Individuell examinationsuppgift: Automatisering, säkerhetskonfiguration och loggning på Linux-server

Daniel Wagenius
ITSX25
Chas Academy

Del 1 – Automatisering av rutinuppgifter med Bash, tre olika skript

Skript 1
Mitt skript består av två separata funktioner. En för att användaren ska kunna avbryta när som helst och en för att kolla att lösenordet är tillräckligt starkt.
När en användare skapas får den begränsade rättigheter, som önskat. Jag lägger även in användaren i en speciell grupp för att göra det lättare att ta bort alla skapade användare med ett annat skript.
Lösenordet måste vara 8 tecken, ha minst en stor bokstav, liten bokstav, siffra och specialtecken. Använde skriptet från en tidigare uppgift till det, med lite modifikationer.
Skript 2
Det mitt skript kan göra är visa öppna portar, öppna och stänga port 80, och skriva ut min loggfil. Jag loggar i en separat fil som jag bara har till den här uppgiften. När port 80 öppnas startas även en Python-server som lyssnar på port 80. Utan den syns inte port 80 när jag listar öppna portar.
Skript 3
Det enklaste skriptet. Tror jag kan ha överarbetat de första två. Dålig vana.
Skriptet skapar en backup-folder om den inte finns. Sedan kopieras den nuvarande loggfilen till backup-foldern och datum och tid läggs in på slutet av filnamnet.
Backuperna finns kvar i 180 dagar, sedan tas de bort.
Skriptet körs med cron varje natt klockan 2.

Del 2 – Simulerad tvånodsmiljö (på en server)

Frustrerande och förvirrande uppgift och jag blandade ihop den med skript 2 i uppgift 1. Att skriva brandväggsregler känns klumpigt. Jag vet inte vad uppgiften menar med ”visa hur du identifierade IP-adresser”. Är det inte bara kommandot *ip a*? För att ställa in två IP-adresser till zonerna external och internal gjorde jag i alla fall så här (mitt Office gjorde om två minusstreck till em-dashes):
sudo ip addr add 172.20.1.50/24 dev eth0
sudo ip addr add 172.20.2.50/24 dev eth1
sudo firewall-cmd —zone=external —add-source=172.20.1.50/32 —permanent
sudo firewall-cmd —zone=internal —add-source=172.20.2.50/32 —permanent
sudo firewall-cmd —reload
Det är /24 när jag skapar IP-adresserna för att det är subnet-masken 255.255.255.0. När jag ställer in zoner vill jag ha precis *den* IP-adressen. Man skapar ett nytt litet nätverk först, och sen använder man en specifik IP-adress.
För loggningen skapade jag filen */etc/rsyslog.d/firewall.conf* och jag använder den även till skript 2 ovan:
:msg, contains, ”IN_” /var/log/firewall.log
:msg, contains, ”firewalld” /var/log/firewall.log
:msg, contains, ”FINAL_FIREWALL_LOG” /var/log/firewall.log
:programname, isequal, ”firewalld” /var/log/firewall.log
:programname, isequal, ”firewall” /var/log/fwconfig.log
& stop
I själva skriptet använder jag sedan logger med taggen ”firewalld”:
logger –t firewalld ”Meddelande till loggen”
I skript 2 ovan testloggar jag med bara ”firewall” i den raden. När konfigurationen är skriven måste man starta om rsyslog. Då kommer meddelanden från logger hamna i min log-fil, */var/log/fwconfig.log*.
Kod och skärmdumpar på testerna ligger i bilagor.

Del 3 – Grundläggande CIS Controls och härdning

Tre kontroller:
CIS Control 2: Inventory and Control of Software Assets
CIS Control 4: Secure Configuration of Enterprise Assets and Software
CIS Control 5: Account Management

Konkreta åtgärder:
CIS Control 2:
2.1 Establish and Maintain a Software Inventory: Kolla vad jag faktiskt har installerat på servern.
2.2 Ensure Authorized Software is Currently Supported: Uppdatera det som finns på servern.
2.3 Address Unauthorized Software: Den här bör göras innan 2.2. Ta bort det som inte ska vara där.
CIS Control 4:
4.4 Implement and Manage a Firewall on Servers: Ställ in brandväggen. Installera eller skriv skript som hjälper till att stoppa intrångsförsök.
CIS Control 5:
5.2 Use Unique Passwords: Stora starka lösenord.

Vad jag gjort och varför:
CIS Control 2, integritet:
Har kört *sudo dnf list installed* och gått igenom listan. Sen vi fick servrarna har jag varit noga med att inte installera eller ladda ner något jag verkligen inte behöver (*reduction of attack surface*, och jo neofetch är oerhört viktigt), och jag ser inget konstigt i listan. Jag behöver därför inte använda mig av 2.3 (*least privilege* på sätt och vis) just nu. Därefter körde jag *sudo dnf upgrade* för att uppdatera allt som är installerat.

CIS Control 4, konfidentialitet, tillgänglighet:
I våra uppgifter har fått ställa in brandväggen på lite olika sätt, och den bör stoppa allt som inte ska komma igenom nu. Jag har, än så länge, inte lyckats låsa mig ute. Förhoppningsvis fortsätter det.
En av de första sakerna jag gjorde var att installera fail2ban (*defense in depth*) och ställa in det att stänga av en IP i 1 timme efter 5 misslyckade försök (*logging and monitoring*). Tiden är nog en aning låg så jag kan behöva justera inställningarna. Man kan även öka tiden för IP-adresser som fortsätter försöka ta sig in.

CIS Control 5, konfidentialitet:
Jag har just nu ett lösenord (*defense in depth*) med en entropi på en bra bit över 90 bitar, vilket borde vara tillräckligt för att skydda min server mot det mesta.

Viktigaste lärdomar

- Skriva brandväggsregler är fruktansvärt tråkigt och kan vara bökigt. Men jag lyckades inte låsa ute mig i alla fall.
- När man har en server på en stor leverantör som Hetzner så är det många som är nyfikna och skannar och försöker ta sig in. Viktigt att ha skydd.
- Förutom brandväggsinställningar är det mesta man kan göra för att skydda sig väldigt enkelt.
- UFW är betydligt enklare och räcker för oss studenter, men gissar att firewalld används av de mer seriösa.

Bilagor

createuser.sh
Skapar en användare och sätter ett starkt lösenord.

```
!/bin/bash

####################################
# Skript för att skapa en användare #
####################################

# Funktion för att avsluta skriptet när som helst
quit_check() {
  if [[ "$1" == [qQ] ]]; then
    clear
    echo "Quitting the script. Thank you for using the buggy service."
    exit 0
  fi
}

# Funktion för att kontrollera lösenordet
password_check() {
  echo "Now create a password." >&2 # >&2 skickar "echo" till error handling,
kunde lika gärna skickas till /dev/null
  echo "Minimum 8 characters, at least one number, one uppercase letter, one
lowercase letter and one special character." >&2

# Måste använda >&2 efter varje "echo" i den här funktionen,
# annars skickar "echo" tecken till variabeln man skapar

  # Dubbelkollar lösenordet
  while true; do
    read -rsp "Enter password: " password
    quit_check "$password"
    echo >&2
    read -rsp "Confirm password: " password2
    quit_check "$password2"
    echo >&2

    # Kollar att lösenordet är starkt nog
    # Det blir lite konstigt när man skriver in
    # lösenordet två gånger innan man kollar
    # styrkan, men det får vara så
    if [ "$password" == "$password2" ]; then
      points=0
      # Här går det lägga till output till användaren om vad lösenordet
missar på
      if [[ ${#password} -ge 8 ]]; then ((points++)); fi # Minst 8 tecken
långt
      if [[ $password =~ [A-Z] ]]; then ((points++)); fi # Minst en stor
bokstav
      if [[ $password =~ [a-z] ]]; then ((points++)); fi # Minst en liten
bokstav
      if [[ $password =~ [0-9] ]]; then ((points++)); fi # Minst en siffra
      if [[ $password =~ [^A-Za-z0-9] ]]; then ((points++)); fi # Minst ett
specialtecken

      if [[ $points -ge 5 ]]; then # Måste få poäng i alla ovanstående
```

```
        echo "Password acceptable." >&2 # Återigen, skickar skräpdata till
error handling
        echo "$password"
        return 0
      else
        echo "Password is bad and you should feel bad. Do it again." >&2 # Om
lösenordet inte är starkt nog
      fi
    else
      echo "Passwords do not match. Please enter the exact same password
twice." >&2 # Om man inte skriver samma lösenord
    fi
  done
}

# Start på skriptet
# Rensar skärmen, tycket det blir snyggt
clear

if [ "$(id -u)" -ne 0 ]; then
    echo -e "This script must be run as root.\n"
    exit 1
fi

echo "Welcome to the user creation script (Bethesda Approved because it is
probably buggy)"
echo "To quit, enter 'q' at any time." # Talar om för användaren att man kan
skriva q när som helst för att avsluta

while true; do # Username-loopen
  while true; do
    read -p "Please enter your desired username: " username
    quit_check "$username"
    if id "$username" &>/dev/null; then # Om användarnamnet redan finns
      echo "Username may not be used." # Talar inte om varför det inte går
att använda för säkerhets skull
    else
      echo "You have entered $username. A brilliant choice."
      break 2
    fi
  done
done
    echo "It has been confirmed."
    echo "Your username has been set as $username."

password=$(password_check) # Får ett bra lösenord av användaren

# echo " "

while true; do # Användaren får bekräfta sitt användarnamn
  read -p "Do you want to create the user $username? (Y/N): " confirm_all
  quit_check "$confirm_all"
  case "$confirm_all" in
    [yY])
      break
      ;;
```

```
    [nN])
      echo "User creation aborted by user." # Avslutar skriptet om användaren
ångrar sig
      exit 1
      ;;
    *)
      echo "Please only reply with 'y' or 'n'."
      ;;
    esac
done

# -m skapar en hemkatalog. Användare vill nog ha en sån. -g sätter användaren
i en specifik grupp
useradd -m -g fakeusers "$username"
# Lägger till användaren ordentligt i gruppen så det är lättare att ta bort
den. Funkar inte annars och det irriterar lite
usermod -aG fakeusers "$username"
echo "$username:$password" | chpasswd # Ändrar lösenordet för användaren
echo "User $username created."
echo "Thank you for using this little script."
# Färdiga!
```

fwconfig.sh
Öppnar och stänger en port, kör igång en Python-server, listar öppna portar och skriver ut senaste loggar.

```
#!/usr/bin/env bash

# Litet skript för att öppna och stänga port 80, visa öppna portar, och
skriva
# ut min hemsnickrade loggfil.
# Case-menyn kommer från Hakim i Boiler Room-gruppen.
# Mycket av det andra kommer från olika övningar vi har gjort,
# samt firewalld-dokumentationen.
# Man behöver ha en tjänst som lyssnar på porten för att den ska synas
# så jag skrev in en python-server i skriptet.
# Använder &>/dev/null för att skicka mycket skräpdata till /null så det
försvinner.
# Det gör att man inte får någon bekräftelse eller några error codes
utskrivet.

# Ställer in variabler för port, timeout (stäng av port automatiskt, loggfil)
port="80"
timeout="300"
logfile="/var/log/fwconfig.log"

# Rensar skärmen så det ser snyggt och prydligt ut
clear

if [ "$(id -u)" -ne 0 ]; then
  echo "This script must be run as root."
  exit 1
fi

# Loopen så man inte kommer ur mitt skript
while true; do
  echo "Meny"
  # Case-meny, bästa sättet att hantera menyer
  select choice in "Show open ports" "Open port 80"\
  "Close port 80" "Print log" "Exit"; do
    case $REPLY in
      1)
        clear
        echo "Open ports: "
        # Visar öppna portar, max 50 men har man så många portar öppna gör
man nog nåt fel
        ss -tuln 2>/dev/null | head -n 50
        break
        ;;
      2)
        clear
        # Frågar systemet om någon lyssnar på porten jag ställt in (80 i det
här fallet)
        # lsof = list open files. -P = bara portnummer, inte namn på service.
        # -i = internet, inte helt klar vad -i gör.
        # -sTCP:LISTEN = filtrerar output att bara visa TCP som lyssnar
(väntar på en inkommande anslutning).
```

```
        # -t = Skriv bara ut process-ID (PID) av processen utan annan
information. "Var kortfattad" ungefär.
        # Den här kollen gör man för att kontrollera om porten är öppen eller
inte.
        if lsof -Pi :"$port" -sTCP:LISTEN -t &>/dev/null; then
          clear
          echo "Port $port already open"
        # Här går man in om porten INTE är öppen.
        else
          # Loggar till min loggfil. Använde ett annat sätt förut, men logger
är så enkelt.
          # -t firewall gör att firewall.conf fångar upp loggen och lägger i
en annan loggfil.
          logger -t firewall "Port $port opened for 5 minutes."
          logger -t firewall "Python web server started for 5 minutes."
          # Öppnar port 80 i 5 minuter.
          firewall-cmd --add-port="$port"/tcp --timeout="$timeout"
          # Startar en python-server.
          nohup python3 -m http.server "$port" --bind 0.0.0.0 >/tmp/http.log
2>&1 &
          P_ID=$!
          clear
          echo "Port 80 is now open for 5 minutes and will close at $(date -d
'+5 minutes' +%H:%M)"
          echo "Python web server with PID $P_ID started."
          # Stänger av python-servern efter 5 minuter, och loggar både att
porten stängs automatiskt
          # och att servern stängs av.
          # Fick sleep-grejen av AI för jag lyckades inte luska ut hur man
gjorde.
          ( sleep "$timeout" && kill "$P_ID" 2>/dev/null && \
          firewall-cmd --remove-port="$port"/tcp 2>/dev/null && \
          logger -t firewall "Server $P_ID shut down automatically." && \
          logger -t firewall "Firewall rule expired automatically.") &
        fi
        break
        ;;
      3)
        # Får ett process-ID från port 80. Om port 80 är stängd och ingen
tjänst lyssnar kommer variabeln vara null.
        server_P_ID=$(sudo lsof -t -i :"$port" -sTCP:LISTEN)
        if [ -n "$server_P_ID" ]; then
          # Loggar att servern stängts ner av användaren.
          logger -t firewall "Server PID $server_P_ID killed by user."
          kill "$server_P_ID" 2>/dev/null
          clear
        fi
        # Kollar om porten är öppen.
        if firewall-cmd --query-port="$port"/tcp &>/dev/null; then
          # Loggar att porten stängts av användaren.
          logger -t firewall "Firewall rule removed by user."
          # Tar bort regeln att porten är öppen.
          firewall-cmd --remove-port="$port"/tcp
          clear
          echo "Port $port has been closed. Much like Toys'R'us."
          echo "Also made sure the Python server was destroyed."
```

```
        # Annars är porten inte öppen.
        else
          clear
          echo "Port $port is not open."
        fi
        break
        ;;
      4)
        clear
        echo "Prints the last 50 log entries"
        tail -n 50 "$logfile" 2>/dev/null
        echo "End of log file."
        break
        ;;
      5)
        clear
        echo "Exited script. Thanks for breaking me."
        exit 0
        ;;
      *)
        echo "Please select a menu option like this was McDonald's, or select
5 to quit."
        ;;
    esac
  done
done
```

logbackup.sh
Gör en backup på logg-filen.

```
#!/usr/bin/env bash

if [ "$(id -u)" -ne 0 ]; then
  echo "Only root may run this script."
  exit 1
fi

logfile="/var/log/fwconfig.log"
backupdir="/var/log/fwconfig_backups"
backupdays=180
timestamp=$(date +%Y%m%d_%H%M%S)
backupfile="$backupdir/fwconfig.log.$timestamp"

# Skapar mappen om den inte finns.
if [ ! -d "$backupdir" ]; then
  mkdir "$backupdir"
fi

cp "$logfile" "$backupfile"

# Kollar exit status på kommandot som körts precis innan den här if-satsen.
# Om 0 så avslutades kommandot korrekt.
if [ $? -eq 0 ]; then
  echo "Log file backup succeeded: $backupfile"
  > "$logfile"
else
  echo "Log file backup failed."
  exit 1
fi

# Hittar och tar bort backupfiler som är äldre än 180 dagar.
find "$backupdir" -type f -name "fwconfig.log.*" -mtime +"$backupdays" -
delete
```

I min crontab finns den här raden för att göra en backup klockan 2 varje natt:

```
0 2 * * * /home/student/logbackup.sh >> /var/log/log_backup.log 2>&1
```

Skärmdumpar

Mina fwconfig.log backups:

```
[student@daniwage ~]$ sudo ls -l /var/log/fwconfig_backups
total 8
-rw-r--r--. 1 root root 3144 Sep 28 14:15 fwconfig.log.20250928_141530
-rw-r--r--. 1 root root    0 Sep 28 14:15 fwconfig.log.20250928_141536
-rw-r--r--. 1 root root  582 Sep 29 02:00 fwconfig.log.20250929_020002
-rw-r--r--. 1 root root    0 Sep 30 02:00 fwconfig.log.20250930_020001
-rw-r--r--. 1 root root    0 Oct  1 02:00 fwconfig.log.20251001_020001
-rw-r--r--. 1 root root    0 Oct  2 02:00 fwconfig.log.20251002_020002
-rw-r--r--. 1 root root    0 Oct  3 02:00 fwconfig.log.20251003_020001
-rw-r--r--. 1 root root    0 Oct  4 02:00 fwconfig.log.20251004_020002
[student@daniwage ~]$
```

fwconfig.sh, visar öppna portar och skriver ut loggen. Värt att notera att filen nollställs efter att en backup gjorts, och därför bör den inte fyllas upp till 50 rader:

```
Open ports:
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:Port
udp   UNCONN 0      0        172.20.1.14:68       0.0.0.0:*
udp   UNCONN 0      0          127.0.0.1:323      0.0.0.0:*
udp   UNCONN 0      0              [::1]:323         [::]:*
tcp   LISTEN 0      5            0.0.0.0:80       0.0.0.0:*
tcp   LISTEN 0      128          0.0.0.0:22       0.0.0.0:*
tcp   LISTEN 0      128             [::]:22          [::]:*
Meny
1) Show open ports
2) Open port 80
3) Close port 80
4) Print log
5) Exit
#?
```

```
Prints the last 50 log entries
Oct  4 09:25:55 daniwage firewall[801468]: Port 80 opened for 5 minutes.
Oct  4 09:25:55 daniwage firewall[801469]: Python web server started for
Oct  4 09:25:59 daniwage firewall[801481]: Server PID 801471 killed by u
Oct  4 09:25:59 daniwage firewall[801484]: Firewall rule removed by user
Oct  4 09:27:24 daniwage firewall[801533]: Port 80 opened for 5 minutes.
Oct  4 09:27:24 daniwage firewall[801534]: Python web server started for
Oct  4 09:27:27 daniwage firewall[801548]: Server PID 801536 killed by u
Oct  4 09:27:27 daniwage firewall[801551]: Firewall rule removed by user
Oct  4 09:27:28 daniwage firewall[801556]: Port 80 opened for 5 minutes.
Oct  4 09:27:28 daniwage firewall[801557]: Python web server started for
Oct  4 09:27:29 daniwage firewall[801568]: Server PID 801559 killed by u
Oct  4 09:27:29 daniwage firewall[801571]: Firewall rule removed by user
Oct  4 10:01:21 daniwage firewall[801911]: Test message to only fwconfig
Oct  4 11:48:48 daniwage firewall[835479]: Port 80 opened for 5 minutes.
Oct  4 11:48:48 daniwage firewall[835480]: Python web server started for
End of log file.
Meny
1) Show open ports
2) Open port 80
3) Close port 80
4) Print log
5) Exit
#?
```