Verksamheten för samhällets informations- och cybersäkerhet
Enheten för operativ cybersäkerhet och it-incidenthantering, CERT-SE
Ásgeir Davidsson
asda@cert.se

Tor – the Onion Routing Network

The Tor network is a low latency, onion routing network operated in part by the Tor Project, Inc. and a worldwide network of volunteers.

The network provides anonymity to both clients and servers, functioning as a 'black box' that hides the routing information of network participants. For this reason, the Tor network has become one of the most popular anonymity tools on the internet with relays and users in more than six thousand servers forwarding traffic on behalf of the network's clients[1].

According to the 'Brief Selected History' of Paul Syverson[2], Tor began its life as a research project of the United States Naval Research Laboratory (NRL) in 1995 when research began on anonymous routing. In 1996, three researchers at the NRL, Goldschlag, Reed, and Syverson, published the seminal paper 'Hiding Routing Information'[3] and coined the term 'onion routing'. An onion routing system is capable of forwarding encrypted traffic between a client and a server on the Internet through a series of proxies. Within this system only the last proxy is able to read the original transmission, and no single proxy has knowledge of both the origin and destination of the traffic.

The onion metaphor derives from the network being setup by encrypting the transmission in layers, much like an onion, in which the outermost 'layer' of encryption is the one each proxy has to decrypt (or peel, metaphorically speaking) in order to learn where to forward the packet.

Following the publication 'Hiding Routing Information', a proof-of-concept network was built on a single machine in the NRL running five relays (the Tor specific term for the proxy servers described above). This was the original Tor network, running what has since been named the 'Generation 0' implementation. This network would run for the next four years, until January of 2000 when it was shut down. During its four year lifespan, the Generation 0-network managed to process over 50 000 connections per day and reported a peak connection load of 85 000 connections at its busiest.[2]

Although this network was primarily used as a testbed for performance tuning and scalability tests, work was already underway in 1997 to build the next

iteration of Tor, the ‘Generation 1’ design. The design specification for the next generation was published by the same team (Goldschlag, Reed and Syverson) under the title 'Anonymous Connections and Onion Routing'[4]. Most notably, the Generation 1 design includes the addition of exit policies to allow relay operators more control over the traffic they wish to forward to the Internet on behalf of the network.

Work continued on the Generation 1 design until 1999 when funding ran out, effectively halting further development. At this point, work on Tor had been funded by the NRL via the Office for Naval Research and the Defense Advanced Research Projects Agency (DARPA).

In 2001, funding for onion routing development was reestablished by DARPA, under the Fault Tolerant Networks Program. A year later, in 2002, a decision was made to abandon the Generation 1 code and build a new version called Generation 2. The Generation 2 design is the foundation of the currently deployed Tor network.

In 2003, further funding was granted the Tor project from the Office of Naval Research (ONR) to deploy and further develop the Generation 2 network. The ONR also funded the development of hidden services, which would later become a popular but controversial feature of the Tor network.

The modern Tor network was deployed in October of 2003, as the project transitioned from being an internal project of the United States Naval Research Lab to becoming its own non-profit organization. As part of this handover, the code of the Tor network was open-sourced under the MIT license.[2] From this point onwards, further development and network management has been handled by the Tor Project, Inc.- a registered US non-profit organization composed of the original developers and newcomers alike.

In 2004, the formal design specification of the Generation 2 Tor network is published under the title 'Tor: The Second Generation Onion Router'[20], by Dingledine, Mathewson and Syverson. Although the Tor network today has several additions that are not covered in this publication, this is the last formal paper on the design of the network. Further changes to the Tor network are documented on the Tor Project, Inc.'s blog[5][6].

Around this point, the Tor network begins to grow as volunteers begin hosting relay servers around the world. At the end of 2003, the Tor network was composed of a handful of nodes. By the end of 2004 it had grown to more than 100, and as of 2015 there are more than 6000 relays operational worldwide, with strong concentrations in Germany and the United States. Every day, the Tor network is estimated to handle connections for more than 200 000 users.[7]

How Does the Tor Network Work?

The Tor network requires three components to anonymously connect a user to a server on the Internet. These are; relays, exit relays and directory servers.

Relays are the most common type of server deployed in the Tor network. Relays can be operated by anyone willing to run the Tor software and donate some amount of bandwidth. Relay servers only forward traffic inside the Tor network and as such, are never used to connect out of the network to the Internet. Relays are considered relatively 'safe' to operate, as the operator is exposed to relatively little liability. However, because relay addresses by necessity must be public, the operator risks appearing on blacklists in countries that censor or block Tor.

Exit relays perform the same function as relays, but also forward traffic from inside the Tor network to the outside Internet. Each exit relay has an associated exit policy, which specifies which kind of traffic, port-wise, the relay is willing to forward out of the network, and to which networks. The introduction of exit policies was a huge step forward in reducing abuse from the Tor network, as exit policy operators could easily choose to reject abuse-prone protocols such as SMTP, NTP, SSDP, etc.

Directory servers are the only centralized part of the Tor network, and are operated by vetted organizations around the world. Directory servers are responsible for gathering a list of active relays in the network, and providing the public keys necessary for clients to establish secure connections to them. In addition, the directory servers also keep track of the exit policies of exit relays. All of this is essential for clients to be able to construct circuits through the Tor network.

The address of the directory servers is included in the distributed Tor software, along with the public key of each directory server. This is the same for relays which will advertise their availability and statistics to the directory servers in order to be listed appropriately. The directory servers will then build a consensus concerning the state of the network between themselves. Once this consensus is reached, the result is signed by the directory servers and distributed to relays willing to cache the directory consensus. To avoid creating single-points of failure in the network architecture, directory servers are not fully trusted; instead clients and relays will verify the signatures of the consensus to make sure that at least half of the directories agree on its contents. This design prevents anything less than a compromise of half the directory servers from affecting the security of the directory consensus.

As a side note, these directory servers are the core of the particular Tor network that is synonymous with Tor itself. Anyone could establish a new Tor network

by providing their own trusted directory servers, as the code that runs the clients, relays and directories is entirely open-source. However, there is a degree of anonymity gained by aggregating a large amount of traffic from different users in the same systems, and so there have been very few initiatives offering an alternative to the default directory servers.

With these three concepts explained, it is possible to build a 'circuit' through the Tor network to the Internet.

As an example, consider the case of the client Alice wishing to anonymously connect to the server Bob (see also [8] for illustrated explanation)

1. Alice begins by downloading and verifying the integrity of the Tor client software.
2. Alice connects to a directory server requests a copy of the directory consensus from the server. Alice verifies that the directory consensus sign signed by at least half of the directory servers, and now has a list of all currently active public relays and exit nodes in the Tor network, as well as their public keys.
3. Alice now selects three relays with which to build a circuit through the Tor network, where one of those is an exit relay with an exit policy compatible with her purpose.
4. Alice gets the IP address and identity digest for Relay 1 from her directory consensus and a temporary session key with the relay. Alice and Relay 1 now share a session key that will be used only for this communication and then destroyed in accordance with the principle of perfect forward secrecy. Alice now has a 'circuit' of one hop.
5. Alice will then request Relay 1 to extend her circuit by one jump to Relay 2 by exactly the same method. Alice establishes another temporary key with Relay 2 through Relay 1 and so gains a session key shared only by her and Relay 2. Alice repeats this process again with Relay 3 and has thus established a 3-jump circuit with a unique session key known only to her and each respective relay.
6. Alice can now construct a request for Bob and sequentially encrypt it with the session keys of the Relays, in the order of farthest-to-closest. When Relay 1 receives this transmission, it decrypts (or peels) the transmission and discovers where to forward the still-encrypted contents to next. When the transmission arrives at Relay 3 (the exit relay), the traffic is decrypted into its original form and sent to Bob.

When Bob eventually replies, the encryption sequence is repeated in reverse with each Relay adding another layer of encryption and passing the reply back to Alice.

Guards and Bridges

Although not strictly essential for building circuits, the Tor network does have two additional types of relay that serve an important role in the modern Tor network. These are guard relays and bridge relays.

Guard relays function in exactly the same way as regular relays (forwarding traffic within the Tor network) but serve a purpose in protecting clients from attacks using traffic analysis. When a client connects to the Tor network, it will randomly pick a set of relays to use as guard relays. When building circuits, the client will always use a relay from this set to build the first jump in the circuit to. This design does not defeat such attacks, but changes the likelihood of its success for the majority of clients.

Bridge relays are relays that are not publicly listed by the Tor network directory servers. Instead, the address of bridge relays are distributed through out-of-band (relative to the Tor network) channels to end users. The benefit of this approach is that an adversary, capable of censoring and/or blocking internet traffic, is unable to block the entire set of relays in the Tor network. Assuming that a user can receive the address of a bridge relay that remains unknown, he or she can connect through the bridge to gain access to the Tor network in its entirety.

Hidden Services

A 'Hidden Service' is a server that communicates exclusively with clients using the Tor network. In addition to providing the same routing anonymity as for clients, Tor includes a mechanism for allowing hidden services to maintain a persistent name, or URL address. This is accomplished through a Top Level Domain system that utilizes an in-network distributed hash table (DHT) for lookups.

The mechanisms used for communicating with hidden services are different from the mechanism used by clients wishing to reach the Internet through the Tor network. Although reaching the hidden service will still require building circuits, a client wishing to reach a hidden service will not need to make use of an exit relay. Instead, two additional types of relays are required to hide the client and hidden service from one another effectively. These are introduction points and rendez-vous points.

When a hidden service is established, its operator will create a public key pair specific to the hidden service. The hidden services will then ask a set of relays to function as introductions points. These relays will accept a public key from

the service and will maintain a persistent circuit to the hidden service. A hidden service will have several introduction points to prevent denial-of-service (DoS) attacks against any single introduction point.

Having established circuits to these introduction points, the hidden service will create a 'service descriptor' composed of the public key of the service and the location of its introduction points. The service descriptor is then signed with the private key of the hidden service and uploaded to the distributed hash table of the Tor network. The descriptor becomes the value in the table, and a 16 character hash derived from the public key of the hidden service will be used as the key. A client can then use this hash with the '.onion' top level domain designator to perform a lookup in the table through the Tor software package.

These 16 character addresses are random alphanumerical strings, reflecting the fact that they are key values in a DHT. However, it is possible and relatively popular for well-known hidden services to brute-force the hashing algorithm performed on the public key of the hidden service until the resulting hash is human readable. Although brute-forcing the entire hash would take far too long, typically the first 8 characters are within the grasp of a patient operator[9].

As the Tor network provides no means for clients to discover a hidden service, a user must receive the address of the service he or she wishes to visit out-of-band. As might be imagined, this design means that users must place their trust in the identity of a service from an outside source. Combined with the fact that popular services have begun using brute-forced addresses, hidden services are relatively vulnerable to look-alike attacks. For example, a user is relatively unlikely to be able to discern the difference between 'thesilkroad10dkr' and 'thesilkroadnz03k9' when hovering over a link on a semi-trusted website such as The Hidden Wiki.

Assuming that a user can find a legitimate '.onion' address for the service he or she wishes to visit, the client will look-up the address in the DHT and retrieve the public key and introduction points of the hidden service.

The client will then build a circuit to relay willing to act as a rendez-vous relay and gives it a one-time secret. The client then builds a circuit to one of the introduction points and sends it a copy of the one-time secret encrypted with the public key of the hidden service and the address of the rendezvous relay. The introduction point recognizes which hidden service the user is trying to reach and forwards the encrypted one-time secret back to the hidden service through its persistent circuit to it.

Upon receiving and decrypting the one-time secret, the hidden service then builds a circuit to the rendez-vous relay and sends it the one-time secret.

Having received the same one-time secret, the rendezvous relay will then forward traffic between the circuit connecting it to the client and the circuit connecting it to the hidden service.

The anonymity of hidden services provides a powerful draw to many different groups of users[10]. Journalists and whistleblowers have made use of hidden services to transfer documents and establish trust. The New Yorker, for example runs a hidden service called 'Strongbox' that allows those wishing to remain anonymous to upload large sets of documents in a secure manner to the organization[11]. Wikileaks likewise maintains a similar service on Tor[12].

Hidden services have also been used by less reputable actors as a mechanism to distribute child sexual abuse material, facilitate the sales of drugs and other contraband, and act as command and control infrastructure for malware.

Freedom Hosting was a large Tor-based hosting company whose infrastructure was dismantled in its entirety when it was discovered that it had become a hotbed for child sexual abuse material.

The Silk Road was for a time the most successful online drug and contraband marketplace until it was taken down by the FBI in 2013 after a prolonged sting operation. The operator of the service, Ross Ulbricht, was also caught as part of the operation and was sentenced in 2015 to life in prison for his part in running the operation. By the time of its seizure, the Silk Road was estimated to have facilitated the sale and movement of two hundred million dollars worth of contraband[13].

Malware has also increasingly been making use of the Tor network and hidden services to host command and control infrastructure. Ransomware (malware that encrypts sensitive data belonging to the victims and attempts to ransom it back to them) in particular has been making increased use of hidden services to host key-generation servers and landing pages. As of 2015, three of the most common ransomware variants (Cryptolocker, CBT-locker and Torrentlocker) all use hidden services in this way[14].

The lack of a discovery mechanism inherent in the design of hidden services also led to the creation of an 'index' of known hidden services called 'The Hidden Wiki'. The hidden wiki has existed in one form or another since the deployment of hidden services in 2004[2]. The hidden wiki and its associated mirrors/imitators maintain a list of known hidden services for users to discover based on their interests. However, the listing is neither exhaustive nor entirely trustworthy, as the various hidden wikis are run by anonymous operators. Additionally, the problem of look-alike addresses, as mentioned previously, is especially pertinent for hidden services such as the hidden wiki, where

changing addresses is relatively trivial or could be willfully overlooked by the operator to directly clients to copycat or compromised hidden services.

Censorship and Anti-Censorship

The design of the Tor network did not originally identify censorship as part of its threat model, nor did it include designs for anti-censorship measures from the outset[15]. However, as the Tor network has become increasingly popular in countries with regimes that censor or otherwise block certain kinds of Internet traffic, the Tor Project, Inc. began extending the Tor design to cope with such threats. The evolution of such measures has typically followed an incremental cat-and-mouse pattern based on the specific techniques used to block or disrupt the network at any time.

The history of Tor and the Great Firewall of China illustrates the evolution of these censorship and anti-censorship measures.

Initially, the simplest way for the censors to block usage of the Tor network was to block traffic destined for the networks directory servers. These are relatively few in number, and blocking traffic destined to these servers would prevent clients from bootstrapping themselves into the network.

In addition, since the directory servers maintain a list of all active relays in the Tor network, it is trivial for an adversary to connect to the directory servers and use this list to create a comprehensive blacklist. This would prevent Tor clients from building the first hop into the Tor network, or relays within the affected network from extending circuits between known relays.

To counter this, the Tor Project, Inc. introduced bridge relays. Bridge relays are not be listed by the directory servers but instead be distributed through a semi-automatic mechanism. Clients operating in hostile network can connect directly to their acquired bridge relay and from there retrieve the directory consensus from the directory servers. As long as the bridge is located outside the hostile network, this measure makes blocking the directory servers on the client's network futile. To further frustrate censorship efforts, bridges are be distributed in small sets so that no client would have a complete list of all the bridges. To receive a bridge, a potential Tor user can send a specifically formed e-mail to a Tor Project, Inc- e-mail account from either a Yahoo or Google e-mail address. The logic behind this limitation being that Yahoo and Google take steps to prevent bots from creating accounts in bulk and that this would hinder efforts by censors to enumerate the bridge relay set by repeatedly asking for bridges from different accounts.

While initially successful, the tide turned once more in favor of the censors in 2011 when the Great Firewall began employing deep packet inspection (DPI) techniques to identify Tor traffic on the wire by inspecting the content of traffic

rather than relying on metadata. Upon detecting, for example, the characteristic start of a Tor connection, the censor could then spoof reset packets to trick the receiver into thinking the connection has been closed, or further block the recipient entirely.

To avoid detection by DPI mechanisms, Tor introduced pluggable transports; a mechanism that allows Tor traffic to more closely imitate the look of various protocols to hide itself on the wire. Work continues in this area to improve the profile of Tor's communication so that traffic can continue to escape detection.

Lastly, as Dingledine and Mathewson point out in their 2006 paper on how to add blocking-resistance to the Tor network[16], it is important to understand the goals and interests of censors in order to fight them more effectively. Given the nigh-impossible task of censoring every participant in a national network at all times, censors typically adopt a 'best-effort' approach to their work. Censors do not, for example, need to devote as many technical resources to actively censoring the network if the chilling-effect of periodic crackdowns is sufficient to make users self-censor themselves.

Likewise, a censor may be content to allow a certain percentage of users to access forbidden information over Tor, as long as they are consuming that information, as opposed to producing it. The design of the Tor network does unfortunately make it possible to distinguish whether a user is more likely to be producing or contributing content rather than consuming it, allowing a censor to focus their efforts on the former.

Limitations of Tor

The Tor design lends itself well to protect users from passive surveillance, malicious relays, and hostile networks. However, there are two primary classes of attacks that Tor cannot provide the same level of protection for; traffic confirmation and end-to-end attacks.

Traffic confirmation allows a network operator or an adversary capable of passive surveillance to determine whether or not a user is using Tor. In recent times efforts have been made to better hide the signature of Tor traffic on the wire to resist censorship measures. The threat model of Tor does not consider traffic confirmation (by itself) to be a threat to the security of the communication: the adversary only knows that the client is using Tor but not which site the user is visiting or why. However, this assumption is less valid in very local contexts. Consider the Harvard bomb threat case of 2013[17], where a student posted a bomb threat using the Tor network. Even though his communication was anonymous, the student was caught when Harvard network administrators were able to investigate which users had recently used

Tor immediately preceding the time of the threat. By using traffic confirmation in this way, the investigating authorities immediately had a lead.

End-to-end attacks are a class of attack that can de-anonymize or compromise a client or server without compromising the routing anonymity of the traffic exchanged between them. By definition, end-to-end attacks are considered to be outside the threat model of the Tor project. Users are typically left to secure themselves against end-to-end threats such as malicious JavaScript, WebRTC exploits and other protocols or services that might send traffic to an adversary-controlled machine over the regular Internet. Since the Tor client functions as an application agnostic SOCKS proxy, it must rely on those applications to send traffic to it to be forwarded. An adversary capable of forcing the client into executing code could therefore explicitly disregard this and send traffic directly to an adversary controlled machine. The Tor Project, Inc. has partially recognized this problem and as a result, a project was started up to build a complete, bootable environment specifically configured to minimize its attack-surface with regard to end-to-end attacks. The result of this project was The Amnesic Incognito Live System (TAILS)[18], which can be booted from a USB-drive and provides a user-friendly way of setting up an ephemeral environment for Tor use.

Blocking Tor

Although censorship efforts to block Tor usage on public networks is generally frowned upon, there are situations in which there exists a legitimate need to block access to anonymization services such as Tor. Tor-enabled ransomware, for example, will generally not function or even encrypt data unless it is able to contact a hidden service command and control to retrieve or upload encryption keys. In this context, blocking Tor traffic can be considered to be an additional layer of defense.

Similarly, private network operators may be skeptical of the abuse-related traffic that originates from the Tor network. In 2014, the U.S Treasury department found that the majority of traffic received by banks originating from Tor exit relays was fraudulent or malicious[19]. An organization considering whether or not to attempt to block access to Tor should consider whether or not there might be a legitimate need for its users to anonymously access its infrastructure or services, especially if it has users in regimes that filter information, and optimally consider whether a less severe option than a total block might be effective. Wikipedia, for example, faced a great deal of abuse from the Tor network where anonymous clients were making changes to articles. Instead of blocking Tor network access entirely, the Wikimedia Foundation opted to disable certain features like editing while still allowing those with a legitimate need for anonymity to access their content.

For those wishing to block traffic originating from Tor, the simplest means of doing so would be to simply block the entire list of exit relays. On the other hand, blocking Tor traffic originating from inside an operator network is somewhat harder, given that it is in principle the same problem faced by censors. Private network operators might be better off locking down endpoints to prevent Tor software from being installed, and further locking down their physical assets by preventing machines from booting live-USBs such as TAILS, rather than attempting to block Tor traffic on the wire.

References

[1] Tor metrics for number of relays and bridges
https://metrics.torproject.org/networksize.html

[2] Early history of Onion Routing
http://www.onion-router.net/History.html

[3] Hiding routing information
http://www.onion-router.net/Publications/IH-1996.pdf

[4] Anonymous Connections and Onion Routing
http://www.onion-router.net/Publications/SSP-1997.pdf

[5] Changes to Tor since the 2004 design, part 1
https://blog.torproject.org/blog/top-changes-tor-2004-design-paper-part-1

[6] Changes to Tor since the 2004 design, part 2
https://blog.torproject.org/blog/top-changes-tor-2004-design-paper-part-2

[7] Tor metrics for number of directly connecting users
https://metrics.torproject.org/userstats-relay-country.html

[8] An overview of how Tor connections work
https://www.torproject.org/about/overview.html.en

[9] Github repository of Shallot, a .onion address bruteforcing tool
https://github.com/katmagic/Shallot

[10] Who uses Tor?
https://www.torproject.org/about/torusers.html.en

[11] New Yorker Strongbox
https://projects.newyorker.com/strongbox/

[12] Wikileaks: Tor
https://wikileaks.org/wiki/WikiLeaks:Tor

[13] Silk Road creator Ross Ulbricht appeals life sentence
http://www.forbes.com/sites/katevinton/2015/06/05/silk-road-creator-ross-ulbricht-appeals-life-sentence/

[14] State of ransomware 2015, Fox-IT
http://blog.fox-it.com/2015/09/07/the-state-of-ransomware-in-2015/

[15] 28c3: How governments have tried to block Tor
https://www.youtube.com/watch?v=DX46Qv_b7F4

[16] Design of a blocking-resistant anonymity system
https://svn.torproject.org/svn/projects/design-paper/blocking.html

[17] Harward student use TOR to send bomb threat without good opsec.
http://www.forbes.com/sites/runasandvik/2013/12/18/harvard-student-receives-f-for-tor-failure-while-sending-anonymous-bomb-threat/

[18] TAILS – The Amnesic Incognito Live System
https://tails.boum.org/

[19] Krebs On Security; Treasury Dept: Tor a Big Source of Bank Fraud
http://krebsonsecurity.com/2014/12/treasury-dept-tor-a-big-source-of-bank-fraud/

[20] Tor: the second generation onion router
https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf