Design Documentation
Security W4181 - Final Project
Team Gamma

Team Members:

Zach Chen, Justin Kim, Sarah Seidman, Jiayang Zhou

Basic Procedure

Server

- When client sends a **getcert** request, validate the username and password from the password file. If it is a valid combination, the server will use the intermediate CA to sign the client's CSR. The server will then write the certificate to mailbox/user/certs/<user>.cert.pem, and will send it back to the client. On success, a 200 OK response is sent. On failure, a 500 response is sent.
- When client **changepw**, validate the username and password from the password file. If it is a valid combination, update the password (hashing and salting again), sign the client's CSR, write the certificate to mailbox/user/certs/<user>.cert.pem, and send it back to the client. On success, a 200 response is sent; on failure, a 400 or 500 response.
- When the server receives a request to the /**sendmsg** endpoint, it validates the client certificate, and if it passes, checks the existence of the needed certificates and sends them back. 200 on success, 400 on failure.
- The second half of **sendmsg** is sent to the **/upload** endpoint. The server checks the validity of the recipients and delivers the encrypted messages to their mailboxes.
- When the server receives a request to the /**recvmsg** endpoint, it validates the client certificate, and if it passes, extracts a message sent to the client from the corresponding directory, sends it back and deletes this message from the mailbox. 200 on success, 400 on failure.

Client

- **getcert**: send username and password as well as a certificate signing request created with the client's private key. Username, password, and path to private key are prompted for. The client then receives a certificate from the server and writes it to <user>.cert.pem in the calling directory.
- **changepw:** send username, password, a new password, and a certificate signing request created with the client's private key. Username, password, new password, and path to private key are prompted for. The client must also enter the password to this private key. The client then receives a certificate from the server and writes it to <user>.cert.pem in the enclosing directory. If the file already exists, it will be overwritten.

- **sendmsg**: send certificate and a list of recipient names. Path to certificate, path to private key, and list of recipients are accepted on the command line. Then you will be prompted for a message, the end of which is denoted by newline + ctrl+d. When the recipient certificates are received, sendmsg uses them to encrypt the message, and then sends these encrypted messages to the server.
- **recvmsg**: send certificate. Path to certificate and path to private key are accepted on the command line. When the message is received, it is verified and displayed.

Installation instructions

From the top-level directory, run **install.sh <dest_name>** (you must be a sudoer). This will create a directory called <dest_name>, copy over all the relevant files, spin up a CA and intermediate CA and generate all the necessary certificates, make the server, and start up the server in a chroot environment. (The server takes in 1 command line argument: fully-qualified domain name. Out installation scripts pass in 'www.server.com' as the FQDN; this is the value that must be passed in for the server to successfully start.)
To compile the client programs, enter the client directory and type `make`. To generate a private key for testing, run **client_gen_key.sh**. For ease of use, the private keys generated by this script are not password-protected.

File layout

Server

<dest_name> as specified to **install.sh**

- intermediate/
    - certs/
    - crl/
    - csr/
    - newcerts/
    - private/
    - public/
    - index.txt
    - serial
- mailbox/
    - addleness
        - certs
        - messages
    - ….
- pwds/
    - rca_pass
    - ica_pass
    - server_pass
- serv_conf/
    - www.server.com.cert.pem

        - www.server.com.key.pem
        - client_config.cnf
        - users.txt
- server

Client

- client/
    - getcert
    - changepw
    - sendmsg
    - recvmsg

Security Decisions

File Permissions

- The contents of the $DEST directory are chowned to root:root, and have the last 3 permission bits turned off.
- The password file, as well as the server's certificate and private key, are located in $DEST/serv_conf. This directory is set to 700. The password file is set to 600, and the certificate and key are set to 400. They are all owned root.
- getcert and changepw write certificate files that, by default, belong to the calling user and have their permissions set to 600.
- CSRs generated by getcert and changepw are generated with `umask u=rw,go=`, and are removed by these client programs as soon as they are sent to the server.

Other Security Decisions

- To mitigate the potential damage of starting the server as root, the server is sandboxed. **install.sh** starts it using chroot so that it can only access what is inside the specified directory $DEST. This protects an attacker that may gain root privileges from accessing the entire system.
- To protect against snooping, passwords on the server side are specified to openssl commands using the `file:` option. These passwords are randomly generated by **install.sh**. Passwords for getcert and changepw are entered using getpass() to similarly avoid using the command line.
- Input taken in by getcert and changepw client programs may only contain printable characters and cannot be longer than 512 characters; if either of these is false they will be rejected.
- In order to prevent impersonation of another user in recvmsg (which one could do after calling sendmsg and receiving certificates back), the client program asks not only for the user's certificate, but also for the path to their private key. The client program then verifies that the certificate is valid using the private key, and only then does it send the certificate on to the server.

- In order to prevent impersonation through use of a client other than our client program, the server also checks if the incoming client certificate is signed and verifies it against the CA. This way the request must be signed with a valid private key even if the request is from a program other than our own.

Test Plan

In order to run our tests, please install expect: `apt-get install expect`

Testing largely falls into 2 categories: attacks using the client programs and attacks on the tree structure/filesystem by malicious non-root users on the same machine. In the test directory, run `./test.sh` to run tests. We have automated many but not all of the following tests.

Client Program Abuse

- Try to use wrong password to log in
- Try to use invalid username to log in
- Try to use a username, password, or path to private key that is too long
- Pass an invalid private key to getcert, changepw
- Sendmsg/recvmsg with an invalid or nonexistent certificate
- Sendmsg with invalid recipient
- Try to use certificates generated by other CAs to log in
    - We have provided the script **evil_certificate.sh** to generate such a certificate and write it to test/certificates/addleness-evil.cert.pem
- Try to use certificates with invalid content (time etc. as in HW2) to log in

Filesystem Attacks

- Try to delete or rename the tree and its components
- Try to generate a certificate by bypassing the server and using the root or intermediate CA directly. (If this were to succeed, the user would have a valid certificate without having to enter a username and password.)
- Suppose man-in-the-middle attack
    - Try to decrypt the messages exchanged by the client and server
    - Try to modify the messages (send attacker’s certificate to the client, etc.)