

Servizi Onion

Dalla teoria all'implementazione

Leonardo Migliorelli¹

leonardo.migliorelli@studenti.unicam.it



COMPUTER SCIENCE @ UNICAM

Tesi Unicam - Servizi Onion, dalla teoria all'implementazione

- ▶ Relatore: Marcantoni Fausto
<https://computerscience.unicam.it/marcantoni/>
- ▶ Studente: Leonardo Migliorelli Mat.113920
leonardo.migliorelli@studenti.unicam.it

INDICE

Introduzione

Onion Routing

Creazione del circuito

Chaum Mix

Tor § Onion v2

Obiettivi

Network Design

Dimostrazione Wireshark

Implementazione

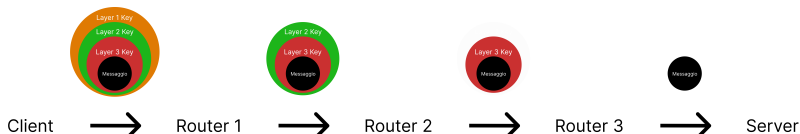
INTRODUZIONE

In questa tesi spiegheremo il funzionamento delle reti **Onion** e parleremo in particolare della rete **Tor**, in fine mostreremo com'è possibile implementare un **servizio Onion/Tor**.

Le reti Onion sono state create per risolvere le due più grandi vulnerabilità di Internet che gravano sulla **privacy** e sull'**anonimato**, ovvero l'**analisi del traffico** e le **intercettazioni**.

Una rete di questo tipo nasconde infatti gli indirizzi e il contenuto di ogni richiesta, consentendo all'utente di navigare in rete senza essere tracciato.

ONION ROUTING



La rete Onion è una rete distribuita composta da nodi chiamati **Onion Router**, collegati tra loro tramite i circuiti creati dai client/proxy.

Ogni pacchetto che passa nel circuito viene decrittato in maniera sequenziale dai relativi nodi prima di essere inoltrato all'exit node che si occupa di instradare il pacchetto nella classica rete Internet.

Grazie a questo meccanismo nessun nodo conosce contemporaneamente l'indirizzo del mittente e del destinatario.

Ci sono 3 proxy usati da Onion:

- ▶ Un proxy che genera e gestisce le connessioni, ha necessità di conoscere la topologia della rete Onion.
- ▶ **Application Specific Proxy**, si occupa di convertire le richieste delle applicazioni in pacchetti Onion.
- ▶ **Application Specific Privacy Filter**, un proxy opzionale che sanifica lo stream di dati rimuovendo informazioni che potrebbero identificare il client.

CREAZIONE DEL CIRCUITO

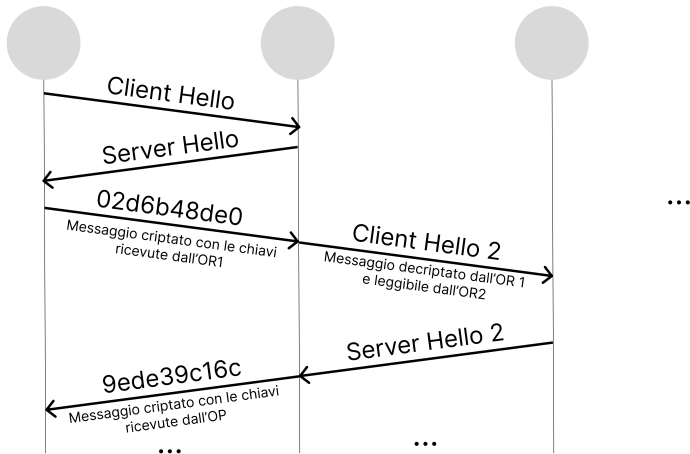
La generazione del circuito è un processo **iterativo** e **progressivo** in cui il proxy server sceglie i nodi del circuito. A partire dal primo nodo viene instaurata una connessione **TLS** e vengono scambiate le **chiavi simmetriche** tramite un processo **asimmetrico** (in maniera simile allo scambio di chiavi di HTTPS), successivamente si usano queste chiavi per cifrare il messaggio che verrà inviato al primo nodo che lo decripta e inoltra al secondo nodo.

Questo processo continua fino all'exit node, a questo punto il circuito è completo e può iniziare a trasmettere i pacchetti.

Onion Proxy

Onion Router 1

Onion Router 2



CHAUM MIX

La rete Onion è basata sullo studio di David Chaum che propose un sistema di comunicazione anonima basato sulla crittografia. Nella sua conclusione una rete di questo tipo doveva avere le seguenti caratteristiche:

- ▶ **Sealing**, una tecnica con cui il messaggio viene annesso ad una stringa casuale prima di essere criptato per aumentarne la sicurezza.
- ▶ Il destinatario deve avere la possibilità di rispondere tramite un indirizzo **non tracciabile** generato dal client a partire da quello reale, tale indirizzo è decifrabile solo dal primo mix che quindi può inoltrare il messaggio al mittente.

Tor

La rete

Tor è la più famosa implementazione di Onion, grazie all'apporto delle seguenti migliorie:

- ▶ Circuiti telescopici
- ▶ Proxy di applicazione tramite SOCKS
- ▶ Controllo di congestione
- ▶ Directory server
- ▶ Politiche di uscita variabili
- ▶ Controllo d'integrità end-to-end



OBIETTIVI

L'obiettivo principale della rete TOR è quello di garantire l'anonimato dell'utente finale, e scoraggiare eventuali attaccanti, sono stati quindi definiti i seguenti obiettivi:

- ▶ Usabilità, la rete deve essere utilizzabile da chiunque, questo è un'aspetto fondamentale per garantire l'anonimato.
- ▶ Semplicità, la rete deve essere semplice da utilizzare, in modo da non scoraggiare gli utenti meno esperti.

NETWORK DESIGN

La rete Tor è una rete che esiste al di sopra delle reti esistenti, per questo viene chiamata overlay network. I pacchetti TOR sono chiamati celle, sono di dimensione fissa 512 bytes, ci sono due tipi di celle:

- ▶ **Control**, gestisce la connessione
- ▶ **Relay**, trasporta stream dati

L'header oltre all'ID del circuito possiede un campo comando che indica come gestire il payload

- ▶ Padding, mantiene viva la connessione
- ▶ Create, crea un circuito
- ▶ Destroy, distrugge un circuito

Applicazioni

Rete TOR

Trasporto (TCP)

Internet

Network Access

GENERAZIONE DEL CIRCUITO

L'onion Proxy per generare un circuito segue un processo di negoziazione incrementale

1. Invia una richiesta relay al primo nodo
2. Avviene la condivisione della chiave simmetrica tramite l'handshake di Diffie-Hellman con la conseguente generazione dell'ID del circuito
3. Viene inviata una richiesta Relay **Extend** indicando l'indirizzo del secondo nodo
4. Iterativamente vengono inviate altre richieste Relay Extend finché il circuito non è completo

Da qui l'Onion Proxy inizia a gestire le richieste di applicazione

CONTROLLO DI CONGESTIONE

Non potendo in una rete Onion cambiare un router quando non è più in grado di gestire il carico, è stato necessario implementare un controllo di congestione che blocca i circuiti o gli stream dati quando questi inviano troppi dati usando due finestre:

- ▶ Packaging window, tiene traccia del numero di celle che un OR può inviare all'OP, viene decrementata ad ogni cella verso l'OP ricevuta e incrementata ad ogni relay sendme ricevuto (dall'OP), quando raggiunge 0 smette di leggere e inoltrare celle
- ▶ Delivery window, tiene traccia del numero di celle che un OR può inviare fuori dalla rete

L'unica differenza tra i due controlli è la dimensione delle finestre e se si riferisce al circuito o allo stream.

TOR RELAY

La rete TOR si basa su un insieme di nodi gestiti da volontari chiamati TOR Relay, possono essere di tre tipi:

- ▶ **Non-exit Relay**, i nodi interni della rete che a loro volta si dividono in:
 - ▶ **Guard Relay**, il primo nodo del circuito
 - ▶ **Middle Relay**, i nodi intermedi del circuito
- ▶ **Exit Relay**, i nodi di uscita della rete Tor, instradano il traffico nella rete comune. Essendo gli unici IP visibili all'esterno sono i più esposti a rischi legali.
- ▶ **Bridge Relay**, nodi non pubblici che non possono quindi essere bloccati

Tutte le informazioni sui Relay esistenti sono visitabili al seguente indirizzo:

<https://metrics.torproject.org/rs.html>

DIRECTORY SERVERS

I Directory Server sono un sottogruppo di onion router che tracciano i cambiamenti nella **topologia di rete** e agiscono da **DNS server** per i servizi onion. Mantengono infatti i **descriptor**, pacchetti generati dai servizi onion criptati con la propria chiave privata, che contengono gli introduction points e la chiave pubblica.

Il sistema di generazione di indirizzi onion fornisce un **meccanismo di sicurezza** per evitare che un malintenzionato possa alterare i descriptor e reindirizzare gli utenti ai propri introduction points, infatti se così fosse la chiave pubblica nascosta nell'indirizzo onion non sarebbe in grado di decifrare il descriptor.

DIMOSTRAZIONE WIRESHARK

In Wireshark possiamo vedere il Client Hello, l'inizio della connessione con il server, in cui il client invia alcune informazioni tra cui la versione di TLS, la lista dei Cipher Suite supportati e la chiave pubblica. Possiamo poi vedere il Server Hello, ovvero la risposta del server con le relative informazioni di crittografia.

Successivamente Tor usa le informazioni ottenute per stabilire una connessione TLS tra client e server, impedendoci di vedere il traffico dati.

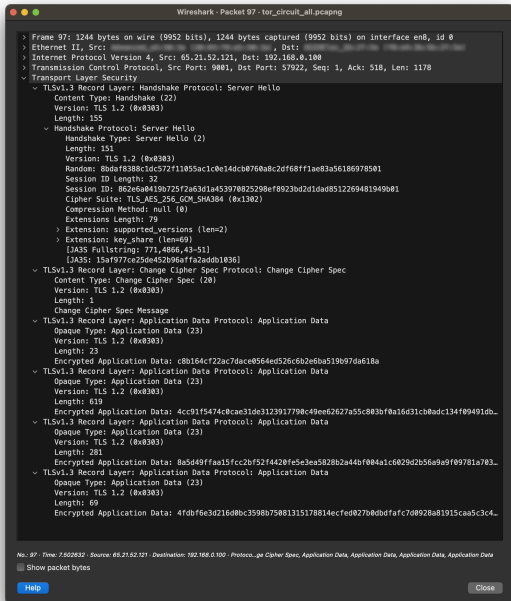
Wireshark · Packet 95 · tor_circuit_all.pcapng

```
> Frame 95: 583 bytes on wire (4664 bits), 583 bytes captured (4664 bits) on interface en8, id 0
> Ethernet II, Src: [redacted], Dst: [redacted]
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 65.21.52.121
> Transmission Control Protocol, Src Port: 57922, Dst Port: 9001, Seq: 1, Ack: 1, Len: 517
> Transport Layer Security
  ✓ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  ✓ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 508
    Version: TLS 1.2 (0x0303)
    Random: 3c0020d5b4afe7e6e0e8b787fcc9f8de8e58f1094320a0169345928eb4d6dfca
    Session ID Length: 32
    Session ID: 862e6a0419b725f2a63d1a453970825298ef8923bd2d1dad8512269481949b01
    Cipher Suites Length: 36
  > Cipher Suites (18 suites)
    Compression Methods Length: 1
  > Compression Methods (1 method)
    Extensions Length: 399
  > Extension: server_name (len=37)
  > Extension: ec_point_formats (len=4)
  > Extension: supported_groups (len=6)
  > Extension: session_ticket (len=0)
  > Extension: encrypt_then_mac (len=0)
  > Extension: extended_master_secret (len=0)
  > Extension: signature_algorithms (len=48)
  > Extension: supported_versions (len=9)
  > Extension: psk_key_exchange_modes (len=2)
  > Extension: key_share (len=71)
  > Extension: padding (len=178)
    [JA3 Fullstring: 771,4866-4867-4865-49199-52393-52392-49196-49200-49162-49161-49171-49172-51-57-
    [JA3: 140e0f0cad708278ade0984528fe8493]
```

No.: 95 · Time: 7.435011 · Source: 192.168.0.100 · Destination: 65.21.52.121 · Protocol: TLSv1.3 · Length: 583 · Info: Client Hello

☐ Show packet bytes

Help Close



IMPLEMENTAZIONE

L'implementazione di un servizio onion consiste nell'utilizzare il **Proxy Onion** per inoltrare le richieste dalla rete Tor al server web, il proxy gestisce la generazione delle chiavi, dell'indirizzo e la definizione e connessione con gli **introduction points**, oltre alla generazione del descriptor e la sua pubblicazione nei Directory Servers.

In particolare la nostra implementazione userà **Onion V3** (dato che Onion V2 è stato deprecato) con alcune migliorie tra cui la maggior sicurezza degli indirizzi.

Useremo un server **Linux EC2** di AWS per ospitare il proxy onion e un **server nginx**.

Innanzitutto è necessario configurare correttamente la macchina virtuale, configurando il firewall AWS (security groups) per permettere le connessioni in ingresso sulla porta 80 e 22.

Nel sistema installiamo i pacchetti nginx (web server) e apt-transport-https (per utilizzare i repository tramite HTTPS). Dopo una serie di configurazioni dei repository possiamo installare tor, configuriamo il relativo torrc con HiddenServiceDir /var/lib/tor/hidden_service e HiddenServicePort 80 unix:/var/run/website.sock per indicare la directory dove verranno salvate le chiavi e connessione del .