Security Engineering, Assignment 1

Date – Jan 20, 2016

Rudimentary Basic Shell with Basic Access Filtering

-Deepak Kumar Sood

MT15013

Assumptions:

- All the directories are stored at relative path, i.e. where the server is run.
- The Users list is stored at /simple_slash/simple_home directory.
- 8 users are hardcoded in the program namely u1 to u8, and any other username expect this will be treated as invalid.
- User will be allocated a directory the first time he/she logs in.
- Every permission is stored in respective user's directory.
- By default every user has got r-w-x permission only on their directories.
- Server can handle up to 50 connections at a time.
- The string should be send along with fput since this is a one-to-one client-server communication.
- User cannot leave its directory as there is no command for that.
- User cannot access anything outside simple_home for security implementation.
- Port 2222 is used as default port address.
- Users are allowed to access their directory from multiple sources without authentication error.
- Consistency between read/write not handled.
- Buffer size is constrained and long files not handled.

Running Procedure:

1. Run command: "make" or "make compile" from the source directory to compile the server and client program using gcc compiler.
2. Next command: "make start" for starting the server on the terminal.
3. Open new terminal and use command: "make run" for running the client on the terminal.
4. Command: "make clean" is used for removing the ./out files and simple_slash directory.

Inputs used for Testing:

1. All the possibilities for username along with numerals, alphanumeric characters, and special characters are used.
2. After successful authentication every possibility is tested on the 4 commands along with wrong attributes and wrong commands are used.
3. Multiple connections on server tested up to 10 connections was smoothly handled.

Errors Handled:

1. Unauthorized access by users and by bruteforcing.
2. Trying to get to server's directory and access its file not possible.
3. Access to other user's directory using absolute path without permissions handled gracefully.
4. Client crashing if wrong input send handled by server by giving appropriate response.
5. If attributes along with commands not passed then server gives the correct implementation of the command as in any file system.

Attacks Handled:

1. BruteForce attack on server authentication.
2. Null pointer exception handled.
3. Access to server's files restricted.

Threat Model:

I have considered threat model mainly in permissions and authentication part from client side. This file system model does not considers servers as adversary and trying to effect clients. Clients have a minimal implementation and all the computations and storage is done by servers. Clients are considered as adversary and work is done around to protect server from client side attacks.

Defense against Threat Model:

1. Least user privilege is implemented and all the user have access to its own directory by default. All the permissions must by hardcoded and there is not any way that client can change its permission.
2. Client does not have root access to protect server side files.
3. User list is outside any client directory therefore client do not have access to user list, only server can create new users.