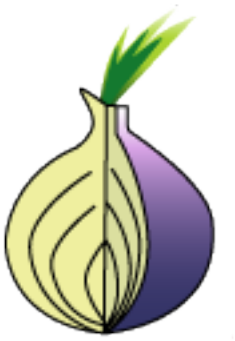
T O R

An Onion Routing Network



KEERAT SHARMA

CSC-288; Fall 2009

Contents

You aren't anonymous

Privacy[1]

1. *the state of being private; retirement or seclusion.*
2. *the state of being free from intrusion or disturbance in one's private life or affairs: the right to privacy.*
3. *secrecy.*

On the network

It is extremely difficult to remain anonymous on the internet. Consider that most Hypertext Transfer Protocol (HTTP) servers log the Internet Protocol (IP) address of a requesting host, along with a plethora of metadata. Here's a real world example to show how much information is available to a typical web server.

A client makes a request to a server:

```
http://[example server].com/index.html
```

A default installation of an Apache2 server installation would log the following:

```
97.119.50.76 - - [01/Oct/2009:21:35:22 -0500] "GET /index.html HTTP/1.1" 200
810 "-" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.1.3)
Gecko/20090824 Firefox/3.5.3"
```

Anyone with access to the server logs now has the requesting host's:

- IP Address
- Timestamp at which point they executed a request
- What they asked for
- The specific browser implementation and version that they used
- The operating system they they were on
- The language that the host is configured to operate on

Using this knowledge, we can leverage some easily available public resources and identify some more information. Let's leverage the IP address in the example above (`97.119.50.76`). We can run the IP against the whois database. Here's a snippet of what we see:

```
$ whois 97.119.50.76
[Querying whois.arin.net]
[whois.arin.net]
OrgName:    Qwest Communications Company, LLC
```

The server administrator can now identify that the requesting host was connected to the internet using Qwest Communications as their Internet Service Provider (ISP).

[1] Definition acquired at http://dictionary.reference.com/browse/privacy

They can dig more though. Consider this commonly available information when you interrogate domain name servers (DNS):

```
$ nslookup
> 97.119.50.76
Server:       XXX.XXX.XXX.XXX
Address:      XXX.XXX.XXX.XXX#53
Non-authoritative answer:
XX.XX.XXX.XX.in-addr.arpa  name = 97-119-50-76.omah.qwest.net.
```

Clearly, there's a hint in the above that the host resides as part of the quest network, and there's a 'omah' segment that seems interesting.

Publicly available databases that resolve IP address to geographic locations are freely available, and perform this service pretty well. According to the MaxMind GeoIP system, the above IP resolves to Omaha, Nebraska, which correlates with the DNS registration of the host.

And if we dig deeper

Network layer analysis isn't exclusive to just the target server though. A very significant threat surrounds compromised or malicious routers along the routing path for a connection allowing for examination of packet data and flow. More sophisticated adversaries can examine packets as they enter networks that they have some control over. In short, once a packet leaves a host, it can be subject to a variety of analysis by intermediate parties before it reaches the target server.

On an application specific basis, the volume of analysis capabilities increases significantly. In an HTTP setting, servers can set cookies to store information about the end user that will be re-transmitted on each subsequent request by the client. Many dynamic web applications can also perform URL re-writing, where session related information is embedded into the links on the page, allowing the server to maintain a session for the end user. Frequently this information flows over non-encrypted channels.

Content served by an HTTP application can execute on the client side and transmit back information to the server. The Capabilities[2] API within Adobe Flash (available on over 90% of end user browsers[3]) can harvest such information as the CPU architecture, whether the client hardware has a web camera, printing capabilities, and even if the executing code can have local file access. Transmitting this information back to the server is within the grounds of a typical client sandbox configuration.

The above are fairly simple ways to harvest information about an end user. There are much more sophisticated techniques that can exploit specific applications as well as network concepts. As an end user, you provide a plethora of information about yourself when you traverse the internet.

---

[2] flash.system.Capabilities - Flash CS4 Professional ActionScript 3.0 Language Reference: http://help.adobe.com/en_US/AS3LCR/Flash_10.0/flash/system/Capabilities.html

[3] Data acquired from: http://www.adobe.com/products/player_census/flashplayer/version_penetration.html

Tor

Tor aims to conceal its user's identities and to prevent their activity from being subject to traffic analysis. Tor provides anonymous communication channels, and eschews solving problems that are part of the application layer (cookies, communication in clear text, and such). Tor incorporates two core concepts- an anonymizing proxy, and a mix-network to create what's known as an Onion routing network.

Tor has been around since 2004, supported by the Electronic Frontier Foundation and the US Naval Research Laboratory. As of 2006, Tor became a 501(c)(3) non profit organization. It is built and maintained at the Tor project (http://www.torproject.org). The software has implementations for most operating systems, and is authored in C. Tor maintains an RFC style documentation body with a plethora of information within their source code repository[4].

Establishing virtual circuits

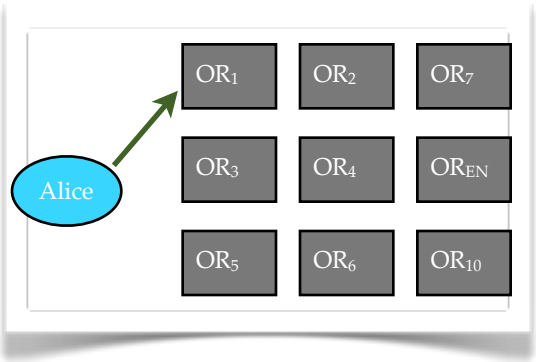
To get on to the Tor network, you need a Tor client. The client can run in user space, and does not need kernel or system level permissions. The client software is seeded with the addresses and public keys of trusted Tor directory servers. Each directory server maintains a redundant, multiply signed copy of the Tor network topology. Clients can only trust this document if it has been signed by a certain threshold of trusted directory servers (whose keys the client is already aware of). Clients poll the directory servers periodically to freshen their view of the network.

The core mechanism that allows Tor to maintain anonymity is the creation of virtual circuits using an onion routing scheme. Messages from the sender travel through a sequence of Onion Routers (ORs) within the Tor network, finally making their way to the recipient from an exit node (EN) in the Tor network. As such, the recipient only sees traffic as originating from the EN of the network, and the sender is kept hidden.

Clients on the Tor network operate through proxies known as Onion Proxies (*OP*). Typically, the Tor client and proxy execute locally in user space. Sometimes the proxy also adds application level filtering[5]- an HTTP specific example includes stripping off, or mutating headers.

An OP is responsible for creating and maintaining a number of virtual circuits that it rotates through, destroys and replaces over time. When Alice connects to the Tor network, she establishes a virtual circuit using a random set of Tor routers first:

1. Alice's OP contacts an Onion Router (OR_1) on the Tor network, with half a symmetric key (k_1) encrypted with the router's public key: $E(PU_{OR1}[k_1])$, and a chosen name for the circuit.
2. In return, OR1 provides Alice with an encrypted message that includes the second half of the symmetric key that will be used for communication k_2, along with a hash of k_{12} (the complete key[6]): $OR_1\{k_2, H(k_{12})\}$.

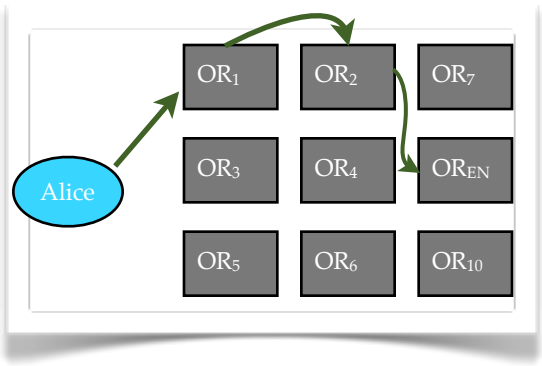


[4] Tor's subversion repository's doc directory: https://svn.torproject.org/svn/tor/trunk/doc/

[5] Tor is frequently bundled with privacy enhanced proxies like Privoxy (http://www.privoxy.org)

[6] The full symmetric key used between an OP and an OR is a math operation between the two halves. To keep the explanations clear, this has been simplified to k12- implying a symmetric key (k) generated using the first ($_1$) and second ($_2$) halves.

3. At this point, Alice and OR_1 have established a connection, and a symmetric key (k_{12}) which both parties had a hand in generating. Alice did not venture any identity of her own to OR_1 other than her network location, but all communication with OR_1 was either encrypted with OR_1's public key, or signed by OR_1.
4. Alice can then choose to extend the virtual circuit to OR_2. To do this, she sends OR_1 a request to extend the circuit to OR_2.
   a. This request contains a a component of a symmetric key to be assembled, encypted with OR_2's public key $E(PU_{OR2}[k_3])$. This is not related to the symmetric key that Alice and OR_1 set up earlier.
   b. OR_1 accepts the instruction to extend the circuit to OR_2, and forwards the encrypted payload from Alice to OR_2, associating the circuit ID that was created from Alice to OR_1, with the newly initiated circuit from OR_1 to OR_2.
   c. OR_2 accepts the encrypted payload and responds to Alice via OR_1 with a second component to finish the symmetric key between OR_2 and Alice and a hash of the total key, in a signed payload: $OR_2\{k_4,H(k_{34})\}$.
5. At this point, Alice has two symmetric keys: k_{12} (for communicating with OR_1), and k_{34} (for communication with OR_2).
6. It is likely that Alice would create one final relay, to a specific Tor node that is also an exit node: OR_{EN}.
7. Alice's virtual circuit now consists of OR_1-OR_2-OR_{EN}, with corresponding symmetric keys: k_{12}, k_{34}, k_{56}.
8. Note that the links established between Alice to OR_1,OR_1 to OR_2, and OR_2 to OR_{EN} are all TLS sockets, keeping traffic encrypted within the Tor network.



Clients typically establish a few virtual circuits when they join the Tor network. Considering that each virtual circuit requires a certain degree of overhead, having them already spooled up helps keep the user experience a little brisker as they don't have to wait on the process described above. Additionally, if there's a pool of virtual circuits at hand, they can be used randomly to further shield the sender, since each circuit will provide the recipient with traffic from a different network location (a different exit node).

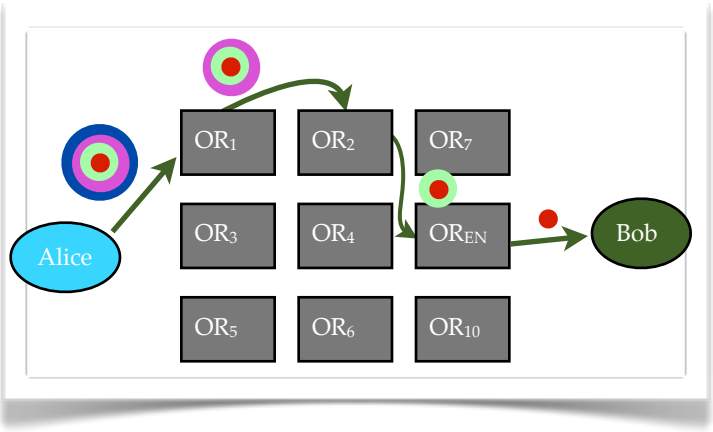
Routing Onions

With a virtual circuit established, a Tor client is ready to communicate with any network destination. Let's assume that Alice is using Tor, with an established virtual circuit: OR_1-OR_2-OR_{EN}. Let's also assume that Alice has negotiated corresponding symmetric keys with each OR: k_{12}, k_{34}, k_{56}. [See Establishing Virtual Circuits above for more detail on these assumptions]. Alice would like to communicate with Bob. The first step is to assemble a Routing Onion, which contains the payload for Bob, as well as the requisite security layer:

1. Alice assembles the message (M) that she would like to send to Bob. This may be a plaintext payload. It is identical to what Alice would transmit if she were not using Tor.
2. Next, Alice encrypts the message with the symmetric key she brokered with OR_{EN}: $E(k_{56},M)$. In addition, Alice adds a routing instruction indicating that the message is to be delivered to Bob. We currently have: $E(k_{56},M,Bob)$
3. Alice then encrypts this message with the key she negotiated with OR_2: $E(k_{34}, [E(k_{56},M,Bob)])$
4. The last encryption operation is with the key she negotiated with OR_1, resulting in: $E(k_{12},[E(k_{34},[E(k_{56},M,Bob)])])$

Alice now begins her transmission:

1. The message: $E(k_{12},[E(k_{34},[E(k_{56},M,Bob)])])$, is transmitted to OR_1. Since OR_1 and Alice had brokered the symmetric key k_{12} earlier, OR_1 can decipher the message, peeling away a layer, and getting the message: $E(k_{34},[E(k_{56},M,Bob)])$
2. OR_1 now transmits $E(k_{34}, [E(k_{56},M,Bob)])$, which it just deciphered to OR_2. In turn, OR_2 peels a layer, resulting in: $E(k_{56},M,Bob)$
3. OR_2 transmits $E(k_{56},M,Bob)$ to OR_{EN}, which peels away the last layer to get the message M, and the destination to send it to.
4. OR_{EN} opens a connection (perhaps even an insecure one, depending on the instructions from Alice) to Bob, and transmits M.
5. Bob sees the traffic as originating from OR_{EN}, effectively shielding Alice
6. Returning traffic from Bob, has a similar set of operations execute on it, except in reverse, with OR_{EN} encrypting the response from Bob first, followed by OR_2 adding its layer, and OR_1 adding a final layer. Alice can now peel away all the layers in a similar fashion to how she added all the layers on to get to Bob's response.



The benefits of using Tor

In any virtual circuit within the Tor network, an Onion Router is only aware of the previous and next OR in the constitution of the virtual circuit. Only the first OR ever knows the network location of the Tor client, and only the last OR (the exit node) ever knows the recipient of the client's messages. As such, even within the Tor network, most ORs on a virtual circuit don't know the terminal points on the circuit.

If messages are intercepted by a malicious agent within the Tor network, it may not be computationally feasible for them to be able to decipher the contents. They would have to be able to recursively decrypt the message, somehow discovering or brute forcing symmetric keys that were never transmitted as a whole. (The symmetric keys are merely generated by a math operation on two halves, which in both cases travel encrypted using asymmetric encryption).

Further, an intercepting party can intercept messages for only a limited duration since Tor cycles through virtual circuits and establishes fresh random ones regularly.

Malicious agents beyond the Tor network boundary will still get all the data that they used to get from a client, but it won't appear to be from the client any longer. Further, when Tor is used with a security enhanced proxy like Privoxy, the headers that would normally betray a client's details (operating system, browser version, and perhaps cookies) can be stripped off or replaced.

Here's an example using Tor, with the same server, and client used in the *You aren't anonymous* section. This is what gets recorded in the server's log:

```
79.136.30.69 - - [12/Oct/2009:20:42:51 -0500] "GET /index.html HTTP/1.1" 200
2638 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.7) Gecko/
2009021910 Firefox/3.0.7"
```

First off, notice that we' originated from a very different address than the original example. Additionally, notice the scrubbing that Privoxy dealt to the client description.

Digging deeper into the IP, we see the following.

```
$ whois 79.136.30.69
OrgName:    [snip] Network Coordination Centre
OrgID:      [snip]
Address:    P.O. Box 100[xx]
City:       Amsterdam
```

As evident, the virtual circuit that we negotiated over the Tor network established an exit node that resides in Amsterdam. There was nothing in that communication that identified who we really were, or where we originated from.

What you don't get when you use Tor

For all the magic that Tor provides, there is one significant issue that it can not overcome: *Timing Analysis*. Consider a malicious traffic analyst that only wants to know when a client is communicating. In that case, they just have to monitor the client. Tor will prevent the analyst from tracing the path of the traffic from the client to their destination, but the eavesdropper will be able to identify when the client is communicating. In the event that the eavesdropper is only interested in the client, and one specific endpoint, they may be able to leverage timing analysis by monitoring the endpoints for activity. The success of this technique is reduced sharply as the destination server gets more and more diverse traffic though.

Tor also doesn't counter bad practices from a client. In the event that they transmit data regularly over non TLS sockets, their data can still be compromised as it leaves an exit node en route to its destination. Even when Tor is used in conjunction with a privacy enhanced proxy, there are only certain protocols that the proxy can help with. And even with the protocols that such a proxy can operate with, there are limits- some sites requires cookies over HTTP, and don't offer HTTPS support. In the event that personal data is embedded in a cookie, the client may have no choice but to transmit in cleartext on the internet (from the exit node onward), and disclose data within a cookie or other parameter.

Why the world needs Tor

"Beware the Four Horsemen of the Information Apocalypse: terrorists, drug dealers, kidnappers, and child pornographers. Seems like you can scare any public into allowing the government to do anything with those four."[7]

"No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks."[8]

"Everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers."[9]

Chances are that you're reading this paper in the free world. Consider that in many nations, all traffic is monitored, and in many cases, a reasonable portion of the internet is cordoned off. Also consider that some governments leverage the internet to monitor their populace and press. Tor is a very effective countermeasure to all of the above.

Many citizens in China use Tor to get to sites that the Golden Shield Project[10] would normally block access to. The US military is a heavy user of Tor as a transmission medium for field operatives. Journalists in countries where the press is monitored or censored use Tor to communicate with their home offices to file reports. Victims of crime who don't want to be traced used Tor. Corporations use Tor to transmit information secretly.

One could argue that if Tor can provide freedom of press, and in broader terms, freedom of speech, then it is worth having, even at the risk of malicious users leveraging it. I saw a Tor operator request a fresh exit node near China, work with a volunteer to assign the requisite trust certificates to it, and get it operational. Within minutes it was in use, and the volume of data that had already flowed through it was remarkable (more than 700 MB in five minutes). Clearly, there's an active segment of a population in East and Southeast Asia that is using it heavily. That alone may be testament enough to why we need Tor.

---

[7] Schneier on Security: Computer Crime Hype,
http://www.schneier.com/blog/archives/2005/12/computer_crime_1.html

[8] Universal Declaration of Human Rights; Article 12

[9] Universal Declaration of Human Rights; Article 19

[10] The Golden Shield Project is sometimes referred to as the Great Firewall of China:
(http://en.wikipedia.org/wiki/Golden_Shield_Project)

Bibliography

Tor: The Second-Generation Onion Router
Roger Dingledine, Nick Mathewson, Paul Syverson
https://svn.torproject.org/svn/tor/trunk/doc/design-paper/tor-design.html

Tor directory protocol, version 3
https://svn.torproject.org/svn/tor/trunk/doc/spec/dir-spec.txt

Untraceable Electronic Mail, Return Addresses and, Digital Pseudonyms
David L Chaum, University of California, Berkeley
http://freehaven.net/anonbib/cache/chaum-mix.pdf

Additional information on Tor and Onion Routing:
http://en.wikipedia.org/wiki/Tor_(anonymity_network)
http://en.wikipedia.org/wiki/Onion_router
https://wiki.torproject.org/noreply/TheOnionRouter/TorFAQ
http://www.torproject.org

Many papers pertinent to anonymity online:
http://freehaven.net/anonbib/topic.html#Anonymous_20communication

With thanks to:
Sebastian Hahn, arma, and dr|z3d, for their information on Tor via IRC at #tor on orc.oftc.net