SFTP and Filesystem

Computer System Administration Homework 2

Hsuchy, NYCU CSIT

Outline

- 2-1: Manage SFTP file server
- 2-2: SFTP logging and Monitoring daemon
- 2-3: BTRFS snapshot script

Setup

- Based on HW0 and HW1
- Make your **judge** user a **no password sudoer**
  - i.e. **judge** should be able to sudo any command without prompted password
- Hint:
  - visudo(8)
  - sudoers(5)
  - NOPASSWD

(2-1) (27%)
Manage SFTP file server

HW2-1: Overview

- Build a file server with SFTP
- Create a **administrator** with full access to the file server
- Create 2 **registered users** with limited access
  - administrator **is not counted** as registered users
- Create a readonly **anonymous** user
- Bury some treasure in your file server 🧐

HW2-1: Requirements (1/4)

- Create following users:
  - `sysadm` (referred as **administrator**)
  - `sftp-u1`, `sftp-u2` (referred as **registered users**)
  - `anonymous`
- **Registered users** and **anonymous** **cannot login via SSH**
  - SFTP only
- Everyone should support login to sftp with ssh key
  - same public key with judge

HW2-1: Requirements (2/4)

- Host SFTP server on `$SFTP_ROOT`.
  - `$SFTP_ROOT = /mnt/hw2/sftp`
- Create 2 directories under `$SFTP_ROOT`, `/public` and `/private`
- **Registered users** and **anonymous** should **chroot** to `$SFTP_ROOT`
- **Administrator** should **not chroot**
- The **SFTP start directory** of **All user** should at `$SFTP_ROOT`

HW2-1: Requirements (3/4)

- `$SFTP_ROOT/public`
  - **Registered user** can `get`/`put` file and `mkdir` in `public/**/*`
  - **Registered user** can only `rm`/`rmdir` **file/dir owned by them** in `public/**/*`
  - **anonymous** can only `get` file in `public/**/*` (readonly)
  - **Administrator** can `get`/`put`/`mkdir`/`rm`/`rmdir` **all content** in `public/**/*`
  - Every uploaded file should **remove** others' Read/Write/Execute permissions

NOTE1: `public/**/*` ➡ all file under `public/`, includes its **subdirctories**
NOTE2: You may assume that registered user will **NOT** try to `rm`/`put` file under **subdirctories** of public **owned by other**

HW2-1: Requirements (4/4)

- `$SFTP_ROOT/private`
  - Create directory "`hidden/`" under `private/`
  - Put a file "`treasure`" under `hidden/`
  - **Registered user** and **anonymous:**
    - **cannot** perform `ls` in `private/**/*`
    - **cannot** write (`put`/`mkdir`/`rm`/`rmdir`) in `private/**/*`
    - can `cd` into `private/**/*`
    - can `get` "`treasure`" in `private/hidden/`
  - **Administrator** can `get`/`put`/`mkdir`/`rm`/`rmdir` **all content** in `private/**/*`

NOTE1: `private/**/*` ➡ all file under `private/`, includes its **subdirctories**

HW2-1: Quick reference

✔️	❌	⚠️
Allowed	Not allowed	only allowed for file owner itself

	sysadm		sftp-u1/2		anonymous	
	public/	hidden/	public/	hidden/	public/	hidden/
ls	✔️	✔️	✔️	❌	✔️	❌
mkdir	✔️	✔️	✔️	❌	❌	❌
rmdir	✔️	✔️	⚠️	❌	❌	❌
put	✔️	✔️	✔️	❌	❌	❌
get	✔️	✔️	✔️	✔️	✔️	✔️
rm	✔️	✔️	⚠️	❌	❌	❌

HW2-1: Grading

- SFTP login only permission & chroot (5%)
- Sysadm access:
  - public (4%) private (3%)
- Sftp-u1/2 access:
  - public (4%) private (4%)
- anonymous access
  - public (3%) private (2%)
- Remove other's permission on uploaded file (2%)

HW2-1: Hint

- sshd_config(5)
- sftp-server(8)
- chmod(1)
- You can make following assumptions:
  - **registered user** will never `put`/`rm` file under subdirctories.
    - unless the subdirctories is owned by them

(2-2) (25%)
SFTP logging
and Monitoring daemon

HW2-2: Overview

- Enable **logging SFTP service** with both **journald** and **syslog**
- Program a **daemon** to **monitor and logging** suspicious action of SFTP user

HW2-2: Requirements (1/5)

- Enable **SFTP logging**, and user should be able to **access the log** via:
  - `journalctl -t internal-sftp -t sftp-server`
  - `cat /var/log/sftp.log`
- The log should be **pure SFTP log**
  - can' t blend with log from other service (e.g. SSH, sudo…)

```
$ sudo journalctl -t sftp-server -t internal-sftp | tail -n 2
Oct 08 18:42:58 sa2025 sftp-server[1979]: session closed for local user sysadm from [10.0.2.2]
Oct 08 18:43:27 sa2025 internal-sftp[2029]: session opened for local user sftp-u1 from [10.0.2.2]
$ cat /var/log/sftp.log | tail -n 2
Oct 08 18:42:58 sa2025 sftp-server[1979]: session closed for local user sysadm from [10.0.2.2]
Oct 08 18:43:27 sa2025 internal-sftp[2029]: session opened for local user sftp-u1 from [10.0.2.2]
```

HW2-2: Requirements (2/5)

- Write a daemon that would watch on **every file SFTP user uploaded.**
  - Named as **sftp_watchd**.
  - The program can be written in any language (e.g. B, C, C++, C#, D, Python)
- **sftp_watchd** should reside in your system `$PATH`
- Requirement of **sftp_watchd**:
  - Checking if any **uploaded file violate the rule**
  - If found, the program should **move** it into `${SFTP_ROOT}/private/.violated/`
    - And **logging** it with specified format (detail in page 18)

HW2-2: Requirements (3/5)

- File violation rule:
  a. Uploading **ELF** (Executable and Linkable format) file is prohibited
     - You may only check the first 4 bytes to identify a ELF file
  b. Uploading **file with specific MD5 hash** is prohibited
     - Prohibited MD5 hash list:

```
209c6ec9c78249031b49b29aef2ee264
288d9c9c945b95bcca9632a594f8ebfc
84cbc60c4b110591d7c287da21067e70
d214f689364c2c19bd02aeb087a354e2
8806b1882ec4ee5f88f4e11641965285
```

HW2-2: Requirements (4/5)

- Logging message format:
  a. ELF file

```
File {file} uploaded by user {user} is an ELF file. Moving to violated directory
```

  b. Prohibited MD5

```
File {file} uploaded by user {user} matches prohibited MD5 hash. Moving to violated directory
```

HW2-2: Requirements (5/5)

- Make **sftp_watchd** a **systemd service**
  - Writing a service unit file
- Enabling control **sftp_watchd** through `systemctl`
  - Should support start, stop, status, restart
- The log of **sftp_watchd** should be **accessed through** `journalctl`
  - `journalctl -u sftp_watchd`

HW2-2: Grading

- SFTP logging (6%)
- sftp_watchd functionality (12%)
- Managed sftp_watchd through systemd (7%)

HW2-2: Hint

- Filter - Rsyslog documentation
- SFTP chroot - Archwiki

(2-3) (48%)
BTRFS on LVM
And BTRFS snapshot

HW2-3: Overview

- Setup **LVM**, and setup **RAID10** with LVM
- Make **BTRFS** on RAID10 LVM
- Manage BTRFS with **flatten layout**
- Write a **SnApper** script to **manage BTRFS snapshot**
  - create, list, delete, rollback

HW2-3: Requirements (1/11)

- Add at least one disk
- **Make 4 PV** from added disks and **group** these 4 PV into a VG
  - named the VG as **SA2025vg**
- **Create a LV** on top of VG
  - named the LV as **SARaid10**
  - Making the LV as **Raid10**
- **Make BTRFS filesystem** on `/dev/mapper/SA2025vg-SARaid10`
- Set the **Label** of the BTRFS filesystem to `SAHW2`

HW2-3: Requirements (2/11)

- **All** of your **subvolume** should **directly under root volume**
  - root voulme: subvolume id = 5
  - **No nested subvolume is allowed** (subvolume under other subvolume)
- Create following **subvolumes** under root volume:
  - **root**, **sftp**, **pool1**, **pool2**
- Create following **directories** under **root volume**:
  - `snapshot/sftp`, `snapshot/pool1`, `snapshot/pool2`

NOTE1: **root** is a **subvolume named root.** While **root volume** is volume **with id 5.** They are different.

HW2-3: Requirements (3/11)

- Mount subvolumes according to the table
  - All the mounting **source device** should be `/dev/mapper/SA2025vg-SARaid10`
  - All the mounting should be **persistent**
    - i.e. in `/etc/fstab`

Subvolume name	Mount target
root	`/mnt/hw2`
sftp	`/mnt/hw2/sftp`
pool1	`/mnt/hw2/pool1`
pool2	`/mnt/hw2/pool2`

NOTE1: **root** is a **subvolume named root.** While **root volume** is volume **with id 5.** They are different.

HW2-3: Requirements (4/11)

- Write a BTRFS snapshot management script. Named as **SnApper**
- Implement following function:
  - `Create`: creating a snapshot for a subvolume
  - `List`: listing all snapshot managed by this script
  - `Delete`: remove a snapshot from the system
  - `Rollback`: Apply a snapshot onto subvolume
- The **SnApper** should resides in your system $PATH
- You are only allowed to use **Bash** (or sh) to implement **SnApper**

HW2-3: Requirements (5/11)

- SnApper should show following help message

```
Usage:
SnApper [-h]: show this message
SnApper snapshot SUBVOL [-c ROTATION_COUNT]
SnApper list [-p SUBVOL] [-i ID]
SnApper delete [ID]
SnApper rollback ID
```

HW2-3: Requirements (6/11)

- **`SnApper snapshot`** *`SUBVOL [-c ROTATION_COUNT]`*
  - create a snapshot for *`SUBVOL`*
  - The created snapshot should also be a subvolume **directly under root volume**
    - The snapshot should placed in directory `<ROOT_VOLUME>/snapshot/<SUBVOL>` and named as `@<YYYYMMDD-hhmmss>`

```
$ sudo SnApper snapshot pool1
Snap 'snapshot/pool1/@20251003-163650' [551]
$ sudo btrfs subvolume list /mnt/hw2/ -p -a -t
ID      gen     parent  top level       path
--      ---     ------  ---------       ----
...
551     671     5       5               <FS_TREE>/snapshot/pool1/@20251003-163650
```

HW2-3: Requirements (7/11)

- After create, make sure snapshot of *SUBVOL* doesn't **exceed** *ROTATION_COUNT*
  - if *-c* is not provided, the rotation count is **default to 5**
  - if exceed, **delete** snapshot with **smallest subvolume ID**
    - repeat until snapshot count <= *ROTATION_COUNT*
- The command should **output on success**:
  - `Snap '<SNAPSHOT_PATH>' [<ID>]`
- Created snapshot should be **readonly**

HW2-3: Requirements (8/11)

- **`SnApper list`** *`[-p SUBVOL] [-i ID]`*
  - If no *`-p`* and *`-i`* provided, **listing all snapshot** managed by **SnApper**
  - If *`-p`* is provided, only show snapshot of *`SUBVOL`*
  - If *`-i`* is provided, only show snapshot with that *`ID`*
  - *`-p`* and *`-i`* can be both provided

```
$ sudo SnApper list
ID    SUBVOLUME  TIME
549   pool1      2025-10-02 20:57:50
553   pool2      2025-10-03 17:23:17
$ sudo SnApper list -i 549
ID    SUBVOLUME  TIME
549   pool1      2025-10-02 20:57:50
$ sudo SnApper list -i 549 -p pool2
ID    SUBVOLUME  TIME
```

HW2-3: Requirements (9/11)

- **`SnApper delete`** *`[ID]`*
  - delete the snapshot with provided *`ID`*
  - if *`ID`* is not provided, **delete all snapshot**
- The command should output on success:
  - `Destroy ID <ID>`

```
$ sudo SnApper delete 549
Destroy ID 549
$ sudo SnApper delete
Destroy ID 554
Destroy ID 555
Destroy ID 556
```

HW2-3: Requirements (10/11)

- **`SnApper rollback` *`ID`***
  - Rollback to snapshot specified by *`ID`*
  - *`ID`* must be provided
  - You should automatically find corresponding subvolume of the snapshot
- The command should output on success
  - `Rollback '<SNAPSHOT_PATH>' [<ID>] to <SUBVOL>`

```
$ sudo SnApper list
ID    SUBVOLUME  TIME
557   pool2      2025-10-03 19:45:29
$ sudo SnApper rollback 557
Rollback 'snapshot/pool2/@20251003-194529' [557] to pool2
```

HW2-3: Requirements (11/11)

- Requirements for rollback:
  - The snapshot should **still exist** after rollback
  - The subvolume **should be writable** after rollback
  - The rollbacked subvolume **state** should be **persist after umount/re mount**

HW2-3: Grading

- BTRFS-setup (12%)
  - BTRFS on LVM
  - subvolume structure
  - mounting layout
- LVM-Raid10 (5%)
- Snapper-Create (8%)
- Snapper-List (7%)
- Snapper-Delete (5%)
- Snapper-Rollback (11%)

HW2-3: Hint

- mount(8)
- fstab(5)
- LVM - Archwiki
- BTRFS - Archwiki

Attention

- Deadline: 11/17 (Mon.) 23:59
- Your work will be scored by Online Judge system
  - Only the **LAST submission** will be **scored**
  - Late submission will **NOT** be accepted
- We will fetch your script from `$(which sftp_watchd)` and `$(which SnApper)`
  - Make sure your script is readable by **judge** user from these path
  - Also, your script must not invoke any other self-written scripts, binaries or executables.

Attention

- **ALWAYS BACKUP** your system before submission
  - We may do malicious actions (e.g. `dd if=/dev/zero of=/dev/sda`)
- TAs reserve the right of final explanations.
  - Specs and the points of each subjudges are subject to change in any time.(with notification)
- Make sure everything works after reboot