per la funzioanlità di autenticazione del protocollo TLS, segue lo standard X.509, il quale rappresenta uno degli modelli in uso per tipologie di infrastrutture a chiave pubblica (PKI)[7].
L'infrastruttura a chiave pubblica rappresenta un sistema ideato appositamente per certificare l'identità di un utente/ente attraverso l'uso di firme digitali, al fine di permettere la verifica dell'identità di quest'ultimo mediante l'uso di strumenti tecnologici e protocolli che basano parte del loro funzionamento su questa infrastruttura, l'associazione dell'identità di un soggetto ad una chiave pubblica che la identifichi necessita di enti terzi che svolgono il ruolo autoritativo di verificare fisicamente l'identità del soggetto in questione che richiede un certificato elettronico, terzi i quali prendono il nome di Certificate Autority (CA). Ruolo sul quale poggia le fondamenta di uno degli aspetti più delicati del protocollo TLS è dettato quindi dai CA, i quali possono essere sia enti pubblici che soggetti privati a cui viene delegato l'incarico di rilasciare dei certificati di autenticazione nei confronti del rispettivo ente/privato che richiede di essere certificato. Questo sistema può essere immaginato come una piramide alla cui cima ci sono i vari CA Root (gli unici ad autocertificarsi), al livello inferiore gli enti certificati ed autenticati dal relativo CA root e così via.
L'iter tramite il quale si viene certificati è il seguente: l'ente che richiede di poter essere certificato invia una Request for Authentication verso il CA designato, richiesta che dovrà contenere la chiave pubblica dell'ente e tutta una serie di documenti e prove identificative di quest'ultimo atte appunto a poter permettere l'identificazione ed il successivo rilascio del certificato da parte della CA alla quale era stata inoltrata la richiesta. Il fatto che i root CA possano autocertificarsi implica che questi soggetti rappresentino un'entità ben nota, rispettabile ed identificabile dal punto di vista sociale oppure che vi sia una tutela che ricade nella sfera giuridica nei confronti dell'identità del CA Root in questione (es. PA->Ministero degli Interni). Questo fattore rappresenta a mio avviso uno degli anelli deboli dell'architettura del protocollo TLS, in quanto il compito più importante, cioè quello dell'autenticazione delle identità viene spesso delegato ad enti che non presentano le adeguate infrastrutture informatiche per la tutela dei dati e la difesa del loro sistema da attacchi esterni, alcuni cenni storici di attacchi e brecce nei sistemi informativi dei CA verranno trattati in seguito.
Facendo un passo indietro e tornando alla forma del certificato in sè, scendendo nel dettaglio delle specifiche dello standard X.509, i principali campi che compongono il certificato digitale sono:

- **Serial number**, un numero di lunghezza in bit variabile che rappresenta l'identificativo univoco relativo a quel certificato nei confronti del CA che rilascia quel documento elettronico.

- **Subject**, il nome giuridico che contraddistingue (Distinguished Name) l'ente certificato, vi possono essere ulteriori attrbuti opzionali usati per definire altri nomi o alias in riferimento a quell'ente (CN=Common Name, C=Contry Name, O=Organization Name, OU=Organization Unit etc...)

- **Issuer**, il Distinguished Name dell'ente che rilascia il certificato.
- **Validity**, campo composto da due attributi, Not Before e Not After, valori che contraddistinguono l'intervallo temporale entro il quale il certificato risulta in validità.
- **Subject public key info**, contiene il valore dell'attributo relativo alla chiave pubblica dell'ente certificato ed il campo Algorythm, contenente le informazioni relative all'algoritmo utilizzato per generare la chiave pubblica
- **Authority info access**, link ipertestuale alle risorse tramite le quali possono essere reperitietutte le informazioni relative a quel certificato all'interno della struttura informatica del CA associato.
- **Authority key identifier**, contenente il valore dell'attributo relativo alla chiave pubblica dell'ente certificatore.

In uno scenario all'interno del quale l'ente che si fa rilasciare un certificato si affida ad un CA non root bensì di livello più basso nella piramide degli Authorities, questi campi verranno inseriti all'interno del pacchetto Hello Server nella zona di intestazione relativa al Certificato in maniera ricorsiva per ogni livello di certificazione risalendo la piramide fino ad un CA Root, il quale, se legittimo, verrà riconosciuto dal Browser in questione, scatenando una reazione a catena che porterà alla validazione di tutti i certificati sottostanti presentati, struttura che prende appunto il nome di Certificate Chain.[8]



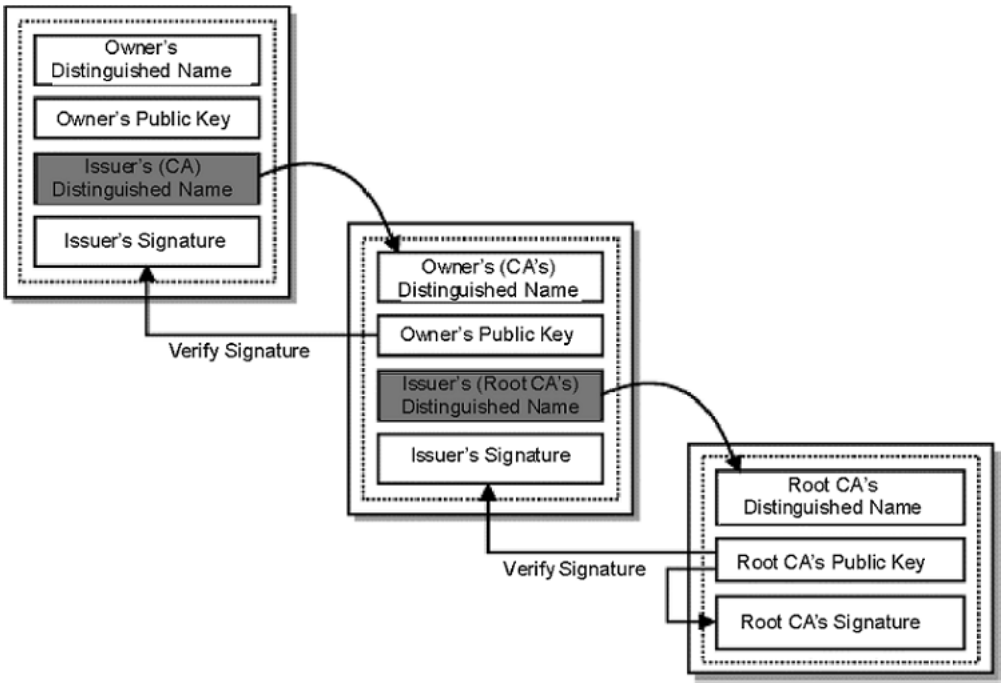

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3976 (0xf88)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=BM, O=QuoVadis Limited, OU=www.quovadisglobal.com,
             CN=QuoVadis Global SSL ICA
        Validity
            Not Before: Jan 15 09:28:27 2009 GMT
            Not After : Jan 15 09:28:27 2011 GMT
        Subject: C=CH, ST=Zuerich, L=Zuerich, O=SWITCH, CN=pki.switch.ch
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:c8:a4:c0:e3:78:8f:dd:76:4a:b7:05:ba:0b:3c:
                    e4:75:a2:d8:db:b3:43:0d:38:bf:74:7b:19:5b:8b:
                                    ...
                    4a:ce:49:58:72:46:6b:41:ad:51:fd:bc:15:43:04:
                    87:a1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Authority Information Access:
                OCSP - URI:http://ocsp.quovadisglobal.com
                CA Issuers - URI:http://trust.quovadisglobal.com/qvsslica.crt

            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.8024.0.2.100.1.2
                  CPS: http://www.quovadisglobal.com/repository

            X509v3 Subject Alternative Name:
                DNS:pki.switch.ch
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Authority Key Identifier:
                keyid:32:4D:A1:4F:EA:F0:AE:99:B6:EE:9B:07:2C:84:08:11:50:8B:E2:7E

            X509v3 CRL Distribution Points:
                URI:http://crl.quovadisglobal.com/qvsslica.crl

            X509v3 Subject Key Identifier:
                96:72:4E:BA:DE:98:F9:78:D2:91:E6:F0:6D:38:1E:0B:FC:67:CC:FB
    Signature Algorithm: sha1WithRSAEncryption
        0c:6e:a5:7d:35:ae:df:cb:4d:fe:b2:41:37:7b:be:e5:bc:b6:
        89:cd:8c:12:04:7a:bb:a8:89:9c:20:63:be:45:3a:2a:26:28:
                                ...
        2e:43:a0:8f:48:39:40:eb:12:92:6d:b4:6f:fd:f0:ee:fd:18:
        27:d1:29:0e
```



1.4 Exploit e problematiche del protocollo SSL/TLS

1.4.1 Timing attacks, RC4 attacks, Cipher attacks, Downgrade attacks, Protocol Bugs, il caso di HeartBleed e Violazioni ai CA

Essendo un prodotto dell'uomo il protocollo TLS è ben lungi dall'essere perfetto e le versioni del protocollo che si sono aggiornate in questi ultimi 20 anni hanno portato a risolvere o comunque mitigare notevolmente quelli che erano i bug e le vulnerabilità intrinseche del protocollo, tanto che le tipologie di attacco si sono dovute evlovere fino ad arrivare ad exploit bassati sul Timing Attack. Gli attacchi di tipo Timing sono tentativi di exploiting a livello crittografico che basano il loro funzionamento sull'analisi di tempo e uso della CPU durante l'elaborazione di algoritmi crittografici, generalmente necessitano di essere a conoscenza dell'hardware dell'host che si sta attaccando. Gli algoritmi di crittografia più moderni, tenendo conto di questo fattore, si sono evoluti in maniera tale che le istruzioni non venissero eseguite nel tempo ottimale dell'algoritmo, bensì randomizzando il tempo di esecuzione di ogni passo dell'algoritmo in un intervallo compreso tra il migliore ed il peggior caso temporale mediante l'uso di semafori temporizzati dinamicamente ad ogni esecuzione dell'algoritmo e minimizzando l'uso di variabili tempo-dipendenti.

1.4.1.1 CRIME

CRIME[10] (Compression Ratio Information Made Easy) è un exploit che sfrutta la compressione dei pacchetti presente in SPYDY e nell'ultima versione di HTTPS per ottenere in chiaro il valore del webcookie che il client sta inviando al server per l'autenticazione rapida. L'attaccante, in ascolto sull'interfaccia di uscita dell'host infetto o del suo gateway di rete in attesa di un Hello Client TLS verso qualche server esterno, forgiando al momento dell'evento richieste simili ma con padding finale in cui inserisce dati grezzi prima di lanciare a sua volta

una serie di richieste. Utilizzando un padding diverso ad ogni Hello Client, analizzando la dimensione della risposta (più risulta piccola e vicina per dimensioni alla risposta lanciata dal server verso l'host che stiamo attaccando più il dato presente all'interno del nostro padding risulterà vicino al dato di nostro interesse), attraverso una strategia di tipo divide et impera e con un numero di tentativi che al massimo risulta pari a quello dei secret bytes che l'attaccante sta tentando di decodificare, sarà infine possibile risalire al valore in plain text del webcookie cercato.

1.4.1.2 BREACH

BREACH (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext) risulta essere una variante di CRIME[11] ottimizzata per effettuare exploiting esclusivamente su protocollo HTTP focalizzando la propria implementazione sugli algoritmi di compressione gzip e DEFLATE, i quali risultano i due unici supportati di default dalle versioni più recenti dei browser più utilizzati. Mentre il livello di lavoro dell'exploit CRIME è relativo al layer di sessione, quindi sfruttando la compressione che viene effettuata lavorando su SPYDY/TLS, BREACH opera soltanto sfruttando richieste e risposte HTTP, è quindi in grado di raggiungere il risultato indipendentemente dal tipo di Cipher utilizzato, stream o block che esso sia, anche se nel secondo caso è richiesta potenza computazionale addizionale per via della maggiore complessità di calcolo intrinseca dei Block.

1.4.1.3 Cipher RC4

RC4 è uno Stream Cipher presente all'interno della lista dei Ciphers supportati ed utilizzati dal protocollo SSL/TLS. Inizialmente implementato nel 1987 da Ron Rivest (RC sta infatti per Ron's Code...) il cui codice e funzionamento è rimasto sconosciuto fino al 1994, anno in cui ne è stato 'leakato' senza autorizzazione il contenuto in rete, RC4 ha trovato la propria sfortuna a causa di quest'evento svelandone le debolezze[12] implementative (RC4 rappresenta comunque lo standard degli Stream Cipher, la cui tipologia basa il proprio funzionamento sulle basi di questo algoritmo), rendendo questo Cipher vulnerabile ad attacchi di tipo bit-flipping e ad attacchi di tipo reverse brute force sfruttando la natura logica dello XOR (un doppio XOR con la medesima chiave di un messaggio in chiaro risulterebbe poi nella ricodifica dello stesso messaggio al termine della seconda operazione logica).

1.4.1.4 HeartBleed e l'NSA

Heartbleed è un exploit per la compromissione dei dati presenti all'interno della memoria volatile di un Server al momento dell'attacco e sfrutta un bug presente all'interno della libreria Heartbeat del progetto open source OpenSSL per ottenere queste informazioni dal server in maniera illegittima. Tramite questo bug un attaccante che bersaglia il server designato, durante l'interazione TLS mediante appunto l'uso di Heartbeat fa polling verso il server per verificare che la connessione sia ancora aperta e che quest'ultimo risponda, secondo il formato riassumibile in maniera naive così: "Se sei ancora lì rispondi 'Ciao' (4 lettere)". In un attacco

di tipo Heartbleed questo formato viene modificato nel seguente modo dal Client: "Se sei ancora lì rispondi 'Ciao' (40004 lettere)"; e la mancanza di verifica da parte della libreria associata sulla lunghezza della richiesta comporta l'invio da parte del server di un pacchetto contenente l'informazione "'Ciao_+dati_presenti_nella_porzione_di_RAM_contigua_a_quella _allocata_per_Ciao'"[13].
La cosa sconcertante in merito alla scoperta di questa vulnerabilità è data dal fatto che la pubblicazione ufficiale della scoperta risale al 7 Aprile 2014, ma indiscrezioni poi ufficializzate hanno confermato che enti governativi come l'NSA sfruttavano questa falla da oltre due anni con il pretesto di tracciare e reperire informazioni per la difesa contro attentati terroristici[14].

1.4.1.5 CA compromessi, i casi di DigiNotar, VeriSign e Trustwave

Come spiegato nella sezione 1.3, il ruolo che rivestono i CA è forse la chiave di volta che tiene in piedi il sistema di verifica dietro all'autenticazione nel protocollo TLS, una breccia da parte di potenziali attaccanti all'interno di un ente certificatore può rappresentare uno scenario catastrofico per l'intero sistema, in quanto la compromissione della chiave privata di un CA a sua volta è in grado di rendere vulnerabili anche tutto il sottoramo della piramide di autenticazione relativo a quell'ente certificatore.
Il 20 settembre 2011 il CA olandese DigiNotar, parte della compagnia VASCO Data Security International Inc.[15] andò in bancarotta a causa di una violazione del loro sistema scoperta 2 settimane prima. Venuto a conoscenza del fatto il governo olandese prese le redini della questione aprendo un'indagine sul fatto, andando a scoprire che oltre 500 dei certificati rilasciati da DigiNotar risultavano compromessi. I risultati non sono stati resi pubblici dal governo per oltre un anno, in quanto la compromissione del sistema risultò al tempo totale, fatto che avrebbe potuto avere ripercussioni dal punto di vista economico di tutte le società affiliate, legate o in qualche modo correlate a DigiNotar.
Dal report[16] sull'accaduto pubblicato nell'ottobre 2012 si evince l'attacco risulta correlato ad oltre 300.000 indirizzi mail Gmail iraniani, anche se non vi può essere al 100% la sicurezza che ci sia il governo iraniano o un qualche hacker iraniano dietro l'accaduto, in quanto mai nessuno ha rivendicato l'attacco. Le negligenze da parte della compagnia sono state molteplici, da un sistema di difesa inadeguato alla mancanza di una tempestiva notifica nei confronti dell'ENISA (Agenzia europea per la sicurezza delle reti e dell'informazione), perfino la mancanza di strategie di individuazione e politiche di blocco e ripristino del servizio in casi di emergenza, tutti fattori che poi a prescindere dalla pubblicazione o meno del report hanno portato alla bancarotta della compagnia. I responsabili dei Browser principali, appena appresa la notizia, hanno reagito tempestivamente etichettando come untrusted ogni certificato contrassegnato dalla firma di DigiNotar, tra gli oltre 500 certificati compromessi risultavano presenti nomi piuttosto importanti quali MI6, domini Google iraniani, Facebook, Microsoft ed altri minori.

VeriSign è una compagnia statunitense che detiene un ruolo piuttosto elitario all'interno dell'INTERA infrastruttura di internet, possiede 2 dei 13 DNS Root, responsabile dei domini di primo livello .com e .net e fino al 2010 risultava essere un CA di primo livello, prima

di vendere tutta l'infrastruttura relativa al servizio di Authority all'azienda Symantec, al tempo gestiva oltre 3 milioni di certificati. Sono giunte a noi informazioni nel 2012 proprio in merito a fatti avvenuti 2 anni prima, rivelando che per un breve periodo (1 mese circa) la loro chiave privata fosse stata compromessa, spianando la strada agli attaccanti. La notizia, confermata dalla socità, ha posto un'importante questione in merito all'uso dell'infrastruttura PKI per l'autenticazione, risonando a causa dell'importanza strategica che la società riveste a livello di risoluzione DNS (50 miliardi di richieste DNS giornaliere), anche se è sempre stato negato ogni tipo di falla relativo ai server DNS l'episodio dovrebbe far riflettere.
L'ultimo caso, forse quello più interessante dal punto di vista di questa ricerca, riguarda il CA Root Trustwave, il quale è stato coinvolto in quello che si potrebbe definire uno scandalo informatico, in quanto è stato dapprima accusato e poi ha ammesso l'uso di certificati digitali firmati senza un Distinguished Name permettendo ad ignote società privata di poter operare da attaccante MITM nelle connessioni TLS che venivano effettuate all'interno del loro network. Trustwave al tempo si difese semplicemento asserendo che era una pratica comune richiesta da alcune società, usata all'interno di infrastrutture aziendali per monitorare più efficacemente il traffico di rete all'interno del network dell'azienda, giustificandosi sostenendo che fosse impossibile estrapolare la chiave privata dai device hardware da loro creati ad-hoc per questo scopo.[17] Un lungo dibattito all'interno della Mozilla Foundation ha riportato a galla la questione in merito alla gerarchia degli Authorities, di come il ruolo dei CA sia strategicamente fondamentale e dell'abuso effettuato dal Root in questione, avviando una request iniziale per rimuovere Trustwave dai CA Root trusted dal Browser Firefox, ma che alla fine si è concluso con un nulla di fatto.

1.5 Casi di studio

La scelta di specifici casi di studio da esaminare ha la finalità di fornire materiale di riscontro concreto per tutta quella che è stata la parte di ricerca e di documentazione collezionata, l'obiettivo è quindi quello di fornire nozioni e dati utili per la fase di testing sui quali poi poter trarre le dovute conclusioni riguardo il livello di sicurezza che il protocollo TLS garantisce e quelli che potrebbero essere dei cambiamenti o miglioramenti attuabili per offrire da un lato un servizio più sicuro e robusto, dall'altro per creare efficaci politiche di filtraggio e blocco dei contenuti o di contrasto nei confronti di questo fenomeno (della serie la miglior difesa è l'attacco).
In questa sezione introduttiva vengono confrontati dapprima quelli che sono i casi di studio da me approfonditi e sui quali poi sono stati reperiti i dati utili a finalizzare quella che è stata la mia attività di ricerca.
La fase di ricerca è stata effettuata simulando una situazione nella quale il traffico di una rete compromessa veniva intercettato ed analizzato da un host attaccante per effettuare una ricerca selettiva delle connessioni TLS instaurate e dei dati reperibili da questo tipo di analisi. Nella fase di testing queste nozioni sono state invece messe in pratica andando ad implementare un Firewall atto ad individuare e bloccare specifici tipi di connessioni TLS ed a frappormi come attaccante MITM mediante la mimica di un Proxy Server.

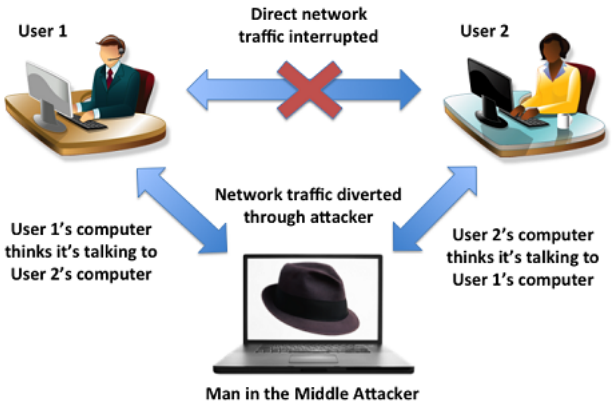


Figura 1.4: ***Schema grafico di riferimento di un attacco MITM.***

Per quanto concerne la parte di analisi, per l'intercettazione del traffico che transita per un gateway di rete ho utilizzato un Router Mikrotik simulando un contesto nel quale l'attaccante ha modo di poter immagazzinare ed inviare tutto il traffico transitante per il router compromesso verso un host esterno attraverso un tool di sniffing presente all'interno del sistema operativo del Router, traffico il quale poi può essere analizzato tramite software di analisi dei pacchetti dall'host esterno ricevente (Wireshark, Tshark, TCPDump etc...), Wireshark nel nostro caso, effettuando quindi una simulazione di attacco di tipo man in the middle passiva. La parte di testing invece è stata effettuata andando ad implementare un Firewall dapprima sullo stesso router utilizzato per la fase di analisi, abbandonato poi in un secondo momento a causa di limiti sia Hardware che Software passando all'uso di una macchina fisica impostata per il packet forwarding e posta come Firewall di frontiera che si frappone tra la rete locale ed Internet, macchina con kernel linux che utilizza il Firewall Iptables come più potente strumento di filtering per testare l'efficia del blocco delle specifiche connessioni TLS testate.

1.5.1 Man In The Middle e Proxy Server

L'attacco di tipo Man In The Middle (da qui abbreviato MITM) è una tipologia di violazione informatica nella quale un host maligno si frappone nella comunicazione legittima tra due client o un client ed un server, con lo scopo di intercettare il traffico da e verso l'utente bersaglio.

Gli attacchi MITM, come già introdotto nella sezione precedente, possono essere di tipo attivo, cioè violazioni nelle quali l'host maligno mima il comportamento del server o dell'altro client con il quale il bersaglio designato sta comunicando, potendo potenzialmente visualizzare in chiaro la parte di data criptata, oppure di tipo passivo, cioè violazioni nelle quali l'host maligno rimane in ascolto all'interno della rete intercettando il traffico relativo all'host bersaglio, utilizzando applicazioni come packet collectors/sniffers per poi analizzare in un secondo momento il traffico intercettato (Sniffing passivo).
Nel caso delle connessioni SSL/TLS questo tipo di attacco è contrastato dall'architettura di

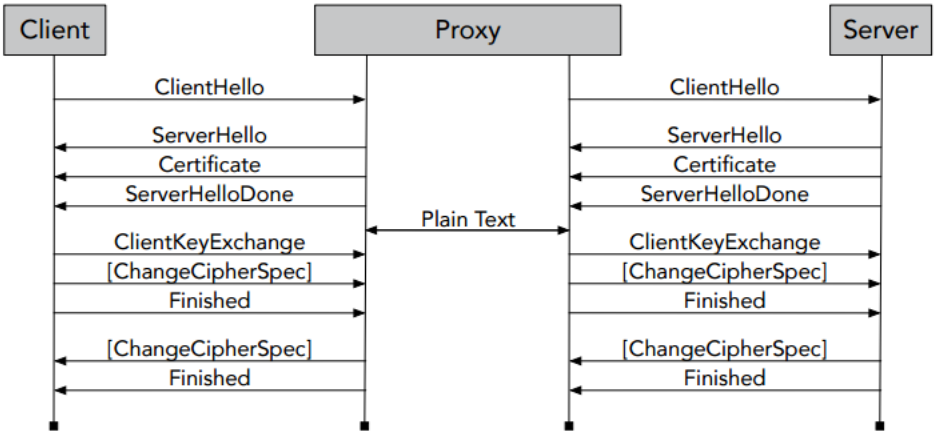


Figura 1.5: ***Proxy Server che si pone come intermediario negli handshake TLS.***

tipo PKI (Public Key Infrastructure) precedentemente descritto e che permette l'identificazione certa dell'ente/soggetto con il quale si sta instaurando un tunnel TLS mitigando gli attacchi di tipo attivo e crittografando la parte di data una volta terminato l'handshake per rendere la vita ostica a sniffer passivi (necessità di strumenti con capacità di calcolo computazionale per decifrare il traffico disumani). La possibilità di generare certificati "on the fly" in maniera dinamica da parte di un attaccante MITM permette a quest'ultimo di potersi frapporre nella comunicazione tra i due end-point della comunicazione ed instaurando un tunnel TLS con entrambi.

Il pacchetto Hello Client viene intercettato dall'host attaccante, viene letta l'intestazione e modificato l'indirizzo IP sorgente con l'indirizzo IP dell'host attaccante, viene reinstradato il pacchetto verso la destinazione legittima, l'host attaccante riceve poi il pacchetto Hello Server, instaurando così nel frattempo un tunnel TLS con il server legittimo con il quale il client bersaglio sta cercando di comunicare, nel frattempo viene forgiato un pacchetto del tutto simile alla risposta ricevuta dal server nel quale però viene inserito un certificato generato in maniera dinamica nel quale la chiave pubblica e la chiave privata vengono modificate con quelle generate dall'attaccante, il pacchetto viene poi inviato al client bersaglio permettendo così all'attaccante di poter instaurare una seconda connessione TLS con il client, questo permette all'host attaccante di poter decrittare il traffico TLS uscente dal client, visualizzarlo in chiaro, manipolarlo a proprio piacimento e reinstradarlo verso il server legittimo, i pacchetti di risposta dal server seguiranno lo stesso percorso permettendo così all'attaccante di poter visualizzare in chiaro tutto il traffico TLS relativo ai due end-point di comunicazione. Questa metodologia di attacco comporta però che il client, una volta ricevuto il pacchetto Hello Server, sia notificato del fatto che il certificato inviato dall'attaccante risulti untrusted, in quanto il suddetto certificato sarà di tipo self-signed o comunque non corrispondente al certificato che si aspetta il browser agent del client bersaglio, il quale sarà costretto ad accettare quel certificato per poter terminare l'handshake ed iniziare la comunicazione. Nel caso di generazione di certificati "on the fly" da parte dell'attaccante, la mancata accettazione del certificato associato a quest'ultimo comporta l'impossibilità da parte del client di poter terminare qualsiasi tipo di handshake che stia tentando, in quanto tutte

le risposte legittime dai server interrogati verranno bloccate e riforgiate dall'host attaccante con il proprio certificato self-signed.
Questo scenario descrive in maniera puntuale ciò che avviene quando ad esempio viene utilizzato un proxy server come intermediario per la navigazione su internet, il client o il sistemista responsabile dell'infrastruttura di rete che fruisce di questo servizio dovrebbe essere a conoscenza di questa implicazione correlata ed appoggiarsi ad un Proxy affidabile in quanto quest'ultimo risulta potenzialmente in grado di visualizzare in chiaro tutto il traffico TLS transitante per esso.

CAPITOLO 2

Dati reperibili da un handshake TLS utili al filtraggio dei servizi ed implicazioni correlate

In questa sezione viene illustrato nel dettaglio come, grazie ai dati reperibli da un handshake TLS, si possa in maniera efficace distinguere il browser agent o l'applicazione in uso associata al client e di come attraverso lo studio del tipo di interazione con il server si possa bloccare l'handshake impossibilitando l'instauramento della connessione e bloccando di fatto il servizio.

2.1 Fingerprinting Browser Agent/Applicazione

Come già spiegato nel capitolo precedente, nel pacchetto Hello Client di un handshake TLS, il browser agent o l'applicazione client che cerca di instaurare un tunnel con il relativo server, invia la lista di cipher che esso supporta, una lista statica ed invariata (a meno di aggiornamenti del software) che viene presa da un qualche file di configurazione dell'applicazione e viene inserita all'interno della parte di data del pacchetto. Ad oggi vi sono oltre 200 cipher[x] supportati per il protocollo TLS, alcuni dei quali però risultano vulnerabili, facendo scendere la lista di quelli ancora fortemente utilizzati a circa 100-130, permettendo un numero notevole di combinazioni possibili con le quali poter andare a popolare la Cipher Suite List da inserire nel pacchetto Hello Client. In virtù di questo elevato numero di combinazioni possibili, effettuando una ricerca sui Ciphers supportati dai browser agent ho potuto constatare che ad ogni versione di un Browser Agent risulta associata una lista che presenta una combinazione spesso (ma non sempre[18]) univoca di Ciphers, estendendo questo concetto a qualsiasi applicazione lato Client che sia costruita per creare un tunnel TLS (come ad esempio una VPN) con un qualche server mediante un generico metodo di discovery (lista di IP statici all'interno di qualche file dell'applicaizone Client, query DNS sul nameserver etc...) si può evincere che questo dettaglio è potenzialmente in grado di permettere il troncamento di un handshake TLS sul nascere già dal primo pacchetto dell'handshake. Nella parte relativa alle metodologie da me implementate per bloccare uno specifico servizio, tentativi di handshake

TLS da parte Browser Agent come il Client TOR e applicazioni VPN come il Client Windows per HotspotShieldVPN sono da me efficacemente stati bloccati sfruttando questo metodo per discriminare un'applicazione dalle altre.
Il fingerprinting sulle Cipher Suite List non è però il solo metodo che può essere utilizzato per discriminare il tentativo di handshake da parte di una applicazione, in quanto all'interno dell'handshake stesso vi sono altri due elementi che permettono di poter bloccare un determianto servizio sfruttando dettagli non più lato Client, bensì Server, ma in maniera comunque efficace rispetto al fingerprinting sui Ciphers bisogna spostarsi sul pacchetto di risposta nell'handshake Hello Server. Tra i vari dati che il server invia come risposta al client vi sono due aspetti che permettono in questo caso di poter applicare regole di blocco e filtraggio su quello specifico servizio, mi riferisco al campo Serial Number all'interno del certificato che il Server invia per autenticarsi nei confronti del Client.

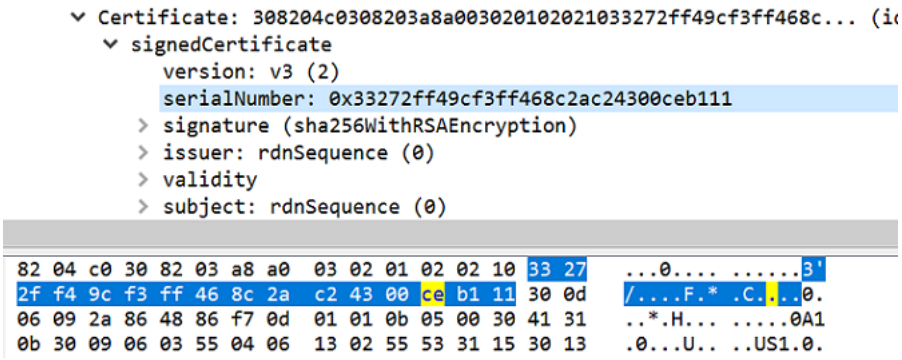


Figura 2.1: ***Estratto Wireshark di un pacchetto Certificate di risposta da un Server Unibo contenente, tra i vari certificati ricorsivamente validati, quello Unibo richiesto per ottenere le risorse associate alle pagine web dell'ateneo.***

Elemento già citato in precedenza nella descrizione della struttura che un certificato ha secondo lo standard definito per il protocollo, il Serial Number è quella stringa alfanumerica rappresentante un id univoco associato a quel certificato nei confronti dell'Authority che ha rilasciato quel certificato. La lunghezza del serial viene scelta in maniera arbitraria dall'authority, anche se lo standard impone una lunghezza massima di 20 bytes, permettendo un'eventuale probabilità di collisione in caso di fingerprinting col serial number rilasciato da un'altra authority a mio avviso trascurabile Estensioni VPN per Browser da me approfondite nella parte di testing sono state bloccate efficacemente mediante regole di blocco che sfruttavano il Serial Number presente all'interno del certificato.

2.2 Network Neutrality e fingerprinting sul Certificato

Per quello che è lo stato odierno della rete Internet ogni pacchetto in transito per un nodo della rete viene trattato in maniera indifferente rispetto agli altri a prescindere dal contenuto, dalla sorgente o dalla destinatazione, funzionamento caratteristico alla base del protocollo

IP e cardine principale del concetto di network neutrality.[19] Aspetto non di poco conto quindi è che gli ISP, in un contesto in cui dovesse venir meno il caposaldo della network neutrality, potrebbero tranquillamente utilizzare il campo Serial Number o il Distinguished Name del certificato come strumento per effettuare politiche di instradamento con priorità nel processamento del traffico di rete in transito sui loro nodi dando la precedenza ad uno stream instaurato in seguito ad un handshake TLS piuttosto che un altro, potendo, senza la necessità di installare fisicamente reti ad-hoc, alterare la qualità del servizio erogato in funzione del contesto relativo alla terna di elementi dati da un handhsake TLS, i due indirizzi IP agli estremi della comunicazione ed il certificato digitale come strumento di mangling del traffico TLS associato all'interazione tra questi due nodi.

2.2.1 Dumb Network, discriminazione per protocollo e Traffic Shaping

La rete Internet per com'è strutturata oggi viene definita stupida (Dumb network) in quanto, per via di quella che è l'implementazione del protocollo IP[20], ogni pacchetto in transito sulla rete viene trattato allo stesso modo, senza alcun meccanismo di marcamento del traffico o di policy a priorità differenti in funzione di sorgente/destinazione/tipologia di protocollo/dato contenuto all'interno del pacchetto.
A livello di gergo informatico questo concetto viene spesso accostato a quello di Dumb Pipe, il quale viene utilizzato per spiegare in termini più concreti questo comportamento mediante l'esempio metaforico della fontanella pubblica, dove appunto l'ente comunale che mette a disposizione questo servizio si fa garante della fruibilità di quest'ultimo in cambio di una tassa per i residenti, indipendentemente da chi poi andrà a beneficiarne ed il metodo con il quale lo utilizzerà (bere, lavarsi...), allo stesso modo all'interno di una Dumb Network gli ISP rendono (o dovrebbero rendere...) disponibile la propria banda di rete in maniera indiscriminata a chi paga per il servizio (con limiti di banda solo in funzione della tipologia di contratti stipulati con l'utenza) indipendentemente da come e quanto poi questa verrà effettivamente utilizzata dal bacino di utenza, senza distinzione e controllo sul traffico associato, riversando ogni responsabilità sui metodi di utilizzo nei confronti dell'end user. I pro ed i contro di questo genere di infrastruttura sono facilmente deducibili:

- PRO: gli ISP non hanno controllo sul traffico dati transitante per la loro rete, non vi è possibilità legale di limitare, bloccare o alterare servizi disponibili nella rete Internet.
- CONS: la QoS (Quality of Service) dipende, oltre che dagli intrinseci fattori fisici di velocità della luce e distanza tra gli host comunicanti, ESCLUSIVAMENTE dallo stato di quella porzione di sottorete che separa gli host comunicanti, questo perché appunto ogni pacchetto che transita su ogni router viene processato allo stesso modo dal router in questione seguendo le specifiche del protocollo IP.

Da un lato, la discriminazione dei pacchetti in funzione del dato/protocollo potrebbe rappresentare una soluzione efficiente per migliorare la QoS di tutta la rete internet, poniamo il caso ad esempio di un host che si connette ad un server che ospita dirette streaming, il flusso TCP associato per garantire la fruibilità del servizio deve essere continuo e costante,

di controparte possiamo sempre utilizzare lo stesso host come esempio che nel frattempo sta messaggiando tramite Whatsapp con i propri contatti, in questo caso un ritardo anche nell'ordine dei secondi risulta tollerabile per il nostro utente. Dall'altro lato però una gestione smart e non più dumb del traffico di rete pone gli ISP più direttamente controllori e responsabili del traffico presente sulla loro rete. Per differenziare il traffico ed applicare politiche intelligienti di instradamento sono necessari scripts di packet inspection da inserire all'interno dei router, ed a prescindere dal contenuto della parte di data, quindi senza andare ad impelagarsi su questioni correlate alla sfera giuridica della privacy, scansionando la parte di header del pacchetto nella stragrande maggioranza dei casi è possibile identificare il protocollo associato a quell'intestazione, potenzialmente potendo andare a manipolare o addirittura bloccare alcuni tipi di contenuto associati in maniera esclusiva ad un determinato protocollo (es. blocco di Facebook da parte di un ISP mediante fingerprinting sul Serial Number associato al certificato di Facebook).
Un caso degno di nota sotto questo punto di vista è quello associato alla Comcast Corporaton, ente televisivo via cavo ed ISP statunitense, il quale in barba ad ogni principio di network neutrality in passato ha stretto accordi con Netflix[21] per canali di comunicazione dedicati grazie al mangling dei pacchetti in ingresso ed uscita dai Comcast TV Box i quali venivano trattatati all'interno della rete Internet di Comcast con delle priorità di forwarding maggiori rispetto ad altri pacchetti. Il lato oscuro di questa vicenda è che questo genere di intervento ha comportato l'installazione di scripts per l'ispezione dei pacchetti al di fuori delle strette necessità del protcollo IP, cosa che ha permesso a Comcast di applicare politiche di filtraggio per quello che secondo loro risultava essere traffico pirata indesiderato, bloccando il traffico associato al protocollo BitTorrent.
Precisamente a questo caso il blocco non è realmente un blocco, bensì il traffico associato al protocollo mediante software predisposti al Traffic Shaping veniva fortemente limitato e rallentato, iniziando ad essere "droppato" soltanto quando i buffer degli hardware associati venivano saturati.
L'esempio pocanzi riportato è utile per definire bene quella che è a mio avviso l'importanza della Network Neutrality, è vero che viceversa la QoS incrementerebbe indubbiamente grazie alla maggiore intelligienza computazionale nell'instradare i pacchetti, però nelle mani sbagliate questo baluardo di efficienza avanguardistico potrebbe essere utilizzato per il mero interesse degli ISP, rendendo l'Internet un luogo potenzialmente meno libero (potere decisionale agli ISP) e ancor più soggetto alle leggi del mercato di quanto esso già purtroppo sia oggi.
Il traffic shaping[22] è una tecnica di ottimizzazione per il processamento dei pacchetti in ingresso/transito/uscita da una determinata interfaccia di rete applicabile sia all'interno di LAN/reti aziendali che da parte degli ISP nella loro porzione di rete Internet, tecnica che permette mediante software predisposti a questo compito di ritardare l'uscita dall'interfaccia di rete di specifici datagrammi UDP/segmenti TCP, dando priorità magari agli uni piuttosto che agli altri grazie a software che permettono di riconoscere la tipologia di pacchetto che sta attraversando la scheda di rete in funzione dei dati presenti all'interno dell'intestazione, e di decidere in quale delle code a diversi livelli di priorità precedentemente create impilare il

pacchetto, andando poi a livello pratico a limitare quella che è la velocità di invio/ricezione disponibile per un determinato flusso e/o processare ed eventualmente reinstradare un pacchetto prima rispetto ad un altro a prescindere dall'ordine di arrivo di questi ultimi.

CAPITOLO 3

Filtraggio e blocco, un esempio completo di attacco e difesa, TOR

TOR (The Onion Router) è un overlay network[23], cioè una rete sovrapposta a quella del tessuto tradizionale di Internet, rete che si prefigge come obiettivo l'anonimato nella navigazione sul web al fine di permettere come più aulico degli obiettivi l'aggiramento delle censura e del filtraggio di contenuti e servizi a volte imposti da alcuni enti governativi agli ISP o dagli ISP stessi.
Utilizzando un'infrastruttura di rete ed algoritmi di oscuramento dei pacchetti volti a difendere l'utenza contro software di analisi del traffico e strumenti di sorveglianza della rete, le tempistiche di latenza nella trasmissione/ricezione dei pacchetti aumentano di conseguenza rendendolo uno strumento non efficiente da utilizzare per la navigazione quotidiana in rete bensì un ottimo alleato per particolari casistiche, come ad esempio per la navigazione su siti internet che presentano un Top Level Domain '.onion', normalmente non risolvibili mediante tradizionali query DNS nei confronti di qualsiasi DNS server.[24]
Per il collegamento all'overlay network viene utilizzata una versione modificata del Browser Firefox, il quale all'avvio dell'applicazione, mediante l'uso di uno degli ormai molteplici algoritmi di discovery degli IP dei nodi di ingresso alla rete TOR e tramite un handshake iniziale, che mima in tutto e per tutto quello applicato nello standard TLS, crea un tunnel di comunicazione con uno dei nodi di ingresso della rete.

3.1 Infrastruttura Rete

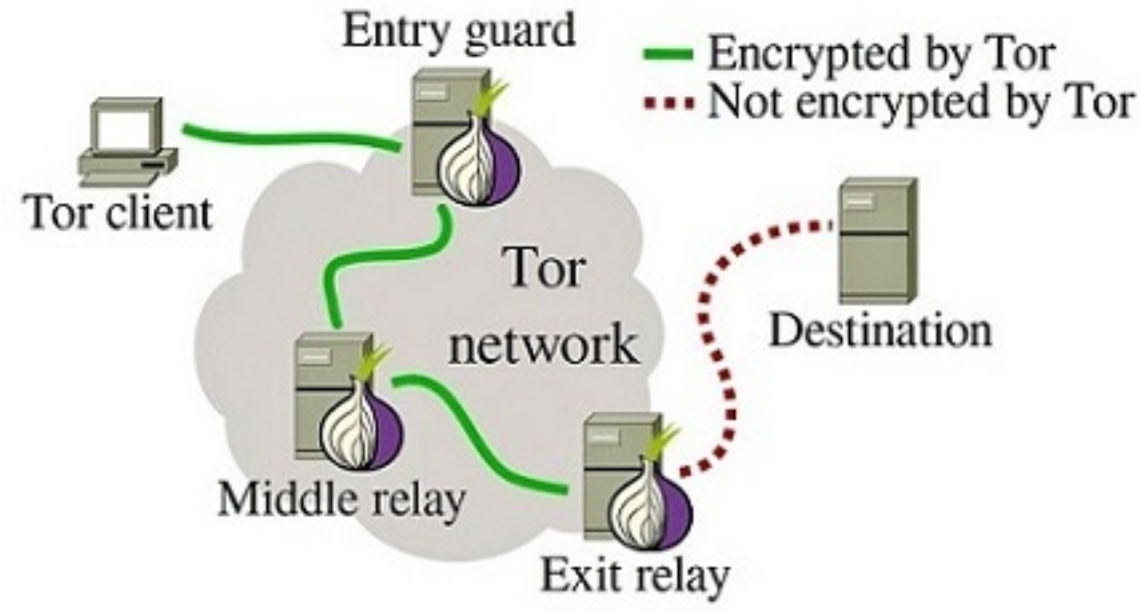


Figura 3.1: ***Principali attori attivi all'interno dell'infrastruttura di rete TOR, l'immagine simula un circuito TOR attivo con destinazione della risorsa cercata esterna alla rete TOR.***

Il grafico sovrastante rappresenta in maniera sintetica ma esplicativa quelli che sono i protagonisti in gioco all'interno della rete TOR:

- il Client Tor, cioè il browser agent modificato precedentemente citato il quale gioca un ruolo fondamentale nella fase iniziale di 'discovery' per la connessione con uno degli entry nodes della rete.

- gli entry nodes, nodi (in gergo relays) che fungono da frontiera per chi è intenzionato a navigare all'interno dell'overlay TOR, sono server che presentano socket (configurate in maniera anche molto eterogenea tra un entry node e l'altro) in attesa di un qualche client si presenti con un pacchetto Hello Client. Nodi di ingresso che, una volta terminato l'handshake, rimarranno per il client associato il primo hop in trasmissione e l'ultimo hop in ricezione dei pacchetti relativi al suddetto client, una perdita della connessione con il nodo di entrata comporta un'impossibilità di navigazione all'interno della rete fino all'instauramento di un nuovo tunnel TLS con un altro (o lo stesso) nodo di ingresso.

- i middle relays, nodi i quali fungono da 'router' per instradare il traffico all'interno dell'overlay verso risorse all'interno di server interni alla rete o verso nodi di uscita nel caso in cui le risorse richieste dal Client associato si trovino all'interno della rete internet tradizionale, non esistono tabelle di routing e l'instradamento viene

effettuato mediante una serie di DHT che interconnettono tra loro i nodi della rete ed attraverso una evoluzione dell'algoritmo di Onion Routing, trattato nel seguito di questa dissertazione.

- gli exit relays, nodi prediposti alla comunicazione con l'esterno della rete, sono i gateway di uscita dall'overlay che si fanno carico delle richieste di risorse web presenti nella rete Internet da parte dei client all'interno della rete TOR camuffandole come proprie e reinstradando poi quella che è la risposta ottenuta all'interno della rete seguendo il circuito instaurato originariamente dal richiedente della risorsa.

Il grafico mostra che tutto il traffico che va e viene da un client TOR fino all'exit node designato risulta criptato, mentre la richiesta che viaggia nella rete Internet tradizionale si comporta come un qualsiasi pacchetto della rete, avente IP sorgente camuffatto con quello dell'exit node (come nel caso delle VPN, il Server con la risorsa richiesta non risponde al Client che si cela dietro la VPN, bensì nell'intestazione IP del pacchetto il richiedente della risorsa per il Server di risposta è il VPN Server dietro al quale il Client si nasconde, VPN server che si preoccuperà poi di reinstradare la risorsa verso il legittimo richiedente).

3.2 Circuito TOR

Per instradare un pacchetto all'interno della rete TOR cercando di garantire l'anonimato del mittente e proteggere il contenuto dei dati interni al messaggio, l'algoritmo utilizzato dai nodi della rete permette di creare un percorso, definito nel gergo dell'Onion Routing per l'appunto circuito[25], tra il client TOR ed uno dei nodi di uscita attraverso una scoperta incrementale da parte del mittente di una porzione dell'overlay, il cui primo hop è dato appunto dall'entry node della rete, il quale sarà collegato e comunicherà direttamente con altri relays dell'overlay mediante DHT, il percorso seguito non sarà di tipo ottimale dal punto di vista della latenza e della distanza fisica, in quanto un comportamento del genere può permettere ad avanzati software di analisi del traffico di rete di poter determinare più o meno efficacemente la sorgente del messaggio mediante algoritmi di Timing Attack, tipo di implementazione che comporta così tempi di latenza per l'invio e la ricezione dei pacchetti non predicibili a priori indifferentemente da quello che è lo stato attuale della rete o della distanza fisica della risorsa cercata. A questo c'è da aggiungere che una volta stabilito un circuito verso uno degli exit nodes presenti o verso uno dei nodi interni della rete in base al tipo di risorsa richiesta, questo circuito rimane attivo per 10 minuti, a meno di disconnessione da parte del client ovviamente, soluzione implementativa che è stata adottata per ragioni di efficienza computazionale (stabilire un circuito verso un nodo della rete è il passo più esoso dal punto di vista dell'uso di risorse computazionali e temporali per un Client TOR ed inficia direttamente in maniera comunque minore ma significativa i nodi associati a quel circuito).

3.2.1 Onion Routing e Handshake Diffie-Hellman

La creazione di ogni circuito della rete TOR basa le proprie fondamenta teoriche sull'architettura di rete dell'Onion Routing (instradamento a cipolla), nella quale i messaggi transitanti per il circuito vengono incapsulati da più strati di crittografia, i quali poi vengono 'sbucciati' ad ogni hop dal relativo nodo del circuito associato a quello strato di crittografia.[25]
Questa architettura di rete è stata sviluppata negli ultimi anni '90 presso lo U.S. Naval Research Laboratory con lo scopo di protezione del sistema di intelligence degli Stati Uniti, è alla base della creazione dei circuiti della rete TOR, il quale algoritmo di instradamento dei pacchetti presenta caratteristiche e funzionalità aggiuntive per ottimizzare tempi di latenza, throughput tra gli end-point della comunicazione, garanzia sulla ricezione dei pacchetti, peculiarità del tutto analoghe a quelle presenti nel protocollo di trasporto TCP.
Il Client ad un lato dei due end-point nell'instauramento del circuito negozierà ad ogni hop una chiave simmetrica con uno dei nodi appena scoperti il quale diventerà designato ad entrare all'interno del circuito, così avanti fino all'exit node o alla risorsa interna all'overlay andando infine, una volta completato il circuito, a crittografare in maniera sequenziale ogni pacchetto con le chiavi associate ad ogni nodo di ogni hop del circuito, permettendo così, al transito del pacchetto sui nodi del circuito, il decriptaggio solamente del relativo strato di crittografia associato alla chiave scambiata tra mittente ed il nodo in questione.[25]
Ad ogni hop il nodo designato a sbucciare quello strato di crittografia è così in grado di determinare gli unici due aspetti fondamentali ai quali è interessato per una efficace politica di instradamento senza conoscere alcuna informazione relativa al circuito del quale è all'interno o su quale sia la sorgente del pacchetto, i quali sono appunto il nodo dal quale il pacchetto proviene (il nodo dell'hop precedente, non il nostro mittente!) e qual'è la destinazione del prossimo hop verso la quale instradare il pacchetto.
La negoziazione delle chiavi precedentemente citata per l'Onion Routing che viene utilizzata dalla rete TOR avviene mimando in tutto e per tutto un handshake TLS (ho utilizzato il termine mimare in quanto ovviamente i certificati utilizzati nella rete TOR hanno una differente struttura rispetto a quelli standard del protocollo rilasciati dalle CA, sono di tipo self-signed ed il browser agent modificato si comporta in maniera differente in funzione della ricezione di un certificato proveniente da una CA e di uno proveniente da un relay TOR), utilizzano un Cipher che implementa l'algoritmo di criptaggio Diffie-Hellmann[26], il cui scopo ultimo è quello di scambiare una chiave segreta condivisa all'interno di un canale insicuro tra due soggetti senza la necessità che essi si scambino ulteriori informazioni che potrebbero compromettere l'anonimato dell'identità di uno dei due, mantenendo così il più elevato standard possibile di oscuramento di uno nei confronti dell'altro.
Alla base dell'algoritmo Diffie-Hellman vi sono diversi Cipher del protocollo TLS che utilizzano per l'appunto questo processo matematico o una sua variante per la negoziazione delle chiavi, che in questo caso è una sola (Key o Secret) rendendo così la crittografia simmetrica. La mancanza di un sistema a chiave pubblica/privata che negli altri capitoli può essere visto come uno dei grandi mali da risolvere alla base del protocollo TLS il quale si prefigge l'autenticazione e l'IDENTIFICAZIONE del soggetto con il quale si sta cercando di comunicare, nel caso dell'Onion Routing invece viene sfruttato come perno sul quale

costruire un sistema robusto per offrire un alto standard di anonimato e nella creazione di un circuito di comunicazione sicuro.

3.3 Debolezze e problemi noti, Great Firewall Of China VS TOR

Per come è stato descritta, la rete TOR sembra una cassaforte invulnerabile nella quale, una volta entrati, si è liberi ed al sicuro da ogni ipotetico attaccante maligno che voglia in qualche modo intercettare, limitare e visualizzare in chiaro il nostro traffico dati, nulla di più sbagliato ovviamente. Se decriptare anche solo un pacchetto della rete TOR richiede uno sforzo computazionale immenso ed una conoscenza della rete che comporterebbe un numero di PC infetti ed un esborso di risorse mastodontico, l'overlay presenta alcune problematiche venute a galla con l'espansione del bacino di utenza e il relativo aumento di interesse da parte di enti governativi e privati, problematiche delle quali sono stati risolti o mitigati alcuni aspetti, ma che comunque non sono intrinsecamente collegate a questioni di sicurezza della rete, bensì più correlate a come il traffico di rete dell'overlay TOR si comporta, i cui tratti distintivi sono stati utilizzati dagli enti sopracitati (nello specifico mi riferisco al caso del Great Firewall of China trattato in seguito) come strumento per il filtraggio ed il blocco del servizio da parte degli ISP.
Il blocco più semplice che si può effettuare tramite una regola firewall è il blocco del forwarding di ogni pacchetto avente come destinazione la porta 9001 o la 9030, le quali rappresentano porte standard sulle quali le socket degli entry nodes della rete rimangono solitamente in ascolto.
Altro problema non di poco conto è come i TOR Client effettuano la discovery degli indirizzi IP degli entri nodes della rete, la maggior parte della rete presenta relays con indirizzi pubblicamente consultabili, basterebbe quindi una semplice funzione php/javascript che scandagli le varie pagine web che riportano la lista degli IP pubblici della rete TOR, come ad esempio quella riportata nell'immagine qui sotto[27], per creare regole firewall efficaci a bloccare ogni tentativo di invio e ricezione di pacchetti nei confronti di una specifica tupla indirizzo IP:porta.

Router Name	Bandwidth (KB/s)	Uptime	Hostname
reactortornode	51858	3 d	tornode.torreactor.ml [78.109.23.1]
aurora	46501	18 d	aurora.enn.lu [176.126.252.12]
dopper	40077	22 d	freedom.ip-eend.nl [192.42.113.102]
chulak	38062	18 d	chulak.enn.lu [176.126.252.11]
apx2	37926	24 d	tor-exit.r2.apx.pub [185.38.14.171]
apx1	36001	24 d	tor-exit.r1.apx.pub [185.38.14.215]
malene	35727	21 d	185.73.220.8 [185.73.220.8]
calliprhugenasty09	35295	43 d	46.166.148.176 [46.166.148.176]
calliprhugenasty10	34446	43 d	46.166.148.177 [46.166.148.177]
csailmitexit	33763	40 d	tor-exit.csail.mit.edu [128.52.128.105]
Onyx	33624	22 d	onyx.ip-eend.nl [192.42.115.102]
cry	33605	22 d	cry.ip-eend.nl [192.42.115.101]
hviv104	32639	7 d	tor-exit.hartvoorinternetvrijheid.nl [192.42.116.1
Dylan	32466	20 d	dylan.exit.torworld.org [62.210.245.138]
hessel1	31619	33 d	hessel2.torservers.net [109.163.234.4]
0x3d004	31393	15 d	shop2.peptest.ch [62.138.7.171]
memcpy	30451	6 d	tor-exit-readme.memcpy.io [163.172.67.180]
redjohn1	30415	60 d	redjohn.tk [62.210.92.11]

Figura 3.2: ***Screen preso da uno dei siti web consultabili per verificare lo stato dei nodi pubblici della rete TOR.***

Ultimo importante tassello debole di questa catena è la Cipher Suite List utilizzata dal Browser Agent dei Client TOR negli handshake TLS con gli entry nodes della rete, Cipher List statica che ad oggi risulta univocamente associata al Browser Agent TOR, router o comunque hardware con funzionalità di firewall che vanno letteralmente a caccia della stringa esadecimale associata alla Cipher Suite List di un Client TOR sono in grado di effettuare il blocco dei pacchetti Hello Client stroncando sul nascere ogni tentativo di handshake con un entry node della rete TOR.[28] Questo è per l'appunto il caso del Great Firewall Of China[29] dove il governo, che controlla direttamente i due principali ISP, China Telecom e China Unicom, utilizza software per l'inspection dei pacchetti in transito sui loro router alla ricerca, tra le tante cose, di questa specifica stringa esadecimale all'interno di pacchetti TCP : ***30 c0 2b c0 2f c0 0a c0 09 c0 13 c0 14 c0 12 c0 07 c0 11 00 33 00 32 00 45 00 39 00 38 00 88 00 16 00 2f***, la quale rappresenta per l'appunto la lista dei Cipher supportati dal Client Browser Agent TOR.

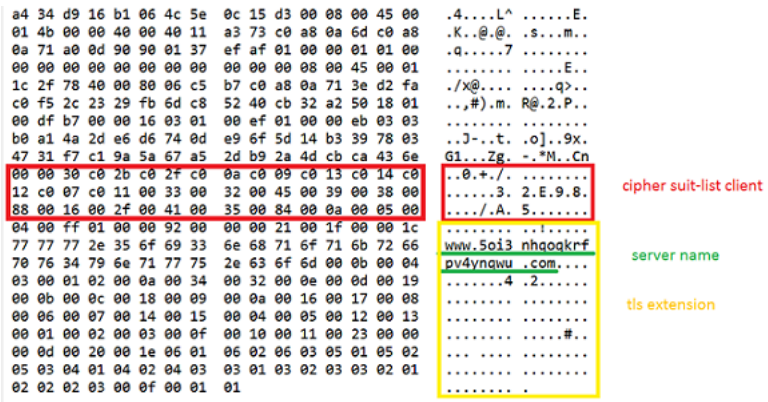


Figura 3.3: ***Estratto Wireshark di un pacchetto Hello Client, in rosso evidenziata la Cipher Suite List associata ai Browser Agent TOR.***

3.3.1 Contromisure adottate, Bridges e Pluggable Transports

Alle prime due problematiche la soluzione adottata per ovviare al blocco IP:porta è stata quella di utilizzare i cosidetti 'TOR Bridges'[30] per l'instauramento della connessione TLS iniziale con un entry node.
Questi bridges non sono altro che entry nodes non pubblici e quindi i loro IP non risultano presenti all'interno della lista dei relays della rete, nella parte di configurazione del browser relativa alla prima connessione con l'overlay network è possibile selezionare l'opzione di connettersi ad un default bridge o di inserire la configurazione manuale di un bridge nel caso ci si voglia connettere specificamente ad un bridge noto o tra quelli reperibili sul web che si aggiornano ed i quali IP variano in continuazione appunto per rendere la vita difficile a blocchi di tipo IP:porta. Riguardo invece al problema della Cipher Suite List, il più difficile da risolvere a causa della sua natura intrinsecamente statica, si sono susseguiti col tempo una dozzina di algoritmi di offuscamento del traffico, chiamati nel gergo dell'ambiente TOR 'Pluggable Transports'[28], algoritmi che in sostanza mantengono inalterato l'header TCP/IP ma che effettuano operazioni di offuscamento del resto dei dati in maniera differente a seconda del tipo di Pluggable Transport utilizzato, andando così a rendere impossibile il fingerprinting da parte di qualsiasi firewall alla ricerca del pattern relativo alla Cipher Suite List.

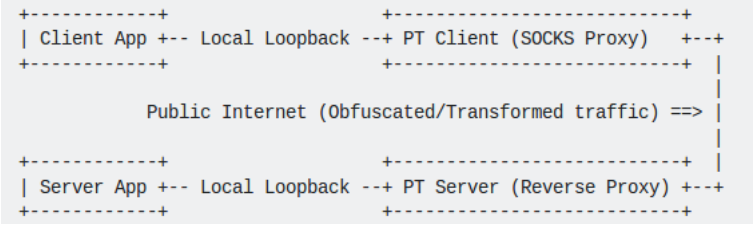


Figura 3.4: ***Flusso logico seguito per l'offuscamento del traffico TOR, sono le socket di Client e Server a dover essere opportunamente configurate con lo stesso Pluggable Transport per far sì che l'handshake vada a buon fine.***

Qui sopra viene mostrato il flusso di rete associato ad una comunicazione tra il Client e l'entry node in una rete TOR che sfrutta i Pluggable Transport come metodo di aggiramento della censura, entrambi necessitano di configurare la socket predisposta alla comunicazione in modo tale da accordarsi sul tipo di pluggable tranport applicato e poter offuscare e rischiarire i pacchetti che vengono scambiati tra le due socket poste agli end-point di comunicazione.
Questo è ad oggi il migliore strumento di difesa utilizzato per aggirare la censura ed il blocco del servizio TOR causato dal Great Firewall of China. In realtà gli analisti dietro al firewall cinese lo hanno fatto evolvere di fronte a questo raggiro riuscendo a creare regole di blocco analizzando i tempi di latenza tra invio e ricezione di pacchetti e la dimensione di questi ultimi (anche mediante pluggable transport, la dimensione dei pacchetti di handshake rimaneva comunque costante) all'interno di un flusso SSL/TLS, blocco nuovamente aggirato grazie alla creazione di pluggable transports in grado di variare in maniera del tutto casuale la velocità di inoltro di un pacchetto ed andando ad alterarne la dimensione aggiungendo

dati/padding casuali.

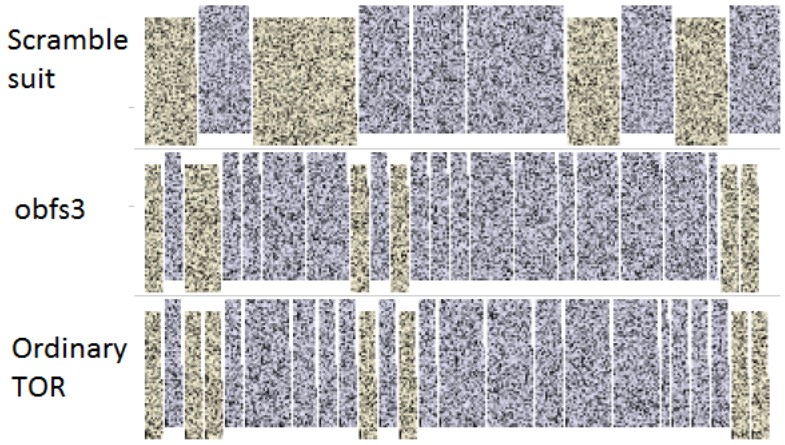


Figura 3.5: ***Handshake a confronto, con e senza Pluggable Transport.***
In giallo i pacchetti inviati dal Client, in grigio quelli di risposta dell'entry node.

Nell'immagine sovrastante sono riportati dall'alto verso il basso pacchetti inviati e ricevuti da un Client TOR nell'ordine:

- Comunicazione mediante Pluggable Transport ScrambleSuit, che aggiunge molto padding e nella velocità di trasmissione e ricezione introduce dei periodi di attesa casuali che vanno a variare la latenza dello scambio di pacchetti.
- Comunicazione mediante Pluggable Transport obfs3, il quale cripta tutti i pacchetti in ingresso ed in uscita mediante una chiave pubblica condivisa in maniera del tutto simile ad un classico handshake Diffie-Hellman, ma con delle variazioni nel criptaggio per rendere il flusso di dati simile ad uno stream di bytes omogeneo e casuale (infatti l'algoritmo utilizzato prende il nome di UniformDH).
- Comunicazione senza Pluggable Transport, traffico TOR ordinario.

Come si può notare, la struttura dei pacchetti offuscati tramite obsf3 e senza Pluggable Transport non differisce molto, questo ha permesso agli analisti dietro al GFW di creare regole, anche molto complesse, per riuscire a bloccare il traffico TOR studiandone appunto la grandezza dei pacchetti e gli intervalli di invio/ricezione di questi ultimi. L'evoluzione del precedentemente citato Scramblesuit (che già di per sè risulta un alleato molto potente nei confronti del GFC), obfs4, ad oggi rappresenta il nemico pubblico n1 per chi come il governo cinese è interessato al blocco delle connessioni su rete TOR, in quanto ancora nessuno è stato ancora in grado di interrompere un tentativo di connessione tra due end-point della rete TOR che utilizzano questo Pluggable Transport nel loro handshake iniziale.

CAPITOLO 4

Filtraggio del traffico, architettura Firewall Analisi, implementazione e risultati

In questa sezione vengono presentati e descritti gli elementi software ed hardware utilizzati per il testing dei filtri per il traffic shaping e delle regole firewall di blocco implementati in base ai concetti dissezionati nei precedenti capitoli, nello specifico verranno descritti:

- l'architettura di rete all'interno della quale è stato effettuato il test.
- RouterOS, sistema operativo del Router da me utilizato come gateway di rete nella prima fase di test, utilizzato però principalmente come strumento per inoltrare verso un dispositivo con Wireshark in ascolto sull'interfaccia di rete associata a questo dispositivo tutto il traffico in ingresso ed in uscita dal network locale.
- Il package netfilter e più nello specifico il subpackage Iptables del kernel Linux per l'implementazione di un firewall con più funzionalità rispetto a quelle fornite dal linguaggio di scripting RouterOS. Vengono descritte visualizzati i risultati ed analizzati i carichi a livello computazionale delle regole implementate.
- nTopng, un 'Traffic Prober' (lett. sondatore del traffico), applicazione integrabile con iptables che offre la possiblità di discriminare ed etichettare pacchetti in funzione del tipo di protocollo (supporta oltre 200 protocolli di comunicazione) usato a livello applicazione tramite Deep Packet Inspection e matching dei pacchetti con specifiche protocollari, riconoscimento nameserver, bit di fingerprint nell'header, latenze di invio/ricezione etc....

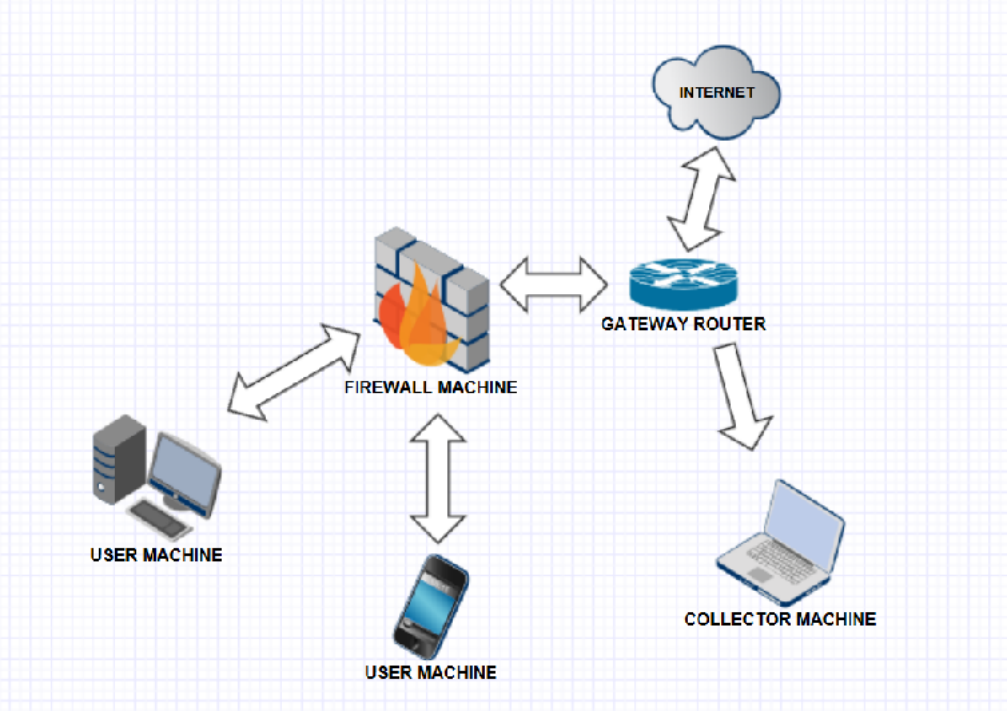


Figura 4.1: ***Schema grafico dell'architettura di rete da me utilizzata per la fase di analisi e di test.***

4.1 Architettura di rete e software

La **User Machine**, macchina fisica con SO Linux Mint versione 18.1, rappresenta l'end-point lato Client di questa simulazione. La scelta di Mint come sistema operativo in questo caso è data per convenienza in funzione del SO già installato sulla macchina durante la fase di testing.
La **Firewall Machine**, con sistema operativo CentOS versione 7, rappresenta la macchina fisica sulla quale verranno implementate le regole firewall mediante l'uso di Iptables ed nTopng.
Il **Gateway Router**, una routerboard Mikrotik con SO RouterOS, rappresenta la nostra interfaccia di comunicazione e di interconnessione tra la rete locale ed Internet.
Infine la **Listener machine**, macchina fisica con SO Windows 10 e con il programma di packet sniffing Wireshark, rappresenta la macchina sulla quale vengono raccolti tutti i pacchetti transitanti per il Gateway router ed analizzati gli handshake TLS tra la User machine ed i vari server disseminati per la rete Internet con il quale il Client tenta di instaurare un tunnel TLS.

4.1.1 RouterOS

RouterOS è il linguaggio di scripting per la programmazione del router da me adottato come strumento per la parte di analisi, grazie ad una delle funzionalità disponibili che permette il forwarding diretto di tutto il traffico transitante per il router verso una specifica tupla IP:porta, che in questo caso è data da una socket sulla macchina in ascolto La comunicazione col router è possibile mediante connessione autenticata SSH su porta 23 o tramite GUI (winbox).

4.1.2 Iptables

Iptables è uno strumento software molto potente, modulo integrante del kernel Linux che offre le funzionalità di un firewall completamente programmabile. I vari pacchetti in ingresso sull'interfaccia di rete della macchina sul quale viene programmato 'attraversano' una serie di catene all'interno delle quali è possibile definire, tra le altre cose, regole per il blocco o meno dei pacchetti che transitanti per quella catena, le catene risultano a loro volta raggruppate all'interno di tabelle, le quali rappresentano delle macroaree suddivise in funzione della tipologia di regole che vi verranno implementate dentro, e sono: Filter, NAT e Mangle.
La tabella Filter rappresenta quella tabella dove inserire le regole di blocco o di transito dei pacchetti, le sue catene sono:

- INPUT per processare i pacchetti con IP destinazione ultimo la macchina fisica sulla quale iptables opera.
- OUTPUT per processare i pacchetti aventi come sorgente la macchina con iptables.
- FORWARD per processare i pacchetti che transitano per la macchina, con aventi quindi sorgente e destinazione differenti dall'indirizzo IP della macchina in questione.

La tabella NAT con le sue catene di PREROUTING e POSTROUTING rappresenta quella tabella nella quale i pacchetti vengono processati per essere reinstradati e quindi per l'appunto NATtati verso una destinazione differente rispetto a quella originale dell'intestazione del pacchetto, la quale appunto viene modificata. Le regole presenti in PREROUTING sono le prime ad essere processate non appena un pacchetto arriva su di una socket aperta nell'interfaccia di rete, le regole di POSTROUTING sono invece le ultime ad essere verificate prima che il pacchetto lasci la scheda di rete, nel mezzo vengono eseguite dapprima le regole della tabella Filter ed in seguito quelle della tabella Mangle.
Come ultima tabella, ma non per importanza, c'è appunto la tabella Mangle, la quale contiene sia le catene che supporta Filter oltre che quelle della tabella NAT, le regole della tabella Mangle vengono espresse per effettuare appunto un mangling (fig. storpiare) i pacchetti possono essere contrassegnati o marchiati con specifiche label le quali possono essere utilizzate per essere inoltrate in catene implementate manualmente o passate ad un'applicazione esterna in grado di operare all'interno dello userland di Iptables.
Essendo iptables componente integrata all'interno del kernel del sistema, tutte le operazioni finora descritte vengono eseguite ad un livello logico molto basso, collegato direttamente all'hardware fisico della scheda di rete, questo principalmente per processare i pacchetti alla massima velocità possibile permessa dall'hardware sul quale stiamo facendo girare iptables, l'uso opzionale dello userland da parte di altri software (esempio nTopng) al fine di processare pacchetti permette da un lato un'estendibilità che si può rendere necessaria quando i controlli da effettuare necessitano di operazioni ben più complesse di quelle offerte dalle regole iptables, ma al contempo comporta una riduzione della velocità di processamento dei pacchetti, questo a causa del differente livello logico nel quale i pacchetti vengono processati, e del diverso dispendio di risorse computazionali utilizzato per processarli. In contesti quindi come possono essere rappresentati da server VPN e Proxy server, nei quali la velocità di

processamento e di inoltro dei pacchetti è direttamente collegata con la latenza e la velocità di navigazione da parte di chi si appoggia a questo genere di servizio, c'è quindi da tenere conto di questo importante fattore nella scelta decisionale di utilizzare o meno un software integrativo nello userland Iptables come strumento d'appoggio per il firewall.

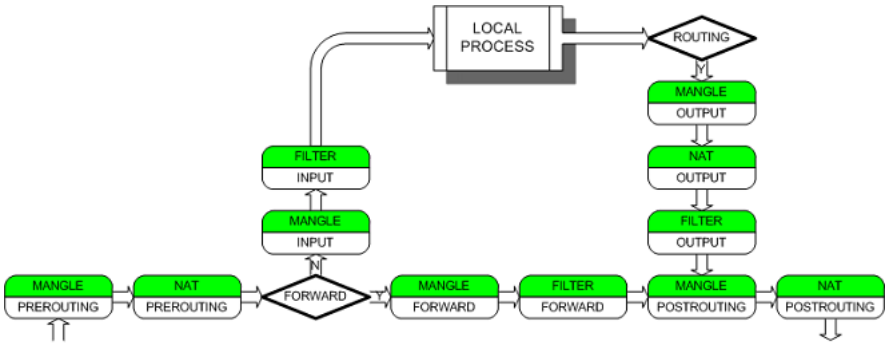


Figura 4.2: ***Flusso logico attraversato dai pacchetti che vengono processati dal firewall Iptables.***

4.1.3 ntopng

Il precedentemente citato nTopng è stato da me utilizzato per applicare regole di blocco iptables più efficienti discriminando esclusivamente gli handshake SSL/TLS nei confronti di tutto il restante traffico TCP, funzionalità che purtroppo TCP non permette.
Teoricamente sarebbe stato possibile creare regole iptables basate sulla verifica della porta destinazione/sorgente con numero 443 (default per HTTPS) o per 9001/9030 (le più comuni per TOR), ma a causa della non standardizzazione del numero di porta nel secondo caso citato, ho preferito sviluppare la mia strategia di filtraggio basandomi sul caso che comprendesse tutte le casistiche possibili, cioè l'identificazione del traffico TLS non solo in funzione del numero di porta, bensì della struttura dell'header nella sua interezza, header nel quale ntopng va appunto ad analizzarne i bytes per effettuare le proprie politiche di etichettatura in caso di collimazione con il protocollo ricercato. Grazie alle funzionalità di etichettatura dei pacchetti in funzione del protocollo identificato (se identificato, ovviamente) mi è stato possibile effettuare politiche di blocco sugli handshake TLS il più efficientemente possibile dal punto di vista del dispendio di risorse computazionale. La possibilità di poter identificare una vasta gamma di protocolli è stata da me utilizzata anche per effettuare una simulazione di traffic shaping sul protocollo bitTorrent con un risultato del tutto analogo a quella effettuata da Comcast, come descritto nel capitolo 1.

4.1.4 tc: Traffic Control

TC è un programma anch'esso parte integrante del kernel linux, il quale offre funzionalità di configurazione e controllo diretto dello scheduler dei pacchetti processati dal kernel. Tc permette, tra le funzionalità presenti, di poter attuare politiche di traffic shaping su specifici pacchetti, potendo creare delle queues (code) specifiche per pacchetti aventi ad esempio uno

specifico IP sorgente, porta, mark etc...

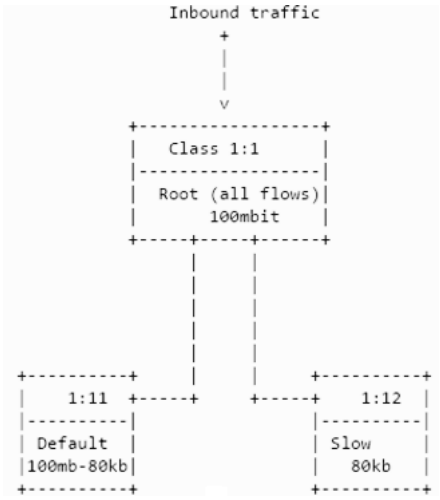


Figura 4.3: ***Schema grafico del flusso differenziato per la coda associata ai pacchetti che verranno marcati come pacchetti bitTorrent e di conseguenza rallentati, e la coda di default sulla quale non verrà effettuato traffic shaping.***

Come è deducibile dall'immagine, nella fase di test sono state create due code, una coda di Default nella quale viene processato alla normale velocità di trasmissione garantitami dall'ISP tutto il traffico non marcato come bitTorrent da ntopng, ed una Slow queue, coda nella quale viene appunto effettuato il traffic shaping dei pacchetti marcati come bitTorrent. Tc di default applica una politica di accodamento di tipo FIFO, all'interno di questo test verrà

4.2 Analisi e collezione dei dati

La parte di analisi e collezione dei dati è stata da me effettuata compiendo simulazioni di normale navigazione su internet da parte dell'utente dietro la User machine, collezionando tutto il traffico prodotto associato all'interno della Listener machine, dati dai quali poi ho estrapolato le stringhe esadecimali da utilizzare poi in seguito nell'iptables della macchina firewall per testarne il relativo blocco. La 'caccia' alle stringhe è stata da me effettuata dapprima andando ad analizzare le Cipher inviate nel pacchetto Hello Client per vedere se esse corrispondessero con quelle relative ad un particolare Browser Agent o un Client di un'applicazione VPN, il problema insorge in questo caso quando si ha bisogno di discriminare in maniera completa una specifica applicazione dalle altre, ad eccezione del caso di TOR la cui già presentata Cipher Suite List ad oggi risulta ancora univocamente associata all'applicazione Client del Browser, ho trovato inefficiente questa strategia in quanto alcuni Client VPN (ExpressVPN, HotspotShield) utilizzavano la stessa Cipher Suite List di una versione di un qualche Browser Agent, la regola associata quindi sarebbe poi andata a bloccare non solo il tentativo di handshake tra Client e Server VPN, bensì mi avrebbe negato ogni possibilità di navigazione con quel particolare Browser Agent su siti che richiedono il protocollo HTTPS per poter essere consultati. La mia attenzione quindi dopo questi primi

tentativi si è focalizzata sul collezionare i valori esadecimali relativi al Serial Number del Certificato che veniva inviato nel messaggio di risposta Hello Server.

4.3 Test

4.3.1 Blocco

Una volta collezionate le varie stringhe esadecimali ho iniziato a creare regole iptables associate a questi valori per testare effettivamente se la comunicazione venisse bloccata o meno. Le regole iptables da me create sono state inserite all'interno della catena FORWARD della tabella Filter della macchina Firewall, questo perché, come già spiegato, la tabella Filter è quella predisposta al blocco o meno di un pacchetto in transito su quella tabella, mentre la catena FORWARD è quella catena dove vanno in ingresso quei pacchetti aventi IP sorgente e destinazione differenti rispetto a quello della macchina sulla quale gira iptables, che è appunto il caso della Firewall Machine.

```
####Modify /etc/ntopng.conf -> add row '-i nf:0' to tell ntopng to start listening and classify packets incoming into the iptables
####netfilter queue n°0#####

iptables -t filter -I FORWARD 1 -j NFQUEUE --queue-num 0
systemctl start ntopng.service

###RULES MATCHING HELLO CLIENT PACKET'S CIPHER SUITE LIST: 1st TOR CLIENT, 2nd HOTSPOTSHIELD VPN CLIENT
iptables -t filter -A FORWARD -m mark --mark 91 -m string --hex-string '|0030c02bc02fc00ac009c013c014c012c007c011003300320
iptables -t filter -A FORWARD -m mark --mark 91 -m string --hex-string '|c02bc02fc00a0009c013c01400330039002f0035000a00ff|' -j

###RULES MATCHING CERTIFICATE PACKET'S SERIAL NUMBER
iptables -t filter -A FORWARD -m mark --mark 91 -m string --hex-string '|c00099b7d789c9f66626317ebcea7c1c|' -j DROP
iptables -t filter -A FORWARD -m mark --mark 91 -m string --hex-string '|33272ff49cf3ff468c2ac24300ceb111|' -j DROP
iptables -t filter -A FORWARD -m mark --mark 91 -m string --hex-string '|4cbd642e234c692792a150b98c69e129|' -j DROP
###1st: DIGICERT CA SERIAL NUMBER FOR FACEBOOK CERTIFICATE
###2nd: THAWTE CA SERIAL NUMBER FOR UNIBO CERTIFICATE
###3rd: SYMANTEC CA SERIAL NUMBER FOR NETFLIX CERTIFICATE
```

Figura 4.4: ***Principali regole per comprendere come è stato effettuato il blocco di specifici servizi Internet.***

Le lables **-m mark** ed **–hex-string** sono associate: la prima al mark specifico che viene applicato da ntopng ad i pacchetti che presentano un'intestazione associata ad un pacchetto TLS (mark 91 in questo caso), la seconda invece comunica ad iptables di preoccuparsi di cercare quella stringa esadecimale all'interno di tutto il contenuto del pacchetto (effettuando fondamentalmente un'operazione di DPI) ed applicare l'azione associata (DROP) in caso di corrispondenza. Con questa spiegazione cerco anche di far capire del perché della mia scelta sul coadiuvamento nei confronti di iptables da parte di un traffic prober esterno, nel caso di ricerca di un pacchetto TLS ntopng effettua l'ispezione esclusivamente sulla parte di intestazione del pacchetto alla ricerca dei bytes di versione SSL/TLS nella parte subito successiva all'header TCP (**16 03 00 per SSL v3.0, 16 03 01 per TLS v1, 16 03 02 per TLS v1.1, 16 03 03 per TLS v1.2**), senza questo accorgimento di etichettatura la ricerca delle

stringhe esadecimali sarebbe stata effettuata su tutto il traffico TCP transitante per quella catena, intestazione e data, mentre in questo caso ci si ferma all'intestazione con ntopng e la ricerca esadecimale da parte di iptables nel data viene resa esclusiva per i pacchetti TLS, rendendo a mio avviso più sensato e selettivo questo processo per un minor dispendio di risorse computazionali.

4.3.2 Traffic Shaping

La scelta da parte mia di effettuare dei test di traffic shaping non sul protocollo TLS, bensì deviando leggermente dall'argomento pregnante di questa tesi e prendere come esempio il protocollo bitTorrent, è stata dettata fondamentalmente da due fattori che poi mi hanno portato a voler approfondire questo aspetto:

- In primis vi è la tangibilità visiva del throughtput associato allo scambio di chunck con gli altri IP presenti nel tracker associato al file Torrent che si sta condividendo, download/upload che all'interno di uno scambio di dati mediante protocollo bitTorrent per come è strutturato il protocollo in sè, ha la tendenza di utilizzare nello scambio tutta la banda disponibile sia in upload che in download. Un traffic shaping su questo specifico protocollo altera in maniera significativa quella che è l'interazione nello scambio di chunck e la conseguente velocità di trasmissione/ricezione.

- Il secondo fattore è dato dal fatto che tra i circa 200 protocolli supportati da ntopng vi è anche l'etichettatura del traffico associato a bitTorrent, in funzione di questo 'due piccioni con una fava' e del precedentemente citato caso di Comcast ho deciso infine di voler approfondire questo aspetto.

4.4 Risultati

Sfruttando il metodo di ricerca della stringa esadecimale associata alla Cipher Suite List dell'Hello Client sono riuscito a bloccare efficacemente ogni tentativo di connessione alla rete TOR mediante configurazione di default e mediante Bridges che non avevano socket configurate per l'uso di Pluggable Transports ed anche il tentativo di instauramento di un tunnel TLS tramite Client VPN (HotspotShield, ExpressVPN), i quali sono stati controllati al fine di non causare conflitto con alcuna versione dei Browser Agent comunemente più utilizzati.
Mediante invece la ricerca delle stringhe esadecimali associate ai Serial Number del Certificato sono riuscito a bloccare ogni tentativo di connessione TLS con i server associati per qualsiasi ente/applicazione associato a quel determinato certificato, stroncando in maniera efficace ogni tentativo di connessione con ad esempio i server di Facebook, di Netflix e dell'Unibo.

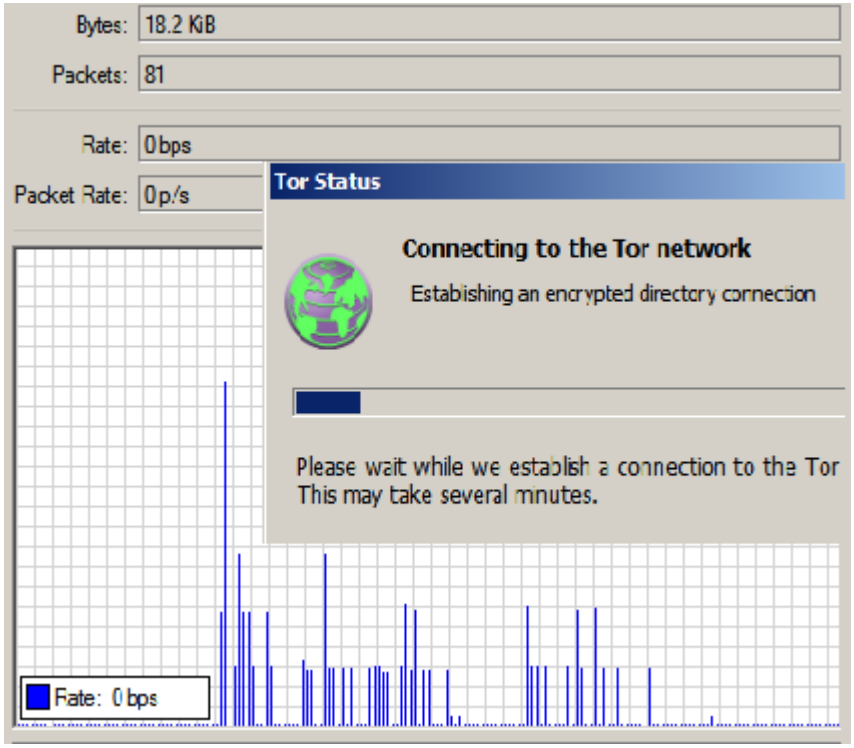


Figura 4.5: ***Grafico di logging dei pacchetti in trasmissione che vengono scartari dal filtro sulla Cipher List associata a TOR Hello Client.***

Nel caso di filtraggio mediante fingerprinting sulla Cipher Suite List il blocco risulta alla massima efficienza possibile dal punto di vista dell'uso della banda di rete in quanto viene negato ogni tipo di comunicazione al di fuori dello scambio SYN/ACK della comunicazione TCP precedentemente instaurata tra Client e Server, dopo un'ottantina di pacchetti pseudo-Hello Client inviati il Client TOR chiude la connessione a causa del timeout di risposta.
Nell'immagine sottostante invece viene mostrato il logging dei pacchetti inviati e ricevuti scartati con il filtro di blocco sul Serial Number associato al certificato rilasciato dal Root CA Thawte nei confronti dell'unibo, filtro il quale da solo è in grado di bloccare l'accesso a qualsiasi sito istituzionale dell'Alma Mater.
In questo caso vi è uno "spreco" maggiore delle risorse di rete della nostra rete in quanto i pacchetti in uscita dal Client passano per il firewall senza colpo ferire, mentre i pacchetti di risposta da parte del Server Unibo con il quale il nostro Client sta comunicando, quando è presente anche la lista di certificati, vengono scartati costringendo il Server alla ritrasmissione di questi ultimi, andando ad utilizzare inutilmente risorse di banda di rete per entrambi gli end-point della comunicazione e ad un dispendio di risorse computazionali che questa volta è bilaterale e non più esclusivamente a carico del Client, in quanto la ritrasmissione dell'Hello Client arriva a destinazione, mentre la risposta processata ed incapsulata dal Server si schianta sul firewall dopo aver attraveresato il gateway della rete interna.
Come si può notare da un confronto dei due grafici, il numero di bytes trasmessi (TX/RX) e di pacchetti scambiati nel caso di blocco su Cipher Suite List è di circa 80 pacchetti /15KiBytes prima che il timeout lato Client (circa 1 minuto) faccia cadere la connessione TCP sulla qual, mentre nel caso di blocco sul Serial Number prima che questo avvenga vi

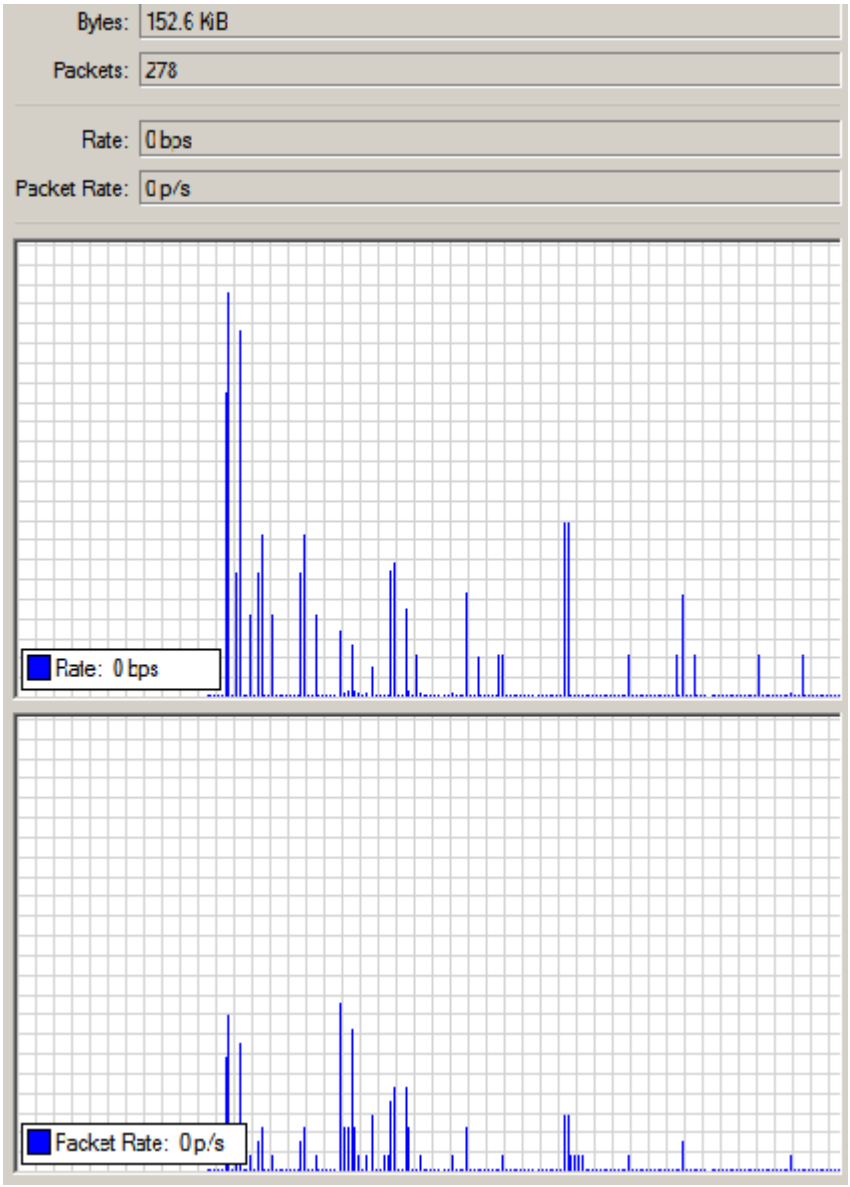


Figura 4.6: ***Grafico di logging che evidenzia flusso bidirezionale tra gli IP del Client e del Server Unibo designato per la risposta, flusso di pacchetti transitante per il firewall e del quale l'handshake non va a buon fine a causa del filtro sul Serial Number associato al Certificato Unibo.***

è una forte interazione tra Client e Server (circa 150KiB spalmati su 280 pacchetti) prima che il Browser Agent risponda all'evento timeout (anche in questo caso si è preso come riferimento temporale il minuto per il confronto dei dati raccolti).

Nella sottostante figura 4.7 invece vengono riportati quelli che sono i risultati riguardo i test da me effettuati sul traffic shaping associato al protocollo bitTorrent. Dalla mia esperienza personale ho denotato un andamento molto irregolare nel flusso di download in quanto le politiche di traffic shaping adottate da tc hanno comportato un throughput in upload non costante, fattore che poi ha causato per via delle specifiche protocollari di bitTorrent in merito alla scelta dei peers con i quali scambiare chunck (unchooked peers) e dei designati ad entrare in questo scambio (optimistically unchooked), scelta che viene effettuata ogni 10

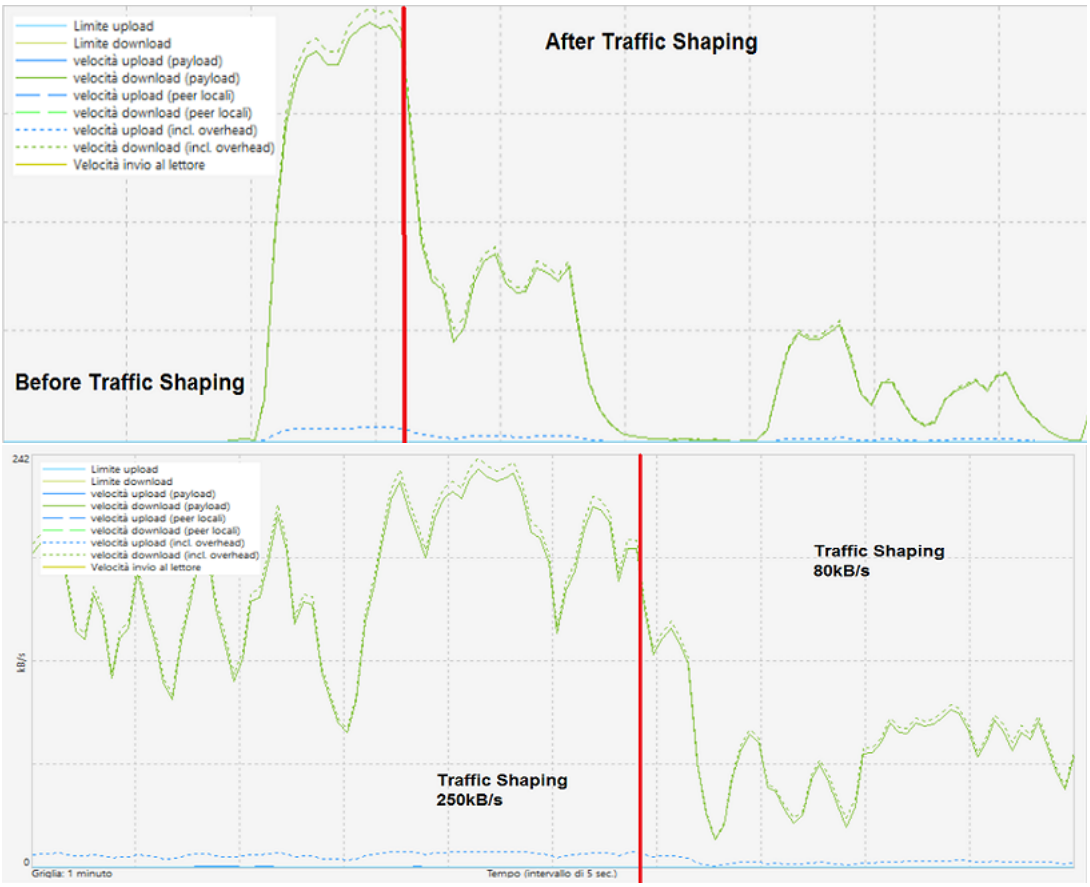


Figura 4.7: ***In alto un grafico di uno scambio di chuck all'interno di un Torrent estrapolato dall'interfaccia grafica di un Client uTorrent, in verde la velocità di download prima e dopo il traffic shaping.***
In Basso l'andamento della stessa interazione in un intervallo di tempo più ristretto e a due diverse velocità di rallentamento.

secondi in funzione delle velocità di trasmissione e ricezione campionate in quell'intervallo temporale per bilanciare in maniera intelligiente gli scambi tra peers in funzione dei loro throughput (peers lenti si scambieranno chunk con peers lenti, peers veloci...).

CAPITOLO 5

Conclusioni

5.1 Chi custodirà i custodi? PGP come alternativa alla PKI

Dai casi riportati nella sezione 1.5 di questo lavoro in merito al ruolo dei CA sulle implicazioni relative alla sicurezza delle infrastrutture che proteggono le chiavi degli Authorities e dell'effetto a catena che un'eventuale compromissione di una chiave privata di un CA può avere nei confronti di tutto il sottoramo di Certificati che hanno come radice quello specifico, ente come accaduto per DigiNotar, è evidente che una soluzione centralizzata come strumento per la certificazione di un'identità digitale come nello standard X.509 basato sull'architettura PKI risulta avere un singolo punto di fallimento, aspetto a mio avviso sconcertante per quelle che sono le ripercussioni dal punto di vista degli attacchi attuabili con una chiave privata compromessa. Oltre a questo c'è da aggiungere del potere che un root CA per l'appunto possiede, citando di nuovo il caso dei Certificati forgiati da Trustwave appositamente per compiere attacchi di tipo MITM pseudo-legittimati dal fine di un migliore controllo aziendale da parte dei privati che richiedevano questo tipo di servizio, giunge spontanea la domanda: chi custodirà i custodi?

Una soluzione alternativa ad un'infrastruttura gerarchica e centralizzata potrebbe essere quella basata sul Web of Trust (lett. rete basata sulla fiducia), tipo di infrastruttura crittografica che si pone il medesimo scopo dell'architettura PKI, cioè offrire un servizio in grado di permettere il maggior standard di garanzia possibile sull'identità digitale di un ente/individuo, spostando però il ruolo certificatore dai CA verso qualsiasi utente della rete. Il protocollo di crittografia PGP (Pretty Good Privacy), sviluppato da Phil Zimmermann nel 1991, risponde proprio a questo concetto, permettendo ad ogni utente di controfirmare un certificato digitale di un determinato soggetto. Basando le proprie fondamenta teoriche sulla teoria del piccolo mondo e dei sei stadi di conoscenza, il sistema di feedback dati dal numero di persone che controfirmano un certificato digitale di un soggetto, è in grado di offrire buone (...pretty good) garanzie sull'identità del soggetto che si cela dietro quel certificato. La struttura dei certificati in sè non differisce molto da quella dello standard X.509, l'handshake iniziale viene effettuato in maniera ibrida mediante crittografia asimmetrica e simmetrica, nell'handshake mediante criptaggio con chiave pubblica e decriptaggio con chiave privata

viene creato un canale sicuro per permettere la negoziazione di una chiave simmetrica che verrà utilizzata poi per crittografare il resto della comunicazione tra i due end-point, questo per via della grande differenza in termini di dispendio computazionale da dover mettere in campo nel caso di una comunicazione crittografata tramite crittografia asimmetrica ed una che invece si basa su crittografia simmetrica (con la bilancia computazionale molto a favore ovviamente per la seconda).
In un mondo come quello di oggi sempre più 'Social', dove l'identità digitale è sempre più direttamente collegata ed intrinsecamente interconnessa con quella fisica, tramite ad esempio account Facebook e profili Twitter, la possibilità di condividere la propria chiave pubblica PGP e rendere il proprio certificato il più 'trusted' possibile secondo un sistema basato su feedback a mio avviso potrebbe essere un'evoluzione sensata e più versatile dell'infrastruttura centralizzata dello standerd X.509.

5.2 Network Neutrality

Nel decorso di questa attività mi sono reso conto di quanto sia facile poter discriminare un servizio da un altro mediante il tipo di protocollo (caso traffic shaping bitTorrent) ed anche mediante un processo del tutto analogo a quello che ho attuato per bloccare l'acceesso a determinati siti mediante il serial number del certificato digitale applicato a politiche di traffic shaping che usano trigger di filtraggio su appunto e non di rifiuto diretto del pacchetto. Anche nel caso di un'infrastruttura basata sul Web of Trust ricadiamo sempre nello stesso problema, cioè l'identificazione di un soggetto, a prescindere da quella che è l'infrastruttura, il protocollo, il tipo di crittografia etc... questo aspetto comporta intrinsecamente sempre e comunque di poter discriminare in qualche modo il traffico di rete associato a quel soggetto nei confronti del resto dei pacchetti che viaggiano per l'Internet, una volta garantita l'identità si garantisce anche un modo di poter escludere il traffico generato dalla macchina fisica dietro quell'identità, e, venuto meno il caposaldo della Network Neutrality, sono convinto che il coltello dalla parte del manico lo avranno ancora più di quanto già non lo abbiano gli ISP, mascherando da specchietto per le allodole un miglioramento della qualità dei servizi offerti (che indubbiamente ci sarebbe) ed ottenendo però in realtà il potere di controllare in maniera davvero invasiva tutti i pacchetti transitanti per le loro porzioni di rete Internet, potere in seguito facilmente sfruttabile da un qualsiasi ente governativo per legittimare la limitazione di un qualsiasi servizio "scomodo" all'interno della rete Internet, come ad esempio si sta già facendo in Cin nel precedentemente citato esempio del GFC nel capitolo 3 di questa dissertazione.

Un coltello è allo stesso tempo un'arma ed un utensile domestico, la differenza tra i due sta tutta negli scopi dietro la persona che lo impugna...

5.3 Lavori futuri

Dalle nozioni apprese durante la stesura di questa dissertazione, il mio interesse attuale mi porta a voler approfondire struttura e funzionamento di OpenPGP, versione open source del precedentemente citato protocollo di crittografia PGP in un'ottica orientata a studiare metodi per poterlo integrare all'interno di un sistema in grado di coesistere con l'infrastruttura PKI oggi utilizzata per l'identificazione digitale e di fornire un ulteriore strumento di validazione per quello che è appunto l'obiettivo della certificazione dell'identità che si pone lo standard X.509.

Riferimenti

[1] **Dr.Thomas Porter**:
https://www.symantec.com/connect/articles/perils-deep-packet-inspection .

[2] **Wireshark**: Sito per il download del packet sniffer *https://www.wireshark.org*

[3] **TLS 1.2 RFC**: attuale standard TLS *https://tools.ietf.org/html/rfc5246*

[4] **What is an SSL Cipher:** *http://www.wisegeek.com/what-is-an-ssl-cipher.htm*

[5] **Stream Cipher vs Block Cipher**: StackExchange discussion
http://security.stackexchange.com/questions/334/advantages-and-disadvantages-of-stream-versus-block-ciphers

[6] **Digital Certificate definition**:
http://www.webopedia.com/TERM/D/digital_certificate.html

[7] **RFC 5280**: Definizione dello standard X.509 *https://tools.ietf.org/html/rfc5280*

[8] **How digital certificates are validated**:
https://technet.microsoft.com/en-us/library/bb123848(v=exchg.65).aspx

[9] **How Browser accept and trust root CAs**:
http://security.stackexchange.com/questions/11464/getting-a-root-ca-accepted-in-systems-and-browsers

[10] **CRIME exploit explained**:
https://www.ekoparty.org/archive/2012/CRIME$_{e}koparty2012.pdf$

[11] **BEAST vs CRIME**: differenze spiegate
http://resources.infosecinstitute.com/beast-vs-crime-attack/

[12] **RC4 deprecated by microsoft browsers due to security issues**:
https://blogs.windows.com/msedgedev/2016/08/09/rc4-now-deprecated/

[13] **Heartbleed bug explained**: *http://heartbleed.com/*

[14] **Bloomberg's paper about NSA using Heartbleed**:
https://www.bloomberg.com/news/articles/2014-04-11/nsa-said-to-have-used-heartbleed-bug-exposing-consumers

[15] **VASCO's security paper about DigiNotar incident**: *https://www.vasco.com/about-vasco/press/2011/news$_d iginotar_r eports_s ecurity_i ncident.html$*

[16] **DigiNotar hack public report**: *https://www.rijksoverheid.nl/ministeries/ministerie-van-binnenlandse-zaken-en-koninkrijksrelaties/documenten/rapporten/2011/09/05/diginotar-public-report-version-1*

[17] **Trustwave admission to sell MITM friendly Certificates***http://www.computerworld.com/article/2501291/internet/trustwave-admits-issuing-man-in-the-middle-digital-certificate–mozilla-debates-punishment.html*

[18] **Browser Agent fingerprinting via Hello Client Message**: *https://isc.sans.edu/forums/diary/Browser+Fingerprinting+via+SSL+Client+Hello+Messages/17210*

[19] **Network Neutrality definition**: *http://www.vox.com/cards/network-neutrality/whats-network-neutrality*

[20] **RFC791**: Internet Protocol standard *https://tools.ietf.org/html/rfc791*

[21] **COMCAST, NetFlix and Net Neutrality**: *https://www.wired.com/2016/07/comcasts-netflix-deal-open-new-front-net-neutrality-war/*

[22] **Traffic Shaping definition**: *http://searchnetworking.techtarget.com/definition/traffic-shaping*

[23] **Overlay Network**: *https://it.wikipedia.org/wiki/Overlay$_n etwork$*

[24] **RFC 7686**: How domains .onion are resolved *https://tools.ietf.org/rfc/rfc7686.txt*

[25] **How TOR circuits are established**: *https://svn.torproject.org/svn/projects/design-paper/tor-design.html*

[26] **Diffie-Hellman algorytm discussion**: *http://security.stackexchange.com/questions/45963/diffie-hellman-key-exchange-in-plain-english*

[27] **Tor Public Nodes Status**: *https://torstatus.blutmagie.de*

[28] **Tor Cipher Suite Issue and Pluggable Transports example reference***https://trac.torproject.org/projects/tor/wiki/doc/AChildsGardenOfPluggableTransports*

[29] **How GFC is blocking TOR**: *http://www.cs.kau.se/philwint/static/gfc/*

[30] **TOR Briges**: *https://www.torproject.org/docs/bridges*

Ringraziamenti

Per te che leggi, che tu sia arrivato qui dal capitolo 1 o che ti sia direttamente fiondato qui a cercare quali cretinate avessi scritto in questa sezione, beh mi dispiace deluderti, mia cara sottospecie di ovillico, ma per una volta sarò serio.
Ringrazio innanzitutto il mio relatore, Andrea Omicini, senza la sua disponibilità mostratami per seguirmi nella fase conclusiva di questo percorso accademico ora non sarei qui a poter scrivere la parola fine a questa tesi, GRAZIE Prof.

Famiglia... Ciao FEEEEEMILYYYYYYY!
Non trovo davvero le parole per descrivervi, sono qui che corro tra Bologna e Cesena mentre batto sulla tastiera queste parole e non nascondo di star trattenendo un grosso groppo alla gola nel pensare a tutti i sacrifici che per me avete fatto in questi tre anni solo per potermi vedere felice, dai viaggi su e giù tra casa e CASA per aiutarmi con i traslochi, quando stavo male, quando solo avevo bisogno di qualcuno con cui poter parlare che mi facesse sentire a casa, le delusioni che avete sopportato per me, gli strilli al telefono che avete ingoiato nei miei periodi di stress da esami, le mangiate che ci siamo fatti ogni volta che siete venuti a trovarmi, il silenzio che siete riusciti a mantenere quando ve l'ho richiesto, l'ansia di avere un figlio squilibrato a spasso nell'universo, le cose dette ma anche quelle che non mi avete voluto dire per non farmi star male.
Famiglia, famiglia mia, mia colonna, mio supporto vitale, mia chiave di volta, mio rifugio e mia salvezza, a voi che nulla mi avete fatto mancare e tutto delle mie pazzie avete voluto assecondare, spero che solo gioia da oggi io a voi possa dare.
Papà, Mamma, Lelle... GRAZIE.

Infine un sincero ringraziamento va ai colleghi di Flashstart, Francesco, Ivan, Meris, Marco: grazie davvero per avermi sopportato ed avere la costanza (...ma soprattuto la scelleratezza) di continuarlo a fare, per quello che mi avete insegnato e per ciò che ancora sto imparando da voi, senza il vostro supporto questo percorso difficilmente si sarebbe concluso con un finale così soddisfacente per quello che è il mio punto di vista, di questo voglio che sappiate che ve ne sarà sempre riconoscente... Grazie

...Beh, che cosa vuoi? sei ancora qui a leggere? Il tuo nome non è tra quelli sopracitati? Evidentemente abbiamo qualche drink in debito o in credito in sospeso altrimenti non si spiegano i tuoi occhi su questa pagina, per te se vuoi la dedica te la faccio a quattrocchi, non vorrei essere arrestato subito dopo la mia laurea a causa della rievocazione dei bei momenti passati... posso dirti solo che a meno di mie fughe in Russia o in Corea del Nord TU CHE LEGGI sai come contattarmi, andiamo a berci qualcosa dai...
ah ovviamente offri tu!