The Onion Router (Tor)

Cornel Bucurescu
Adrian Manea
Cristian Nicolae

3 martie 2019

Cuprins

DE ADĂUGAT SAU CLARIFICAT

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.

citări pentru SOCKS și Privoxy

(3) *Nu influențează traficul*: OR cerea gruparea și reordonarea celulelor care ajungeau la norduri și adăugarea de padding între utilizatori și OR-uri.

padding?

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.

(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.

(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 împiedicarea 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.

  probleme la p2p — exemple

- 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.

  detalii despre aceste atacuri!

- 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 — nu ascunde cine este conectat în rețea.

  interesant cuvînt... detalii?

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.

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.

Traciul între OR circulă în *celule*, fiecare avînd mărimea de 512 bytes. 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 `keepalive`;
- `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.

clarificat asta ultima

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, $K = g^{xy}$.

cîte ceva despre Diffie-Hellman

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, care 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 cricuit 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: $g^y, H\,(K \mid \text{“handshake”})$.

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.

Mai departe

Detalii despre componente și exemplul de bază de la Jordan Wright + desenele de la Skerritt (2018)

CAPITOLUL 3

ATACURI

- din whitepaper §7;
- cenzură și ”guvern ostil” (v. Jordan Wright) (Wright (2015c));

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].

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].