ECE422 Security File System Project

March 13, 2021

Hao Xuan 1537913

Maocheng Shi 1538829

Abstract

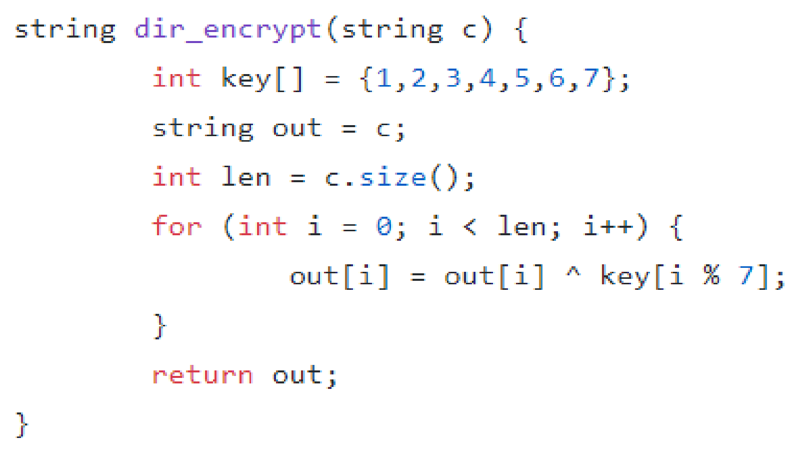
This project is a practice for using cryptography we learnt from the course to build a secure file system. In the secure file system developed in this project, users can store their data in on an untrusted server safely. The file system is established using client-server architecture and multiple encryption method is used to ensure the user data is confidential and not been maliciously modified by the other users in the untrusted server.
The operating logic of the secure file system is like the UNIX file system, users can use a series of UNIX like commands to create user groups, read or write files in the directories.

Introduction

In many situations, the servers in the public net can not be fully trusted for that hackers or malicious users may have set up the backdoor program on the server and all the files stored on it might be breached. One of the major topics to be investigated in security system design is to build a system that can protect user files in untrusted servers. In this project, a collection of cryptography methods is used to develop a secure file system which can keep the user files confidential in the untrusted server. The main paradigm for the secure file system is to use asymmetric encryption to encrypt the files when communicating between the client and the server and avoid storing the files and account information using plain text.

Technical Details and Methodologies

This project used client-server architecture, in the client side the user inputs are split and a socket which connects the server, and the client is established. In the server side, the user inputs are first processed, if the user command is to create an account, the server will first go through all the existing accounts to make sure there is no duplicate usernames, then instead of using plain text to store the passwords, the secure file system uses a hash function to encrypt the data and then store it. Figure1 shows the implementation of the hash function used in this project



```
string dir_encrypt(string c) {
        int key[] = {1,2,3,4,5,6,7};
        string out = c;
        int len = c.size();
        for (int i = 0; i < len; i++) {
                out[i] = out[i] ^ key[i % 7];
        }
        return out;
}
```

Figure 1, Implementation of the Hash Function

When the users try to login into his account, the server first try to find the input username in saved accounts, and then compare the hashed input password to the corresponding password of the username. To maintain the integrity of the user files and detect any potential malicious modification of the files, the secure file system uses BSD check sum algorithm to check the files every time a user logs into his account or log out his account, if the system detects that the checksum is inconsistent the user will get notified. The BSD checksum algorithm computes the check sum by adding up all bytes in the input data with the checksum accumulator rotated to the right by one bit at each step before a new char is added. If the file is maliciously modified, the changed file content will result an inconsistent checksum.

The files in the secure file system are encrypted using AES256 algorithm, so even the malicious users find the files in the secure files system, he cannot read the content of the files. The implementation of the AES256 algorithm is by using the functions built in OpenSSL library. AES is based on a design principle known as a substitution-permutation network and is efficient in software. To achieve the highest secure level, our project uses AES256 algorithm which performs 14 rounds of transformation for 256 keys.

The files in the secure file system are managed like in the UNIX system. When a user account is created, a directory is named by the username is created in the server directory at the same time. This directory is used at the home directory for its owner and is invisible to the other users unless they are added to the same group. The UNIX file commands are also applied to this system, users can do the operations including creating a directory, read or write on a file and delete the files.

Design Artifacts and Explanations

There are to separate folders in the design folder. The folder named “client” contains the all the files necessary to build the client of the secure file system while the folder named “server” contains the files for building the server-side program.

In the folder named “server”, the file named “cmd_line.cpp” contains all the functions that process the input commands. In the file “directory.cpp” contains the functions for getting the directories and hide the directory does not belong to the user. It also includes the functions for finding the contents in a directory and a function for changing the directory name. The file “security.cpp” has a function for encrypting and decrypting strings using our customized hash function. It also has the functions for calculating the BSD sum of the files and verifying the BSD checksum. The encryption of the file contents also relies on “security.cpp”, the function named “file_encrypt()” use the AES256 algorithm to encrypt the files in the secure file system. In the file named “utilities.cpp” the function for calculating BSD checksum is implemented and it also contains the functions for doing string processing and get the members in a user group

In the folder named client, it contains a file named “client.cpp” which creates the socket connection and handling user inputs. In the file named “login.cpp” it contains the functions for authenticate the users. In this file there is also a function save the groups and update the group info in the server side. In the function named “user_sign_up()” the password is hashed and sent to the server. The file named “utilities.cpp” in client folder is similar to that in server folder, they both implement some functions for handling the input strings.

Deployment Instructions

The system consists of two parts, client and server, which can be placed anywhere with file read/write permission. However, to compile both parts, openssl library must be installed.
The following command needed to be executed to install the development package.

```
sudo apt-get install libssl-dev
```

After successfully installing the package, change directory to server and client package, and execute make command separately.

```
make
```

The server program needs to run first before any client executes.

```
./server (ip address) (port number)
```

Client program can be executed in the same way.

```
./client (ip address) (port number)
```

Server and client must have the same ip address and port number to communicate successfully.

User Guide for Secure File System

To user the Secure File System, user needs to correctly login.
In the entering page, user can select for group operation or user operation. In group operation page, user can view the existing group or create a new group. In user operation page, user can login or sign up. Sign up operation requires user to create a non-existing user ID with a password and an existing group, and login needs correct user ID and password input.
After successful authentication, user can enter the SFS page. In here, user can read, write, and edit files. The following commands are supported by the system:

ls: list all file and directory under current directory

cd <directory>: change to the target directory

mkdir <directory_name>: create new directory under current directory

*rm <name>: delete a file or directory. (**delete a directory will also delete everything inside**)*

touch <file_name>: create an empty file with the given name

cat <file_name>: read the given file content and display it

echo "<content>" <file_name>: append the given content to the end of the given file. (If the file doesn't exist, it will create an empty file with its name and write into it)

mv <old_name/location> <new_name/location>: move the given file or *directory to a new given location. This can also be used to change file or directory name under current directory.*

logout: logout current account. This command will take user back to entering page

exit: exit client. Logout account and exit client process.

All space inside command must be applied correctly in order to successfully execute the command.

Conclusion

In this work, we have developed a secure file system which allow the users to keep their files confidential in an untrusted server. Several kinds of cryptographic algorithms are used in this project to achieve a high security level. The password of the users is encrypted using MD5 algorithm, the files saved in the server is encrypted using the AES256 algorithm. BSD checksum is used in to check the integrity of the file and detect malicious modifications on the file. The requirements specified in the project descriptions are met.

Reference

Wikipedia contributors. (2021, February 3). BSD checksum. In *Wikipedia, The Free Encyclopedia*. Retrieved 13:12, March 14, 2021, from https://en.wikipedia.org/w/index.php?title=BSD_checksum&oldid=1004562577

Wikipedia contributors. (2021, February 27). MD5. In *Wikipedia, The Free Encyclopedia*. Retrieved 13:14, March 14, 2021, from https://en.wikipedia.org/w/index.php?title=MD5&oldid=1009226881

Wikipedia contributors. (2021, March 12). OpenSSL. In *Wikipedia, The Free Encyclopedia*. Retrieved 13:15, March 14, 2021, from https://en.wikipedia.org/w/index.php?title=OpenSSL&oldid=1011694654