Highly Dependable Systems

Highly Dependable Location Tracker

2nd Semester 2020/2021

Group 15:
Francisco Silva, nº 98848
Guilherme Cardoso, nº 98495
Tiago Domingues, nº 88045

May 21st, 2021

System Design

The system was configured to support 4 servers and 10 clients. From those, up to 1 server and 3 clients can be byzantine, in order to satisfy the following Byzantine Fault Tolerance formula:

$$N - f > \frac{N + f}{2}$$

When a user first joins the system, the client application interacts with all servers so they can exchange a symmetric key, which is going to be used to encrypt all data in future messages. To guarantee confidentiality in the key exchange process, asymmetric cryptography is used. To avoid the overutilization of symmetric keys, all servers keep track of how many times each users' key has been used and, upon exceeding a configured threshold (10), the server generates a new key for the user. Also, if a server crashes, it generates a new symmetric key for each user.

These symmetric keys are only kept in memory in order to safeguard the system in case of a user or a server machine gets compromised. Regarding asymmetric keys, these are properly stored in a KeyStore.

To ensure authenticity and integrity in the system, all messages exchanged between users and servers have a digital signature. When a client sends a message to a server, it signs the message payload with its private key. Upon receiving the message, the server is going to check the signature with that user's public key. Therefore, we guarantee that:

- The message was not altered by a MITM. Otherwise, the MITM would have to regenerate a valid digital signature and, without the user's private key, it is not possible.
- The message was sent by that user and no other. Again, for an attacker to be able to spoof the origin of the message, he would need to have the victim's private key.

To ensure freshness, every message has a counter. This counter is pseudo-randomly generated by each client during the process of exchanging a symmetric key with all servers. Every time the user and the server send each other a message, the counter is incremented. Then, each of them verifies if the counter received is higher than the last sent counter. If it is, the message is accepted. If it is not, then there is a high probability that the received message is an old message being retransmitted, probably by an attacker, and it is rejected. This counter is regenerated every time there is a key exchange in order to avoid its overutilization.

For a server to accept a location report from a user, the report should have at least 3 valid location proofs because that is the only way to ensure that a location is correct, since there can be up to 2 byzantine users near another user.

Possible threats and Protection mechanisms

The possible threats to the system are the following:

1. Creating a proof or report on behalf of another user;
2. Creating a false proof or report of its own location;
3. Obtaining information on other users from the servers;
4. MITM attacks;
5. Replay attacks.

And the corresponding protection mechanisms are as follows:

1. An attacker cannot sign a proof nor send a report on behalf of another user because he does not have his private key.
2. Attackers cannot create their own proof of location because servers do not accept self-signed proofs and cannot create a false report because they need at least 3 different proofs.
3. An attacker may send a spoofed request to obtain info on a victim, but it does not have he's private key, so it cannot sign the message properly.
4. A MITM can intercept messages between users and servers but those are always encrypted, so he cannot see the content. Plus, if the MITM arbitrarily changes the content, the digital signature will not be valid, as the MITM cannot produce a valid signature without the victim's private key.
5. Replay attacks cannot be performed because, as already discussed, both users and servers use an incremental counter (protected by a digital signature) in their messages. Plus, servers do not allow location reports to be overwritten.

Servers may also be exposed to DoS attacks. Therefore, it was introduced a mechanism to combat these attacks with the objective of forcing users to execute computationally expensive operations in order to produce valid requests. It was implemented a Proof-of-Work approach, where clients need to find a nonce that, together with the message, produces a hash that starts with 4 zeros. This mechanism was applied to the *obtainLocationReport* and *requestMyProofs* operations, so that it is computationally expensive for attackers to try to overload servers with these messages. The *submitLocationReport* operation does not need this mechanism, since servers only process 1 message from each client per epoch.

Other guarantees

In case of a server crash, the system remains functional as users will keep communicating with the rest of the servers. The crashed server may later be relaunched and recover its previous state. If users sent reports to the remaining servers during the crash, this server may eventually receive these reports when the user or the HA client make a reading operation on those reports, via a write-back.

The system also guarantees non-repudiation on the location reports and proofs. Regarding reports, servers store the users' digital signature over the entire report. Servers also store all proofs of each report and these guarantee this property themselves, given that they consist of a digital signature over the user's location. Therefore, none of these 2 can ever be denied by the author of the digital signature.

If a byzantine user sends fake reports only to a byzantine server in order to trick the HDLT, that server might accept those reports. However, the system will not deem them valid because for a read operation to be successful, the reader needs to receive data from at least 3 servers. The final value will correspond to the most common answer.