Ćwiczenie 7
SSH i zdalna administracja

7.1. Rozpoczęcie pracy

1. Otworzyć terminal lokalny i sprawdzić dostępność narzędzi SSH oraz Bash:

```
ssh -V
command -v ssh-keygen
command -v scp
command -v bash
git --version
```

2. Odczytać od prowadzącego (lub od kolegi, w wariancie poniżej) i zapisać lokalnie cztery parametry: `SSH_HOST`, `SSH_PORT`, `SSH_USER` oraz odcisk (*fingerprint*) klucza hosta.
Wariant w parach (serwer na komputerze kolegi obok): Zamiast serwera prowadzącego można wykorzystać komputer kolegi (lub jego maszynę wirtualną), jeżeli urządzenia mogą komunikować się w sieci lokalnej, a jego właściciel i administrator pracowni wyrazili zgodę. Na komputerze serwera jego właściciel wykonuje:

```
sudo apt install openssh-server
sudo systemctl start ssh
sudo adduser ndp-guest
hostname -I
ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub
```

   Zanotować adres IP właściwego interfejsu i odcisk klucza hosta. Na **drugim komputerze** po sprawdzeniu odcisku podczas łączenia:

```
ssh ndp-guest@ADRES_IP
```

   Do dalszych punktów przyjąć `SSH_HOST=ADRES_IP`, `SSH_PORT=22` oraz `SSH_USER=ndp-guest`. Używać oddzielnego konta bez `sudo` i skonfigurować na nim własny klucz SSH.

3. Pliki do ćwiczenia pobrać ze strony z materiałami jako pakiet `starter/` (archiwum ZIP) lub z załączników niniejszego PDF. Pakiet ma zawierać trzy pliki: `dane.txt`, `process.sh` i `deploy.sh`. Rozpakować archiwum, skopiować pliki do `~/cw7/starter` i sprawdzić ich obecność. Użyć panelu załączników albo kliknąć nazwę pliku: `dane.txt`, `process.sh`, `deploy.sh`. Jeżeli używana aplikacja do pdf nie obsługuje załączników, pobrać archiwum ZIP. Wszystkie trzy pliki umieścić obok siebie.

```
mkdir -p ~/cw7/starter
cd ~/cw7/starter
```

```
ls -la
cat dane.txt
```

Dane wejściowe mają cztery linie: `alfa`, `beta`, `gamma` i `delta`. Nie zmieniać pliku `dane.txt` na początku ćwiczenia.

7.2. Połączenie i tożsamość zdalnego serwera

4. Zdefiniować w **lokalnym** terminalu zmienne, zastępując przykładowe wartości danymi od prowadzącego:

```
export SSH_HOST='serwer.example.org'
export SSH_PORT='22'
export SSH_USER='student'
```

Nie zapisywać hasła do serwera w zmiennej ani w skrypcie.

5. Nawiązać połączenie za pomocą:

```
  ssh -p "$SSH_PORT" "$SSH_USER@$SSH_HOST"
```

Nie stosować opcji `StrictHostKeyChecking=no`.

6. Na serwerze wyświetlić nazwę komputera, użytkownika, katalog domowy i identyfikatory konta:

```
hostname
whoami
id
pwd
uname -a
```

Zakończyć sesję przez `exit`

7. Wykonać pojedyncze polecenie na serwerze (bez uruchamiania interaktywnej powłoki):

```
ssh -p "$SSH_PORT" "$SSH_USER@$SSH_HOST" \
    'printf "Uzytkownik: "; whoami; hostname'
```

Porównać wynik z poprzednim punktem i sprawdzić kod zakończenia poleceniem `echo $?`.

7.3. Uwierzytelnianie kluczem publicznym

8. Sprawdzić, czy istnieje katalog `~/.ssh`, oraz jego uprawnienia:

```
mkdir -p ~/.ssh
chmod 700 ~/.ssh
ls -ld ~/.ssh
```

Używać **osobnej pary kluczy do ćwiczenia**; nie nadpisywać posiadanych kluczy.

9. Wygenerować parę kluczy typu Ed25519, nadając kluczowi prywatnemu hasło ochronne (*passphrase*):

```
ssh-keygen -t ed25519 \
    -f "$HOME/.ssh/ndp_lab07_ed25519" \
    -C "ndp-lab07"
ls -l ~/.ssh/ndp_lab07_ed25519*
```

Wskazać, który z dwóch plików wolno przekazać na serwer, a którego nie wolno kopiować, publikować, ani przesyłać do repozytorium Git.

10. Wyświetlić klucz **publiczny** i odcisk własnej pary:

```
cat ~/.ssh/ndp_lab07_ed25519.pub
ssh-keygen -lf ~/.ssh/ndp_lab07_ed25519.pub
```

Dodać klucz publiczny do konta na serwerze. Jeżeli serwer pozwala na wcześniejsze logowanie hasłem i dostępne jest narzędzie `ssh-copy-id`, można użyć:

```
ssh-copy-id -i ~/.ssh/ndp_lab07_ed25519.pub \
    -p "$SSH_PORT" "$SSH_USER@$SSH_HOST"
```

Jeżeli metoda nie jest udostępniona, poprosić administratora o dodanie klucza do `~/.ssh/authorized_keys` **własnego konta**.

11. Sprawdzić dostęp kluczem i uprawnienia po stronie serwera:

```
ssh -i ~/.ssh/ndp_lab07_ed25519 \
    -p "$SSH_PORT" "$SSH_USER@$SSH_HOST" \
    'ls -ld ~/.ssh; ls -l ~/.ssh/authorized_keys'
```

Typowe uprawnienia to `700` dla katalogu `~/.ssh` i `600` dla `authorized_keys`. Nie modyfikować cudzych kont.

12. W lokalnym pliku `~/.ssh/config` dodać wpis (bez nadpisywania istniejącej konfiguracji):

```
Host ndp-serwer
    HostName serwer.example.org
    User student
    Port 22
    IdentityFile ~/.ssh/ndp_lab07_ed25519
    IdentitiesOnly yes
```

Zastąpić przykłady parametrami prowadzącego, wykonać `chmod 600 ~/.ssh/config` oraz przetestować `ssh ndp-serwer`.

7.4. Uprawnienia, pliki i przesyłanie danych

13. Poleceniem `ssh ndp-serwer` przejść na serwer. Utworzyć katalog roboczy w swoim katalogu domowym, ustalić prywatne uprawnienia i przeanalizować wynik:

```
umask 077
mkdir -p ~/ndp-lab07
chmod 700 ~/ndp-lab07
ls -ld ~/ndp-lab07
```

14. Na komputerze lokalnym, z katalogu `~/cw7/starter`, skopiować dwa pliki na serwer:

```
cd ~/cw7/starter
scp dane.txt process.sh ndp-serwer:ndp-lab07/
```

Przypomnieć sobie różnicę między opcją `-p` w programie `ssh`, a opcją `-P` w programie `scp` przy ręcznym podawaniu niestandardowego portu.

15. Wyświetlić metadane i zawartość przesłanych plików bezpośrednio na serwerze:

```
ssh ndp-serwer 'ls -la ~/ndp-lab07; \
    wc -l ~/ndp-lab07/dane.txt; \
    head -n 4 ~/ndp-lab07/dane.txt'
```

Przeanalizować działanie praw `r`, `w`, `x` oraz rolę poleceń `chmod`, `stat` i `umask`.

16. Przesłać pojedynczy plik również przez strumień SSH, bez korzystania z `scp`:

```
printf 'Laboratorium 7\n' | \
    ssh ndp-serwer 'cat > ~/ndp-lab07/info.txt'
ssh ndp-serwer 'cat ~/ndp-lab07/info.txt'
```

7.5. Uruchamianie skryptów i automatyzacja

17. Przejrzeć zawartość `process.sh`. Ustalić, jakie argumenty przyjmuje skrypt, co oznacza `set -euo pipefail` i w jakich sytuacjach program zakończy się kodem różnym od zera.

18. Zdalnie wykonać przetwarzanie danych z użyciem interpretera Bash:

```
ssh ndp-serwer \
'bash ~/ndp-lab07/process.sh \
  ~/ndp-lab07/dane.txt ~/ndp-lab07/result.txt'
```

Sprawdzić istnienie pliku wynikowego poleceniem `ssh ndp-serwer 'cat ~/ndp-lab07/result.txt'`. Oczekiwany napis to `Liczba_wierszy=4`.

19. Pobrać wynik na komputer lokalny i sprawdzić jego zawartość:

```
scp ndp-serwer:ndp-lab07/result.txt .
cat result.txt
grep -Fx 'Liczba_wierszy=4' result.txt
```

Zastanowić się jakie jest znaczenie kodu zakończenia polecenia `grep`.

20. Otworzyć szkielet `deploy.sh`. Prześledzić parametry `SSH_USER`, `SSH_HOST`, `SSH_PORT`, `SSH_KEY`, `REMOTE_DIR` i `SCRIPT_DIR`. Uzupełnić cztery fragmenty oznaczone `TODO` tak, aby skrypt tworzył katalog, kopiował pliki, uruchamiał przetwarzanie i pobierał wynik. Nie wpisywać do niego haseł, ani danych dostępowych.

21. Uruchomić uzupełniony skrypt z terminala lokalnego:

```
cd ~/cw7/starter
export SSH_KEY="$HOME/.ssh/ndp_lab07_ed25519"
bash -n deploy.sh
rm -f result.txt
```

```
# Sprawdzamy ponowne wdrożenie bez usuwania zdalnego wyniku.
bash deploy.sh
echo $?
grep -Fx 'Liczba_wierszy=4' result.txt
```

Pamiętać o ustawieniu zmiennych `SSH_HOST`, `SSH_USER`, `SSH_PORT` z punktu 4.
W skrypcie użyć osobnych tablic opcji `ssh` i `scp`, w tym `IdentitiesOnly=yes`; klient `scp` przyjmuje wielkie `-P` dla portu. Usuwany jest tylko lokalny plik wynikowy. Zdalnego wyniku nie usuwać przed ponownym wdrożeniem. Jeśli stosowany jest alias `ndp-serwer`, skrypt powinien nadal działać także na podstawie niezależnych parametrów środowiskowych.

22. Uruchomić `deploy.sh` **ponownie**. Zweryfikować, że nie powstają zbędne kopie danych, wynik pozostaje prawidłowy i skrypt nie wymaga ingerencji operatora poza ewentualnym odblokowaniem klucza. Następnie wykonać kontrolowany test błędu, wskazując nieistniejący plik klucza przez `SSH_KEY=/brak/klucza bash deploy.sh`. Sprawdzić, czy skrypt kończy się niezerowym kodem, bez modyfikowania danych na serwerze.

7.6. Podstawowa diagnostyka serwera Linux

23. Korzystając z połączenia SSH, wykonać na przydzielonym serwerze tylko operacje odczytu:

```
ssh ndp-serwer 'uptime; date; df -h "$HOME"'
ssh ndp-serwer 'ps -u "$USER" -o pid,etime,comm'
```

Obciążenie systemu odczytać przez `uptime`; miejsce zajmowane przez katalog `~/ndp-lab07` sprawdzić za pomocą `du -sh ~/ndp-lab07`. Nie uruchamiać testów obciążeniowych na serwerze współdzielonym.

24. Na serwerze przeanalizować pamięć i procesy poleceniami `free -h`, `top` (w sesji interaktywnej lub przez `ssh -t ndp-serwer top`, zakończyć klawiszem `q`) i `ps`. Sprawdzić, czy na komputerze zdalnym działają te same wersje narzędzi co lokalnie. Gdy polecenie nie istnieje, zanotować różnicę bez instalowania oprogramowania na cudzym serwerze.

25. Sprawdzić stan usługi SSH za pomocą `systemctl status ssh --no-pager` albo `systemctl status sshd --no-pager` albo `systemctl status ssh.socket --no-pager`, zależnie od dystrybucji i aktywacji gniazdem. Jeżeli dostęp do szczegółów lub logów jest ograniczony, poprzestać na dozwolonych informacjach. Zapoznać się z poleceniem `journalctl -u ssh -n 15 --no-pager`.

26. Utworzyć w bieżącym **lokalnym** katalogu log wykonania zdalnego zadania, rozdzielający standardowe wyjście i błędy:

```
ssh ndp-serwer \
'date; bash ~/ndp-lab07/process.sh \
  ~/ndp-lab07/dane.txt ~/ndp-lab07/result.txt' \
> uruchomienie.log 2> uruchomienie.err
wc -c uruchomienie.log uruchomienie.err
```

7.7. Administracja własnym serwerem

27. Jeżeli prowadzący udostępnił **maszynę wirtualną** z prawami administratora, sprawdzić jej stan i automatyczny start usługi SSH:

```
sudo systemctl status ssh --no-pager
sudo systemctl is-enabled ssh.socket
sudo ss -tlnp 'sport = :22'
```

Na dystrybucjach bez aktywacji gniazdem sprawdzić `ssh.service` lub `sshd.service` i odpowiednio zmienić nazwę jednostki. Niestandardowy port uwzględnić w filtrze `ss`. Jeśli serwer nie posiada `systemd`, odszukać właściwy dla dystrybucji mechanizm obsługi usług.

28. Na własnej VM skontrolować konfigurację usługi bez jej zmieniania: `sudo sshd -t` (w zależności od dystrybucji pełna ścieżka do `sshd`), `sudo systemctl cat ssh` i `sudo journa lctl -u ssh -n 20 --no-pager`. **Nie wyłączać logowania hasłem, ani nie zmieniać zapory bez zachowania działającej sesji administracyjnej.**

29. Na własnej VM wykonać tunelowanie lokalnego portu na przykładową usługę HTTP dostępną jedynie pod adresem `127.0.0.1:8000` serwera. Najpierw uruchomić tam w **osobnej sesji** testowy serwer:

```
mkdir -p ~/ndp-web
printf 'SSH tunnel OK\n' > ~/ndp-web/index.html
cd ~/ndp-web && python3 -m http.server 8000 --bind 127.0.0.1
```

`HOST_VM` oznacza alias własnej VM z pliku `~/.ssh/config`. Na komputerze lokalnym uruchomić `ssh -N -o ExitOnForwardFailure=yes -L 8080:127.0.0.1:8000 HOST_VM` i otworzyć `http://localhost:8080/`. Zakończyć obie sesje przez `Ctrl+C`. W przypadku zajętego portu lokalnego wybrać inny.

7.8. Zadania dodatkowe

30. **Wykonać kontrolę tożsamości serwera.** Na własnej VM odczytać odcisk klucza hosta poleceniem `ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub`. Porównać z komunikatem klienta przy pierwszym połączeniu. Następnie wyjaśnić, dlaczego nie należy pomijać sprawdzenia odcisku.

31. **Wdrożenie odporne na błędy.** Uzupełnić skrypt o sprawdzanie dostępności klucza przed połączeniem, kontrolę kodów wyjścia i wykrywanie braku wyniku. Pokazać poprawne uruchomienie i jeden bezpieczny test błędu bez zmiany danych wejściowych.

7.9. Zakończenie ćwiczenia

32. Przedstawić prowadzącemu działające połączenie SSH, katalog prywatny na serwerze, skrypt `deploy.sh` oraz wynik `result.txt` zawierający `Liczba_wierszy=4`. Do oddania `deploy.sh` i `result.txt` (zgodnie z istniejącym materiałem `lab-07`). Nie oddawać klucza prywatnego, pliku `~/.ssh/config`, haseł ani logów ujawniających informacje o koncie. Zachować lokalne pliki robocze (będą potrzebne w ćwiczeniu 8) lub wykonać ich kopię. Po zakończeniu usunąć własne dane uwierzytelniające i katalogi robocze.