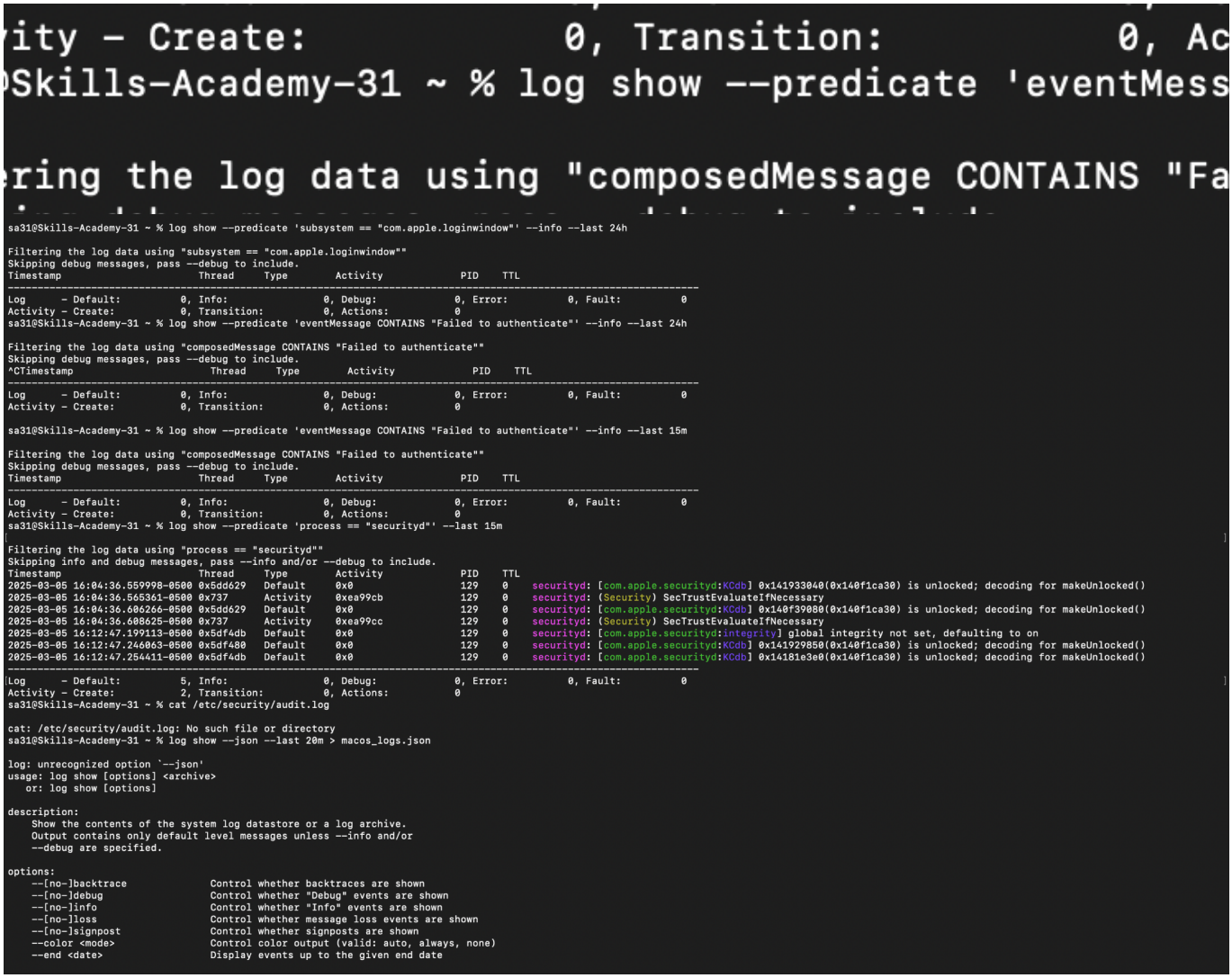
# Cybersecurity Implementation and Documentation Report

Prepared by: Hamza Fayyad

Date: April 21, 2025

## Section 1: macOS Log Collection and Analysis

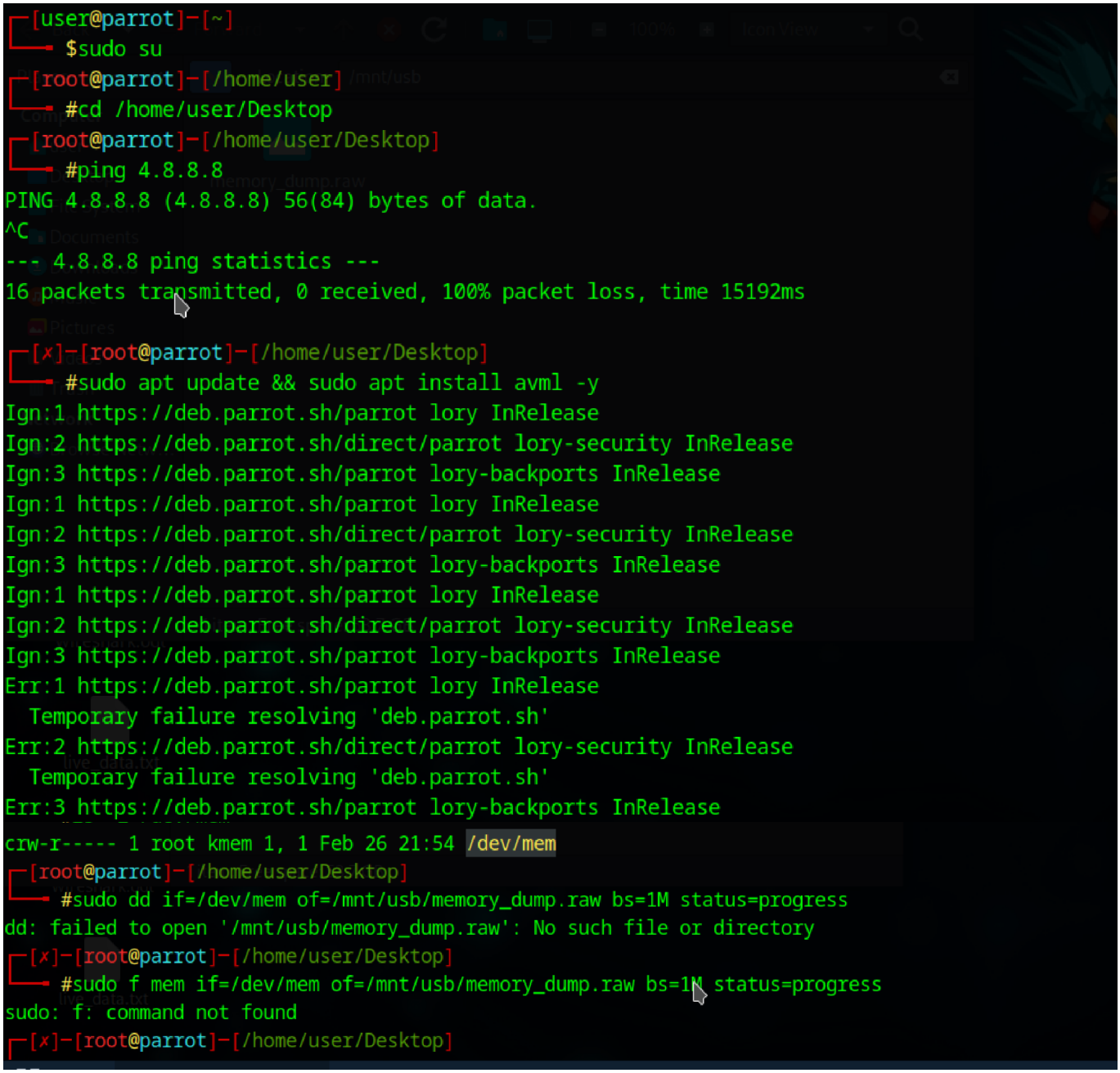
This section demonstrates macOS log filtering using the `log show` command with predicate-based queries. Multiple filters were used such as subsystem (`com.apple.loginwindow`), predicate message content (e.g., 'Failed to authenticate'), and specific process targeting (`securityd`). The purpose of this log analysis is to isolate potential authentication failures, unauthorized login attempts, and system security events. These findings can then be exported in NDJSON format to integrate with SIEM platforms like Elastic Security for visualization, correlation, and real-time alerting.

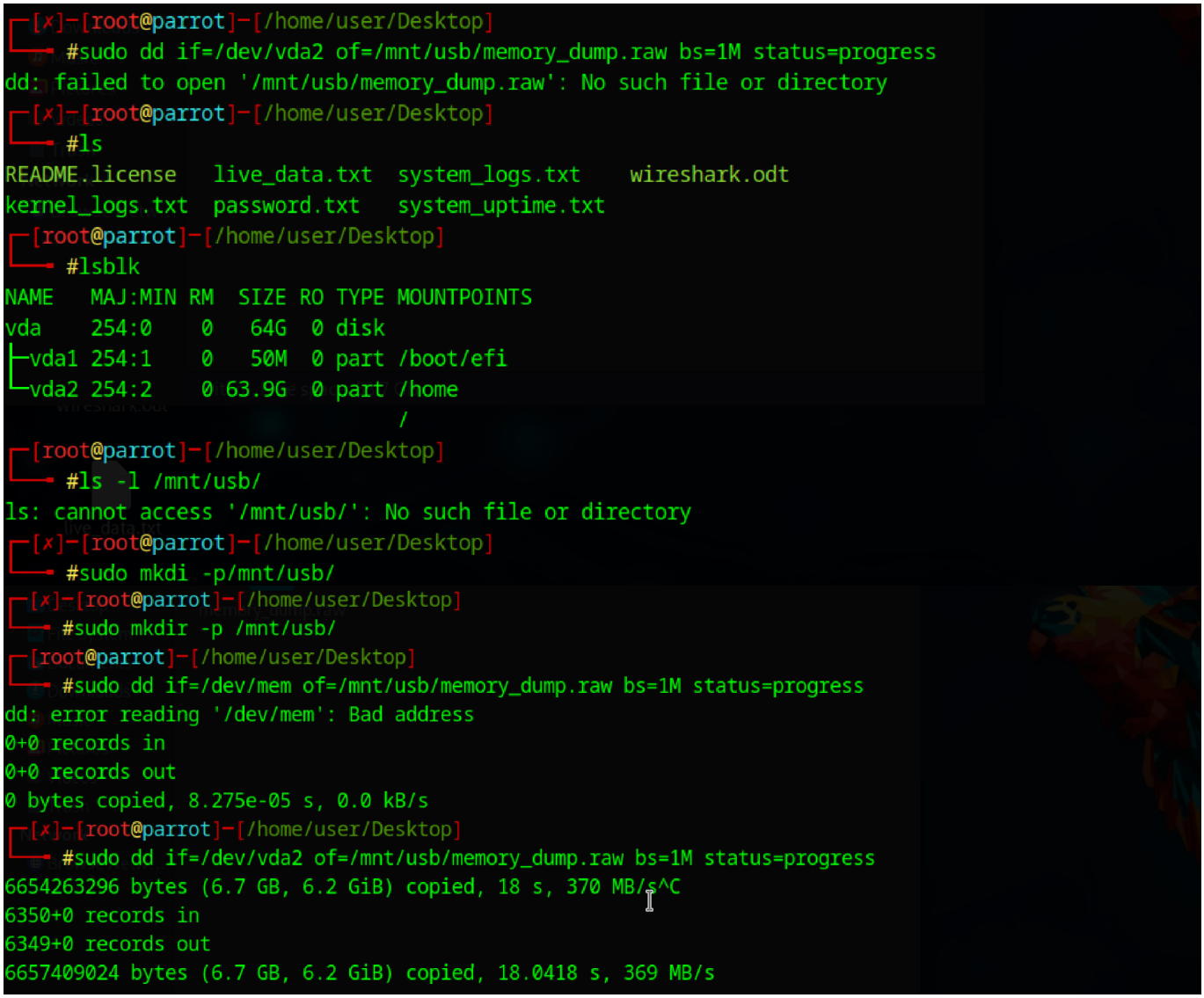


```
options:
    --[no-]backtrace              Control whether backtraces are shown
    --[no-]debug                  Control whether "Debug" events are shown
    --[no-]info                   Control whether "Info" events are shown
    --[no-]loss                   Control whether message loss events are shown
    --[no-]signpost               Control whether signposts are shown
    --color <mode>                Control color output (valid: auto, always, none)
    --end <date>                  Display events up to the given end date
    --last <num>[m|h|d]           Display recent events up to the given limit
    --[no-]pager                  Paginate output using less.
    --predicate <predicate>       Filter events using the given predicate
    --process <pid> | <process>   Filter events using the specified process
    --source                      Annotate output with source file and line-number
    --start <date>                Display events from the given start date
    --style <style>               Output format (valid: default, syslog, json, ndjson, compact)
    --timezone local | <tz>       Use the given timezone when displaying event timestamps
    --mach-continuous-time        Print mach continuous time timestamps rather than walltime
    --unreliable                  Annotate output with whether the log was emitted unreliably

valid time formats:
    'Y-M-D H:m:s+zzzz', 'Y-M-D H:m:s', 'Y-M-D', '@unixtime'

predicate usage:
    Filter predicates follow the NSPredicate format described at:
    https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/Predicates/AdditionalChapters/Introduction.html

    For predicate field/type details, see `log help predicates`.

sa31@Skills-Academy-31 ~ % log show --style ndjson --last 20m > macos_logs.ndjson

sa31@Skills-Academy-31 ~ % ls -l macos_logs.ndjson

-rw-r--r--  1 sa31  staff  174073352 Mar  5 16:17 macos_logs.ndjson
sa31@Skills-Academy-31 ~ % open ~

sa31@Skills-Academy-31 ~ %
```

## Section 2: Live Memory Forensics with Parrot OS

This section captures the process of conducting live memory forensics using Parrot OS. The initial plan involved installing AVML, a memory acquisition tool, but due to DNS and network errors, alternative methods were used. The `dd` command was attempted on `/dev/mem` and `/dev/vda2`, representing different memory or disk sources. Several troubleshooting steps were taken, including confirming the existence of the mount point `/mnt/usb`, verifying device paths using `lsblk`, and using proper syntax with `bs` block size for optimal performance. The final memory dump was successful, yielding over 6.2 GiB of data in raw format. This data can later be parsed using Volatility or Rekall for forensic investigations.




[✗]-[root@parrot]-[/home/user/Desktop]
#sudo dd if=/dev/vda2 of=/mnt/usb/memory_dump.raw bs=1M status=progress
dd: failed to open '/mnt/usb/memory_dump.raw': No such file or directory
[✗]-[root@parrot]-[/home/user/Desktop]
#ls
README.license live_data.txt system_logs.txt wireshark.odt
kernel_logs.txt password.txt system_uptime.txt
[root@parrot]-[/home/user/Desktop]
#lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
vda 254:0 0 64G 0 disk
├vda1 254:1 0 50M 0 part /boot/efi
└vda2 254:2 0 63.9G 0 part /home
/
[root@parrot]-[/home/user/Desktop]
#ls -l /mnt/usb/
ls: cannot access '/mnt/usb/': No such file or directory
[✗]-[root@parrot]-[/home/user/Desktop]
#sudo mkdi -p/mnt/usb/
[✗]-[root@parrot]-[/home/user/Desktop]
#sudo mkdir -p /mnt/usb/
[root@parrot]-[/home/user/Desktop]
#sudo dd if=/dev/mem of=/mnt/usb/memory_dump.raw bs=1M status=progress
dd: error reading '/dev/mem': Bad address
0+0 records in
0+0 records out
0 bytes copied, 8.275e-05 s, 0.0 kB/s
[✗]-[root@parrot]-[/home/user/Desktop]
#sudo dd if=/dev/vda2 of=/mnt/usb/memory_dump.raw bs=1M status=progress
6654263296 bytes (6.7 GB, 6.2 GiB) copied, 18 s, 370 MB/s^C
6350+0 records in
6349+0 records out
6657409024 bytes (6.7 GB, 6.2 GiB) copied, 18.0418 s, 369 MB/s

## Section 3: VLAN Configuration and Testing

VLAN segmentation enhances network security by logically isolating groups of devices. This section documents the configuration of VLAN 10 and VLAN 20 on a Cisco switch using CLI commands. The configuration process included creating VLANs, assigning names, configuring access ports, and verifying setup via `show vlan brief`. The ping tests demonstrate inter-device connectivity and verify the operational status of each VLAN. Implementing VLANs is an essential step in securing internal networks, limiting broadcast domains, and controlling lateral movement by potential attackers.

- **Task:** Patch management procedure.
- **Steps:** Vulnerability scanning, patch deployment, validation.

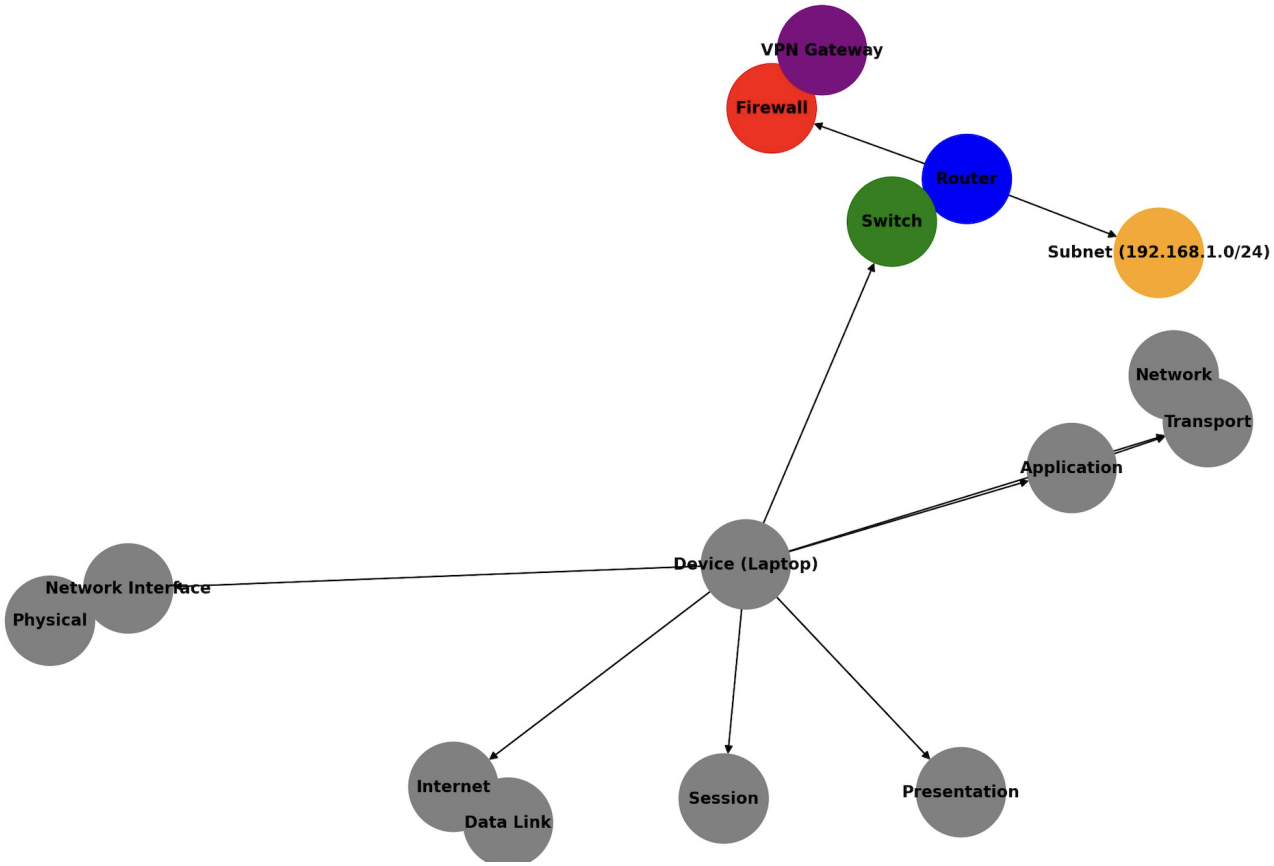
**5.3 Security Playbooks**

1. **Phishing Incident Response:** Identify, contain, remove, report.
2. **Ransomware Response:** Isolate, recover, forensic analysis.

**5.4 Knowledge Base Management**

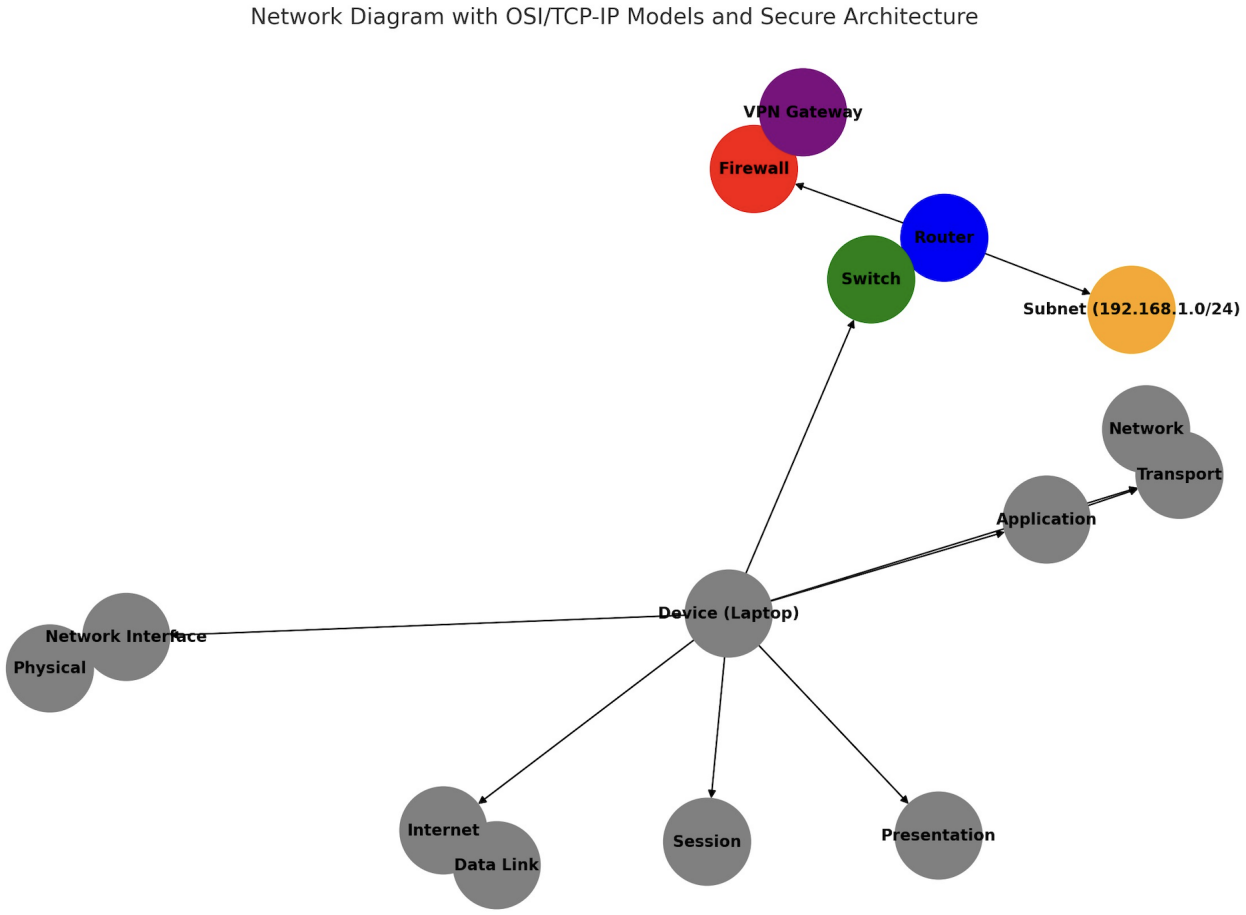
- **Categorized Resources:**
  - Incident Handling Guides
  - Threat Intelligence Reports
  - Compliance Checklists
-

Network Diagram with OSI/TCP-IP Models and Secure Architecture



## Section 4: Network Architecture

A secure network begins with a well-designed architecture. This diagram illustrates layered defenses using OSI and TCP/IP models. Devices connect to a switch, routed through a firewall and VPN gateway, before reaching segmented subnets. Each component aligns with Zero Trust principles and defense-in-depth. The visual highlights both physical layers (like Network Interface and Physical) and logical layers (such as Application, Presentation, and Transport). Designing around this model ensures resilience against compromise and enforces role-based access control.



## Section 5: Security Documentation and Improvements

This section details revisions based on feedback from a security implementation review. Key improvements include the addition of detailed incident response playbooks (for phishing and ransomware), a formalized patch management plan with validation steps, and a categorized knowledge base. Security roles within a SOC (Security Operations Center) were outlined, identifying Tier 1 and Tier 2 analysts, and the SOC Manager. A technical policy document covering Access Control, Data Protection, and System Use was developed with implementation validation using CLI operations. These updates now meet documentation standards expected in real-world enterprise environments.