## 1. Abstract

EZShare is a distributed file sharing system on the basis of client-server architecture. Users can share URI or files with servers, fetch files from server and provide a list of available servers for any server. The system allows users to query from more than one single server per connection.
The challenges of security arise as a result of the need to share or to distribute resources. To achieve this challenge, we implement a secure socket between Client and a subscribe function to establish an asynchronous and consistent connection.
In this report, we will firstly talk about security policy (mechanism) we used, then discuss three security threats have been solved and four issues that still remain. Secondly, explain how we implement the relay function between servers and compare persistent asynchronous connections with short synchronous connection.

## 2. Security policies

### 2.1 SSL implementation

We were asked to implement security without being given a security policy, so we chose to use Secure Socket Layer (SSL). There are 5 steps to achieve message authentication (Lamprecht and Moorsel, 2007):
1 Generate two jks (both on server and client side)
2 Create a CSR and send it to a trusted third-party Certificate Authority (Unimelb)
3 Get the certificates signed by Unimelb and import them into KeyStore
4 The client and the server will check each other's keys to verify the identity
5 Handshake and negotiate an encryption method to communicate

### 2.2 Problems been handled

Using the security mechanism (SSL) can solve the security issues to some extent. Here are some security threads are well handled:

#### 2.2.1 Leakage and Tampering

With the use of the public/private key and the certificate, server and client must first confirm the other parties to establish a connection. Further more, all data has been encrypted and sent in the secured channel. Thus the acquisition of information cannot be the unauthorized recipients. Similarly, the unauthorized one cannot alter the information. So the system eliminated the risk of information leakage and tampering.

#### 2.2.2 Eavesdropping and Message tampering

A short-term session key is randomly generated by a hash function whenever a channel is established and different keys are used for different connections. Although, keys are somehow open to public, using RSA to exchange keys and to replace them frequently can solve the eavesdropping and message tampering issues.

#### 2.2.3 Relaying

SSL uses the serial number to protect the sender from replay attacks. This serial number is encrypted as the load of the packet. In the entire SSL handshake, there is a unique random number to mark the SSL handshake, so that the playback can take nothing. This serial number also prevents the attacker from recording packets and sending them in a different order.

### 2.3 Issue remain

However, there are still many other security threats that could not be addressed by the implemented SSL.

#### 2.3.1 Information leakage (analysis data flow)

Since SSL protocol only protects the data and ignores IP header and TCP header, it is easy to know the operation of a system and its output by checking for unencrypted IP addresses and TCP port numbers. Then there is the potential for information leakage. However, exposing business secrets and personal relationships are not a big deal for most of the people, so we do not plan to handle this security threats.

#### 2.3.2 Denial of service (DoS)

During the SSL handshake process, in the negotiation encryption algorithm where the server CPU overhead is about 15 times the client overhead. An attacker exploited this feature to quickly reconsider a TCP connection (which is allowed by SSL) to exhaust server CPU resources, called SSL-DoS. If multiple zombie hosts initiate SSL-DoS to the server, it is an SSL-DDoS attack.

#### 2.3.3 Masquerading

Because the SSL protocol is designed to protect the Web site and online transactions, SSL protocol is not the default requirements for client authentication. Which means sending or receiving messages using the identity of another principal without their authority (Park, Se-Won, 2014). To solve this problem, it can configure the SSL protocol when necessary to choose the authentication of the client authentication.

#### 2.3.4 Untrusted certificate

According to Certificate chains, we got a root certificate from Unimelb and in general all its subsidiaries should also be trusted. But the subsidiaries and root one should be treat in different secure level.

### 2.4 Other security mechanisms to improve applicability

Now the ‘owner’ and the ‘channel’ information are sensitive in order to protect the data. However, if a client forgets the ‘owner’ then it cannot update the resource. Moreover, the scalability of the system is limited, because the ‘owner’ information is not relayed between servers, nor is it shown in the results of user queries. There are two basic ways to improve the system security, symmetric algorithm and asymmetric algorithm. We use asymmetric algorithm.

First, user generates a private key to protect ‘owner’ and ‘channel’. Second, client will send the public key to server then the server will encrypt the ‘owner’ using public key. Third, both public key and resource will be passed to other servers while the relay is set as true. Finally, the true owner who has the private key can decrypt and see the ‘owner’ and ‘channel’ information of the resources. In this way, the information of ‘owner’ is protected and relayed.

The reasons why we use asymmetric algorithm to encrypt are:

1 For symmetric encryption, both sides use the same secret key, if one side of the key is leaked, then the entire communication will be cracked (ZHOU and HUANG, 2013). For asymmetric algorithm, it has a pair of secret keys, the public key is public for encryption and the secret key is stored by itself for decryption.

2 It is more suitable for a small amount of data encryption. Although the processing of encryption and decryption takes a long time, the only data need to be encrypted is ‘owner’.

## 3. Subscripted Relay

### 3.1 Implementation

When client send SUBSCRIBE and set relay as ‘true’, server should relay a subscription to other servers. We choose to implement in a periodical asynchronous way. There are 6 steps to achieve:

1 Client send SUBSCRIBE and set relay as ‘true’

2 Create an array to store the resources already existed on server

3 Server send SUBSCRIBE to destination server with relay set as false
4 Create new threads and initiate persistent connections
5 The destination server sends back the resources when it receives PUBLISH or SHARE resources
6 Compare with the array and send new different resources back to client

**3.2 Compare and analysis**

Persistent asynchronous connections are used for frequent, point-to-point communication, and the number of connections cannot be too much (Wu, 1999). Each connection requires a three-step handshake, which takes time. If for each time, we have to connect first and then communicate; the speed will be reduced a lot. So the persistent connections are more suitable for high frequency communication. The short connection with frequent communication will cause the socket error, and frequent socket creation is also a waste of resources.
Second, the persistent connections will perform better, if the numbers of client are in a small scale. For example, all clients have persistent connections servers, and then a lot of resources will be occupied on server, such as bandwidth.
Third, this implementation is weak at individual failure. For example, if a middle server is broken or lost connection, then the client cannot subscribe any more.
Forth, malicious connection may happen. Since the server does not automatically close the connection from the client, some clients may maliciously connect to the server for a long time and take up server resources.
At last, the client can receive the subscripted content immediately, because once a resource has been PULISH or SHARE on destination server, it will send back straightaway through local server without any delay.

**3.3 Improve the mechanism**

**3.3.1 Implementation**

The improved mechanism is establishing the short synchronous connections periodically. The implementation has 4 steps:
1 Client send SUBSCRIBE and set relay as 'true'
2 The server will establish a synchronous connection with destination server and send the request periodically, like 10 minutes
3 Destination server will send back all the PUBLISH and SHARE resources gathering in last period (10 minutes)
4 Close connection and send back the resources to client

**3.3.2 Pros and Cons**

There are several advantages for short synchronous connection:
1 Scalability
We pretend that one server can have 100 threads at the same time and if we use persistent connection, the thread pool will be full quickly and other clients cannot connect anymore. With the short connection, clients are not occupying the server when they are not subscribing, so the threads can hardly be full.
2 Consistency
The consistency of the resources depends on the period time setting. If it is a long time, like 1 day, the resources update and send back to client will have a high delay. However, if the time set to small, like 1 second, then send request frequently will also consume the resources on server.

## 4 Conclusions

In conclusion, the report discussed about two main aspects: to what extent the security policy solves the threats and how to implement subscribe relay function.

## 5 References

[1] Lamprecht, C. and Moorsel, A. (2007). Adaptive SSL. 1st ed. Newcastle upon Tyne: University of Newcastle upon Tyne, Computing

Science.

[2] Park, Se-Won (2014). A Study on Examination Standard of ISBP745 - Certificate of Origin, Beneficiary's Certificate and Inspection Certificate -. KOREA INTERNATIONAL COMMERCIAL REVIEW, 29(1), pp.27-44.

[3] ZHOU, Q. and HUANG, D. (2013). Encryption algorithm for QR code based on Ising model. Journal of Computer Applications, 33(10), pp.2861-2864.

[4] Wu, J. (1999). Distributed system design. 1st ed. Boca Raton [etc.]: CRC Press.