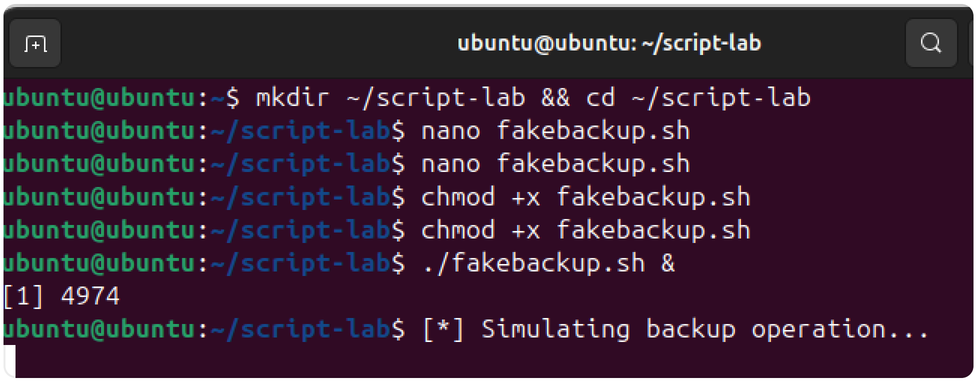
Day 7: Incident Response, Suspicious Bash Script Execution.md

Objective:
The objective of this project is to demonstrate incident response fundamentals by simulating the execution, detection, analysis, and removal of a suspicious bash script on a Linux system. The exercise aims to apply the core phases of incident response—preparation, detection and analysis, containment, eradication, recovery, and post-incident review—to a realistic but controlled scenario.

Tools Used:

- Ubuntu Linux
- Bash scripting
- ps (process monitoring)
- grep (process searching and filtering)
- shred (secure file deletion)

As part of the incident response exercise, a harmless bash script was created and executed on an Ubuntu system to simulate the behavior of a suspicious process. The intention was to mimic the activities of a potential attacker and observe how such activity could be detected and handled during an incident response scenario.

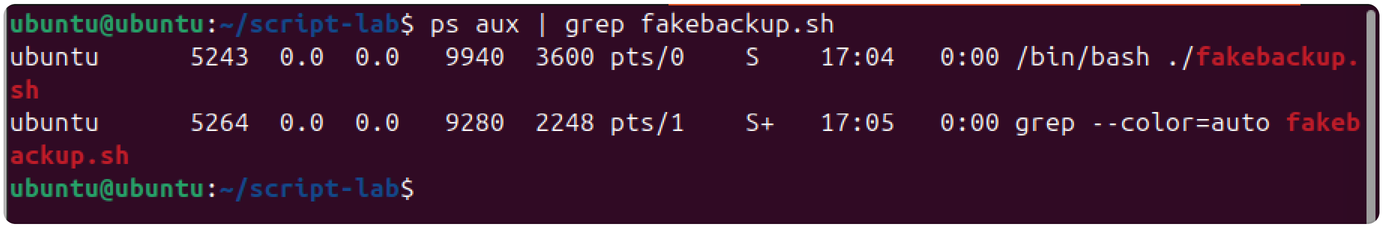


```
ubuntu@ubuntu:~$ mkdir ~/script-lab && cd ~/script-lab
ubuntu@ubuntu:~/script-lab$ nano fakebackup.sh
ubuntu@ubuntu:~/script-lab$ nano fakebackup.sh
ubuntu@ubuntu:~/script-lab$ chmod +x fakebackup.sh
ubuntu@ubuntu:~/script-lab$ chmod +x fakebackup.sh
ubuntu@ubuntu:~/script-lab$ ./fakebackup.sh &
[1] 4974
ubuntu@ubuntu:~/script-lab$ [*] Simulating backup operation...
```

Once the script was launched, it initiated a running process and generated a process ID, just as a real malicious script might. The first step in the response was detection. On the Linux endpoint, active processes were examined using the command:

ps aux | grep

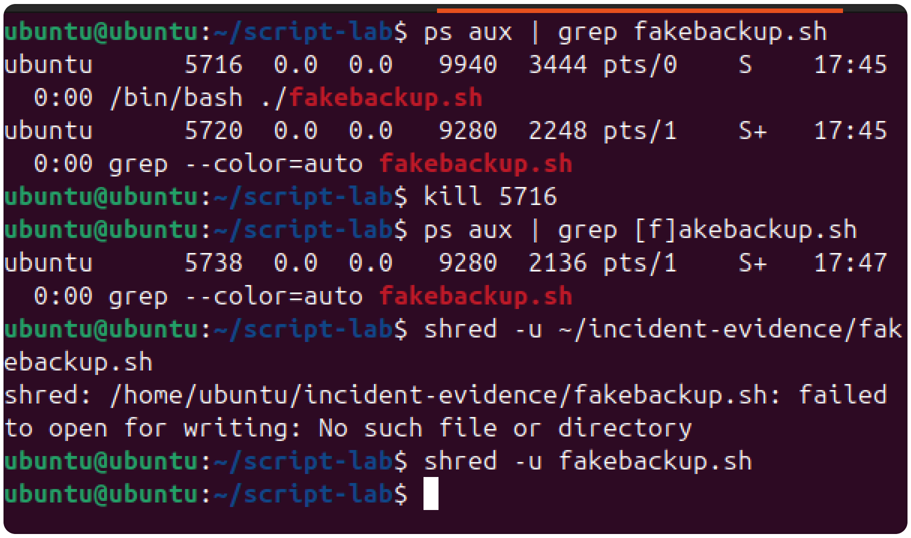
This allowed the process associated with the simulated script to be identified. Following established incident response methodology, the workflow proceeded through the key phases: preparation, detection and analysis, containment, eradication, recovery, and finally post-incident activity.



```
ubuntu@ubuntu:~/script-lab$ ps aux | grep fakebackup.sh
ubuntu      5243  0.0  0.0   9940  3600 pts/0    S    17:04   0:00 /bin/bash ./fakebackup.
sh
ubuntu      5264  0.0  0.0   9280  2248 pts/1    S+   17:05   0:00 grep --color=auto fakeb
ackup.sh
ubuntu@ubuntu:~/script-lab$
```

After confirming the suspicious process, the file location was identified and removed. As part of the eradication phase, the process was terminated, and the script file was securely deleted using:

shred -u fakebackup.sh



```
ubuntu@ubuntu:~/script-lab$ ps aux | grep fakebackup.sh
ubuntu      5716  0.0  0.0   9940  3444 pts/0    S    17:45
  0:00 /bin/bash ./fakebackup.sh
ubuntu      5720  0.0  0.0   9280  2248 pts/1    S+   17:45
  0:00 grep --color=auto fakebackup.sh
ubuntu@ubuntu:~/script-lab$ kill 5716
ubuntu@ubuntu:~/script-lab$ ps aux | grep [f]akebackup.sh
ubuntu      5738  0.0  0.0   9280  2136 pts/1    S+   17:47
  0:00 grep --color=auto fakebackup.sh
ubuntu@ubuntu:~/script-lab$ shred -u ~/incident-evidence/fak
ebackup.sh
shred: /home/ubuntu/incident-evidence/fakebackup.sh: failed
to open for writing: No such file or directory
ubuntu@ubuntu:~/script-lab$ shred -u fakebackup.sh
ubuntu@ubuntu:~/script-lab$ 
```

This ensured the file was unrecoverable, simulating the complete removal of a malicious payload. The scenario demonstrated how basic script-based intrusions on Linux systems can be detected, investigated, and resolved using standard command-line tools and structured incident response practices. n the road?