ANON paper

Rolf Jagerman, Wendo Sabée, Laurens Versluis, Martijn de Vos

***Abstract*—The abstract goes here. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc et elit nec metus ultrices faucibus. Mauris volutpat libero eu arcu sodales, ac bibendum neque pulvinar. Sed et ipsum nunc. Quisque ac sem ut nunc tincidunt commodo. Cras at odio nibh. Pellentesque sodales ut tortor sed dictum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.**

I. Introduction

I wish you the best of success.

II. Tor design

This section describes

A. Onion routing

B. Directory servers

C. Relay and exit nodes

[gebruikt: the second generation onion router] The Tor network consists of several components. The clients in the Tor network are known as Onion Proxies. The software to run an Onion Proxy is available for free on the Tor website [linkje] and is easy to configure. The Onion Proxies are responsible for downloading the directory information, establish circuits across the network and handle connections from user applications.

The routing in the network is done by Onion Routers, also called relay nodes. The relay nodes relays the data from the Onion Proxy to the webserver across a circuit (circuits are described in 2D). Each Onion Router is connected to every other Onion Router with a TLS connection [link naar TLS protocol]. Each circuit has three type of Onion Routers [link naar download.pdf]:

- The entrance Tor router: this is the router that is directly connected to an Onion Proxy and can observe the origin of a request through the Tor network. The entrance router sends the packet to the middle Tor router.
- The middle Tor router: this router is connected to the entrance router and the exit router.
- The exit Tor router: this router is connected to the webserver. Note that the exit Tor router is the only router that can observe the final destination of the request.

The first router in a circuit is the entrance router. The entrance router sends the data to one of the middle router which forwards the data to the exit router.

D. Circuit creation

As described in the previous section, data on the Tor networks travels over circuits. The data travels over the circuit in fixed-size cells that are 512 bytes long. Each cell has a header and a payload. The header consists of a number that identifies the circuit of the cell and a command that indicates what to do with the cell's payload.

Before a circuit can be established, a path has to be chosen. The current path selection algorithm in Tor selects nodes based on the bandwidth of the nodes [link naar cacr2011-20]. Nodes that have more bandwidth, have a higher probability to be chosen for the circuit setup, however, the same node can't be used more than once in a circuit.

Suppose Alice is an Onion Proxy that wants to connect through the Tor network to a webserver. Circuit setup uses the Diffie-Hellman key-exchange protocol [linkje] to establish a shared secret between nodes. To create a new circuit, Alice first sends a *create* cell with the first half of the Diffie-Hellman handshake (g^b) to the first node in her selected path (for example, OR1). OR1 sends a *created* cell back with the second half of the key (g^b) along with a hash of the final key. Now both Alice and OR1 have a shared key they use to encrypt and decrypt data between them.

So now Alice has a connection with the first Onion Router in the circuit. To extend the circuit to OR2, Alice first sends a *relay extend* cell to OR1. This cell contains the address of the next Onion Router in the circuit and the first half of the key to use in the communication between her and OR2 (g^{a_2}). OR1 takes this first half of the key and sends a *create* cell with this key to OR2. When OR1 receives a created cell, OR1 passes this cell to Alice. Now Alice and OR2 share a common key: $K = g^{a_2 b_2}$. The same procedure can be used to extend the circuit with more nodes.

E. Disadvantages

While Tor guaranteers anonimity, there are some disadvantages using it. The main disadvantage is that the Tor network is slow. According to the Tor Metrics project [link naar tor metrics project], it takes about 6 seconds to download 1 MiB of data.

According to Dingleding et al [link naar 2009-03-11-performance.pdf], there are six reasons why Tor is not optimal. In this section, we will summarize these reasons and explain what could be done to fix them.

First of all, the congestion control does not work well.

The network has some problems handling bulk transfers, such as when downloading large files or streaming high-quality videos. The congestion control could be improved by using an unreliable protocol for links between Tor relays.

Some Tor users put more traffic on the network than they contribute by running a Onion Router. This means that these users are slowing the network down as they use more traffic than giving to the network. A possible solution for this is to throttle certain protocols at exit nodes or at Onion Proxies.

Also, the Tor network doesn't have the capacity to handle all the users that want privacy on the Internet. By increasing the amount of Onion Routing in the Tor network, the capacity is increased. Incentives such as LIRA [link naar lira paper] could make more users run a Onion Router, thus increasing the capacity of the network.

The current path selection algorithm of Tor doesn't distribute the load evenly over the network. The problem is that the current selection strategy is optimal when the network is fully loaded. Using a better path selection algorithm could increase the capacity of the network and the overall user experience.

Another problem is that the Tor clients are not optimal at handling latency and connection failures. For example, if extending a circuit fails, the entire circuit is abandoned. An improvement would be to first try to extend the circuit to some other places. If that fails, the circuit could be abandoned. Also, a better timeout mechanism could be chosen for building circuits.

Much of the overhead of the network is in downloading the directory information. There is also overhead in the TLS connection between the nodes in the network. Removing the empty TLS application record which could reduce the overhead in the TCP/IP header with 6.3% [citaat].

III. Conclusion

The conclusion goes here.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.