NSS1 Assignment1:

Question 1:

Commands to run before compiling
sudo useradd fakeroot
sudo useradd bill
sudo useradd david
sudo useradd steve
sudo useradd joe
sudo useradd kane
sudo useradd travis
sudo passwd “for each user”
sudo mkdir simple_slash
sudo chown -R fakeroot simple_slash

Defending against attacks/Bugs/Errors:
1) Prevents unauthorized users from changing acl entries to a file, only fakeroot and the owner of the file can modify the access of different users on a particular file.
2) Prevents data leakage of sensitive resources. Each time a user tries to access a file resource, it is made sure that the user has necessary permissions to read /write from a file.
3) Take care when a user tries to edit the file which he is permitted to edit, then the content which he writes is actually written into the file by appending it with already existing content such that the previous content of the file is never lost.

Description:
The whole system contains multiple executable files like setacl, getacl and so on. Whenever the user tries to run an executable file the structure instance is first de-serialized from a file ser.txt and is serialized upon further changes.
The user can simply run the executable by running commands on terminal and passing adequate parameters to it. When passing file information, the relative/absolute path of the file can be shared as a parameter to the executable.
For example:
For setacl: ./setacl joe/file3.txt user:bill:rw-
For getacl: ./getacl joe/file3.txt
For fput: ./fput joe/file3.txt HelloJoeThisside

Etc.

Commands to execute:
1) make -f makefile.mak
2) ./precompute

Assumptions:
1) Whenever a new file is created it’s read, write, execute access as well as acl modification access lies with fakeroot only. No other user can modify the ACL entry of the file.
2) ser.txt is an encrypted file and it’s only available with fakeroot and no other user can access it.
3) In fput the content to be written into the file does not contain any spaces. For example:
```
./fput joe/file3.txt HelloJoeThisside
```

Question 2:

Defending against attacks/Bugs/Errors:
1) Take care that any arbitrary executable other than the acl executables do not get root privileges, thereby preventing privilege escalation attack.
2) Provides Authentication mechanisms to confirm only verified users get a chance to run their programs under root privileges. Therefore it prevents random users from accessing the system through root privileges.
3) Pass.txt is configured such that only root users can read and write in it, and no other user has the privilege of reading the secret passphrase thus preventing malicious users from reading the secret passphrase.

Description:
The system when run will ask for the secret passphrase before granting access to root privileges, upon authentication it will allow the user to run any acl related command with root privileges. First it looks whether the running user is root or non-root, in case of root it runs the given command whereas in case of non-root it uses a setuid system call to execute the given program as the owner of the program.

Commands to execute:
```
Make -f make.mak
sudo chmod u+s sudo
sudo chown root:root sudo
sudo chown root:root pass.txt
sudo chmod -rwx pass.txt
```

[last four commands to be run with users present in sudoer list]
Example command: `./sudo getacl bill/file1.txt`

Assumptions:
1) It assumes that the authorized user which runs sudo executable has the access to a secret passphrase.