Version Wed, Feb 25th, 2017

Week 03

Deadline: Saturday, March 4th 2017

Get to know what boot is all about

1. Login to your badak account
2. create a new text named "`what-is-boot.txt`" right where you first login

```
$ vi what-is-boot.txt
```

3. Answer the question below on "`what-is-boot.txt`"

> Explain the order of events when a linux computer start up!

4. Create a new script file named "`.profile`"

```
$ vi .profile
```

5. Echo your name and print your "`what-is-boot.txt`" in "`.profile`" script

```
echo "<<example-name>>"
cat what-is-boot.txt
```

6. Run the "`.profile`" script to check your work

```
$ bash ".profile"
```

7. Exit from badak and login with your badak account again to see the work showing on start-up

```
$ exit
```

8. If you see your name and your answer when you first login, you have succeeded your first subtask!

Example of start up script

9. From badak home, go to `/etc/init.d/`

```
$ cd /etc/init.d/
```

10. Copy `sudo` to your badak home

```
$ cp sudo ~/
```

prepared by **AS** and **MA**

11. Go back to your badak home

```
$ cd ~/
```

12. Open `sudo` in vi

```
$ vi sudo
```

13. Answer the question below on the script below the code

> Explain what happen when the system runs sudo

Use comment to answer it, this is how you comment

```
# this is an example of comment,
# make sure you put the hashtag
# in every row
```

14. Save and exit your `sudo` file

```
Esc
:wq
```

15. Rename your `sudo` into `sudo-explanation.txt`

```
$ mv <<current file name>> <<new file name>>
```

16. Create a new folder inside your `work` folder named `work03`

```
$ cd work
$ mkdir work03
```

17. Move your files from your badak home into `work03` folder

```
$ cd ~/
$ mv <<file name>> <<new file location>>
```

Remember, there are 3 files to be moved into `work03` folder; `.profile`, `what-is-boot.txt`, `sudo-explanation.txt`

18. Congratulation! You have learned about boot and startup script.

Back to Privacy Matters, Encryption and Digital Signature using GnuPG

1. now let's get back to our encryption learning session: sign your works so the other know it truly your works. First we generate hash of all your works. So, instead we sign every files, we just need to sign the hash. We will use *SHA-1 algorithm* for now. (ask *uncle G* for more information about *hashing* and *SHA-1*). Execute the following commands

```
$ sha1sum * > SHA1SUM
$ sha1sum -c SHA1SUM
```

the first command is for creating `SHA1SUM` to all files in current directory **(`work02`)** and save it into a file named "`SHA1SUM`"

the second command is for verify the hash of your files. The "`OK`" output means there is no modification in your current files (try to modify some of your files and then verify it). So, **if you modify your files, you must generate the new hash of it.**

2. To sign a file, use command "`gpg --sign --armor --detach <file>`". The "`--sign --armor --detach`" argument means *"create detached signature with ASCII armored (human readable) output"*

```
$ gpg --sign --armor --detach SHA1SUM
```

after this, there should be a file called `SHA1SUM.asc`, the signature file. This file contains hash and information of the one who signing the file (again, checked by public key)

3. try to verify your signature. Use command "`gpg -verify <signature_file> [<file>]`". The "`<file>`" argument is optional. If not provided, it will try to use `<signature_file>` without "`.asc`" suffix (e.g `test.asc` → `test`)

```
$ gpg --verify SHA1SUM.asc
```

so, as long as the signature is good, we can ensure that the files in this folder **(work03)** is not being modified from the last time of your works

4. create a tar ball. Tar is a way to create an archive file. They will be a new file called "`work03.tbj`". You can ask uncle G for more information.

```
$ cd ..
$ tar cvfj work03.tbj work03/
```

*Useful info: to untar the .tbj file, you can use command "`tar xvfj <file>`"

5. now you can start encrypting your files. To encrypt a file, use command "`gpg --encrypt [--recipient <recipient_identifier>] <file>`".

The "`--recipient <recipient_identifier>`" argument define the one who can decrypt the file. You can define n many recipients.

*Note you must import the recipient's public key before.

encrypt the tar file

```
$ gpg --output work03.tbj.gpg --encrypt --recipient OSTEAM --recipient your@email.com work03.tbj
```

*Use the same email as your Email input on GnuPG key generator.

**Usefull info: after this, there should be a file called `<file>.gpg`. That is the encrypted version of your `<file>` (try to read the content if you can). That file only can be decrypted using receipent's private key (in this case either use osteam's or yours). To decrypt a file, use command "`gpg <file>`"

```
$ gpg <file>.gpg
```

6. copy the file to your github account, under the file week02/

```
$ cp work03.tbj.gpg ~/os171/week03/work03.tbj.gpg
```

7. change your directory to `~/os171/key/`
8. check whether there is a file named "`mypublickey1.txt`" if you don't find it, do the copy once more from the "`work03`" directory
9. change your directory to "`~/os171/week03/`"
10. remove file named "`dummy`"
11. check whether there is a file named "`work03.tbj.gpg`" if you dont find it, do the copy once more from the "`work03`" directory
12. push the change to GitHub server. You can see GitHub tutorial in week01 forum
13. done

Review your Work

Dont forget to check your files/folders. After this lab, your current `os171` folder should looks like:

```
os171
      key
            mypublickey1.txt
      log
            log01.txt
            log02.txt
            log03.txt
      SandBox
            <some_random_name>
      week00
            laporan.txt
      week01
            lab01.txt
            laporan.txt
            myExpectation.txt
            what-time-script.sh
      week02
            work02.tbj.gpg
                  work02
                        *00-toc.txt
                        *01-public-osteam.txt
                        *02-ls-al.txt
                        *03-list-keys1.txt
                        *04-list-keys2.txt
                        *hello.c
                        *hello
                        *status.c
                        *status
                        *loop.c
                        *loop
                        *exercise.c
                        *exercise
                        *SHA1SUM
                        *SHA1SUM.asc
      week03
```

```
        work03.tbj.gpg
              work03
                    *.profile
                    *sudo-explanation.txt
                    *what-is-boot.txt
                    *SHA1SUM
                    *SHA1SUM.asc
week04
        dummy
week05
        dummy
week06
        dummy
week07
        dummy
week08
        dummy
week09
        dummy
week10
        dummy
xtra
        dummy
```

keep in mind for every files/folders with wrong name, you will get penalty point.

*means file that should be inside the archived file.