

DeAnonimize TOR Network

whit fingerprinting analisys

Paola Guarasci

July 2022

Il progetto prevede la deanonimizzazione di un utente TOR a partire dalle fingerprint dei siti web visitati.

In particolare e' possibile creare una correlazione tra le risorse e il traffico generato per la loro fruizione.

La correlazione avviene facendo un'analisi statistica dei pacchetti TCP che viaggiano attraverso la rete TOR.

Come funziona TOR

La rete Tor e' una rete di server connessi tra loro attraverso tunnel virtuali (detti circuiti) che consentono di anonimizzare il traffico su internet.

Il protocollo di connessione e' detto Onion perche', proprio come gli strati di una cipolla, aggiunge layer di crittazione ad ogni "rimbalzo" che compiono i dati originali.

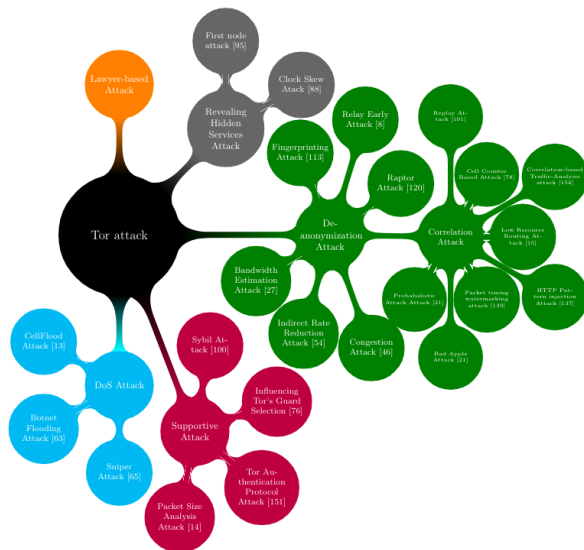
All'interno della rete TOR abbiamo tre tipi di server:

- **Guard Relay** Nodo di entrata della rete
- **Middle Relay** Uno o piu' server interni al circuito
- **Exit Relay** Nodo di uscita dalla rete (comunica in chiaro con l'application server)

Esiste poi un server particolare, detto **Directory Server** che rilascia ai client le liste di nodi della rete.

La generazione del circuito avviene casualmente. La selezione del circuito e' ad opera del client TOR.

Attacchi alla rete TOR

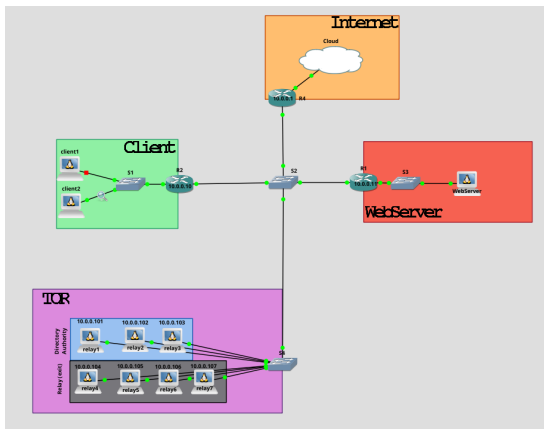


Un attacco fingerprint puo' essere realizzato in ambiente chiuso e in ambiente aperto. In ambiente chiuso Darth e' a conoscenza di tutti i siti web visitati dall'utente TOR, in ambiente aperto invece Alice e Bob possono visitare ogni sito web esistente. Per realizzare questo progetto e' stato creato un ambiente chiuso, in GNS3, in cui il possibile ventaglio dei siti visitabili e' limitato a 5. Nel blog del progetto TOR si trova una descrizione piu' dettagliata di questo tipo di attacco. Un attacco di questo tipo prevede che Darth si posizioni tra il client e il primo relay del circuito, il Guard relay. Potrebbe ad esempio essere sfruttato da un ISP per censurare alcuni contenuti.

Le informazioni analizzate sono: lunghezza dei pacchetti, timing.

La metodologia scelta per la demo è lo Sniffing tra Client e ISP. Questo tipo di attacco consente di comprendere quali website sta visitando il nostro target, anche se il traffico passa attraverso una rete TOR. Il punto principale dell'attacco è l'analisi dei pacchetti. Quindi facendo inferenze su dimensione, direzione e quantità dei pacchetti TCP (le fingerprint) si può ipotizzare a quale sito (noto) appartiene il traffico generato.

Il laboratorio GNS3 realizzato per questo progetto conta un totale di 10 host Debian 11 emulati con QEMU e Tor 0.4.5.10, 3 router Cisco 7200 e 4 switch Cisco 3745.

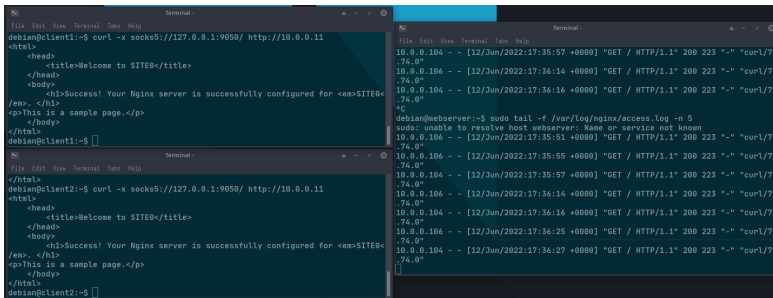


La rete Tor e' stata creata partendo dalle configurazioni suggerite dal simulatore di reti Tor private (Chutney). Al suo interno la rete e' divisa in:

- 3 DA (Relay autoritativi)
- 4 Relay di uscita

Ognuno di questi host ha un accesso diretto alla rete interna del laboratorio, per simulare il piu' possibile il fatto che presumibilmente in un contesto reale questi server dispongono di un ip pubblico. La configurazione con 3 directory authority e' conseguenza di alcuni esperimenti in cui la rete non funzionava a dovere e ho poi compreso che le reti Tor necessitano di almeno 3 host auth/mid/exit per creare le rotte. In questi 3 elementi pero' ogni host non include se stesso. Questa configurazione 3+4 e' per me la configurazione minima funzionante.

Test della rete



```
File Edit View Terminal Tabs Help
debian@client1:~$ curl -x socks5://127.0.0.1:9050/ http://10.0.0.11
<html>
  <head>
    <title>Welcome to SITE0</title>
  </head>
  <body>
    <h1>Success! Your Nginx server is successfully configured for <em>SITE0</em></h1>
  </body>
</html>
  <p>This is a sample page.</p>
</html>
debian@client1:~$
```

```
File Edit View Terminal Tabs Help
10.0.0.104 - - [12/Jun/2022:17:35:57 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
10.0.0.106 - - [12/Jun/2022:17:36:14 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
10.0.0.104 - - [12/Jun/2022:17:36:16 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
^C
debian@webserver:~$ sudo tail -f /var/log/nginx/access.log -n 5
sudo: unable to resolve host webserver: Name or service not known
10.0.0.106 - - [12/Jun/2022:17:35:51 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
10.0.0.106 - - [12/Jun/2022:17:35:55 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
10.0.0.104 - - [12/Jun/2022:17:35:57 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
10.0.0.106 - - [12/Jun/2022:17:36:14 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
10.0.0.104 - - [12/Jun/2022:17:36:16 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
10.0.0.106 - - [12/Jun/2022:17:36:25 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
10.0.0.104 - - [12/Jun/2022:17:36:27 +0000] "GET / HTTP/1.1" 200 223 "-" "curl/7.74.0"
^C
```

```
File Edit View Terminal Tabs Help
</html>
debian@client2:~$ curl -x socks5://127.0.0.1:9050/ http://10.0.0.11
<html>
  <head>
    <title>Welcome to SITE0</title>
  </head>
  <body>
    <h1>Success! Your Nginx server is successfully configured for <em>SITE0</em></h1>
  </body>
</html>
  <p>This is a sample page.</p>
</html>
debian@client2:~$
```

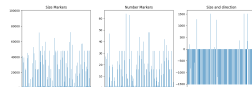
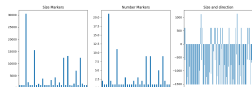
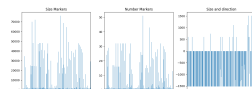
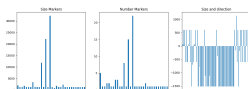
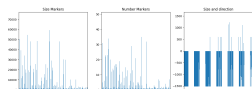
Esecuzione di un attacco

Preparazione: Analisi di traffico noto e generazione di grafici per il successivo confronto.

Esecuzione:

- Wireshark in ascolto su link client-isp
- Visita di un sito random da parte del client
- Generazione delle fingerprint (python, su macchina host, da cattura in csv)
- Confronto tra grafico random e grafici noti

Confronti 1



Confronti 2

