Threat Model

T8 DDOS

An authorized client who has an account on the system with a valid password attempts to bombard the Group Server with multiple connection requests. Assuming either this is on purpose by the client or the client's account got compromised. This single client can connect from multiple computers with the same account at the same time to bombard connection requests.

Attack

The client has created their own Client.java implementation where the client makes use of the connect() method from GroupClient.java in this alternate client application. The client connects and performs their own Diffie Hellman exchange to create a symmetric key. The attack follows by repeatedly performing connect() method to spam Diffie Hellman exchanges. Using the key generated from Diffie Hellman, the attacker encrypts a single message and sends it to the Group Server. The Group Server then decrypts the message and looks to handle the request sent, however the request isn't a valid one, resulting in an infinite loop upon that thread. If enough attackers spam these requests, the system running the Group Server would get slow enough to possibly impede other clients' requests and even enough to crash the server. This was proven when running a Group Server with multiple attacks resulting in an increase in CPU usage that is correlated with the number of systems used in the attack. In other words, each new system added to the attack resulted in an increase in a percentage of CPU usage on the Group Server system.

Countermeasure

To handle the attack, multiple connections from the same IP address shouldn't be allowed. To ensure this is held true, after the initial connection request, the IP address is stored within the Group Server. Per each connection, the IP address is checked with the stored IP addresses. If one of the stored IP addresses matches the connection's IP address, we reject the connection. After the initial connection is disconnected, that connection's IP address is removed from the Group Server's stored list of addresses. Should they wish to reconnect in the future, their IP address wouldn't be found in the stored list, so the server would approve of the connection. In addition, if any message besides the pre-approved messages are sent from the client to the server, the connection is immediately disconnected. This is because any message that isn't pre-approved is being sent from a driver that we didn't create, which shouldn't be trusted and accepted.

Justification

Our countermeasure works because a single client attempting to do multiple connections with the same IP address will be sent to the Group Server multiple times. If the Group Server can check to make sure that same IP address isn't received multiple times, it can protect against this attack. While this attack works with an attacker connecting from a reasonable number of systems, if an attacker can get ahold of an absurdly large amount of systems, they can still theoretically crash the Group Server. This however, is not too much of an issue because the

number of systems required is assumed to be absurdly large. Our countermeasure ensures that a single IP address cannot connect multiple times until the first connection from that address is disconnected. Our method of countermeasure was done in place of putting a limited number of connections allowed because putting a connection limit could potentially hinder everyday work usage of the file sharing system during busy times.

Experiment Results

Before Solution Implementation:

Number of Attackers	CPU	Memory
0	0	62.3 MiB
1	10	94.4 MiB
2	17	134.2 MiB
3	25	200.7 MiB
4	34	195.5 MiB

After Solution Implementation:

Number of Attackers	CPU	Memory
0	0	62.7 MiB
1	0	82.6 MiB
2	0	101.5
3	0	140.9 MiB
4	0	138.4 MiB

While having 4 attackers attempt to DDOS the server, we had another user attempt to connect and perform some tasks. This valid user was successfully able to get to the server and still do his work even with the attacks in progress.