



# MODUL PRAKTIKUM SISTEM OPERASI DAN JARINGAN KOMPUTER

## 2. Pipeline dan Scripting

# PIPELINE AND SCRIPTING

## A. Text Editor

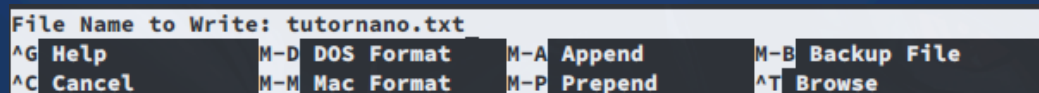
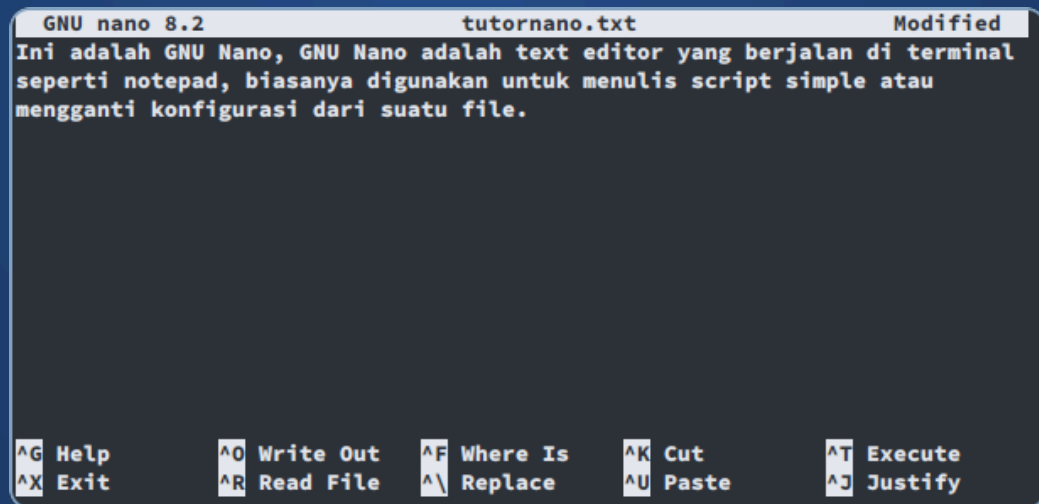
Text editor adalah suatu program di sistem operasi berbasis unix, dimana kita bisa melakukan **edit file** melalui **terminal/command line**, diantaranya adalah nano dan vim.

### 1. GNU Nano

GNU Nano adalah editor teks yang mudah digunakan pada sistem UNIX atau Linux. Ini sering digunakan untuk mengedit file konfigurasi atau menulis skrip sederhana langsung dari command line. Nano sangat bermanfaat bagi pengguna yang membutuhkan editor teks **tanpa fitur rumit**.

Nano dapat dijalankan dengan menggunakan command

```
$ nano [nama file]
```



Tekan **Ctrl + O** kemudian **[enter]** untuk save, disini anda akan di prompt untuk save beserta nama file-nya (save-as). Kemudian **Ctrl + X** untuk keluar.

## 2. VIM / VI Improved

- 1 Ini adalah VI Improved atau Vim.
- 2 VIM adalah suatu text editor yang memiliki banyak fitur
- 3 VIM adalah mode-based text editor, sehingga memiliki mode-mode
- 4 yang digunakan untuk use-case masing masing

Vim adalah suatu text editor yang berbasis mode, artinya text editor ini memiliki beberapa mode yang digunakan untuk navigasi.

Vim memiliki 3 mode, yaitu **INSERT**, **NORMAL**, dan **VISUAL**.

### a. NORMAL mode

-- NORMAL --

**NORMAL** mode digunakan ketika kita ingin melakukan vim motion, yaitu macro motion khusus untuk vim. Saat vim di-run, kita secara otomatis akan memakai **NORMAL** mode.

Kita bisa masuk **NORMAL** mode dengan tombol [esc]

By default, vim saat dijalankan otomatis masuk ke normal mode, beberapa macro di **NORMAL** mode diantaranya:

- [h/j/k/l] untuk navigasi karakter
- [y] untuk copy
- [p] untuk paste
- [gg] untuk ke awal file
- [G] untuk ke akhir file
- [v] untuk pindah ke **VISUAL** mode
- [i] untuk pindah ke **INSERT** mode
- [x] untuk delete suatu karakter
- [r] untuk replace suatu karakter

Dan masih banyak lagi, contohnya di bawah ini:

[Vim cheatsheet](#)

#### b. INSERT mode

```
-- INSERT --
```

**INSERT** mode adalah mode utama yang digunakan untuk mengedit text di dalam file, perilaku **INSERT** mode persis sama dengan notepad pada windows.

\*Kita bisa masuk **INSERT** mode dengan menekan tombol [i]

\*Kita bisa kembali ke **NORMAL** mode menggunakan [esc]

#### c. VISUAL mode

```
-- VISUAL --
```

**VISUAL** mode adalah mode dimana kita melakukan select text, dimana text yang telah kita select dapat diproses menggunakan vim motion.

\*Kita bisa masuk **VISUAL** mode dengan menekan tombol [v]

\*Kita bisa kembali ke **NORMAL** mode menggunakan [esc]

#### d. Command mode

```
:wqa!|
```

Vim memiliki command line tersendiri, dimana kita bisa menjalankan vim commands, vim commands disini ada banyak, tetapi kita akan fokus bagaimana cara melakukan Save and Exit.

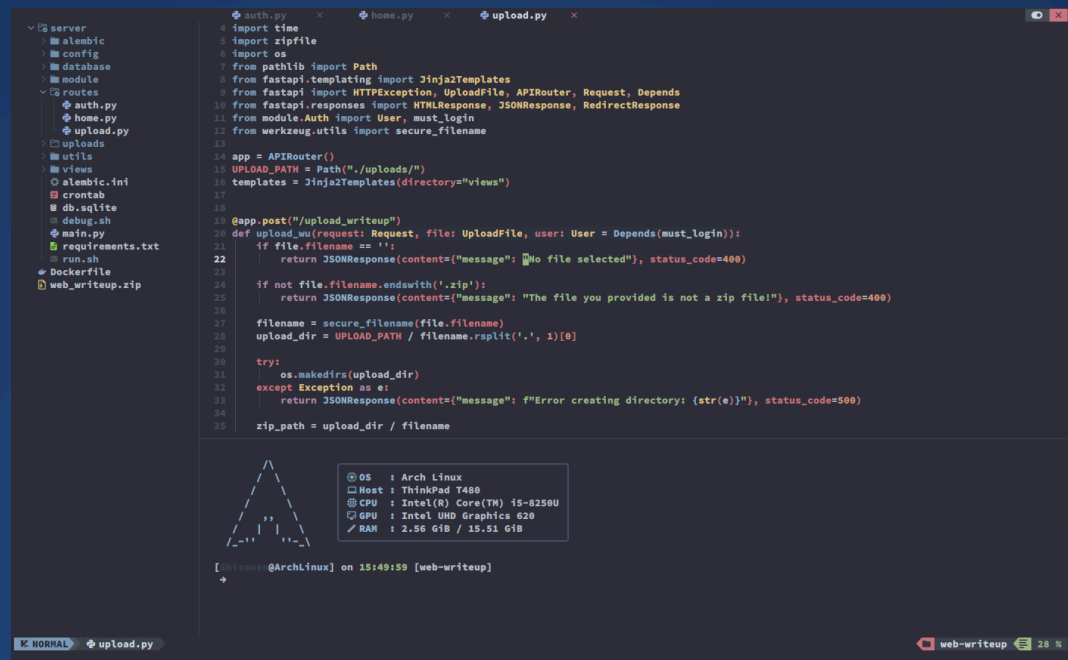
\*Kita bisa meng-input command dengan cara mengetik [:] di dalam NORMAL mode. Seperti [/] pada minecraft

Command penting di VIM:

- :q = untuk quit
- :q! = untuk memaksa quit tanpa mengganti file
- :w = untuk save
- :wq = untuk save and quit

Karena macro untuk vim sangat banyak, bisa melihat referensi berikut sebagai panduan: <https://devhints.io/vim>

Vim juga memiliki plugin system yang dapat membuat ui plain seperti diatas, menjadi IDE yang lengkap, dengan LSP, linting, dan autocomplete seperti dibawah:



## B. Pipeline dan Redirection

### 1. Konsep dasar Pipe dan Redirection

Kita telah belajar mengenai command seperti echo, grep, cat, dan lain sebagainya. Muncul sebuah pertanyaan, “Bagaimana jika saya ingin menyimpan hasil command ke dalam suatu file?

Jawabannya dengan menggunakan Pipe dan Redirection

**Pipe ( | )** : Pipe digunakan untuk mengirim output dari satu perintah langsung sebagai input ke perintah lain. Ini membuat chaining dari beberapa perintah di terminal.

**Redirection (>, >>, <, <<)** : Redirection digunakan untuk mengalihkan output atau input dari suatu perintah ke file atau dari file. Ini penting untuk mengelola data dan log secara efisien.

## 2. Menggunakan Pipe ( | )

Contoh sederhana:

```
$ ls | grep ".txt"
```

Di sini, ls akan menampilkan daftar file, dan outputnya diteruskan melalui pipe ke grep, yang kemudian mencari file dengan ekstensi .txt.

Chaining multiple commands: Kalian bisa menghubungkan beberapa perintah dalam satu baris dengan menggunakan beberapa pipe.

Contoh:

```
$ cat file.txt | grep "hello" | sort
```

File.txt akan dibaca isinya, setelah itu dicari kata "hello" dari output file.txt, setelah itu akan di sorting.

## 3. Redirection Output (>, >>)

**Overwrite (>):** Mengalihkan output dari perintah ke file, menimpa isi file jika sudah ada.

Contoh:

```
$ echo "Hello, World!" > myfile.txt
```

Output dari echo "Hello, World!" akan dipakai untuk mengisi myfile.txt

**Append (>>):** Menambahkan output dari perintah ke akhir file tanpa menimpa isi yang sudah ada.

Contoh:

```
$ echo "This is new content" >> myfile.txt
```

Bedanya >> dan > , yaitu >> tidak menimpa isi dari myfile.txt



#### 4. Redirection Input (<, <<)

**Input dari file (<):** Menggunakan konten dari file sebagai input untuk perintah.

Contoh:

```
$ sort < unsorted.txt
```

Command ini akan mengambil isi dari unsorted.txt sebagai input untuk command sort

**Here Document (<<):** Membuat input multi-baris langsung di terminal.

contoh:

```
$ file.txt << "Isi dari File ini adalah.. Bla bla bla"
```

Command ini akan mengisi akhir dari file.txt dengan input kita, yaitu "Isi dari File ini adalah.. Bla bla bla"

#### 5. Penggunaan Kombinasi Pipe dan Redirection

Contoh:

```
$ grep "error" logfile.txt | sort > sorted_errors.txt
```

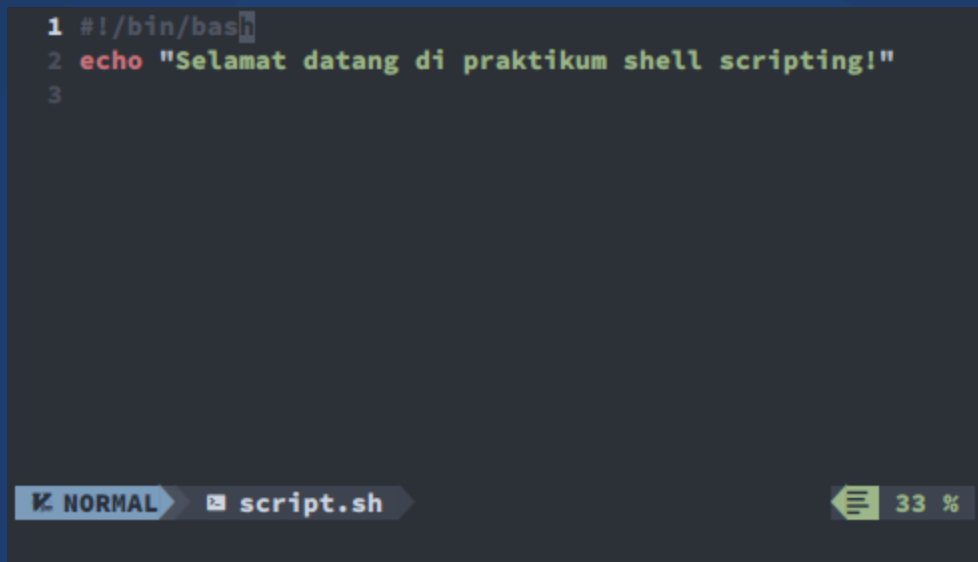
Di sini, kita mencari kata "error" dalam logfile.txt, kemudian mengurutkan hasilnya, dan akhirnya menyimpannya dalam sorted\_errors.txt.

### C. Shell Scripting

**Shell scripting** adalah cara untuk mengotomatisasi tugas-tugas yang sering dilakukan di terminal Unix/Linux. Skrip ini merupakan **kumpulan perintah yang disimpan dalam file dan dieksekusi secara berurutan**. Dengan shell scripting, kita dapat menyederhanakan proses-proses manual, membuat alur kerja lebih efisien, dan mengurangi kesalahan manusia.

Untuk membuat shell script, buatlah suatu file yang memiliki ekstensi **.sh**, kemudian tambahkan **header** seperti berikut:

```
1 #!/bin/bash
2 echo "Selamat datang di praktikum shell scripting!"
3
```



Script ini akan memberi output “Selamat datang di praktikum shell scripting”  
Kita dapat menjalankan suatu shell script dengan melakukan

```
$ ./[nama_script].sh
```

(SQUARE BRACKET ([ ]) TIDAK USAH DIKETIK, ITU HANYA PENANDA)

\*apabila tidak bisa dijalankan, lakukan ini terlebih dahulu di terminal:

```
$ chmod +x [nama_script].sh
```



## 1. Variables

Shell Script dapat dianggap bahasa pemrogramannya sendiri, ia juga memiliki **percabangan**, **looping**, **variable assignment** dan seterusnya. Namun, di praktikum ini kita hanya akan fokus terhadap aspek command line-nya saja. Contoh:

```
1 #!/bin/bash
2
3 name="Igor"
4 age=47
5 rank="Ubersturm III"
6 grade='C'
7
8 echo "Name : $name, Age : $age, Ranking at East Coalition : $rank, Mission Grade : $grade"
```

NORMAL script.sh Shisones 100 %

Akan memberi output seperti ini

```
[Shisones@ArchLinux] on 16:21:08 [~]
→ ./script.sh
Name : Igor, Age : 47, Ranking at East Coalition : Ubersturm III, Mission Grade : C
```

Shell Script juga dapat melakukan operasi aritmatika, contohnya seperti ini:

```
1 #!/bin/bash
2
3 a=10
4 b=5
5 result=$((a + b))
6
7 echo "$a + $b = $result"
```

NORMAL script.sh 100 %

```
[Shisones@ArchLinux] on 16:26:25 [~]
→ ./script.sh
10 + 5 = 15
```

Perhatikan bahwa operasi aritmatika dikurung dengan 2 buah kurung dan diawali dengan tanda '\$'.

## 2. Command Variables

Selain variabel biasa, kita dapat melakukan command dan memasukkan output-nya ke dalam variabel menggunakan **sub-commands**, ditandai dengan **\$()**. contohnya:

```
1 #!/bin/bash
2
3 workdir=$(pwd)
4 echo "Current working directory is $workdir"
5
```

NORMAL script.sh 80 %

Dia akan menjalankan **pwd**, kemudian menyimpan outputnya di dalam variabel **workdir**, kemudian akan di print menggunakan **echo**.

```
[Shisones@ArchLinux] on 16:31:13 [~]
→ ./script.sh
Current working directory is /home/Shisones
```

## 3. Conditionals

Conditional adalah konsep percabangan (if-else), dimana kita dapat **membandingkan** suatu variable dengan variabel lain.

```
1 #!/bin/bash
2 # Skrip menggunakan percabangan
3
4 angka=10
5
6 # -gt = greater than ( > )
7 # -lt = less than ( < )
8 # -le = less than or equal ( <= )
9 # -ge = greater than or equal ( >= )
10
11 if [ $angka -gt 5 ]; then
12     echo "$angka lebih besar dari 5."
13 else
14     echo "$angka tidak lebih besar dari 5."
15 fi
16
```

NORMAL script.sh 62 %  
AutoSave: saved at 16:36:49

Diatas adalah suatu program yang memiliki variabel angka, dimana angka itu akan dibandingkan dengan 5, apabila lebih besar, dia akan masuk ke line 12, apabila tidak lebih besar, dia akan masuk ke line 14. Output akan mengikuti percabangan tersebut :

```
[Shisones@ArchLinux] on 16:38:03 [~]  
→ ./script.sh  
10 lebih besar dari 5.
```

#### 4. Looping

Looping, atau **perulangan**, merupakan konsep dimana program dapat melakukan suatu **instruksi secara berulang**, dibawah ini adalah program yang membuat variabel i, kemudian mengisinya dengan angka satu sampai 5.

```
1 #!/bin/bash  
2 # Pengulangan for  
3  
4 for i in 1 2 3 4 5; do  
5     echo "Angka ke-$i"  
6 done  
7
```

Outputnya akan seperti ini:

```
[Shisones@ArchLinux] on 16:42:05 [~]  
→ ./script.sh  
Angka ke-1  
Angka ke-2  
Angka ke-3  
Angka ke-4  
Angka ke-5
```

Masih banyak hal seperti **while loop**, **functions**, **switch case**, **dsb**. Tetapi materi itu akan difokuskan di mata kuliah **Dasar dasar pemrograman**.

Scripting merupakan suatu teknik yang sangat berguna, khususnya untuk System Administrator yang ingin mengganti konfigurasi tanpa ribet, Linux User untuk dapat memudahkan workflownya, maupun Ethical Hacker yang dapat membuat shell script untuk melakukan exploit.

Contoh asli dunia nyata pemanfaatan shell script:

```
1 /bin/bash
2
3 discord_path=$(readlink -f $(which discord))
4 json_file="${discord_path/Discord/resources/build_info.json}"
5
6 current_version=$(sed -n 's/.*"version": "\(.*\)".*/\1/p' "$json_file")
7
8 # Increment the version using awk
9 new_version=$(echo "$current_version" | awk -F. -v OFS=. '
10 {
11     if (NF == 3) {
12         $3++
13     } else if (NF == 2) {
14         $2++
15     }
16     print
17 }')
18
19 sed -i "s/\"version\": \"$current_version\"/\"version\": \"$new_version\"/" "$json_file"
20
21 echo "Version updated to $new_version"
```

K NORMAL updateddiscord.sh Scripts 4 %

Script untuk update discord pada linux

```
73
74 sudo -u hduser chmod o+w /home/$hdUserName/.bashrc
75 sudo -u hduser chmod o+w /home/$hdUserName/hadoop/etc/hadoop/hadoop-env.sh
76 sudo -u hduser chmod o+w /home/$hdUserName/hadoop/etc/hadoop/core-site.xml
77 sudo -u hduser chmod o+w /home/$hdUserName/hadoop/etc/hadoop/mapred-site.xml
78 sudo -u hduser chmod o+w /home/$hdUserName/hadoop/etc/hadoop/hdfs-site.xml
79 sudo -u hduser chmod o+w /home/$hdUserName/hadoop/etc/hadoop/yarn-site.xml
80
81 #hadoop-env.sh
82
83 sudo sed -i 's|${JAVA_HOME}|$java_home|g' /home/$hdUserName/hadoop/etc/hadoop/hadoop-env.sh
84
85 echo -e '\n\n #Hadoop Variable START \n export HADOOP_HOME=/home/'$hdUserName'/hadoop \n export HADOOP_INSTALL=$HADOOP_HOME \n export HADOOP_MAPRED_HOME=$HADOOP_HOME \n export HADOOP_COMMON_HOME=$HADOOP_HOME \n export HADOOP_HDFS_HOME=$HADOOP_HOME \n export YARN_HOME=$HADOOP_HOME \n export HADOOP_CONF_DIR=$HADOOP_HOME/conf \n export HADOOP_LOG_DIR=$HADOOP_HOME/logs \n export HADOOP_PID_DIR=$HADOOP_HOME/run \n export HADOOP_SCRIPTS_DIR=$HADOOP_HOME/bin \n #Hadoop Variable END\n\n' >> /home/$hdUserName/.bashrc
86
87 source /home/$hdUserName/.bashrc
88
89 #core-site.xml
90
91 sudo sed -i 's|<configuration>\n<property>\n<name>hadoop.tmp.dir</name>\n<value>/home/hduser/hadoop/app/hadoop/tmp</value>\n</property>\n<property>\n<name>fs.default.name</name>\n<value>hdfs://localhost:54310</value>\n</property>' /home/$hdUserName/hadoop/etc/hadoop/core-site.xml
92
93 #mapred-site.xml
94
95 sudo sed -i 's|<configuration>\n<property>\n<name>mapreduce.framework.name</name>\n<value>yarn</value>\n</property>' /home/$hdUserName/hadoop/etc/hadoop/mapred-site.xml
96
97 #yarn-site.xml
98
99 sudo sed -i 's|<configuration>\n<property>\n<name>yarn.nodemanager.aux-services</name>\n<value>mapreduce_shuffle</value>\n</property>' /home/$hdUserName/hadoop/etc/hadoop/yarn-site.xml
100
101 #revoking write permission for .bashrc, hadoop-env.sh, core-site.xml, mapred-site.xml, hdfs-site.xml, yarn-site.xml files.
102 sudo -u hduser chmod o-w /home/$hdUserName/.bashrc
103 sudo -u hduser chmod o-w /home/$hdUserName/hadoop/etc/hadoop/hadoop-env.sh
104 sudo -u hduser chmod o-w /home/$hdUserName/hadoop/etc/hadoop/core-site.xml
105 sudo -u hduser chmod o-w /home/$hdUserName/hadoop/etc/hadoop/mapred-site.xml
106 sudo -u hduser chmod o-w /home/$hdUserName/hadoop/etc/hadoop/hdfs-site.xml
107 sudo -u hduser chmod o-w /home/$hdUserName/hadoop/etc/hadoop/yarn-site.xml
108
109 echo "-----HADOOP DIRECTORY-----"
110 sudo ls /home/$hdUserName/hadoop/
111 echo "-----Proceed with 'yes' for continue connecting-----"
112 sudo -u hduser ssh localhost
```

K NORMAL hadoopinstall.sh Scripts 69 %

Script untuk auto install hadoop distributed file system

```

4373 if [ "$PSTORAGE_RPCD" ] || [ "$DEBUG" ]; then
4374     print_2title "Analyzing Rpcd Files (limit 70)"
4375     if ! [ "`echo \"$PSTORAGE_RPCD\" | grep -E \"rpcd$\"`" ]; then if [ "$DEBUG" ]; then
4376         echo_not_found "rpcd"; fi; fi; printf "%s" "$PSTORAGE_RPCD" | grep -E "rpcd$" | while
4377         read f; do ls -ld "$f" 2>/dev/null | sed -${E} "s,rpcd$,${SED_RED},"; cat "$f" 2>/dev/
4378         null | grep -IEv "^$" | sed -${E} "s,username.+|password.+,$(SED_RED),g"; done; echo
4379         ""
4380     fi
4381     if [ "$PSTORAGE_BITCOIN" ] || [ "$DEBUG" ]; then
4382         print_2title "Analyzing Bitcoin Files (limit 70)"
4383         if ! [ "`echo \"$PSTORAGE_BITCOIN\" | grep -E \"bitcoin\\.conf$\"`" ]; then if [ "$
4384         DEBUG" ]; then echo_not_found "bitcoin.conf"; fi; fi; printf "%s" "$PSTORAGE_BITCOIN"
4385         | grep -E "bitcoin\\.conf$" | while read f; do ls -ld "$f" 2>/dev/null | sed -${E} "s,b
4386         itcoin\\.conf$,${SED_RED},"; cat "$f" 2>/dev/null | grep -IEv "^$" | grep -Ev "^#" | se
4387         d -${E} "s,user=.*|password=.*|auth=.*,${SED_RED},g"; done; echo "";
4388     fi
4389     if [ "$PSTORAGE_GLUSTERFS" ] || [ "$DEBUG" ]; then
4390         print_2title "Analyzing GlusterFS Files (limit 70)"
4391         if ! [ "`echo \"$PSTORAGE_GLUSTERFS\" | grep -E \"glusterfs\\.pem$\"`" ]; then if [
4392         "$DEBUG" ]; then echo_not_found "glusterfs.pem"; fi; fi; printf "%s" "$PSTORAGE_GLUST
4393         ERFS" | grep -E "glusterfs\\.pem$" | while read f; do ls -ld "$f" 2>/dev/null | sed -${
4394         E} "s,glusterfs\\.pem$,${SED_RED},"; done; echo "";
4395         if ! [ "`echo \"$PSTORAGE_GLUSTERFS\" | grep -E \"glusterfs\\.ca$\"`" ]; then if [
4396         "$DEBUG" ]; then echo_not_found "glusterfs.ca"; fi; fi; printf "%s" "$PSTORAGE_GLUSTER
4397         FS" | grep -E "glusterfs\\.ca$" | while read f; do ls -ld "$f" 2>/dev/null | sed -${E}
4398         "s,glusterfs\\.ca$,${SED_RED},"; done; echo "";
4399         if ! [ "`echo \"$PSTORAGE_GLUSTERFS\" | grep -E \"glusterfs\\.key$\"`" ]; then if [
4400         "$DEBUG" ]; then echo_not_found "glusterfs.key"; fi; fi; printf "%s" "$PSTORAGE_GLUST
4401         ERFS" | grep -E "glusterfs\\.key$" | while read f; do ls -ld "$f" 2>/dev/null | sed -${
4402         E} "s,glusterfs\\.key$,${SED_RED},"; done; echo "";
4403     fi
4404     if [ "$PSTORAGE_TERRAFORM" ] || [ "$DEBUG" ]; then
4405         print_2title "Analyzing Terraform Files (limit 70)"
4406         if ! [ "`echo \"$PSTORAGE_TERRAFORM\" | grep -E \"\\.tfstate$\"`" ]; then if [ "$DE
4407         BUG" ]; then echo_not_found ".tfstate"; fi; fi; printf "%s" "$PSTORAGE_TERRAFORM" | g
4408         rep -E "\\.tfstate$" | while read f; do ls -ld "$f" 2>/dev/null | sed -${E} "s,\\.tfstat
4409         e$,${SED_RED},"; cat "$f" 2>/dev/null | grep -IEv "^$" | sed -${E} "s,secret.*,${SED_R
4410         ED},g"; done; echo "";
4411         if ! [ "`echo \"$PSTORAGE_TERRAFORM\" | grep -E \"\\.tf$\"`" ]; then if [ "$DEBUG"
4412         ]; then echo_not_found ".tf"; fi; fi; printf "%s" "$PSTORAGE_TERRAFORM" | grep -E "\.
4413         tf$" | while read f; do ls -ld "$f" 2>/dev/null | sed -${E} "s,\\.tf$,${SED_RED},"; cat
4414         "$f" 2>/dev/null | grep -IEv "^$" | sed -${E} "s,secret.*,${SED_RED},g"; done; echo
4415         "";
4416     fi

```

Script untuk Privilege Escalation

#### D. Tugas Praktikum

Dua negara sedang berperang: Yagongawi dan Third Incarnation of Prussia. **Willhelm Brzeszkiewicz**, seorang System Administrator dari Third Incarnation of Prussia, memiliki **log file berukuran 40 baris** yang mencatat aktivitas militer negaranya. Tugas kalian adalah membantu Willhelm untuk memproses log aktivitas tersebut.

Sebagai tangan kanan Willhelm, kalian diminta untuk membuat script yang dapat:

- **Membersihkan** data **top\_secret.txt** dari simbol aneh, kirim ke **military\_journal.txt**
- Menyaring log file untuk menampilkan **serangan** yang **berhasil**. Dikirim ke file **successful\_attack.txt**
- Menampilkan aksi yang terjadi dalam jam **22:00 sampai 05:00**, dimana Prussia bertahan, Dikirim ke file **night\_raids.txt**

Logfile bisa di copy dari sini :

[top\\_secret.txt](#)

Tips and Tricks:

\*Gunakan **sed** untuk membersihkan file dari simbol %, #, dan ?

\*Gunakan pipeline (|) untuk mengalirkan hasil dari satu perintah ke perintah lain.

\*Simpan output ke file menggunakan redirection (> atau >>).

Dokumen dikumpulkan dengan format file **NIM\_Nama\_Kelas-Angkatan.zip**.

contoh : **2205123\_Igor Kachankov\_C3-1945.zip**

Di dalam dokumen harus ada komponen:

- Script.txt (script kalian yang diganti menjadi .txt)
- File pdf yang berisi:
  - Penjelasan Script
  - Screenshot dari isi ketiga file setelah menjalankan script

**AGAR TIDAK BERANTAKAN, FILE TUGAS DI PC LAB DISIMPAN DALAM 1 FOLDER DI DALAM HOME**

**Contoh : /home/labum08/TP2, jangan disimpan di /home/labum**

**Belajar rapih cuy**