Tor

Valerio Tanferna, Alessandro Martinelli

October 25, 2016

Contents

Abstract

This document ...

1 Introduction

1.1 Traffic Analysis

Before we start talking of Tor, let's see the motivations that led to its development.

As you probably know, when packets travel the internet, even if their payload may be encrypted, their header is not. Thus, an entity that have access to the link on which a packet is traveling may acquire some information about it, e.g. the source IP address and the destination IP address, therefore finding out who is getting in touch with whom.

Is there a problem with this? yes: aside from the obvious privacy concern, traffic analysis is problematic also for other reasons, e.g:

- ISP may block traffic to set destinations (e.g. in totalitarian regime).
- In some country, journalists may not be able to safely communicate with whistleblowers and dissidents.
- If you're travelling abroad and you connect to your employer's computers to check or send mail, you can inadvertently reveal your national origin and professional affiliation to anyone observing the network, and you may not want this.

More examples are avaible here. Let's see what Tor is and how it try to address these problems.

1.2 The Onion Router

Technopedia.com give us the following definition:
The Onion Router (TOR) is an open-source software program that allows users to protect their privacy and security against a common form of Internet surveillance known as traffic analysis.

TOR is based on OR, a project developed in the 1990 by the U.S. Navy in an effort to protect government communications.

In 2002 Paul Syverson, a member of the original OR project, together with computer scientists Roger Dingledine and Nick Mathewson, gave birth to Tor.

In 2006 the Tor Project was founded, with the financial support of the Electronic Frontier Foundation and other organizations (more information here).

TOR brings a lot of improvement with respect to OR, like

- Directory servers
- Configurable exit policies
- A pratical design for location-hidden services via rendevouz points
- Perfect forward secrecy
- Congestion control
- Integrity checking

In this paper, only the first three of them will be covered. More information can be found here.

2 How Does Tor Work?

2.1 Onion Routing: a first look

The main idea behind TOR is the following: starting from a standard scenario like in figure 1, where an user U is trying to contact an user S, we move to a

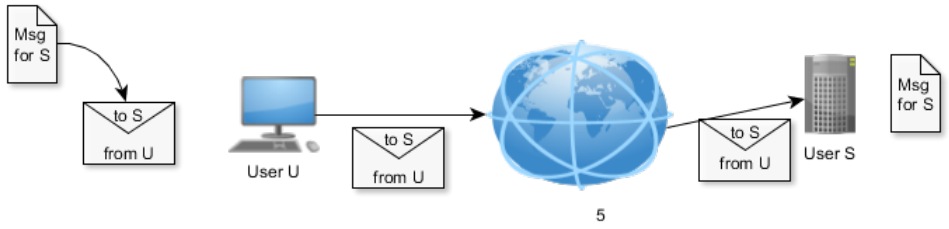


Figure 1: A standard scenario

scenario like the one in figure 2 on the next page, where

- the packet is not directly routed from U to S but it first travels to user A, then to user B, then to user C, and only lastly to user S, so that observing e.g. the link between U and the Wide Area Network won't let an observer know who has been contacted by U.
- For achieving the packet to travel the desided path, the user have to incapsulate the original payload several time, first with a header containing C as source and S as destination, then the first header and the original

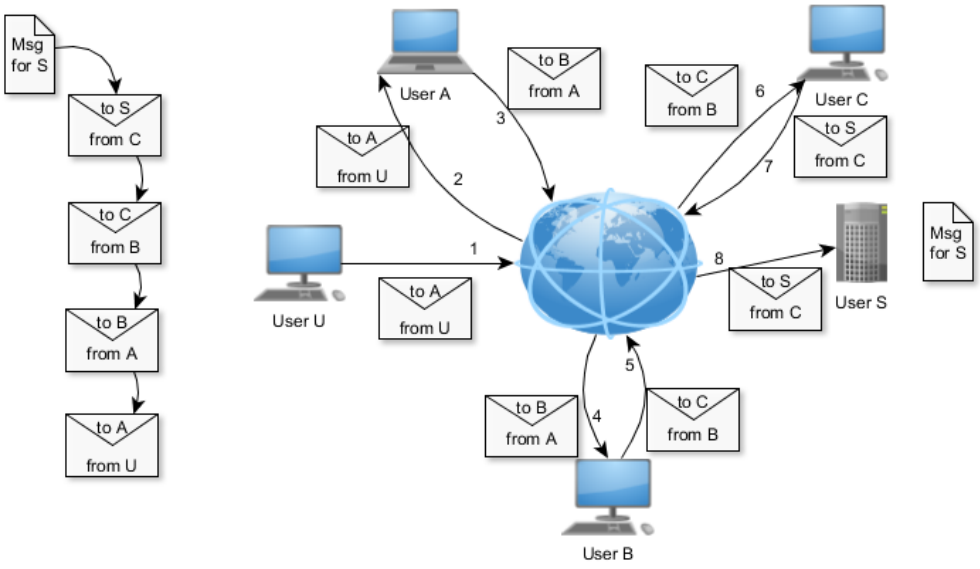


Figure 2: The packet passed through three intermediate nodes

payload are in turn incapsulated into an header containing B as source and C as destination and so on as described in figure 2 and have packets from S travel somehow the same path.

- The previous mechanisms alone wouldn't be enought for granting anonimity, since if an intermediate user, e.g. A, wants to perform some data analysis, he could successfully perform it by reading all the header contained in the packet, thus discovering that the original source is U and the intended destination is S. Therefore the solution used in TOR, shown in figure 3 on the next page, is the following: the original node encrypts the packet that will cover the last step of the path by means of the key shared with the last node, then he encrypts in turn this packet by means of the key shared with the penultimate node and so on. Lastly, a last header is added, containing the original node as source and the first node of the path as destination. This way each intermediate node can only discover the previous and the following step in the path, and both intermediate nodes and final nodes have no way to discover both the real source and the real destination of the packet. The Onion Router protocol is named after this way of encapsulating packets. We will see later how keys are negotiated.

Of course, if U put some personal information in the payload, S will know who is him.

Notice that, in this configuration, the packet traveling the last link is not

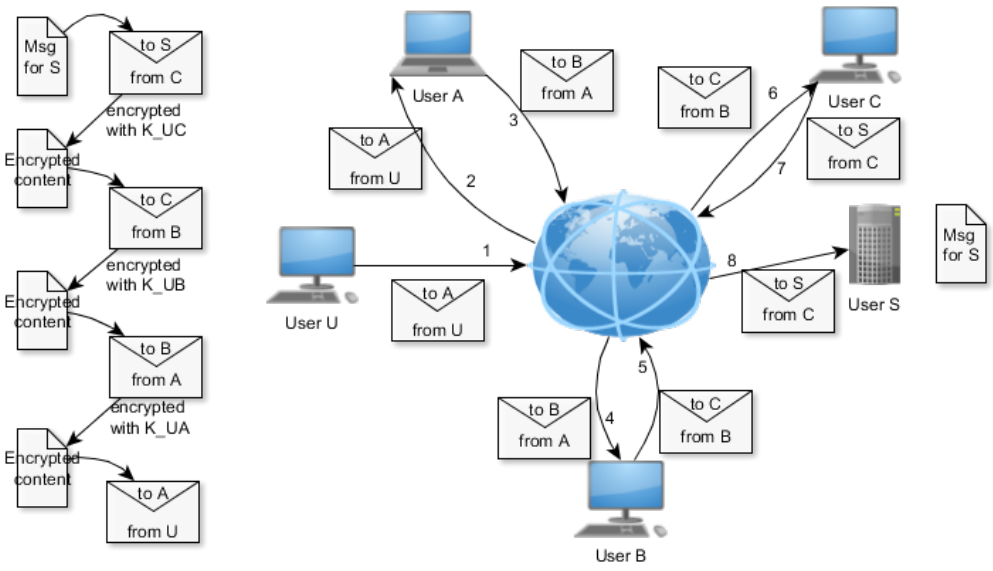


Figure 3: The packet passed through three intermediate nodes. Headers are encrypted

encrypted, since the payload was not originally encrypted. As a consequence, in this case we can achieve anonimity but not confidentiality. We will go back on this in section 2.4.3

This is the appropriate moment to introduce some terminology: the nodes (running the Tor software) who make themselves available for forwarding Tor packets are called *relay*; a subset of them, who make themself available to forward Tor packets also to the final destination, are called *exit relay*. Why is there such a distinction? Because if the packet forwarded to destination causes some trouble (e.g. it contains prohibited communications), the one who gets in troble usually is the exit-relay. Therefore, this role involve more risks. We will come back on this in section 2.4.2 for discovering how Tor address this problem.

2.2 A more detailed example

Before data can be sent through the prearranged path, a *circuit* has to be built. Figure 4 on the following page shows how this can be done (the example use two intermediate nodes, but there would not be much difference with three). Traffic passes along these connections in fixed-size cells. Each cell is 512 bytes, and consists of a header and a payload. In the header a command field is present; based on their command, cells are either *control cells*, which are always interpreted by the node that receives them, or *relay cells*, which carry end-to-end stream data.

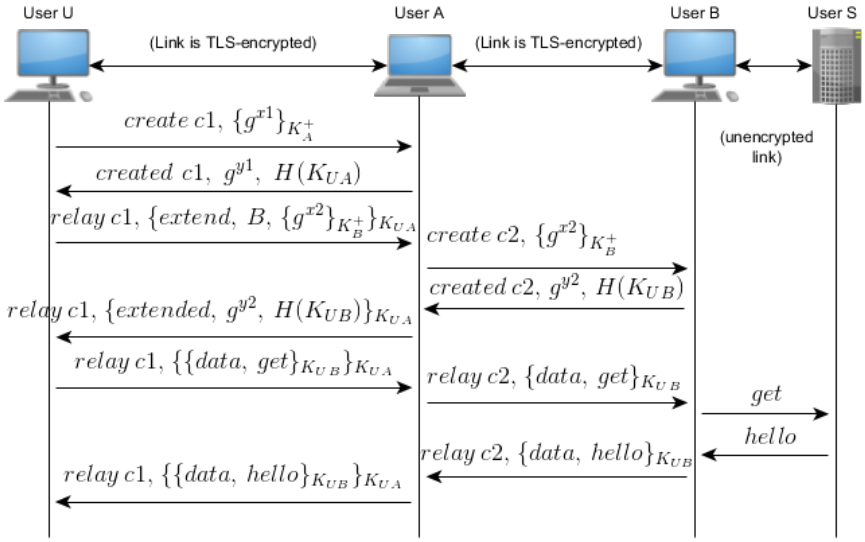


Figure 4: How to build a circuit

In the example, U asks A to create the first hop of the circuit, called *c1*, and exchange with him his half of the Diffie Hellman key. The key is encrypted by means of the public key of A, so that U can authenticate A. A doesn't authenticate U, thus this is an *unilateral authentication*. This first message is intended to be interpreted by the node that received it, thus the first is a control cell, with *create* as command.

Then A answer saying the circuit has been created, communicating his half of Diffie Hellman key and the hash of the just calculated session key. This one is a control cell too, with *created* as command.

The third message is contained in a relay cell, with *relay extend* as command. With this message, U ask A to extend the circuit to B, using the passed parameters (such as the half of Diffie Hellman key generated by U). This message is encrypted by means of the symmetric key previously negotiated between U and A.

Messages exchanged between A and B are similar to the ones used for establishing circuit c1, then B will confirm to A that the circuit has been extended. After these, A forward to U the second half of the Diffie Hellman key of B and the hash calculated by B on the session key between U and B.

Now a path is ready, and U can start communicating anonymously with S. The first message would be a *relay begin* command, but we've omitted it for brevity. U will exchange messages with S using *relay data* commands. Packets will be encrypted in the onion-like way showed in figure 3 on the previous page.

2.3 The cell header format

Since we've mentioned cells, it's worthwhile to show the cell format: The *Cir-*

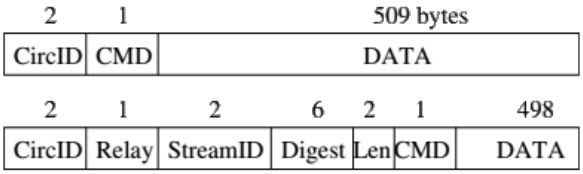


Figure 5: Control and relay cell format

cId field contains a circuit identifier that specifies which circuit the cell refers to (many circuits can be multiplexed over the single TLS connection). *CMD* describe what to do with the cells payload. On one hand, *CMD* in control cells may assume value such as:

- *create, created* to set up a new circuit
- *destroy* to tear down a circuit

On the other hand, *CMD* in relay cells may assume value such as:

- *relay data* for data flowing down the stream
- *relay begin, relay end* to open and close a stream
- *relay extend, relay extended* to extend the circuit by a hop, and to acknowledge
- *relay truncate, relay truncated* to tear down only part of the circuit, and to acknowledge

A relay cell's header contains some more field, like a Stream Identifier, but since we will not examine what a stream is in tor, there would be no point in explaining the meaning of such a field here. *Digest* is used as end-to-end checksum for integrity checking (more info here)

2.4 Some further details

2.4.1 Tor Proxy

Until now, we've not ansewered yet the following question: who is actually in charge of establishing circuits across the network, exchanging the symmetric key etc? The answer is the Tor Proxy Software, illustrated in figure 6 on the following page. For a correct use of Tor, the browser must have all the connection to pass through the tor proxy, who is in charge of making the connections use a circuit it has established.

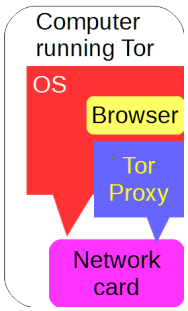


Figure 6: The figure shows at which level the Tor Proxy operates

An attacker may try to exploit this mechanism forcing the user browser to bypass the Tor Proxy, so the user IP address is revealed. This can be done e.g. if the attacker succeeds in having the user running a script choosen by the attacker. A possibile solution is to use the NoScript extension. We will come back on this.

Note that Onion Routing originally required a separate application proxy for each supported application protocol - most of which were never written, so many applications were never supported.

2.4.2 Directory Servers

Let's answer another question: when the tor proxy must build the circuit, who tells him which nodes he can use as relay or exit-relay? The answer is the Directory Servers. As of now (2016), there are nine of them across the world. Tor client software is pre-loaded with a list of the directory servers and their keys. Each such Directory Server acts as an HTTP server, so clients can fetch current network state and routers list, and so other ORs (Onion Routers, another name for relays) can upload state information.

The Directory Servers try to agree upon a list of reliable relays, called *consensus directory*. This list will be signed by every Directory Server who trust it. When a node connects to the Tor network, he asks a directory server for the consensus directory, and will trust the received list only if it is signed by more than half of the DSs. Therefore, an attacker who wants to have a Tor node connecting to his own (probably comprimised) relay, must first succeed in compromising more than half Directory Servers.

Another task of Directory Servers is to mantain some information, e.g. the public key of the Tor relays. In addition, they mantain an information useful to solve the exit-relay problem previously explained: an user may want to make himself available as exit-relay, but he may want to avoid trouble with the law. What he can do is specify a set of destination he wants (or not wants) to forward traffic to, by means of an *exit policy*. A destination may be specified e.g. in terms of IP address, port number etc. A Tor node, when choosing a path, will avoid exit-relay that are unavaible to forward the kind of traffic he wants to

send. More information here.

Of course, the Directory Servers represent also a single point of failure: since they mantain the list of the nodes partecipating to the Tor protocol, it is not infrequent that, under totalitarian regimes, this list is used for blocking all the nodes partecipating to Tor and thus blocking Tor itself. We will come back on this.

2.4.3 Link Encryption

The last question - are all the connections along the path encrypted? - has been already answered: no, the last step is outside the Tor network, so its up to the website whether to implement a secure connection or not, as shown in figure 4 on page 6. A possible solution is to use the *HTTPS everywhere* extension, which ask websites to use https when it is available, and refuse connection if it is not available. Here you can find an interactive pages that show the benefits of using Tor alone, HTTPS alone, or the benefits of the two of them combined together.

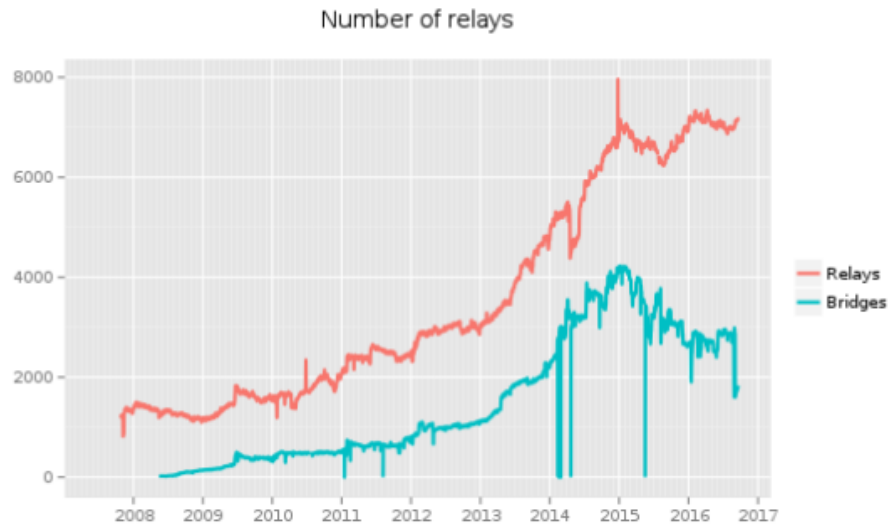


Figure 7: The following graph shows the number of running relays and bridges in the network. Credit: metrics.torproject.org

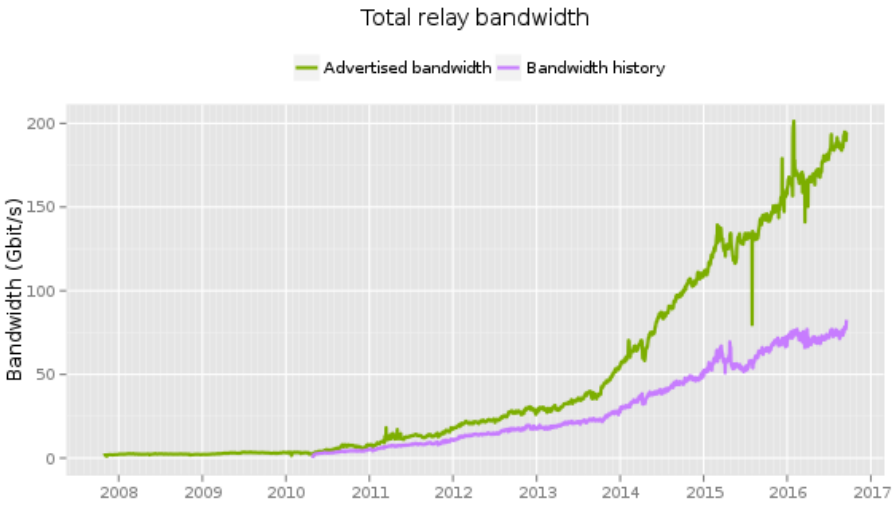


Figure 8: The following graph shows the total advertised and consumed bandwidth of all relays in the network. Credit: metrics.torproject.org

3 Hidden Service

Hidden Services are web sites or other service such forum, blog, buyers portal and so on, that are located in the Tor net. These services aren't accessible via the clear web (e.g the standard way of surfing web) since Tor hides the real location and the real IP of Hidden Services. At the same time an Hidden Service (*HS* from now on), when receives packets from an user, doesn't know the real source of that traffic. Therefore, this mechanism guarantees anonimity for both parts.

Anyone can create and host an Hidden Service.

3.1 Creation of a Hidden Service

For being reachble, a Hidden Service must advertise its existence in the Tor network. Let's see how this is done.

First of all, the server that is willing to host an HS generates a long-term secret, composed by a public key, private key pair.

Then it selects two or more nodes of the Tor network. These nodes, called *Introduction Points* (IPs), will have the task of letting the clients connect to the Hidden Service.

As third step, the HS server connects to the previously selected IPs and send them its public key. As usual, connections happen using a three-step circuit, so that it's difficult for anyone to associate an Introduction Point to the Hidden Service server's IP address.

Finally the HS creates a descriptor that contains its public key and the list of

the Introduction Points, it signs it by means of its private key and it uploads the signed descriptor on the proper Directory Server - the descriptors are stored in a distributed hash table, so first it has to understand which the correct Directory Server is.

The Directory Server now generates a 16-character sequence derived from the Hidden Service's public key. This sequence is called *onion address* of the HS, is used to locate the HS's descriptor inside the distributed hash table, and is advertised e.g. on a webpage.

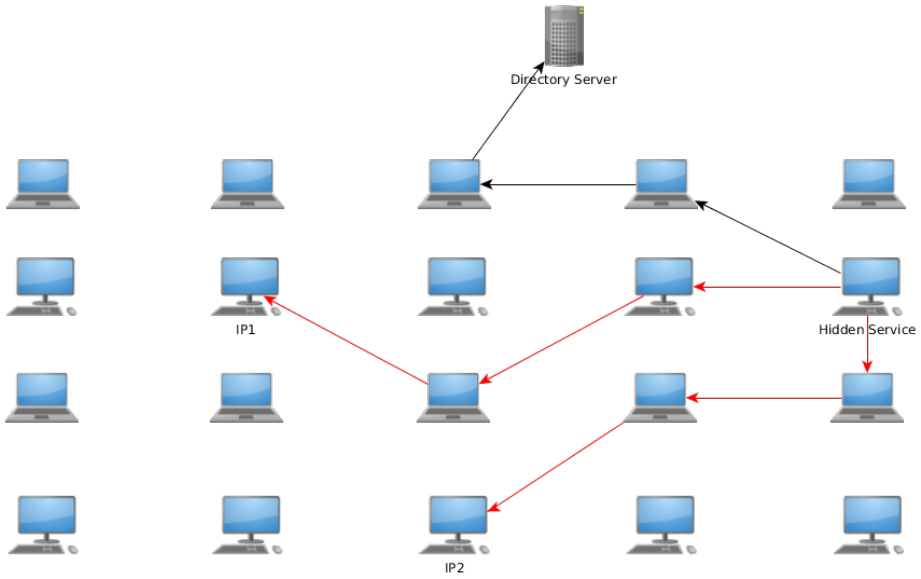


Figure 9: The Hidden Service server contacts the Introduction Point (red) and the Directory Server(black)

3.2 Connection to the hidden service

Let's see which step a client has to perform in order to connect to an HS (assume it has already downloaded the HS's onion address from a website).

Firstly, it has to connect to the proper Directory Server in order to download the HS'descriptor, by means of that it can retrieve the set of Introduction Points and the Public Key of the HS it is trying to connect to.

As second step it picks up at random a Tor relay and ask it to work as Rendezvous Point. It may or may not accept the task. Once a Tor relay accepting the task is found, the client sends him a one-time secret.

Then it opens an anonymous stream to one of the HS's Introduction Points, Then client creates a message that contains the address of the Rendezvous Point, a one-time secret and its first half of the DH key, encrypts the message with

the Hidden Service's public key, and send it to the selected Introduction Point, requesting the delivery of the messaage to the Hidden Service.

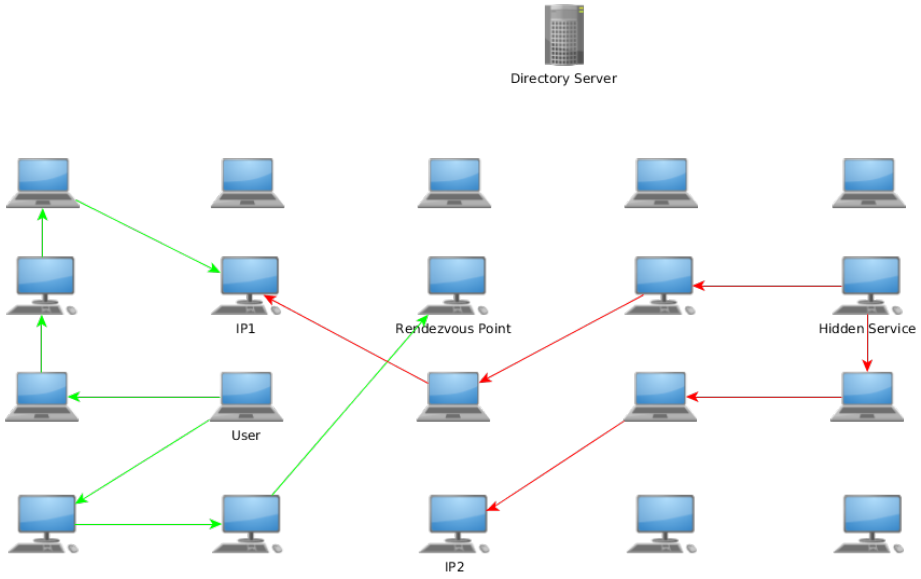


Figure 10: A client choose a Rendezvous Point and then communicate with one of the Hidden Service's Introduction Points (green lines)

The connection between the Hidden Service and the Introduction Point, client and Rendezvous Point, client and Introduction Point, and lastly connections to Directory Server, take place via a Tor circuit. This way no one is able relate sender and receiver of the messages.

3.3 Establishing of a connection

The Hidden Service receives a message from a client, decrypts it and find the address of Rendezvous Point and the one-time secret and the client's half of DH simmetric key. The HS creates a circuit to client's Rendezvous Point and send to it the one-time secret just read, his half of the DH key and a hash of the session key they now share.

If all went the right way, the RP connects client's circuit to HS's, therefore the connection between client and Hiddens Service now is composed of 6 relays. The Introduction Points are not used, so that the Rendezvous Point know nothing about them.

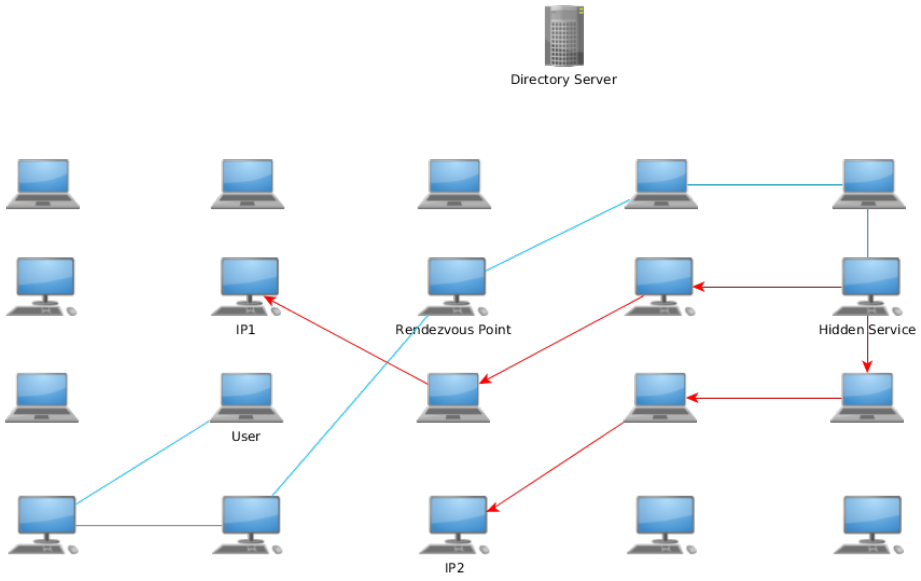


Figure 11: A path between client and Hidden Service has been built (light blue)

3.4 How to access the Dark Web by means of the Tor network?

Now that we've explained what An Hidden Service is, we may introduce some definitions: we call

- *Surface Web* anything that a search engine can access;
- *Deep Web* anything that a search engine cannot access;
- *Dark Web* a small portion of the Deep Web that has been intentionally hidden and is inaccessible through standard web browsers.

The Hidden Services are part of the dark web. For accessing the dark web, an user can choose among these three methods (we will come back on these method in section 6)

- Tor Proxy with standard browser (deprecated).
- Tor2web (deprecated).
- Tor Browser Bundle.

Main pages of the deep web are The Hidden Wiki (via Tor 2 web or direct link). In this page there is all hidden services of Tor and many search services for Tor network, such ahmia.

4 Vulnerabilities of Tor

Now that we've seen how Tor works, we are able to understand the vulnerabilities of this type of communication. Let's see a list of possible attack vs Tor.

As mentioned in section 2.4.1, an attacker may try to exploit the mechanism used by Tor forcing the user browser to bypass the Tor Proxy, so the user IP address is revealed. This can happed e.g. having the user browser running some javascript code (*javascript injection*) or a malicious plug-in, so that the code can bypass the sandbox of execution of the browser and send out the real IP of the user.

Wired report that with the malware, FBI controlled many server in the Tor network.

According to eff.org, the largest known hacking operation in U.S. law enforcement history occurred during December 2014. FBI has arrested and charged hundreds of suspects that were using an Hidden Service on Tor called *Playpen* for child pornography hosting.

The technique used by FBI are based on sending a malware to visitors of the site, exploiting a vulnerability in Firefox bundled in the Tor browser. The malware used is called NIT (Network Investigative Technique) that copied certain identifying information from a user's computer and send it back to the FBI.

In similar manner Silk Road 2.0 was closed. Silk Road has been closed by an infiltrated of FBI that has become administrator.

Unfortunatley this technique can be used against journalists and dissidents.

Is there a way of protecting ourself from this kind of attacks?

4.1 Router with Tor

Referring to figure 12 on the following page, an attacker can exploit a bug only in the user machine, bypass the Tor proxy and send a packet with the real IP of the user (in the example Italy, 93.34.75.148). If instead the tor proxy is in execution on the router, link in figure 13, when the attacker exploit a bug and send the packets with the clear IP of user, the packet pass through the router, on which the Tor proxy is running. The router forwards all packets to the Tor network independently of the destination IP. This way the server see even the fake IP (Switzerland, 137.93.194.09).

This mechanism can be obtained using a laptop configured to work as router, or buying a special purpose hardware, like safeplug.

The problem is not always user-side, but also server-side, for example default configuration on the apache server may potentially unmask the real identity of hidden service.

4.2 Tor with others software

When an user connect to Tor, the Tor proxy force all connections of the browser through the Tor network, but doesn't do the same with other softwares that may

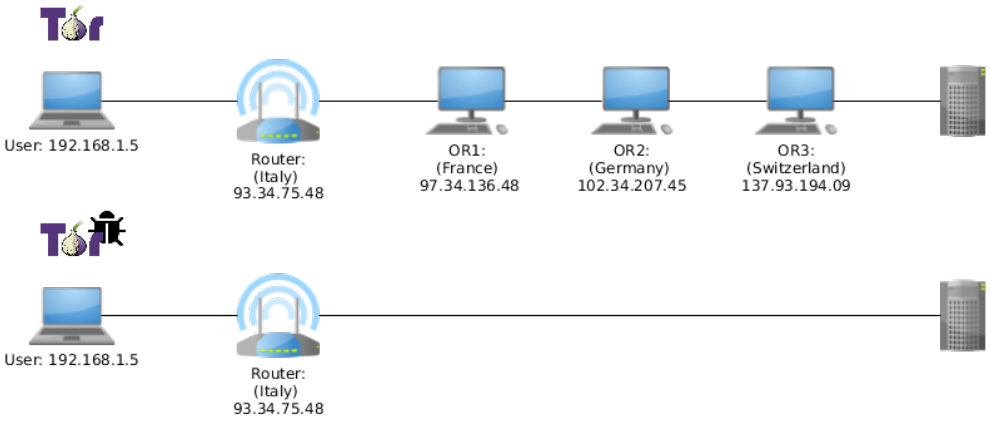


Figure 12: Tor proxy on user PC

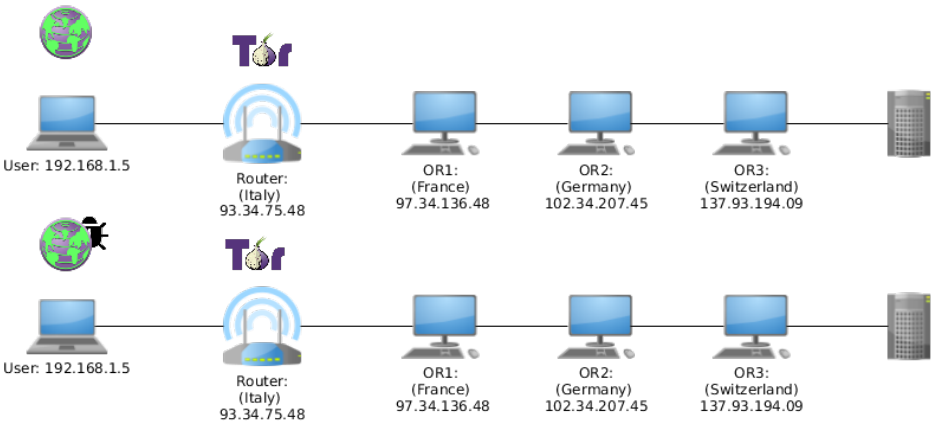


Figure 13: Tor proxy on router

access to the internet.

It is necessary configure other softwares for using the Tor proxy; Tor community advise against using a non configured software while using Tor.

Furthermore, the Tor community advise against the use of the other software when running Tor browser, since otherwise it would be possible for an attacker to perform a special statistical attack that analyze the entry node and the exit node.

4.3 Tor and censorship

As previously seen that the Directory Server in Tor have a very important function: keep trace of all node (named relay) of the tor net. When an user wants connect to Tor must connect one of this server for download the list of relay and create a circuit for become anonymous.

Tor Browser by default makes all of its users to avoid fingerprinting attack, but this doesn't hide that you're connecting to Tor. Directory Server are dislocated around the world, but what happens if this server are unreachable from the user? Tor don't work.

Some totalitarian regime have a intranet and block all connection to outside, simply getting the list of relays and blocking their IP address. Tor provides a solution to this problem: **Tor Bridge Relay**.

These relays are an *alternative entry point* to the Tor network. They're preconfigured in Tor browser and, in order to avoiding censorship problems, are not publicly known. When you start Tor browser, it asks you if you are in a totalitarian regime or if you want help users who live under censorship. If the user agree to help these person, Tor work as Bridge Relay, if the user is under censorship Tor use a Bridge Relay for connect to Tor network.

But the ISP can still block Tor, recognize that the packet is a Tor packet with Deep Packet Inspection (DPI). According to Tor's documentation, *Over the last few years, censors have found ways to block Tor even when clients are using bridges. They usually do this by installing special boxes at ISPs that peek into network traffic and detect Tor; when Tor is detected they block the traffic flow.* For bypass such sophisticated censorship Tor use a **pluggable transports**. These transport manipulate all Tor traffic by *obfuscation* between the client and the first hop of cicuit such that it is not identifiable as a Tor connection. They take advantage of various transport and make encrypted traffic to Tor look like not-interesting or garbage traffic.
This aren't the finally solution: pluggable transports are not immune to detection but this require a lot of time to identify it. Periodically Tor community change the obfuscation algorithms.
The user can add more bridge relay or pluggable transports.

4.4 Are Directory Servers and relays secure?

Tor uses 3 relay for guaranteeing the anonimity of the user, this way any relay of the circuit doesn't know both sender and receiver. An attacker must control more than half of the Directory servers for discover the route of the packets. The Tor community use more technical for guarantee the security and the reilability of the relay, example "consensus directory", but aren't sufficiently. In according to thehackernews in the july 26, 2016, over 100 Tor nodes found designed to spy on deep web users.
We can make an analogy with torrent community: is based on the user, anything avoid that a malicius user used it for insert a malicius torrent, in the same mode anything avoid that malicius user create a relay for spy others user, remember:

someone that run Tor proxy can make a relay and contribute to the net.
Directory Servers are reliable, but this part of the net are often under attack because is the "core" of Tor.

5 So, Is Tor Secure?

Tor can't guarantee the total anonimity, it solve only the problem of sending the packets hiding the real IP of the user. Tor browser integrates plug-in for improove the security of the browser, such HTTPS everywhere for solve the problem of the exit node, and NoScript for blocking all script in the web pages. The default configuration of Firefox are changed and by default block flash-player.
This is not sufficient for the total safety to the monitoring of the data.

5.1 Fingerprinting

When an user surf the internet, more companies track it, ISP, NSA or Google, Facebook and so on, with cookies or other methods such display recognize for improve the user experience for see the web pages, for send AdChoice.

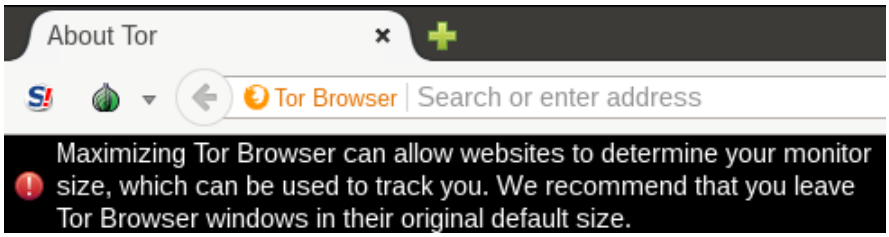


Figure 14: Maximization of browser's window is not recommended

When you load a web page, the browser broadcast some information about it to the website such plug-in installed, extensions, language, version of browser, screen resolution and depth timezone, a hash of the image generated by canvas/WebGL fingerprinting, request of "do not track" (turn off by default in Tor browser) etc. All of this things make a users profiling, this is not good if you want be anonymously. When you put the Tor browser full screen, it advice you with a message:

5.1.1 Test the fingerprinting

The EFF(Electronic Fountier Foundation) provide a page for test if your browser is safe against tracking. The test with a standard browser and Tor browser he gave very different results. *With security level low (default) Tor Browser does not provide at blocking tracking ads.* We can see that Tor browser provide a protect from fingerprinting against standard browser, but this not assure us that

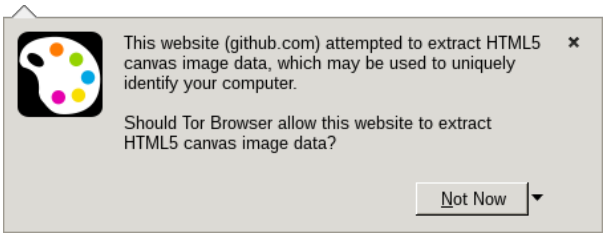


Figure 15: HTML5 canvas image data may let an attacker to uniquely identify a computer

Test	Result
Is your browser blocking tracking ads?	✗ no
Is your browser blocking invisible trackers?	✗ no
Does your browser unblock 3rd parties that promise to honor Do Not Track?	✗ no
Does your browser protect from fingerprinting?	✗ your browser has a nearly-unique fingerprint

Figure 16: Result of Chrome 53.x and Firefox 48.x with default settings

Test	Result
Is your browser blocking tracking ads?	✓ yes
Is your browser blocking invisible trackers?	✓ yes
Does your browser unblock 3rd parties that promise to honor Do Not Track?	✗ no
Does your browser protect from fingerprinting?	✗ your browser has a unique fingerprint

Note: because tracking techniques are complex, subtle, and constantly evolving, Panopticlick does not measure all forms of tracking and protection.

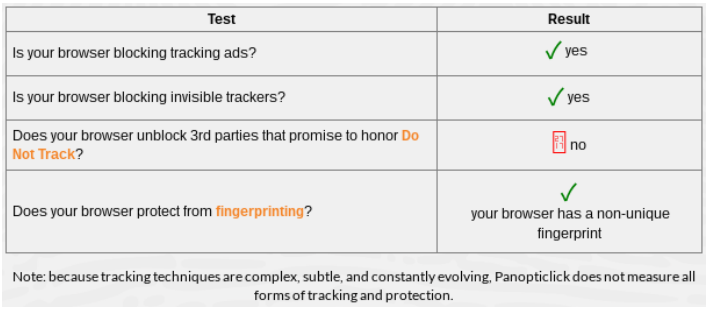
Figure 17: Result of Chrome 53.x and Firefox 48.x with AdBlockPlus, Ghostery and Blocking of third-party cookies and site data

we are safe from it.

5.2 Next improvements

The developers of Tor project they are working in the new version of the circuit:

- To improve the user experience of the 55-character-long onion addresses for next generation onion services (compared to the 16-character-long onion



Test	Result
Is your browser blocking tracking ads?	✓ yes
Is your browser blocking invisible trackers?	✓ yes
Does your browser unblock 3rd parties that promise to honor Do Not Track?	no
Does your browser protect from fingerprinting?	✓ your browser has a non-unique fingerprint

Note: because tracking techniques are complex, subtle, and constantly evolving, Panopticlick does not measure all forms of tracking and protection.

Figure 18: Result of Tor browser 6.0.5 (Firefox 45.4.0) with security level high

addresses used currently)

- A system for distributed random number generation on the Tor network.

6 Connecting To The Tor Network

There are two ways of connecting to the Tor network:

- Use a standard browser (e.g firefox or chrome) over the Tor network
- Use Tor Browser, the browser supplied by the Tor Project

6.1 Standard browser over Tor

For using a standard browser over the Tor network, download the Expert Bundle from `http://www.torproject.org`. Once launched, this software will

- open a socket listener
- contact a directory server, asking the necessary information for building a circuit
- eventually connect to the Tor network and establish a Tor circuit

The output of the program is shown in figure 19 on the next page. Now we have to configure the browser (we've used Mozilla Firefox 49.0) to use a proxy, choosing the parameters used by the Tor proxy. Therefore, as shown in figure 20 on page 21, we will select *manual proxy configuration*, put the loopback address in the *SOCKS Host* field, an finally put 9050 (in this example) in *Port*. Now we may check `https://check.torproject.org` to know whether we are using Tor or not. You should now see the content of figure 21 on page 21

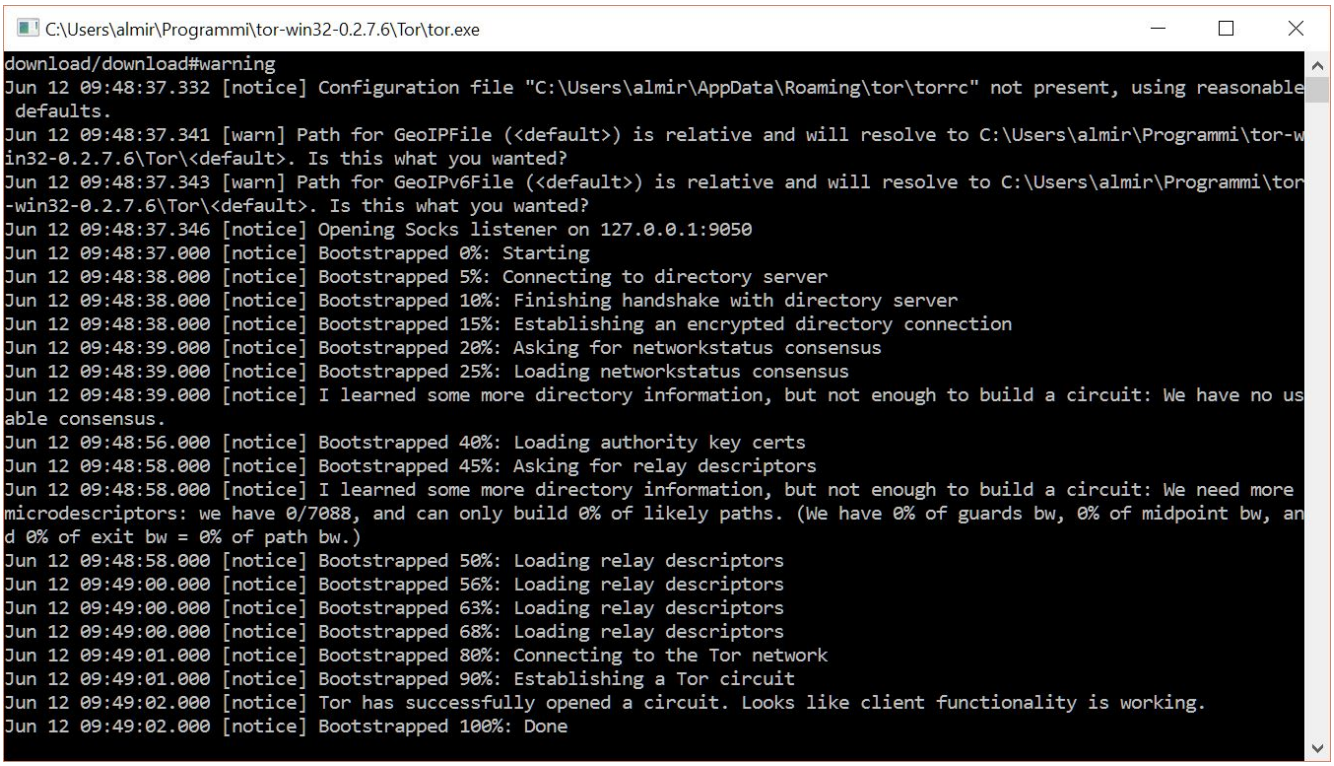


Figure 19: Output of tor.exe

6.2 The Tor browser

Even just being connected to the Tor network alone doesnt guarantees anonimity on the Internet: as seen in the previous sections, infomation leaks are still possibile, e.g running third-party script, or using HTTP between the exit-relay and the server. A kind of private browsing mode is required. Nevertheless:

- Firefox Private Browsing and Chrome Incognito mode are still lacking some important features (they mainly address local attack only)
- Althrough a lot of work has been done in the years between 2010 and 2014 by means of a collaboration between Torproject and Google (here is an article from 2010), the Chrome Incognito mode still not provides good defenses against fingerprinting and a network adversary. You can find on the faq the current situation, here you can find a *Why not Google Chrome?* paragraph, and finally here we have the tracker of Google Chrome Bugs for what concern Tor usbility.

The Tor browser (some important instruction on how to install it can be found at the end of the section) was developed to address this kind of problems. Based

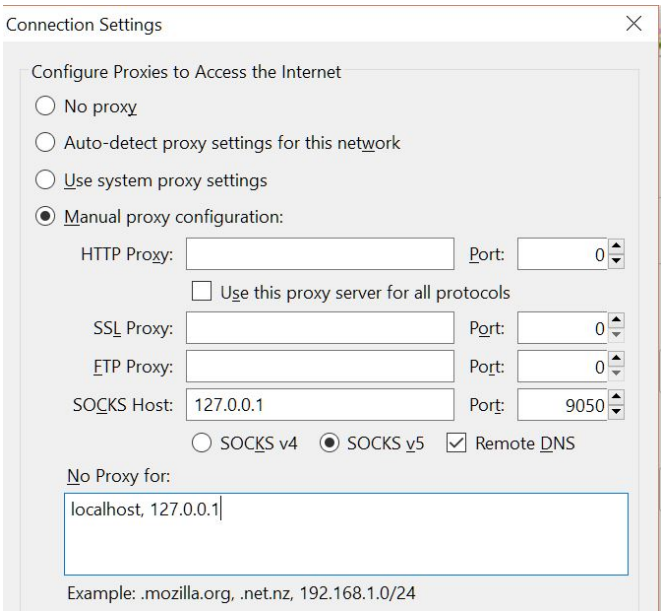


Figure 20: firefox proxy configuration

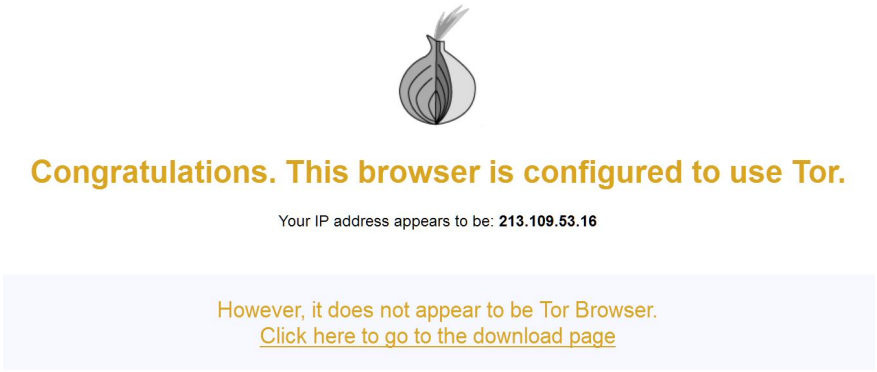


Figure 21: Partial success

on Mozilla Extended Support Release (ESR) Firefox branch, its developing is driven by the following requirements, aiming to defend against both network and local forensic adversaries:

- Security Requirements: The security requirements are primarily concerned with ensuring the safe use of Tor. Violations in these properties typically result in serious risk for the user in terms of immediate deanonymization and/or observability. With respect to browser support, security requirements are the minimum properties in order for Tor to support the use of

a particular browser. They are

  - Proxy Obedience: The browser MUST NOT bypass Tor proxy settings for any content.
  - State Separation: The browser MUST NOT provide the content window with any state from any other browsers or any non-Tor browsing modes. This includes shared state from independent plugins, and shared state from operating system implementations of TLS and other support libraries.
  - Disk Avoidance: The browser MUST NOT write any information that is derived from or that reveals browsing activity to the disk, or store it in memory beyond the duration of one browsing session, unless the user has explicitly opted to store their browsing history information to disk.
  - Application Data Isolation: The components involved in providing private browsing MUST be self-contained, or MUST provide a mechanism for rapid, complete removal of all evidence of the use of the mode. In other words, the browser MUST NOT write or cause the operating system to write any information about the use of private browsing to disk outside of the application's control.

- Privacy Requirements: The privacy requirements are primarily concerned with reducing linkability: the ability for a user's activity on one site to be linked with their activity on another site without their knowledge or explicit consent. With respect to browser support, privacy requirements are the set of properties that cause us to prefer one browser over another. They are:

  - Cross-Origin Identifier Unlinkability
  - Cross-Origin Fingerprinting Unlinkability
  - Long-Term Unlinkability

Here you can find *The Design and Implementation of the Tor Browser* document. For satisfying this requirements, a lot of modification w.r.t firefox as been brought, such as

- More than 100 about:config entry have been modified
- Over 60 patches to Firefox have been applied
- Tor uses some extensions, like HTTPS everywhere, NoScrpit and Torbutton

In figure 22 on the following page we may see an example of some Firefox about:config entry changed to improve Tor security. In the example we can see the entry modified regarding disk activity: they are mean to disable browsing history storage (do you remember the *Disc Avoidance* requirement?). A list of

all the entry changed in Firefox about-config can be found here, while here we have the list of the preferences modified, and finally here we have the list of the patches applied to Firefox in Tor Browser.

https://gitweb.torproject.org/tor-browser.git/tree/browser/app/profile/(

```
20
21  // Disk activity: Disable Browsing History Storage
22  pref("browser.privatebrowsing.autostart", true);
23  pref("browser.cache.disk.enable", false);
24  pref("browser.cache.offline.enable", false);
25  pref("dom.indexedDB.enabled", false);
26  pref("permissions.memory_only", true);
27  pref("network.cookie.lifetimePolicy", 2);
28  pref("browser.download.manager.retention", 1);
29  pref("security.nocertdb", true);
30
```

Figure 22: Some example of modification made to about:config

Torbutton is the component in Tor Browser that takes care of application-level security and privacy concerns in Firefox. To keep you safe, Torbutton disables many types of active content. Figure 23 shows the Torbutton windows, where we can see the circuit choosen for reaching this site. The inizial request started from Italy, so you may see as it travel across different countries. We may also change the circuit for this site, if we think this is necessary. More information about Torbutton can be found here.

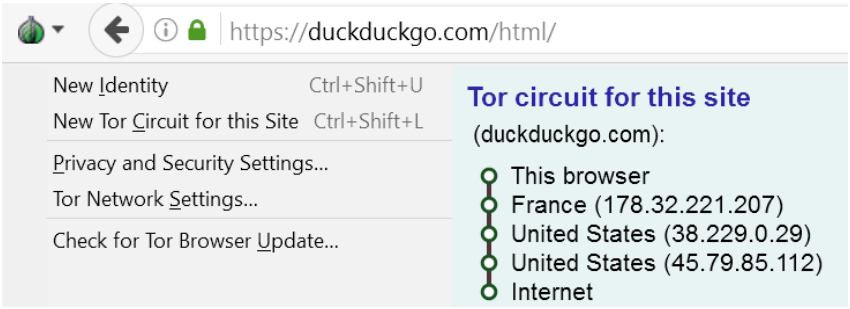


Figure 23: TorButton screen

6.2.1 What to know before choosing to being a relay

Let's say you want to being a relay and help the Tor community. Before doing this, make sure you have a Tor-friendly ISP that knows you're running an exit relay and supports you in that goal. This will help ensure that your Internet access isn't cut off due to abuse complaints. The Tor community maintains a list of ISPs that are particularly Tor-savvy, as well as ones that aren't.
There are three main kind of policy:

- ISP who allow any kind of Tor traffic (in this case, the disclaimer usually is *We do not block traffic, but you must respond to all abuse reports within 24 hours.*)
- ISP who don't allow only exit-relay
- ISP who prohibits *any activity* related to TOR.

Figure 24 show the output of `https://check.torproject.org` when we connect to the Tor network using the Tor browser.

Figure 24: Success screen