CSE-352 SECURITY ENGINEERING

Implementing Access Control Lists

Reeshabh Kumar Ranjan
2017086

1 ASSUMPTIONS

1. The required files: RSA private-public key pair corresponding to every user with correct ownership and permission will be present.
2. The private key will be present as file: <username>.private.
3. The public key will be present as file: <username>.public.
4. The random-file will be automatically generated on calling fput_encrypt_rsa first time per user, and the same will be used for future calls.

   Command to generate the above files is in Section-3 "Using the executable.
5. The user will enter commands in the correct syntax. The correct syntax can be looked up by just running the binary without arguments. For example:

   ```
   ./myls
   ```
6. User will provide the full path of the arguments to binaries.
7. do_exec is supposed to run as the owner of the executable in the argument provided.
8. For the operations, the custom functions will refer to only the owner and owning group specified in custom-acl.
9. Anyone can call getacl on any file to check the ACL-permissions.

10. The permissions are checked in the following order:
    - Owner
    - Owning-group
    - Named-users
    - Named-groups
    - Others
11. Owner's permissions are exempted from mask.
12. User will follow the directions to use this program. (Described below).

2 Error handling

1. If the signature is altered, the verification will fail and execution will halt.
2. If the signature file is not present, the execution will halt.
3. If RSA public-private key pair files are not present, the execution will halt.
4. If the random file is not present, it will be created and encrypted.
5. If the user enters a username which does not exist, the user will receive a suitable error message.
6. If the user enters a group-name which does not exist, the user will receive a suitable error message.
7. If the user enters a path which does not exist, the user will receive a suitable error message.
8. If the user enters a path for a file when he/she was supposed to enter a path for directory or vice/versa, the user will get a suitable error message.
9. If the user tries to execute a file as a user having insufficient permissions, the user will get a permission denied message.
10. At the end of mydo_exec, the euid is reset to the initial value.
11. Only the owner (real/sudo) and root (real) can change acl-entries by calling setacl executable.
12. Any suspicious/erroneous operation will be detected and the execution of the whole program will be terminated immediately.

3 USING THE PROGRAMS

3.1 RUNNING TEST-SCRIPT

Run the script as follows:

```
source ./test.sh
```

3.2 BUILDING THE FILES

In order to generate the executable file, you just need to run the following command in the working directory containing the Makefile and the source code files.

```
make build
sudo make perm
```

It will ask for your password for completion, because chmod and chown are called in the "perm" recipe for make.

3.3 CREATING SUPPORT FILES

In order to generate the public-private key pair, use the following command:

```
cd part-2/
openssl genrsa -out <username>.private 2048
openssl rsa -in <username>.public -pubout > <username>.public
sudo chown <username>:<groupname> <username>.private
sudo chown <username>:<groupname> <username>.public
sudo chmod 600 <username>.private
sudo chmod 600 <username>.public
```

3.4 USING THE EXECUTABLES

First, change the working directory of your shell/terminal to the directory containing the executables.

- fput_encrypt

  ```
  ./fput_encrypt filename
  ```

  This will prompt for the text to be written into the file via stdin. A signature file will be created too with the name filename.sign.

- fget_decrypt

```
./fget_decrypt filename
```

This will first verify the checksum from filename.sign and proceed to decrypt and display the text on stdout if verification passes.

- fput_encrypt_rsa

  ```
  ./fput_encrypt_rsa filename
  ```

  This will prompt for the text to be written into the file via stdin. A signature file will be created too with the name filename.sign.

- fget_decrypt_rsa

  ```
  ./fget_decrypt_rsa filename
  ```

  This will first verify the checksum from filename.sign and proceed to decrypt and display the text on stdout if verification passes.

- setacl

  ```
  ./setacl -m u:username(or leave for other permissions):r-x <filepath>
  ./setacl -m g:groupname:r-x <filepath>
  ./setacl -m m::r-x <filepath> (sets mask)
  ./setacl -o u:username: <filepath> (changes owner)
  ./setacl -o g:groupname: <filepath> (changes owning group)
  ```

- getacl

  ```
  ./getacl <filepath>
  ```

- ls

  ```
  ./myls <path_to_directory>
  ```

  Note: Permissions are printed in binary. Hence for example, a permission "-wx" will be printed as "10" and "rwx" as "111".

- fput

```
./myfput <path_to_file>
<prompt for input>
<input text to be written to the file>
```

- fget

```
./myfget <path_to_file>
```

- create_dir

```
./mycreate_dir <path_to_new_directory>
```

- do_exec

```
./mydo_exec <path_to_executable --with args>
```