

Week 03

Deadline: Saturday, September 23rd 2017

Get to know what boot is all about

1. Login to your badak account
2. Create a new text file named "what-is-boot.txt" right where you first login. We will be using vi, the powerful text editor.

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

3. Write the following text on "what-is-boot.txt" by typing 'i' (insert) while in command mode.

```
Hello, this is explanation about linux booting!
```

When you're done, save your file

```
Esc  
:wq
```

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

```
$ vi .profile
```

5. Add commands to echo your name and print the contents of "what-is-boot.txt" in ".profile" script

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

When you're done, save your file

```
Esc  
:wq
```

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

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

7. Exit from badak

```
$ exit
```

8. Login with your badak account again to see the work showing on start-up

```
$ ssh your_username@badak.cs.ui.ac.id
```

Example:

```
Last login: Sat Feb 25 08:56:06 2017 from kawung.cs.ui.ac.id
nama kamu
contoh jawaban
```

9. Answer the question below on "what-is-boot.txt":

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

WARNING: Don't forget to put the source of your answer. Make sure it is your own answer, don't plagiarize!

When you're done, save your file

```
Esc
```

```
:wq
```

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

Example of start up script

11. From badak home, go to /etc/init.d/

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

12. Copy sudo to your badak home

```
$ cp sudo ~/
```

13. Go back to your badak home

```
$ cd ~/
```

14. Open sudo in vi and try to understand the script

```
$ vi sudo
```

15. Answer the question below on the script below the code:

Explain what happen when the system runs this command!

```
$ /etc/init.d/sudo start
```

WARNING: You can't actually run the command because you don't have the privilege. Use comment to answer it. This is how you comment:

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

WARNING: Make sure it is your own answer, don't plagiarize!

16. Save and exit your sudo file

```
Esc
```

```
:wq
```

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

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

18. Move `.profile`, `what-is-boot.txt`, and `sudo-explanation.txt` from your `badak` home into `work03` folder

```
$ cd ~/
```

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

19. Make `work03` folder in your `work` directory change your present working directory to `work03`

```
$ cd work
```

```
$ mkdir work03
```

```
$ cd work03
```

20. Make sure there are 3 files in your `/work/work03`

```
$ ls -al
```

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

21. Congratulations! You have learned about boot and startup script.

Back to Privacy Matters, Encryption and Digital Signature using GnuPG

22. 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

```
$ shasum * > SHA1SUM
```

```
$ shasum .profile >> SHA1SUM
```

```
$ shasum -c SHA1SUM
```

The first command is for creating `SHA1SUM` to all files in current directory (**work03**) and save it into a file named `"SHA1SUM"`

The second command is to append `.profile` because it cannot be added automatically with `shasum * > SHA1SUM`

The third 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.**

23. To sign a file using your signature (key), 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).

24. 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

25. 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>"`

26. 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 have imported the recipient's public key in week02, if you've done it, ignore this note.

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.

**useful 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 recipient's private key (in this case either use osteam's or yours). To decrypt a file, use command `"gpg <file>"`

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

27. Copy the file to your github account, under the file week03/

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

28. Change your directory to "~/os171/week03/"

29. Remove file named "dummy"

30. Check whether there is a file named "work03.tbj.gpg" if you dont find it, do the copy once more from the "work03" directory

31. Push the change to GitHub server. You can see GitHub tutorial in week01 forum

32. 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
│   └── report.txt
├── week01
│   ├── lab01.txt
│   ├── report.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.**