

How Tor works: introduzione all'onion routing

10 Maggio 2008

Marco Bonetti
marco.bonetti@slackware.it

Infosecurity 2008 - Firenze



Questo lavoro è distribuito sotto
*Creative Commons Attribution-Share Alike 2.5
Italy License.*

Per maggiori informazioni visita:

<http://creativecommons.org/licenses/by-sa/2.5/it/>

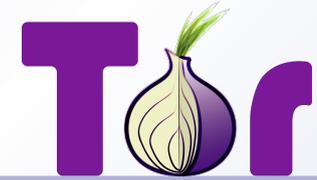
Tor: The Second-Generation Onion Router

Cronologia



- Anni '80: David Chaum teorizza e implementa le “mix networks”, catene di proxy server
- Anni '90: lo United States Naval Research Laboratory si interessa alla materia e sviluppa la tecnologia dell'onion routing
 - Onion Routing briefing slides, 1996
 - "Hiding Routing Information," Information Hiding, R. Anderson (editor), Springer-Verlag LNCS 1174, 1996, pp. 137-150
- Oggi: “Tor: The Second-Generation Onion Router”, Venerdì 13 Agosto 2004 @ 13th USENIX Security Symposium

Cosa è Tor?



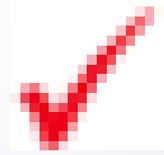
- Uno strumento per persone e organizzazioni che vogliono migliorare la loro sicurezza in internet
- Un programma per anonimizzare la navigazione, la pubblicazione di contenuti, lo scambio di messaggi, IRC, SSH e altre applicazioni che usano il protocollo TCP
- Una piattaforma per sviluppare nuovi programmi dotati di caratteristiche di anonimità, sicurezza e privacy
- Uno strumento per proteggersi dall'analisi del traffico

Analisi del traffico



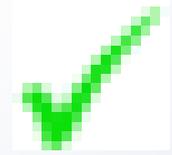
- La raccolta di dati riguardanti le comunicazioni permette di ricostruire il profilo degli interessi e dei gusti dei partecipanti
- Dimmi dove vai e ti dirò chi sei ;-)
- L'impiego di protocolli insicuri (smtp, vnc, telnet) lascia filtrare troppe informazioni
- Esempi di analisi del traffico:
 - Un sito di e-commerce può applicare prezzi differenti a seconda del paese di origine del visitatore
 - Controllare la posta dall'estero permette di scoprire da dove si proviene o chi si è

Una prima soluzione

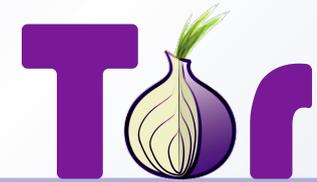


- I pacchetti che viaggiano in Internet sono composti da:
 - Header, contiene le informazioni di instradamento
 - Payload, contiene i dati
- Se riesco a criptare il payload nessuno può “leggere” il contenuto della mia sessione
- È vero, ma questa tecnica da sola non è sufficiente a proteggermi: l'header contiene ancora troppe informazioni
- Allora cripto anche l'header!
- Provateci... ;-)

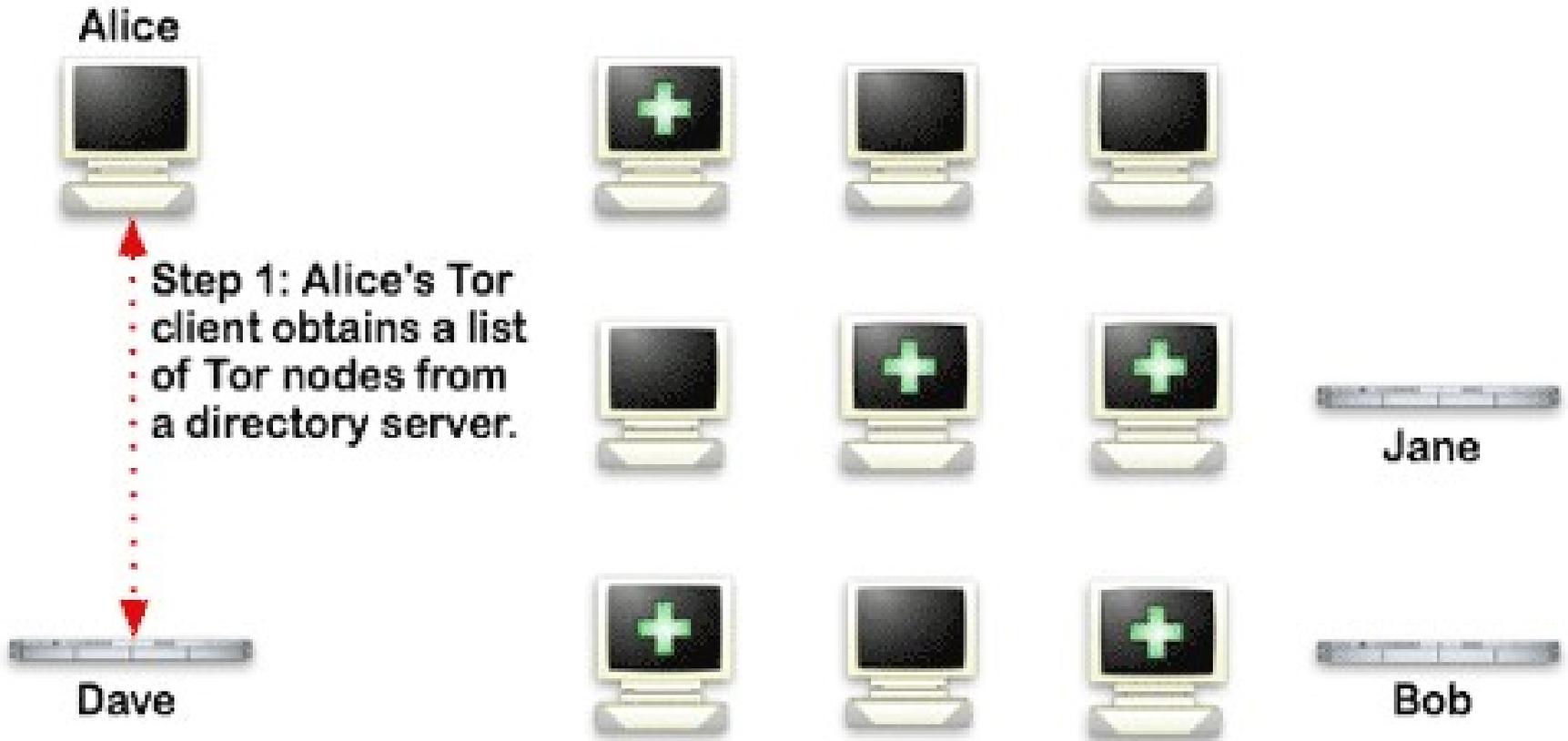
La soluzione proposta da Tor

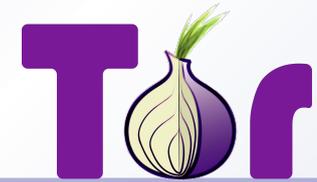


- Creiamo una rete di nodi parallela a Internet per l'instradamento dei pacchetti
- La rete di Tor funziona come una scatola nera (black box): i pacchetti che vi entrano scompaiono e appaiono “auto magicamente” all'uscita, dopo aver percorso un viaggio all'interno della rete parallela
- L'idea è quella di raggiungere la destinazione cancellando le tracce che ci lasciamo dietro, in modo da rendere impossibile l'analisi del traffico
- Come accade la magia?



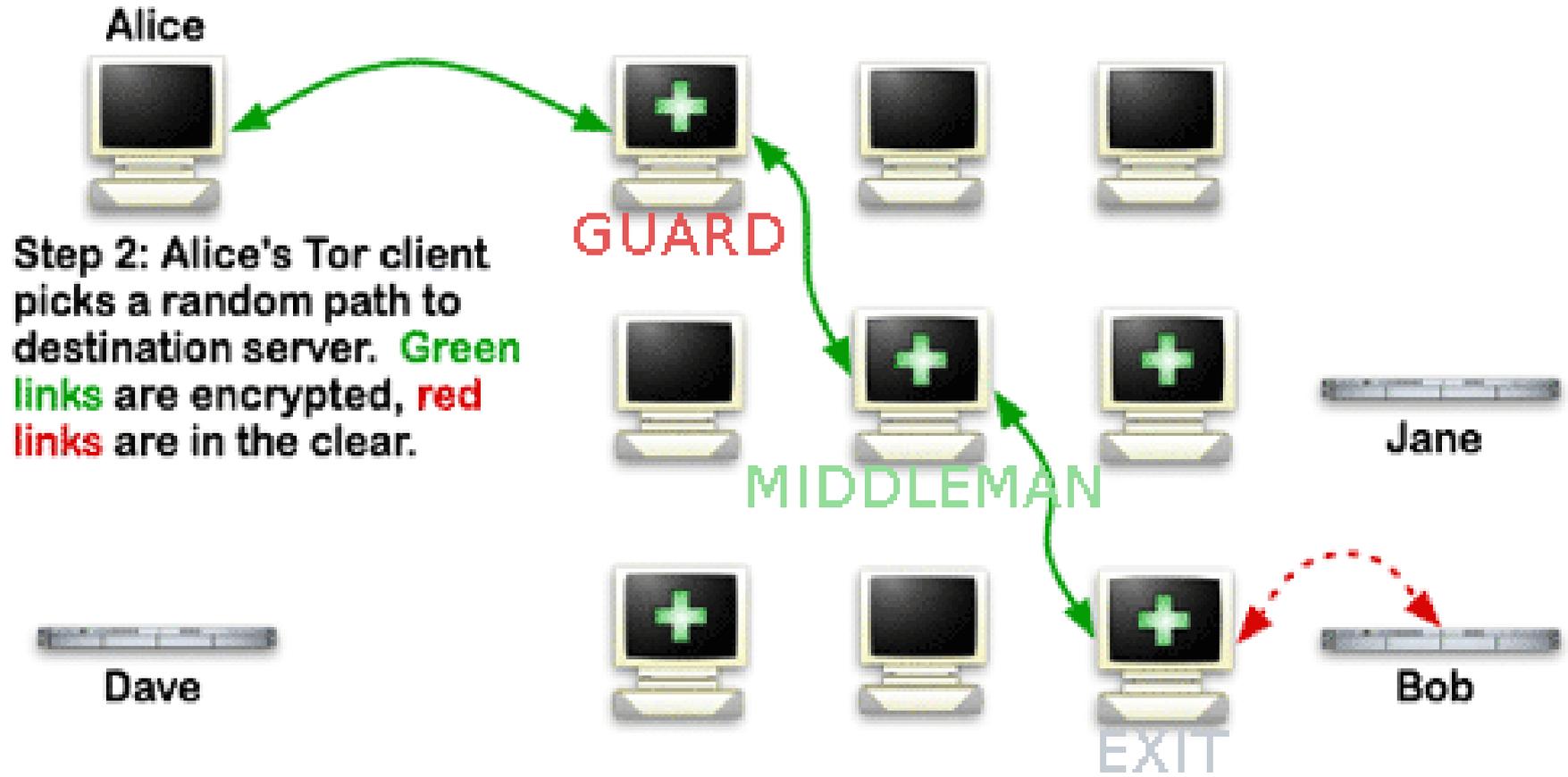
How Tor Works: 1



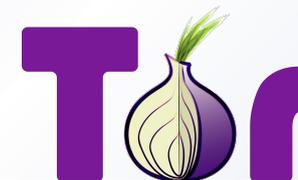


How Tor Works: 2

- Tor node
- unencrypted link
- encrypted link



Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.



How Tor Works: 3

- Tor node
- unencrypted link
- encrypted link

Alice



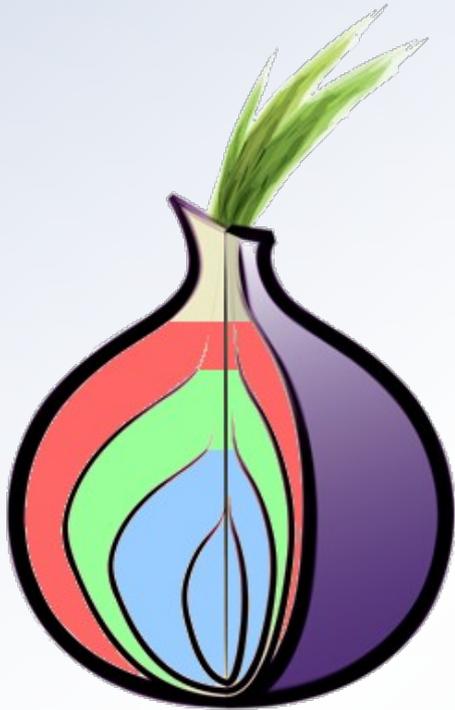
Step 3: If the user wants access to another site, Alice's Tor client selects a second random path. Again, **green links** are encrypted, **red links** are in the clear.

Dave

Jane

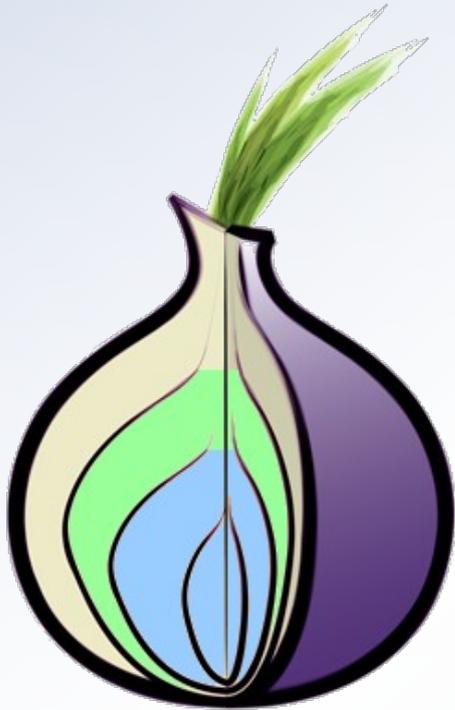
Bob

Creazione di un circuito - 1



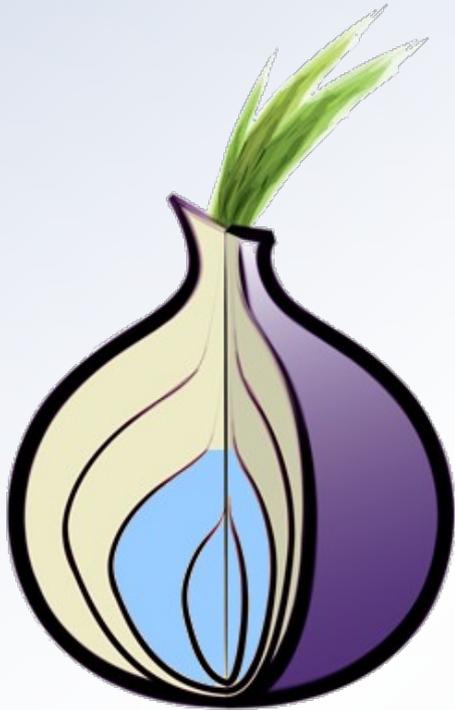
- Il client invia al nodo di guardia (GUARD) il pacchetto completo
- Il nodo di guardia decrittta il primo strato e individua il nodo di transito a cui inviare il rimanente payload

Creazione di un circuito - 2

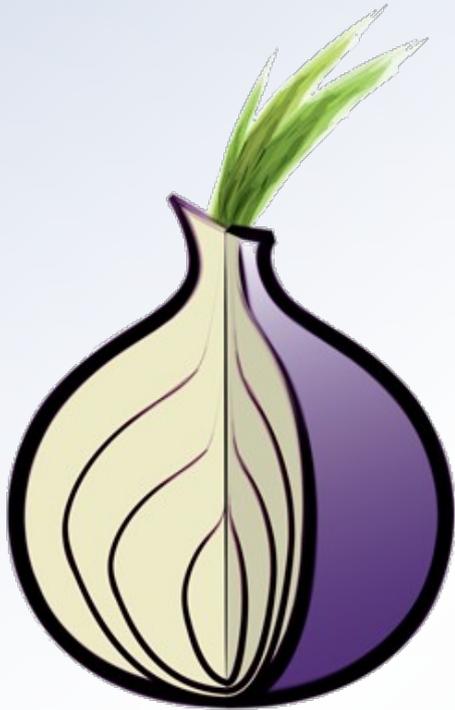


- Il nodo di transito (MIDDLEMAN) riceve dal guardiano il payload ridotto
- Come nel caso precedente, decrittato lo strato di sua competenza per conoscere quale sarà il prossimo nodo a cui inviare il resto del payload

Creazione di un circuito - 3



- Il nodo di uscita (EXIT) riceve le istruzioni finali per la creazione del circuito
- Decrittando le informazioni ricevute, il nodo individua la macchina da contattare e la specifica richiesta da inviare

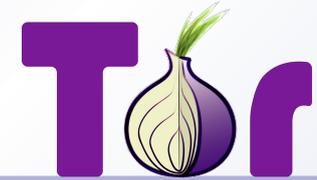


- Il circuito è completo!
- Con le informazioni ottenute al passaggio precedente, il nodo di uscita si collega alla macchina finale e chiede le informazioni volute dal client di partenza
- Una volta ottenuta una risposta provvederà ad inoltrarla all'indietro, utilizzando il circuito appena stabilito

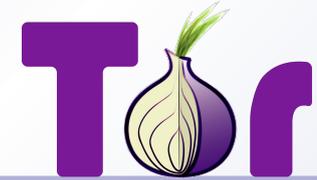
Cipolle!



- Avete capito perché si chiama “router a cipolla”?
- One-hop routing: ogni nodo conosce solo che un pacchetto gli arriva dal nodo a monte e devo consegnarlo al nodo a valle
- I nodi intermedi non possono leggere il contenuto del payload di destinazione
- In questo modo riusciamo a fuggire dalle tecniche di analisi del traffico in quanto non è possibile risalire agli attori del dialogo senza riuscire a leggere TUTTO il traffico che viaggia all'interno della rete di Tor e, anche in questo malaugurato caso, non si avrebbe la certezza matematica dell'individuazione dei partecipanti ma solo una approssimazione.

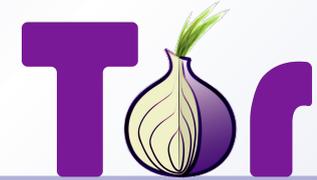


- Perché limitarsi a oscurare le comunicazioni?
- Nascondere i servizi!
- Un server Tor è in grado di pubblicare informazioni riguardanti particolari servizi (sito web, server IM) offerti esclusivamente ad altri utenti Tor
- Questi servizi (gli “hidden service”) non sono visibili dall'esterno ma solo dalla rete torificata



- Tor è free software rilasciato sotto la 3-clause BSD e liberamente scaricabile all'indirizzo <https://www.torproject.org/download.html.en>
- Il client ascolta su **localhost** sulle seguenti porte:
 - porta 9050 per il proxy SOCKS v. 4/4A/5
 - porta 9051 per la control port (opzionale)
- Non c'è differenza tra il programma client e quello server, solo che il secondo caso deve essere esplicitamente configurato dall'utente
- Il server ascolta **all'esterno** su diverse porte:
 - porta 9001 (443) per la creazione di circuiti
 - porta 9030 (80) per fornire servizio directory (opzionale)
 - porta 9040 per eseguire transparent proxying (opzionale)

Configurare un Tor server



- La rete Tor funziona solamente grazie alla buona volontà degli utilizzatori che decidono di impiegare la propria macchina anche come server
- Se si hanno almeno 20KB di banda in upload e download è consigliabile settare un server Tor
- Le istruzioni si trovano all'indirizzo <http://www.torproject.org/docs/tor-doc-relay>
- Si può scegliere quale porte permettere in uscita dalla propria macchina
- Per chi non ha un abbonamento flat è possibile selezionare le finestre orarie di utilizzo della banda oppure una quota di banda totale mensile
- Per non incidere troppo sulle performance della rete locale si possono settare i picchi di utilizzo

Tor in pratica



- La navigazione via web è semplice da anonimizzare: basta selezionare 127.0.0.1:9050 come socks proxy per il proprio browser
- L'utilizzo della versione 4a rispetto alla 4 o alla 5 è importante per intercettare le richieste DNS
- Tor viene distribuito accoppiato con Privoxy e TorButton
 - Privoxy è un proxy http/https, da usare in cascata prima di Tor, che esegue information stripping delle richieste del browser in modo da aumentare la confidenza della propria sessione e diminuire le informazioni raccolte dal server finale
 - TorButton è un plugin per Firefox che si preoccupa di impostare automaticamente il browser per l'utilizzo di Tor ed esegue anche una serie di controlli aggiuntivi volti a limitare al massimo il filtraggio di informazioni



- I maggiori protocolli di instant messaging forniscono la possibilità di utilizzare proxy http e/o socks per la comunicazione, basta utilizzarli attraverso Tor o Privoxy
- Un buon compagno di Tor+IM è l'utilizzo del plugin OTR <http://www.cypherpunks.ca/otr/>
- Purtroppo non è così semplice utilizzare Tor+IRC, la maggior parte dei server blocca l'accesso via proxy dei client per motivi di ordine pubblico
- Freenode fornisce ben due hidden server per la propria rete: <irc://mejokbp2brhw4omd.onion/> per l'accesso autenticato via GPG e <irc://5t7o4shdbhotfuzp.onion/> per l'accesso libero (down durante i periodi di abuso)



- Non ha molto senso torificare il flusso di informazioni di un programma P2P, effetto leech.
- Il protocollo più flessibile è BitTorrent
 - Il metodo più comune è torificare le informazioni scambiate con il tracker e lasciare in chiaro le connessioni ai peer, sia bittorrent (l'originale) che Azureus e gli altri client supportano l'impiego di socks e http proxy
 - Il secondo metodo è impiegare una rete di filesharing completamente torificata, con il tracker come hidden service, funziona ma impatta negativamente sulle performance globali della rete

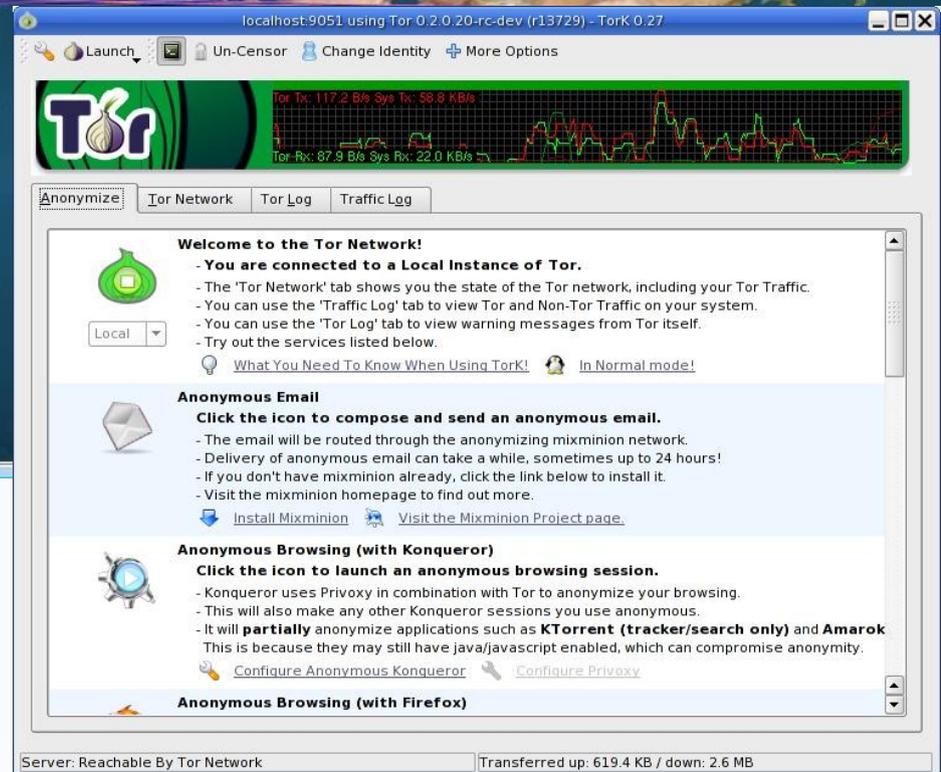
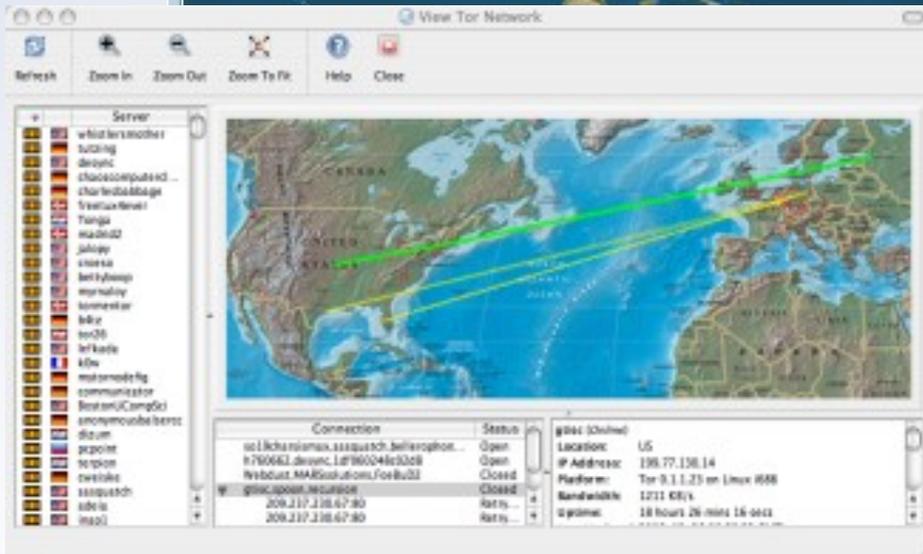
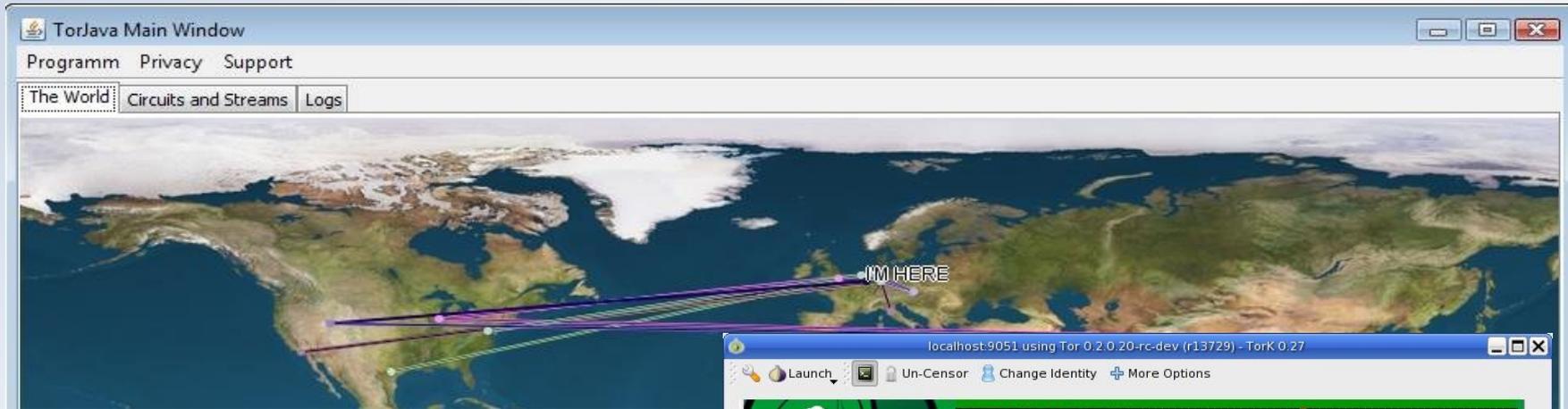


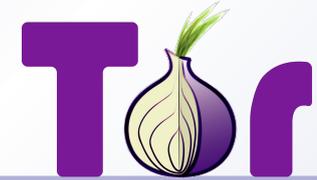
- Il protocollo ssh prevede l'utilizzo di programmi proxy, con Tor e OpenSSH è utile impiegare un proxy command come Connect:

<http://www.meadowy.org/~gotosh/projects/connect>

- Non tutti i programmi supportano nativamente l'utilizzo di proxy: tsocks (linux e *bsd) permette di wrappare le chiamate di sistema alla funzione connect() in modo da instradarla attraverso un socks proxy, è un metodo brutale ma funziona
 - `tsocks nc $IP $PORT`
 - `tsocks links http://www.google.com/`
- Per tutti gli altri: <http://shellscripts.org/project/toraliases/>

Il futuro





- Ringraziamenti in ordine sparso:
 - A Roger Dingledine, Nick Mathewson, Peter Palfrader, tutti gli altri sviluppatori di Tor e la EFF per portare avanti un tale progetto
 - A Ren Bucholz per le immagini di “How Tor works”
 - A Mike Perry per il suo lavoro su Tor e TorButton
- Domande?

