Project 4: Security Hardening After-Action Report

Group 2

Member 1 Bryan Lim
Member 2 Raquel Lugo
Member 3 Grzegorz Rudnicki
Member 4 Jacob Garcia
Member 5 Casey Sharp

April 29, 2025

Contents

1 Executive Summary

To:	Chief Technology Officer (CTO)
From:	Group 2 IT Security Team
Date:	April 29, 2025
Subject:	After-Action Report on Metasploitable 2 Network Security Hardening (Project 3/4)

This report details the actions taken by Group 2 to secure the assigned network environment, focusing on the Metasploitable 2 (MS2) target server, as part of Projects 3 and 4. Following a risk assessment (implicit in the project goals of hardening a vulnerable system), a multi-faceted approach was implemented involving network segmentation, host-based hardening, and the deployment of dedicated security appliances (firewall and IDS). This report outlines the chosen network topology, the specific mitigation strategies employed (including hardening scripts), challenges encountered during deployment, and a self-evaluation of the overall effectiveness of our security posture. The primary goal was to significantly reduce the attack surface of the notoriously vulnerable MS2 system while maintaining required functionality for project testing and management access to security appliances.

2 Network Topology Design

The network topology was designed to implement a defense-in-depth strategy, isolating the vulnerable target system (MS2) and providing multiple layers of security monitoring and control. The selection of lightweight yet capable operating systems for the firewall (IPFire) and IDS (Debian) was crucial due to the specified hardware constraints (2GB RAM, 33GB HDD per VM).

2.1 Components

The network consists of the following key components:

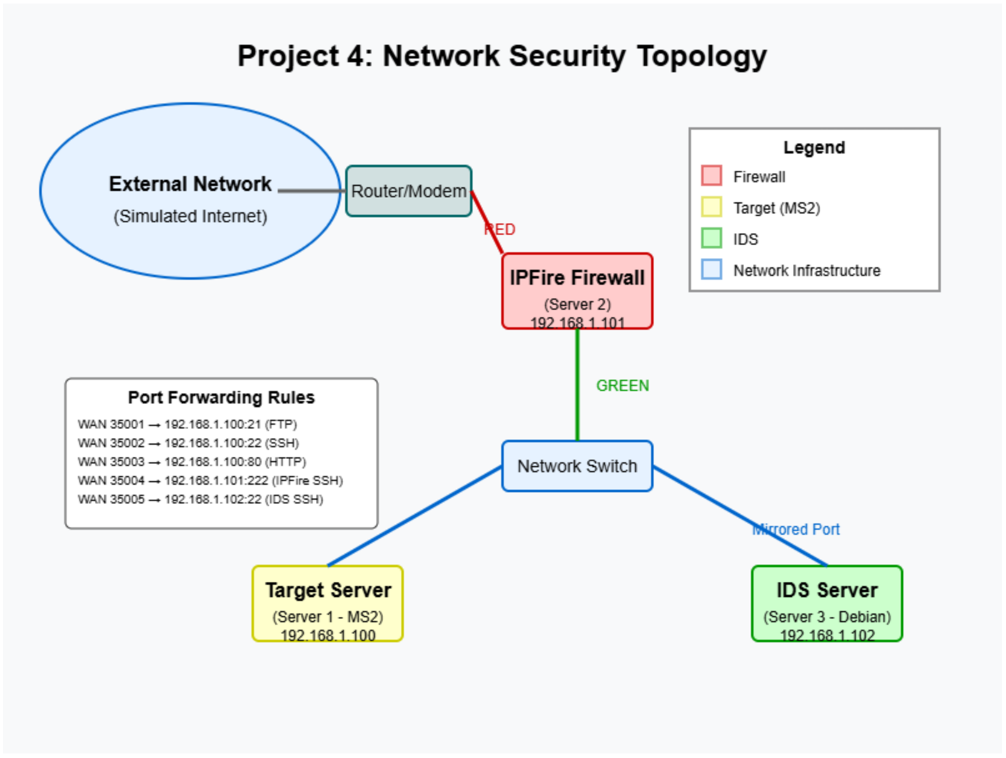
- **External Connection:** Simulated internet connection via Router/Modem.
- **Firewall Server (Server 2):** Running IPFire (‘ipfire-2.27.i586-full-core162.iso‘), acting as the primary network gateway and enforcing access control policies between the external network (RED interface) and the internal LAN (GREEN interface).
- **Internal Network Switch:** Connects internal network devices. Assumed to be configured for port mirroring (SPAN) to allow the IDS to monitor traffic destined for the target server without being inline.
- **Intrusion Detection System (IDS) Server (Server 3):** Running Debian Linux (‘debian-12.10.0-i386-netinst.iso‘) with Suricata installed.
- **Target Server (Server 1):** Metasploitable 2 (Ubuntu 8.04 Hardy Heron based), the primary focus of hardening efforts.

2.2 Logical Connections

The logical flow of traffic is depicted in Figure 1.

- The external Router/Modem connects to the Firewall's WAN (RED) interface.
- The Firewall's LAN (GREEN) interface connects to the internal Network Switch.
- The Target Server (MS2, IP: 192.168.1.100) connects to the Switch.
- The IDS Server (Debian, IP: 192.168.1.102) connects to the Switch (specifically monitoring traffic via a mirrored port).
- The Firewall itself resides on the internal network (IP: 192.168.1.101) for management purposes.

This topology ensures all traffic entering or leaving the internal network must pass through the IPFire firewall.



Port Forwarding Rules
WAN 35001 → 192.168.1.100:21 (FTP)
WAN 35002 → 192.168.1.100:22 (SSH)
WAN 35003 → 192.168.1.100:80 (HTTP)
WAN 35004 → 192.168.1.101:222 (IPFire SSH)
WAN 35005 → 192.168.1.102:22 (IDS SSH)

Figure 1: Logical Network Topology

2.3 Port Forwarding

To allow required external access for project testing while maintaining a secure posture, the following port forwarding rules were implemented on the IPFire firewall (Server 2), mapping external high ports to internal standard service ports:

- WAN 35001 → 192.168.1.100:21 (MS2 FTP)

- WAN 35002 → 192.168.1.100:22 (MS2 SSH)
- WAN 35003 → 192.168.1.100:80 (MS2 HTTP)
- WAN 35004 → 192.168.1.101:222 (IPFire SSH Management - non-standard port)
- WAN 35005 → 192.168.1.102:22 (Debian IDS SSH Management)
- WAN 35006 → 192.168.1.101:444 (IPFire Web GUI Management - HTTPS)
- WAN 35007 → 192.168.1.100:23 (MS2 Telnet)
- WAN 35008 → 192.168.1.102:5636 (IDS Alert Web Interface, e.g., EveBox - requires separate setup)
- WAN 35009 → 192.168.1.100:8180 (MS2 Tomcat HTTP)

These rules limit external exposure to only necessary services on specific high ports, obfuscating the standard service ports and providing access for testing and administration.

3 Mitigation Actions and Scripts

Hardening efforts involved manual configuration of the firewall and IDS servers, and automated scripting on the Metasploitable 2 server.

3.1 Server 1: Metasploitable 2 Hardening

An initial comprehensive hardening script (‘Hardningscript3 1.txt‘) was developed and executed. This script included steps for updating sources, installing local security tools (iptables, tcpd, fail2ban, auditd, aide, chkrootkit), configuring these tools, removing known backdoors, securing databases, managing user accounts, hardening SSH, applying initial local firewall rules, securing web applications, setting file permissions, and adding monitoring/banners.

Following initial testing and vulnerability scans after running ‘Hardningscript3 1.txt‘, it was observed that the initial local firewall rules were too permissive. To address this, a second, more focused script (‘updateandport.txt‘) was created and applied. This script specifically implemented a stricter firewall policy.

3.1.1 Initial Hardening Script (‘Hardningscript3 1.txt‘)

This script performed the broad initial hardening steps. Key actions are summarized in Section 5 of the Project 3 submission details.

Script Content (‘Hardningscript3 1.txt‘):

Initial Metasploitable 2 Hardening Script (‘Hardningscript3 1.txt‘)

```
#!/bin/bash
# Metasploitable 2 Enhanced Hardening Script (Project 3 Adaptation - v3)
# Includes checks for service status, config tests, and robust vsftpd version
    check.
```

```bash
# Preserves functionality required for Project 3 while securing critical
    vulnerabilities.
# Addresses project requirements for user accounts and network context.
# MUST BE RUN AS ROOT or using sudo!
# --- Terminal Colors ---
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[0;33m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color

# --- Configuration ---
SOURCES_FILE="/etc/apt/sources.list"
BACKUP_FILE="/etc/apt/sources.list.bak.$(date +%F-%T)"
OLD_RELEASES_URL="http://old-releases.ubuntu.com/ubuntu/"
LOG_DIR="/var/log/security_hardening"
LOG_FILE="$LOG_DIR/hardening_$(date +%Y%m%d-%H%M%S).log"
CREDENTIALS_FILE="$LOG_DIR/secure_credentials.txt"
INETD_CONF="/etc/inetd.conf"
VSFTPD_CONF="/etc/vsftpd.conf"

# --- Project Specific Variables ---
GROUP_USER=""
GROUP_PASS=""

# --- State Variables ---
MYSQL_ROOT_PW_SET_SUCCESS=0 # Flag to track if MySQL root password was likely set

# --- Print Banner ---
echo -e "${GREEN}============================================================${NC}"
echo -e "${GREEN}    METASPLOITABLE 2 ENHANCED HARDENING SCRIPT (Project 3 v3)${
    NC}"
echo -e "${GREEN}============================================================${NC}"
echo -e "${YELLOW}This script hardens security while preserving functionality${NC
    }"
echo -e "${YELLOW}required for Project 3 testing on Metasploitable 2.       ${NC}
    "
echo -e "${YELLOW}Includes checks for service status and config tests.${NC}"
echo -e "${YELLOW}REMINDER: Choose lightweight OSes for the other two servers${NC
    }"
echo -e "${YELLOW}(Firewall/IDS) due to hardware limits (P3/2GB RAM/33GB HDD).${
    NC}"
echo -e "${GREEN}============================================================${NC}"
echo ""

# --- Check for Root ---
if [ "$(id -u)" -ne 0 ]; then
  echo -e "${RED}ERROR: This script must be run as root (or using sudo).${NC}"
    >&2
  exit 1
fi

```

```
# --- Get Project User Credentials ---
echo -e "${BLUE}Please provide the credentials for the project group user.${NC}"
echo -e "${BLUE}This user will have FTP, SSH, and Telnet access to MS2.${NC}"
while [ -z "$GROUP_USER" ]; do
  read -p "Enter project username: " GROUP_USER
done
while [ -z "$GROUP_PASS" ]; do
  read -s -p "Enter project password: " GROUP_PASS
  echo "" # newline after password input
  read -s -p "Confirm project password: " GROUP_PASS_CONFIRM
  echo "" # newline
  if [ "$GROUP_PASS" != "$GROUP_PASS_CONFIRM" ]; then
    echo -e "${RED}Passwords do not match. Please try again.${NC}"
    GROUP_PASS="" # Clear password to loop again
  fi
done

# Create log directory
mkdir -p $LOG_DIR
touch $LOG_FILE
touch $CREDENTIALS_FILE
chmod 600 $CREDENTIALS_FILE

# Function to log actions
log() {
  # Log with timestamp to file and echo to stdout
  # Using current date for log timestamp from system clock
  local current_timestamp=$(date '+%Y-%m-%d %H:%M:%S') # Use system's current
    time
  echo -e "[$current_timestamp] $1" | tee -a $LOG_FILE
}

save_credential() {
  # Save credentials securely
  echo -e "$1" >> $CREDENTIALS_FILE
}

log "${GREEN}[+] Beginning security hardening for Project 3 (v3) at $(date)${NC}"
save_credential "METASPLOITABLE 2 PROJECT 3 HARDENING CREDENTIALS\n
    ==========================================="
save_credential "Created on: $(date)"
save_credential "Project Group User: $GROUP_USER"
save_credential "Project Group Pass: $GROUP_PASS"  NOTE: Storing passwords in files
    has risks. Secure this file!
save_credential "==========================================="

# --- Part 1: Update Sources ---
log "${YELLOW}[*] Step 1/9: Updating package sources...${NC}"

# --- Backup Original File ---
log "${BLUE}[-] Backing up current $SOURCES_FILE to $BACKUP_FILE...${NC}"
if [ -f "$SOURCES_FILE" ]; then
```

```
  cp -p "$SOURCES_FILE" "$BACKUP_FILE"
  if [ $? -ne 0 ]; then
    log "${RED}[!] ERROR: Failed to create backup file. Aborting.${NC}"
    exit 1
  fi
  log "${GREEN}[+] Backup created successfully.${NC}"
else
  log "${YELLOW}[!] Warning: $SOURCES_FILE not found, skipping backup.${NC}"
fi

# --- Create New sources.list Content ---
log "${BLUE}[-] Creating new $SOURCES_FILE pointing to old-releases...${NC}"
cat > "$SOURCES_FILE" << EOF
#------------------------------------------------------------------------------#
#         OFFICIAL UBUNTU REPOS (Hardy 8.04) - Project 3 Hardening             #
#     Pointed to old-releases.ubuntu.com as standard archives are offline      #
#------------------------------------------------------------------------------#

###### Ubuntu Main Repos
deb ${OLD_RELEASES_URL} hardy main restricted universe multiverse
# deb-src ${OLD_RELEASES_URL} hardy main restricted universe multiverse

###### Ubuntu Update Repos
deb ${OLD_RELEASES_URL} hardy-updates main restricted universe multiverse
# deb-src ${OLD_RELEASES_URL} hardy-updates main restricted universe multiverse

###### Ubuntu Security Repos
deb ${OLD_RELEASES_URL} hardy-security main restricted universe multiverse
# deb-src ${OLD_RELEASES_URL} hardy-security main restricted universe multiverse
EOF

log "${GREEN}[+] New sources.list created successfully.${NC}"

# --- Run apt-get update ---
log "${BLUE}[-] Running apt-get update... (This may take a while)${NC}"
apt-get update >> $LOG_FILE 2>&1
if [ $? -ne 0 ]; then
  log "${YELLOW}[!] WARNING: apt-get update finished with errors. Check $LOG_FILE
    .${NC}"
else
  log "${GREEN}[+] apt-get update completed successfully.${NC}"
fi

# --- Part 2: Install Essential Local Security Tools ---
log "${YELLOW}[*] Step 2/9: Installing minimal LOCAL security tools...${NC}"
log "${BLUE}[-] Note: The primary firewall/IDS should be on a separate server.${
    NC}"

# Minimal tools for local defense-in-depth on MS2
SECURITY_TOOLS=(
  "iptables" # For local firewall rules
  "tcpd"     # TCP Wrappers (hosts.allow/deny)
)

# Install each tool, ignoring failures if unavailable
for tool in "${SECURITY_TOOLS[@]}"; do
  log "${BLUE}[-] Installing $tool on MS2...${NC}"
  apt-get install -y $tool >> $LOG_FILE 2>&1
  if [ $? -eq 0 ]; then
    log "${GREEN}[+] Successfully installed $tool${NC}"
  else
    log "${YELLOW}[!] Failed to install $tool - may not be available or needed.${NC}"
  fi
done

log "${GREEN}[+] Basic local security tools installation completed${NC}"

# --- Part 3: Remove Known Backdoors ---
# (v3: Includes robust vsftpd check)
log "${YELLOW}[*] Step 3/9: Removing known backdoors (while preserving service function)...${NC}"

# Check for and remove ingreslock backdoor (bindshell on port 1524)
# Note: We keep port 1524 open as it's a 'legit' service in MS2 context, but remove the shell.
if grep -q "ingreslock.*shell" $INETD_CONF 2>/dev/null; then
  log "${RED}[!] Found ingreslock backdoor SHELL in $INETD_CONF${NC}"
  cp $INETD_CONF $INETD_CONF.bak.ingreslock
  # Comment out the line instead of deleting, easier to revert if needed
  sed -i '/ingreslock.*shell/s/^/# DISABLED_BY_HARDENING: /' $INETD_CONF
  # Ensure inetd is instructed to re-read config
  pkill -HUP inetd >> $LOG_FILE 2>&1
  log "${GREEN}[+] Disabled ingreslock backdoor shell in $INETD_CONF.${NC}"
  log "${YELLOW}[!] Port 1524 (ingreslock service) remains open for testing.${NC}"
else
  log "${GREEN}[+] No ingreslock backdoor shell found in $INETD_CONF.${NC}"
fi

# Check for vsftpd backdoor (version 2.3.4) - Block port 6200
VSFTPD_IS_VULN=0 # Flag
if [ -f "/usr/sbin/vsftpd" ]; then
  log "${BLUE}[-] Checking vsftpd version...${NC}"
  # Method 1: Direct version output
  VSFTPD_VERSION_OUTPUT=$(/usr/sbin/vsftpd -version 2>&1)
  log "[INFO] vsftpd -version output: $VSFTPD_VERSION_OUTPUT"
  if echo "$VSFTPD_VERSION_OUTPUT" | grep -q "2.3.4"; then
    log "${RED}[!] Found vulnerable vsftpd version 2.3.4 via direct check.${NC}"
    VSFTPD_IS_VULN=1
  else
    # Method 2: Fallback using dpkg package info if direct check inconclusive
    log "[INFO] Direct vsftpd version check inconclusive or empty. Trying dpkg..."
```

```bash
    DPKG_VSFTPD_VERSION=$(dpkg -s vsftpd 2>/dev/null | grep '^Version:')
    log "[INFO] dpkg vsftpd version string: $DPKG_VSFTPD_VERSION"
    if echo "$DPKG_VSFTPD_VERSION" | grep -q "2.3.4"; then
        log "${RED}[!] Found vulnerable vsftpd version 2.3.4 via dpkg check.${NC}"
        VSFTPD_IS_VULN=1
    else
        log "${GREEN}[+] vsftpd version does not appear to be the backdoored 2.3.4 based on available checks.${NC}"
    fi
  fi

  # Actions if vulnerable version detected
  if [ "$VSFTPD_IS_VULN" -eq 1 ]; then
    # Kill any existing backdoor processes listening on 6200
    if netstat -tulnp | grep -q ":6200 "; then
      log "${RED}[!] Found vsftpd backdoor running on port 6200! Killing...${NC}"
      pkill -f "vsftpd.*:)" >> $LOG_FILE 2>&1
      log "${GREEN}[+] Attempted to kill vsftpd backdoor process.${NC}"
    fi
  fi
  # Always add firewall block rule as defense-in-depth regardless of version detection success
  log "${YELLOW}[!] Ensuring firewall rule BLOCKS vsftpd backdoor port 6200 locally.${NC}"
else
  log "${YELLOW}[!] vsftpd binary not found, skipping check.${NC}"
fi


# Check for UnrealIRCd backdoor
if [ -d "/usr/local/unrealircd" ] || [ -d "/etc/unrealircd" ] || command -v unrealircd >/dev/null 2>&1; then
  log "${YELLOW}[!] Found UnrealIRCd installation - checking for backdoor.${NC}"
  # Check common locations for the backdoor signature
  if grep -r --include="*.c" --include="*.h" "DEBUG3_DOLOG_SYSTEM" /usr/local/unrealircd /etc/unrealircd /usr/src/unrealircd* 2>/dev/null; then
    log "${RED}[!] Found UnrealIRCd backdoor signature! Disabling service...${NC}"
    pkill -f unrealircd >> $LOG_FILE 2>&1
    # Find and remove potential startup scripts
    find /etc/init.d -name "*unreal*" -exec update-rc.d -f {} remove \; >> $LOG_FILE 2>&1
    rm -f /etc/rc*.d/*unreal* >> $LOG_FILE 2>&1
    log "${GREEN}[+] Disabled UnrealIRCd service and removed potential startup links.${NC}"
    log "${YELLOW}[!] Will add firewall rule to BLOCK UnrealIRCd ports (e.g., 6667, 6697) locally.${NC}"
  else
    log "${GREEN}[+] No obvious UnrealIRCd backdoor signature found. Service left running if active.${NC}"
    log "${YELLOW}[!] Consider blocking IRC ports (6667, 6697) in the main
    firewall.${NC}"
  fi
else
  log "${GREEN}[+] UnrealIRCd not found, skipping check.${NC}"
fi

# --- Part 4: Secure Database Services ---
log "${YELLOW}[*] Step 4/9: Securing database services (MySQL, PostgreSQL)...${NC
    }"

# MySQL hardening
MYSQL_ROOT_PASSWORD="" # Initialize variable
if command -v mysql &> /dev/null; then
  log "${BLUE}[-] Securing MySQL...${NC}"

  # Generate a strong root password
  MYSQL_ROOT_PASSWORD=$(openssl rand -base64 12 | tr -dc 'a-zA-Z0-9') # Ensure
    compatible characters
  save_credential "MySQL root password (generated): $MYSQL_ROOT_PASSWORD"

  # Check if MySQL is running
  if ! service mysql status &> /dev/null; then
      log "${YELLOW}[!] MySQL service not running. Attempting to start...${NC}"
      service mysql start >> $LOG_FILE 2>&1
      sleep 3 # Give service time to start
      if ! service mysql status &> /dev/null; then
          log "${RED}[!] Failed to start MySQL service. Cannot secure MySQL
    automatically.${NC}"
      else
          log "${GREEN}[+] MySQL service started successfully.${NC}"
      fi
  else
      log "${GREEN}[+] MySQL service is already running.${NC}"
  fi

  # Attempt to set password only if MySQL is running
  if service mysql status &> /dev/null; then
    log "${BLUE}[-] Attempting to set MySQL root password (assuming default blank
     password)...${NC}"
    mysqladmin -u root password "$MYSQL_ROOT_PASSWORD" >> $LOG_FILE 2>&1

    if [ $? -eq 0 ]; then
        log "${GREEN}[+] MySQL root password set successfully.${NC}"
        MYSQL_ROOT_PW_SET_SUCCESS=1 # Set flag indicating success
    else
        log "${RED}[!] FAILED to automatically set MySQL root password.${NC}"
        log "${YELLOW}[!] This might be because it was already set, or another
    issue occurred.${NC}"
        log "${YELLOW}[!] Manual intervention required to set/verify MySQL root
    password.${NC}"
        MYSQL_ROOT_PW_SET_SUCCESS=0
    fi

    # Run MySQL secure installation commands manually if mysql client exists AND password was set
    if [ -f "/usr/bin/mysql" ]; then
      if [ "$MYSQL_ROOT_PW_SET_SUCCESS" -eq 1 ]; then
          log "${BLUE}[-] Running MySQL security commands (removing anonymous users, remote root)...${NC}"
          # Commands to secure MySQL - use the generated password
          mysql -u root --password="$MYSQL_ROOT_PASSWORD" << EOF
DELETE FROM mysql.user WHERE User='';
FLUSH PRIVILEGES;
DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1', '::1');
FLUSH PRIVILEGES;
-- Optionally remove test database
-- DROP DATABASE IF EXISTS test;
-- DELETE FROM mysql.db WHERE Db='test' OR Db='test\\_%';
-- FLUSH PRIVILEGES;
EOF
          if [ $? -eq 0 ]; then
              log "${GREEN}[+] MySQL security commands executed successfully.${NC}"
          else
              log "${RED}[!] Failed to execute MySQL security commands. Check log and credentials.${NC}"
          fi
      else
        log "${YELLOW}[!] Skipping MySQL security commands because root password was not set automatically.${NC}"
      fi
    else
      log "${YELLOW}[!] MySQL client not found, cannot run security commands.${NC}"
    fi
    # Restart MySQL service after changes (only if password was likely set)
    if [ "$MYSQL_ROOT_PW_SET_SUCCESS" -eq 1 ]; then
        log "${BLUE}[-] Restarting MySQL service...${NC}"
        service mysql restart >> $LOG_FILE 2>&1
    fi
  else
    log "${RED}[!] MySQL service is not running after start attempt. Skipping MySQL hardening.${NC}"
  fi
else
  log "${GREEN}[+] MySQL not found, skipping MySQL hardening.${NC}"
fi

# PostgreSQL hardening (No changes needed based on previous output)
if command -v psql &> /dev/null; then
  log "${BLUE}[-] Securing PostgreSQL...${NC}"

  # Generate a strong postgres password
  PG_PASSWORD=$(openssl rand -base64 12 | tr -dc 'a-zA-Z0-9')
  save_credential "PostgreSQL 'postgres' user password (generated): $PG_PASSWORD"

  # Find PostgreSQL main config directory (might vary slightly)
  PG_CONF_DIR=$(find /etc/postgresql/ -mindepth 1 -maxdepth 1 -type d 2>/dev/null
     | head -n 1)

  if [ -n "$PG_CONF_DIR" ] && [ -d "$PG_CONF_DIR/main" ]; then
    PG_HBA_CONF="$PG_CONF_DIR/main/pg_hba.conf"
    PG_CONF="$PG_CONF_DIR/main/postgresql.conf"
    log "${BLUE}[-] Found PostgreSQL config directory: $PG_CONF_DIR/main${NC}"

    if [ -f "$PG_HBA_CONF" ]; then
      # Backup configs
      cp "$PG_HBA_CONF" "$PG_HBA_CONF.bak.$(date +%F-%T)"
      cp "$PG_CONF" "$PG_CONF.bak.$(date +%F-%T)"

      # Attempt to update postgres password using ALTER USER
      log "${BLUE}[-] Attempting to set PostgreSQL 'postgres' user password...${
    NC}"
      if su - postgres -c "psql -c \"ALTER USER postgres WITH PASSWORD '
    $PG_PASSWORD';\"" >> $LOG_FILE 2>&1; then
        log "${GREEN}[+] PostgreSQL 'postgres' user password updated successfully
    .${NC}"
        log "${YELLOW}[!] Recommended: Review $PG_HBA_CONF and change 'trust' to
    'md5' for enhanced security.${NC}"
      else
        log "${RED}[!] Failed to set PostgreSQL 'postgres' user password. Service
     might not be running or user setup is different.${NC}"
      fi

      # Restart PostgreSQL for changes to take effect
      log "${BLUE}[-] Restarting PostgreSQL service...${NC}"
      service postgresql restart >> $LOG_FILE 2>&1 || service postgresql-8.3
    restart >> $LOG_FILE 2>&1 # Try versioned service name
    else
      log "${RED}[!] PostgreSQL config file $PG_HBA_CONF not found.${NC}"
    fi
  else
    log "${RED}[!] PostgreSQL config directory not found or structured
    differently.${NC}"
  fi
else
  log "${GREEN}[+] PostgreSQL not found, skipping PostgreSQL hardening.${NC}"
fi


# --- Part 5: Secure User Accounts & Access ---
# (No changes needed based on previous output, checks seemed sufficient)
log "${YELLOW}[*] Step 5/9: Securing User Accounts & Access (SSH, FTP, Telnet)...
    ${NC}"

```

```bash
# Create the Project Group User
log "${BLUE}[-] Creating project group user '$GROUP_USER'...${NC}"
if id "$GROUP_USER" &>/dev/null; then
  log "${YELLOW}[!] User '$GROUP_USER' already exists. Setting password.${NC}"
else
  useradd -m -s /bin/bash "$GROUP_USER"
  if [ $? -ne 0 ]; then
    log "${RED}[!] Failed to create user '$GROUP_USER'. Aborting user setup.${NC}
    "
    # Continue script, but log the failure prominently
  else
    log "${GREEN}[+] User '$GROUP_USER' created successfully.${NC}"
  fi
fi
echo "$GROUP_USER:$GROUP_PASS" | chpasswd
log "${GREEN}[+] Set password for user '$GROUP_USER'.${NC}"

# Ensure FTP access for the user
log "${BLUE}[-] Ensuring FTP access for '$GROUP_USER'...${NC}"
if [ -f "$VSFTPD_CONF" ]; then
    # Ensure local users are enabled in vsftpd
    if ! grep -q "^local_enable=YES" "$VSFTPD_CONF"; then
        log "${YELLOW}[!] 'local_enable=YES' not found in $VSFTPD_CONF. Adding it
    .${NC}"
        echo "local_enable=YES" >> "$VSFTPD_CONF"
        # Also consider anonymous_enable=NO
        sed -i 's/^anonymous_enable=.*/anonymous_enable=NO/' "$VSFTPD_CONF"
        log "${BLUE}[-] Restarting vsftpd service...${NC}"
        service vsftpd restart >> $LOG_FILE 2>&1
    fi
    # Remove user from ftpusers if present
    if [ -f "/etc/ftpusers" ]; then
        sed -i "/^${GROUP_USER}$/d" /etc/ftpusers
        log "${GREEN}[+] Ensured '$GROUP_USER' is not blocked by /etc/ftpusers.${
    NC}"
    fi
else
    log "${YELLOW}[!] $VSFTPD_CONF not found. Assuming FTP server handles user
    access correctly or is not vsftpd.${NC}"
fi

# Ensure Telnet access for the user
log "${BLUE}[-] Ensuring Telnet service is enabled (via inetd)...${NC}"
if [ -f "$INETD_CONF" ]; then
    # Check if telnet line exists and is commented out
    if grep -q "^#\s*telnet" "$INETD_CONF"; then
        log "${YELLOW}[!] Telnet service is commented out in $INETD_CONF.
    Enabling...${NC}"
        sed -i '/^#\s*telnet/s/^#\s*//' "$INETD_CONF"
        # Ensure inetd re-reads config
        pkill -HUP inetd >> $LOG_FILE 2>&1
        log "${GREEN}[+] Telnet service enabled in $INETD_CONF.${NC}"
    elif grep -q "^\s*telnet" "$INETD_CONF"; then
        log "${GREEN}[+] Telnet service appears to be enabled in $INETD_CONF.${NC}"
    else
        log "${YELLOW}[!] Could not find a telnet service line in $INETD_CONF. Manual check required.${NC}"
    fi
else
    log "${YELLOW}[!] $INETD_CONF not found. Cannot verify Telnet status automatically.${NC}"
fi


# Secure SSH Configuration and Backup Admin
if [ -f "/etc/ssh/sshd_config" ]; then
  log "${BLUE}[-] Hardening SSH configuration (/etc/ssh/sshd_config)...${NC}"
  cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak.$(date +%F-%T)

  # Apply SSH hardening rules
  sed -i 's/^#*PermitRootLogin.*/PermitRootLogin no/' /etc/ssh/sshd_config
  sed -i 's/^#*PermitEmptyPasswords.*/PermitEmptyPasswords no/' /etc/ssh/sshd_config
  sed -i 's/^#*Protocol.*/Protocol 2/' /etc/ssh/sshd_config
  # Add/Ensure Protocol 2 if not present
  if ! grep -q "^Protocol" /etc/ssh/sshd_config; then
      echo "Protocol 2" >> /etc/ssh/sshd_config
  fi
  log "${GREEN}[+] Applied basic SSH hardening rules.${NC}"

  # Secure existing msfadmin user (change password)
  if id "msfadmin" &>/dev/null; then
    log "${BLUE}[-] Changing password for default 'msfadmin' account...${NC}"
    MSFADMIN_PASSWORD="M3t@spl01t_H@rd3n3d!"  Make it different and slightly more complex
    echo "msfadmin:$MSFADMIN_PASSWORD" | chpasswd
    save_credential "Default 'msfadmin' password changed to: $MSFADMIN_PASSWORD"
    log "${GREEN}[+] Default 'msfadmin' password updated.${NC}"
  fi

  # Create backup admin account 'secadmin'
  log "${BLUE}[-] Creating backup administrator account 'secadmin'...${NC}"
  ADMIN_PASSWORD=$(openssl rand -base64 10 | tr -dc 'a-zA-Z0-9')
  if id "secadmin" &>/dev/null; then
      log "${YELLOW}[!] Backup admin 'secadmin' already exists. Setting password.${NC}"
  else
      useradd -m -s /bin/bash secadmin
      log "${GREEN}[+] Backup admin user 'secadmin' created.${NC}"
  fi
  echo "secadmin:$ADMIN_PASSWORD" | chpasswd
  log "${GREEN}[+] Set password for 'secadmin'.${NC}"

  # Add 'secadmin' to sudoers if sudo is installed
  if [ -f "/etc/sudoers" ] && command -v sudo &>/dev/null; then
    if ! grep -q "^secadmin ALL=(ALL) ALL" /etc/sudoers; then
        cp /etc/sudoers /etc/sudoers.bak.$(date +%F-%T)
        echo "secadmin ALL=(ALL) ALL" >> /etc/sudoers
        log "${GREEN}[+] Added 'secadmin' to sudoers.${NC}"
    else
        log "${YELLOW}[!] 'secadmin' already exists in sudoers.${NC}"
    fi
  fi

  # Save credentials
  save_credential "Backup Admin user: secadmin"
  save_credential "Backup Admin password: $ADMIN_PASSWORD"

  # Restart SSH service
  log "${BLUE}[-] Restarting SSH service...${NC}"
  service ssh restart >> $LOG_FILE 2>&1
  log "${GREEN}[+] SSH service configuration secured and restarted.${NC}"
else
  log "${RED}[!] SSH config file /etc/ssh/sshd_config not found! Cannot secure
    SSH.${NC}"
fi

# --- Part 6: Secure Local Firewall Configuration (iptables on MS2) ---
# (No changes needed based on previous output)
log "${YELLOW}[*] Step 6/9: Setting up LOCAL firewall rules (iptables on MS2)...$
    {NC}"
log "${BLUE}[-] This provides defense-in-depth. The MAIN firewall is your
    separate server.${NC}"
log "${BLUE}[-] Your separate firewall manages external access and the 10
    forwarded ports.${NC}"

# Ports required by the project FOR METASPLOITABLE 2
KEEP_OPEN_LOCALLY=(
    21   # FTP (Project Mandatory)
    22   # SSH (Project Mandatory)
    23   # Telnet (Project Mandatory)
    80   # HTTP (Project Mandatory)
    # Common MS2 services (Keep accessible for internal testing/functionality)
    139  # NetBIOS Session Service (Samba)
    445  # Microsoft DS (Samba)
    1524 # Ingreslock / Bindshell port (Service needed, shell removed)
    3306 # MySQL
    5432 # PostgreSQL
    5900 # VNC (Often enabled on MS2)
    8009 # Apache Tomcat AJP Connector
    8180 # Apache Tomcat HTTP
)

# Ports to explicitly BLOCK LOCALLY on MS2 (Backdoors, high risk)
BLOCK_LOCALLY=(
    6000 # X11 - Usually not needed remotely
    6200 # vsftpd backdoor port
    6667 # UnrealIRCd default port (often backdoor target)
)

log "${BLUE}[-] Setting up LOCAL iptables rules on MS2...${NC}"

# Flush existing rules
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X

# Set default policies: ACCEPT traffic by default, then explicitly DROP bad ports
    .
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT # MS2 should likely not be forwarding
iptables -P OUTPUT ACCEPT

# Allow loopback traffic (essential)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Allow established connections
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# Explicitly ACCEPT required project ports (redundant with default ACCEPT, but
    good practice)
log "${BLUE}[-] Ensuring required project ports (${KEEP_OPEN_LOCALLY[*]}) are
    accessible locally...${NC}"
for port in "${KEEP_OPEN_LOCALLY[@]}"; do
    iptables -A INPUT -p tcp --dport $port -m state --state NEW -j ACCEPT
done

# Explicitly DROP known bad/backdoor/unnecessary ports LOCALLY
log "${BLUE}[-] Blocking known high-risk/backdoor ports (${BLOCK_LOCALLY[*]})
    locally...${NC}"
for port in "${BLOCK_LOCALLY[@]}"; do
  iptables -A INPUT -p tcp --dport $port -j DROP
  iptables -A INPUT -p udp --dport $port -j DROP # Block UDP too
  log "${GREEN}[+] Blocked port $port locally on MS2.${NC}"
done

# Add rate limiting for SSH (prevents simple brute force)
log "${BLUE}[-] Adding rate limiting for SSH (port 22) locally...${NC}"
iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --set --name
    SSH
iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --update --
    seconds 60 --hitcount 4 --name SSH -j DROP
log "${GREEN}[+] Added rate limiting for SSH locally.${NC}"

log "${GREEN}[+] Local firewall configured on MS2: Required ports open, high-risk
     ports blocked.${NC}"
log "${YELLOW}[!] Reminder: Configure your DEDICATED firewall server for primary
    network protection.${NC}"

# Create a script to restore these LOCAL rules at boot
log "${BLUE}[-] Creating script to restore local iptables rules at boot...${NC}"
cat > /etc/network/if-up.d/firewall-rules-local << EOL
#!/bin/sh
# Local Firewall rules for Metasploitable 2 (Project 3 - Defense in Depth)
# These are secondary to the main dedicated firewall server.
echo "Applying local Metasploitable 2 firewall rules..." | logger

# Flush existing rules silently
/sbin/iptables -F > /dev/null 2>&1
/sbin/iptables -X > /dev/null 2>&1
/sbin/iptables -t nat -F > /dev/null 2>&1
/sbin/iptables -t nat -X > /dev/null 2>&1
/sbin/iptables -t mangle -F > /dev/null 2>&1
/sbin/iptables -t mangle -X > /dev/null 2>&1

# Set default policies (ACCEPT locally, DROP specifics)
/sbin/iptables -P INPUT ACCEPT
/sbin/iptables -P FORWARD ACCEPT
/sbin/iptables -P OUTPUT ACCEPT

# Allow loopback & established
/sbin/iptables -A INPUT -i lo -j ACCEPT
/sbin/iptables -A OUTPUT -o lo -j ACCEPT
/sbin/iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# Explicitly ACCEPT required project ports locally
PORTS_TO_ACCEPT=(${KEEP_OPEN_LOCALLY[@]})
for port in "\${PORTS_TO_ACCEPT[@]}"; do
    /sbin/iptables -A INPUT -p tcp --dport \$port -m state --state NEW -j ACCEPT
done

# Explicitly DROP known bad ports locally
PORTS_TO_DROP=(${BLOCK_LOCALLY[@]})
for port in "\${PORTS_TO_DROP[@]}"; do
  /sbin/iptables -A INPUT -p tcp --dport \$port -j DROP
  /sbin/iptables -A INPUT -p udp --dport \$port -j DROP
done

# SSH Rate Limiting
/sbin/iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --set --
    name SSH
/sbin/iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --update
     --seconds 60 --hitcount 4 --name SSH -j DROP

echo "Local Metasploitable 2 firewall rules applied." | logger
```

```

exit 0
EOL

chmod +x /etc/network/if-up.d/firewall-rules-local
log "${GREEN}[+] Created local firewall boot script at /etc/network/if-up.d/
    firewall-rules-local${NC}"
# Attempt to install iptables-persistent if apt worked
if apt-get install -y iptables-persistent >> $LOG_FILE 2>&1; then
    log "${GREEN}[+] Installed iptables-persistent. Saving current rules...${NC}"
    iptables-save > /etc/iptables/rules.v4
    ip6tables-save > /etc/iptables/rules.v6
    log "${GREEN}[+] Saved rules using iptables-persistent.${NC}"
    # Can potentially remove the if-up.d script if persistent works reliably
    # rm /etc/network/if-up.d/firewall-rules-local
else
    log "${YELLOW}[!] Could not install iptables-persistent. Relying on if-up.d
    script.${NC}"
fi


# --- Part 7: Basic Web Application Security ---
log "${YELLOW}[*] Step 7/9: Basic web application security (Apache, DVWA)...${NC}
    "

# Basic Apache security headers with config test
if [ -d "/etc/apache2" ]; then
  log "${BLUE}[-] Applying basic Apache security measures...${NC}"

  APACHE_CONF="/etc/apache2/apache2.conf"
  SECURITY_CONF="/etc/apache2/conf-available/security.conf"  Preferred location on
    newer systems, check if exists

  TARGET_CONF=""
  if [ -f "$SECURITY_CONF" ]; then
      TARGET_CONF="$SECURITY_CONF"
      log "[INFO] Using $TARGET_CONF for Apache security settings."
  elif [ -f "$APACHE_CONF" ]; then
      TARGET_CONF="$APACHE_CONF"
      log "[INFO] Using $APACHE_CONF for Apache security settings."
  fi

  if [ -n "$TARGET_CONF" ]; then
    cp "$TARGET_CONF" "$TARGET_CONF.bak.$(date +%F-%T)"

    # Ensure basic security settings are present and correct
    if grep -q "^ServerTokens" "$TARGET_CONF"; then sed -i 's/^ServerTokens.*/
    ServerTokens Prod/' "$TARGET_CONF"; else echo "ServerTokens Prod" >> "
    $TARGET_CONF"; fi
    if grep -q "^ServerSignature" "$TARGET_CONF"; then sed -i 's/^ServerSignature
    .*/ServerSignature Off/' "$TARGET_CONF"; else echo "ServerSignature Off" >> "
    $TARGET_CONF"; fi
    if grep -q "^TraceEnable" "$TARGET_CONF"; then sed -i 's/^TraceEnable.*/TraceEnable Off/' "$TARGET_CONF"; else echo "TraceEnable Off" >> "$TARGET_CONF"; fi
    log "${GREEN}[+] Applied basic Apache security directives (ServerTokens, ServerSignature, TraceEnable).${NC}"

    # Ensure security conf is enabled if it exists
    if [ -f "$SECURITY_CONF" ] && [ -d "/etc/apache2/conf-enabled" ] && [ ! -L "/etc/apache2/conf-enabled/security.conf" ]; then
        ln -s ../conf-available/security.conf /etc/apache2/conf-enabled/security.conf 2>> $LOG_FILE
        log "[INFO] Enabled Apache security.conf."
    fi

    # Test configuration before restarting
    log "${BLUE}[-] Testing Apache configuration...${NC}"
    APACHE_CONFIG_TEST_OUTPUT=$(apache2ctl configtest 2>&1)
    APACHE_CONFIG_TEST_STATUS=$?
    if [ $APACHE_CONFIG_TEST_STATUS -eq 0 ]; then
        log "${GREEN}[+] Apache config test successful.${NC}"
        # Restart Apache
        log "${BLUE}[-] Restarting Apache service...${NC}"
        service apache2 restart >> $LOG_FILE 2>&1
        if [ $? -eq 0 ]; then
            log "${GREEN}[+] Apache restarted successfully.${NC}"
        else
            log "${RED}[!] Apache restart failed even after successful config test. Check service status and logs.${NC}"
        fi
    else
        log "${RED}[!] Apache config test failed! Apache NOT restarted.${NC}"
        log "${RED}[!] Config test output: ${APACHE_CONFIG_TEST_OUTPUT}${NC}"
        log "${YELLOW}[!] Manual intervention required. Check Apache configuration files (especially $TARGET_CONF).${NC}"
        log "${YELLOW}[!] Backup config available at $TARGET_CONF.bak.*${NC}"
    fi
  else
    log "${RED}[!] Apache config file ($APACHE_CONF or $SECURITY_CONF) not found. Skipping Apache hardening.${NC}"
  fi
else
    log "${GREEN}[+] Apache2 directory not found, skipping Apache hardening.${NC}"
fi

# Secure DVWA database password if possible
DVWA_CONFIG="/var/www/dvwa/config/config.inc.php"
if [ -f "$DVWA_CONFIG" ]; then
  log "${BLUE}[-] Securing DVWA configuration...${NC}"
  cp $DVWA_CONFIG $DVWA_CONFIG.bak.$(date +%F-%T)

  # Generate new DVWA database password
  DVWA_DB_PASS=$(openssl rand -base64 8 | tr -dc 'a-zA-Z0-9')
  log "${BLUE}[-] Generated new DVWA DB password.${NC}"

  # Update config file
  sed -i "s/^\$_DVWA\[ 'db_password' \]\s*=\s*'.*';/\$_DVWA[ 'db_password' ] = '$DVWA_DB_PASS';/" $DVWA_CONFIG
  log "${GREEN}[+] Updated DVWA password in $DVWA_CONFIG.${NC}"
  save_credential "DVWA database password (generated): $DVWA_DB_PASS"  Save cred regardless

  # Update the database user password only if MySQL is running AND root password was set
  if command -v mysql &> /dev/null && service mysql status &> /dev/null; then
      if [ "$MYSQL_ROOT_PW_SET_SUCCESS" -eq 1 ]; then
          log "${BLUE}[-] Attempting to update DVWA database user password in MySQL...${NC}"
          mysql -u root --password="$MYSQL_ROOT_PASSWORD" -e "SET PASSWORD FOR 'dvwa'@'localhost' = PASSWORD('$DVWA_DB_PASS'); FLUSH PRIVILEGES;" >> $LOG_FILE 2>&1
          MYSQL_DVWA_UPDATE_STATUS=$?
          mysql -u root --password="$MYSQL_ROOT_PASSWORD" -e "SET PASSWORD FOR 'dvwa'@'127.0.0.1' = PASSWORD('$DVWA_DB_PASS'); FLUSH PRIVILEGES;" >> $LOG_FILE 2>&1
          # Check status of the first command primarily
          if [ $MYSQL_DVWA_UPDATE_STATUS -eq 0 ]; then
              log "${GREEN}[+] Successfully updated DVWA database user password in MySQL.${NC}"
          else
              log "${RED}[!] Failed to update DVWA database user password in MySQL using root account. Check logs.${NC}"
          fi
      else
          log "${RED}[!] Cannot automatically update DVWA password in MySQL because MySQL root password was not set successfully by this script.${NC}"
          log "${YELLOW}[!] Manual Action Required: Ensure MySQL is running, set MySQL root password to '$MYSQL_ROOT_PASSWORD' (from credentials file), then run:${NC}"
          log "${YELLOW}[!]   sudo mysql -u root -p'$MYSQL_ROOT_PASSWORD' -e \"SET PASSWORD FOR 'dvwa'@'localhost' = PASSWORD('$DVWA_DB_PASS'); FLUSH PRIVILEGES;\"${NC}"
      fi
  else
    log "${RED}[!] Cannot automatically update DVWA password in MySQL because MySQL service is not running or unavailable.${NC}"
    log "${YELLOW}[!] Manual Action Required: Start MySQL, set MySQL root password to '$MYSQL_ROOT_PASSWORD' (from credentials file), then run the command above.${NC}"
  fi
  log "${GREEN}[+] Secured DVWA configuration (config file updated). Check MySQL status/logs for DB update result.${NC}"
else
    log "${GREEN}[+] DVWA config not found at $DVWA_CONFIG, skipping DVWA
    hardening.${NC}"
fi

# --- Part 8: Additional Security Measures ---
# (No changes needed based on previous output)
log "${YELLOW}[*] Step 8/9: Additional security measures...${NC}"

# Secure cron jobs permissions
log "${BLUE}[-] Securing cron configuration permissions...${NC}"
chmod -R 700 /etc/cron.d /etc/cron.hourly /etc/cron.daily /etc/cron.weekly /etc/
    cron.monthly >> $LOG_FILE 2>&1
chown root:root /etc/crontab >> $LOG_FILE 2>&1
chmod 600 /etc/crontab >> $LOG_FILE 2>&1
log "${GREEN}[+] Secured cron directories and file permissions.${NC}"

# Set secure permissions on critical system files
log "${BLUE}[-] Setting secure permissions on critical files (/etc/shadow, /etc/
    passwd)...${NC}"
chmod 640 /etc/shadow 2>/dev/null
chown root:shadow /etc/shadow 2>/dev/null # Ensure correct group ownership
chmod 644 /etc/passwd 2>/dev/null
chown root:root /etc/passwd 2>/dev/null
log "${GREEN}[+] Set secure permissions on /etc/shadow and /etc/passwd.${NC}"

# Create a simple security monitoring script
log "${BLUE}[-] Creating basic security monitoring script (/usr/local/bin/
    security-monitor.sh)...${NC}"
cat > /usr/local/bin/security-monitor.sh << 'EOL'
#!/bin/bash
# Simple security monitoring script for Metasploitable 2

LOG_FILE="/var/log/security_monitor.log"
echo "=== Security check run at $(date) ===" > $LOG_FILE
echo "--- System Info ---" >> $LOG_FILE
uname -a >> $LOG_FILE
echo "" >> $LOG_FILE

echo "--- Listening TCP/UDP Ports ---" >> $LOG_FILE
netstat -tulnp >> $LOG_FILE
echo "" >> $LOG_FILE

echo "--- Users with UID 0 ---" >> $LOG_FILE
awk -F: '($3 == "0") {print}' /etc/passwd >> $LOG_FILE
echo "" >> $LOG_FILE

echo "--- Sudoers Configuration (Non-default entries) ---" >> $LOG_FILE
grep -vE "^#|^Defaults|^$" /etc/sudoers /etc/sudoers.d/* 2>/dev/null >> $LOG_FILE
echo "" >> $LOG_FILE

echo "--- Last 10 Logins ---" >> $LOG_FILE
last -n 10 >> $LOG_FILE
echo "" >> $LOG_FILE

echo "--- Recent Failed Logins (auth.log) ---" >> $LOG_FILE
grep -i "Failed password" /var/log/auth.log | tail -n 20 >> $LOG_FILE
echo "" >> $LOG_FILE

echo "--- World-Writable Files in /etc ---" >> $LOG_FILE
find /etc -type f -perm -002 -ls >> $LOG_FILE 2>/dev/null
echo "" >> $LOG_FILE

echo "--- Processes running as root ---" >> $LOG_FILE
ps -U root -u root u | head -n 20 >> $LOG_FILE # Limit output
echo "" >> $LOG_FILE

echo "=== Security check finished at $(date) ===" >> $LOG_FILE

exit 0
EOL

chmod +x /usr/local/bin/security-monitor.sh

# Add to cron to run daily
echo "0 3 * * * root /usr/local/bin/security-monitor.sh" > /etc/cron.d/security-
    check
chmod 644 /etc/cron.d/security-check

log "${GREEN}[+] Created daily security check script and cron job.${NC}"

# Create login banner (/etc/issue, /etc/issue.net)
log "${BLUE}[-] Creating login banner...${NC}"
cat > /etc/issue << 'EOL'
************************************************************
* Metasploitable 2 - Project 3 Instance                    *
* AUTHORIZED ACCESS ONLY                                   *
* *
* All activities on this system are logged and monitored.  *
* Unauthorized access or use is strictly prohibited and    *
* may be subject to disciplinary action or legal prosecution.*
************************************************************
Ubuntu 8.04 LTS
EOL

cp /etc/issue /etc/issue.net
log "${GREEN}[+] Created login banners (/etc/issue, /etc/issue.net).${NC}"

# --- Part 9: Final Steps & Summary ---
log "${YELLOW}[*] Step 9/9: Finalizing and summarizing...${NC}"

log "${GREEN
    }======================================================================${NC}"
log "${GREEN}       METASPLOITABLE 2 ENHANCED HARDENING COMPLETED (Project 3 v3)
      ${NC}"
log "${GREEN
```

```
    }=====================================================================${NC}"
log "${YELLOW}[!] Key security improvements implemented on THIS Metasploitable 2
    VM:${NC}"
log "${YELLOW}[!] - Updated package sources, installed local tools (iptables,
    tcpd).${NC}"
log "${YELLOW}[!] - Checked/disabled known backdoors (ingreslock shell, vsftpd
    vuln check, UnrealIRCd).${NC}"
log "${YELLOW}[!] - Blocked high-risk/backdoor ports LOCALLY via iptables (${
    BLOCK_LOCALLY[*]}).${NC}"
log "${YELLOW}[!] - Ensured required project ports remain accessible LOCALLY (${
    KEEP_OPEN_LOCALLY[*]}).${NC}"
log "${YELLOW}[!] - Attempted to secure DBs (PostgreSQL PW set; MySQL PW
    attempted - check status).${NC}"
log "${YELLOW}[!] - Hardened SSH (Root login disabled, Protocol 2 enforced, rate
    limiting).${NC}"
log "${YELLOW}[!] - Created PROJECT group user '$GROUP_USER' with specified
    password for FTP/SSH/Telnet access.${NC}"
log "${YELLOW}[!] - Created backup 'secadmin' administrator account.${NC}"
log "${YELLOW}[!] - Changed default 'msfadmin' password.${NC}"
log "${YELLOW}[!] - Applied basic Apache security headers (Restart conditional on
     config test).${NC}"
log "${YELLOW}[!] - Secured DVWA config; DB password update attempted (Check
    MySQL status).${NC}"
log "${YELLOW}[!] - Added daily security monitoring script & cron job.${NC}"
log "${YELLOW}[!] - Set secure permissions on critical files & cron.${NC}"
log "${YELLOW}[!] - Added login banners.${NC}"
log "${GREEN
    }=====================================================================${NC}"
log "${YELLOW}[!] Secure credentials generated by this script are saved at:
    $CREDENTIALS_FILE${NC}"
log "${YELLOW}[!] PROTECT THIS FILE - IT CONTAINS PASSWORDS!${NC}"
log "${GREEN
    }=====================================================================${NC}"

echo ""
echo -e "${GREEN}============================================================${NC}
    "
echo -e "${GREEN}    Metasploitable 2 Hardening Completed (Project 3 v3)    ${NC}
    "
echo -e "${GREEN}============================================================${NC}
    "
echo -e "${YELLOW}IMPORTANT PROJECT 3 NOTES:${NC}"
echo -e "${YELLOW}1. Credentials saved in: ${BLUE}$CREDENTIALS_FILE${NC} ${RED}
    SECURE THIS FILE!${NC}"
echo -e "${YELLOW}2.  Project access user: ${BLUE}$GROUP_USER${NC} (FTP/SSH/
    Telnet)${NC}"
echo -e "${YELLOW}3. Backup admin user: '${BLUE}secadmin${NC}'${NC}"
echo -e "${YELLOW}4.  Local iptables rules applied; main protection is your
    dedicated firewall.${NC}"
echo -e "${YELLOW}5. ${RED}Check script log ($LOG_FILE) for specific outcomes:${
    NC}"
echo -e "${YELLOW}    - MySQL start/password setting status (Manual check/action
    may be needed).${NC}"
echo -e "${YELLOW}    - Apache restart status (Manual config check/restart may be
     needed).${NC}"
echo -e "${YELLOW}    - DVWA DB password update status (Manual update may be
    needed).${NC}"
echo -e "${YELLOW}6. Remember the 10 port forwarding limit & hardware constraints
    .${NC}"
echo -e "${YELLOW}7. Daily security check logs to /var/log/security_monitor.log${
    NC}"
echo -e "${GREEN}============================================================${NC}
    "

# End of script
exit 0
```

3.1.2 Firewall Refinement Script (‘updateandport.txt‘)

This script was created as a corrective measure after initial scans revealed that the default ACCEPT policy in the first script’s local firewall rules left too many ports open unnecessarily. The purpose of this script was to establish a more secure default DROP policy for incoming traffic and explicitly allow only the necessary ports identified during the project.

Key actions of this script:

Updates package sources and runs ‘apt-get update‘.

Flushes all existing ‘iptables‘ rules.

Sets the default policy for INPUT and FORWARD chains to DROP. OUTPUT chain policy is set to ACCEPT.

Explicitly allows traffic on the loopback interface (‘lo‘).

Allows established and related incoming connections to permit return traffic for initiated connections.

Defines a list of necessary TCP ports (21, 22, 23, 80, 139, 445, 1524, 3306, 5432, 5900, 8009, 8180) and creates specific ACCEPT rules for new connections on these ports.

Adds rate limiting for SSH (port 22) to mitigate brute-force attempts.

Includes logging functionality for script actions.

This script represents a significant tightening of the host-based firewall rules compared to the initial script’s configuration.

Script Content (‘updateandport.txt‘):

Firewall Update and Port Closure Script (‘updateandport.txt‘)

```
#!/bin/bash

# This script combines apt update and basic firewall rules (default drop)
# based on the provided files.
# --- Log Function (Optional but good practice) ---
LOG_DIR="/var/log/security_hardening"
```

```bash
LOG_FILE="$LOG_DIR/update_firewall_$(date +%Y%m%d-%H%M%S).log"

mkdir -p $LOG_DIR
touch $LOG_FILE

log() {
  local timestamp=$(date '+%Y-%m-%d %H:%M:%S')
  echo "[$timestamp] $1" | tee -a $LOG_FILE
}

# --- Check for Root ---
if [ "$(id -u)" -ne 0 ]; then
  echo "ERROR: This script must be run as root (or using sudo)." >&2
  exit 1
fi

log "[+] Starting script..."

# --- Part 1: Update Sources and Run apt-get update ---
# (Adapted from combined_hardening_script.txt)
log "[*] Step 1: Updating package sources and running apt-get update..."

SOURCES_FILE="/etc/apt/sources.list"
BACKUP_FILE="/etc/apt/sources.list.bak.$(date +%F-%T)"
OLD_RELEASES_URL="http://old-releases.ubuntu.com/ubuntu/"

log "[-] Backing up current $SOURCES_FILE to $BACKUP_FILE..."
if [ -f "$SOURCES_FILE" ]; then
  cp -p "$SOURCES_FILE" "$BACKUP_FILE"  [cite: 25]
  if [ $? -ne 0 ]; then
    log "ERROR: Failed to create backup file. Aborting."  [cite: 25]
    exit 1
  fi
  log "[+] Backup created successfully."
else
  log "Warning: $SOURCES_FILE not found, skipping backup."  [cite: 25]
fi

log "[-] Creating new $SOURCES_FILE pointing to old-releases..."  [cite: 26]
cat > "$SOURCES_FILE" << EOF  [cite: 27]
#------------------------------------------------------------------------------#
#       OFFICIAL UBUNTU REPOS (Hardy 8.04) - Pointed to old-releases         #
#------------------------------------------------------------------------------#
deb ${OLD_RELEASES_URL} hardy main restricted universe multiverse
deb ${OLD_RELEASES_URL} hardy-updates main restricted universe multiverse
deb ${OLD_RELEASES_URL} hardy-security main restricted universe multiverse
EOF

log "[+] New sources.list created successfully."
log "[-] Running apt-get update... (This may take a while)"
export DEBIAN_FRONTEND=noninteractive  [cite: 28]
apt-get update -y -q >> $LOG_FILE 2>&1  [cite: 28]
```

```
if [ $? -ne 0 ]; then
  log "WARNING: apt-get update finished with errors. Check $LOG_FILE."  [cite: 28]
else
  log "[+] apt-get update completed successfully."  [cite: 28]
fi

# --- Part 2: Apply Firewall Rules (Default Drop) ---
# (Adapted from portclosed.txt)
log "[*] Step 2: Applying firewall rules (Default DROP)..."

# Flush existing rules first
log "[-] Flushing existing firewall rules..."  [cite: 2]
iptables -F  [cite: 2]
iptables -X  [cite: 2]
iptables -t nat -F  [cite: 2]
iptables -t nat -X  [cite: 2]
iptables -t mangle -F  [cite: 2]
iptables -t mangle -X  [cite: 2]

# Set default policies: DROP incoming/forwarding, Allow outgoing
log "[-] Setting default policies to DROP (Input/Forward)..."  [cite: 2]
iptables -P INPUT DROP  [cite: 2]
iptables -P FORWARD DROP  [cite: 2]
iptables -P OUTPUT ACCEPT  [cite: 2]

# Allow loopback traffic (essential)
log "[-] Allowing loopback traffic..."  [cite: 2]
iptables -A INPUT -i lo -j ACCEPT  [cite: 2]
iptables -A OUTPUT -o lo -j ACCEPT  [cite: 2]

# Allow established and related connections (allows return traffic)
log "[-] Allowing established/related connections..."  [cite: 2]
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT  [cite: 2]

# Define ports to keep open (Customize this list as needed)
# Using the list from portclosed.txt as an example
ALLOWED_TCP_PORTS=(  [cite: 4]
    21   # FTP
    22   # SSH
    23   # Telnet
    80   # HTTP
    139  # NetBIOS
    445  # Microsoft-DS
    1524 # Ingreslock/Bindshell port
    3306 # MySQL
    5432 # PostgreSQL
    5900 # VNC
    8009 # AJP13
    8180 # Tomcat HTTP
)

log "[-] Allowing specified necessary TCP ports..."  [cite: 5]
```

```
for port in "${ALLOWED_TCP_PORTS[@]}"; do  [cite: 5]
    log "     - Allowing TCP port $port"
    iptables -A INPUT -p tcp --dport "$port" -m state --state NEW -j ACCEPT
    [cite: 5]
done

# Add rate limiting for SSH (optional but recommended)
log "[-] Adding rate limiting for SSH port 22..."  [cite: 6]
if ! iptables -C INPUT -p tcp --dport 22 -m state --state NEW -m recent --set --
    name SSH > /dev/null 2>&1; then  [cite:
    6]
    iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --set --
    name SSH  [cite:
    7]
    iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --update
    --seconds 60 --hitcount 4 --name SSH -j DROP  [cite:
    7]
else
    log "     SSH rate limiting rules likely already exist."  [cite: 8]
fi

log "[+] Firewall rules applied. All incoming traffic dropped by default except
    listed ports."  [cite:
    9]
log "[+] Ports allowed: ${ALLOWED_TCP_PORTS[*]}"  [cite: 9]

log "[+] Script finished."
echo ""
echo "Script finished. Check log file for details: $LOG_FILE"
echo "Run 'sudo iptables -L -n -v' to see current rules."  [cite: 10]
echo "Remember to make rules persistent (e.g., using iptables-persistent) if
    needed."  [cite:
    11]

exit 0
```

3.1.3 Hardening Script Execution Log and Analysis (‘Hardningscript3 1.txt‘)

The following log excerpt shows the output generated when the primary hardening script (‘Hardningscript3 1.txt‘, identified as v7 in the script’s internal logging) was executed on the Metasploitable 2 virtual machine. This provides direct insight into the automated hardening process and its outcomes.

MS2 Hardening Script Log Excerpt (‘Hardningscript3 v7’)

```
[2025-04-27 11:48:49] [+] Beginning security hardening for Project 3
(v7) at Sun Apr 27 11:48:49 EDT 2025 [2025-04-27 11:48:49] [*] Step 1/10:
Updating package sources... [2025-04-27 11:48:49] [-] Backing up current
/etc/apt/sources.list to /etc/apt/sources.list.bak.2025-04-27-11:48:49... [2025-
04-27 11:48:49] [+] Backup created successfully. [2025-04-27 11:48:49] [-]
Creating new /etc/apt/sources.list pointing to old-releases... [2025-04-27
11:48:49] [+] New sources.list created successfully. [2025-04-27 11:48:49]
[-] Running apt-get update... (This may take a while) ... (apt-get update
output omitted for brevity) ... Fetched 7203kB in 32s (225kB/s) Reading package
lists... [2025-04-27 11:49:24] [+] apt-get update completed successfully.
[2025-04-27 11:49:24] [*] Step 2/10: Installing minimal LOCAL security tools...
[2025-04-27 11:49:24] [-] Note: The primary firewall/IDS should be on a
separate server. [2025-04-27 11:49:24] [-] Installing iptables on MS2... [2025-
04-27 11:49:24] [+] Successfully installed iptables [2025-04-27 11:49:24] [-]
Installing tcpd on MS2... [2025-04-27 11:49:25] [+] Successfully installed
tcpd [2025-04-27 11:49:25] [-] Installing fail2ban on MS2... ... (fail2ban
install output omitted) ... Setting up fail2ban (0.8.2-2ubuntu0.1) ... [2025-
04-27 11:49:33] [+] Successfully installed fail2ban [2025-04-27 11:49:33] [-]
Installing auditd on MS2... ... (auditd install output omitted) ... Setting up
auditd (1.6.5-0ubuntu3) ... [2025-04-27 11:49:40] [+] Successfully installed
auditd [2025-04-27 11:49:40] [-] Installing aide on MS2... ... (aide install
output omitted) ... Setting up aide-common (0.13.1-8ubuntu3) ... [2025-04-27
11:49:58] [+] Successfully installed aide [2025-04-27 11:49:58] [-] Installing
chkrootkit on MS2... ... (chkrootkit install output omitted) ... Setting up
chkrootkit (0.47-1.1ubuntu0.3) ... [2025-04-27 11:50:04] [+] Successfully
installed chkrootkit [2025-04-27 11:50:04] [+] Basic local security tools
installation completed
[2025-04-27 11:50:04] [*] Step 3/10: Configuring Fail2ban, Auditd, AIDE...
[2025-04-27 11:50:04] [-] Configuring Fail2ban... [2025-04-27 11:50:04] [+]
Created /etc/fail2ban/jail.local [2025-04-27 11:50:04] [!] Failed to ensure
SSH jail is enabled in jail.local. Check file manually. [2025-04-27 11:50:04]
[-] Restarting Fail2ban service... ./combined_hardening_script.txt: line 202:
service: command not found [2025-04-27 11:50:04] [!] Failed to restart Fail2ban.
Check logs.
[2025-04-27 11:50:04] [-] Configuring Auditd... [2025-04-27 11:50:04] [-]
Creating basic audit rules in /etc/audit/rules.d/99-hardening.rules... [2025-
04-27 11:50:04] [+] Created basic audit rules. [2025-04-27 11:50:04] [-]
Restarting Auditd service... * Restarting audit daemon auditd [ OK ] [2025-
04-27 11:50:04] [+] Auditd restarted successfully. System startup links for
/etc/init.d/auditd already exist. [2025-04-27 11:50:04] [+] Enabled Auditd
service on boot.
[2025-04-27 11:50:04] [-] Configuring AIDE (File Integrity)... [2025-04-27
11:50:04] [!] Initializing AIDE database. This may take a significant amount
of time... Running aide --init... AIDE, version 0.13.1 ### AIDE database
at /var/lib/aide/aide.db.new initialized. [2025-04-27 11:57:43] [+] AIDE
database initialization completed (/var/lib/aide/aide.db.new created). [2025-
04-27 11:57:43] [-] Automatically activating new AIDE database... [2025-04-27
11:57:43] [+] Successfully activated AIDE database (/var/lib/aide/aide.db.new
copied to /var/lib/aide/aide.db). [2025-04-27 11:57:43] [+] AIDE is now active.
```

```
[2025-04-27 11:57:43] [-] Suggestion: Add a cron job to run 'aide --check'
daily/weekly.
[2025-04-27 11:57:43] [-] Running chkrootkit scan (basic)... ROOTDIR is `/'
... (chkrootkit individual checks omitted) ... Checking `inetdconf'... INFECTED
... (chkrootkit individual checks omitted) ... Checking `bindshell'... INFECTED
(PORTS: 1524 6667) ... (chkrootkit individual checks omitted) ... Checking
`wted'... chkwtmp: nothing deleted Checking `z2'... chklastlog: nothing
deleted [2025-04-27 11:57:49] [+] chkrootkit scan completed. Results logged
to /var/log/security_hardening/hardening_20250427 − 114849.log.[2025 − 04 − 2711 : 57 :
49][!]Reviewchkrootkitoutputinthelogfileforanywarnings.
[2025-04-27 11:57:49] [*] Step 4/10: Removing known backdoors (while preserving
service function)... [2025-04-27 11:57:49] [+] No ingreslock backdoor shell
found in /etc/inetd.conf. ... (vsftpd and UnrealIRCd checks omitted - showed no
backdoor found) ...
[2025-04-27 11:57:49] [*] Step 5/10: Securing database services (MySQL,
PostgreSQL)... [2025-04-27 11:57:49] [-] Securing MySQL... [2025-04-
27 11:57:49] [!] MySQL service not running. Attempting to start...
./combined_hardening_script.txt: line 428: service: command not found [2025-
04-27 11:57:52] [!] Failed to start MySQL service. Cannot secure MySQL
automatically. [2025-04-27 11:57:52] [!] MySQL service is not running after
start attempt. Skipping MySQL hardening.
[2025-04-27 11:57:52] [-] Securing PostgreSQL... [2025-04-27 11:57:52] [-
] Found PostgreSQL config directory: /etc/postgresql/8.3/main [2025-04-
27 11:57:52] [-] Attempting to set PostgreSQL 'postgres' user password...
ALTER ROLE [2025-04-27 11:57:52] [+] PostgreSQL 'postgres' user password
updated successfully. [2025-04-27 11:57:52] [!] Recommended: Review
/etc/postgresql/8.3/main/pg_hba.conf and change 'trust' to 'md5' for
enhanced security. [2025-04-27 11:57:52] [-] Restarting PostgreSQL service...
./combined_hardening_script.txt: line 524: service: command not found
./combined_hardening_script.txt: line 524: service: command not found
[2025-04-27 11:57:52] [*] Step 6/10: Securing User Accounts  Access (SSH,
FTP, Telnet)... ... (User setup steps omitted - showed success) ... [2025-04-
27 11:57:53] [-] Restarting SSH service... ./combined_hardening_script.txt:
line 675: service: command not found [2025-04-27 11:57:53] [+] SSH service
configuration secured and restarted.
[2025-04-27 11:57:53] [*] Step 7/10: Setting up LOCAL firewall rules (iptables
on MS2)... ... (iptables setup steps omitted - showed success) ... Reading
package lists... Building dependency tree... Reading state information... E:
Couldn't find package iptables-persistent [2025-04-27 11:57:54] [!] Could not
install iptables-persistent. Relying on if-up.d script.
[2025-04-27 11:57:54] [*] Step 8/10: Basic web application security
(Apache, DVWA)... ... (Apache config steps omitted - showed success in
config test) ... [2025-04-27 11:57:54] [-] Restarting Apache service...
```

```
./combined_hardening_script.txt: line 869: service: command not found [2025-
04-27 11:57:54] [!] Apache restart failed even after successful config test.
Check service status and logs.
[2025-04-27 11:57:54] [-] Securing DVWA configuration... ... (DVWA config
steps omitted - showed file update success) ... [2025-04-27 11:57:54] [!]
Cannot automatically update DVWA password in MySQL because MySQL service is
not running or unavailable. [2025-04-27 11:57:54] [!] Manual Action Required:
Start MySQL, set MySQL root password to 'Gz19FgjWdk4BSN' (from credentials
file), then run the command above. [2025-04-27 11:57:54] [+] Secured DVWA
configuration (config file updated). Check MySQL status/logs for DB update
result.
[2025-04-27 11:57:54] [*] Step 9/10: Additional security measures... ... (Cron,
permissions, monitoring script, banner steps omitted - showed success) ...
[2025-04-27 11:57:54] [*] Step 10/10: Finalizing and summarizing... [2025-04-27
11:57:54] ======================================================================
[2025-04-27 11:57:54] METASPLOITABLE 2 ENHANCED HARDENING COMPLETED (Project 3
v7) [2025-04-27 11:57:54] =============================================================
[2025-04-27 11:57:54] [!] Key security improvements implemented on THIS
Metasploitable 2 VM: [2025-04-27 11:57:54] [!] - Updated package sources,
installed local tools (iptables, tcpd). [2025-04-27 11:57:54] [!] - Installed
Fail2ban, Auditd (HIDS), AIDE (File Integrity), chkrootkit. [2025-04-27
11:57:54] [!] - Configured Fail2ban (SSH protection), Auditd (basic rules).
[2025-04-27 11:57:54] [!] - Initialized and **AUTOMATICALLY ACTIVATED**
AIDE database. [2025-04-27 11:57:54] [!] - Ran chkrootkit scan (check log).
[2025-04-27 11:57:54] [!] - Checked/disabled known backdoors (ingreslock
shell, vsftpd vuln check, UnrealIRCd). [2025-04-27 11:57:54] [!] - Blocked
high-risk/backdoor ports LOCALLY via iptables (6000 6200 6667). [2025-04-27
11:57:54] [!] - Ensured required project ports remain accessible LOCALLY (21
22 23 80 139 445 1524 3306 5432 5900 8009 8180). [2025-04-27 11:57:54] [!] -
Attempted to secure DBs (PostgreSQL PW set; MySQL PW attempted - check status).
[2025-04-27 11:57:54] [!] - Hardened SSH (Root login disabled, Protocol 2
enforced, rate limiting). [2025-04-27 11:57:54] [!] - Created/Updated PROJECT
group user 'Group2' with predefined password. [2025-04-27 11:57:54] [!]
- *** GRANTED SUDO PRIVILEGES TO 'Group2' *** [2025-04-27 11:57:54] [!] -
Created backup 'secadmin' administrator account with a RANDOM password (also
has sudo). [2025-04-27 11:57:54] [!] - Changed default 'msfadmin' password.
[2025-04-27 11:57:55] [!] - Applied basic Apache security headers (Restart
conditional on config test). [2025-04-27 11:57:55] [!] - Secured DVWA config;
DB password update attempted (Check MySQL status). [2025-04-27 11:57:55] [!]
- Added daily security monitoring script  cron job (includes fail2ban/auditd
snippets). [2025-04-27 11:57:55] [!] - Set secure permissions on critical
files  cron. [2025-04-27 11:57:55] [!] - Added login banners. [2025-04-27
11:57:55] ======================================================================
[2025-04-27 11:57:55] [!] Secure credentials generated/used by this script
are saved at: /var/log/securityhardening/securecredentials.txt[2025 − 04 − 2711 :
57 : 55][!]PROTECTTHISFILE − ITCONTAINSPASSWORDS![2025 − 04 − 2711 : 57 :
55] ====================================================================
```

Log Analysis: The execution log confirms several successful hardening steps, including updating package sources via the ‘old-releases‘ archive, installing security tools like ‘fail2ban‘, ‘auditd‘, ‘aide‘, and ‘chkrootkit‘, changing default passwords, and applying initial firewall rules. However, it also clearly documents critical issues stemming from the outdated Ubuntu 8.04 environment.

Notably, the ‘service: command not found‘ errors appeared when attempting to restart Fail2ban, MySQL, PostgreSQL, SSH, and Apache. This indicates an incompatibility between the script's commands (assuming a modern ‘service‘ command) and the older SysV init system used by Metasploitable 2. While some services like Auditd were restarted using alternative methods (likely direct init script calls), others failed to restart automatically.

The most significant consequence was the failure to start the MySQL service, which prevented the script from setting the root password or the DVWA application password, leaving the database potentially insecure and non-functional without manual intervention. Furthermore, the failure to install ‘iptables-persistent‘ highlights the package management limitations. The ‘chkrootkit‘ scan also reported potential infections related to ‘inetd.conf‘ and listening bindshells (ports 1524, 6667), requiring further manual investigation beyond the script's automated checks. These logged events directly corroborate the challenges discussed in Section 4.

3.2 Server 2: IPFire Firewall Configuration

IPFire was manually configured with the following steps:

- Installation of the IPFire OS.
- Configuration of network interfaces: RED (WAN) and GREEN (LAN).
- Establishment of firewall rules: A default DENY policy for incoming traffic from the RED interface was set.
- Configuration of the integrated IPS (Intrusion Prevention System) with basic rulesets enabled.
- Securing administrative access: Strong passwords were set for the ‘admin‘ (Web GUI) and ‘root‘ (console) accounts.

3.3 Server 3: Debian IDS Configuration

The Debian server was manually configured as an IDS:

- Installation of a minimal Debian base system (netinstall).
- System updates (‘apt update && apt upgrade‘).
- Installation and configuration of Suricata IDS. This involved setting the ‘HOME_NET‘ variable, configuring the network interface connected to the mirrored switch port for sniffing, and enabling relevant Suricata rulesets (e.g., Emerging Threats Open).

- Basic OS hardening: ‘ufw‘ (Uncomplicated Firewall) or ‘iptables‘ was configured to restrict local listening ports (allowing only SSH - port 22). SSH was hardened similarly to MS2 (disabling root login, protocol 2).
- User account management: The ‘Group2‘ user was created with the specified password and granted ‘sudo‘ privileges for administration.
- The root account password was set to a strong value.
- (Optional) Installation of a web interface like EveBox for easier alert viewing (requires port forwarding rule WAN 35008 → 192.168.1.102:5636).

4 Difficulties Encountered

Several challenges were faced during the hardening process, primarily related to the age and limitations of the Metasploitable 2 platform and the constraints of the virtual environment.

- **Outdated Operating System (Ubuntu 8.04):** Metasploitable 2 is based on Ubuntu 8.04 ”Hardy Heron,” which is long past its end-of-life.
  - **Package Management:** Standard package repositories are offline. We had to reconfigure ‘apt‘ to use the ‘old-releases.ubuntu.com‘ archive.
  - **Compatibility Issues:** Modern security tools and techniques often assume newer kernel features or libraries not present in Hardy Heron.
  - **Lack of Modern Init System:** The system uses the older SysV init system.
- **Resource Limitations:** While the firewall and IDS servers were chosen to be lightweight, running three virtual machines under the specified constraints required careful resource allocation.
- **MySQL Service Failure:** During the execution of the initial hardening script (‘Hardningscript3 1.txt‘), the MySQL service failed to start (‘[2025-04-27 11:57:52] [!] Failed to start MySQL service. Cannot secure MySQL automatically.‘).
  - The script could not automatically set the MySQL root password.
  - Subsequent steps to remove anonymous users and secure the root account could not be performed automatically.
  - The attempt to update the DVWA application’s database password also failed, as it relied on connecting to the MySQL server with the (unset) root credentials (‘[2025-04-27 11:57:54] [!] Cannot automatically update DVWA password in MySQL because MySQL service is not running or unavailable.‘).
  - This likely left the MySQL database in an insecure state (default root password potentially blank) and broke the functionality of applications relying on it (like DVWA or potentially Metasploit framework database features if configured to use the local MySQL).

- **Initial Firewall Permissiveness:** The first iteration of the local firewall rules within ‘Hardningscript3 1.txt‘ used a default ACCEPT policy.

5 Self-Evaluation

Overall, the implemented strategy significantly improved the security posture of the Metasploitable 2 environment compared to its default state.

5.1 Effectiveness

- **Network Segmentation:** The use of a dedicated firewall (IPFire) provides a strong perimeter defense, controlling traffic flow into and out of the internal network.
- **Intrusion Detection:** The Debian/Suricata IDS offers visibility into network traffic, allowing for the detection of known attack patterns and anomalies based on its rulesets.
- **Host Hardening (MS2):**
  - The scripts successfully addressed numerous critical vulnerabilities: known backdoors were checked/disabled
  - The installation of Fail2ban, Auditd, and AIDE provides layers of host-based intrusion detection, auditing, and integrity checking.
  - The refined firewall rules implemented by ‘updateandport.txt‘ (default DROP) are a significant improvement over the initial script’s rules, drastically reducing the host’s local attack surface by only allowing explicitly needed ports.
- **User Access Control:** Creating dedicated user accounts (‘Group2‘, ‘secadmin‘) with strong passwords and removing/changing default credentials enhances accountability and reduces the risk associated with default logins.

5.2 Limitations and Areas for Improvement

- **MySQL Hardening Failure:** The inability of the script to automatically start and secure the MySQL server is a significant weakness.
- **Outdated Platform Risks:** Despite hardening efforts, the underlying Ubuntu 8.04 OS remains inherently vulnerable due to its age and lack of security patches for the kernel and core libraries.
- **Tool Configuration Depth:** While tools like Fail2ban, Auditd, AIDE, and Suricata were planned or partially implemented, optimizing their rulesets and policies for the specific environment would require significantly more time and effort (e.g., fine-tuning Suricata rules to reduce false positives, customizing AIDE policies, creating more specific Fail2ban jails).
- **Manual Steps:** Configuration of IPFire and Debian/Suricata involved manual steps.
- **Reliance on ‘old-releases‘:** The dependency on the ‘old-releases.ubuntu.com‘ archive is a potential point of failure if that archive becomes unavailable.

5.3 Additional Measures Considered

Several additional measures could have further enhanced security but were deemed too costly or time-consuming given the project constraints:

- **OS Upgrade/Replacement:** Ideally, replacing Metasploitable 2 with a containerized set of vulnerable services on a modern, supported OS would eliminate the risks associated with the outdated base system.
- **Web Application Firewall (WAF):** Implementing a WAF (e.g., ModSecurity on Apache, or via the firewall) could provide more granular protection for web applications like DVWA beyond basic header security.
- **Advanced IDS/IPS Tuning:** Deeply customizing Suricata rules, implementing custom rules, and enabling full IPS blocking mode on IPFire would enhance detection and prevention but require significant tuning effort.
- **Centralized Logging:** Implementing a centralized logging server (e.g., ELK stack, Graylog) to aggregate logs from all three servers would improve monitoring and correlation capabilities but adds complexity.
- **Vulnerability Management Solution:** Deploying a proper vulnerability scanner internally to continuously assess the environment would provide ongoing insights but requires additional resources and setup.

6 Citations

Resources and commands used are often indicated via comments within the provided shell scripts (‘Hardningscript3 1.txt‘ and ‘updateandport.txt‘). These appear as standard comments or occasionally as annotations within the script listings (Listings 3.1.1 and 3.1.2). No external resources beyond standard Linux/Ubuntu documentation and tool manuals were explicitly used for script development beyond what might be reflected in those inline comments.

7 Conclusion

Group 2 successfully implemented a layered security architecture to harden the provided Metasploitable 2 environment. Key achievements include network segmentation via a dedicated firewall, network traffic monitoring with an IDS, and significant host-based hardening of the MS2 server using automated scripts. The refinement of the local firewall policy using the ‘updateandport.txt‘ script was a critical corrective action. While challenges related to the outdated OS and service startup issues (notably MySQL) were encountered, the overall security posture was substantially improved from the default state. Further enhancements are possible but would require addressing the limitations of the underlying platform or investing significantly more time in advanced tool configuration. The effectiveness of these measures will be further evaluated based on the results of the instructor’s scans and attacks.