10/22: Tor

Scribe: Ashkan Hosseini

1 Introduction

Tor is a free software that allows people to connect to servers on the internet anonymously.

1.1 Anonymity

A user communicates with a server through transmission of packets, which contain user's IP address. Anonymity here means an adversary should not be able to find out what IP address is communicating with what server, and the server should not be able to identify what IP address is sending those packets.

2 Main idea

At a high level the main idea behind Tor is to transmit traffic of users through bunch of different nodes such that each node has no information about where the packet is coming from, and where it is going to. To achieve this, Tor uses onion routing. In an onion network, messages (packets) are enclosed in layers of encryption, called Onions (due to their layered architecture). Each node in this network is called an onion router (OR), and each OR is responsible to peel away a single layer (decrypts), uncovering the data for the next node in the network.

2.1 Design

A user first needs to establish a circuit in order to establish a connection with a server. The circuit path is established via Onion Proxy (OP), a local software that runs on client's machine, which uses SOCKS. Tor has a directory of approved ORs along with their public keys. Having a directory that is manually checked, prevents malicious users from creating many ORs to control the network traffic, and it also helps the OP to be aware of available ORs. OP picks a sequence of ORs by downloading the list of available ORs from the directory, and uses their public key to form its circuit. The first OR in the circuit is called the entry guard, and the last OR which decrypts the final layer is called the exit node. The current design of Tor uses 3 ORs, and Onion routers communicate with each other and OP via TLS connections with ephemeral keys. According to the paper, this is how Tor provides forward secrecy.

2.2 Cells

A circuit path consists of a few ORs. The traffic that comes from the client is put in fixed size cells (512 bytes) to make traffic analysis harder. The cells get unwrapped at each OR by a symmetric

key. Cells are the unit of communication in Tor, which consist of a payload and a header. Cells are either control cells (create, destroy, etc.) or relay cells (carry end to end stream data). For example, if the user wants to establish a TCP stream it first sends a relay extend (which is a control cell) to the first relay, specifying the address of the next OR, and exchange symmetric key using Diffie-Hellman with the second relay. This keeps on going in a similar fashion until the circuit is setup. After that the user sends relay cells, and the relay cells carry the end-to-end stream data.

2.3 Constructing a circuit

As mentioned above, the circuit path is established via Onion Proxy (OP), a local software that runs on client's machine. User's OP construct circuit paths incrementally (one hop at a time).
Each circuit has an ID. To create a circuit, c_1, OP first picks a secret key x_1, and then OP sends a "create" operation to an entry guard, OR_1, and they the do a Diffie-Hellman key exchange as follows:
Note: pk_i denotes the public key of OR_i which OP finds from the directory.

- OP sends "c_1, $Enc_{pk_1}(g^{x_1})$"
- OR_1 chooses random y_1 and replies with "created c_1, g^{y_1}"
- OP computes $k_1 = g^{x_1 y_1}$

Client (OP), and OR_1 now have shared a key which is k_1. For each subsequent OR, OP sends a "relay extend" message via circuit (i.e. Enc_{k_1}(extend, OR_2, $Enc_{pk_2}(g^{x_2})$)), and at the end OP shares a symmetric key with each one of the ORs.
Note: OR_2 does not have have any information about client's network.

2.4 Relay

Once the client has established a circuit they can send relay cells. When an OR receives a relay cell, it looks up the corresponding circuit to decrypt the relay header and payload with the session key of that circuit. Similarly, when OP receives relay cells it iteratively unwraps the relay header and payload with the session keys shared with each OR on the circuit, from the closest to farthest.

- OP sends $\xrightarrow{\text{Relay c}_1,\, Enc_{pk_1}(Enc_{pk_2}(\text{Begin connection}))} OR_1 \xrightarrow{Relay\ c_2,\, Enc_{pk_2}(\text{Begin connection})} OR_2 \rightarrow$ Begin Connection
- OP recieves $\xleftarrow{\text{Relay c}_1,\, Enc_{pk_1}(Enc_{pk_2}(\text{Connected}))} OR_1 \xleftarrow{Relay\ c_2,\, Enc_{pk_2}(\text{Connected})} OR_2 \leftarrow$ Website sends

3 Threat model

In low-latency anonymity systems such as Tor, an adversary who can control the traffic at entry node and exit node might be able to tell what IP address is talking to what server by finding a timing pattern, and Tor does not protect against these sorts of attacks. However, this attack is quite hard to do, because Tor relays are chosen at random for every connection. Thus, an attacker needs to control a very large number of entry nodes and exit nodes to have a good chance of success. Tor also does not protect against a global adversary.

4 Discussion

The list of exit nodes are known to public, so some services might choose to block Tor. In fact there are services that are currently blocking Tor, such as Fox News. Wikipedia also doesn't allow its users to edit articles using Tor, unless they get a "Tor exception" by sending an email request to Wikipedia.
There are also countries that block Tor such as China. They do this by ensuring that ISPs in the country filter connections to entry points (since they are known to public). To get around this issue, users can use bridges (also called Tor bridge relays) that are alternative entry points to the Tor network that are not all listed publicly.

5 Presentations

5.1 Circuit Fingerprinting Attacks

Tor allows users to hide their locations while offering various kinds of services, such as web publishing or an instant messaging server. The client and the hidden service can communicate using a rendezvous point (RP) that they agree to at one of the hidden service's introduction point (IP). The objective behind circuit finger printing attacks is to:

- Identify if a user is using any hidden services.
- If they are using hidden services, then identify the hidden services that they are using and de-anonymize the users.

5.1.1 Threat model:

- Passive attackers.
- Malicious entry guards.

5.1.2 How it works:

- Guess an entry guard.
- Compromise a set of tor nodes.

5.1.3 Why it works:

- Only need to compromise one entry guard.
- Users and hidden services frequently change the set of tor node used as entry guards.

5.1.4 Characterstics:

- HS-IP has exactly 3 outgoing cells but slightly more incoming cells.
- HS-IP's are long-lived and Client-IP's are short-lived.
- Only Client-RP can have a packet sequence of (-1-1+1).
- HS-RP circuits have more outgoing than incoming.

5.1.5 Main idea:

- Guess if the circuits are used to communicate with hidden services. This is done via traffic monitoring.
- Guess what hidden service is user using. This is done via website fingerprinting attack.

5.1.6 Defense:

- Obfuscate the special features of IP/RP circuits.
- Destroy the access pattern by sending padding cells.

5.2 Performance and Security Improvements for Tor:

This work provides a survey on how previous research proposals address Tor's design weaknesses.

5.2.1 Previous work:

- Relieve network congestion
- Improve router selection
- Enhance scalability
- Reduce the communication/computational cost of circuit construction and improve its security

Within each of these categories, the paper surveys the literature and compares the available techniques and their advantages and disadvantages in terms of anonymity, deployability and practicality.