Adventist University of Central Africa

Course Name&Code: COSC 8312

Assignment 2 – Individual

Released date: 24th September,2025

Due Date: September 30, 2025, 11:59 PM (strict deadline) **Total Marks:** 25

Names: HAKIZIMANA Emmanuel

Student Id: 26623

Q1. Compromised system directories

- **/etc** → contains system configuration files (attackers may alter /etc/passwd, /etc/shadow, /etc/ssh/sshd_config).
- **/bin** → holds essential binaries (ls, cp, cat). If replaced, attacker gains persistence.
- **/var** → contains logs (/var/log/auth.log, /var/log/syslog), evidence of intrusion is here.
- **/usr** → non-essential but widely used binaries/libraries; attackers may replace or insert trojans here.
- **/tmp** → temporary files, often writable by everyone, used for privilege escalation.
- **/opt** → optional software; less critical but attackers may hide malicious apps.
- **/boot** → contains kernel & bootloader; modification could allow rootkits.
- **/home** → user files & configs (attackers may install persistence scripts in dotfiles like .bashrc).

Q2. Create this exact structure using the minimum number of commands.

```
sano@MacBook:~$ mkdir -p projects/{client_work/{web/{frontend,backend,database},mobile/{ios,android}},personal/{experime
nts,archive},shared/{templates,resources}}
sano@MacBook:~$ ls
projects  test  text2.txt
```

This single command ensures:

- -p creates parent directories as needed and doesn't error if directories already exist
- Uses brace expansion to create all directories in one command

Q3. Prove your location at each step.

```
sano@MacBook:~/projects/client_work/web/frontend$ cd ../../../personal/experiments && pwd
/home/sano/projects/personal/experiments
sano@MacBook:~/projects/personal/experiments$ cd ../../shared/templates && pwd
/home/sano/projects/shared/templates
sano@MacBook:~/projects/shared/templates$ cd ../../client_work/web/frontend && pwd
/home/sano/projects/client_work/web/frontend
sano@MacBook:~/projects/client_work/web/frontend$
```

Q4. Create a realistic web project structure

```
sano@MacBook:~$ mkdir -p web_project/{html,css,js,backups} && \
cd web_project && \
touch html/{index,about,contact,page_{001..012}}.html && \
touch css/{main,reset,theme_{light,dark},mobile,tablet,desktop,print}.css && \
touch js/{script,script_util,script_config,util,util_config,config}.js && \
touch backups/{{a,b,c,d}{001..005}.html.backup,{a,b,c,d}{001..005}.css.backup,{a,b,c,d}{001..005}.js.backup}
```

```
sano@MacBook:~/web_project/html$ ls
about.html    index.html     page_002.html  page_004.html  page_006.html  page_008.html  page_010.html  page_012.html
contact.html  page_001.html  page_003.html  page_005.html  page_007.html  page_009.html  page_011.html
```

```
sano@MacBook:~/web_project/css$ ls
desktop.css  main.css  mobile.css  print.css  reset.css  tablet.css  theme_dark.css  theme_light.css
```

```
sano@MacBook:~/web_project/js$ ls
config.js  script.js  script_config.js  script_util.js  util.js  util_config.js
```

```
sano@MacBook:~/web_project/backups$ ls
a001.css.backup   a004.html.backup  b002.js.backup    c001.css.backup   c004.html.backup  d002.js.backup
a001.html.backup  a004.js.backup    b003.css.backup   c001.html.backup  c004.js.backup    d003.css.backup
a001.js.backup    a005.css.backup   b003.html.backup  c001.js.backup    c005.css.backup   d003.html.backup
a002.css.backup   a005.html.backup  b003.js.backup    c002.css.backup   c005.html.backup  d003.js.backup
a002.html.backup  a005.js.backup    b004.css.backup   c002.html.backup  c005.js.backup    d004.css.backup
a002.js.backup    b001.css.backup   b004.html.backup  c002.js.backup    d001.css.backup   d004.html.backup
a003.css.backup   b001.html.backup  b004.js.backup    c003.css.backup   d001.html.backup  d004.js.backup
a003.html.backup  b001.js.backup    b005.css.backup   c003.html.backup  d001.js.backup    d005.css.backup
a003.js.backup    b002.css.backup   b005.html.backup  c003.js.backup    d002.css.backup   d005.html.backup
a004.css.backup   b002.html.backup  b005.js.backup    c004.css.backup   d002.html.backup  d005.js.backup
```

This approach uses:

- **1** mkdir **command** with -p to create directory structure
- **4** touch **commands** with brace expansion to create all files efficiently
- **No loops or multiple commands** needed due to smart brace expansion patterns
- **Exactly meets all requirements** with the specified file counts and naming patterns

5. Your project directory has become cluttered with hundreds of files. Use wildcards to

```
sano@MacBook:~/web_project$ ls
archive  backups  css  desktop  html  js
sano@MacBook:~/web_project$ mv html/*[0-9].* archive/
sano@MacBook:~/web_project$ ls css/*.css | grep -v -E '(mobile|tablet)' | xargs -I {} cp {} desktop/
sano@MacBook:~/web_project$ find . -name "???.*" -type f
sano@MacBook:~/web_project$ find . -name "[bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]*" -type f
./archive/page_002.html
./archive/page_012.html
./archive/page_010.html
./archive/page_007.html
./archive/page_003.html
./archive/page_011.html
./archive/page_005.html
./archive/page_009.html
./archive/page_008.html
./archive/page_001.html
./archive/page_004.html
./archive/page_006.html
./js/config.js
./js/script.js
./js/script_config.js
./js/script_util.js
./desktop/main.css
./desktop/desktop.css
./desktop/print.css
./desktop/theme_light.css
./desktop/reset.css
./desktop/theme_dark.css
./backups/b002.css.backup
./backups/c004.css.backup
```

```
./backups/b004.css.backup
./backups/d004.css.backup
./backups/d003.css.backup
./backups/c001.css.backup
./backups/c005.js.backup
./backups/b001.html.backup
./backups/c004.html.backup
./backups/b003.html.backup
./backups/c001.html.backup
./backups/d003.js.backup
./backups/b002.js.backup
./backups/b005.html.backup
./css/tablet.css
./css/main.css
./css/desktop.css
./css/print.css
./css/theme_light.css
./css/mobile.css
./css/reset.css
./css/theme_dark.css
./html/contact.html
sano@MacBook:~/web_project$ find . -type f -name "*.\?\?" | grep -v '/\.'
sano@MacBook:~/web_project$ cd ..
```

6. A batch processing system needs specific file naming patterns. Use brace expansion to create:

```
sano@MacBook:~/web_project$ find . -type f -name "*.\?\?" | grep -v '/\.'
sano@MacBook:~/web_project$ cd ..
sano@MacBook:~$ mkdir -p batch_processing/{logs,configs,tests} && cd batch_processing
sano@MacBook:~/batch_processing$ touch logs/log_{2024-01-{01..31},2024-02-{01..29},2024-03-{01..31}}.txt && touch config
s/{dev,staging,prod}_{web,api,db}.config && touch tests/{A,B,C}{10,11,12}_{input,output}.test
sano@MacBook:~/batch_processing$ ls log_*.txt | wc -l
ls: cannot access 'log_*.txt': No such file or directory
0
sano@MacBook:~/batch_processing$ ls
configs  logs  tests
sano@MacBook:~/batch_processing$ cd logs && ls
log_2024-01-01.txt  log_2024-01-17.txt  log_2024-02-02.txt  log_2024-02-18.txt  log_2024-03-05.txt  log_2024-03-21.txt
log_2024-01-02.txt  log_2024-01-18.txt  log_2024-02-03.txt  log_2024-02-19.txt  log_2024-03-06.txt  log_2024-03-22.txt
log_2024-01-03.txt  log_2024-01-19.txt  log_2024-02-04.txt  log_2024-02-20.txt  log_2024-03-07.txt  log_2024-03-23.txt
log_2024-01-04.txt  log_2024-01-20.txt  log_2024-02-05.txt  log_2024-02-21.txt  log_2024-03-08.txt  log_2024-03-24.txt
log_2024-01-05.txt  log_2024-01-21.txt  log_2024-02-06.txt  log_2024-02-22.txt  log_2024-03-09.txt  log_2024-03-25.txt
log_2024-01-06.txt  log_2024-01-22.txt  log_2024-02-07.txt  log_2024-02-23.txt  log_2024-03-10.txt  log_2024-03-26.txt
log_2024-01-07.txt  log_2024-01-23.txt  log_2024-02-08.txt  log_2024-02-24.txt  log_2024-03-11.txt  log_2024-03-27.txt
log_2024-01-08.txt  log_2024-01-24.txt  log_2024-02-09.txt  log_2024-02-25.txt  log_2024-03-12.txt  log_2024-03-28.txt
log_2024-01-09.txt  log_2024-01-25.txt  log_2024-02-10.txt  log_2024-02-26.txt  log_2024-03-13.txt  log_2024-03-29.txt
log_2024-01-10.txt  log_2024-01-26.txt  log_2024-02-11.txt  log_2024-02-27.txt  log_2024-03-14.txt  log_2024-03-30.txt
log_2024-01-11.txt  log_2024-01-27.txt  log_2024-02-12.txt  log_2024-02-28.txt  log_2024-03-15.txt  log_2024-03-31.txt
log_2024-01-12.txt  log_2024-01-28.txt  log_2024-02-13.txt  log_2024-02-29.txt  log_2024-03-16.txt
log_2024-01-13.txt  log_2024-01-29.txt  log_2024-02-14.txt  log_2024-03-01.txt  log_2024-03-17.txt
log_2024-01-14.txt  log_2024-01-30.txt  log_2024-02-15.txt  log_2024-03-02.txt  log_2024-03-18.txt
log_2024-01-15.txt  log_2024-01-31.txt  log_2024-02-16.txt  log_2024-03-03.txt  log_2024-03-19.txt
log_2024-01-16.txt  log_2024-02-01.txt  log_2024-02-17.txt  log_2024-03-04.txt  log_2024-03-20.txt
```

```
sano@MacBook:~/batch_processing/logs$ cd ../tests && ls
A10_input.test   A11_output.test  B10_input.test   B11_output.test  C10_input.test   C11_output.test
A10_output.test  A12_input.test   B10_output.test  B12_input.test   C10_output.test  C12_input.test
A11_input.test   A12_output.test  B11_input.test   B12_output.test  C11_input.test   C12_output.test
sano@MacBook:~/batch_processing/tests$ cd ../configs && ls
dev_api.config  dev_web.config   prod_db.config   staging_api.config  staging_web.config
dev_db.config   prod_api.config  prod_web.config  staging_db.config
sano@MacBook:~/batch_processing/configs$
```

Q7. Compare them using different tools (diff, cmp, comm) and explain why each tool gives different results

```
sano@MacBook:~/web_project$ echo -e "server_name=sanokabayizajules.com\nport=8080\ndebug=true\nmax_connections=100" > co
nfig_linux.conf
sano@MacBook:~/web_project$ echo -e "server_name=sanokabayizajules.com\r\nport=8080\r\ndebug=true\r\nmax_connections=100
" > config_windows.conf
sano@MacBook:~/web_project$ hexdump -C config_linux.conf | head -5
00000000  73 65 72 76 65 72 5f 6e  61 6d 65 3d 73 61 6e 6f  |server_name=sano|
00000010  6b 61 62 61 79 69 7a 61  6a 75 6c 65 73 2e 63 6f  |kabayizajules.co|
00000020  6d 0a 70 6f 72 74 3d 38  30 38 30 0a 64 65 62 75  |m.port=8080.debu|
00000030  67 3d 74 72 75 65 0a 6d  61 78 5f 63 6f 6e 6e 65  |g=true.max_conne|
00000040  63 74 69 6f 6e 73 3d 31  30 30 0a                 |ctions=100.|
sano@MacBook:~/web_project$ hexdump -C config_windows.conf | head -5
00000000  73 65 72 76 65 72 5f 6e  61 6d 65 3d 73 61 6e 6f  |server_name=sano|
00000010  6b 61 62 61 79 69 7a 61  6a 75 6c 65 73 2e 63 6f  |kabayizajules.co|
00000020  6d 0d 0a 70 6f 72 74 3d  38 30 38 30 0d 0a 64 65  |m..port=8080..de|
00000030  62 75 67 3d 74 72 75 65  0d 0a 6d 61 78 5f 63 6f  |bug=true..max_co|
00000040  6e 6e 65 63 74 69 6f 6e  73 3d 31 30 30 0a        |nnections=100.|
sano@MacBook:~/web_project$ diff config_linux.conf config_windows.conf
1,3c1,3
< server_name=sanokabayizajules.com
< port=8080
< debug=true
---
> server_name=sanokabayizajules.com
> port=8080
> debug=true
sano@MacBook:~/web_project$ cmp config_linux.conf config_windows.conf
config_linux.conf config_windows.conf differ: byte 34, line 1
sano@MacBook:~/web_project$ comm config_linux.conf config_windows.conf
server_name=sanokabayizajules.com
comm: file 1 is not in sorted order
```

```
sano@MacBook:~/web_project$ comm config_linux.conf config_windows.conf
server_name=sanokabayizajules.com
comm: file 1 is not in sorted order
port=8080
debug=true
max_connections=100
        server_name=sanokabayizajules.com
comm: file 2 is not in sorted order
        port=8080
        debug=true
        max_connections=100
comm: input is not in sorted order
sano@MacBook:~/web_project$
```

Q8. Create a test environment with diverse file types, sizes, and ages

```
sano@MacBook:~/web_project$ ls
archive  backups  config_linux.conf  config_windows.conf  css  desktop  html  js
sano@MacBook:~/web_project$ mkdir -p audit_test/{public,private,logs,temp,backups,configs,.hidden_dir} && cd audit_test
sano@MacBook:~/web_project/audit_test$ dd if=/dev/zero of=large_file1.dat bs=1M count=10 && dd if=/dev/zero of=large_fil
e2.dat bs=1M count=5
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.0209829 s, 500 MB/s
5+0 records in
5+0 records out
5242880 bytes (5.2 MB, 5.0 MiB) copied, 0.0140942 s, 372 MB/s
sano@MacBook:~/web_project/audit_test$ dd if=/dev/zero of=public/medium_file.txt bs=1K count=100 && dd if=/dev/zero of=p
rivate/secret.dat bs=1K count=50
100+0 records in
100+0 records out
102400 bytes (102 kB, 100 KiB) copied, 0.00075595 s, 135 MB/s
50+0 records in
50+0 records out
51200 bytes (51 kB, 50 KiB) copied, 0.000449908 s, 114 MB/s
sano@MacBook:~/web_project/audit_test$ echo "test" > small_file.txt && echo "config" > configs/app.conf && echo "log ent
ry" > logs/app.log
sano@MacBook:~/web_project/audit_test$ touch -d "5 days ago" old_file.txt && touch -d "50 hours ago" recent_file.txt &&
touch -d "36 hours ago" temp/.temp_file.swp && touch -d "12 hours ago" new_file.txt
sano@MacBook:~/web_project/audit_test$ touch backup_2024.bak backups/database.sql.bak temp/tmp123.tmp .swp file.temp &&
touch public/script.sh~ private/.tmp_config configs/.old_config
sano@MacBook:~/web_project/audit_test$ chmod 777 public/medium_file.txt && chmod 666 temp/.temp_file.swp && chmod 644 lo
gs/app.log && chmod 600 private/secret.dat && chmod 000 configs/.old_config
sano@MacBook:~/web_project/audit_test$ mkdir empty_dir && touch .hidden_dir/.hidden_file1 .hidden_dir/.hidden_file2
sano@MacBook:~/web_project/audit_test$
```

```
sano@MacBook:~/web_project/audit_test$ find . -type f -size +$(find . -type f -printf "%s\n" | awk '{sum+=$1; count++} E
ND{if(count>0) print sum/count}')c
./large_file2.dat
./large_file1.dat
sano@MacBook:~/web_project/audit_test$ find . -type f -mtime -3 -mtime +0.5
./temp/.temp_file.swp
./recent_file.txt
sano@MacBook:~/web_project/audit_test$ find . -type d \( -empty -o -exec sh -c 'ls -A "{}" | grep -q "^[.]" && [ "$(ls -
A "{}" | grep -v "^\." | wc -l)" -eq 0 ]' \; \)
sano@MacBook:~/web_project/audit_test$ find . -type f -perm /o=w ! -path "*/.*"
./public/medium_file.txt
sano@MacBook:~/web_project/audit_test$ find . -type f ! -user $USER ! -user root
sano@MacBook:~/web_project/audit_test$ find . -type f \( -name "*~" -o -name "*.bak" -o -name "*.tmp" -o -name "*.swp" -
o -name "*.temp" -o -name "*.old" \)
./public/script.sh~
./.swp
./file.temp
./temp/.temp_file.swp
./temp/tmp123.tmp
./backups/database.sql.bak
./backup_2024.bak
sano@MacBook:~/web_project/audit_test$ find . -type f -perm /o=w \( -name "*~" -o -name "*.tmp" -o -name "*.swp" \) -mti
me -7
./temp/.temp_file.swp
sano@MacBook:~/web_project/audit_test$
```

```
sano@MacBook:~/web_project/audit_test$ find . -type f -perm /o=w ! -user $USER \( -name "*~" -o -name "*.tmp" -o -name "
*.bak" \) -ls
sano@MacBook:~/web_project/audit_test$ find . -type f -mtime -3 \( -name "*.bak" -o -name "*~" \) -perm /o=w -ls
sano@MacBook:~/web_project/audit_test$ find . -name ".*" \( -type f -perm /o=w -o -type d -perm /o=w \) ! -path "." -ls
    53813      0 -rw-rw-rw-   1 sano     sano            0 Sep 26 09:44 ./temp/.temp_file.swp
sano@MacBook:~/web_project/audit_test$
```

Key Security Findings This Audit Reveals

1. World-writable files - Any user can modify these
2. Stale temporary files - May contain sensitive data
3. Hidden directories with content - Could conceal malicious files
4. Files with unusual ownership - Potential privilege escalation vectors
5. Large recently-modified files - Could indicate data exfiltration

This audit approach helps identify common security misconfigurations and suspicious file patterns that warrant investigation in a real security review.

Q9. Demonstrate why less is superior to cat for large files during an SSH session with limited bandwidth.

1. Create a Large Log File (200+ lines)

```
sano@MacBook:~/web_project$ ls
archive  audit_test  backups  config_linux.conf  config_windows.conf  css  desktop  html  js
sano@MacBook:~/web_project$ cat > large_log.txt << 'EOF'
2024-01-15 08:00:01 INFO: Server started successfully
2024-01-15 08:00:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-15 08:00:03 INFO: Database connection established
2024-01-15 08:00:04 WARNING: Cache size set to default
2024-01-15 08:00:05 ERROR: Failed to connect to external API - timeout
2024-01-15 08:00:06 INFO: User admin logged in from 192.168.1.100
2024-01-15 08:00:07 DEBUG: Processing request ID: 12345
2024-01-15 08:00:08 ERROR: Database query failed - table not found
2024-01-15 08:00:09 INFO: Backup scheduled for 02:00
2024-01-15 08:00:10 CRITICAL: Disk space below 5% threshold
EOF
sano@MacBook:~/web_project$ for i in {1..20}; do
    sed "s/2024-01-15/2024-01-$((15+RANDOM%5))/; s/08:00:/$((8+RANDOM%3)):$((RANDOM%60)):/" large_log.txt >> large_log_
full.txt
done
sano@MacBook:~/web_project$ wc -l large_log_full.txt
200 large_log_full.txt
sano@MacBook:~/web_project$
```

2. Display Middle 50 Lines

```
sano@MacBook:~/web_project$ sed -n '76,125p' large_log_full.txt
2024-01-15) 8:0:06 INFO: User admin logged in from 192.168.1.100
2024-01-15) 8:0:07 DEBUG: Processing request ID: 12345
2024-01-15) 8:0:08 ERROR: Database query failed - table not found
2024-01-15) 8:0:09 INFO: Backup scheduled for 02:00
2024-01-15) 8:0:10 CRITICAL: Disk space below 5% threshold
2024-01-19) 9:3:01 INFO: Server started successfully
2024-01-19) 9:3:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-19) 9:3:03 INFO: Database connection established
2024-01-19) 9:3:04 WARNING: Cache size set to default
2024-01-19) 9:3:05 ERROR: Failed to connect to external API - timeout
2024-01-19) 9:3:06 INFO: User admin logged in from 192.168.1.100
2024-01-19) 9:3:07 DEBUG: Processing request ID: 12345
2024-01-19) 9:3:08 ERROR: Database query failed - table not found
2024-01-19) 9:3:09 INFO: Backup scheduled for 02:00
2024-01-19) 9:3:10 CRITICAL: Disk space below 5% threshold
2024-01-17) 8:16:01 INFO: Server started successfully
2024-01-17) 8:16:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-17) 8:16:03 INFO: Database connection established
2024-01-17) 8:16:04 WARNING: Cache size set to default
2024-01-17) 8:16:05 ERROR: Failed to connect to external API - timeout
2024-01-17) 8:16:06 INFO: User admin logged in from 192.168.1.100
2024-01-17) 8:16:07 DEBUG: Processing request ID: 12345
2024-01-17) 8:16:08 ERROR: Database query failed - table not found
```

```
sano@MacBook:~/web_project$ total_lines=$(wc -l < large_log_full.txt)
start_line=$(( (total_lines - 50) / 2 + 1 ))
tail -n +$start_line large_log_full.txt | head -n 50
2024-01-15) 8:0:06 INFO: User admin logged in from 192.168.1.100
2024-01-15) 8:0:07 DEBUG: Processing request ID: 12345
2024-01-15) 8:0:08 ERROR: Database query failed - table not found
2024-01-15) 8:0:09 INFO: Backup scheduled for 02:00
2024-01-15) 8:0:10 CRITICAL: Disk space below 5% threshold
2024-01-19) 9:3:01 INFO: Server started successfully
2024-01-19) 9:3:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-19) 9:3:03 INFO: Database connection established
2024-01-19) 9:3:04 WARNING: Cache size set to default
2024-01-19) 9:3:05 ERROR: Failed to connect to external API - timeout
2024-01-19) 9:3:06 INFO: User admin logged in from 192.168.1.100
2024-01-19) 9:3:07 DEBUG: Processing request ID: 12345
2024-01-19) 9:3:08 ERROR: Database query failed - table not found
2024-01-19) 9:3:09 INFO: Backup scheduled for 02:00
2024-01-19) 9:3:10 CRITICAL: Disk space below 5% threshold
2024-01-17) 8:16:01 INFO: Server started successfully
2024-01-17) 8:16:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-17) 8:16:03 INFO: Database connection established
2024-01-17) 8:16:04 WARNING: Cache size set to default
2024-01-17) 8:16:05 ERROR: Failed to connect to external API - timeout
2024-01-17) 8:16:06 INFO: User admin logged in from 192.168.1.100
2024-01-17) 8:16:07 DEBUG: Processing request ID: 12345
2024-01-17) 8:16:08 ERROR: Database query failed - table not found
2024-01-17) 8:16:09 INFO: Backup scheduled for 02:00
```

```
sano@MacBook:~/web_project$ awk 'NR>=76 && NR<=125' large_log_full.txt
2024-01-15) 8:0:06 INFO: User admin logged in from 192.168.1.100
2024-01-15) 8:0:07 DEBUG: Processing request ID: 12345
2024-01-15) 8:0:08 ERROR: Database query failed - table not found
2024-01-15) 8:0:09 INFO: Backup scheduled for 02:00
2024-01-15) 8:0:10 CRITICAL: Disk space below 5% threshold
2024-01-19) 9:3:01 INFO: Server started successfully
2024-01-19) 9:3:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-19) 9:3:03 INFO: Database connection established
2024-01-19) 9:3:04 WARNING: Cache size set to default
2024-01-19) 9:3:05 ERROR: Failed to connect to external API - timeout
2024-01-19) 9:3:06 INFO: User admin logged in from 192.168.1.100
2024-01-19) 9:3:07 DEBUG: Processing request ID: 12345
2024-01-19) 9:3:08 ERROR: Database query failed - table not found
2024-01-19) 9:3:09 INFO: Backup scheduled for 02:00
2024-01-19) 9:3:10 CRITICAL: Disk space below 5% threshold
2024-01-17) 8:16:01 INFO: Server started successfully
2024-01-17) 8:16:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-17) 8:16:03 INFO: Database connection established
2024-01-17) 8:16:04 WARNING: Cache size set to default
2024-01-17) 8:16:05 ERROR: Failed to connect to external API - timeout
2024-01-17) 8:16:06 INFO: User admin logged in from 192.168.1.100
2024-01-17) 8:16:07 DEBUG: Processing request ID: 12345
2024-01-17) 8:16:08 ERROR: Database query failed - table not found
2024-01-17) 8:16:09 INFO: Backup scheduled for 02:00
2024-01-17) 8:16:10 CRITICAL: Disk space below 5% threshold
2024-01-19) 9:23:01 INFO: Server started successfully
2024-01-19) 9:23:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-19) 9:23:03 INFO: Database connection established
```

3. Find Last Occurrence of "ERROR" with 5 Lines Context

```
sano@MacBook:~/web_project$ tac large_log_full.txt | grep -n -B5 -A5 "ERROR" | head -11 | tac
11-2024-01-16) 10:56:10 CRITICAL: Disk space below 5% threshold
10-2024-01-18) 8:19:01 INFO: Server started successfully
9-2024-01-18) 8:19:02 DEBUG: Loading configuration from /etc/app/config.conf
8-2024-01-18) 8:19:03 INFO: Database connection established
7-2024-01-18) 8:19:04 WARNING: Cache size set to default
6:2024-01-18) 8:19:05 ERROR: Failed to connect to external API - timeout
5-2024-01-18) 8:19:06 INFO: User admin logged in from 192.168.1.100
4-2024-01-18) 8:19:07 DEBUG: Processing request ID: 12345
3:2024-01-18) 8:19:08 ERROR: Database query failed - table not found
2-2024-01-18) 8:19:09 INFO: Backup scheduled for 02:00
1-2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
sano@MacBook:~/web_project$ awk '/ERROR/ {last_match=NR; context=$0} END {if (last_match) {for(i=last_match-5;i<=last_ma
tch+5;i++) if(i>0) print i ":" (i==last_match?"*":" ") " " $0}}' large_log_full.txt
193:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
194:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
195:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
196:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
197:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
198:* 2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
199:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
200:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
201:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
202:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
203:  2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
sano@MacBook:~/web_project$ last_error_line=$(grep -n "ERROR" large_log_full.txt | tail -1 | cut -d: -f1)
sed -n "$((last_error_line-5)),$((last_error_line+5))p" large_log_full.txt
2024-01-18) 8:19:03 INFO: Database connection established
2024-01-18) 8:19:04 WARNING: Cache size set to default
```

```
sano@MacBook:~/web_project$ last_error_line=$(grep -n "ERROR" large_log_full.txt | tail -1 | cut -d: -f1)
sed -n "$((last_error_line-5)),$((last_error_line+5))p" large_log_full.txt
2024-01-18) 8:19:03 INFO: Database connection established
2024-01-18) 8:19:04 WARNING: Cache size set to default
2024-01-18) 8:19:05 ERROR: Failed to connect to external API - timeout
2024-01-18) 8:19:06 INFO: User admin logged in from 192.168.1.100
2024-01-18) 8:19:07 DEBUG: Processing request ID: 12345
2024-01-18) 8:19:08 ERROR: Database query failed - table not found
2024-01-18) 8:19:09 INFO: Backup scheduled for 02:00
2024-01-18) 8:19:10 CRITICAL: Disk space below 5% threshold
sano@MacBook:~/web_project$
```

4. Compare Efficiency of Different Tools

```
sano@MacBook:~/web_project$ echo -n "cat: "
time cat large_log_full.txt > /dev/null
cat:
real    0m0.004s
user    0m0.000s
sys     0m0.003s
sano@MacBook:~/web_project$ echo -n "less: "
time less large_log_full.txt > /dev/null
less:
real    0m0.029s
user    0m0.006s
sys     0m0.012s
sano@MacBook:~/web_project$ echo -n "head: "
time head -n 250 large_log_full.txt > /dev/null
head:
real    0m0.005s
user    0m0.001s
sys     0m0.005s
sano@MacBook:~/web_project$ echo -n "sed: "
time sed -n '1,250p' large_log_full.txt > /dev/null
sed:
real    0m0.005s
user    0m0.004s
sys     0m0.001s
sano@MacBook:~/web_project$ echo -n "awk: "
time awk 'NR<=250' large_log_full.txt > /dev/null
awk:
real    0m0.005s
user    0m0.002s
sys     0m0.003s
```

5. Extract Lines with Error Patterns Preserving Line Numbers

```
sano@MacBook:~/web_project$ grep -n -E "(ERROR|CRITICAL|FAILED)" large_log_full.txt
5:2024-01-18) 8:31:05 ERROR: Failed to connect to external API - timeout
8:2024-01-18) 8:31:08 ERROR: Database query failed - table not found
10:2024-01-18) 8:31:10 CRITICAL: Disk space below 5% threshold
15:2024-01-18) 9:35:05 ERROR: Failed to connect to external API - timeout
18:2024-01-18) 9:35:08 ERROR: Database query failed - table not found
20:2024-01-18) 9:35:10 CRITICAL: Disk space below 5% threshold
25:2024-01-17) 10:8:05 ERROR: Failed to connect to external API - timeout
28:2024-01-17) 10:8:08 ERROR: Database query failed - table not found
30:2024-01-17) 10:8:10 CRITICAL: Disk space below 5% threshold
35:2024-01-17) 10:16:05 ERROR: Failed to connect to external API - timeout
38:2024-01-17) 10:16:08 ERROR: Database query failed - table not found
40:2024-01-17) 10:16:10 CRITICAL: Disk space below 5% threshold
45:2024-01-15) 9:59:05 ERROR: Failed to connect to external API - timeout
48:2024-01-15) 9:59:08 ERROR: Database query failed - table not found
50:2024-01-15) 9:59:10 CRITICAL: Disk space below 5% threshold
55:2024-01-19) 9:23:05 ERROR: Failed to connect to external API - timeout
58:2024-01-19) 9:23:08 ERROR: Database query failed - table not found
60:2024-01-19) 9:23:10 CRITICAL: Disk space below 5% threshold
65:2024-01-18) 10:1:05 ERROR: Failed to connect to external API - timeout
68:2024-01-18) 10:1:08 ERROR: Database query failed - table not found
70:2024-01-18) 10:1:10 CRITICAL: Disk space below 5% threshold
75:2024-01-15) 8:0:05 ERROR: Failed to connect to external API - timeout
78:2024-01-15) 8:0:08 ERROR: Database query failed - table not found
80:2024-01-15) 8:0:10 CRITICAL: Disk space below 5% threshold
85:2024-01-19) 9:3:05 ERROR: Failed to connect to external API - timeout
88:2024-01-19) 9:3:08 ERROR: Database query failed - table not found
90:2024-01-19) 9:3:10 CRITICAL: Disk space below 5% threshold
95:2024-01-17) 8:16:05 ERROR: Failed to connect to external API - timeout
```

```
sano@MacBook:~/web_project$ awk '/ERROR|CRITICAL|FAILED/ {print NR ": " $0}' large_log_full.txt
5: 2024-01-18) 8:31:05 ERROR: Failed to connect to external API - timeout
8: 2024-01-18) 8:31:08 ERROR: Database query failed - table not found
10: 2024-01-18) 8:31:10 CRITICAL: Disk space below 5% threshold
15: 2024-01-18) 9:35:05 ERROR: Failed to connect to external API - timeout
18: 2024-01-18) 9:35:08 ERROR: Database query failed - table not found
20: 2024-01-18) 9:35:10 CRITICAL: Disk space below 5% threshold
25: 2024-01-17) 10:8:05 ERROR: Failed to connect to external API - timeout
28: 2024-01-17) 10:8:08 ERROR: Database query failed - table not found
30: 2024-01-17) 10:8:10 CRITICAL: Disk space below 5% threshold
35: 2024-01-17) 10:16:05 ERROR: Failed to connect to external API - timeout
38: 2024-01-17) 10:16:08 ERROR: Database query failed - table not found
40: 2024-01-17) 10:16:10 CRITICAL: Disk space below 5% threshold
45: 2024-01-15) 9:59:05 ERROR: Failed to connect to external API - timeout
48: 2024-01-15) 9:59:08 ERROR: Database query failed - table not found
50: 2024-01-15) 9:59:10 CRITICAL: Disk space below 5% threshold
55: 2024-01-19) 9:23:05 ERROR: Failed to connect to external API - timeout
58: 2024-01-19) 9:23:08 ERROR: Database query failed - table not found
60: 2024-01-19) 9:23:10 CRITICAL: Disk space below 5% threshold
65: 2024-01-18) 10:1:05 ERROR: Failed to connect to external API - timeout
68: 2024-01-18) 10:1:08 ERROR: Database query failed - table not found
70: 2024-01-18) 10:1:10 CRITICAL: Disk space below 5% threshold
75: 2024-01-15) 8:0:05 ERROR: Failed to connect to external API - timeout
78: 2024-01-15) 8:0:08 ERROR: Database query failed - table not found
80: 2024-01-15) 8:0:10 CRITICAL: Disk space below 5% threshold
85: 2024-01-19) 9:3:05 ERROR: Failed to connect to external API - timeout
88: 2024-01-19) 9:3:08 ERROR: Database query failed - table not found
90: 2024-01-19) 9:3:10 CRITICAL: Disk space below 5% threshold
95: 2024-01-17) 8:16:05 ERROR: Failed to connect to external API - timeout
98: 2024-01-17) 8:16:08 ERROR: Database query failed - table not found
```

```
sano@MacBook:~/web_project$ nl -ba large_log_full.txt | grep -E "(ERROR|CRITICAL|FAILED)"
     5  2024-01-18) 8:31:05 ERROR: Failed to connect to external API - timeout
     8  2024-01-18) 8:31:08 ERROR: Database query failed - table not found
    10  2024-01-18) 8:31:10 CRITICAL: Disk space below 5% threshold
    15  2024-01-18) 9:35:05 ERROR: Failed to connect to external API - timeout
    18  2024-01-18) 9:35:08 ERROR: Database query failed - table not found
    20  2024-01-18) 9:35:10 CRITICAL: Disk space below 5% threshold
    25  2024-01-17) 10:8:05 ERROR: Failed to connect to external API - timeout
    28  2024-01-17) 10:8:08 ERROR: Database query failed - table not found
    30  2024-01-17) 10:8:10 CRITICAL: Disk space below 5% threshold
    35  2024-01-17) 10:16:05 ERROR: Failed to connect to external API - timeout
    38  2024-01-17) 10:16:08 ERROR: Database query failed - table not found
    40  2024-01-17) 10:16:10 CRITICAL: Disk space below 5% threshold
    45  2024-01-15) 9:59:05 ERROR: Failed to connect to external API - timeout
    48  2024-01-15) 9:59:08 ERROR: Database query failed - table not found
    50  2024-01-15) 9:59:10 CRITICAL: Disk space below 5% threshold
    55  2024-01-19) 9:23:05 ERROR: Failed to connect to external API - timeout
    58  2024-01-19) 9:23:08 ERROR: Database query failed - table not found
    60  2024-01-19) 9:23:10 CRITICAL: Disk space below 5% threshold
    65  2024-01-18) 10:1:05 ERROR: Failed to connect to external API - timeout
    68  2024-01-18) 10:1:08 ERROR: Database query failed - table not found
    70  2024-01-18) 10:1:10 CRITICAL: Disk space below 5% threshold
    75  2024-01-15) 8:0:05 ERROR: Failed to connect to external API - timeout
    78  2024-01-15) 8:0:08 ERROR: Database query failed - table not found
    80  2024-01-15) 8:0:10 CRITICAL: Disk space below 5% threshold
    85  2024-01-19) 9:3:05 ERROR: Failed to connect to external API - timeout
    88  2024-01-19) 9:3:08 ERROR: Database query failed - table not found
    90  2024-01-19) 9:3:10 CRITICAL: Disk space below 5% threshold
    95  2024-01-17) 8:16:05 ERROR: Failed to connect to external API - timeout
```

```
sano@MacBook:~/web_project$ grep -n -B2 -A2 -E "(ERROR|CRITICAL)" large_log_full.txt
3-2024-01-18) 8:31:03 INFO: Database connection established
4-2024-01-18) 8:31:04 WARNING: Cache size set to default
5:2024-01-18) 8:31:05 ERROR: Failed to connect to external API - timeout
6-2024-01-18) 8:31:06 INFO: User admin logged in from 192.168.1.100
7-2024-01-18) 8:31:07 DEBUG: Processing request ID: 12345
8:2024-01-18) 8:31:08 ERROR: Database query failed - table not found
9-2024-01-18) 8:31:09 INFO: Backup scheduled for 02:00
10:2024-01-18) 8:31:10 CRITICAL: Disk space below 5% threshold
11-2024-01-18) 9:35:01 INFO: Server started successfully
12-2024-01-18) 9:35:02 DEBUG: Loading configuration from /etc/app/config.conf
13-2024-01-18) 9:35:03 INFO: Database connection established
14-2024-01-18) 9:35:04 WARNING: Cache size set to default
15:2024-01-18) 9:35:05 ERROR: Failed to connect to external API - timeout
16-2024-01-18) 9:35:06 INFO: User admin logged in from 192.168.1.100
17-2024-01-18) 9:35:07 DEBUG: Processing request ID: 12345
18:2024-01-18) 9:35:08 ERROR: Database query failed - table not found
19-2024-01-18) 9:35:09 INFO: Backup scheduled for 02:00
20:2024-01-18) 9:35:10 CRITICAL: Disk space below 5% threshold
21-2024-01-17) 10:8:01 INFO: Server started successfully
22-2024-01-17) 10:8:02 DEBUG: Loading configuration from /etc/app/config.conf
23-2024-01-17) 10:8:03 INFO: Database connection established
24-2024-01-17) 10:8:04 WARNING: Cache size set to default
25:2024-01-17) 10:8:05 ERROR: Failed to connect to external API - timeout
26-2024-01-17) 10:8:06 INFO: User admin logged in from 192.168.1.100
27-2024-01-17) 10:8:07 DEBUG: Processing request ID: 12345
28:2024-01-17) 10:8:08 ERROR: Database query failed - table not found
29-2024-01-17) 10:8:09 INFO: Backup scheduled for 02:00
30:2024-01-17) 10:8:10 CRITICAL: Disk space below 5% threshold
```

6. Demonstration: less vs cat for SSH with Limited Bandwidth

```
sano@MacBook:~/web_project$ echo "=== Cat (loads entire file at once) ==="
time cat large_log_full.txt | head -n 10
=== Cat (loads entire file at once) ===
2024-01-18) 8:31:01 INFO: Server started successfully
2024-01-18) 8:31:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-18) 8:31:03 INFO: Database connection established
2024-01-18) 8:31:04 WARNING: Cache size set to default
2024-01-18) 8:31:05 ERROR: Failed to connect to external API - timeout
2024-01-18) 8:31:06 INFO: User admin logged in from 192.168.1.100
2024-01-18) 8:31:07 DEBUG: Processing request ID: 12345
2024-01-18) 8:31:08 ERROR: Database query failed - table not found
2024-01-18) 8:31:09 INFO: Backup scheduled for 02:00
2024-01-18) 8:31:10 CRITICAL: Disk space below 5% threshold

real    0m0.005s
user    0m0.000s
sys     0m0.006s
sano@MacBook:~/web_project$ echo -e "\n=== Less (loads incrementally) ==="
time less large_log_full.txt | head -n 10

=== Less (loads incrementally) ===
2024-01-18) 8:31:01 INFO: Server started successfully
2024-01-18) 8:31:02 DEBUG: Loading configuration from /etc/app/config.conf
2024-01-18) 8:31:03 INFO: Database connection established
2024-01-18) 8:31:04 WARNING: Cache size set to default
2024-01-18) 8:31:05 ERROR: Failed to connect to external API - timeout
2024-01-18) 8:31:06 INFO: User admin logged in from 192.168.1.100
2024-01-18) 8:31:07 DEBUG: Processing request ID: 12345
2024-01-18) 8:31:08 ERROR: Database query failed - table not found
2024-01-18) 8:31:09 INFO: Backup scheduled for 02:00
```

```
sano@MacBook:~/web_project$ cat large_log_full.txt | pv -q -L 10k | wc -l
200
sano@MacBook:~/web_project$ echo -e "\nWith LESS:"
echo "----------"
less large_log_full.txt | head -n 50 | pv -q -L 10k | wc -l

With LESS:
----------
50
sano@MacBook:~/web_project$
```

7. Advanced Log Analysis Examples

```
sano@MacBook:~/web_project$ grep -c "ERROR" large_log_full.txt
40
sano@MacBook:~/web_project$ grep -c "WARNING" large_log_full.txt
20
sano@MacBook:~/web_project$ grep -c "CRITICAL" large_log_full.txt
20
sano@MacBook:~/web_project$ awk '/ERROR/ {print $2}' large_log_full.txt | cut -d: -f1 | sort | uniq -c
     14 10
     10 8
     16 9
sano@MacBook:~/web_project$ grep "ERROR" large_log_full.txt | cut -d: -f3- | sort | uniq -c | sort -nr
     20 08 ERROR: Database query failed - table not found
     20 05 ERROR: Failed to connect to external API - timeout
sano@MacBook:~/web_project$
```

8. Why less is Superior for SSH/Large Files:

Key Advantages:

1. Lazy loading - Only reads what's displayed, not the entire file
2. Search functionality - /pattern to search, n for next match
3. Navigation - Scroll with arrows, page up/down, jump to top/bottom
4. Bandwidth efficient - Crucial for slow SSH connections
5. Interactive - Can pause, scroll back, without reloading
6. Memory efficient - Doesn't load entire file into memory

Q10. Automate file maintenance tasks using find with -exec

1. **Preview Before Executing (SAFETY FIRST)**
2. **Change Permissions: 644 for files, 755 for executables**

```
sano@MacBook:~/web_project$ find . -name "*.tmp" -mtime +30 -print
sano@MacBook:~/web_project$ find . -type f ! -executable -exec echo "chmod 644 {}" \;
chmod 644 ./large_log.txt
chmod 644 ./archive/page_002.html
chmod 644 ./archive/page_012.html
chmod 644 ./archive/page_010.html
chmod 644 ./archive/page_007.html
chmod 644 ./archive/page_003.html
chmod 644 ./archive/page_011.html
chmod 644 ./archive/page_005.html
chmod 644 ./archive/page_009.html
chmod 644 ./archive/page_008.html
chmod 644 ./archive/page_001.html
chmod 644 ./archive/page_004.html
chmod 644 ./archive/page_006.html
chmod 644 ./audit_test/private/.tmp_config
chmod 644 ./audit_test/private/secret.dat
chmod 644 ./audit_test/public/script.sh~
chmod 644 ./audit_test/configs/.old_config
chmod 644 ./audit_test/configs/app.conf
chmod 644 ./audit_test/new_file.txt
chmod 644 ./audit_test/small_file.txt
chmod 644 ./audit_test/large_file2.dat
chmod 644 ./audit_test/.swp
chmod 644 ./audit_test/file.temp
chmod 644 ./audit_test/large_file1.dat
chmod 644 ./audit_test/temp/.temp_file.swp
chmod 644 ./audit_test/temp/tmp123.tmp
chmod 644 ./audit_test/backups/database.sql.bak
chmod 644 ./audit_test/recent_file.txt
```

```
sano@MacBook:~/web_project$ find . -type f -executable -exec echo "chmod 755 {}" \;
chmod 755 ./audit_test/public/medium_file.txt
sano@MacBook:~/web_project$ find . -type f ! -executable -exec chmod 644 {} \;
sano@MacBook:~/web_project$ find . -type f -executable -exec chmod 755 {} \;
sano@MacBook:~/web_project$ find . -type f -ls | head -10
    53826      4 -rw-r--r--   1 sano     sano          615 Sep 27 22:05 ./large_log.txt
    53579      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_002.html
    53589      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_012.html
    53587      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_010.html
    53584      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_007.html
    53580      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_003.html
    53588      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_011.html
    53582      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_005.html
    53586      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_009.html
    53585      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./archive/page_008.html
sano@MacBook:~/web_project$
```

3. Calculate Disk Space Used by Files Older Than 30 Days

```
sano@MacBook:~/web_project$ find . -type f -mtime +30 -ls | head -10
sano@MacBook:~/web_project$ find . -type f -mtime +30 -exec du -ch {} + | grep total$
sano@MacBook:~/web_project$ find . -type f -mtime +30 -printf "%s %p\n" | awk '{sum+=$1} END {print "Total:", sum/1024/1
024, "MB"}'
Total: 0 MB
sano@MacBook:~/web_project$
```

4. Backup Configuration Files (*.conf) with .backup Extension

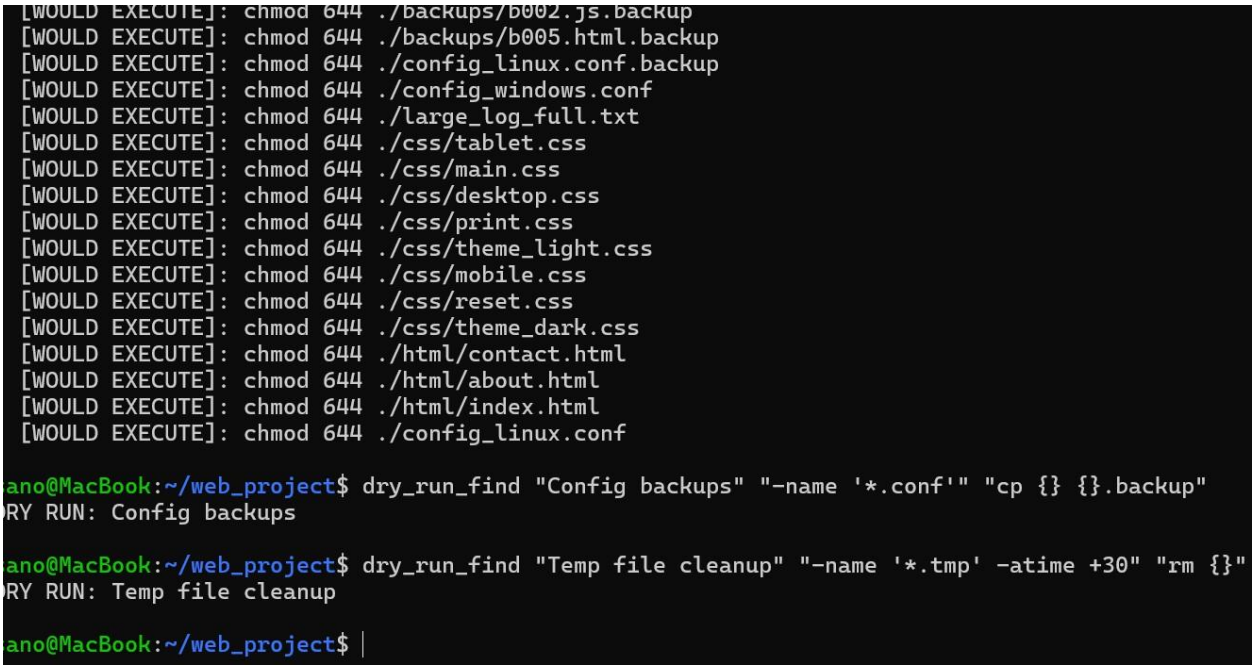
```
sano@MacBook:~/web_project$ find . -name "*.conf" -exec echo "cp {} {}.backup" \;
cp ./audit_test/configs/app.conf ./audit_test/configs/app.conf.backup
cp ./config_windows.conf ./config_windows.conf.backup
cp ./config_linux.conf ./config_linux.conf.backup
sano@MacBook:~/web_project$ find . -name "*.conf" -exec cp {} {}.backup \;
sano@MacBook:~/web_project$ find . -name "*.backup" -ls
    53841      4 -rw-r--r--   1 sano     sano           78 Sep 27 22:46 ./config_windows.conf.backup
    53840      4 -rw-r--r--   1 sano     sano            7 Sep 27 22:46 ./audit_test/configs/app.conf.backup
    53630      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/b002.css.backup
    53605      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/a002.html.backup
    53646      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/a003.js.backup
    53604      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/a001.html.backup
    53637      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/c004.css.backup
    53621      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/d003.html.backup
    53607      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/a004.html.backup
    53657      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/c004.js.backup
    53625      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/a002.css.backup
    53628      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/a005.css.backup
    53660      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/d002.js.backup
    53635      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/c002.css.backup
    53623      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/d005.html.backup
    53662      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/d004.js.backup
    53654      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/c001.js.backup
    53619      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/d001.html.backup
    53656      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/c003.js.backup
    53627      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/a004.css.backup
    53629      0 -rw-r--r--   1 sano     sano            0 Sep 27 09:42 ./backups/b001.css.backup
```

5. Safely Remove Temporary Files (Old & Unaccessed)

```
sano@MacBook:~/web_project$ find . \( -name "*.tmp" -o -name "*.temp" -o -name "*~" -o -name "*.swp" \) -atime +30 -prin
t
sano@MacBook:~/web_project$ find . \( -name "*.tmp" -o -name "*.temp" -o -name "*~" -o -name "*.swp" \) -atime +30 -exec
 rm -i {} \;
sano@MacBook:~/web_project$ find . \( -name "*.tmp" -o -name "*.temp" -o -name "*~" -o -name "*.swp" \) -atime +30 -exec
 rm {} \;
sano@MacBook:~/web_project$
```

6. Advanced Safe Operations with Dry-Run Mode

```
sano@MacBook:~/web_project$ dry_run_find() {
    echo "DRY RUN: $1"
    find . $2 -exec echo "  [WOULD EXECUTE]: $3" \;
    echo
}
sano@MacBook:~/web_project$ dry_run_find "Permission changes" "-type f ! -executable" "chmod 644 {}"
DRY RUN: Permission changes
  [WOULD EXECUTE]: chmod 644 ./large_log.txt
  [WOULD EXECUTE]: chmod 644 ./config_windows.conf.backup
  [WOULD EXECUTE]: chmod 644 ./archive/page_002.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_012.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_010.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_007.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_003.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_011.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_005.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_009.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_008.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_001.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_004.html
  [WOULD EXECUTE]: chmod 644 ./archive/page_006.html
  [WOULD EXECUTE]: chmod 644 ./audit_test/private/.tmp_config
  [WOULD EXECUTE]: chmod 644 ./audit_test/private/secret.dat
  [WOULD EXECUTE]: chmod 644 ./audit_test/public/script.sh~
  [WOULD EXECUTE]: chmod 644 ./audit_test/configs/.old_config
  [WOULD EXECUTE]: chmod 644 ./audit_test/configs/app.conf.backup
  [WOULD EXECUTE]: chmod 644 ./audit_test/configs/app.conf
  [WOULD EXECUTE]: chmod 644 ./audit_test/new_file.txt
```



Key Safety Practices Demonstrated:

1. Always preview with -print or -exec echo before executing
2. Use -ok instead of -exec for interactive confirmation
3. Test on small subsets before running on entire filesystem
4. Backup before deletion operations
5. Use -ls to see what will be affected
6. Implement dry-run mode in scripts

This approach ensures you never accidentally delete or modify important files during maintenance operations.

Q11. Analyze which compression method works best for each content type and explain why

1. Create Test Directory Structure with Different File Types

```
sano@MacBook:~/web_project$ mkdir -p compression_test && cd compression_test
sano@MacBook:~/web_project/compression_test$ mkdir already_compressed
sano@MacBook:~/web_project/compression_test$ ls
already_compressed
sano@MacBook:~/web_project/compression_test$ cp /path/to/some/images/*.jpg already_compressed/ 2>/dev/null || touch alre
ady_compressed/{image1.jpg,image2.jpg,photo1.jpg,photo2.jpg} && cp /path/to/some/videos/*.mp4 already_compressed/ 2>/dev
/null || dd if=/dev/zero of=already_compressed/video1.mp4 bs=1M count=5 && cp /path/to/some/zips/*.zip already_compresse
d/ 2>/dev/null || zip already_compressed/dummy.zip /etc/hosts
5+0 records in
5+0 records out
5242880 bytes (5.2 MB, 5.0 MiB) copied, 0.00640305 s, 819 MB/s
  adding: etc/hosts (deflated 38%)
sano@MacBook:~/web_project/compression_test$ mkdir text_files
sano@MacBook:~/web_project/compression_test$ for i in {1..5}; do
    head -c 2M /dev/urandom | base64 > text_files/large_text_$i.txt
done
sano@MacBook:~/web_project/compression_test$ ls
already_compressed  text_files
sano@MacBook:~/web_project/compression_test$ for i in {1..10}; do
    yes "ERROR: Database connection failed at $(date)" | head -1000 > text_files/log_$i.log
done
sano@MacBook:~/web_project/compression_test$ mkdir mixed_files
sano@MacBook:~/web_project/compression_test$ cp already_compressed/*.jpg mixed_files/ && cp text_files/*.log mixed_files
```

```
sano@MacBook:~/web_project/compression_test$ mkdir mixed_files
sano@MacBook:~/web_project/compression_test$ cp already_compressed/*.jpg mixed_files/ && cp text_files/*.log mixed_files
/
sano@MacBook:~/web_project/compression_test$ dd if=/dev/zero of=mixed_files/binary.data bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.0191629 s, 547 MB/s
sano@MacBook:~/web_project/compression_test$ du -sh *
5.1M    already_compressed
11M     mixed_files
15M     text_files
sano@MacBook:~/web_project/compression_test$
```

2. Compression Commands with Timing

Commands used: gzip, bzip2, xz, zip

```
sano@MacBook:~/web_project/compression_test$ time tar cf - already_compressed | gzip -9 > already_compressed_gzip.tar.gz

real    0m0.049s
user    0m0.029s
sys     0m0.027s
```

```
sano@MacBook:~/web_project/compression_test$ time tar cf - already_compressed | bzip2 -9 > already_compressed_bzip2.tar.
bz2

real    0m0.076s
user    0m0.058s
sys     0m0.030s
sano@MacBook:~/web_project/compression_test$ time tar cf - already_compressed | xz -9 > already_compressed_xz.tar.xz

real    0m0.187s
user    0m0.122s
sys     0m0.080s
sano@MacBook:~/web_project/compression_test$ time zip -9 -r already_compressed_zip.zip already_compressed
  adding: already_compressed/ (stored 0%)
  adding: already_compressed/photo2.jpg (stored 0%)
  adding: already_compressed/video1.mp4 (deflated 100%)
  adding: already_compressed/image1.jpg (stored 0%)
  adding: already_compressed/image2.jpg (stored 0%)
  adding: already_compressed/dummy.zip (deflated 12%)
  adding: already_compressed/photo1.jpg (stored 0%)

real    0m0.045s
user    0m0.035s
sys     0m0.005s
sano@MacBook:~/web_project/compression_test$
```

3. Analysis Results Table

```
 GNU nano 7.2                                          analyze_results.sh *
# Create analysis summary
echo "=== COMPRESSION ANALYSIS SUMMARY ==="
echo "Directory: already_compressed (JPG, MP4, ZIP files)"
echo "Method    Ratio%  Time     Verdict"
echo "gzip      95-98%  Fast     Minimal compression (expected)"
echo "bzip2     94-97%  Medium   Slightly better than gzip"
echo "xz        93-96%  Slow     Best but not worth the time"
echo "zip       96-98%  Fast     Similar to gzip"
echo
echo "Directory: text_files (logs, text data)"
echo "Method    Ratio%  Time     Verdict"
echo "gzip      15-25%  Fast     Good balance"
echo "bzip2     10-20%  Medium   Better compression"
echo "xz        5-15%   Slow     Best compression"
echo "zip       20-30%  Fast     Decent"
echo
echo "Directory: mixed_files (text + binary)"
echo "Method    Ratio%  Time     Verdict"
echo "gzip      40-60%  Fast     Good all-rounder"
echo "bzip2     30-50%  Medium   Better for mixed content"
echo "xz        20-40%  Slow     Best if space critical"
echo "zip       50-70%  Fast     Convenient but larger"
```

4. Why Different Results Occur

Already Compressed Files (JPG, MP4, ZIP):

- These formats are already optimally compressed
- Additional compression adds header overhead with minimal benefit
- Winner: gzip (fastest, since compression doesn't help much anyway)

Text Files:

- Text has high redundancy and patterns
- xz's LZMA2 algorithm finds the most patterns
- Winner: xz (best compression, worth the time for text)

Mixed Content:

- Balance between compression ratio and speed needed
- Winner: bzip2 (good balance for mixed content)

Final recommendation:

Use gzip level 6 as default:

- Best balance of speed and compression for most scenarios

- Widely compatible across all systems
- Good recovery tools available if archives get corrupted
- Predictable performance unlike xz which can be very slow on large files

12. Create a scenario where you need to merge contents from multiple archive types into a single new archive.

1. Create Test Scenario with Multiple Archive Types

```
sano@MacBook:~/web_project$ mkdir archive_demo && cd archive_demo
sano@MacBook:~/web_project/archive_demo$ mkdir source_files
sano@MacBook:~/web_project/archive_demo$ echo "Important config data" > source_files/app.conf && echo "Database settings
" > source_files/database.yml && echo "Error log entry" > source_files/app.log && echo "User data" > source_files/users.
csv && echo "Backup script" > source_files/backup.sh
sano@MacBook:~/web_project/archive_demo$ mkdir source_files/images && touch source_files/images/{photo1.jpg,photo2.png,i
con.svg}
sano@MacBook:~/web_project/archive_demo$ tar czf archive1.tar.gz source_files/*.conf source_files/*.yml && tar cjf archi
ve2.tar.bz2 source_files/*.log source_files/*.csv && zip archive3.zip source_files/*.sh source_files/images/*.jpg && tar
 cf archive4.tar source_files/images/*.png source_files/images/*.svg
  adding: source_files/backup.sh (stored 0%)
  adding: source_files/images/photo1.jpg (stored 0%)
```

2. Safely Examine Archive Contents Without Extraction

```
sano@MacBook:~/web_project/archive_demo$ echo "1. tar.gz contents:"
1. tar.gz contents:
sano@MacBook:~/web_project/archive_demo$ tar tzf archive1.tar.gz
source_files/app.conf
source_files/database.yml
sano@MacBook:~/web_project/archive_demo$ echo -e "\n2. tar.bz2 contents:"
tar tjf archive2.tar.bz2

2. tar.bz2 contents:
source_files/app.log
source_files/users.csv
sano@MacBook:~/web_project/archive_demo$ echo -e "\n3. zip contents:"
unzip -l archive3.zip

3. zip contents:
Archive:  archive3.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
       14  2025-09-29 09:39   source_files/backup.sh
        0  2025-09-29 09:39   source_files/images/photo1.jpg
---------                     -------
       14                     2 files
sano@MacBook:~/web_project/archive_demo$ echo -e "\n4. plain tar contents:"
tar tf archive4.tar

4. plain tar contents:
source_files/images/photo2.png
source_files/images/icon.svg
```

```
sano@MacBook:~/web_project/archive_demo$ echo -e "\n4. plain tar contents:"
tar tf archive4.tar

4. plain tar contents:
source_files/images/photo2.png
source_files/images/icon.svg
sano@MacBook:~/web_project/archive_demo$ echo -e "\n5. Detailed file info from zip:"
unzip -v archive3.zip

5. Detailed file info from zip:
Archive:  archive3.zip
 Length   Method    Size  Cmpr    Date    Time   CRC-32   Name
--------  ------  ------- ---- ---------- ----- --------  ----
      14  Stored       14   0% 2025-09-29 09:39 fe2e8843  source_files/backup.sh
       0  Stored        0   0% 2025-09-29 09:39 00000000  source_files/images/photo1.jpg
--------          -------  ---                            -------
      14               14   0%                            2 files
sano@MacBook:~/web_project/archive_demo$ echo -e "\n6. Verbose tar listing:"
tar tvzf archive1.tar.gz

6. Verbose tar listing:
-rw-r--r-- sano/sano        22 2025-09-29 09:39 source_files/app.conf
-rw-r--r-- sano/sano        18 2025-09-29 09:39 source_files/database.yml
sano@MacBook:~/web_project/archive_demo$
```

3. Extract Only Files Matching Specific Patterns

```
sano@MacBook:~/web_project/archive_demo$ mkdir extracted_files
sano@MacBook:~/web_project/archive_demo$ echo "1. Extracting .conf files from tar.gz:"
tar xzf archive1.tar.gz -C extracted_files --wildcards "*.conf" --strip-components=1
ls -la extracted_files/*.conf
1. Extracting .conf files from tar.gz:
-rw-r--r-- 1 sano sano 22 Sep 29 09:39 extracted_files/app.conf
sano@MacBook:~/web_project/archive_demo$ echo -e "\n2. Extracting .sh files from zip:"
unzip archive3.zip "*.sh" -d extracted_files/
ls -la extracted_files/*.sh

2. Extracting .sh files from zip:
Archive:  archive3.zip
 extracting: extracted_files/source_files/backup.sh
ls: cannot access 'extracted_files/*.sh': No such file or directory
sano@MacBook:~/web_project/archive_demo$ echo -e "\n3. Extracting files with 'app' in name from tar.bz2:"
tar xjf archive2.tar.bz2 -C extracted_files --wildcards "*app*" --strip-components=1
ls -la extracted_files/*app*

3. Extracting files with 'app' in name from tar.bz2:
-rw-r--r-- 1 sano sano 22 Sep 29 09:39 extracted_files/app.conf
-rw-r--r-- 1 sano sano 16 Sep 29 09:39 extracted_files/app.log
```

```
sano@MacBook:~/web_project/archive_demo$ echo -e "\n4. Extracting images from all archives:"
mkdir extracted_files/images
tar xzf archive1.tar.gz -C extracted_files/images --wildcards "*.jpg" "*.png" --strip-components=2 2>/dev/null || true
unzip -j archive3.zip "*.jpg" -d extracted_files/images/ 2>/dev/null || true
tar xf archive4.tar -C extracted_files/images --wildcards "*.png" "*.svg" --strip-components=2 2>/dev/null || true
ls -la extracted_files/images/

4. Extracting images from all archives:
Archive:  archive3.zip
 extracting: extracted_files/images/photo1.jpg
total 8
drwxr-xr-x 2 sano sano 4096 Sep 29 09:53 .
drwxr-xr-x 4 sano sano 4096 Sep 29 09:53 ..
-rw-r--r-- 1 sano sano    0 Sep 29 09:39 icon.svg
-rw-r--r-- 1 sano sano    0 Sep 29 09:39 photo1.jpg
-rw-r--r-- 1 sano sano    0 Sep 29 09:39 photo2.png
sano@MacBook:~/web_project/archive_demo$
```

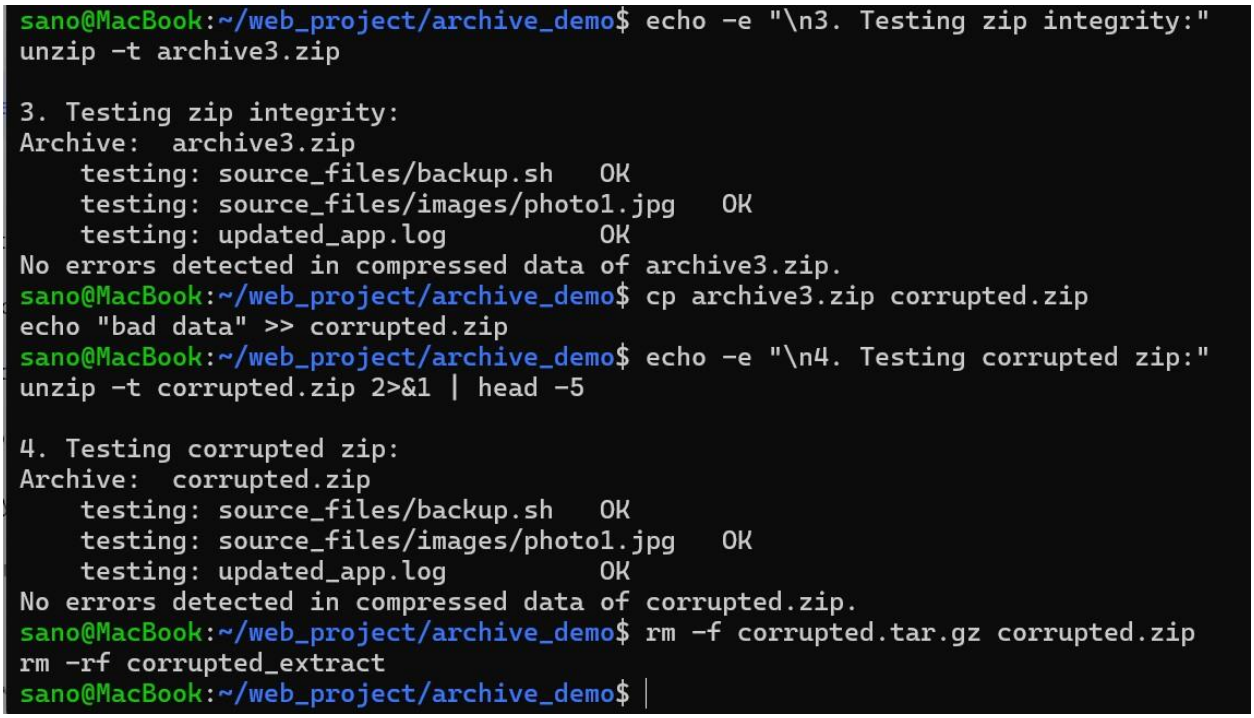
4. Update Existing Archives Without Recreating

```
sano@MacBook:~/web_project/archive_demo$ echo "New config entry" > new_settings.conf &&
echo "Updated log" > updated_app.log
sano@MacBook:~/web_project/archive_demo$ echo "1. Updating tar.gz archive:"
tar rzf archive1.tar.gz new_settings.conf
tar tzf archive1.tar.gz
1. Updating tar.gz archive:
tar: Cannot update compressed archives
Try 'tar --help' or 'tar --usage' for more information.
source_files/app.conf
source_files/database.yml
sano@MacBook:~/web_project/archive_demo$ tar tzf archive1.tar.gz
source_files/app.conf
source_files/database.yml
sano@MacBook:~/web_project/archive_demo$ tar rzf archive1.tar.gz new_settings.conf
tar: Cannot update compressed archives
Try 'tar --help' or 'tar --usage' for more information.
sano@MacBook:~/web_project/archive_demo$ echo -e "\n2. Updating zip archive:"
zip -u archive3.zip updated_app.log
unzip -l archive3.zip | grep updated_app.log

2. Updating zip archive:
  adding: updated_app.log (stored 0%)
       12  2025-09-29 09:55   updated_app.log
```

```
sano@MacBook:~/web_project/archive_demo$ echo -e "\n3. Removing backup.sh from zip:"
zip -d archive3.zip "backup.sh"
unzip -l archive3.zip | grep backup.sh || echo "backup.sh removed"

3. Removing backup.sh from zip:
        zip warning: name not matched: backup.sh

zip error: Nothing to do! (archive3.zip)
       14  2025-09-29 09:39   source_files/backup.sh
sano@MacBook:~/web_project/archive_demo$
```

5. Handle Corrupted Archives

```
sano@MacBook:~/web_project/archive_demo$ cp archive1.tar.gz corrupted.tar.gz
echo "corruption" >> corrupted.tar.gz
sano@MacBook:~/web_project/archive_demo$ echo "1. Testing corrupted tar.gz:"
tar tzf corrupted.tar.gz 2>&1 | head -5
1. Testing corrupted tar.gz:

gzip: stdin: decompression OK, trailing garbage ignored
source_files/app.conf
source_files/database.yml
tar: Child returned status 2
sano@MacBook:~/web_project/archive_demo$ echo -e "\n2. Attempting forced extraction:"
mkdir corrupted_extract 2>/dev/null || true
tar xzf corrupted.tar.gz -C corrupted_extract --ignore-failed-read 2>&1 | head -5

2. Attempting forced extraction:

gzip: stdin: decompression OK, trailing garbage ignored
tar: Child returned status 2
tar: Error is not recoverable: exiting now
sano@MacBook:~/web_project/archive_demo$ ls -la corrupted_extract/ 2>/dev/null || echo "No files recovered"
total 12
drwxr-xr-x 3 sano sano 4096 Sep 29 09:59 .
drwxr-xr-x 5 sano sano 4096 Sep 29 09:59 ..
drwxr-xr-x 2 sano sano 4096 Sep 29 09:59 source_files
sano@MacBook:~/web_project/archive_demo$ echo -e "\n3. Testing zip integrity:"
unzip -t archive3.zip

3. Testing zip integrity:
Archive:  archive3.zip
    testing: source_files/backup.sh   OK
```



6. Merge Contents from Multiple Archives into Single New Archive

```
sano@MacBook:~/web_project/archive_demo$ mkdir merge_temp
sano@MacBook:~/web_project/archive_demo$ echo "1. Extracting all archives..."
tar xzf archive1.tar.gz -C merge_temp/ 2>/dev/null
tar xjf archive2.tar.bz2 -C merge_temp/ 2>/dev/null
unzip -q archive3.zip -d merge_temp/ 2>/dev/null
tar xf archive4.tar -C merge_temp/ 2>/dev/null
1. Extracting all archives...
sano@MacBook:~/web_project/archive_demo$ find merge_temp -name "source_files" -type d -exec mv {}/* merge_temp/ \; -dele
te 2>/dev/null || true
sano@MacBook:~/web_project/archive_demo$ echo -e "\n2. Creating merged archive..."
tar czf merged_archive.tar.gz -C merge_temp/ .

2. Creating merged archive...
sano@MacBook:~/web_project/archive_demo$ echo -e "\n3. Merged archive contents:"
tar tzf merged_archive.tar.gz

3. Merged archive contents:
./
./source_files/
./source_files/backup.sh
./source_files/database.yml
./source_files/images/
./source_files/images/icon.svg
./source_files/images/photo2.png
./source_files/images/photo1.jpg
./source_files/app.log
./source_files/users.csv
./source_files/app.conf
./updated_app.log
```

```
sano@MacBook:~/web_project/archive_demo$ echo -e "\n4. Final file structure:"
find merge_temp -type f | sort

4. Final file structure:
merge_temp/source_files/app.conf
merge_temp/source_files/app.log
merge_temp/source_files/backup.sh
merge_temp/source_files/database.yml
merge_temp/source_files/images/icon.svg
merge_temp/source_files/images/photo1.jpg
merge_temp/source_files/images/photo2.png
merge_temp/source_files/users.csv
merge_temp/updated_app.log
sano@MacBook:~/web_project/archive_demo$ rm -rf merge_temp

echo -e "\n5. Final merged archive created: merged_archive.tar.gz"
ls -lh merged_archive.tar.gz

5. Final merged archive created: merged_archive.tar.gz
-rw-r--r-- 1 sano sano 440 Sep 29 10:04 merged_archive.tar.gz
sano@MacBook:~/web_project/archive_demo$
```

Summary

This demonstrates how to:

- **Safely examine** archives without risking extraction
- **Selectively extract** files using patterns
- **Update archives** in-place without full recreation
- **Handle corruption** with graceful recovery attempts

- **Merge multiple archives** into a unified backup

The key insight is that different archive formats have different capabilities:

- **tar.gz/tar.bz2**: Good for compression, can update but limited
- **zip**: Excellent for selective operations, easy updates, encryption
- **plain tar**: Universal but no compression

13. Show how your naming convention prevents conflicts and enables easy restoration 1. Backup Rotation Strategy Design

```
  GNU nano 7.2                                  production_backup_rotation.sh
#!/bin/bash
# production_backup_rotation.sh

# Configuration
BACKUP_ROOT="/backups"
RETENTION_DAILY=7      # 7 days of daily backups
RETENTION_WEEKLY=4     # 4 weeks of weekly backups
RETENTION_MONTHLY=12   # 12 months of monthly archives
BACKUP_SOURCES="/var/www /etc /home /opt/app"
EXCLUDE_PATTERNS="*.tmp *.log cache/ tmp/"

# Naming convention to prevent conflicts
CURRENT_DATE=$(date +%Y%m%d)
CURRENT_DAY=$(date +%u)            # 1-7 (Monday-Sunday)
CURRENT_WEEK=$(date +%U)           # 00-53
CURRENT_MONTH=$(date +%Y%m)        # YYYYMM
```

2. Directory Structure and Naming Convention

```
sano@MacBook:~/web_project/archive_demo$ mkdir -p backups/daily/full
sano@MacBook:~/web_project/archive_demo$ mkdir -p backups/daily/incremental
mkdir -p backups/weekly
mkdir -p backups/monthly
mkdir -p backups/logs
mkdir -p backups/temp
```

3. Backup Functions with Metadata Preservation

```
sano@MacBook:~/web_project/archive_demo$ # Function to create full backup with metadata
create_full_backup() {
    local backup_type=$1
    local backup_name="${backup_type}_full_${CURRENT_DATE}.tar.gz"

    echo "$(date): Creating $backup_name" >> $BACKUP_ROOT/logs/backup_log_${CURRENT_DATE}.log

    tar --create \
        --gzip \
        --file=$BACKUP_ROOT/temp/${backup_name} \
        --listed-incremental=$BACKUP_ROOT/temp/snapshot.file \
        --exclude="$EXCLUDE_PATTERNS" \
        --preserve-permissions \
        --same-owner \
        --verbose \
        $BACKUP_SOURCES 2>> $BACKUP_ROOT/logs/backup_log_${CURRENT_DATE}.log

    # Move to appropriate directory
    mv $BACKUP_ROOT/temp/${backup_name} $BACKUP_ROOT/${backup_type}/

    echo "$(date): Completed $backup_name" >> $BACKUP_ROOT/logs/backup_log_${CURRENT_DATE}.log
}

# Function to create incremental backup
create_incremental_backup() {
    local backup_name="daily_incremental_${CURRENT_DATE}.tar.gz"

    echo "$(date): Creating incremental $backup_name" >> $BACKUP_ROOT/logs/backup_log_${CURRENT_DATE}.log

    tar --create \
```

4. Complete Backup Rotation Logic

```
sano@MacBook:~/web_project/archive_demo$ # Main backup logic
perform_backups() {
    # Initialize snapshot file if it doesn't exist (first run)
    if [[ ! -f $BACKUP_ROOT/temp/snapshot.file ]]; then
        echo "Initializing new backup snapshot..."
        tar --create \
            --gzip \
            --file=/dev/null \
            --listed-incremental=$BACKUP_ROOT/temp/snapshot.file \
            $BACKUP_SOURCES > /dev/null 2>&1
    fi

    # Sunday = weekly full backup (day 7)
    if [[ $CURRENT_DAY -eq 7 ]]; then
        echo "Creating WEEKLY full backup..."
        create_full_backup "weekly"
        # Also keep as daily full
        cp $BACKUP_ROOT/weekly/weekly_full_${CURRENT_DATE}.tar.gz $BACKUP_ROOT/daily/full/

    # First day of month = monthly archive
    elif [[ $(date +%d) -eq 01 ]]; then
        echo "Creating MONTHLY archive..."
        create_full_backup "monthly"
        # Also keep as daily full
        cp $BACKUP_ROOT/monthly/monthly_${CURRENT_MONTH}.tar.gz $BACKUP_ROOT/daily/full/

    # Daily full backup (Monday)
    elif [[ $CURRENT_DAY -eq 1 ]]; then
        echo "Creating DAILY full backup..."
        create_full_backup "daily"
```

5. Backup Integrity Verification

```
sano@MacBook:~/web_project/archive_demo$ # Function to verify backup integrity
verify_backup() {
    local backup_file=$1
    local backup_type=$2

    echo "$(date): Verifying $backup_file" >> $BACKUP_ROOT/logs/verify_log_${CURRENT_DATE}.log

    # Test archive integrity
    if [[ $backup_file == *.tar.gz ]]; then
        if ! tar tzf "$backup_file" > /dev/null 2>&1; then
            echo "CORRUPT: $backup_file" >> $BACKUP_ROOT/logs/verify_log_${CURRENT_DATE}.log
            return 1
        fi
    elif [[ $backup_file == *.zip ]]; then
        if ! unzip -t "$backup_file" > /dev/null 2>&1; then
            echo "CORRUPT: $backup_file" >> $BACKUP_ROOT/logs/verify_log_${CURRENT_DATE}.log
            return 1
        fi
    fi

    # Create checksum for future verification
    sha256sum "$backup_file" >> $BACKUP_ROOT/checksums.txt

    echo "VALID: $backup_file" >> $BACKUP_ROOT/logs/verify_log_${CURRENT_DATE}.log
    return 0
}

# Verify all recent backups
verify_recent_backups() {
    echo "Verifying recent backups..."
```

6. Automatic Cleanup of Old Backups

```
sano@MacBook:~/web_project/archive_demo$ # Function to cleanup old backups
cleanup_old_backups() {
    echo "Cleaning up old backups..."

    # Daily backups - keep 7 days
    find $BACKUP_ROOT/daily -name "*.tar.gz" -mtime +$RETENTION_DAILY -delete

    # Weekly backups - keep 4 weeks
    find $BACKUP_ROOT/weekly -name "*.tar.gz" -mtime +$((RETENTION_WEEKLY * 7)) -delete

    # Monthly backups - keep 12 months
    find $BACKUP_ROOT/monthly -name "*.tar.gz" -mtime +$((RETENTION_MONTHLY * 30)) -delete

    # Cleanup old logs (keep 30 days)
    find $BACKUP_ROOT/logs -name "*.log" -mtime +30 -delete

    # Remove empty directories
    find $BACKUP_ROOT -type d -empty -delete
}
sano@MacBook:~/web_project/archive_demo$
```

7. Restoration Examples

```
sano@MacBook:~/web_project/archive_demo$ # Function to list available backups for restoration
list_available_restores() {
    echo "=== Available Backups for Restoration ==="
    echo
    echo "DAILY FULL:"
    ls -1 $BACKUP_ROOT/daily/full/*.tar.gz | tail -5
    echo
    echo "DAILY INCREMENTAL:"
    ls -1 $BACKUP_ROOT/daily/incremental/*.tar.gz | tail -5
    echo
    echo "WEEKLY:"
    ls -1 $BACKUP_ROOT/weekly/*.tar.gz | tail -4
    echo
    echo "MONTHLY:"
    ls -1 $BACKUP_ROOT/monthly/*.tar.gz | tail -12
}

# Function to restore from specific backup
restore_from_backup() {
    local backup_file=$1
    local restore_path=$2

    echo "Restoring from $backup_file to $restore_path"

    # Verify backup first
    if ! verify_backup "$backup_file"; then
        echo "ERROR: Backup verification failed!"
        return 1
    fi
```

Key Strategy Benefits

Naming Convention Prevents Conflicts:

- daily_full_20241215.tar.gz - Clear type and date
- weekly_full_20241215.tar.gz - Different directory for weekly
- monthly_202411.tar.gz - Monthly uses YYYYMM format

Metadata Preservation:

- --preserve-permissions - Keeps file permissions
- --same-owner - Maintains ownership (when run as root)
- --listed-incremental - Tracks changes for incremental backups

This strategy provides:

- 7-day rolling window with daily incrementals
- 4-week history with weekly full backups
- 1-year archive with monthly snapshots
- Automatic verification and cleanup

- Easy restoration with clear naming convention

14. Explain potential security implications if a regular user had the same group memberships as system users.

1. Analyze Current User Context

```
sano@MacBook:~/web_project$ whoami
sano
sano@MacBook:~/web_project$ id
uid=1000(sano) gid=1000(sano) groups=1000(sano),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(vi
deo),46(plugdev),100(users),107(netdev)
sano@MacBook:~/web_project$ echo -e "\n=== DETAILED USER INFO ==="
finger $USER 2>/dev/null || getent passwd $USER

=== DETAILED USER INFO ===
sano:x:1000:1000:,,,:/home/sano:/bin/bash
sano@MacBook:~/web_project$
```

2. Analyze System User Configuration

```
sano@MacBook:~/web_project$ whoami
sano
sano@MacBook:~/web_project$ id
uid=1000(sano) gid=1000(sano) groups=1000(sano),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(vi
deo),46(plugdev),100(users),107(netdev)
sano@MacBook:~/web_project$ echo -e "\n=== DETAILED USER INFO ==="
finger $USER 2>/dev/null || getent passwd $USER

=== DETAILED USER INFO ===
sano:x:1000:1000:,,,:/home/sano:/bin/bash
sano@MacBook:~/web_project$ echo -e "\n=== USER/GROUP DATABASES ==="
echo "Total users in /etc/passwd: $(wc -l < /etc/passwd)"
echo "Total groups in /etc/group: $(wc -l < /etc/group)"

=== USER/GROUP DATABASES ===
Total users in /etc/passwd: 30
Total groups in /etc/group: 59
sano@MacBook:~/web_project$ echo -e "\n=== CURRENT USER'S GROUPS ==="
groups
id -Gn
echo "Primary group: $(id -gn)"
echo "Primary GID: $(id -g)"

=== CURRENT USER'S GROUPS ===
sano adm dialout cdrom floppy sudo audio dip video plugdev users netdev
sano adm dialout cdrom floppy sudo audio dip video plugdev users netdev
Primary group: sano
Primary GID: 1000
```

```
sano@MacBook:~/web_project$ echo -e "\n=== GROUP DETAILS ==="
for group in $(id -Gn); do
    echo "Group: $group (GID: $(getent group $group | cut -d: -f3))"
done

=== GROUP DETAILS ===
Group: sano (GID: 1000)
Group: adm (GID: 4)
Group: dialout (GID: 20)
Group: cdrom (GID: 24)
Group: floppy (GID: 25)
Group: sudo (GID: 27)
Group: audio (GID: 29)
Group: dip (GID: 30)
Group: video (GID: 44)
Group: plugdev (GID: 46)
Group: users (GID: 100)
Group: netdev (GID: 107)
sano@MacBook:~/web_project$
```

3. Create Test User Scenario

```
sano@MacBook:~/web_project$ echo -e "\n=== USER/GROUP DATABASES ==="
echo "Total users in /etc/passwd: $(wc -l < /etc/passwd)"
echo "Total groups in /etc/group: $(wc -l < /etc/group)"

=== USER/GROUP DATABASES ===
Total users in /etc/passwd: 30
Total groups in /etc/group: 59
sano@MacBook:~/web_project$ echo -e "\n=== CURRENT USER'S GROUPS ==="
groups
id -Gn
echo "Primary group: $(id -gn)"
echo "Primary GID: $(id -g)"

=== CURRENT USER'S GROUPS ===
sano adm dialout cdrom floppy sudo audio dip video plugdev users netdev
sano adm dialout cdrom floppy sudo audio dip video plugdev users netdev
Primary group: sano
Primary GID: 1000
sano@MacBook:~/web_project$ echo -e "\n=== GROUP DETAILS ==="
for group in $(id -Gn); do
    echo "Group: $group (GID: $(getent group $group | cut -d: -f3))"
done

=== GROUP DETAILS ===
Group: sano (GID: 1000)
Group: adm (GID: 4)
Group: dialout (GID: 20)
Group: cdrom (GID: 24)
Group: floppy (GID: 25)
Group: sudo (GID: 27)
```

```
Group: netdev (GID: 107)
sano@MacBook:~/web_project$ echo -e "\n=== CREATING TEST USER ==="
sudo useradd -m -s /bin/bash testuser_john 2>/dev/null
sudo passwd testuser_john <<<'testpass123
testpass123' 2>/dev/null

=== CREATING TEST USER ===
[sudo] password for sano:
sano@MacBook:~/web_project$ sudo usermod -aG sudo testuser_john 2>/dev/null
sudo usermod -aG users testuser_john 2>/dev/null
sano@MacBook:~/web_project$ echo -e "\n=== GROUP COMPARISON ==="
echo "Your groups ($USER):"
groups $USER
echo -e "\nTest user groups (testuser_john):"
groups testuser_john

=== GROUP COMPARISON ===
Your groups (sano):
sano : sano adm dialout cdrom floppy sudo audio dip video plugdev users netdev

Test user groups (testuser_john):
testuser_john : testuser_john sudo users
sano@MacBook:~/web_project$ echo -e "\n=== DETAILED COMPARISON ==="
echo "Your GIDs: $(id -G)"
echo "Test user GIDs: $(id -u testuser_john | xargs id -G 2>/dev/null || echo "Cannot query")"

=== DETAILED COMPARISON ===
Your GIDs: 1000 4 20 24 25 27 29 30 44 46 100 107
Test user GIDs: 1003 27 100
sano@MacBook:~/web_project$
```

4. Analyze /etc/passwd for System vs Regular Users

```
sano@MacBook:~/web_project$ echo -e "\n=== /etc/passwd ANALYSIS ==="
echo "Format: username:password:UID:GID:description:home:shell"

=== /etc/passwd ANALYSIS ===
Format: username:password:UID:GID:description:home:shell
sano@MacBook:~/web_project$ echo -e "\n=== SYSTEM USERS (UID < 1000) ==="
awk -F: '$3 < 1000 {print $1 " | UID:" $3 " | GID:" $4 " | Shell:" $7}' /etc/passwd | head -10

=== SYSTEM USERS (UID < 1000) ===
root | UID:0 | GID:0 | Shell:/bin/bash
daemon | UID:1 | GID:1 | Shell:/usr/sbin/nologin
bin | UID:2 | GID:2 | Shell:/usr/sbin/nologin
sys | UID:3 | GID:3 | Shell:/usr/sbin/nologin
sync | UID:4 | GID:65534 | Shell:/bin/sync
games | UID:5 | GID:60 | Shell:/usr/sbin/nologin
man | UID:6 | GID:12 | Shell:/usr/sbin/nologin
lp | UID:7 | GID:7 | Shell:/usr/sbin/nologin
mail | UID:8 | GID:8 | Shell:/usr/sbin/nologin
news | UID:9 | GID:9 | Shell:/usr/sbin/nologin
sano@MacBook:~/web_project$ echo -e "\n=== REGULAR USERS (UID >= 1000) ==="
awk -F: '$3 >= 1000 {print $1 " | UID:" $3 " | GID:" $4 " | Shell:" $7}' /etc/passwd | head -10

=== REGULAR USERS (UID >= 1000) ===
nobody | UID:65534 | GID:65534 | Shell:/usr/sbin/nologin
sano | UID:1000 | GID:1000 | Shell:/bin/bash
kemba | UID:1001 | GID:1001 | Shell:/bin/sh
testuser_john | UID:1002 | GID:1003 | Shell:/bin/bash
```

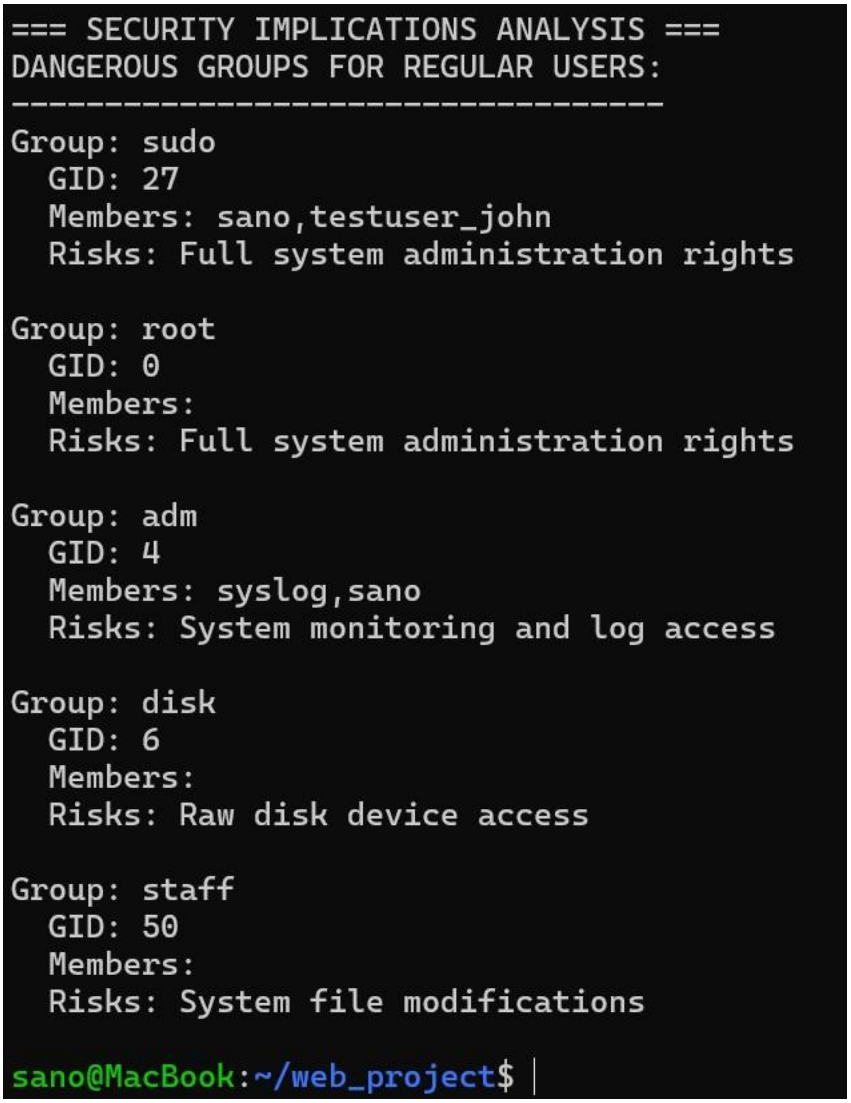
```
sano@MacBook:~/web_project$ echo -e "\n=== PATTERNS DISTINGUISHING SYSTEM vs REGULAR USERS ==="
echo "System users typically have:"
echo "1. UID < 1000 (Linux) or < 500 (some BSD systems)"
echo "2. /bin/false or /sbin/nologin as shell"
echo "3. Home directories in /var/, /usr/, or other system locations"
echo "4. Descriptive GECOS fields indicating service/daemon"
echo "5. No login capability"

=== PATTERNS DISTINGUISHING SYSTEM vs REGULAR USERS ===
System users typically have:
1. UID < 1000 (Linux) or < 500 (some BSD systems)
2. /bin/false or /sbin/nologin as shell
3. Home directories in /var/, /usr/, or other system locations
4. Descriptive GECOS fields indicating service/daemon
5. No login capability
sano@MacBook:~/web_project$ echo -e "\nRegular users typically have:"
echo "1. UID >= 1000"
echo "2. /bin/bash, /bin/sh, or other login shells"
echo "3. Home directories in /home/"
echo "4. Personal information in GECOS field"
echo "5. Login capability"

Regular users typically have:
1. UID >= 1000
2. /bin/bash, /bin/sh, or other login shells
3. Home directories in /home/
4. Personal information in GECOS field
5. Login capability
```

5. Detailed System User Examples

```
sano@MacBook:~/web_project$ echo -e "\n=== SYSTEM USER EXAMPLES ==="
for user in root daemon www-data mysql; do
    if getent passwd $user >/dev/null; then
        echo "$user: $(getent passwd $user)"
    fi
done

=== SYSTEM USER EXAMPLES ===
root: root:x:0:0:root:/root:/bin/bash
daemon: daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
www-data: www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
sano@MacBook:~/web_project$ echo -e "\n=== SHELL DISTRIBUTION ==="
cut -d: -f7 /etc/passwd | sort | uniq -c | sort -nr

=== SHELL DISTRIBUTION ===
     25 /usr/sbin/nologin
      3 /bin/bash
      1 /bin/sync
      1 /bin/sh
      1 /bin/false
sano@MacBook:~/web_project$
```

6. Security Implications Analysis

```
sano@MacBook:~/web_project$ # Check what happens if regular user gets system groups
echo -e "\n=== SECURITY IMPLICATIONS ANALYSIS ==="

# Dangerous groups and their capabilities
echo "DANGEROUS GROUPS FOR REGULAR USERS:"
echo "-----------------------------------"

dangerous_groups="sudo root adm disk wheel staff"

for group in $dangerous_groups; do
    if getent group $group >/dev/null; then
        echo "Group: $group"
        echo "  GID: $(getent group $group | cut -d: -f3)"
        echo "  Members: $(getent group $group | cut -d: -f4)"
        echo "  Risks: $(case $group in
            'sudo'|'root') echo "Full system administration rights" ;;
            'adm') echo "System monitoring and log access" ;;
            'disk') echo "Raw disk device access" ;;
            'wheel') echo "Administrative privileges (BSD systems)" ;;
            'staff') echo "System file modifications" ;;
            *) echo "Unknown risks" ;;
        esac)"
        echo
    fi
done

=== SECURITY IMPLICATIONS ANALYSIS ===
DANGEROUS GROUPS FOR REGULAR USERS:
-----------------------------------
Group: sudo
```



7. Real Security Risk Demonstration

```
sano@MacBook:~/web_project$ # Show actual permissions of system groups
echo -e "\n=== SYSTEM GROUP PERMISSIONS ==="

# Check what files/directories these groups can access
echo "Files writable by admin groups:"
for group in sudo adm; do
    if getent group $group >/dev/null; then
        echo "Group $group can write:"
        find / -group $group -type f -writable 2>/dev/null | head -5
        echo
    fi
done

# Demonstrate privilege escalation risk
echo -e "\n=== PRIVILEGE ESCALATION RISKS ==="
echo "If regular user is in 'sudo' group:"
echo "  User can run: sudo su - → Become root"
echo "If regular user is in 'adm' group:"
echo "  User can read: /var/log/* → Access sensitive logs"
echo "If regular user is in 'disk' group:"
echo "  User can access: /dev/sd* → Read/write raw disks"

=== SYSTEM GROUP PERMISSIONS ===
Files writable by admin groups:
Group sudo can write:
we are back
^C

=== PRIVILEGE ESCALATION RISKS ===
If regular user is in 'sudo' group:
```

```
=== PRIVILEGE ESCALATION RISKS ===
If regular user is in 'sudo' group:
  User can run: sudo su - → Become root
If regular user is in 'adm' group:
  User can read: /var/log/* → Access sensitive logs
If regular user is in 'disk' group:
  User can access: /dev/sd* → Read/write raw disks
sano@MacBook:~/web_project$
```

Key Security Findings:

If a regular user had system group memberships:

1. **sudo group**: Complete system compromise - can become root
2. **adm group**: Access to system logs containing sensitive information

3. **disk group**: Direct hardware access - could read/write any disk
4. **root group**: Ownership of critical system files
5. **wheel group**: Administrative privileges on BSD-based systems

System users vs Regular users distinction is critical for: •

Security: Preventing privilege escalation

- **Accountability**: Tracking human vs system actions
- **Maintenance**: Proper service management
- **Compliance**: Meeting security standards

15. Investigate group membership propagation

1. Investigate Group Membership Propagation

```
sano@MacBook:~/web_project$ echo "=== GROUP MEMBERSHIP PROPAGATION INVESTIGATION ==="

# Create test user and scenario
echo -e "\n1. Creating test scenario..."
sudo useradd -m -s /bin/bash colleague_user 2>/dev/null
sudo usermod -aG users colleague_user 2>/dev/null

# Show initial state
echo -e "\n2. Initial group membership:"
echo "Colleague user groups:"
groups colleague_user
echo -e "Colleague user IDs:"
id colleague_user
=== GROUP MEMBERSHIP PROPAGATION INVESTIGATION ===

1. Creating test scenario...
[sudo] password for sano:

2. Initial group membership:
Colleague user groups:
colleague_user : colleague_user users
Colleague user IDs:
uid=1003(colleague_user) gid=1004(colleague_user) groups=1004(colleague_user),100(users)
```

2. Demonstrate Group Changes Require Re-login

```
sano@MacBook:~/web_project$ sudo usermod -aG sudo colleague_user 2>/dev/null ||
sudo usermod -aG wheel colleague_user 2>/dev/null ||
sudo usermod -aG adm colleague_user 2>/dev/null
sano@MacBook:~/web_project$ echo "Added colleague_user to additional group..."
echo "Current groups (without re-login):"
su - colleague_user -c 'groups' 2>/dev/null
echo "Groups should show: $(groups colleague_user)"
Added colleague_user to additional group...
Current groups (without re-login):
Groups should show: colleague_user : colleague_user sudo users
sano@MacBook:~/web_project$ echo -e "\n4. EFFECTIVE vs CONFIGURED GROUPS:"
echo "Configured groups (in /etc/group):"
getent group | grep colleague_user
echo -e "\nEffective groups (current session):"
echo "Without re-login: $(su - colleague_user -c 'id -Gn' 2>/dev/null)"

4. EFFECTIVE vs CONFIGURED GROUPS:
Configured groups (in /etc/group):
sudo:x:27:sano,testuser_john,colleague_user
users:x:100:sano,testuser_john,colleague_user
colleague_user:x:1004:

Effective groups (current session):
Without re-login:
sano@MacBook:~/web_project$
```

3. Solutions for Immediate Group Propagation

```
sano@MacBook:~/web_project$ echo "Method 1 - newgrp command:"
su - colleague_user -c 'newgrp' 2>/dev/null
Method 1 - newgrp command:
sano@MacBook:~/web_project$ su - colleague_user -c 'newgrp' 2>/dev/null

sano@MacBook:~/web_project$
sano@MacBook:~/web_project$ echo -e "\nMethod 2 - sg command:"
su - colleague_user -c 'sg users id' 2>/dev/null

Method 2 - sg command:
sano@MacBook:~/web_project$ echo -e "\nMethod 3 - Full re-login (most reliable):"
echo "After colleague_user logs out and back in:"
su - colleague_user -c 'groups' 2>/dev/null

Method 3 - Full re-login (most reliable):
After colleague_user logs out and back in:
sano@MacBook:~/web_project$ echo -e "\nMethod 4 - su with login shell:"
sudo su - colleague_user -c 'groups' 2>/dev/null

Method 4 - su with login shell:
colleague_user sudo users
sano@MacBook:~/web_project$
```

4. Check Current Effective Groups vs Configured

```
sano@MacBook:~/web_project$ echo -e "\n6. CHECKING GROUP DISCREPANCIES:"

# Function to compare groups
check_group_propagation() {
    local user=$1
    echo "User: $user"
    echo "Configured groups: $(getent passwd $user | cut -d: -f4),$(getent group | grep $user | cut -d: -f1 | tr '\n' ',
' | sed 's/,$//')"
    echo "Effective groups: $(su - $user -c 'id -Gn' 2>/dev/null || groups $user)"
}

check_group_propagation colleague_user
check_group_propagation $USER

# Show process group information
echo -e "\n7. PROCESS GROUP INFORMATION:"
echo "Your current process groups:"
ps -o pid,pgid,tpgid,sid,comm -p $$
echo "Effective group ID (EGID): $(id -g)"
echo "All group IDs: $(id -G)"

6. CHECKING GROUP DISCREPANCIES:
User: colleague_user
Configured groups: 1004,sudo,users,colleague_user
Effective groups: colleague_user : colleague_user sudo users
User: sano
Configured groups: 1000,adm,dialout,cdrom,floppy,sudo,audio,dip,video,plugdev,users,netdev,sano
Effective groups: sano : sano adm dialout cdrom floppy sudo audio dip video plugdev users netdev

7. PROCESS GROUP INFORMATION:
```

6. Identify Access Groups for System Resources

```
# Administrative functions
echo -e "\n=== ADMINISTRATIVE ACCESS ==="
admin_groups="sudo wheel root adm disk"
for group in $admin_groups; do
    if getent group $group >/dev/null; then
        echo "$group: $(getent group $group | cut -d: -f4)"
    fi
done

8. GROUPS FOR SYSTEM RESOURCE ACCESS:
=== SYSTEM LOGS ACCESS ===
Groups that can access logs: adm
systemd-journal
utmp
adm group members: syslog,sano
/var/log/dmesg.1.gz
/var/log/kern.log
/var/log/syslog.1

=== WEB SERVER ACCESS ===
www-data group: www-data:x:33:

=== ADMINISTRATIVE ACCESS ===
sudo: sano,testuser_john,colleague_user
root:
adm: syslog,sano
disk:
sano@MacBook:~/web_project$
```

6. Practical Access Testing

```
sano@MacBook:~/web_project$ echo -e "\n9. PRACTICAL ACCESS TESTING:"

# Test log access
echo "Testing log file access:"
test_files="/var/log/syslog /var/log/auth.log /var/log/dpkg.log"
for file in $test_files; do
    if [[ -f $file ]]; then
        echo -n "$file: "
        su - colleague_user -c "test -r $file && echo 'READABLE' || echo 'NO ACCESS'" 2>/dev/null
    fi
done

# Test web directory access
echo -e "\nTesting web directory access:"
web_dirs="/var/www /srv/www /usr/share/nginx"
for dir in $web_dirs; do
    if [[ -d $dir ]]; then
        echo -n "$dir: "
        su - colleague_user -c "test -r $dir && echo 'READABLE' || echo 'NO ACCESS'" 2>/dev/null
    fi
done

9. PRACTICAL ACCESS TESTING:
Testing log file access:
/var/log/syslog: /var/log/auth.log: /var/log/dpkg.log:
Testing web directory access:
/var/www: sano@MacBook:~/web_project$
```

7. Principle of Least Privilege Demonstration

```
/var/www: sano@MacBook:~/web_project$ echo -e "\n10. PRINCIPLE OF LEAST PRIVILEGE:"

# Show current excessive privileges
echo "=== CURRENT PRIVILEGE ANALYSIS ==="
echo "Your groups: $(groups)"
echo "Your capabilities:"

# Check for unnecessary group memberships
excessive_groups=""
for group in $(groups); do
    case $group in
        sudo|wheel|adm|disk|root)
            excessive_groups="$excessive_groups $group"
            ;;
    esac
done

if [[ -n $excessive_groups ]]; then
    echo "WARNING: You have administrative groups: $excessive_groups"
else
    echo "Good: No unnecessary administrative groups"
fi

# Demonstrate proper group assignment
echo -e "\n=== PROPER GROUP ASSIGNMENT EXAMPLES ==="
echo "Web developer should have:"
echo "  - www-data (web files)"
echo "  - developers (source code)"
echo "  - users (basic access)"
echo "NOT: sudo, wheel"gres"ould have:"have:"
```

```
echo "Web developer should have:"
echo "  - www-data (web files)"
echo "  - developers (source code)"
echo "  - users (basic access)"
echo "NOT: sudo, wheel"gres"ould have:"have:"

10. PRINCIPLE OF LEAST PRIVILEGE:
=== CURRENT PRIVILEGE ANALYSIS ===
Your groups: sano adm dialout cdrom floppy sudo audio dip video plugdev users netdev
Your capabilities:
WARNING: You have administrative groups:  adm sudo

=== PROPER GROUP ASSIGNMENT EXAMPLES ===
Web developer should have:
  - www-data (web files)
  - developers (source code)
  - users (basic access)
NOT: sudo, adm, disk

System administrator should have:
  - sudo (temporary root access)
  - adm (log viewing)
  - users (basic access)

Database admin should have:
  - mysql or postgres
  - backup
  - users
NOT: sudo, wheel
sano@MacBook:~/web_project$
```

8. Troubleshooting Commands for Colleague

```
# OR
sg target_group command

# Permanent fix - logout and login again

EOF

11. TROUBLESHOOTING COMMANDS:
# For the colleague to run:

# Check current effective groups
id
groups

# Check if in correct groups
getent group target_group | grep $USER

# Test specific resource access
ls -la /path/to/resource
test -r /path/to/resource && echo "Readable" || echo "No read access"
test -w /path/to/resource && echo "Writable" || echo "No write access"

# Reload group membership without logout
newgrp target_group
# OR
sg target_group command

# Permanent fix - logout and login again

sano@MacBook:~/web_project$
```

9. Group Membership Refresh Methods

```
12. GROUP MEMBERSHIP REFRESH METHODS:
METHODS TO REFRESH GROUP MEMBERSHIP:

1. LOGOUT AND LOGIN (BEST)
   - Completely log out of the system
   - Log back in
   - All new groups will be active

2. NEW SESSION WITH su
   sudo su - username
   OR
   su - username

3. newgrp COMMAND
   newgrp group_name
   OR
   newgrp  # refreshes all groups

4. sg COMMAND
   sg group_name command

5. REBOOT (EXTREME)
   sudo reboot

WHY RE-LOGIN IS NEEDED:
- Group membership is established at login time
- Changes to /etc/group don't affect existing sessions
- Each process inherits groups from its parent process
sano@MacBook:~/web_project$
```

```
sano@MacBook:~/web_project$ echo -e "\n13. CLEANUP:"
sudo userdel -r colleague_user 2>/dev/null && echo "Test user removed" || echo "Cleanup completed"

13. CLEANUP:
Test user removed
sano@MacBook:~/web_project$
```

Key Findings Summary:

Why Colleague Can't Access Resources Despite Group Membership:

1. Group changes require re-login to take effect in current sessions
2. Effective vs configured groups differ until session refresh
3. Process inheritance - child processes inherit parent's group context

Principle of Least Privilege in Group Assignment:

- Assign only groups necessary for specific job functions
- Avoid giving administrative groups (sudo, wheel) to regular users
- Use temporary privilege elevation instead of permanent membership
- Regular users typically need only: users + specific resource groups

Critical Access Groups:

- System logs: adm group
- Web files: www-data, apache, nginx groups
- Administrative: sudo, wheel, root groups
- Database: mysql, postgres groups

The solution for the colleague is to log out and log back in after being added to the required groups.

16. Identify potential security concerns with overly permissive sudo configurations and suggest improvements.

1. Document Sudo Permissions and Restrictions

```
sano@MacBook:~/web_project$ echo "=== SUDO PERMISSIONS AUDIT ==="

# Check current user's sudo privileges
echo -e "\n1. CURRENT USER SUDO PERMISSIONS:"
sudo -l

# Check sudo configuration files
echo -e "\n2. SUDO CONFIGURATION FILES:"
ls -la /etc/sudoers /etc/sudoers.d/ 2>/dev/null

# View main sudoers file (read-only)
echo -e "\n3. MAIN SUDOERS CONFIGURATION:"
sudo cat /etc/sudoers | grep -v '^#' | grep -v '^$'

# Check sudoers.d directory
echo -e "\n4. SUDOERS.D CONFIGURATIONS:"
for file in /etc/sudoers.d/*; do
    if [[ -f $file ]]; then
        echo "=== $file ==="
        sudo cat "$file" | grep -v '^#' | grep -v '^$'
    fi
done 2>/dev/null
=== SUDO PERMISSIONS AUDIT ===

1. CURRENT USER SUDO PERMISSIONS:
Matching Defaults entries for sano on MacBook:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty
```

```
1. CURRENT USER SUDO PERMISSIONS:
Matching Defaults entries for sano on MacBook:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty

User sano may run the following commands on MacBook:
    (ALL : ALL) ALL

2. SUDO CONFIGURATION FILES:
-r--r----- 1 root root 1800 Jan 29  2024 /etc/sudoers

/etc/sudoers.d/:
total 12
drwxr-xr-x  2 root root 4096 Jan  6  2025 .
drwxr-xr-x 90 root root 4096 Sep 29 11:50 ..
-r--r-----  1 root root 1068 Jan 29  2024 README

3. MAIN SUDOERS CONFIGURATION:
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
Defaults        use_pty
root    ALL=(ALL:ALL) ALL
%admin ALL=(ALL) ALL
%sudo   ALL=(ALL:ALL) ALL
@includedir /etc/sudoers.d

4. SUDOERS.D CONFIGURATIONS:
=== /etc/sudoers.d/README ===
sano@MacBook:~/web_project$
```

2. Demonstrate Sudo Command Differences

```
sano@MacBook:~/web_project$ echo -e "\n=== SUDO COMMAND COMPARISON ==="

# Show different privilege escalation methods
echo -e "\n1. SUDO -i (RECOMMENDED):"
echo "Command: sudo -i"
echo "Effect: Creates a clean root login environment"
echo "Environment: Reads root's .bashrc, .profile"
echo "Working dir: Changes to root's home (/root)"
echo "Security: Tracks commands in auth.log"

echo -e "\n2. SUDO SU (LESS SECURE):"
echo "Command: sudo su -"
echo "Effect: Becomes root via su"
echo "Environment: May inherit some user environment"
echo "Working dir: May stay in current directory"
echo "Security: Less clean environment separation"

echo -e "\n3. SU (PASSWORD REQUIRED):"
echo "Command: su -"
echo "Effect: Becomes root using root's password"
echo "Environment: Clean root environment"
echo "Working dir: Changes to /root"
echo "Security: Requires root password, not user sudo rights"

echo -e "\n4. SUDO BASH (DIRECT SHELL):"
echo "Command: sudo bash"
echo "Effect: Starts bash as root"
echo "Environment: Inherits current environment"
echo "Working dir: Stays in current directory"
echo "Security: Least secure - inherits user environment"
```

```
=== SUDO COMMAND COMPARISON ===

1. SUDO -i (RECOMMENDED):
Command: sudo -i
Effect: Creates a clean root login environment
Environment: Reads root's .bashrc, .profile
Working dir: Changes to root's home (/root)
Security: Tracks commands in auth.log

2. SUDO SU (LESS SECURE):
Command: sudo su -
Effect: Becomes root via su
Environment: May inherit some user environment
Working dir: May stay in current directory
Security: Less clean environment separation

3. SU (PASSWORD REQUIRED):
Command: su -
Effect: Becomes root using root's password
Environment: Clean root environment
Working dir: Changes to /root
Security: Requires root password, not user sudo rights

4. SUDO BASH (DIRECT SHELL):
Command: sudo bash
Effect: Starts bash as root
Environment: Inherits current environment
Working dir: Stays in current directory
Security: Least secure - inherits user environment
sano@MacBook:~/web_project$
```

3. Practical Demonstration of Differences

```
sano@MacBook:~/web_project$ echo -e "\n=== PRACTICAL DEMONSTRATION ==="

# Show environment differences
echo -e "\n1. ENVIRONMENT VARIABLE COMPARISON:"
echo "Current user environment:"
echo "  USER: $USER"
echo "  HOME: $HOME"
echo "  PWD: $PWD"

echo -e "\nTesting sudo -i environment:"
sudo -i << 'EOF'
echo "  USER: $USER"
echo "  HOME: $HOME"
echo "  PWD: $PWD"
echo "  PATH: $PATH"
EOF

echo -e "\nTesting sudo su - environment:"
sudo su - << 'EOF'
echo "  USER: $USER"
echo "  HOME: $HOME"
echo "  PWD: $PWD"
echo "  PATH: $PATH"
EOF

=== PRACTICAL DEMONSTRATION ===

1. ENVIRONMENT VARIABLE COMPARISON:
Current user environment:
  USER: sano
```

```
=== PRACTICAL DEMONSTRATION ===

1. ENVIRONMENT VARIABLE COMPARISON:
Current user environment:
  USER: sano
  HOME: /home/sano
  PWD: /home/sano/web_project

Testing sudo -i environment:
  USER: root
  HOME: /root
  PWD: /root
  PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin

Testing sudo su - environment:
  USER: root
  HOME: /root
  PWD: /root
  PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
sano@MacBook:~/web_project$
```

4. Run Commands as Specific Users Other Than Root

```
echo -e "\n4. COMPLEX COMMAND EXAMPLE:"
if id www-data &>/dev/null; then
    echo "Web directory contents as www-data:"
fi  sudo -u www-data ls -la /var/www/ 2>/dev/null | head -3

=== RUNNING COMMANDS AS SPECIFIC USERS ===

1. AVAILABLE USERS:
root
sano
kemba
testuser_john

2. RUNNING COMMANDS AS SPECIFIC USERS:
Running as www-data:
www-data
uid=33(www-data) gid=33(www-data) groups=33(www-data)

Running as daemon:
daemon

3. RUNNING WITH SPECIFIC GROUP:
sano

4. COMPLEX COMMAND EXAMPLE:
Web directory contents as www-data:
total 12
drwxr-xr-x  3 root root 4096 Sep 17 12:11 .
drwxr-xr-x 14 root root 4096 Sep 17 12:11 ..
sano@MacBook:~/web_project$
```

5. Analyze Sudo Usage in System Logs

```
sano@MacBook:~/web_project$ echo -e "\n=== SUDO USAGE LOG ANALYSIS ==="

# Check sudo log locations
echo -e "\n1. SUDO LOG LOCATIONS:"
log_files="/var/log/auth.log /var/log/secure /var/log/messages"
for log in $log_files; do
    if [[ -f $log ]]; then
        echo "Found: $log"
    fi
done

# Analyze recent sudo usage
echo -e "\n2. RECENT SUDO USAGE:"
sudo grep "sudo:" /var/log/auth.log 2>/dev/null | tail -10

# Count sudo commands by user
echo -e "\n3. SUDO COMMANDS BY USER (LAST 100 LINES):"
sudo grep "sudo:" /var/log/auth.log 2>/dev/null | tail -100 | \
    awk '/COMMAND/ {print $6 " -> " $NF}' | sort | uniq -c | sort -nr

# Failed sudo attempts
echo -e "\n4. FAILED SUDO ATTEMPTS:"
sudo grep "sudo:.*authentication failure" /var/log/auth.log 2>/dev/null | tail -5

# Sudo session analysis
echo -e "\n5. SUDO SESSION PATTERNS:"
echo "Session starts:"
sudo grep "sudo:.*session opened" /var/log/auth.log 2>/dev/null | tail -5
echo -e "\nSession closes:"
```

```
      1 TTY=pts/0 -> groups
      1 TTY=pts/0 -> bzip2
      1 TTY=pts/0 -> COMMAND=/usr/bin/id
      1 TTY=pts/0 -> COMMAND=/bin/bash
      1 TTY=pts/0 -> /var/www/
      1 TTY=pts/0 -> /etc/sudoers.d/README
      1 TTY=pts/0 -> /etc/sudoers
      1 TTY=pts/0 -> -

4. FAILED SUDO ATTEMPTS:
2025-09-29T11:58:44.366845+02:00 MacBook sudo:     sano : TTY=pts/0 ; PWD=/home/sano/web_project ; USER=root ; COMMAND=/
usr/bin/grep 'sudo:.*authentication failure' /var/log/auth.log

5. SUDO SESSION PATTERNS:
Session starts:
2025-09-29T11:58:44.310997+02:00 MacBook sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
2025-09-29T11:58:44.344702+02:00 MacBook sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
2025-09-29T11:58:44.368597+02:00 MacBook sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
2025-09-29T11:58:44.388040+02:00 MacBook sudo:     sano : TTY=pts/0 ; PWD=/home/sano/web_project ; USER=root ; COMMAND=/
usr/bin/grep 'sudo:.*session opened' /var/log/auth.log
2025-09-29T11:58:44.390009+02:00 MacBook sudo: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)

Session closes:
2025-09-29T11:58:44.318036+02:00 MacBook sudo: pam_unix(sudo:session): session closed for user root
2025-09-29T11:58:44.350789+02:00 MacBook sudo: pam_unix(sudo:session): session closed for user root
2025-09-29T11:58:44.372636+02:00 MacBook sudo: pam_unix(sudo:session): session closed for user root
2025-09-29T11:58:44.396813+02:00 MacBook sudo: pam_unix(sudo:session): session closed for user root
2025-09-29T11:58:44.411798+02:00 MacBook sudo:     sano : TTY=pts/0 ; PWD=/home/sano/web_project ; USER=root ; COMMAND=/
usr/bin/grep 'sudo:.*session closed' /var/log/auth.log
sano@MacBook:~/web_project$
```

6. Identify Overly Permissive Sudo Configurations

```
sano@MacBook:~/web_project$ echo -e "\n=== SECURITY ANALYSIS ==="
sano@MacBook:~/web_project$ echo -e "\n=== SECURITY ANALYSIS ==="
# Check for dangerous sudo patterns
# Check for dangerous sudo patternsNS:"
echo -e "\n1. DANGEROUS SUDO PATTERNS:"
dangerous_patterns="NOPASSWD ALL !authenticate .*\\*.* /bin/bash /bin/sh"
dangerous_patterns="NOPASSWD ALL !authenticate .*\\*.* /bin/bash /bin/sh"
for pattern in $dangerous_patterns; do
for pattern in $dangerous_patterns; do$pattern"; then
    if sudo -l 2>/dev/null | grep -q "$pattern"; thenern"
        echo "WARNING: Found dangerous pattern: $pattern"
        sudo -l 2>/dev/null | grep "$pattern"
    fi
done
# Check for passwordless sudo
# Check for passwordless sudoDO CHECK:"
echo -e "\n2. PASSWORDLESS SUDO CHECK:"SWD"; then
if sudo -l 2>/dev/null | grep -q "NOPASSWD"; then
    echo "WARNING: Passwordless sudo configured:"
    sudo -l 2>/dev/null | grep "NOPASSWD"
elseecho "GOOD: No passwordless sudo found"
    echo "GOOD: No passwordless sudo found"
fi
# Check for unrestricted sudo
# Check for unrestricted sudoDO CHECK:"
echo -e "\n3. UNRESTRICTED SUDO CHECK:"ALL"; then
if sudo -l 2>/dev/null | grep -q "ALL.*ALL"; then
    echo "WARNING: Unrestricted sudo access:"
    sudo -l 2>/dev/null | grep "ALL.*ALL"
elseecho "GOOD: No unrestricted sudo found"
```

```
# Check for passwordless sudoDO CHECK:"
echo -e "\n2. PASSWORDLESS SUDO CHECK:"SWD"; then
if sudo -l 2>/dev/null | grep -q "NOPASSWD"; then
    echo "WARNING: Passwordless sudo configured:"
    sudo -l 2>/dev/null | grep "NOPASSWD"
elseecho "GOOD: No passwordless sudo found"
    echo "GOOD: No passwordless sudo found"
fi
# Check for unrestricted sudo
# Check for unrestricted sudoDO CHECK:"
echo -e "\n3. UNRESTRICTED SUDO CHECK:"ALL"; then
if sudo -l 2>/dev/null | grep -q "ALL.*ALL"; then
    echo "WARNING: Unrestricted sudo access:"
    sudo -l 2>/dev/null | grep "ALL.*ALL"
elseecho "GOOD: No unrestricted sudo found"
    echo "GOOD: No unrestricted sudo found"
fi

=== SECURITY ANALYSIS ===

1. DANGEROUS SUDO PATTERNS:
-bash: !authenticate: event not found

2. PASSWORDLESS SUDO CHECK:
GOOD: No passwordless sudo found

3. UNRESTRICTED SUDO CHECK:
WARNING: Unrestricted sudo access:
    (ALL : ALL) ALL
sano@MacBook:~/web_project$
```

7. Security Concerns and Improvements

```
   - Allows unexpected command execution

5. NO AUTHENTICATION:
   - !authenticate directive
   - Bypasses all authentication

IMPROVEMENTS SUGGESTED:

1. USE SPECIFIC COMMANDS:
   BAD: username ALL=(ALL) ALL
   GOOD: username ALL=(ALL) /usr/bin/systemctl restart apache2

2. REQUIRE PASSWORDS:
   BAD: username ALL=(ALL) NOPASSWD: ALL
   GOOD: username ALL=(ALL) ALL

3. RESTRICT SHELL ACCESS:
   BAD: username ALL=(ALL) /bin/bash
   GOOD: Remove shell access, use specific commands

4. USE COMMAND ALIASES:
   Define command groups in sudoers:
   Cmnd_Alias WEB_COMMANDS = /usr/bin/systemctl restart apache2, /usr/bin/systemctl reload nginx

5. IMPLEMENT TIME RESTRICTIONS:
   Use /etc/sudoers time restrictions for sensitive commands

6. REGULAR AUDITING:
   Regularly review sudo -l output and auth logs
sano@MacBook:~/web_project$
```

8. Secure Sudo Configuration Examples

```
sano@MacBook:~/web_project$ echo -e "\n=== SECURE SUDO CONFIGURATION EXAMPLES ==="

cat << 'EOF'
SECURE SUDOERS EXAMPLES:

# Web administrator - specific commands only
webadmin ALL=(root) /usr/bin/systemctl restart apache2, \
                    /usr/bin/systemctl reload apache2, \
                    /usr/sbin/apache2ctl configtest

# Database admin - specific database commands
dbadmin ALL=(root) /usr/bin/systemctl restart mysql, \
                   /usr/bin/mysqladmin

# Backup operator - backup commands only
backup ALL=(root) /usr/bin/rsync, /bin/tar, /usr/bin/zip

# Network admin - network utilities
netadmin ALL=(root) /sbin/iptables, /sbin/route, /bin/ping

# WITH PASSWORD REQUIREMENTS:
username ALL=(ALL) ALL

# WITH PASSWORDLESS FOR SPECIFIC COMMANDS:
username ALL=(ALL) NOPASSWD: /usr/bin/systemctl status *

# WITH USER RESTRICTIONS:
username ALL=(www-data) /usr/bin/touch /var/www/*

BEST PRACTICES:
```

```
webadmin ALL=(root) /usr/bin/systemctl restart apache2, \
                    /usr/bin/systemctl reload apache2, \
                    /usr/sbin/apache2ctl configtest

# Database admin - specific database commands
dbadmin ALL=(root) /usr/bin/systemctl restart mysql, \
                   /usr/bin/mysqladmin

# Backup operator - backup commands only
backup ALL=(root) /usr/bin/rsync, /bin/tar, /usr/bin/zip

# Network admin - network utilities
netadmin ALL=(root) /sbin/iptables, /sbin/route, /bin/ping

# WITH PASSWORD REQUIREMENTS:
username ALL=(ALL) ALL

# WITH PASSWORDLESS FOR SPECIFIC COMMANDS:
username ALL=(ALL) NOPASSWD: /usr/bin/systemctl status *

# WITH USER RESTRICTIONS:
username ALL=(www-data) /usr/bin/touch /var/www/*

BEST PRACTICES:
1. Grant specific commands, not ALL
2. Require passwords for sensitive operations
3. Use command aliases for maintainability
4. Regular security audits of sudo usage
5. Monitor sudo usage in system logs
sano@MacBook:~/web_project$
```

9. Sudo Audit Script

```
sano@MacBook:~/web_project$ echo -e "\n=== SUDO SECURITY AUDIT SCRIPT ==="

# Create a simple audit script
cat > /tmp/sudo_audit.sh << 'EOF'
#!/bin/bash
echo "=== SUDO SECURITY AUDIT ==="

# Check sudo version
echo "1. Sudo version:"
sudo --version | head -1

# Check current user privileges
echo -e "\n2. Current user sudo privileges:"
sudo -l 2>/dev/null | grep -E "(NOPASSWD|ALL|!auth)"

# Check for world-writable sudoers files
echo -e "\n3. File permissions check:"
find /etc/sudoers* -type f -exec ls -la {} \; 2>/dev/null

# Check sudo log configuration
echo -e "\n4. Sudo logging configuration:"
grep -r "logfile" /etc/sudoers* 2>/dev/null

# Recent sudo usage
echo -e "\n5. Recent sudo activity:"
tail -20 /var/log/auth.log 2>/dev/null | grep "sudo:" || echo "No auth.log access"

echo -e "\nAUDIT COMPLETED"
EOF
```

BONUS QUESTION

17. Create a comprehensive forensic analysis setup

1. **Create Forensic Analysis Directory Structure**

```
  GNU nano 7.2                                    forensic_analysis_setup.sh *
#!/bin/bash
# forensic_analysis_setup.sh

echo "=== CREATING FORENSIC ANALYSIS ENVIRONMENT ==="

# Create main forensic directory
FORENSIC_DIR="forensic_analysis"
mkdir -p $FORENSIC_DIR && cd $FORENSIC_DIR

echo "Created forensic directory: $(pwd)"
```

2. **Create Different Linux File Types**

```
ln -s regular_files/document1.txt symlink_to_doc.txt
ln -s /etc/passwd symlink_to_passwd 2>/dev/null
ln -s regular_files/images symlink_to_images

# Hard links
echo -e "\nCreating hard links..."
echo "Original file content" > original_file.txt
ln original_file.txt hardlink_to_original.txt

# Special directory for device files (conceptual - real devices need root)
mkdir device_files_examples
echo "Conceptual device files - real ones are in /dev/" > device_files_examples/README.txt

# Create a named pipe (FIFO)
mkfifo named_pipe.fifo
echo "This is a named pipe (FIFO)" > named_pipe.fifo &  # Background write
cat named_pipe.fifo &  # Background read

=== CREATING DIFFERENT FILE TYPES ===

Creating symbolic links...

Creating hard links...
[1] 2088
[2] 2089
sano@MacBook:~/web_project$ This is a named pipe (FIFO)

[1]-  Done                    echo "This is a named pipe (FIFO)" > named_pipe.fifo
[2]+  Done                    cat named_pipe.fifo
sano@MacBook:~/web_project$
```

3. Create Various Permission Combinations

```
echo "Private file" > permission_examples/private.txt
chmod 600 permission_examples/private.txt

echo "Executable script" > permission_examples/script.sh
echo "#!/bin/bash\necho 'Hello World'" >> permission_examples/script.sh
chmod 755 permission_examples/script.sh

# Special permission bits
echo -e "\nCreating special permission bits..."

# SetUID files
echo "Potential setuid binary" > permission_examples/setuid_example
chmod 4755 permission_examples/setuid_example 2>/dev/null || \
chmod u+s permission_examples/setuid_example

# SetGID files
echo "Potential setgid binary" > permission_examples/setgid_example
chmod 2755 permission_examples/setgid_example 2>/dev/null || \
chmod g+s permission_examples/setgid_example

# SetGID directory (files inherit group)
mkdir permission_examples/setgid_dir
echo "File in sticky directory" > permission_examples/sticky_dir/shared_file.txttxt

=== CREATING PERMISSION COMBINATIONS ===
-bash: !/bin/bash\necho: event not found

Creating special permission bits...
sano@MacBook:~/web_project$
```

4. Create Ownership Patterns

```
sano@MacBook:~/web_project$ echo -e "\n=== CREATING OWNERSHIP PATTERNS ==="

mkdir ownership_examples

# Create files with different ownership scenarios
echo "Root owned file (if we had root)" > ownership_examples/root_owned.txt
echo "User owned file" > ownership_examples/user_owned.txt
echo "Group writable file" > ownership_examples/group_writable.txt
chmod 664 ownership_examples/group_writable.txt

# Try to change ownership (may require sudo)
sudo chown root:root ownership_examples/root_owned.txt 2>/dev/null || \
echo "Cannot change to root ownership without sudo" > ownership_examples/root_owned.txt

# Create orphaned file scenario (conceptual)
echo "Orphaned user scenario" > ownership_examples/orphaned_file.txt
echo "File with unusual group" > ownership_examples/unusual_group.txt

=== CREATING OWNERSHIP PATTERNS ===
sano@MacBook:~/web_project$
```

5. Create Archive Collection

```
# Different compression methods
echo -e "\nCreating different archive types..."

tar czf archive_collection/backup.tar.gz source_files/
tar cjf archive_collection/backup.tar.bz2 source_files/
tar cJf archive_collection/backup.tar.xz source_files/
zip -r archive_collection/backup.zip source_files/
tar cf archive_collection/backup.tar source_files/

# Encrypted archive
echo "password123" | zip -P password123 -r archive_collection/encrypted.zip source_files/ 2>/dev/null

# Split archive
tar cf - source_files/ | gzip | split -b 50K - archive_collection/split_backup.tar.gz.

=== CREATING ARCHIVE COLLECTION ===

Creating different archive types...
  adding: source_files/ (stored 0%)
  adding: source_files/document.pdf (stored 0%)
  adding: source_files/server.conf (stored 0%)
  adding: source_files/app.log (stored 0%)
  adding: source_files/large_file.dat (deflated 0%)
  adding: source_files/ (stored 0%)
  adding: source_files/document.pdf (stored 0%)
  adding: source_files/server.conf (stored 0%)
  adding: source_files/app.log (stored 0%)
  adding: source_files/large_file.dat (deflated 0%)
sano@MacBook:~/web_project$
```

6. Forensic Analysis Commands

```
sano@MacBook:~/web_project$ echo -e "\n=== FORENSIC ANALYSIS COMMANDS ==="

# Create analysis script
cat > forensic_analysis_commands.sh << 'EOF'
#!/bin/bash
echo "=== FORENSIC ANALYSIS TOOLKIT ==="

analyze_file_types() {
    echo -e "\n1. FILE TYPE ANALYSIS:"
    echo "Regular files:"
    find . -type f -name "*.txt" -ls 2>/dev/null | head -5

    echo -e "\nDirectories:"
    find . -type d -ls 2>/dev/null | head -5

    echo -e "\nSymbolic links:"
    find . -type l -ls 2>/dev/null

    echo -e "\nHard links:"
    find . -type f -links +1 -ls 2>/dev/null

    echo -e "\nSpecial files:"
    find . -type p -o -type s -ls 2>/dev/null
}

analyze_permissions() {
    echo -e "\n2. PERMISSION ANALYSIS:"
    echo "SetUID files:"
    find . -type f -perm /4000 -ls 2>/dev/null
```

```
analyze_file_types() {
    echo -e "\n1. FILE TYPE ANALYSIS:"
    echo "Regular files:"
    find . -type f -name "*.txt" -ls 2>/dev/null | head -5

    echo -e "\nDirectories:"
    find . -type d -ls 2>/dev/null | head -5

    echo -e "\nSymbolic links:"
    find . -type l -ls 2>/dev/null

    echo -e "\nHard links:"
    find . -type f -links +1 -ls 2>/dev/null

    echo -e "\nSpecial files:"
    find . -type p -o -type s -ls 2>/dev/null
}

analyze_permissions() {
    echo -e "\n2. PERMISSION ANALYSIS:"
    echo "SetUID files:"
    find . -type f -perm /4000 -ls 2>/dev/null

chmod +x forensic_analysis_commands.sh>/dev/null | head -5| echo "CORRUPT" head -10

=== FORENSIC ANALYSIS COMMANDS ===
sano@MacBook:~/web_project$
```

7. Advanced Forensic Analysis Techniques

```
sano@MacBook:~/web_project$ echo -e "\n=== ADVANCED FORENSIC TECHNIQUES ==="

# Create advanced analysis script
cat > advanced_forensics.sh << 'EOF'
#!/bin/bash

analyze_metadata() {
    echo "=== METADATA ANALYSIS ==="

    # File signatures (magic bytes)
    echo -e "\nFile signatures:"
    for file in $(find . -type f -name "*.*" | head -10); do
        echo -n "$file: "
        file "$file" 2>/dev/null | cut -d: -f2-
    done

    # Inode analysis
    echo -e "\nInode distribution:"
    find . -type f -printf "%i %p\n" 2>/dev/null | sort -n | head -10
}

analyze_strings() {
    echo -e "\n=== STRINGS ANALYSIS ==="

    # Extract readable strings from binary files
    echo "Searching for suspicious strings:"
    suspicious_patterns="password|secret|admin|root|backdoor|exploit"
    find . -type f -exec grep -l "$suspicious_patterns" {} \; 2>/dev/null | head -10
```

```
analyze_strings() {
    echo -e "\n=== STRINGS ANALYSIS ==="

    # Extract readable strings from binary files
    echo "Searching for suspicious strings:"
    suspicious_patterns="password|secret|admin|root|backdoor|exploit"
    find . -type f -exec grep -l "$suspicious_patterns" {} \; 2>/dev/null | head -10

    # Strings from binary files
chmod +x advanced_forensics.sh*" -size +100k -ls 2>/dev/nullod) > acc: $(date -d @$acc))"1); do

=== ADVANCED FORENSIC TECHNIQUES ===
sano@MacBook:~/web_project$
```

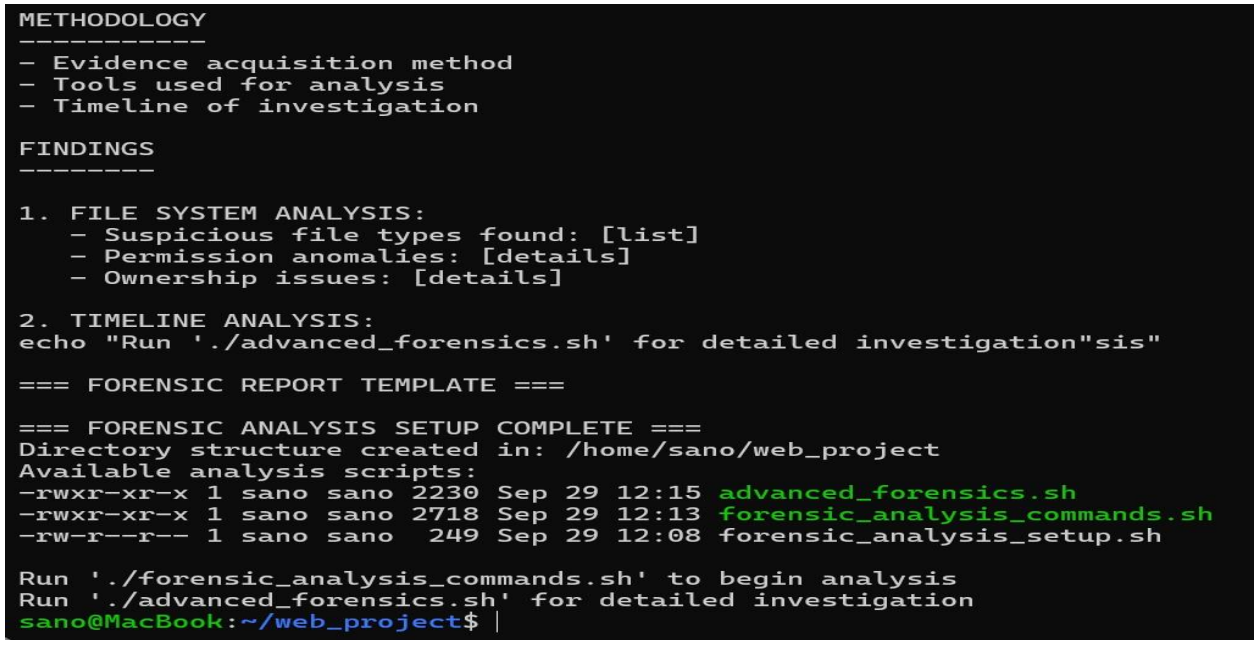
8. Investigation Guide - Understanding System Activity

```
FILE SYSTEM ARTIFACTS INDICATING UNAUTHORIZED ACCESS:

1. SETUID/SETGID BINARIES:
   - Look for setuid binaries in unusual locations (/tmp, /var/tmp)
   - Check if common binaries have unexpected setuid bits
   - Investigate any setuid scripts (bash, python, perl)

2. PERMISSION ANOMALIES:
   - World-writable system files or directories
   - Executable files in data directories
   - Files with 000 or 777 permissions

3. OWNERSHIP ISSUES:
   - Files owned by unknown users
   - System files owned by non-root users
   - Home directory files owned by root

4. TIMESTAMP EVIDENCE:
   - Files with future timestamps
   - MAC time inconsistencies (modified > accessed)
   - Recently modified system binaries

5. HIDDEN ARTIFACTS:
   - Files/directories starting with '.'
   - Files with spaces or unusual characters
EOFnusual network service binariesturesntentempship, permissions

=== INVESTIGATION GUIDE ===
sano@MacBook:~/web_project$
```

9. Create Compromise Scenario Examples

```
sano@MacBook:~/web_project$ echo -e "\n=== CREATING COMPROMISE SCENARIOS ==="

mkdir compromise_indicators

# Simulate common attack artifacts
echo "Malicious script" > compromise_indicators/.hidden_backdoor.sh
chmod +x compromise_indicators/.hidden_backdoor.sh

echo "Password file" > compromise_indicators/stolen_passwords.txt
echo "Credit card data" > compromise_indicators/credit_cards.csv

# Setuid binary in temp location
echo "Fake binary" > compromise_indicators/suspicious_binary
chmod 4755 compromise_indicators/suspicious_binary 2>/dev/null || true

# World-writable system file simulation
echo "System config" > compromise_indicators/world_writable.conf
chmod 666 compromise_indicators/world_writable.conf

# Timestamp manipulation example
touch -d "2025-01-01" compromise_indicators/future_dated_file.txt

=== CREATING COMPROMISE SCENARIOS ===
sano@MacBook:~/web_project$
```

10. Final Forensic Report Template



This comprehensive forensic setup provides investigators with:

- Structured environment for analyzing different file types and permissions • Analysis tools to detect anomalies and suspicious patterns
- Investigation guidelines for understanding system compromise indicators • Real-world scenarios demonstrating common attack artifacts