CS 6238: Secure Computer Systems

Project 2: Simple Distributed File System

Team Members

Rohan Tahiliani (rtahiliani3)

Steps to Complete Project

Design, Implementation along with Bonus credit part (CRLs), Testing, Performance Evaluation.

Project Details

Project has been implemented completely in C, using the OpenSSL library on a 64-bit machine with Ubuntu 10.04. There are three components to the server:

1. Certificate Authority (CA)
2. SDFS Server
3. SDFS Client

Protocol Description

Certificate Authority

The CA generates certificates for the client and server based on the common name provided as well as other information like the user's country, state, city and organization. This certificate along with the private key is written to two separate files which are sent offline to the user for security reasons. The CA generates a 1024 bit RSA key which is used to generate the certificate. The certificate is then signed by the CA. The program used for certificate generation is 'makecert'.

SDFS Server

The server handles client requests after authentication using SSL. It verifies the validity of the certificate provided by the client and also checks the Certificate Revocation List to see if the client certificate has been revoked. If the certificate fails the validity test, the server responds with a “Revoked” message and tears down the connection. The SSL connection uses the public keys to establish a session key. Key exchange takes place using Diffie-Hellman.

If the server's certificate has been revoked, it handles this by accepting an extra argument which will be the old private key of the server. The server will decrypt the file keys using this session key and then encrypt them again using the new public key.

If the client passes authentication then the server will handle the following requests:

PUT – Store file to FileSystem/files. Verify if file exists and user has write permission, and update meta-data information. Encrypt file data using SHA1 of file content as key, using AES-256 and then encrypt the SHA1 key using RSA with the server's public key.

GET – Fetches files stored in the folder FileSystem/files. Verifies the access control for the user based on the metadata for the file stored in FileSystem/metadata. If the user passes, the key is decrypted using

RSA with the server's private key. The SHA1 key obtained is used to decrypt the file content with AES-256 and then the file is transferred.

DELEGATE(*) - Delegates access to another user if the the current user has the required access rights. Updates the meta-data information and records the time for which the user will be granted access. If a user has received Delegate*, then the user can only delegate permissions for the file up to the time ticket that the user has. If everything is in place the meta-data file is updated.

SDFS Client

The client makes a connection to the server using the certificate and private key provided. It verifies that the server's certificate is valid and has not been revoked. If the server's certificate does not pass validation then the Client sends an END message to terminate the connection.

Use the START and END commands to Start and End a session with the server.

For the PUT request, the client can provide any file name along with a fully qualified UNIX path. The file name should be unique as compared to any file created by any other user. If the file is not found on the client machine, a File Not Found error is shown. If the user does not have permission to write to the file, a Permission Denied message is sent by the server.

For the GET request, the file name is checked to see if it exists at the server. The server could reply with a File not found error, or a Permission Denied error. If access is granted the file content is sent to the client and stored in the 'files' folder at the client system.

For the DELEGATE(*) request, the server will reply with either a Permission Denied error, File Not Found error, or Delegate Successful message.

Performance Evaluation

The connection set up time is in milliseconds which includes establishing the socket connection, exchanging keys, establishing session key and verifying certificates.

The time required to transfer files is also very short but can vary from a few milliseconds to seconds depending on the size of the file being transferred. This includes cost of RSA encryption, decryption of the AES key as well as encryption, decryption of the file contents and transfer.

Other communication is almost instantaneous since the client and server are on the same machine.

Flawfinder Results

After running flawfinder on the code I saw that the errors reported by it were in functions like strcat, strncat, strcpy, sprintf, sscanf, strlen for variable length buffers which could result in a buffer overflow. These functions have only been used in places where the buffer size is known before hand so it is not a real vulnerability.

Functions like getpass and fopen have been reported by flawfinder to prevent password leaks and compromising confidential files. Since there is no alternative to these functions and these vulnerabilities exist in every application using these functions (many UNIX utilities) nothing can be done about this.