# Network Defense (Fall 2024) – Weekly Portfolio

This formal portfolio presents what I learned and built week-by-week in **CSC-7303 Network Defense**.
It is written for a public audience (hiring managers, technical peers) and includes inline references to screenshots and full scripts.

> **Screenshots** live in `./screenshots/`.
> **Scripts** live in `./scripts/` and are embedded below in the **Appendix**.

## Week 1 — Lab Environment Setup & Network Fundamentals
- Built a multi-VM lab: Windows Server 2022 (future DC), Windows 11, Ubuntu 24, CentOS 9, Kali, and a pfSense/OPNsense gateway.
- Configured internal LAN (e.g., `192.168.1.0/24`) with DNS/DHCP as needed; verified host-to-host and gateway connectivity.
- Established firewall defaults: deny inbound, allow necessary outbound; confirmed traffic paths with ping and basic tools.
- **Evidence:** see `screenshots/Screenshot1_LabSetup.png`.

## Week 2 — Active Directory & Domain Join
- Promoted Windows Server 2022 to a Domain Controller (AD DS + DNS); created domain (e.g., `LAB.LOCAL`), OUs, and test users.
- Joined Windows 11 to the domain; verified domain logon and GPMC visibility.
- **Evidence:** `screenshots/Screenshot2_ADDomain.png`.

## Week 3 — Perimeter & Host Firewall Policies
- Tuned gateway firewall to block unused services and explicitly allow required ones (e.g., RDP only from management host).
- Enabled Windows Defender Firewall for all profiles; pruned unnecessary services (e.g., removed SMBv1, Telnet).
- **Evidence:** `screenshots/Screenshot3_FirewallConfig.png`.

## Week 4 — Linux Hardening (Ubuntu/CentOS)
- Applied updates (`apt`, `dnf/yum`), enabled UFW/firewalld with default deny, allowed only required services (SSH/HTTP).
- Secured SSH: no root login, prefer keys; installed Fail2Ban on Ubuntu.
- **Evidence:** `screenshots/Screenshot4_LinuxFirewall.png`.

## Week 5 — Data-at-Rest Protection
- **Windows 11:** Enabled BitLocker (with group policy to allow w/out TPM in VM lab); stored recovery key securely.
- **CentOS:** Created LUKS file-backed encrypted volume, mounted under `/mnt/encrypted` for sensitive data.
- **Evidence:** `screenshots/Screenshot5_BitLocker.png`, `screenshots/Screenshot6_LUKSsetup.png`.

## Week 6 — Vulnerability Assessment (Nessus + Nmap)

- Ran credentialed Nessus scans on Linux hosts; correlated with Nmap service/version detection.
- Findings guided remediation (e.g., SSH hardening, close 4444, patch Apache/Suricata).
- **Evidence:** `screenshots/Screenshot7_NessusScan.png`.

## Week 7 — Ubuntu Remediation via Automation
- Authored two Bash scripts to fix Nessus/Nmap findings on Ubuntu (see **Appendix**):
  - `Fix_Ubuntu_Nessus.sh`
  - `Fix_Ubuntu_Nmap.sh`
- Post-fix Nmap verified closed ports and disabled HTTP TRACE/TRACK.
- **Evidence:** `screenshots/Screenshot8_UbuntuFixScript.png`.

## Week 8 — CentOS Remediation via Automation
- Authored two Bash scripts to fix Nessus/Nmap findings on CentOS (see **Appendix**):
  - `Fix_CentOS_Nessus.sh`
  - `Fix_CentOS_Nmap.sh`
- Ensured `firewalld` running, hardened httpd headers, restricted services; re-scan confirmed minimal exposure.
- **Evidence:** `screenshots/Screenshot9_CentOSFixScript.png`.

## Week 9 — Patch & Config Management (Endpoint Central + Ansible)
- Installed ManageEngine Endpoint Central; deployed agents; executed Windows/Linux patch tasks.
- Provisioned WSL + Ubuntu; used **Ansible** to enforce security state across Linux hosts (idempotent changes).
- **Evidence:** `screenshots/Screenshot10_ManageEngine.png`, `screenshots/Screenshot11_AnsiblePlaybook.png`.

## Week 10 — OpenVAS (Greenbone) on Kali + Auto-Updates
- Added `kali_update.sh` and nightly cron for auto-updates; installed OpenVAS via script; ran authenticated network scans.
- **Evidence:** `screenshots/Screenshot12_KaliCron.png`, `screenshots/Screenshot13_OpenVAS.png`.

## Weeks 11–12 — Final Project (Capstone)
- Hardened a multi-VM scenario end-to-end; verified with Nessus/OpenVAS; documented initial vs. final state.
- Incorporated AD/GPO, host firewalls, encryption, automation, and monitoring into a cohesive defense-in-depth.

# Appendix A — Scripts (Embedded)

> Copies of all scripts used in the labs. The same files exist in `./scripts/` for execution.

## `scripts/Fix_Ubuntu_Nessus.sh`
```bash
#!/bin/bash
# Fix_Ubuntu_Nessus.sh – Script to address vulnerabilities identified by Nessus on Ubuntu
# Machine-Specific Precheck
```

```bash
EXPECTED_HOSTNAME="ubuntu-desktop"
if [[ "$(hostname)" != "$EXPECTED_HOSTNAME" ]]; then
  echo "Error: This script is intended for the system with hostname '$EXPECTED_HOSTNAME'."
  echo "Current hostname is '$(hostname)'. Exiting to prevent errors."
  exit 1
fi
echo "Starting Fix_Ubuntu_Nessus.sh on '$EXPECTED_HOSTNAME'..."
# Update Suricata
echo "Updating Suricata..."
apt-get install -y suricata || echo "Error: Suricata update failed. Check repository
configuration."
# Secure OpenSSH configuration
echo "Securing OpenSSH..."
sed -i 's/^#PermitRootLogin.*/PermitRootLogin no/' /etc/ssh/sshd_config
sed -i 's/^PasswordAuthentication.*/PasswordAuthentication no/' /etc/ssh/sshd_config
systemctl restart sshd || echo "Error: Failed to restart OpenSSH."
# Upgrade Apache HTTP Server
echo "Upgrading Apache HTTP Server..."
apt-get install -y apache2 || echo "Error: Apache update failed. Check repository
configuration."
# Remove malicious cron job
echo "Removing malicious cron jobs..."
if crontab -u root -l 2>/dev/null | grep -q 'malicious_command'; then
  crontab -u root -l | grep -v 'malicious_command' | crontab -u root
  echo "Malicious cron job removed."
else
  echo "No malicious cron job found."
fi
# Disable Netcat backdoor service
echo "Disabling Netcat backdoor service..."
if systemctl list-units --full -all | grep -q "malicious.service"; then
  systemctl stop malicious.service || echo "Error: Failed to stop malicious.service."
  systemctl disable malicious.service || echo "Error: Failed to disable malicious.service."
  rm -f /etc/systemd/system/malicious.service
  echo "Malicious service removed."
else
  echo "No malicious service found."
fi
# Final message
echo "Fix_Ubuntu_Nessus.sh completed. Review logs for any errors."
```

## `scripts/Fix_Ubuntu_Nmap.sh`
```bash
#!/bin/bash
# Fix_Ubuntu_Nmap.sh – Script to address vulnerabilities identified by Nmap on Ubuntu
# Ensure the script is run on the correct system
if [[ "$(hostname)" != "ubuntu-desktop" ]]; then
  echo "This script is intended for Ubuntu. Exiting to prevent errors."
  exit 1
fi
echo "Starting Fix_Ubuntu_Nmap.sh on 'ubuntu-desktop'..."
# Fix SSH Configuration (disable root login)
echo "Securing SSH configuration..."
if grep -q "^PermitRootLogin yes" /etc/ssh/sshd_config; then
  sed -i 's/^PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config
  echo "Root login disabled in SSH."
else
  echo "Root login already disabled."
fi
# Restart SSH service
echo "Restarting SSH service..."
systemctl restart ssh
# Restrict HTTP methods (disable TRACE/TRACK in Apache)
echo "Restricting HTTP methods..."
if [[ -f /etc/apache2/apache2.conf ]]; then
  echo "
<Directory /var/www/html>
    <IfModule mod_headers.c>
        Header always unset X-Powered-By
    </IfModule>
    RewriteEngine On
    RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
    RewriteRule .* - [F]
</Directory>
" >> /etc/apache2/apache2.conf
  echo "HTTP methods TRACE and TRACK restricted."
else
  echo "Apache configuration file not found. Skipping HTTP methods restriction."
fi
# Restart Apache service
echo "Restarting Apache service..."
systemctl restart apache2
# Close unused ports via UFW
echo "Closing unused ports..."
ufw enable
ufw deny 4444
ufw deny 3389
ufw reload
echo "Ports 4444 and 3389 closed."
echo "Fix_Ubuntu_Nmap.sh completed. Please verify changes and re-run Nmap for validation."
```

## `scripts/Fix_CentOS_Nessus.sh`
```bash
#!/bin/bash
# Fix_CentOS_Nessus.sh – Address Nessus-reported vulnerabilities on CentOS
# Machine-specific check
```

```
if [[ "$(hostname)" != "localhost.localdomain" ]]; then
   echo "This script is intended for the CentOS VM only (hostname: localhost.localdomain).
Exiting to prevent errors."
   exit 1
fi
echo "Starting Fix_CentOS_Nessus.sh on $(hostname)..."
# Function to handle errors
handle_error() {
   echo "Error encountered: $1"
   echo "Please review logs and manually address the issue."
}
# 1. Disable Netcat Backdoor Service
echo "Disabling Netcat backdoor service..."
if systemctl is-active --quiet malicious.service; then
   systemctl stop malicious.service || handle_error "Failed to stop malicious.service"
   systemctl disable malicious.service || handle_error "Failed to disable malicious.service"
   echo "Netcat backdoor service disabled."
else
   echo "No malicious Netcat service found."
fi
# 2. Correct File Permissions
echo "Correcting file permissions..."
chmod 644 /etc/passwd || handle_error "Failed to set permissions for /etc/passwd"
chmod 600 /etc/shadow || handle_error "Failed to set permissions for /etc/shadow"
echo "File permissions corrected."
# 3. Update Apache HTTP Server
echo "Updating Apache HTTP server..."
yum clean all && yum update -y httpd || handle_error "Failed to update Apache HTTP server"
echo "Apache HTTP server updated."
# 4. Update FFmpeg
echo "Updating FFmpeg..."
if rpm -q ffmpeg; then
   yum clean all && yum update -y ffmpeg || handle_error "Failed to update FFmpeg"
else
   echo "FFmpeg not installed. Skipping update."
fi
# 5. Remove Malicious Cron Jobs
echo "Removing malicious cron jobs..."
crontab -l | grep -v 'malicious_job_command' | crontab - || handle_error "Failed to remove
malicious cron jobs"
echo "Malicious cron jobs removed."
# 6. Secure SSH Configuration
echo "Securing SSH configuration..."
if grep -q "^PermitRootLogin" /etc/ssh/sshd_config; then
   sed -i 's/^PermitRootLogin.*/PermitRootLogin no/' /etc/ssh/sshd_config || handle_error
"Failed to update SSH configuration"
else
   echo "PermitRootLogin no" >> /etc/ssh/sshd_config
fi
```

```
systemctl restart sshd || handle_error "Failed to restart SSH service"
echo "SSH configuration secured."
# 7. Upgrade Suricata
echo "Updating Suricata..."
if rpm -q suricata; then
    yum clean all && yum update -y suricata || handle_error "Failed to update Suricata"
else
    echo "Suricata not installed. Skipping update."
fi
echo "Fix_CentOS_Nessus.sh completed. Please review the output and re-run Nessus for
validation."
exit 0
```

## `scripts/Fix_CentOS_Nmap.sh`
```bash
#!/bin/bash
# Fix_CentOS_Nmap.sh – Address Nmap-identified issues on CentOS
# Machine-specific check
if [[ "$(hostname)" != "localhost.localdomain" ]]; then
    echo "This script is intended for the CentOS VM only. Exiting to prevent errors."
    exit 1
fi
echo "Starting Fix_CentOS_Nmap.sh on $(hostname)..."
# Function to handle errors
handle_error() {
    echo "Error encountered: $1"
    echo "Please review logs and manually address the issue."
}
# Ensure firewalld is running
echo "Checking if firewalld is running..."
if ! systemctl is-active --quiet firewalld; then
    echo "firewalld is not running. Attempting to start it..."
    systemctl start firewalld || handle_error "Failed to start firewalld"
    systemctl enable firewalld || handle_error "Failed to enable firewalld"
fi
echo "firewalld is active."
# 1. Disable TRACE and OPTIONS HTTP methods in Apache
echo "Disabling TRACE and OPTIONS HTTP methods in Apache..."
if grep -q "TraceEnable" /etc/httpd/conf/httpd.conf; then
    sed -i 's/^TraceEnable.*/TraceEnable Off/' /etc/httpd/conf/httpd.conf
else
    echo "TraceEnable Off" >> /etc/httpd/conf/httpd.conf
fi
cat <<EOF >> /etc/httpd/conf/httpd.conf
<Directory />
    Options -Indexes
    AllowOverride None
</Directory>
```

```
EOF
systemctl restart httpd || handle_error "Failed to restart Apache"
echo "Apache HTTP server secured."
# 2. Secure SSH service
echo "Securing SSH service..."
if grep -q "^PermitRootLogin" /etc/ssh/sshd_config; then
    sed -i 's/^PermitRootLogin.*/PermitRootLogin no/' /etc/ssh/sshd_config
else
    echo "PermitRootLogin no" >> /etc/ssh/sshd_config
fi
systemctl restart sshd || handle_error "Failed to restart SSH service"
echo "SSH service secured."
# 3. Restrict access to open ports via firewalld
echo "Restricting access to open ports..."
firewall-cmd --add-service=ssh --permanent || handle_error "Failed to add SSH to firewall"
firewall-cmd --add-service=http --permanent || handle_error "Failed to add HTTP to firewall"
firewall-cmd --reload || handle_error "Failed to reload firewall rules"
echo "Access to open ports restricted."
# 4. Hide HTTP server version and other sensitive headers
echo "Hiding HTTP server version and other sensitive headers..."
if ! grep -q "ServerTokens" /etc/httpd/conf/httpd.conf; then
    echo "ServerTokens Prod" >> /etc/httpd/conf/httpd.conf
fi
if ! grep -q "ServerSignature" /etc/httpd/conf/httpd.conf; then
    echo "ServerSignature Off" >> /etc/httpd/conf/httpd.conf
fi
systemctl restart httpd || handle_error "Failed to restart Apache"
echo "HTTP server headers secured."
# 5. Verify and clean up unnecessary services
echo "Verifying services and cleaning up unnecessary ones..."
systemctl disable avahi-daemon || handle_error "Failed to disable avahi-daemon"
systemctl stop avahi-daemon || handle_error "Failed to stop avahi-daemon"
echo "Unnecessary services cleaned."
echo "Fix_CentOS_Nmap.sh completed. Please re-run Nmap to validate fixes."
```

## `scripts/kali_update.sh`
```bash
#!/bin/bash
# kali_update.sh — nightly updates for Kali
set -e
apt update
DEBIAN_FRONTEND=noninteractive apt -y upgrade
apt -y autoremove
echo "[kali_update] Completed at $(date)"
```

## `scripts/install_openvas.sh`
```bash
```

```
#!/bin/bash
# install_openvas.sh – Automates OpenVAS installation and setup on Kali
set -e
apt update
apt install -y openvas
gvm-setup       # Initialize Greenbone Vulnerability Manager (fetch feeds, etc.)
gvm-start       # Start OpenVAS services (gvmd, ospd-openvas, and gsad)
echo "OpenVAS installation and setup complete."
```

# Appendix B — Screenshots Index

> Place the following (or similarly named) files in `./screenshots/`

- Screenshot1_LabSetup.png
- Screenshot2_ADDomain.png
- Screenshot3_FirewallConfig.png
- Screenshot4_LinuxFirewall.png
- Screenshot5_BitLocker.png
- Screenshot6_LUKSsetup.png
- Screenshot7_NessusScan.png
- Screenshot8_UbuntuFixScript.png
- Screenshot9_CentOSFixScript.png
- Screenshot10_ManageEngine.png
- Screenshot11_AnsiblePlaybook.png
- Screenshot12_KaliCron.png
- Screenshot13_OpenVAS.png