INSTITUTO SUPERIOR TÉCNICO

HIGHLY DEPENDABLE SYSTEMS

SISTEMAS DE ELEVADA CONFIABILIDADE

MEIC

Project 3: Block server with intrusion tolerant replication
Report

Group: 9

2015/2016

Ricardo Filipe Fonseca Silva
Rui Filipe do Rosário Galão Ribeiro
João Daniel Jorge Machado

1 Introduction

For the third and final stage of this course's project, we were asked to further improve the file server, by adding a replication protocol that was robust against intrusions that could cause the server replicas to behave in a byzantine manner.

In this report, we will go over the design chosen for the replication protocol, as well as all the modifications done to the base algorithm, and the rationale behind them. We will also cover any dependability guarantees the new protocol brings, as well as its limitations.

2 Overview

For the final stage of the project, we chose to implement a simple 1-to-N authenticated data protocol, that supports byzantine behaviors from the N block server replicas.

The client of the block server becomes a client of the protocol, and through it, issues reads/write requests to the file system. These requests are sent to all replicas, and once a a sufficient number of valid replies have been received, the request process is finalized (for writes, the value is stored by all servers, and for reads, the value is returned to the client).

The authenticity of the data is checked via the client's signature, which he uses to sight the public key blocks when issuing a write request.

3 Design Choices

The protocol requires each write request to a file to be signed by the entity that issues it. Since it is still assumed that only one writer exists per file (the owner of that file), he is the only one that is required to sign the write requests. To prevent replay attacks, each request is accompanied by a timestamp. The protocol guarantees that only fresh write requests are accepted.

During a read request, a file's public key block is returned by the block replicas, and the signature is verified. If it matches the signature of the owner, then the public key block is guaranteed to be valid. Once a majority of replies from the block servers is achieved, the information contained within the public key block is used to fetch the content blocks belonging to the file.

As far as the communication channel between the client and the block server replicas is concerned, while the protocol suggested the use of authenticated perfect point-to-point links, a compromise was made to use a TCP channel instead, which offers similar reliability. This compromise was greatly due to time constrains faced during development.

4 Dependability

The final implementation brings much higher Availability and Reliability to the file system. Assuming a real world scenario, as long as a sufficient number of block server replicas (enough to reach a majority vote during read/write requests) are functioning in a correct manner, then the file system is available to clients. Increasing the number of replicas would increase the number of supported byzantine nodes.

The transport layer lacks integrity guarantees as it stands. An authenticated point-to-point link protocol would be best, with MACs to thwart the forging of messages, and ACK/Timeout mechanisms to improve reliability.

5 Conclusion

The final implementation of the file server offers much better reliability and availability guarantees, via the use of replication on the block servers.

It also offers a way to tolerate byzantine behaviors from a set number of replicas, by making sure the results of the read/write requests are all obtained from a majority voting, guaranteeing a correct output, if enough correct replicas are available.

Reliability of the communication between the client and the replicas is assured by the use of a TCP protocol. For future work, it might be more adequate to replace this with a point-to-point link protocol, using MACs and ACK/Timeouts.