Communication in p2pdb:
An Exploration of Tor

David Lauschke
lauschk2
CS460: Security Laboratory
Spring 2015

Evaluation of Tor:

Tor, or the onion routing network, is run by a non-profit organization named the Tor Project. Tor is built upon the SOCKS proxy protocol, and anonymizes client communications through at least three routing nodes, encrypting communication packets with an additional layer of protection through each hop. The points of weakness are the entry and exit nodes; however, the entry node only knows that it has received encrypted communication from your IP address, or that "This IP address is using Tor." Since Tor is not illegal, there is no harm done in this. The exit node knows of course the destination and has the possibility of seeing unencrypted traffic data sent to the destination. However, if you are providing an additional layer of encryption then the node cannot see into the encrypted data stream.

I downloaded the Tor bundle and started the Tor Browser.

Configuration of the hidden service

Torrc file: located only after finding the Tor Browser in Finder, then right-clicking and selecting "Show Package Contents" (Macs r great.) The torrc file in this package wasn't the normal that Tor has for an example, so I modified it myself following the hidden service tutorial on Tor Project. After I changed this, the Tor Browser crashed. After searching for more information on the subject to no avail, I had to look for another alternative. The solution was to download the tor tarball directly and compile from source so that I had the normal torrc file. This allowed me to modify the torrc file such that my Tor Browser was still functional, and my hidden service could be operated by starting Tor from the command line.

Whenever a hidden service is activated through the torrc file, Tor automatically assigns a new hostname and private_key to the hidden service. This hostname is supposed to be the unique identifier of each service, derived from the generated private key. Hidden service functionality is activated by uncommenting the "HiddenServiceDir" line and providing a valid directory with the correct permissions. Next, the "HiddenServicePort" line can also be modified. The first argument is a virtual port to set up, to which Tor clients connect when visiting your site. The second argument is the actual port which hosts the service to which you wish them to connect (usually hosted locally in order to be invisible outside of the host).

I successfully completed my configuration of the hidden service by creating a virtual port 5000 to which I connected the p2pdb port (127.0.0.1:5000).

On the duplication of private keys in the HiddenServiceDir:

During my work, the question of "what would happen if a service duplicated its secret private_key" arose. So, I set up an AWS Linux AMI server and pursued an answer. After

modifying my configuration files to accommodate my new hidden service on the AWS distribution, I was auto-assigned my hostname and private_key. As I said earlier, this hostname is derived from your private key. Whenever I replaced the private_key file with the file located on my machine, Tor changed the hostname of the AWS hidden service to the exact same random 16 letter string that was assigned to my machine.

Conclusively, the uniqueness of the hostnames for hidden services seem to be rooted in the uniqueness of private keys assigned to hidden services. A host of a hidden service, once published, sets up a set of introduction points across the Tor network. It then sends a descriptor to a distributed hash table containing its introduction points and public key, signed by its private key. In order to access a hidden service, a Tor client requests information from the hash table using the unique descriptor and creates a rendezvous point with which to communicate with the service. The client then utilizes one of the introduction points to negotiate the tunnel through the rendezvous point to the hidden service and begin utilizing it. From this understanding of the Tor network, I believe that if a private key is compromised, it creates a fight in the distributed hash table. From what I have read, it seems that the most recently uploaded descriptor is the one utilized.

Client Implementation:

As part of my project, I also created a relatively easy-to-use client with which a user can connect to a database and store named table and recipient keys. The client service is started with

```
python client.py
```

and is pretty straightforward from there onwards.

The “list” function lists the existing stored table or recipient names in the keyring, with the option of actually viewing the key to distribute to your intended users.

The “add” function allows users to add keys to their keyring with the given table_name or recipient_name.

The “put” function attempts to use the given table and recipient names to find the respective keys in the user’s keyring, then use those keys to encrypt and send the specified message to the p2pdb. If no keys exist for either the table or recipient, the client program will create the keys and output them for distribution to the intended users.

The “get” function attempts to gather the union of all messages from the given parameters (union functionality through server, not client). If a recipient is specified, the client program will attempt to decrypt the messages with the recipient key.

If you don’t get the “exit” command, don’t utilize this service.

<u>Current weaknesses of client implementation</u>:

- VERY UNSECURE KEY STORAGE!
- Seriously don’t use this to actually whistleblow on the government, private keys are stored in plaintext on disk.
- Can only retrieve messages for one table|recipient at a time
- Table and recipient names cannot contain spaces

Additional Guidelines:

Tor is only as private as you are. If you've made connections over unsecure channels, this browsing history can be used to identify you on the Tor network. Any sensitive communication passed over those unsecure channels can also be of course used against you. Overall, be conscious about what you are broadcasting when using the internet. This invention was not created with security in mind.