Paris Hom 862062330

Brandon Cai 862080765

CS165

Fall 2020

12/10/2020

CS165 Final Project Readme

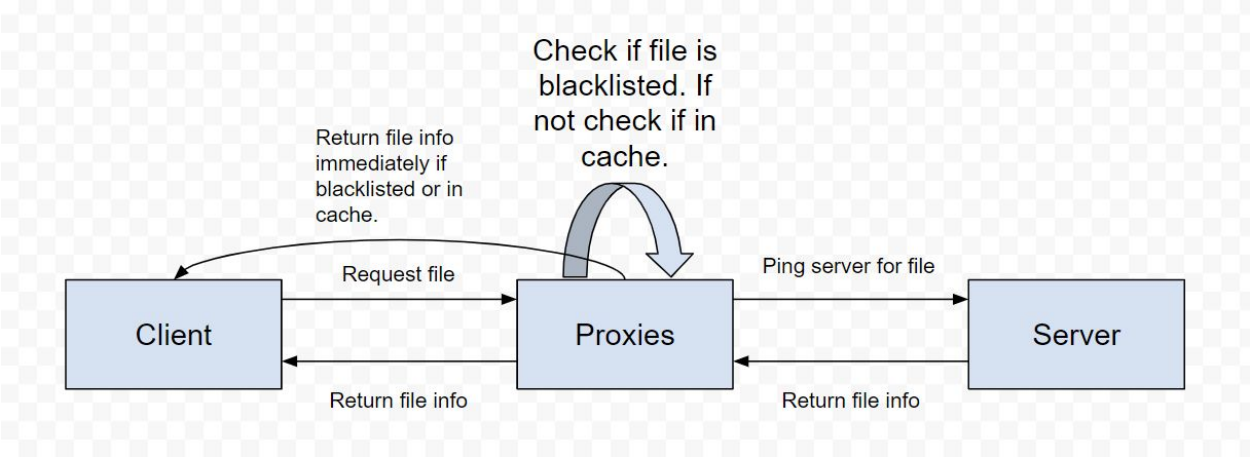
Roles

Paris was in charge of selecting proxies using rendezvous hashing in the client.
Paris was also in charge of the bloom filter implementation. This included reading the forbidden objects from a file and mapping each file to their respective proxy. As well as identifying forbidden objects using the bloom filter and logic for when to check the cache.
Paris was also in charge of implementing the proxy cache and multiple (5) proxies.

Brandon was in charge of TLS implementation. This included creating the TLS sockets and connections as well as verifying TLS handshakes between clients, proxy, and server. He was also responsible for file transfers between entities using TLS.

Details and Assumptions about Project

At a high level the system interacts in a simple manner.



At every interaction between entities, there is a TLS handshake in order to ensure that the connection is secured with TLS. Information is then sent over a secure TLS connection. The client will only interact with the proxy, we can see this as a security feature as to not allow direct interaction with the server from a random client.

The proxy will only ping the server for information if the file is not blacklisted and not found in the cache, otherwise the proxy will immediately send the file content back to the client if it is found within the cache or a message saying that the file is blacklisted if the file is found within the bloom filter. Files contents returned from the server are stored within a cache inside the proxies for quick retrieval the next time the file is requested.

We assume that the server port and proxy ports will be static as it's something on the backend that is set up and left alone. For simplicity, the server uses port 9998 and proxies use port 9990-9995. We assume that the client will decide which port to connect

to via rendezvous hashing . This is so that the user does not choose a server port that is already being used by the 5 proxies.

We assume that when the proxy needs to ping the server, that the required certificates will be present already so the handshake is simple.

Running and testing the project

Setting up the project

1. Have file ‘blacklisted.txt’ in ‘proxy’ folder and ‘files.txt’ in ‘server’ folder
2. In the ‘build/src/’ folder, start up the three parties
    a. Start up the server using ‘./server’
    b. Start up proxies using ‘./proxy’
    c. Start up client using ‘./client <fileName>
3. You should be able to see ‘fileName: content’ in the client side if the request was accepted
    a. If the request was denied (blacklisted or not available), you should see ‘Access Denied’

Example file names to test

Note: These valid and blacklisted files are listed within files.txt and blacklistedfiles.txt respectively.

Valid files: text1.txt ; skeleton.txt ; witcher.txt ; catherine.txt

Blacklisted file: blacklistedfile.txt

Example command line: (within /build/src/) use

./client text1.txt

To test cache

Clients can request the same file twice in a row. On the second run you should see a message indicating that the file was found in the cache. This is because whenever a file is not found in cache, it is requested from the server and then stored into a local cache within the proxy.

References

https://github.com/bob-beck/libtls/blob/master/TUTORIAL.md

https://man.openbsd.org/tls_init.3