S.I.G.I.E
Chronos
Ciberseguridad

Rol	Apellido	Nombre	Cédula Identidad	Email	Tel./Cel.
Coordinador	Chapuis	Juan	5.630.283-4	jchapuis@scuolaitaliana.edu.uy	098844043
Sub-Coordinador	Roizen	Agustin	6.339.592-9	aroizen@scuolaitaliana.edu.uy	097259510
Integrante	Marino	Bruno	5.707.620-6	bmarino@scuolaitaliana.edu.uy	092617596

Docente: Farias, Gustavo

Fecha de entrega: 03/11/2025

Tercera Entrega

CIBERSEGURIDAD

PRIMERA ENTREGA

a) Relevamiento de configuraciones básicas de seguridad del SO seleccionado

SCRIPT DE AUDITORÍA DE SEGURIDAD: `security_audit.sh`

```
#!/bin/bash

echo "============================================="
echo "        AUDITORIA DE SEGURIDAD S.I.G.I.E"
echo "Fecha: $(date)"
echo "Servidor: $(hostname)"
echo "============================================="

# 1. Verificación de permisos del sistema
echo "1. PERMISOS DE ARCHIVOS CRÍTICOS:"
echo "---------------------------------"
ls -l /etc/passwd | awk '{print "   /etc/passwd: " $1 " " $3 " " $4}'
ls -l /etc/shadow | awk '{print "   /etc/shadow: " $1 " " $3 " " $4}'
ls -l /etc/sudoers | awk '{print "   /etc/sudoers: " $1 " " $3 " " $4}'
ls -l /etc/ssh/sshd_config | awk '{print "   /etc/ssh/sshd_config: " $1 " " $3 " " $4}'

# 2. Políticas de contraseñas
echo ""
echo "2. POLÍTICAS DE CONTRASEÑAS:"
echo "----------------------------"
if [ -f "/etc/security/pwquality.conf" ]; then
    echo "   Configuración pwquality.conf:"
    grep -E "minlen|minclass|ucredit|lcredit|dcredit|ocredit" /etc/security/pwquality.conf
else
    echo "   /etc/security/pwquality.conf no existe"
fi

echo ""
echo "   Configuración login.defs:"
grep -E "PASS_MAX_DAYS|PASS_MIN_DAYS|PASS_WARN_AGE" /etc/login.defs

# 3. Verificación de UMASK
echo ""
echo "3. CONFIGURACIÓN UMASK:"
```

```
echo "-----------------------"
echo "   UMASK global: $(umask)"
echo "   UMASK en /etc/profile: $(grep -i umask /etc/profile | head -1)"
echo "   UMASK en /etc/bash.bashrc: $(grep -i umask /etc/bash.bashrc | head -1)"

# 4. Usuarios y grupos
echo ""
echo "4. USUARIOS DEL SISTEMA:"
echo "-----------------------"
echo "   Usuarios con shell: $(grep -v "/nologin\|/false" /etc/passwd | cut -d: -f1 | wc
-l)"
echo "   Usuarios con UID 0: $(awk -F: '$3==0 {print $1}' /etc/passwd)"
echo "   Usuarios sin contraseña: $(awk -F: '$2=="" {print $1}' /etc/shadow | wc -l)"

# 5. Servicios en ejecución
echo ""
echo "5. SERVICIOS ACTIVOS:"
echo "---------------------"
systemctl list-units --type=service --state=running --no-pager | head -10

# 6. Puertos abiertos
echo ""
echo "6. PUERTOS ABIERTOS:"
echo "--------------------"
ss -tuln | head -15

# 7. Configuración Docker
echo ""
echo "7. CONFIGURACIÓN DOCKER:"
echo "-----------------------"
echo "   Usuarios en grupo docker: $(getent group docker | cut -d: -f4)"
echo "   Contenedores ejecutándose: $(docker ps -q | wc -l)"
echo "   Todas las imágenes: $(docker images -q | wc -l)"

# 8. Configuración SSH
echo ""
echo "8. CONFIGURACIÓN SSH:"
echo "---------------------"
if [ -f "/etc/ssh/sshd_config" ]; then
    echo "   Puerto SSH: $(grep -E "^Port" /etc/ssh/sshd_config || echo "22")"
    echo "   PermitRootLogin: $(grep -E "^PermitRootLogin" /etc/ssh/sshd_config ||
echo "yes")"
    echo "   PasswordAuthentication: $(grep -E "^PasswordAuthentication"
/etc/ssh/sshd_config || echo "yes")"
```

```
else
    echo "   /etc/ssh/sshd_config no existe"
fi

# 9. Firewall
echo ""
echo "9. ESTADO FIREWALL:"
echo "--------------------"
if command -v ufw > /dev/null; then
    ufw status verbose
else
    echo "   UFW no instalado"
fi

# 10. Fail2ban
echo ""
echo "10. ESTADO FAIL2BAN:"
echo "--------------------"
if systemctl is-active --quiet fail2ban; then
    fail2ban-client status
else
    echo "   Fail2ban no activo"
fi

echo ""
echo "================================================"
echo "        AUDITORIA COMPLETADA"
echo "================================================"
```

b) Implementación en el script de administración de usuarios de reglas básicas de seguridad

SCRIPT DE CREACIÓN DE USUARIOS SEGURO: `crear_usuarios_seguro.sh`

```
#!/bin/bash

echo "================================================"
echo "    CREACION DE USUARIOS CON SEGURIDAD"
echo "================================================"

UMASK_VALUE="0027"
PASSWORD_MIN_LENGTH=12

declare -A usuarios=(
```

```
    ["admin_sis"]="Administrador de Sistemas"
    ["admin_sec"]="Administrador de Seguridad"
    ["operador"]="Operador del Sistema"
    ["dev_user"]="Usuario Desarrollo"
    ["docker_user"]="Usuario Docker"
)

function check_password_strength() {
    local password="$1"
    local username="$2"

    if [ ${#password} -lt $PASSWORD_MIN_LENGTH ]; then
        echo "ERROR: La contraseña debe tener al menos
$PASSWORD_MIN_LENGTH caracteres"
        return 1
    fi

    local has_upper=$(echo "$password" | grep -E '[A-Z]' | wc -l)
    local has_lower=$(echo "$password" | grep -E '[a-z]' | wc -l)
    local has_digit=$(echo "$password" | grep -E '[0-9]' | wc -l)
    local has_special=$(echo "$password" | grep -E '[!@#$%^&*()_+-=]' | wc -l)

    local complexity_score=0
    [ $has_upper -gt 0 ] && ((complexity_score++))
    [ $has_lower -gt 0 ] && ((complexity_score++))
    [ $has_digit -gt 0 ] && ((complexity_score++))
    [ $has_special -gt 0 ] && ((complexity_score++))

    if [ $complexity_score -lt 3 ]; then
        echo "ERROR: La contraseña debe contener al menos 3 tipos de caracteres"
        return 1
    fi

    if echo "$password" | grep -i "$username" > /dev/null; then
        echo "ERROR: La contraseña no debe contener el nombre de usuario"
        return 1
    fi

    echo "OK: Contraseña segura"
    return 0
}

function configure_secure_umask() {
    local user="$1"
```

```
    echo "Configurando UMASK seguro ($UMASK_VALUE) para $user"

    sudo tee -a /home/$user/.bashrc > /dev/null << EOF

# Configuracion de seguridad - UMASK
umask $UMASK_VALUE
EOF

    sudo tee -a /home/$user/.profile > /dev/null << EOF

# Configuracion de seguridad - UMASK
umask $UMASK_VALUE
EOF

    sudo chmod 700 /home/$user
    sudo chown $user:$user /home/$user/.bashrc
    sudo chown $user:$user /home/$user/.profile
}

function create_secure_user() {
    local user="$1"
    local comment="$2"

    echo "Creando usuario seguro: $user"

    sudo useradd -m -c "$comment" -s /bin/bash "$user"

    local temp_password="Temp$(date +%s | tail -c 4)!Segura"
    echo "$user:$temp_password" | sudo chpasswd

    sudo chage -d 0 "$user"
    sudo chage -M 90 "$user"
    sudo chage -W 7 "$user"

    echo "Usuario: $user"
    echo "Contraseña temporal: $temp_password"
    echo "Debe cambiar la contraseña en primer login"

    configure_secure_umask "$user"
}

echo "Creando grupos del sistema..."
sudo groupadd -f admin_sis
```

```
sudo groupadd -f admin_sec
sudo groupadd -f operadores
sudo groupadd -f desarrolladores
sudo groupadd -f docker

for usuario in "${!usuarios[@]}"; do
    if ! id "$usuario" &>/dev/null; then
        create_secure_user "$usuario" "${usuarios[$usuario]}"

        case $usuario in
            admin_sis)
                sudo usermod -aG admin_sis,docker "$usuario"
                echo "Grupos: admin_sis, docker"
                ;;
            admin_sec)
                sudo usermod -aG admin_sec "$usuario"
                echo "Grupos: admin_sec"
                ;;
            operador)
                sudo usermod -aG operadores "$usuario"
                echo "Grupos: operadores"
                ;;
            dev_user)
                sudo usermod -aG desarrolladores,docker "$usuario"
                echo "Grupos: desarrolladores, docker"
                ;;
            docker_user)
                sudo usermod -aG docker "$usuario"
                echo "Grupos: docker"
                ;;
        esac
        echo "-----------------------------------------------"
    else
        echo "Usuario $usuario ya existe"
    fi
done

echo "Configurando permisos sudo restringidos..."
sudo tee /etc/sudoers.d/admin_roles > /dev/null << 'EOF
# Configuracion de seguridad S.I.G.I.E
# Permisos minimos necesarios

%admin_sis ALL=(ALL) /usr/bin/systemctl, /usr/bin/docker,
/usr/local/bin/docker-compose, /usr/local/admin-scripts/
```

```
%admin_sec ALL=(ALL) /usr/sbin/ufw, /usr/bin/passwd, /usr/bin/systemctl restart
ssh, /usr/bin/fail2ban-client
%operadores ALL=(ALL) /usr/local/admin-scripts/menu_usuario.sh, /usr/bin/passwd,
/usr/local/admin-scripts/backup_sistema.sh
%docker ALL=(ALL) /usr/bin/docker, /usr/local/bin/docker-compose

# Comandos prohibidos para todos
Cmnd_Alias DANGEROUS = /usr/bin/passwd root, /usr/bin/visudo, /usr/bin/chmod
777, /usr/bin/chown root:root

ALL ALL=(ALL) !DANGEROUS
EOF

sudo chmod 440 /etc/sudoers.d/admin_roles

echo "============================================="
echo "    CONFIGURACION DE SEGURIDAD COMPLETADA"
echo "============================================="
echo "UMASK seguro: $UMASK_VALUE"
echo "Longitud minima contraseña: $PASSWORD_MIN_LENGTH"
echo "Contraseñas expiran: 90 dias"
echo "Aviso de expiracion: 7 dias"
```

c) Establecimiento de una política de contraseñas simples

POLÍTICA DE CONTRASEÑAS SEGURAS:

Requisitos Mínimos:

1. **Longitud:** 12 caracteres mínimo
2. **Complejidad:** 3 de 4 tipos de caracteres:
    - Letras mayúsculas (A-Z)
    - Letras minúsculas (a-z)
    - Números (0-9)
    - Caracteres especiales (!@#$%^&*()_+-=)
3. **Restricciones:**
    - No contener el nombre de usuario
    - No más de 2 caracteres repetidos consecutivos
    - No usar palabras del diccionario comunes
4. **Vigencia:**
    - Expiración: 90 días
    - Historial: No reutilizar últimas 5 contraseñas
    - Bloqueo: Después de 3 intentos fallidos

Ejemplos:

- Correcto: "M1_contraseñaSegura!"
- Correcto: "C4f3@2024!Seguro"
- Incorrecto: "password123"
- Incorrecto: "admin123456"

SCRIPT DE CONFIGURACIÓN: `configure_password_policy.sh`

bash
```
#!/bin/bash

echo "============================================="
echo "  CONFIGURACION POLITICA DE CONTRASEÑAS"
echo "============================================="

# Instalar libpam-pwquality si no está presente
if ! dpkg -l | grep -q libpam-pwquality; then
    echo "Instalando libpam-pwquality..."
    sudo apt update
    sudo apt install -y libpam-pwquality
fi

echo "Configurando politica de contraseñas seguras..."

# Configurar política de contraseñas en pwquality
sudo tee /etc/security/pwquality.conf > /dev/null << 'EOF'
# Politica de contraseñas seguras - S.I.G.I.E
minlen = 12
minclass = 3
maxrepeat = 2
dictcheck = 1
usercheck = 1
enforcing = 1
retry = 3
EOF

# Configurar PAM para usar pwquality
sudo tee /etc/pam.d/common-password > /dev/null << 'EOF'
# Contraseñas: politica S.I.G.I.E
password requisite pam_pwquality.so retry=3
password [success=1 default=ignore] pam_unix.so obscure use_authtok
try_first_pass yescrypt
password requisite pam_deny.so
password required pam_permit.so
EOF

# Configurar envejecimiento de contraseñas
echo "Configurando envejecimiento de contraseñas..."
sudo sed -i 's/^PASS_MAX_DAYS.*/PASS_MAX_DAYS   90/' /etc/login.defs
sudo sed -i 's/^PASS_MIN_DAYS.*/PASS_MIN_DAYS   1/' /etc/login.defs
```

```
sudo sed -i 's/^PASS_WARN_AGE.*/PASS_WARN_AGE   7/' /etc/login.defs

# Aplicar políticas a usuarios existentes
echo "Aplicando politicas a usuarios existentes..."
for user in $(getent passwd | grep -E "/bin/bash|/bin/sh" | cut -d: -f1); do
    if [ "$user" != "root" ]; then
        sudo chage -M 90 -m 1 -W 7 "$user"
        echo "  $user: Contraseña expira en 90 días"
    fi
done

# Crear script educativo de contraseñas
sudo tee /usr/local/bin/password-policy > /dev/null << 'EOF'
#!/bin/bash
echo "POLITICA DE CONTRASEÑAS S.I.G.I.E"
echo "================================="
echo "REQUISITOS:"
echo "✅ Minimo 12 caracteres"
echo "✅ Al menos 3 de estos tipos:"
echo "   - Letras mayusculas (A-Z)"
echo "   - Letras minusculas (a-z)"
echo "   - Numeros (0-9)"
echo "   - Caracteres especiales (!@#$%^&*)"
echo "✅ No mas de 2 caracteres repetidos"
echo "✅ No debe contener el nombre de usuario"
echo "✅ Expira cada 90 dias"
echo "✅ No reutilizar contraseñas recientes"
echo ""
echo "EJEMPLOS:"
echo "✅ Correcto: 'M1_contraseñaSegura!'"
echo "✅ Correcto: 'C4f3@2024!Seguro'"
echo "❌ Incorrecto: 'password123'"
echo "❌ Incorrecto: 'admin123456'"
EOF

sudo chmod +x /usr/local/bin/password-policy

echo ""
echo "============================================="
echo "  POLITICA DE CONTRASEÑAS CONFIGURADA"
echo "============================================="
echo "✅ Expiración: 90 días"
echo "✅ Longitud mínima: 12 caracteres"
echo "✅ Complejidad: 3 de 4 tipos de caracteres"
```

```
echo "✅ Reintentos: 3"
echo ""
echo "Comando: password-policy  # Ver politica completa"
```

d) Verificación de permisos con el comando umask al crear usuarios

CONFIGURACIÓN UMASK SEGURA:

Valor UMASK: 0027

- Archivos: 666 - 027 = 640 (rw-r-----)
- Directorios: 777 - 027 = 750 (rwxr-x---)

Implementación en Scripts:

```
# Verificar UMASK actual
echo "UMASK actual: $(umask)"

# Configurar UMASK para nuevos usuarios
function configure_secure_umask() {
    local user="$1"

    # Configurar en bashrc
    sudo tee -a /home/$user/.bashrc > /dev/null << EOF
# Configuracion de seguridad - UMASK
umask 0027
EOF

    # Configurar en profile
    sudo tee -a /home/$user/.profile > /dev/null << EOF
# Configuracion de seguridad - UMASK
umask 0027
EOF

    # Aplicar permisos seguros
    sudo chmod 750 /home/$user
    sudo chown $user:$user /home/$user/.bashrc
    sudo chown $user:$user /home/$user/.profile

    echo "UMASK 0027 configurado para $user"
}

# Verificar UMASK de usuarios existentes
function verify_user_umask() {
    local user="$1"
```

```
    echo "Verificando UMASK para $user:"
    sudo su - $user -c "umask"
}

# Script de verificacion de UMASK
sudo tee /usr/local/bin/check-umask > /dev/null << 'EOF'
#!/bin/bash
echo "VERIFICACION DE CONFIGURACION UMASK"
echo "==================================="
echo "UMASK global: $(umask)"
echo ""
echo "UMASK por usuario:"
for user in $(getent passwd | grep -E "/bin/bash|/bin/sh" | cut -d: -f1); do
    umask_val=$(sudo su - $user -c "umask" 2>/dev/null)
    echo "  $user: $umask_val"
done
EOF

sudo chmod +x /usr/local/bin/check-umask
```

SEGUNDA ENTREGA

a) Configuración segura del servicio SSH

CONFIGURACIÓN SSH SEGURO: `secure_ssh.sh`

```
#!/bin/bash

SSH_PORT="666"
SSH_CONFIG="/etc/ssh/sshd_config"
BACKUP_DIR="/backups/ssh"

echo "================================================"
echo "        CONFIGURACION SSH SEGURO"
echo "================================================"

# Crear directorio de backups
mkdir -p "$BACKUP_DIR"

# Crear backup de la configuracion actual
if [ -f "$SSH_CONFIG" ]; then
    sudo cp "$SSH_CONFIG" "$BACKUP_DIR/sshd_config.backup.$(date
+%Y%m%d_%H%M%S)"
```

```
    echo "Backup creado en: $BACKUP_DIR/sshd_config.backup.$(date
+%Y%m%d_%H%M%S)"
fi

echo "Configurando SSH seguro en puerto $SSH_PORT..."

# Configuracion segura de SSH
sudo tee "$SSH_CONFIG" > /dev/null << EOF
# Configuracion SSH Segura - S.I.G.I.E
Port $SSH_PORT
Protocol 2

# Autenticacion
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
PermitEmptyPasswords no
ChallengeResponseAuthentication no

# Seguridad de sesion
X11Forwarding no
AllowTcpForwarding no
AllowAgentForwarding no
MaxAuthTries 3
MaxSessions 3
ClientAliveInterval 300
ClientAliveCountMax 2
LoginGraceTime 60

# Control de acceso
AllowGroups admin_sis admin_sec
DenyUsers root

# Cifrado
KexAlgorithms curve25519-sha256@libssh.org
Ciphers
chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@open
ssh.com
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com

# Logging
LogLevel VERBOSE
SyslogFacility AUTH
```

```
# Restricciones
UsePAM yes
IgnoreRhosts yes
HostbasedAuthentication no
PrintMotd no
PrintLastLog yes
StrictModes yes
EOF

# Configurar firewall para el nuevo puerto SSH
echo "Configurando firewall..."
sudo ufw allow $SSH_PORT/tcp comment 'SSH Seguro S.I.G.I.E'
sudo ufw deny 22/tcp comment 'SSH por defecto deshabilitado'

# Generar claves SSH para administradores
echo "Generando claves SSH para administradores..."
for user in admin_sis admin_sec; do
    if id "$user" &>/dev/null; then
        user_home=$(eval echo ~$user)
        if [ ! -f "$user_home/.ssh/id_ed25519" ]; then
            echo "Generando claves para $user..."
            sudo -u "$user" mkdir -p "$user_home/.ssh"
            sudo -u "$user" ssh-keygen -t ed25519 -f "$user_home/.ssh/id_ed25519" -N
"" -C "$user@$(hostname)"
            sudo -u "$user" cat "$user_home/.ssh/id_ed25519.pub" >>
"$user_home/.ssh/authorized_keys"
            sudo chmod 700 "$user_home/.ssh"
            sudo chmod 600 "$user_home/.ssh/authorized_keys"
            sudo chmod 644 "$user_home/.ssh/id_ed25519.pub"
            sudo chmod 600 "$user_home/.ssh/id_ed25519"
            echo "Claves SSH generadas para $user"
            echo "Clave publica:"
            sudo -u "$user" cat "$user_home/.ssh/id_ed25519.pub"
        else
            echo "El usuario $user ya tiene claves SSH"
        fi
    fi
done

# Reiniciar servicio SSH
echo "Reiniciando servicio SSH..."
sudo systemctl restart ssh

# Verificar configuracion
```

```
echo "Verificando configuracion SSH..."
sudo sshd -t
if [ $? -eq 0 ]; then
    echo "Configuracion SSH valida"
else
    echo "Error en configuracion SSH - Revise los logs"
    exit 1
fi

echo ""
echo "================================================"
echo "        SSH CONFIGURADO EXITOSAMENTE"
echo "================================================"
echo "Puerto: $SSH_PORT"
echo "Autenticacion: Solo claves SSH"
echo "Root login: Deshabilitado"
echo "Grupos permitidos: admin_sis, admin_sec"
echo ""
echo "INSTRUCCIONES IMPORTANTES:"
echo "1. Configure su cliente SSH para usar el puerto $SSH_PORT"
echo "2. Use claves SSH para autenticarse"
echo "3. Pruebe la conexion antes de cerrar esta sesion"
echo "4. Ejemplo de conexion:"
echo "   ssh -p $SSH_PORT -i ~/.ssh/clave_privada usuario@servidor"
```

b) Comprobación de firewall activo y creación de reglas mínimas

CONFIGURACIÓN FIREWALL: `configure_firewall_secure.sh`

```
#!/bin/bash

echo "================================================"
echo "        CONFIGURACION FIREWALL SEGURO"
echo "================================================"

# Instalar UFW si no está presente
if ! command -v ufw > /dev/null; then
    echo "Instalando UFW..."
    sudo apt update
    sudo apt install -y ufw
fi

echo "Reseteando configuracion de firewall..."
sudo ufw --force reset
```

```
echo "Configurando politicas por defecto..."
sudo ufw default deny incoming
sudo ufw default allow outgoing

echo "Aplicando reglas minimas esenciales..."

# SSH seguro
sudo ufw allow 666/tcp comment 'SSH Seguro S.I.G.I.E'

# Servicios de aplicacion Docker
sudo ufw allow 3000/tcp comment 'App Chronos'
sudo ufw allow 8080/tcp comment 'Admin Interface'

# Servicios esenciales salientes
sudo ufw allow out 53 comment 'DNS'
sudo ufw allow out 80 comment 'HTTP'
sudo ufw allow out 443 comment 'HTTPS'

# Habilitar UFW
echo "Activando firewall..."
echo "y" | sudo ufw enable

# Verificar estado
echo "Estado del firewall:"
sudo ufw status verbose

# Configurar iptables persistente
echo "Configurando persistencia..."
sudo apt install -y iptables-persistent

# Crear script de verificacion
sudo tee /usr/local/bin/firewall-status > /dev/null << 'EOF'
#!/bin/bash
echo "ESTADO DEL FIREWALL S.I.G.I.E"
echo "============================="
sudo ufw status verbose
echo ""
echo "REGLAS ACTIVAS:"
sudo ufw status numbered
EOF

sudo chmod +x /usr/local/bin/firewall-status
```

```
echo ""
echo "============================================="
echo "       FIREWALL CONFIGURADO EXITOSAMENTE"
echo "============================================="
echo "POLITICAS:"
echo "  Entrante: DENY"
echo "  Saliente: ALLOW"
echo ""
echo "PUERTOS ABIERTOS:"
echo "  SSH: 666/tcp"
echo "  App Chronos: 3000/tcp"
echo "  Admin Interface: 8080/tcp"
echo ""
echo "COMANDOS UTILES:"
echo "  firewall-status          # Ver estado completo"
echo "  sudo ufw allow <puerto>  # Abrir puerto"
echo "  sudo ufw delete <numero> # Eliminar regla"
```

c) Implementación de control básico de acceso a los scripts mediante permisos

CONTROL DE ACCESO A SCRIPTS: `secure_script_permissions.sh`

```
#!/bin/bash

echo "============================================="
echo "    CONTROL DE ACCESO A SCRIPTS"
echo "============================================="

SCRIPT_DIR="/usr/local/admin-scripts"
BACKUP_DIR="/backups/script_permissions"

# Crear backup de permisos actuales
mkdir -p "$BACKUP_DIR"
echo "Creando backup de permisos actuales..."
getfacl -R "$SCRIPT_DIR" > "$BACKUP_DIR/script_permissions_backup_$(date
+%Y%m%d).acl"

# Establecer propietario y grupo seguro
echo "Configurando propietarios..."
sudo chown -R root:admin "$SCRIPT_DIR"

# Aplicar permisos seguros
echo "Aplicando permisos seguros..."
```

```
# Directorio principal - solo root y admin pueden acceder
sudo chmod 750 "$SCRIPT_DIR"

# Scripts individuales - ejecutables solo por propietario y grupo
for script in "$SCRIPT_DIR"/*.sh; do
    if [ -f "$script" ]; then
        script_name=$(basename "$script")

        # Permisos: root (rwx), admin (r-x), otros (---)
        sudo chmod 750 "$script"

        # Verificar que no tenga permisos peligrosos
        dangerous_perms=$(stat -c "%a" "$script" | grep -E ".[2-7][2-7].")
        if [ -n "$dangerous_perms" ]; then
            echo "Ajustando permisos peligrosos en: $script_name"
            sudo chmod 750 "$script"
        fi

        echo "  $script_name - $(stat -c "%A" "$script")"
    fi
done

# Scripts específicos con permisos especiales
if [ -f "$SCRIPT_DIR/menu_usuario.sh" ]; then
    sudo chmod 755 "$SCRIPT_DIR/menu_usuario.sh"
    echo "  menu_usuario.sh - Permisos especiales para usuarios"
fi

# Verificar integridad de permisos
echo "Verificando permisos finales..."
find "$SCRIPT_DIR" -type f -name "*.sh" -exec ls -l {} \; | while read line; do
    perms=$(echo $line | awk '{print $1}')
    file=$(echo $line | awk '{print $9}')

    # Verificar que no haya permisos de escritura para grupo u otros
    if echo "$perms" | grep -qE "^...w.|^.....w"; then
        echo "PERMISOS DE ESCRITURA INSEGUROS EN: $file"
        sudo chmod o-w "$file"
        sudo chmod g-w "$file"
    fi
done

# Configurar ACL avanzadas para auditoría
echo "Configurando ACL de auditoría..."
```

```
sudo setfacl -R -m g:admin_sec:r-x "$SCRIPT_DIR"
sudo setfacl -R -m g:operadores:r-x "$SCRIPT_DIR/menu_usuario.sh"

# Crear script de verificación de permisos
sudo tee /usr/local/bin/check-script-perms > /dev/null << 'EOF'
#!/bin/bash
echo "VERIFICACION DE PERMISOS DE SCRIPTS"
echo "==================================="
echo "Directorio: /usr/local/admin-scripts"
echo ""

for script in /usr/local/admin-scripts/*.sh; do
    if [ -f "$script" ]; then
        perms=$(stat -c "%A %U %G" "$script")
        echo "$perms - $(basename $script)"
    fi
done

echo ""
echo "RESUMEN:"
echo "  Propietario: root"
echo "  Grupo: admin"
echo "  Permisos: 750 (rwxr-x---)"
echo "  No escritura para otros"
EOF

sudo chmod +x /usr/local/bin/check-script-perms

echo ""
echo "============================================="
echo "    PERMISOS CONFIGURADOS EXITOSAMENTE"
echo "============================================="
echo "Propietario: root"
echo "Grupo: admin"
echo "Permisos: 750 (rwxr-x---)"
echo "ACL: admin_sec (r-x), operadores (r-x en menu_usuario.sh)"
echo ""
echo "Comando: check-script-perms  # Verificar permisos"
```

d) Verificación de integridad básica de scripts usando sha256sum

VERIFICACIÓN DE INTEGRIDAD: `verify_script_integrity.sh`

```
#!/bin/bash
```

```
INTEGRITY_FILE="/var/log/admin-system/script_integrity.log"
HASH_FILE="/etc/admin-scripts/script_hashes.sha256"
SCRIPT_DIR="/usr/local/admin-scripts"

echo "================================================"
echo "    VERIFICACION DE INTEGRIDAD DE SCRIPTS"
echo "================================================"

# Crear directorio para hashes si no existe
sudo mkdir -p /etc/admin-scripts

function calculate_hashes() {
    echo "Calculando hashes de scripts..."
    sudo mkdir -p $(dirname "$HASH_FILE")
    sudo touch "$HASH_FILE"
    sudo chmod 600 "$HASH_FILE"

    > /tmp/current_hashes.txt

    for script in "$SCRIPT_DIR"/*.sh; do
        if [ -f "$script" ]; then
            hash=$(sha256sum "$script" | awk '{print $1}')
            name=$(basename "$script")
            echo "$hash $name" >> /tmp/current_hashes.txt
            echo "  $name - $hash"
        fi
    done

    # Incluir el menú unificado
    if [ -f "/usr/local/bin/admin-menu" ]; then
        hash=$(sha256sum "/usr/local/bin/admin-menu" | awk '{print $1}')
        echo "$hash admin-menu" >> /tmp/current_hashes.txt
        echo "  admin-menu - $hash"
    fi

    sudo mv /tmp/current_hashes.txt "$HASH_FILE"
    echo "Hashes guardados en $HASH_FILE"
}

function verify_hashes() {
    echo "Verificando integridad..."

    if [ ! -f "$HASH_FILE" ]; then
```

```
        echo "ERROR: No se encontro archivo de hashes."
        echo "Ejecute primero: $0 --calculate"
        return 1
    fi

    errors=0
    while read -r line; do
        expected_hash=$(echo "$line" | awk '{print $1}')
        script_name=$(echo "$line" | awk '{print $2}')

        if [ "$script_name" == "admin-menu" ]; then
            current_hash=$(sha256sum "/usr/local/bin/admin-menu" | awk '{print $1}')
            script_path="/usr/local/bin/admin-menu"
        else
            current_hash=$(sha256sum "$SCRIPT_DIR/$script_name" | awk '{print $1}')
            script_path="$SCRIPT_DIR/$script_name"
        fi

        if [ "$current_hash" == "$expected_hash" ]; then
            echo "  ✅ $script_name - INTEGRO"
        else
            echo "  ❌ $script_name - MODIFICADO"
            echo "ALERTA: Script $script_path modificado" | sudo tee -a
"$INTEGRITY_FILE"
            ((errors++))
        fi
    done < "$HASH_FILE"

    if [ $errors -eq 0 ]; then
        echo "TODOS LOS SCRIPTS ESTAN INTEGROS"
    else
        echo "Se encontraron $errors scripts modificados"
        return 1
    fi
}

function setup_monitoring() {
    echo "Configurando monitoreo continuo..."

    # Crear script de monitoreo
    sudo tee /usr/local/bin/monitor-scripts > /dev/null << 'EOF'
#!/bin/bash
/usr/local/admin-scripts/verify_script_integrity.sh --verify
if [ $? -ne 0 ]; then
```

```
    echo "ALERTA: Scripts modificados detectados en $(hostname) a las $(date)" | \
    mail -s "ALERTA DE SEGURIDAD S.I.G.I.E" root
fi
EOF

    sudo chmod +x /usr/local/bin/monitor-scripts

    # Agregar a crontab para verificación cada hora
    (sudo crontab -l 2>/dev/null; echo "0 * * * * /usr/local/bin/monitor-scripts") | sudo
crontab -

    echo "Monitoreo configurado cada hora"
}

case "$1" in
    "--calculate"|"-c")
        calculate_hashes
        ;;
    "--verify"|"-v")
        verify_hashes
        ;;
    "--monitor"|"-m")
        setup_monitoring
        ;;
    *)
        echo "USO: $0 [OPCION]"
        echo ""
        echo "OPCIONES:"
        echo "  --calculate, -c   Calcular hashes de scripts"
        echo "  --verify, -v      Verificar integridad de scripts"
        echo "  --monitor, -m     Configurar monitoreo continuo"
        echo ""
        echo "EJEMPLO DE FLUJO COMPLETO:"
        echo "  1. $0 --calculate  # Calcular hashes iniciales"
        echo "  2. $0 --monitor    # Configurar monitoreo"
        echo "  3. $0 --verify     # Verificacion manual"
        ;;
esac

# Log de ejecución
echo "$(date): $0 $@" | sudo tee -a "$INTEGRITY_FILE"
```

e) Introducción al uso de fail2ban para protección de servicios como SSH

CONFIGURACIÓN FAIL2BAN: `configure_fail2ban.sh`

```
#!/bin/bash

echo "================================================"
echo "        CONFIGURACION FAIL2BAN"
echo "================================================"

# Instalar Fail2ban
echo "Instalando Fail2ban..."
sudo apt update
sudo apt install -y fail2ban

echo "Configurando Fail2ban..."

# Crear configuración local
sudo tee /etc/fail2ban/jail.local > /dev/null << 'EOF'
# Configuracion Fail2ban S.I.G.I.E
[DEFAULT]
# Direccion de baneo
banaction = iptables-multiport

# Tiempo de baneo por defecto (1 hora)
bantime = 3600

# Ventana de tiempo para intentos fallidos (10 minutos)
findtime = 600

# Intentos maximos antes del baneo
maxretry = 3

# Backend para leer logs
backend = auto

[sshd]
enabled = true
port = 666
logpath = /var/log/auth.log
maxretry = 3
bantime = 86400

[sshd-ddos]
```

```
enabled = true
port = 666
logpath = /var/log/auth.log
maxretry = 5
bantime = 86400

[docker-auth]
enabled = true
port = 2375,2376
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600

[recidive]
enabled = true
logpath = /var/log/fail2ban.log
action = iptables-allports[name=recidive]
bantime = 604800  # 1 semana
findtime = 86400  # 1 dia
maxretry = 5
EOF

# Crear filtro personalizado para SSH
sudo tee /etc/fail2ban/filter.d/sshd-secure.conf > /dev/null << 'EOF'
[Definition]
failregex = ^%(__prefix_line)s(?:error: PAM: )?Authentication failure for .* from
<HOST>\s*$
            ^%(__prefix_line)s(?:error: PAM: )?User not known to the underlying
authentication module for .* from <HOST>\s*$
            ^%(__prefix_line)sFailed publickey for .* from <HOST> port \d+ ssh2\s*$
            ^%(__prefix_line)sFailed password for .* from <HOST> port \d+ ssh2\s*$
            ^%(__prefix_line)sReceived disconnect from <HOST>: 3: .*: Auth fail$
ignoreregex =
EOF

# Iniciar y habilitar servicio
echo "Iniciando servicios Fail2ban..."
sudo systemctl enable fail2ban
sudo systemctl start fail2ban

# Verificar estado
echo "Verificando estado de Fail2ban..."
sudo fail2ban-client status
```

```
# Configurar logrotate para Fail2ban
sudo tee /etc/logrotate.d/fail2ban > /dev/null << 'EOF'
/var/log/fail2ban.log {
    weekly
    missingok
    rotate 4
    compress
    delaycompress
    notifempty
    postrotate
        /usr/bin/systemctl reload fail2ban > /dev/null 2>/dev/null || true
    endscript
}
EOF

# Crear script de monitoreo de Fail2ban
sudo tee /usr/local/bin/fail2ban-status > /dev/null << 'EOF'
#!/bin/bash
echo "ESTADO DE FAIL2BAN S.I.G.I.E"
echo "============================="
echo "Servicios protegidos:"
echo "  SSH (puerto 666)"
echo "  Docker API"
echo ""
echo "Estado general:"
sudo fail2ban-client status
echo ""
echo "IPs baneadas:"
sudo fail2ban-client status sshd | grep "Banned IP list:" -A 10
EOF

sudo chmod +x /usr/local/bin/fail2ban-status

# Probar configuración
echo "Probando configuracion..."
sudo fail2ban-client reload

sleep 2

echo "Estado de los jails:"
sudo fail2ban-client status sshd

echo ""
echo "============================================="
```

```
echo "       FAIL2BAN CONFIGURADO EXITOSAMENTE"
echo "============================================="
echo "SERVICIOS PROTEGIDOS:"
echo "  SSH (puerto 666) - 3 intentos -> 24h baneo"
echo "  SSH-DDOS - 5 intentos -> 24h baneo"
echo "  Docker API - 3 intentos -> 1h baneo"
echo "  Recidive - Reincidentes -> 1 semana baneo"
echo ""
echo "COMANDOS UTILES:"
echo "  fail2ban-status          # Ver estado completo"
echo "  fail2ban-client status   # Estado tecnico"
echo "  fail2ban-client unban <IP>  # Desbanear IP"
```

TERCERA ENTREGA

a) Configuración de Roles y Permisos (Mínimos Privilegios)

Script: `configure_least_privileges.sh`

bash

```
#!/bin/bash

echo "============================================="
echo "  CONFIGURACIÓN DE MÍNIMOS PRIVILEGIOS"
echo "============================================="

# 1. Configurar permisos de archivos críticos
echo "1. Ajustando permisos de archivos del sistema..."

# Archivos sensibles
sudo chmod 644 /etc/passwd
sudo chmod 600 /etc/shadow
sudo chmod 644 /etc/group
sudo chmod 600 /etc/gshadow
sudo chmod 600 /etc/ssh/sshd_config

# 2. Configurar servicios para ejecutar con usuarios no-root
echo "2. Configurando usuarios de servicio..."
```

```
# Crear usuario específico para servicios
sudo useradd -r -s /bin/false service_user
sudo useradd -r -s /bin/false docker_operator

# 3. Configurar sudoers específicos
echo "3. Configurando políticas sudo..."

sudo tee /etc/sudoers.d/sigie_roles > /dev/null << 'EOF'
# Admin Sistemas - Solo comandos específicos
%admin_sis ALL=(ALL) /usr/bin/systemctl, /usr/bin/docker,
/usr/local/bin/docker-compose, /usr/bin/apt, /usr/bin/apt-get

# Admin Seguridad - Solo comandos de seguridad
%admin_sec ALL=(ALL) /usr/sbin/ufw, /usr/sbin/fail2ban-client, /usr/bin/auditctl,
/usr/sbin/ausearch

# Operadores - Comandos limitados
%operadores ALL=(ALL) /usr/local/admin-scripts/menu_usuario.sh,
/usr/local/admin-scripts/backup_sistema.sh

# Docker - Sin acceso sudo completo
%docker ALL=(ALL) /usr/bin/docker, /usr/local/bin/docker-compose
EOF

sudo chmod 440 /etc/sudoers.d/sigie_roles

# 4. Configurar capabilities en lugar de privilegios completos
echo "4. Configurando capabilities..."

# Ejemplo: dar capacidad de binding a puertos sin root
sudo setcap 'cap_net_bind_service=+ep' /usr/bin/docker 2>/dev/null || true

# 5. Configurar umask por defecto más seguro
echo "5. Configurando umask..."

sudo tee /etc/profile.d/secure-umask.sh > /dev/null << 'EOF'
# Umask más seguro para todos los usuarios
if [ "$(id -u)" = 0 ]; then
    umask 022
else
    umask 027
fi
EOF
```

```
echo "============================================="
echo "  CONFIGURACIÓN COMPLETADA"
echo "============================================="
```

b) Configuración de Alertas de Seguridad en Zabbix

Script: `configure_security_alerts.sh`

```
bash
#!/bin/bash

echo "============================================="
echo "  CONFIGURACIÓN ALERTAS DE SEGURIDAD ZABBIX"
echo "============================================="

# Configurar items de seguridad para Zabbix Agent
sudo tee /etc/zabbix/zabbix_agentd.d/security_monitoring.conf > /dev/null << 'EOF'
# Monitoreo de seguridad S.I.G.I.E
UserParameter=auth.failed.attempts,grep "Failed password" /var/log/auth.log | wc -l
UserParameter=auth.invalid.users,grep "Invalid user" /var/log/auth.log | wc -l
UserParameter=auth.sudo.usage,grep "sudo:" /var/log/auth.log | grep -v "NOT in
sudoers" | wc -l
UserParameter=ssh.failed.attempts,grep "Failed password" /var/log/auth.log | grep
"ssh" | wc -l
UserParameter=user.account.changes,grep "usermod\\|useradd\\|userdel"
/var/log/auth.log | wc -l
UserParameter=firewall.denied,ufw status | grep "DENY" | wc -l
UserParameter=fail2ban.banned,fail2ban-client status sshd | grep "Total banned" |
awk '{print $4}'
EOF

# Reiniciar servicio
sudo systemctl restart zabbix-agent

echo "Items de seguridad configurados:"
echo "- auth.failed.attempts: Intentos de login fallidos"
echo "- auth.invalid.users: Intentos con usuarios inválidos"
echo "- auth.sudo.usage: Uso de comandos sudo"
echo "- ssh.failed.attempts: Intentos SSH fallidos"
echo "- user.account.changes: Cambios en cuentas de usuario"
echo "- firewall.denied: Conexiones bloqueadas por firewall"
echo "- fail2ban.banned: IPs baneadas por fail2ban"
```

echo ""
echo "Configuración completada. Las alertas estarán disponibles en Zabbix Web."

c) Introducción a Logs de Seguridad

Documentación: `logs_seguridad.md`
markdown
LOGS DE SEGURIDAD - ANÁLISIS BÁSICO

📋 LOGS PRINCIPALES DE SEGURIDAD

1. /var/log/auth.log
Contiene: Todas las actividades de autenticación del sistema

Eventos importantes:
- Intentos de login (exitosos y fallidos)
- Uso de comandos sudo
- Creación/modificación de usuarios
- Autenticación SSH

2. /var/log/secure (en sistemas RedHat-based)
Equivalente a auth.log en distribuciones basadas en RedHat

🔍 ANÁLISIS BÁSICO DE LOGS

Comandos útiles para análisis:

```bash
# Últimos 10 intentos fallidos de SSH
sudo grep "Failed password" /var/log/auth.log | tail -10

# Intentos de login con usuarios inexistentes
sudo grep "Invalid user" /var/log/auth.log

# Logins exitosos
sudo grep "Accepted" /var/log/auth.log

# Uso de comandos sudo
sudo grep "sudo:" /var/log/auth.log

# Conexiones SSH
```

sudo grep "sshd" /var/log/auth.log

# Intentos de acceso a cuentas específicas
sudo grep "Failed password for root" /var/log/auth.log
sudo grep "Failed password for admin" /var/log/auth.log

Patrones a monitorear:

1. **Múltiples intentos fallidos desde misma IP**
2. **Intentos de acceso a cuenta root**
3. **Actividad en horarios no laborales**
4. **Usuarios inexistentes en intentos de login**
5. **Cambios inesperados en privilegios**

d) Script de Filtrado de Logs Sospechosos

Script: `security_log_analyzer.sh`

```bash
#!/bin/bash

echo "=============================================="
echo "  ANALIZADOR DE LOGS SOSPECHOSOS"
echo "=============================================="

LOG_FILES="/var/log/auth.log /var/log/syslog /var/log/secure"
SUSPICIOUS_KEYWORDS="invalid|fail|error|denied|refused|unauthorized|attack|breach|intrusion|malware|virus|exploit|shell|reverse_tcp"
CRITICAL_KEYWORDS="root.*fail|admin.*fail|sudo.*invalid|password.*changed|useradd|usermod"
```

```
# Colores para output
RED='\033[0;31m'
YELLOW='\033[1;33m'
GREEN='\033[0;32m'
NC='\033[0m'

analyze_auth_log() {
    echo -e "${YELLOW}=== ANALIZANDO AUTH.LOG ===${NC}"

    # Intentos fallidos recientes
    echo -e "\n${RED}INTENTOS FALLIDOS RECIENTES:${NC}"
    sudo grep -i "failed password" /var/log/auth.log | tail -20

    # Usuarios inválidos
    echo -e "\n${RED}USUARIOS INVÁLIDOS:${NC}"
    sudo grep -i "invalid user" /var/log/auth.log | tail -15

    # Actividad sudo
    echo -e "\n${YELLOW}ACTIVIDAD SUDO:${NC}"
    sudo grep "sudo:" /var/log/auth.log | tail -10

    # Conexiones SSH
    echo -e "\n${GREEN}CONEXIONES SSH EXITOSAS:${NC}"
    sudo grep "Accepted" /var/log/auth.log | tail -10
}
```

```
analyze_system_logs() {
    echo -e "${YELLOW}=== ANALIZANDO SYSTEM LOGS ===${NC}"

    # Palabras clave sospechosas
    echo -e "\n${RED}PALABRAS CLAVE SOSPECHOSAS:${NC}"
    for log in $LOG_FILES; do
        if [ -f "$log" ]; then
            echo -e "\nEn $log:"
            sudo grep -E -i "$SUSPICIOUS_KEYWORDS" "$log" | tail -15
        fi
    done
}

check_fail2ban_status() {
    echo -e "${YELLOW}=== ESTADO FAIL2BAN ===${NC}"

    if systemctl is-active --quiet fail2ban; then
        echo -e "${GREEN}Fail2ban activo${NC}"
        sudo fail2ban-client status
        echo -e "\nIPs baneadas:"
        sudo fail2ban-client status sshd | grep "Banned IP" || echo "Ninguna IP
baneada"
    else
        echo -e "${RED}Fail2ban inactivo${NC}"
```

```
    fi
}

check_suspicious_ips() {
    echo -e "${YELLOW}=== IPS SOSPECHOSAS ===${NC}"

    echo -e "\n${RED}IPS CON MÚLTIPLES INTENTOS FALLIDOS:${NC}"
    sudo grep "Failed password" /var/log/auth.log | awk '{print $11}' | sort | uniq -c | sort -nr | head -10

    echo -e "\n${YELLOW}CONEXIONES ACTIVAS:${NC}"
    sudo netstat -tunp | grep ESTABLISHED
}

generate_security_report() {
    local report_file="/var/log/admin-system/security_report_$(date +%Y%m%d).txt"

    {
        echo "=== REPORTE DE SEGURIDAD - $(date) ==="
        echo "Intentos fallidos últimos 7 días:"
        sudo grep "Failed password" /var/log/auth.log | grep "$(date -d '7 days ago' +'%Y-%m-%d')" | wc -l
        echo ""
        echo "IPs con múltiples intentos:"
        sudo grep "Failed password" /var/log/auth.log | awk '{print $11}' | sort | uniq -c | sort -nr | head -5
```

```
        echo ""
        echo "Comandos sudo ejecutados:"
        sudo grep "sudo:" /var/log/auth.log | grep "$(date +'%Y-%m-%d')" | wc -l
    } > "$report_file"

    echo -e "${GREEN}Reporte generado: $report_file${NC}"
}

# Menú principal
while true; do
    echo ""
    echo "1. Analizar auth.log"
    echo "2. Analizar system logs"
    echo "3. Ver estado fail2ban"
    echo "4. Ver IPs sospechosas"
    echo "5. Generar reporte diario"
    echo "6. Salir"
    read -p "Seleccione opción: " choice

    case $choice in
        1) analyze_auth_log ;;
        2) analyze_system_logs ;;
        3) check_fail2ban_status ;;
        4) check_suspicious_ips ;;
        5) generate_security_report ;;
```

```
        6) break ;;
        *) echo "Opción inválida" ;;
    esac

    read -p "Presione Enter para continuar..."
done
```

e) Revisión de Dockerfile para Seguridad

Script: `dockerfile_security_audit.sh`

```
bash
#!/bin/bash

echo "=============================================="
echo "  AUDITORÍA DE SEGURIDAD DOCKERFILE"
echo "=============================================="

DOCKERFILE=${1:-"Dockerfile"}

if [ ! -f "$DOCKERFILE" ]; then
    echo "Error: No se encontró $DOCKERFILE"
    exit 1
fi

echo "Analizando: $DOCKERFILE"
echo ""

# 1. Verificar uso de usuario root
echo "1. VERIFICACIÓN DE USUARIO:"
if grep -q "USER root" "$DOCKERFILE"; then
    echo "    ALERTA: Usa USER root explícitamente"
elif ! grep -q "USER" "$DOCKERFILE"; then
    echo "    ALERTA: No define USER (por defecto usa root)"
else
    echo "    OK: Define usuario no-root"
    grep "USER" "$DOCKERFILE"
fi
```

```
# 2. Verificar versión de imagen base
echo ""
echo "2. IMAGEN BASE:"
if grep -q "FROM.*:latest" "$DOCKERFILE"; then
    echo "     ADVERTENCIA: Usa 'latest' tag (usar versión específica)"
else
    echo "   OK: Usa versión específica"
    grep "FROM" "$DOCKERFILE"
fi

# 3. Verificar actualización de paquetes
echo ""
echo "3. ACTUALIZACIONES:"
if grep -q "apt-get update && apt-get upgrade" "$DOCKERFILE"; then
    echo "    OK: Actualiza paquetes"
elif grep -q "apt-get update" "$DOCKERFILE"; then
    echo "     ADVERTENCIA: Actualiza lista pero no paquetes"
else
    echo "    ALERTA: No actualiza paquetes"
fi

# 4. Verificar limpieza de cache
echo ""
echo "4. LIMPIEZA:"
if grep -q "apt-get clean" "$DOCKERFILE" || grep -q "rm -rf /var/lib/apt/lists/"
"$DOCKERFILE"; then
    echo "    OK: Limpia cache"
else
    echo "     ADVERTENCIA: No limpia cache"
fi

# 5. Verificar puertos expuestos
echo ""
echo "5. PUERTOS:"
if grep -q "EXPOSE" "$DOCKERFILE"; then
    echo "   Puertos expuestos:"
    grep "EXPOSE" "$DOCKERFILE"
else
    echo "    OK: No expone puertos innecesarios"
fi

# 6. Verificar healthcheck
echo ""
echo "6. HEALTHCHECK:"
```

```
if grep -q "HEALTHCHECK" "$DOCKERFILE"; then
    echo "    OK: Tiene healthcheck"
else
    echo "     ADVERTENCIA: No tiene healthcheck"
fi

# 7. Verificar secrets en Dockerfile
echo ""
echo "7. SECRETS:"
if grep -q "ARG.*password" "$DOCKERFILE" || grep -q "ARG.*secret"
"$DOCKERFILE"; then
    echo "    ALERTA: Posibles secrets en Dockerfile"
else
    echo "    OK: No contiene secrets evidentes"
fi

# 8. Verificar tamaño de imagen
echo ""
echo "8. RECOMENDACIONES:"
echo "   - Usar imágenes base minimales (alpine, slim)"
echo "   - Combinar RUN commands para reducir layers"
echo "   - Usar .dockerignore para excluir archivos innecesarios"
echo "   - Escanear imagen con trivy/docker scan"

echo ""
echo "================================================"
echo "  AUDITORÍA COMPLETADA"
echo "================================================"
```

f) Revisión de Actualizaciones del Sistema

Script: `system_security_updates.sh`

```
bash
#!/bin/bash

echo "================================================"
echo "  REVISIÓN DE ACTUALIZACIONES DE SEGURIDAD"
echo "================================================"

# Colores
RED='\033[0;31m'
YELLOW='\033[1;33m'
GREEN='\033[0;32m'
```

```
NC='\033[0m'

check_system_updates() {
    echo -e "${YELLOW}1. VERIFICANDO ACTUALIZACIONES
DISPONIBLES...${NC}"

    sudo apt update > /dev/null 2>&1

    # Actualizaciones de seguridad
    SECURITY_UPDATES=$(sudo apt list --upgradable 2>/dev/null | grep -i security |
wc -l)
    ALL_UPDATES=$(sudo apt list --upgradable 2>/dev/null | wc -l)

    if [ "$SECURITY_UPDATES" -gt 0 ]; then
        echo -e "${RED} Hay $SECURITY_UPDATES actualizaciones de seguridad
pendientes${NC}"
        echo "Actualizaciones de seguridad disponibles:"
        sudo apt list --upgradable | grep -i security
    else
        echo -e "${GREEN} No hay actualizaciones de seguridad pendientes${NC}"
    fi

    if [ "$ALL_UPDATES" -gt 0 ]; then
        echo -e "${YELLOW}  Hay $((ALL_UPDATES-1)) actualizaciones generales
pendientes${NC}"
    fi
}

check_automatic_updates() {
    echo -e "\n${YELLOW}2. VERIFICANDO ACTUALIZACIONES
AUTOMÁTICAS...${NC}"

    if systemctl is-active --quiet unattended-upgrades; then
        echo -e "${GREEN} Actualizaciones automáticas activas${NC}"
        echo "Configuración:"
        sudo cat /etc/apt/apt.conf.d/20auto-upgrades | grep -E "APT::Periodic::"
    else
        echo -e "${RED} Actualizaciones automáticas inactivas${NC}"
    fi
}

check_vulnerable_packages() {
    echo -e "\n${YELLOW}3. BUSCANDO PAQUETES CON
VULNERABILIDADES...${NC}"
```

```
    # Instalar aptitude si no está presente
    if ! command -v aptitude >/dev/null 2>&1; then
        echo "Instalando aptitude para análisis de vulnerabilidades..."
        sudo apt install -y aptitude
    fi

    # Buscar paquetes con vulnerabilidades conocidas
    VULNERABLE=$(aptitude search '~U ~i' 2>/dev/null | wc -l)

    if [ "$VULNERABLE" -gt 0 ]; then
        echo -e "${RED} Se encontraron paquetes con vulnerabilidades${NC}"
        aptitude search '~U ~i'
    else
        echo -e "${GREEN} No se encontraron paquetes vulnerables${NC}"
    fi
}

check_service_versions() {
    echo -e "\n${YELLOW}4. VERSIONES DE SERVICIOS CRÍTICOS...${NC}"

    # SSH
    if command -v sshd >/dev/null 2>&1; then
        SSH_VERSION=$(sshd -V 2>&1 | head -n1)
        echo "SSH: $SSH_VERSION"
    fi

    # Docker
    if command -v docker >/dev/null 2>&1; then
        DOCKER_VERSION=$(docker --version)
        echo "Docker: $DOCKER_VERSION"
    fi

    # Nginx
    if command -v nginx >/dev/null 2>&1; then
        NGINX_VERSION=$(nginx -v 2>&1)
        echo "Nginx: $NGINX_VERSION"
    fi
}

check_kernel_security() {
    echo -e "\n${YELLOW}5. VERIFICANDO SEGURIDAD DEL KERNEL...${NC}"

    # Verificar si kernel necesita actualización
```

```
    CURRENT_KERNEL=$(uname -r)
    INSTALLED_KERNEL=$(dpkg -l | grep linux-image | grep ii | awk '{print $3}' | sort
-V | tail -1)

    echo "Kernel actual: $CURRENT_KERNEL"
    echo "Último kernel instalado: $INSTALLED_KERNEL"

    if [ "$CURRENT_KERNEL" != "$INSTALLED_KERNEL" ]; then
        echo -e "${YELLOW}  Kernel no actualizado, requiere reinicio${NC}"
    fi

    # Verificar configuraciones de seguridad del kernel
    echo -e "\nConfiguraciones de seguridad:"
    if [ "$(cat /proc/sys/kernel/yama/ptrace_scope)" = "1" ]; then
        echo -e " ptrace_scope: restringido"
    else
        echo -e "  ptrace_scope: permisivo"
    fi
}

apply_security_updates() {
    echo -e "\n${YELLOW}6. APLICAR ACTUALIZACIONES DE
SEGURIDAD...${NC}"

    read -p "¿Aplicar actualizaciones de seguridad? (s/N): " choice
    if [[ $choice == "s" || $choice == "S" ]]; then
        echo "Aplicando actualizaciones de seguridad..."
        sudo apt upgrade --only-upgrade -y
        echo -e "${GREEN} Actualizaciones aplicadas${NC}"

        # Verificar si se requiere reinicio
        if [ -f /var/run/reboot-required ]; then
            echo -e "${YELLOW}  Se requiere reinicio del sistema${NC}"
            read -p "¿Reiniciar ahora? (s/N): " reboot_choice
            if [[ $reboot_choice == "s" || $reboot_choice == "S" ]]; then
                sudo reboot
            fi
        fi
    else
        echo "Actualizaciones pendientes."
    fi
}

# Ejecutar verificaciones
```

```
check_system_updates
check_automatic_updates
check_vulnerable_packages
check_service_versions
check_kernel_security
apply_security_updates

echo -e
"\n${GREEN}================================================${NC}"
echo -e "${GREEN}  REVISIÓN COMPLETADA${NC}"
echo -e
"${GREEN}================================================${NC}"
```