16 giugno 2025

Durata esame: 2 ore e 30 minuti

Sezione	Punti
Processo periodico (§1)	16
- Script Python (§1.1)	8/16
- Service (§1.2)	4/16
- Timer (§1.3)	4/16
Filtraggio dei pacchetti e NAT (§2)	8
Domande a risposta aperta (§3)	9

Per stampare

```
$ stampa <path/file/da/stampare>
```

> [!warning] 1. Scrivere **nome**, **cognome** e numero di matricola su ogni file che si stampa 2. Una volta mandati in stampa i file, avvisare il docente e **rimanere seduti al posto**
>
> [!tip] 1. Se si nota un errore sul file stampato, lo si può correggere a penna

1. Processo periodico

1.1. Script Python

Scrivi uno script Python che sposta ogni file più vecchio di un certo numero di secondi da una directory specificata e da tutte le sue sottodirectory in una cartella di archivio denominata `archive`, creata (se necessario) in `~`. Nella tua home directory, crea una directory chiamata `file-archiver` e, al suo interno, un file chiamato `app.py`, utilizzando questo template:

```python
# nome e cognome:
# matricola:
#
# path:

import argparse
import os
import sys
import time
import shutil

def main():
    pass
```

```python
if __name__ == "__main__":
    main()
```

Lo script deve accettare esattamente due argomenti da linea di comando, analizzati con il modulo `argparse`. Il primo argomento, `--path`, è una stringa obbligatoria che indica il percorso assoluto della directory da controllare. Il secondo argomento, `--seconds`, è un intero obbligatorio che specifica l'età massima (in secondi) oltre la quale i file devono essere spostati.

Dopo il parsing, valida entrambi gli input: controlla che il percorso sia assoluto (`os.path.isabs`), esista (`os.path.exists`) e sia una directory (`os.path.isdir`); verifica inoltre che `--seconds` sia un intero positivo. Se un controllo fallisce, stampa un messaggio d'errore esplicativo sullo standard error (`print`) ed esci con un codice di stato diverso da zero (`sys.exit`).

Dopo la validazione, assicurati che in `~` esista la cartella `archive` (`os.path.expanduser`, `os.makedirs`). Percorri quindi ricorsivamente l'albero delle directory al percorso fornito come primo argomento (`os.listdir`, `os.path.join`, `os.path.isdir`). Per ogni file incontrato (`os.path.isfile`), calcola da quanti secondi non viene modificato (`os.path.getmtime`, `time.time`). Se l'età in secondi è maggiore o uguale al valore di soglia, sposta il file nella cartella `~/archive` (`shutil.move`) e stampa un messaggio di log sullo standard output (`print`). Non tentare di spostare directory.

Ad esempio, eseguendo

```
$ python ~/file-archiver/app.py --path ~/target --seconds 30
```

lo script dovrà spostare nella cartella `~/archive` tutti i file più vecchi di 30 secondi presenti in `~/target` e in tutte le sue sottodirectory.

1.2. Service

Crea un'unità *service* denominata `file-archiver.service` nella tua istanza utente di `systemd`. Configurala per avviare `~/file-archiver/app.py` con gli argomenti `--path %h/mydocs` e `--seconds 30`. Usa questo template:

```
# nome e cognome:
# matricola:
#
# path:
```

1.3. Timer

Crea un'unità *timer* denominata `file-archiver.timer` nella tua istanza utente di `systemd`. Configurala per attivare `file-archiver.service` alle 04:00 di ogni sabato e domenica. Usa questo template:

```
# first and last name:
```

```
# serial number:
#
# path:
#
# comando per abilitare il timer:
# comando per avviare il timer:
```

2. Filtraggio dei pacchetti e NAT

Configura un firewall Linux usando `iptables`. Il firewall dispone di due interfacce:

NIC	Indirizzo di rete	IP del firewall	Ambito
`eth0`	`203.0.113.0/24`	`203.0.113.10`	Pubblico
`eth1`	`192.168.50.0/24`	`192.168.50.1`	Privato

I nodi sulla rete `192.168.50.0/24` utilizzano questo firewall come gateway di default. L'host `192.168.50.20` esegue un server web che supporta HTTPS.

Applica le seguenti regole:

Tabella	Catena	Regola
`filter,nat`	`*`	Elimina le regole esistenti
`filter`	`INPUT,FORWARD`	Scarta tutto a meno che non sia esplicitamente permesso
`filter`	`INPUT`	Consenti pacchetti ICMP ricevuti su `eth1`
`filter`	`INPUT`	Consenti pacchetti SSH (`tcp/22`) ricevuti su `eth1`
`filter`	`FORWARD`	Consenti pacchetti HTTP (`tcp/80`) e HTTPS (`tcp/443`) ricevuti su `eth0` e `eth1`
`filter`	`FORWARD`	Consenti pacchetti con stato `ESTABLISHED,RELATED`
`nat`	`POSTROUTING`	SNAT per i pacchetti in uscita su `eth0` affinché gli host privati ricevano risposte da Internet
`nat`	`PREROUTING`	DNAT per pacchetti HTTPS (`tcp/443`) ricevuti su `eth0` verso `192.168.50.20:30443`

Usa questo template:

```
# nome e cognome:
# matricola:
```

3. Domande a risposta aperta

1. Perché un *lazy unmount* (`umount -l`) è considerato non sicuro, quale comando permette di individuare i processi che mantengono ancora riferimenti al filesystem occupato e come si può eseguire invece un *unmount* pulito?

2. Che cos'è una vulnerabilità software, qual è un esempio specifico di tale vulnerabilità e in che modo le pratiche di revisione del codice open-source possono contribuire a ridurre queste vulnerabilità?
3. Che cos'è la crittografia a chiave simmetrica, come funziona e quali sono i suoi principali vantaggi e svantaggi?

Usa questo template:

```
# nome e cognome:
# matricola:

1.

2.

3.
```