The Onion Router (`Tor`)

Cornel Bucurescu
Adrian Manea
Cristian Nicolae

14 martie 2019

Cuprins

CAPITOLUL 1

GENERALITĂȚI

1.1 Idei principale

„Rutarea în foi de ceapă“ (eng. `onion routing`, OR) a fost concepută pentru anonimizarea aplicațiilor care funcționează pe baza protocolului TCP, precum browsere, servicii de mesagerie sau `ssh`. În acest model de rutare, clienții aleg un drum prin rețea, construind un circuit, dar astfel încît fiecare nod cunoaște doar predecesorul și succesorul său, fără restul nodurilor care ar putea exista în rețea.

Traficul propriu-zis circulă în pachete de mărime fixată, numite *celule*, care sînt decodate cu o cheie simetrică în fiecare nod, apoi transmise mai departe.

După înființarea proiectului, a apărut destul de repede o rețea de astfel de noduri, dar chiar și la început, atunci cînd singurul nod era doar o mașină de teste, procesa conexiuni de la peste 60 000 IP-uri din întreaga lume, cu un flux de aproximativ 50 000 conexiuni pe zi.

Tor a fost conceput pentru a rezolva deficiențele găsite în modelul inițial OR, iar principalele îmbunătățiri privesc:

(1) *Secretizarea perfectă a înaintării traficului* (eng. `perfect forward secrecy`): în modelul inițial, dacă exista un nod ostil într-un punct al conexiunii, putea să refuze înaintarea traficului și să decripteze conținutul care a ajuns pînă la el. În modelul actual, se folosește o *criptare telescopică*, prin care nodul curent adaugă chei de sesiune la criptarea de pînă atunci și totodată pierde accesul la cheile din urmă, astfel încît mesajul să nu mai poată fi decriptat (nodul curent are acces doar la ultima parte a cheii, cu care contribuie).

(2) *Separarea „curățirii protocolului“ de anonimitate:* în varianta inițială, OR cerea proxy-uri pentru aplicații, corespunzătoare fiecărei aplicații suportate. Dacă proxy-urile nu

erau scrise (fiind un proces laborios), protocoalele suportau doar puține aplicații. Tor folosește o interfață proxy standard, SOCKS și proxy-uri de filtrare pentru aplicații de tip Privoxy, care sînt bine cunoscute și dezvoltate.

Privoxy[1] este un proxy care poate realiza filtrare mult mai detaliată și atentă decît un ad blocker. Este un proiect evluat dintr-un alt proxy cu efect de filtrare (`Internet Junkbuster`), căruia i s-au adus îmbunătățiri și actualizări.

SOCKS[2] este un protocol care permite schimbul de pachete printr-un server. El funcționează pe Layer 5 din modelul OSI și acceptă conexiuni pe portul TCP 1080. În varianta inițială, SOCKS nu oferea niciun fel de protecție sau autentificare, dar în ultima versiune, SOCKS5, se oferă și autentificare.

(3) *Nu influențează traficul*: OR cerea gruparea și reordonarea celulelor care ajungeau la norduri și adăugarea de padding între utilizatori și OR-uri. Această prelucrare suplimentară cerea standardizarea și găsirea unor algoritmi care să facă operațiunile sigure — algoritmi care încă nu s-au găsit. Pînă la găsirea unor implementări satisfăcătoare, Tor nu „modelează“ în niciun fel traficul.[3]

(4) *Mai multe fire TCP pot folosi același circuit*: în implementarea originală, OR cerea construirea cîte unui circuit separat pentru fiecare cerere. Dar aceasta ridica probleme de securitate, deoarece în cazul unui volum foarte mare de trafic, cheile generate pentru anonimizare și criptare puteau să se repete. Tor permite utilizarea simultană a aceluiași circuit de către mai multe fire TCP.

(5) *Implementarea unei topologii de tip „țeavă cu scurgeri“*: clientul poate forța celulele să iasă din circuit pe oriunde, nu doar pe la capete, în cazul în care observă vreo problemă sau nu dorește să continue rutarea.

(6) *Controlul blocajelor:* Tor implementează un control și management al blocajelor și aglomerărilor de tip decentralizat, prin intermediul celor care participă în rețea (devenind noduri).

(7) *Servere directoare:* În forma inițială, OR răspîndea informația prin rețea fără o anume organizare. Tor stabilește existența unor servere directoare, care sînt noduri de încredere și conțin semnături care asigură integritatea rețelei. Clienții descarcă periodic aceste semnături prin HTTP, pentru a se asigura de integritatea rețelei și a traficului.

[1] `https://www.privoxy.org/faq/general.html`

[2] `https://en.wikipedia.org/wiki/SOCKS`

[3] Afirmația este bazată pe Dingledine et al. (2004). Dar începînd cu versiunea 0.3.1.7, Tor a introdus padding între celule, care funcționează astfel: la intervale aleatorii între 1,5 și 9,5 secunde, se trimite cîte o celulă de 512 octeți, bidirecțional, între client și releu. Acest mecanism are rol de a a introduce „zgomot“ în trafic, pentru a-l face mai greu de modelat. Detalii pe Reddit și în specificațiile oficiale.

(8) *Politici standardizate:* pentru intrarea și ieșirea traficului în și din fiecare nod — un element-cheie în cazul rețelelor construite din voluntari.

(9) *Puncte de întîlnire și servicii ascunse:* în forma originală, în OR existau noduri speciale folosite pentru a construi circuite speciale către servere ascunse. Dar utilitatea lor era nulă dacă vreunul se defecta. În varianta Tor, clienții negociază „puncte de întîlnire“ (eng. `rendezvous-points`) pe parcurs, care ghidează traficul pe drum și nu se bazează doar pe „autorități speciale“.

Ca implementare, Tor nu necesită instalarea de module speciale în kernel. Dezavantajul este că nu se pot anonimiza decît protocoale TCP, dar avantajul principal este dat de portabilitate.

De asemenea, o altă caracteristică de implementare este aceea că Tor se încadrează in categoria design-urilor cu latență scăzută, deoarece este folosit pentru anonimizarea traficului în timp real.

1.2 Obiective de design și asumpții

Obiectivul principal este împiediarea atacurilor care ar viza un singur utilizator, ca țintă unică. Totodată, următoarele aspecte esențiale au contribuit la dezvoltarea Tor în forma actuală:

- Ușurința lansării în aplicații reale (eng. `deployability`): este necesar ca implementarea și utilizarea să fie cît mai simple și economice, iar cerința de anonimitate este imperioasă. Singurele noduri care nu satisfac această cerință de anonimitate sînt punctele de întîlnire, pentru asigurarea integrității traficului.

- Ușurința de utilizare: simplitatea contribuie la securitate, iar anonimatul nu trebuie să fie dependent de sistemul de operare.

- Flexibilitate: protocolul trebuie să fie suficient de flexibil și în același timp suficient de bine specificat astfel încît să poată servi drept tester pentru cercetări în domeniu.

De asemenea, s-au trasat și *non-obiective*, aspecte care sînt în mod intenționat ignorate pe parcursul dezvoltării serviciului:

- Nu servește drept conexiune peer-to-peer: abordarea are probleme serioase de securitate, chiar dacă a fost implementată de alte servicii. Problemele specifice provin din faptul că într-o conexiune peer to peer, aplicațiile servesc atît drept client, cît și drept server, astfel că pot fi atacate sau corupte în mai multe moduri, iar cîteva exemple se pot găsi aici.

- Nu este sigur împotriva atacurilor end-to-end: Tor nu pretinde să rezolve această problemă, împotriva atacurilor *end-to-end timing* sau *intersection attacks*. Problema nu este rezolvată satisfăcător în domeniu, dar ar putea să ajute ca utilizatorii să-și ruleze propriile OR.

  Pentru clarificare, adăugăm că atacul de tip *end-to-end timing* implică faptul că atacatorul poate observa traficul la ambele capete (la client și la server) și, în funcție de asta, poate descoperi tipare de utilizare și temporizare.

  Atacul de tip *intersecție* studiază comportamentul normal în rețea (trafic, flux de date, ore de încărcare, ore mai lejere etc.) și își organizează atacul încît să semene cît mai mult cu comportamentul normal. Numele atacului provine de la faptul că atacatorul trebuie să aibă un comportament care se află la intersecția între maliționizitate și normalitate.

- Nu oferă normalizarea protocoalelor: dacă se dorește securizarea informației transmise folosind protocoale complexe precum HTTP sau UDP, trebuie să se folosească servicii suplimentare; Tor nu normalizează securitatea.

- Nu este steganografic[4]— nu ascunde cine este conectat în rețea.

1.3 Modelul general al amenințărilor

Design-urile de anonimizare au drept adversar tipic unul *global*, *pasiv*, adică unul care poate asculta tot traficul.

Ca orice alt proiect pentru sisteme cu latență scăzută, Tor nu poate proteja împotriva unui asemenea adversar. Vom analiza însă, cazul adversarului care poate observa o porțiune a traficului, care poate genera, modifica, șterge sau întîrzia traficul, care ar putea să ruleze propriile OR și care ar putea compromite o parte a OR existente.

În sistemele de anonimizare cu latență scăzută care folosesc criptarea pe straturi (eng. `layered encryption`), obiectivul tipic al adversarului este să observe inițiatorul și respondentul traficului. Astfel, prin observare, ar putea confirma că Alice vorbește cu Bob dacă în trafic sînt prezente anumite variabile de flux și temporizare. Un atacator activ poate introduce asemenea variabile pentru a se asigura.

[4] *Steganografia*, o tehnică relativ recentă apărută în criptografie și securitate, înseamnă ascunderea unui mesaj sau fișier sau date sub forma unui alt tip de date. Începînd cu 2016, au apărut și metode de steganografie pentru comunicarea și protocoalele de rețea. Două exemple tipice sînt:

– `LACK` (Lost Audio Packets Steganography), care trimite pachete corupte, întîrziate sau „zgomot“ într-o conexiune de tip VoIP;

– `HICCUPS` (Hidden Communication System for Corrupted Networks), pentru WLAN, detaliat în Szczypiorski (2003).

Tor urmărește să împiedice nu atacurile de confirmare a traficului, ci pe cele de *învățare*, pentru ca adversarul să nu poată folosi informații preluate pasiv pentru a afla structura rețelei sau a routerelor. De exemplu, învățînd structura rețelei, ar putea afla poziția nodurilor de încredere (e.g. serverele directoare sau punctele de întîlnire) și ar putea destabiliza întreaga rețea atacîndu-le pe acelea.

CAPITOLUL 2

DESIGN

Rețeaua Tor este una care se adaugă peste rețeaua existentă (eng. `overlay network`). Fiecare OR funcționează ca un proces inițiat de utilizator, fără privilegii speciale. OR păstrează o conexiune TLS cu toate celelalte OR, iar utilizatorii rulează software local, numit *onion proxy* (OP) pentru a obține directoarele, a stabili circuite în rețea și a administra conexiuni cu aplicații ale utilizatorului. Aceste OP acceptă fluxuri TCP și le trimit în format multiplex prin circuit. OR din capătul celălalt conectează destinațiile cerute și pune datele la dispoziție.

Fiecare OR ține o *cheie de identitate* pe termen lung și o *cheie onion* pe termen scurt. Cheia de identitate este folosită pentru semnarea certificatelor TLS, pentru semnarea descriptorului OR (care conține un rezumat al cheilor, adreselor, lățimii de bandă, politicilor de ieșire etc.) și, prin serverele directoare, să semneze directoarele.

Cheie onion este folosită pentru a decripta cererile utilizatorilor, a stabili circuitele și a negocia cheile efemere.

Protocolul TLS stabilește o cheie de legătură pe termen scurt atunci cînd comunică între OR. Toate cheile de termen scurt sînt rotite periodic și independent.

2.1 Celule

OR comunică între ele și cu utilizatorii prin conexiuni TLS cu chei efemere. Astfel se ascund datele conexiunii cu securitate perfectă la înaintare (eng. `perfect forward secrecy`), împiedicînd orice atacator să modifice date pe drum sau să pretindă că este un OR.

Traficul între OR circulă în *celule*, fiecare avînd mărimea de 512 octeți. Celulele sînt alcătuite dintr-un cap (*header*) și o încărcătură (*payload*).

Capul conține un identificator al circuitului prin care trece celula respectivă (dat fiind că mai multe fire TLS sînt trecute prin același circuit, în format multiplex) și o comandă care specifică ce să se întîmple cu sarcina din celulă. Indetificatorul circuitului este unic pentru fiecare conexiune.

Pe baza comenzilor, celulele pot fi *de control*, care sînt mereu interpretate de nodul care le primește, sau *de transfer*, care cară informație. Comenzile din celulele de control sînt:

- `padding`, folosită pentru `TCP keepalive`[1], dar și pentru securitate, cf. §1.1.
- `create` sau `created`, pentru organizarea unui nou circuit, respectiv confirmarea reușitei;
- `destroy`, pentru eliminarea circuitului curent.

Celulele de transfer (eng. `relay`) au un cap suplimentar, care specifică fluxul de date pentru celula respectivă, apoi o sumă de control bidirecțională (eng. `end to end checksum`) pentru a asigura integritatea, lungimea sarcinii ce se va transfera și comanda de transfer. Conținutul capului și sarcinii din celula de transfer sînt criptate cu cifrul AES pe 128 biți.

Comenzile de transfer sînt:

- `relay data`, pentru datele care circulă pe acel fir;
- `relay begin`, pentru începutul transferului;
- `relay teardown`, pentru a închide un transfer stricat;
- `relay connected`, pentru a notifica reușita conexiunii;
- `relay extend` și `relay extended`, pentru a mai face un pas în conexiune și a notifica de el;
- `relay truncate` și `relay truncated`, pentru distrugerea unei părți de circuit și notificare;
- `relay sendme` pentru rezolvarea congestiilor;
- `relay drop` pentru implementarea long range dummies.[2]

[1] `http://tldp.org/HOWTO/TCP-Keepalive-HOWTO/overview.html`

[2] celule „zgomot“ care pot fi distinse de datele normale abia la ultimul pas, cf. M. Pfajfar.

2.2 Construcția unui circuit

OP-ul unui utilizator construiește un circuit din aproape în aproape, negociind cîte o cheie simetrică cu fiecare OR de pe drum.

Fie Alice utilizatorul care lansează cererea. El trimite o celulă cu comanda `create` către primul nod din drumul ales, fie el Bob. Alice alege un `circID` C_{AB} pentru această conexiune. Sarcina primei celule create conține prima jumătate a criptării *Diffie-Hellman handshake* (g^x), criptată ca fiind cheia onion a OR-ului. Bob răspunde cu celula care conține comanda `created`, care conține g^y și un hash al cheii complet negociate, $K = g^{xy}$.

Facem o mică digresiune pentru a aminti funcționarea criptării Diffie-Hellman. Detalii și alte explicații preluate de la `@tylerl` (2014). Pașii sînt următorii:

(1) Alice alege două numere prime g și p și le transmite lui Bob.

(2) Bob alege un număr secret a, pe care nu-l transmite nimănui. El calculează apoi $A = g^a \bmod p$ și transmite rezultatul lui Alice.

(3) Alice alege un număr secret b și face un calcul similar, adică $B = g^b \bmod p$ și transmite lui Bob rezultatul.

(4) Bob calculează acum $B^a \bmod p$, iar Alice calculează $A^b \bmod p$, iar amîndoi obțin același rezultat. Aceasta deoarece au loc egalitățile:

$$(g^a \bmod p)^b \bmod p = g^{ab} \bmod \text{p}$$
$$(g^b \bmod p)^a \bmod p = g^{ba} \bmod \text{p}$$

După stabilirea circuitului, Alice și Bob își pot trimite mesaje folosind cheia K.

Pentru extinderea ulterioară a circuitului, Alice trimite o celulă cu comanda `relay extend` către Bob, în care specifică adresa următorului OR pe care vrea să-l acceseze, să zicem Carol, și partea g^{x_2} a cheii pentru el. Bob își face o copie a jumătății de cheie g^{x_2} și trimite o celulă `create` către Carol. Totodată, Bob își alege un `circID` pentru legătura cu Carol, pe care Alice poate să nu-l știe; este suficient ca Bob să lege C_{AB} de C_{BC}.

Carol răspunde cu o celulă `created` către Bob și cheia g^{y_2}, iar Bob apoi trimite sarcina către Carol și răspunde cu `relay extended` către Alice. Acum circuitul este extins și legătura A–C se poate face cu cheia $K_2 = g^{x_2 y_2}$.

Procesul continuă analog pentru extinderi ulterioare.

Protocolul unilateral de handshake la nivel de circuit permite ca Alice să știe cu ce OR face legătura, dar OR-ul să nu știe — astfel se păstrează anonimitatea celulei care lansează cererea. Nu există chei publice în această legătură, iar autentificarea este unilaterală (Alice și OR negociază cheia, iar Alice știe doar că OR a învățat cheia din cele două jumătăți).

Protocolul are, totodată, secret la înaintare și noutate a cheilor.

Formal, fie $E_{PK_{Bob}}$ criptarea cu cheia lui Bob, H este funcția hash, iar | este concatenarea. Avem:

- Alice $\longrightarrow$ Bob: $E_{PK_{Bob}}(g^x)$;
- Bob $\longrightarrow$ Alice: g^y, H (K | "handshake").

În a doua parte, Bob arată că el este cel care primește g^x și alege y-ul corespunzător. În primul pas se folosește cheie publică, deoarece celula este prea mică pentru a stoca o cheie publică și semnătura.

Schema acestei conexiuni, cu tot cu celule și chei este redată în figura 2.1.

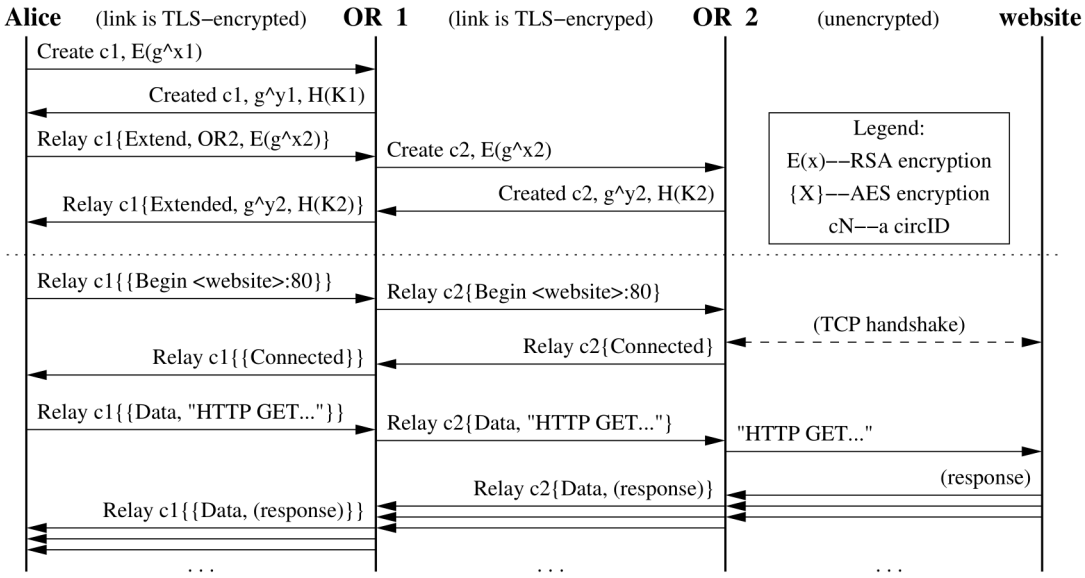

Figura 2.1: Celulele schimbate la inițierea unui circuit, cf. Dingledine et al. (2004), §4.1

Alte detalii despre protocolul Tor și specificațiile oficiale se pot găsi pe site-ul oficial.

CAPITOLUL 3

FUNCȚIONAREA, PE SCURT

3.1 Relee și „foi de ceapă“

Într-o formă simplificată, Tor funcționează prin transmiterea conexiunii printr-o serie de *relee* (eng. `relay`) de la computerul inițiator pînă la destinație.

Actualmente, există peste 6000 de relee în toată lumea, care se ocupă cu această redirecționare a traficului. Releele sînt localizate în întreaga lume și puse la dispoziție de voluntari.

Într-o conexiune standard, Tor realizează conexiunea cu 3 relee, fiecare dintre acestea avînd cîte un rol standard.

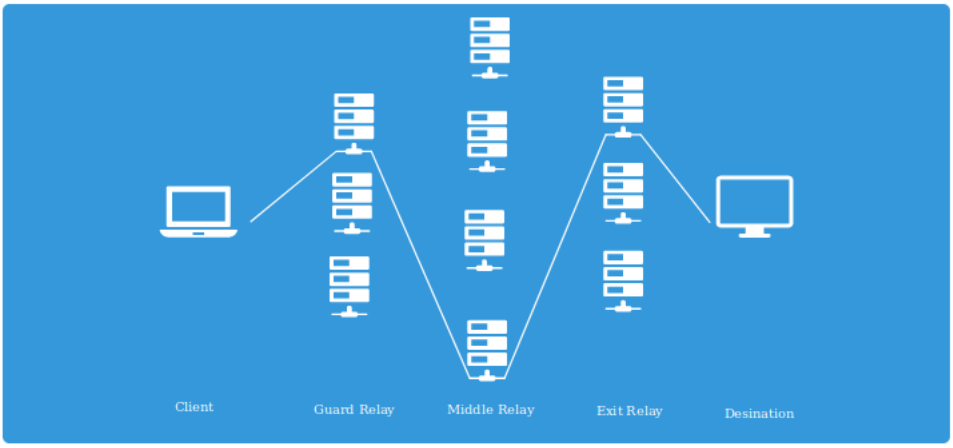

Figura 3.1: Cele 3 relee standard (Wright (2015a))

- *Releul de intrare* (eng. `entry/guard`), prin care conexiunea intră în rețeaua Tor. Asemenea relee sînt alese după ce au dovedit o vechime în rețea, stabilitate și lățime de bandă corespunzătoare.

- *Relee intermediare* sînt cele care transmit conexiunea mai departe. Totodată, în ideea anonimizării, rolul lor este ca releele de intrare și cele de ieșire să nu se cunoască între ele.

- *Releul de ieșire*, care se află la capătul rețelei Tor și trimit traficul către destinația finală dorită de client.

De remarcat este faptul că, dacă releele intermediare pot fi orice calculator, server etc., care nu se compromit în niciun fel, deoarece ele nu fac decît să transporte trafic deja criptat, releele de ieșire au o responsabilitate deosebită. În cazul unei conexiuni ilicite, traficul către destinație apare ca fiind transmis de la releul de ieșire, ceea ce-i expune în mod deosebit.

La fiecare pas se realizează o decriptare, dacă traficul circulă de la client către server și o criptare, dacă traficul circulă invers. Așa cum am menționat în secțiunea anterioară, fiecare nod intermediar adaugă sau elimină cîte un strat criptografic, realizînd „rutarea în foi de ceapă“.

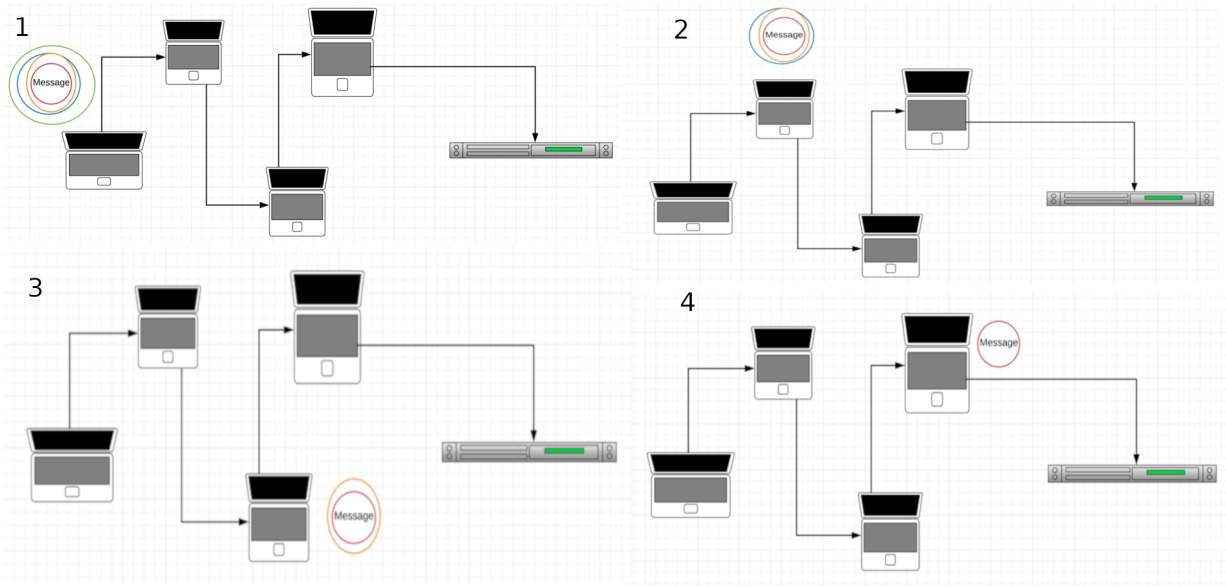

Figura 3.2: „Foile de ceapă“ (Skerritt (2018))

Prin acest mecanism, fiecare releu cunoaște doar minimul necesar: nodul anterior și nodul care va urma, realizînd *criptarea telescopică* despre care am mai vorbit. De remarcat

este faptul că releul de ieșire vede datele inițiale trimise de client. Astfel, dacă se trimit date sensibile prin protocoale care folosesc text clar, precum HTTP sau FTP, releul de ieșire poate să intercepteze traficul.

3.2 Poduri

Utilizarea releelor, în forma descrisă mai sus, ridică o vulnerabilitate serioasă. Astfel, atunci cînd un client se conectează la rețea, trebuie să aibă acces la lista tuturor releelor de intrare, mediane și de ieșire, pentru a ști unde s-ar putea conecta. Lista releelor nu este secretă, ceea ce ridică o potențială problemă de securitate.

O soluție pentru această problemă este utilizarea *podurilor* (eng. `bridges`). Într-o formă simplificată, putem privi podurile ca intrări secrete în relee, pe care le pot accesa, de exemplu, utilizatori care se află în spatele unor rețele cenzurate.

Există și o listă completă de poduri, deținută de proiectul Tor, dar această listă nu este publicată. Cei de la Tor au găsit o soluție prin care utilizatorii să aibă acces doar la o porțiune mică a listei podurilor, suficiente pentru a iniția o conexiune. Astfel, utilizatorul nici nu are nevoie de toate podurile disponibile, ci doar de cîteva pentru a-și face intrarea în rețea.

Cercetătorii au reușit să identifice între 79% și 86% din lista totală a podurilor (cf. Wright (2015a)), realizînd o analiză a întregului spațiu de adrese IPv4, dar chiar și așa, putem considera că secretul listei podurilor este suficient de bine păstrat.

3.3 Autorități directoare

Am menționat în prima parte faptul că în rețea există o serie de autorități directoare, care sînt nodurile cele mai de încredere și, într-un fel, țin funcționare proiectului în spate.

Statusul releelor Tor este ținut într-un document dinamic, numit *consens*. Acest document este întreținut de autoritățile directoare (AD) și actualizat în fiecare oră prin voturi, în următorul mod:

- fiecare AD face o listă de relee cunoscute;
- fiecare AD calculează și ceilalți parametri necesari despre relee (țara, lățimea de bandă etc.);
- AD transmite această informație sub forma unui status celorlalte AD;
- fiecare AD primește acest status, din care își actualizează propria listă;
- toți parametrii adunați de la toate AD sînt combinate și se calculează un vot, care este transmis cu o semnătură de către fiecare AD;

- releele care primesc majoritatea voturilor sînt păstrate în consens, care se actualizează și se transmite tuturor AD.

Procesul de vot și actualizare de mai sus este public, transmis prin HTTP, astfel încît poate fi accesat de orice utilizator.

3.4 Servicii ascunse

Un server poate fi configurat să funcționeze ca un serviciu ascuns. Atunci cînd se realizează această configurare, serverul trimite un mesaj către un OR ales (aleatoriu sau după o procedură specifică) pentru a cere ca acel OR să devină punct de intrare pentru serverul respectiv.

Apoi, serverul creează un descriptor pentru servicii ascunse, care este criptat cu o cheie publică și conține IP-urile fiecăror puncte de introducere care a acceptat să conexiunea. Această informație este trimisă unui tabel de hash-uri distribuit, deci fiecare OR va avea doar o parte a informației despre serviciul ascuns. Cheia pentru acest tabel este adresa onion, care este calculată de server.

Ideea de bază este că această adresă nu este publică la nivel de rețea Tor, ci este cunoscută doar de cei ce vor să o acceseze. Este vorba despre adresele de tip `[hash].onion`. Deci fiecare OR are cunoștințe minime despre aceste servicii ascunse, dar utilizatorul care vrea să se conecteze în mod explicit, o poate face.

Atunci cînd utilizatorul cere explicit conexiunea la servicii ascunse, introducînd adresa `.onion`, primește lista de IP-uri care pot servi drept noduri de intrare către adresa respectivă. Se alege una aleatoriu și se stabilește circuitul, folosind celulele de mărime fixată (pentru ca un observator să nu-și dea seama, de exemplu, că celulele mai mari corespund imaginilor sau videoclipurilor).

Pe scurt, funcționarea serviciilor ascunse urmează pașii:

(1) Calculează perechea de chei pentru criptarea asimetrică;

(2) Alege un releu ca punct de introducere;

(3) Comunică cheia publică releurilor alese;

(4) Creează un descriptor specific, care conține cheia publică și punctele de introducere;

(5) Scrie descriptorul într-o listă de hash-uri distribuită;

(6) Clientul află adresa de tip `[hash].onion`, obținută din cheia publică;

(7) Clientul se conectează la tabelul distribuit și cere să acceseze serviciul corespunzător hash-ului din adresa `.onion`;

(8) Dacă serviciul există, clientul află cheia publică a serviciului și punctele de introducere;

(9) Clientul alege un punct de introducere la întîmplare și-i comunică un cod de tip one time. Releul ales pentru introducere joacă rol de punct de întîlnire;

(10) Clientul creează o celulă de introducere, care conține adresa punctului de întîlnire de accesat și codul one time, pe care le criptează cu cheia publică a serviciului ascuns;

(11) Clientul trimite mesajul de mai sus în rețeaua Tor către punctul de introducere ales, cerînd să se facă legătura cu serviciul ascuns;

(12) Serviciul ascuns decriptează mesajul de introducere cu cheia privată pentru a obține informațiile trimise;

(13) De îndată ce serviciul ascuns descoperă punctul de întîlnire ales, îl transmite clientului ca fiind acceptat;

(14) Clientul și serviciul ascuns comunică folosind acest punct de întîlnire, avînd traficul criptat la ambele capete. Atît clientul, cît și serverul ascuns își creează propriul circuit pînă la punctul de întîlnire, fiecare circuit avînd cel puțin 3 noduri, deci o asemenea conexiune are cel puțin 6 noduri.

CAPITOLUL 4

ATACURI ȘI STRATEGII DE APĂRARE

În lucrarea inițială de prezentare a tehnologiei Tor (Dingledine et al. (2004)), apar diverse tipuri de atacuri pe care cercetătorii le prevăzuseră, împreună cu modurile de apărare.

4.1 Atacuri pasive

Un atac pasiv clasic este reprezentat de *observarea tiparelor din traficul utilizatorilor*. Astfel, deși identitatea utilizatorilor rămîne necunoscută, se poate profila traficul și găsi diverse tipare în modul lor de utilizare. Totuși, sarcina este îngreunată de faptul că pot circula mai multe fire prin același circuit, așa cum am menționat în prima parte a lucrării, deci conexiunile interceptate pot să nu aparțină utilizatorului-țintă.

Un alt tip de atac tipic este dat de *observarea conținutului utilizatorului*. Deși la nivelul utilizatorului, conținutul este criptat, conexiunile către serverele care trimit răspunsul, pot să nu fie (de exemplu, conexiunea de la releul final la serverul adresat). Se poate chiar ca serverul accesat să fie ostil și să încerce să atace utilizatorul. Tor nu urmărește să filtreze conținutul, dar se poate apăra împotriva acestui atac folosind servicii terțe precum Privoxy sau altele similare pentru filtrare.

Faptul că este permis fiecărui utilizator să-și configureze conexiunea poate fi o vulnerabilitate. De exemplu, utilizatorii mai avansați sau cei care ar avea nevoie de anonimitate sau intimitate mai pronunțată, pot cere schimbarea circuitului foarte des. Dar aceasta poate fi o vulnerabilitate pentru utilizatorii „normali“, deoarece îi evidențiază ca pe o posibilă minoritate.

Așa cum am mai menționat, un alt atac clasic poate fi *corelația la ambele capete*. Un observator global ar putea să identifice și să profileze mai bine traficul, dar în realitate,

acest atac este aproape imposibil. Mai mult decît atît, cum Tor folosește topologia țevii cu scurgeri, se poate întîmpla ca un utilizator să ceară ieșirea din rețea înainte de capăt, iar atunci observatorul, chiar cu puteri globale, este compromis.

Atacurile de mai sus se încadrează în categoria *atacurilor de confirmare*. Prin observarea traficului, se poate confirma că s-a realizat o anumită conexiune de interes. Dar există și un alt atac cu potențial mai periculos, de care Tor nu poate apăra foarte eficient: *amprentarea site-urilor* (eng. `website fingerprinting`). În locul căutării conexiunilor de intrare sau ieșire pentru profilare, atacatorul poate să-și facă o bază de date de „amprente" care conțin tipare de acces și mărimi de fișiere schimbate de client cu site-uri țintă. În cazul Tor, eficiența acestui atac nu este foarte mare, deoarece se folosesc mai multe fire în format multiplex prin aceeași rețea și mai mult, amprentele sînt limitate de mărimea celulelor de trafic, adică de 512 octeți. Sugestii pentru îmbunătățirea securității din acest punct de vedere ar fi să se utilizeze scheme de grupare a traficului către anumite site-uri sau padding aplicat link-urilor.

4.2 Atacuri active

Un atacator care află cheia de sesiune TLS ar putea să controleze celulele și releele din circuit. Totodată, aflarea acestei chei îi permite să desfacă un strat din criptare și, dacă află cheia unui OR, poate să „devină" acel OR pe durata vieții cheii. Însă atacul nu este foarte util, deoarece, pentru a putea crea într-adevăr impresia că este acel OR, trebuie să aibă acces și la cheia onion pentru a decripta celulele `create`. Dar, cum conexiunea are securitate perfectă la înaintare, de îndată ce s-a făcut o legătură în rețea, ea nu mai poate fi compromisă. Totodată, rotația cheilor limitează perioada în care asemenea atacuri pot acționa. Astfel că, în realitate, un asemenea atac poate fi eficient doar dacă se poate lansa pe durata vieții unei conexiuni. Situația este puțin probabilă, dar pot exista cazuri legale sau extralegale cînd autorități de un anume rang pot cere accesul la anumite noduri. A fost cazul în 2003, cînd autoritățile germane au obținut un mandat de spargere a protocoalelor serviciului de anonimizare `Java Anon Proxy` (Wikipedia (2003)).

În plus, dacă atacatorul află cheia de identitate a unui nod, poate contacta serverele directoare și să trimită descrieri false, infiltrîndu-se în aceste servere.

Un alt tip de atac activ care poate fi lansat este prin rularea unui server. Dacă un atacator creează un server web care să ofere conținut prin care să atragă utilizatori ai rețelei Tor, atunci el are șanse să intre în rețea și să fie accesat de noduri de ieșire. În acest caz, poate afla diverse tipare de trafic sau poate trimite conținut malițios, deoarece are control asupra unui capăt al conexiunii. Similar este și cazul cînd un atacator deține un OR ostil. Dar, dacă atacatorul deține controlul asupra $m > 1$ din N noduri, se poate arăta că el poate corela cel mult $\left(\frac{m}{n}\right)^2$ din trafic, conform Skerritt (2018). Acest număr, deși poate fi neglijabil (amintim că $N > 6000$), poate fi grupat cu alte atacuri, precum acela al

rulării unui server care să atragă trafic.

În ce privește atacurile asupra celulelor, amintim că la fiecare nod se fac verificări de integritate, deci acestea nu pot fi atacate cu succes.

În cazul în care utilizatorul vrea să se conecteze la un server printr-un protocol nesecurizat, precum HTTP, se poate ca un atacator să se dea drept serverul-țintă. De aceea, utilizatorii se pot proteja de o asemenea situație conectîndu-se cu protocoale care fac autentificarea la ambele capete.

Desigur, alte tipuri de atacuri care țin de faptul că rețeaua este menținută activă de voluntari pot fi distribuirea de cod ostil din partea unui utilizator de încredere sau acțiuni interzise din punctul de vedere al politicilor de ieșire sau de rutare ori din punct de vedere social.

4.3 Atacuri asupra directoarelor

În cazul în care se distrug servere directoare, rețeaua este proiectată pentru a continua să funcționeze, deoarece încă se primesc semnături și voturi în consens din partea serverelor încă active. Însă, din design, dacă se distrug peste jumătate din serverele directoare, este necesară intervenția umană, deoarece consensul nu mai are suficiente date.

Atacul asupra unei singure autorități directoare, însă, nu este tocmai eficient, deoarece consensul înregistrează voturi, iar acest atac ar putea, în cel mai rău caz (dar total nepractic) să influențeze aceste voturi, înclinînd balanța către autorizarea unor servere ostile. În practică, asemenea atacuri nu au apărut, deci nu există precedențe pentru a se face analize de impact.

Și în aceste cazuri, pot apărea atacuri datorate naturii umane. De exemplu, se poate păcăli o autoritate directoare că un OR funcționează, deși este stricat sau că este valid, deși este malițios sau convingerea pentru listarea unui server drept autoritate directoare. Tor nu apără împotriva unor asemenea atacuri.

4.4 Cenzură și atacuri politice

Un atac special, care poate fi lansat de un atacator ostil sau comandat politic (sub formă de cenzură, de exemplu), ar putea să vizeze traficul care intră sau care iese din Tor, adică să împiedice utilizatorii să intre în rețea sau să iasă către destinație.

Blocarea nodurilor de ieșire este o situație care poate avea loc, deoarece lista nodurilor de ieșire este publică, iar Tor nu se poate apăra de asemenea atacuri. Un atacator poate pur și simplu să dezactiveze nodul de ieșire sau să-l inunde cu cereri, lucru care l-ar face inutilizabil.

În celălalt caz, însă, blocarea nodurilor de intrare este imposibilă, mulțumită podurilor. Aceasta deoarece lista tuturor nodurilor de intrare nu este disponibilă, iar lista

podurilor poate fi descoperită doar parțial. Se poate, cel mult, combina acest tip de atac cu cel de corelarea traficului. Dacă atacatorul are date acumulate despre anumiți utilizatori, despre care se crede că se conectează la anumite noduri de intrare (sau acest lucru este surprins în timp ce se întîmplă), se pot lansa atacuri asupra acelui nod. Atacurile de tip „inundație“ (eng. `flood`) sînt cele mai ușoare și ar putea dezactiva nodul de intrare.

CAPITOLUL 5

TOR VS VPN

VPN (eng. `Virtual Private Network`) este o altă opțiune pentru anonimizarea traficului online, care funcționează într-o manieră aparent similară rețelei Tor. De aceea, poate apărea întrebarea privitoare la diferența între cele două și care este de preferat, în funcție de scopul și aplicațiile urmărite.

De aceea, vom descrie pe scurt această comparație, evidențiind cazurile tipice cînd un VPN este de preferat sau cînd Tor este de preferat, luînd în discuție și avantajele și dezavantajele fiecăreia dintre aceste două soluții.

Deoarece tehnologia Tor a fost descrisă pe larg în secțiunile anterioare, ne oprim acum atenția în special asupra VPN.

Funcționarea de bază a unui VPN este centrată pe o rețea de servere, de obicei plasate în țări diferite, iar utilizatorul care vrea să se conecteze la un serviciu web prin intermediul unui VPN va folosi unul dintre serverele puse la dispoziție drept intermediar. Astfel că site-ul sau serviciul-țintă va vedea traficul ca venind dinspre serverul intermediar, iar nu din partea clientului, ceea ce realizează un anonimizare a traficului. De asemenea, în cadrul acestui schimb, mesajele transmise de la client la serverul VPN și apoi de la serverul VPN și serviciul-țintă, precum și înapoi, sînt criptate.

Principalele avantaje și dezavantaje ale serviciului VPN provin din aceeași sursă: faptul că serviciul este deținut de o firmă privată, iar mesajul trece exclusiv prin serverul pe care îl pun ei la dispoziție.

Avantajele care derivă de aici se leagă de eficiență și viteză. Deoarece o firmă privată pune la dispoziție serverele, se va asigura că instalarea serviciului este foarte simplă, că funcționează pentru majoritatea dispozitivelor și sistemelor de operare și că viteza este cît se poate de mare.

Dar tot de aici apar și principalele dezavantaje. Mai întîi, este un serviciu comercial, în majoritatea cazurilor și, dacă utilizatorul alege să plătească puțin sau deloc, poate avea

probleme în cel puțin 3 privințe:

- Mai întîi, este posibil ca serviciul să folosească o criptare slabă. Există servicii VPN care folosesc metode de criptare standard și foarte sigure, precum AES pe 256 biți, dar există și servicii care folosesc o criptare slabă.

- Calitatea software-ului folosit pentru intermedierea conexiunii poate fi îndoielnică, mai ales în cazul VPN-urilor ieftine. Cum conexiunea trece obligatoriu prin programul oferit de firma care deține serverele, defecțiuni de software au efect asupra oricărei conexiuni pe care clientul o lansează.

- În funcție de țara sau țările în care se află serverele VPN, pot exista reglementări legale diferite, precum impunerea unei căi de acces forțat (`backdoor`) pentru organe legale sau păstrarea înregistrărilor activității (`logs`)[1]. Tot la acest capitol putem menționa și politica de achiziție. Un VPN respectabil, deoarece este menit să fie folosit pentru anonimizare, acceptă ca utilizatorul să devină client cu date personale minime. Au existat chiar cazuri cînd servicii de VPN au vîndut datele utilizatorilor lor.[2] De cealaltă parte, serviciul `MULLVAD` din Suedia, de exemplu, permite ca un client să trimită bani în numerar, în plic, fără datele expeditorului pentru a-și plăti abonamentul. Va primi, apoi, un cod unic de înregistrare și poate începe să folosească serverele lor.[3]

Acum, revenind la comparația cu Tor, putem vedea că există anumite avantaje de partea acestui serviciu. Întîi de toate, este gratuit și nu este deținut de vreo autoritate sau firmă privată. Nodurile intermediare care sînt folosite drept relee pentru o conexiune aparțin voluntarilor și nu pot fi puse în pericol de o politică privată. Fiind mai multe noduri, descentralizate și aparținînd unor persoane fizice, serviciul este greu de atacat, în ansamblul său. În plus, codul utilizat este complet disponibil (`open source`), caz mai rar întîlnit în cazul VPN-urilor.

În plus, rutarea „în foi de ceapă“ a serviciului Tor oferă criptare suplimentară și o anonimizare mai bună, așa cum am descris în secțiunile anterioare.

De aici provin, totodată, și dezavantajele Tor. Fiind rulat de voluntari, pot exista diverse probleme politice sau sociale și, în general, compatibilitatea cu diverse dispozitive poate fi problematică. De exemplu, actualmente, Tor nu poate fi folosit pe dispozitive iOS, iar pentru Android este în faza beta.

Mai mult, așa cum am descris în secțiunile anterioare, Tor nu a fost făcut pentru a permite conexiuni și transferuri *peer-to-peer*, în vreme ce un VPN poate fi folosit și pentru

[1] Diverse politici corecte sau mincinoase de păstrare a log-urilor sînt discutate aici

[2] Cazuri în care servicii de VPN au inclus explicit sau implicit în politica de confidențialitate faptul că datele utilizatorilor pot fi folosite în scop comercial sînt prezentate aici

[3] Resurse suplimentare privitoare la jurisdicția serviciilor VPN, politici de confidențialitate și, în general, soluții optime pentru criptare și anonimizare sînt colectate la PrivacyTools.io

așa ceva. La fel este și cazul transferurilor mari de date, precum vizionarea unui film sau descărcarea unei cantități mari de informații.

Pe scurt, deci: Tor poate fi folosit cu succes atunci cînd traficul de date nu este foarte mare și cînd se dorește o conexiune anonimă folosind o soluție a comunității. VPN-ul se pretează cazurilor în care avem trafic mare de date, viteza este foarte importantă, iar anonimitatea perfectă nu este prioritatea numărul unu. În plus, un serviciu VPN bun (cu criptare bună, fără politici de păstrare a log-urilor, fără preluarea datelor pentru înregistrare, cu viteză bună, uptime bun, relații cu clienții de calitate etc.) poate fi costisitor.

Desigur, soluția optimă este utilizarea unui VPN *și* a Tor, de exemplu accesînd serverele VPN înainte de nodul de intrare în Tor sau după nodul de ieșire.

Mai menționăm la acest capitol comparativ și serviciile `proxy`. Ele funcționează pe un principiu similar, adică direcționarea traficului către o instanță intermediară înainte de accesarea serverului de destinație. Totuși, serviciile proxy pot avea o criptare deficitară (SOCKS și HTTP nu oferă niciun fel de criptare, iar HTTPS oferă criptare la același nivel cu orice website folosind SSL). În plus, serviciile proxy au fost gîndite pentru cîte o aplicație particulară, de obicei un browser web. Astfel, trebuie configurate diferit și individual pentru e-mail, chat sau alte aplicații terțe.

INDEX

BIBLIOGRAFIE

Dingledine, R., Mathewson, N., and Sylverson, P. (2004). Tor: The second generation onion router. *SSYM Proceedings of the 13th conference on USENIX Security Symbposium*, 13.

Skerritt, B. (2018). How Tor *really* works. https://hackernoon.com/how-does-tor-really-work-c3242844e11f. [Online; accesat martie 2019].

Szczypiorski, K. (2003). Steganography in TCP/IP Networks. http://www.tele.pw.edu.pl/~krzysiek/pdf/steg-seminar-2003.pdf. [Online; accesat martie 2019].

`@tylerl` (2014). Diffie-Hellman in plain English. https://security.stackexchange.com/questions/45963/diffie-hellman-key-exchange-in-plain-english. [Online; accesat martie 2019].

Tor (2019). The Tor Project. https://www.torproject.org/. [Online; accesat martie 2019].

VA (2019). The Best VPN. https://thebestvpn.com. [Online; accesat martie 2019].

Wikipedia (2003). Java Anon Proxy. https://en.wikipedia.org/wiki/Java_Anon_Proxy. [Online; accesat martie 2019].

Wright, J. (2015a). How Tor Works: Part One. https://jordan-wright.com/blog/2015/02/28/how-tor-works-part-one/. [Online; accesat martie 2019].

Wright, J. (2015b). How Tor Works: Part Three. https://jordan-wright.com/blog/2015/05/14/how-tor-works-part-three-the-consensus/. [Online; accesat martie 2019].

Wright, J. (2015c). How Tor Works: Part Two. https://jordan-wright.com/blog/2015/05/09/how-tor-works-part-two-relays-vs-bridges/. [Online; accesat martie 2019].