# TAREA 3 - PLAN DE MITIGACIÓN COMPLETO

## Contenido adicional para integrar en el informe principal (Sección 4)

## 4.4 ESTRATEGIA COMPLETA DE BACKUPS Y RECUPERACIÓN

### 4.4.1 Política de Backups Detallada

***Tipos de Backup***

**A) Backup Completo (Full Backup)**

- **Frecuencia:** Domingo 02:00 AM
- **Contenido:**
  - Todas las bases de datos (MySQL)
  - Directorio completo de aplicación web (`/var/www/html`)
  - Archivos de configuración del sistema
  - Logs críticos de seguridad
  - Reglas de firewall y Suricata
- **Método:** Copia bit a bit de todos los datos
- **Ventaja:** Restauración completa y rápida
- **Desventaja:** Mayor consumo de espacio
- **Tiempo estimado:** 30-45 minutos
- **Tamaño aproximado:** 5-10 GB

**B) Backup Incremental**

- **Frecuencia:** Lunes a Sábado, cada 6 horas (00:00, 06:00, 12:00, 18:00)
- **Contenido:** Solo archivos modificados desde el último backup
- **Método:** Comparación de timestamps
- **Ventaja:** Rápido, poco espacio
- **Desventaja:** Restauración más compleja (requiere full + todos los incrementales)
- **Tiempo estimado:** 5-10 minutos
- **Tamaño aproximado:** 100-500 MB

**C) Backup de Configuración (Pre-Change)**

- **Frecuencia:** Antes de CUALQUIER cambio en producción
- **Contenido:**
  - Configuraciones de Apache/Nginx
  - Configuraciones de MySQL
  - Configuraciones de PHP
  - Configuraciones de SSH
  - Reglas de firewall (iptables)

    - Configuración de Suricata
- **Método:** Snapshot de configuración
- **Script automatizado:**

```
#!/bin/bash

# pre-change-backup.sh

DATE=$(date +%Y%m%d_%H%M%S)

CHANGE_ID="$1"  # ID del ticket de cambio


if [ -z "$CHANGE_ID" ]; then

    echo "Uso: $0 <CHANGE_ID>"

    exit 1

fi


BACKUP_DIR="/backups/pre-change/$CHANGE_ID"

mkdir -p "$BACKUP_DIR"


# Backup de configuraciones

tar -czf "$BACKUP_DIR/config_$DATE.tar.gz" \

    /etc/apache2 \

    /etc/mysql \

    /etc/php \

    /etc/ssh \

    /etc/suricata


# Documentar cambio

cat > "$BACKUP_DIR/change_info.txt" << EOF

Change ID: $CHANGE_ID

Date: $(date)

User: $(whoami)

Hostname: $(hostname)

Description: [A completar manualmente]

EOF
```

```
echo "Backup pre-cambio guardado en: $BACKUP_DIR"
```

### *Ubicaciones de Almacenamiento*

#### Almacenamiento Primario

- **Ubicación:** `/backups` (disco local dedicado)
- **Tipo:** SSD de alta velocidad
- **Capacidad:** 500 GB
- **Retención local:**
  - Backups diarios: 30 días
  - Backups semanales: 12 semanas
  - Backups mensuales: 12 meses

#### Almacenamiento Secundario (Offsite)

- **Ubicación:** Servidor remoto (NAS/Cloud)
- **Protocolo:** rsync over SSH
- **Frecuencia de sincronización:** Diaria 04:00 AM
- **Encriptación:** AES-256
- **Script de sincronización:**

```
#!/bin/bash

# sync-offsite.sh

REMOTE_USER="backup"

REMOTE_HOST="backup.company.local"

REMOTE_PATH="/mnt/backups/production"

LOCAL_PATH="/backups"


rsync -avz --delete \

    -e "ssh -i /root/.ssh/backup_key" \

    "$LOCAL_PATH/" \

    "$REMOTE_USER@$REMOTE_HOST:$REMOTE_PATH/"


if [ $? -eq 0 ]; then

    logger "Backup offsite sincronizado correctamente"

else

    logger "ERROR: Fallo en sincronización offsite"

    # Enviar alerta
```

```
    echo "Error en backup offsite" | mail -s "ALERTA BACKUP"
admin@company.com

fi
```

#### Almacenamiento Terciario (Opcional)

- **Ubicación:** Nube pública (AWS S3 / Google Cloud Storage)
- **Frecuencia:** Mensual (primer domingo del mes)
- **Contenido:** Solo backups full mensuales
- **Retención:** 24 meses (2 años)
- **Costo estimado:** $20-50/mes

## 4.4.2 Automatización de Backups

### *Configuración de Cron Jobs*

Añadir al crontab del usuario root (`sudo crontab -e`):

```
# ======================================================================

# BACKUPS AUTOMATIZADOS - PAI-4

# ======================================================================


# Backup completo - Domingos 02:00 AM

0 2 * * 0 /opt/scripts/backup-system.sh full >> /var/log/backup-full.log 2>&1


# Backups incrementales - Lunes a Sábado, cada 6 horas

0 */6 * * 1-6 /opt/scripts/backup-system.sh incremental >> /var/log/backup-
inc.log 2>&1


# Sincronización offsite - Diario 04:00 AM

0 4 * * * /opt/scripts/sync-offsite.sh >> /var/log/backup-sync.log 2>&1


# Verificación de integridad - Diario 05:00 AM

0 5 * * * /opt/scripts/verify-backups.sh >> /var/log/backup-verify.log 2>&1


# Limpieza de backups antiguos - Diario 06:00 AM

0 6 * * * find /backups -name "*.gz" -mtime +30 -delete


# Reporte mensual de backups - Primer día del mes 08:00 AM
```

```
0 8 1 * * /opt/scripts/backup-report.sh | mail -s "Backup Monthly Report"
admin@company.com
```

### *Script de Verificación de Integridad*

```
#!/bin/bash

# verify-backups.sh


BACKUP_DIR="/backups"

LOG_FILE="/var/log/backup-verify.log"

ERRORS=0


log() {

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

}


log "========================================="

log "VERIFICACIÓN DE INTEGRIDAD DE BACKUPS"

log "========================================="


# Verificar existencia de backups recientes

LATEST_FULL=$(find "$BACKUP_DIR" -name "mysql_*.sql.gz" -mtime -1 | head -n1)

if [ -z "$LATEST_FULL" ]; then

    log "ERROR: No se encontró backup full reciente (últimas 24h)"

    ERRORS=$((ERRORS + 1))

else

    log "OK: Backup full encontrado: $(basename $LATEST_FULL)"

fi


# Verificar checksums MD5

LATEST_MD5=$(find "$BACKUP_DIR" -name "checksums_*.md5" -mtime -1 | head -n1)

if [ -n "$LATEST_MD5" ]; then

    cd "$BACKUP_DIR"
```

```
    if md5sum -c "$LATEST_MD5" --quiet; then
        log "OK: Checksums MD5 verificados correctamente"
    else
        log "ERROR: Fallos en verificación de checksums"
        ERRORS=$((ERRORS + 1))
    fi
fi

# Verificar espacio en disco
DISK_USAGE=$(df -h "$BACKUP_DIR" | tail -1 | awk '{print $5}' | sed 's/%//')
if [ "$DISK_USAGE" -gt 90 ]; then
    log "WARNING: Uso de disco alto: $DISK_USAGE%"
    ERRORS=$((ERRORS + 1))
else
    log "OK: Uso de disco: $DISK_USAGE%"
fi

# Verificar permisos
if [ ! -w "$BACKUP_DIR" ]; then
    log "ERROR: Sin permisos de escritura en $BACKUP_DIR"
    ERRORS=$((ERRORS + 1))
fi

# Resumen
log "========================================"
if [ $ERRORS -eq 0 ]; then
    log "RESULTADO: TODOS LOS CHECKS PASADOS"
    exit 0
else
    log "RESULTADO: $ERRORS ERRORES DETECTADOS"
    # Enviar alerta
```

```
    echo "Errores en verificación de backups" | mail -s "ALERTA BACKUP"
admin@company.com

    exit 1

fi
```

### 4.4.3 Manual Completo de Restauración

***Escenario 1: Restauración de Base de Datos***

**Caso de uso:** Corrupción de datos, borrado accidental, o compromiso de integridad.

**Pasos:**

1. **Preparación (2-3 minutos)**

```
# Detener servicios que usan MySQL

systemctl stop apache2

systemctl stop nginx


# Verificar estado de MySQL

systemctl status mysql


# Crear backup del estado actual (por precaución)

mysqldump --all-databases > /tmp/pre-restore-backup.sql
```

1. **Identificar backup a restaurar (1-2 minutos)**

```
# Listar backups disponibles

ls -lht /backups/mysql_*.sql.gz | head -10


# Verificar integridad del backup seleccionado

BACKUP_FILE="/backups/mysql_20251110_020000.sql.gz"

md5sum "$BACKUP_FILE"

# Comparar con checksums_20251110_020000.md5
```

1. **Restaurar base de datos (5-15 minutos)**

```
# Descomprimir y restaurar
```

```
gunzip < "$BACKUP_FILE" | mysql -u root -p

# O si prefieres mantener el .gz:
zcat "$BACKUP_FILE" | mysql -u root -p
```

1. **Verificación (3-5 minutos)**

```
# Conectar a MySQL
mysql -u root -p

# Verificar bases de datos
SHOW DATABASES;

# Verificar tablas de la aplicación
USE dvwa;
SHOW TABLES;

# Verificar integridad
SELECT COUNT(*) FROM users;

# Verificar permisos
SELECT user, host FROM mysql.user;
```

1. **Reiniciar servicios (1-2 minutos)**

```
systemctl start mysql
systemctl status mysql

systemctl start apache2
systemctl status apache2
```

1. **Prueba funcional (5 minutos)**

```
# Probar login en aplicación
```

```
curl -v http://localhost/login.php


# Verificar logs

tail -f /var/log/mysql/error.log

tail -f /var/log/apache2/error.log
```

**Tiempo total estimado:** 20-30 minutos

**Checklist de verificación:**

- [ ] MySQL corriendo sin errores
- [ ] Todas las bases de datos presentes
- [ ] Usuarios y permisos correctos
- [ ] Aplicación web funcional
- [ ] Logs sin errores críticos
- [ ] Conexiones de aplicación funcionan

***Escenario 2: Restauración de Aplicación Web***
**Caso de uso:** Archivos comprometidos, defacement, o archivos corruptos.

**Pasos:**

1. **Modo mantenimiento (1 minuto)**

```
# Crear página de mantenimiento

cat > /var/www/html/maintenance.html << EOF

<!DOCTYPE html>

<html>

<head><title>Mantenimiento</title></head>

<body>

    <h1>Sistema en Mantenimiento</h1>

    <p>Estaremos de vuelta pronto.</p>

</body>

</html>

EOF


# Redirigir todo el tráfico

cat > /etc/apache2/sites-available/000-default.conf << EOF
```

```
<VirtualHost *:80>
    DocumentRoot /var/www/html
    RewriteEngine On
    RewriteCond %{REQUEST_URI} !^/maintenance.html$
    RewriteRule ^(.*)$ /maintenance.html [L]
</VirtualHost>
EOF

systemctl reload apache2
```

1. **Backup del estado actual (2-3 minutos)**

```
# Mover aplicación comprometida a cuarentena
QUARANTINE_DIR="/var/quarantine/$(date +%Y%m%d_%H%M%S)"
mkdir -p "$QUARANTINE_DIR"
mv /var/www/html "$QUARANTINE_DIR/"
mkdir /var/www/html
```

1. **Restaurar desde backup (3-5 minutos)**

```
# Identificar backup limpio (anterior al compromiso)
BACKUP_FILE="/backups/webapp_20251109_020000.tar.gz"

# Extraer
tar -xzf "$BACKUP_FILE" -C /var/www/

# Verificar permisos
chown -R www-data:www-data /var/www/html
chmod -R 755 /var/www/html
find /var/www/html -type f -exec chmod 644 {} \;
```

1. **Aplicar parches de seguridad (10-20 minutos)**

```
# Actualizar dependencias
```

```
cd /var/www/html

composer update --no-dev  # Si usa Composer


# Aplicar parches críticos identificados

# (Basado en análisis de vulnerabilidades - Tarea 2)


# Ejemplo: Parchear SQL Injection

sed -i 's/$query = "SELECT \* FROM users WHERE id = .*"/$stmt =
$mysqli->prepare("SELECT * FROM users WHERE id = ?")/' login.php
```

1. **Configuración de seguridad adicional (5-10 minutos)**

```
# .htaccess de seguridad

cat > /var/www/html/.htaccess << EOF

# Protección contra directory listing

Options -Indexes


# Protección de archivos sensibles

<FilesMatch "\.(htaccess|htpasswd|ini|log|sh|inc|bak)$">

    Order Allow,Deny

    Deny from all

</FilesMatch>


# Headers de seguridad

Header set X-Content-Type-Options "nosniff"

Header set X-Frame-Options "SAMEORIGIN"

Header set X-XSS-Protection "1; mode=block"

EOF


# Eliminar archivos temporales y de backup

find /var/www/html -name "*.bak" -delete

find /var/www/html -name "*.old" -delete

find /var/www/html -name "*~" -delete
```

1. **Salir de modo mantenimiento (1 minuto)**

```
# Restaurar configuración original de Apache

systemctl reload apache2


# Eliminar página de mantenimiento

rm /var/www/html/maintenance.html
```

1. **Verificación exhaustiva (10-15 minutos)**

```
# Escanear en busca de webshells

grep -r "eval(" /var/www/html/

grep -r "base64_decode" /var/www/html/

grep -r "system(" /var/www/html/

grep -r "exec(" /var/www/html/


# Verificar integridad con checksums

find /var/www/html -type f -exec md5sum {} \; > /tmp/post-restore-
checksums.txt


# Prueba funcional completa

curl -I http://localhost/

curl -I http://localhost/login.php

curl -I http://localhost/setup.php
```

**Tiempo total estimado:** 35-60 minutos

**Checklist de verificación:**

- [ ] Aplicación restaurada completamente
- [ ] Permisos correctos (www-data)
- [ ] Sin archivos sospechosos (webshells)
- [ ] Headers de seguridad configurados
- [ ] Todas las páginas responden correctamente
- [ ] Logs sin errores
- [ ] Parches de seguridad aplicados

***Escenario 3: Restauración Completa del Sistema***

**Caso de uso:** Compromiso total del servidor, ransomware, o fallo de hardware.

**Requisitos previos:**

- Servidor nuevo o reformateado
- Sistema operativo base instalado (Ubuntu 20.04 LTS)
- Acceso root
- Backups disponibles (local u offsite)

**Pasos:**

1. **Instalación base del sistema (30-45 minutos)**

```
# Actualizar sistema
apt update && apt upgrade -y

# Instalar dependencias básicas
apt install -y apache2 mysql-server php libapache2-mod-php \
    php-mysql php-curl php-gd php-mbstring php-xml \
    openssh-server ufw fail2ban

# Configurar firewall básico
ufw allow 22/tcp
ufw allow 80/tcp
ufw allow 443/tcp
ufw enable
```

1. **Recuperar backups desde offsite (15-30 minutos)**

```
# Si los backups están en servidor remoto
REMOTE_USER="backup"
REMOTE_HOST="backup.company.local"
REMOTE_PATH="/mnt/backups/production"

mkdir -p /backups
rsync -avz --progress \
    -e "ssh -i /root/.ssh/backup_key" \
```

```
    "$REMOTE_USER@$REMOTE_HOST:$REMOTE_PATH/" \
    /backups/

# Verificar integridad
cd /backups
md5sum -c checksums_*.md5
```

1. **Restaurar configuraciones del sistema (10-15 minutos)**

```
# Extraer configuraciones
LATEST_CONFIG=$(ls -t /backups/config_*.tar.gz | head -n1)
tar -xzf "$LATEST_CONFIG" -C /

# Restaurar permisos críticos
chmod 600 /etc/ssh/sshd_config
chmod 644 /etc/apache2/apache2.conf
chmod 644 /etc/mysql/mysql.conf.d/mysqld.cnf

# Reiniciar servicios
systemctl restart sshd
systemctl restart apache2
systemctl restart mysql
```

1. **Restaurar base de datos (10-20 minutos)**

```
# Ver pasos detallados en "Escenario 1: Restauración de Base de Datos"
LATEST_DB=$(ls -t /backups/mysql_*.sql.gz | head -n1)
zcat "$LATEST_DB" | mysql -u root -p
```

1. **Restaurar aplicación web (10-15 minutos)**

```
# Ver pasos detallados en "Escenario 2: Restauración de Aplicación Web"
LATEST_WEB=$(ls -t /backups/webapp_*.tar.gz | head -n1)
tar -xzf "$LATEST_WEB" -C /var/www/
```

```
chown -R www-data:www-data /var/www/html
```

1. **Hardening post-restauración (20-30 minutos)**

```
# Cambiar TODAS las contraseñas
passwd root
mysql -e "ALTER USER 'root'@'localhost' IDENTIFIED BY 'NewSecurePassword!';"

# Generar nuevas claves SSH
rm /etc/ssh/ssh_host_*
dpkg-reconfigure openssh-server

# Actualizar y aplicar parches
apt update && apt upgrade -y

# Configurar fail2ban
systemctl enable fail2ban
systemctl start fail2ban

# Instalar y configurar Suricata
# (Ver scripts de instalación en el proyecto)
```

1. **Validación completa del sistema (30-45 minutos)**

```
# Verificar servicios
systemctl status apache2
systemctl status mysql
systemctl status ssh

# Verificar conectividad
ping -c 4 google.com

# Verificar aplicación
```

```
curl -I http://localhost/


# Verificar MySQL

mysql -u root -p -e "SHOW DATABASES;"


# Verificar logs

tail -100 /var/log/syslog

tail -100 /var/log/apache2/error.log

tail -100 /var/log/mysql/error.log


# Verificar seguridad

# Ejecutar escaneo con OpenVAS (ver Tarea 1)

# Verificar reglas de Suricata activas (ver Tarea 5)
```

**Tiempo total estimado:** 2.5 - 4 horas

**Checklist de verificación completa:**

- [ ] Sistema operativo actualizado
- [ ] Todos los servicios corriendo
- [ ] Base de datos restaurada y funcional
- [ ] Aplicación web restaurada y funcional
- [ ] Configuraciones de seguridad aplicadas
- [ ] Firewall configurado
- [ ] Fail2ban activo
- [ ] Suricata IDS desplegado
- [ ] Todas las contraseñas cambiadas
- [ ] Nuevas claves SSH generadas
- [ ] Sin errores en logs
- [ ] Pruebas funcionales exitosas
- [ ] Escaneo de vulnerabilidades realizado

### 4.4.4 Tabla de Tiempos de Recuperación (RTO/RPO)

| Escenario | RTO (Recovery Time Objective) | RPO (Recovery Point Objective) | Prioridad |
|---|---|---|---|
| Fallo de base de datos | 30 minutos | 6 horas (último incremental) | CRÍTICA |
| Compromiso de aplicación web | 1 hora | 6 horas | ALTA |
| Corrupción de configuración | 15 minutos | 0 (pre-change backup) | MEDIA |

| | | | |
|---|---|---|---|
| Fallo completo del servidor | 4 horas | 24 horas (último full) | ALTA |
| Pérdida de datos por ransomware | 6 horas | 24 horas | CRÍTICA |

**Definiciones:**

- **RTO:** Tiempo máximo aceptable para restaurar el servicio
- **RPO:** Pérdida máxima aceptable de datos (tiempo desde último backup)

# 4.5 CONFIGURACIÓN DE LOGS Y ANÁLISIS FORENSE

## 4.5.1 Configuración Centralizada de Logs

***Rsyslog - Servidor Central de Logs***

**Instalación del servidor de logs (máquina separada recomendada):**

```
# En servidor de logs central

apt install -y rsyslog


# Configurar como receptor

cat >> /etc/rsyslog.conf << EOF


# Habilitar recepción UDP

module(load="imudp")

input(type="imudp" port="514")


# Habilitar recepción TCP (más confiable)

module(load="imtcp")

input(type="imtcp" port="514")


# Template para organizar logs por host

\$template RemoteLogs,"/var/log/remote/%HOSTNAME%/%PROGRAMNAME%.log"

*.* ?RemoteLogs

& stop

EOF


systemctl restart rsyslog
```

```
# Abrir firewall
ufw allow 514/tcp
ufw allow 514/udp
```

### Configuración de clientes (servidor de aplicación):

```
# En el servidor de aplicación
cat >> /etc/rsyslog.conf << EOF

# Enviar todos los logs al servidor central
*.* @@log-server.company.local:514
EOF

systemctl restart rsyslog
```

### *Logs Críticos a Monitorizar*

#### 1. Logs de Apache

### Configuración en `/etc/apache2/apache2.conf`:

```
# Log Format personalizado con info de seguridad
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\" %T %D" detailed

# Logs por virtualhost
<VirtualHost *:80>
    ServerName dvwa.local

    # Log de accesos
    CustomLog /var/log/apache2/dvwa_access.log detailed

    # Log de errores (nivel debug para análisis)
    LogLevel warn
    ErrorLog /var/log/apache2/dvwa_error.log
```

```
    # Log de seguridad (opcional con mod_security)
    SecAuditLog /var/log/apache2/modsec_audit.log
</VirtualHost>
```

**Rotación de logs:**

```
# /etc/logrotate.d/apache2
/var/log/apache2/*.log {
    daily
    rotate 30
    missingok
    notifempty
    compress
    delaycompress
    create 640 root adm
    sharedscripts
    postrotate
        if systemctl is-active --quiet apache2; then
            systemctl reload apache2 > /dev/null 2>&1
        fi
    endscript
}
```

#### 2. Logs de MySQL

**Configuración en `/etc/mysql/mysql.conf.d/mysqld.cnf`:**

```
[mysqld]

# Log general de queries (solo para debugging, alto overhead)
# general_log = 1
# general_log_file = /var/log/mysql/general.log

# Log de queries lentas (>2 segundos)
```

```
slow_query_log = 1
slow_query_log_file = /var/log/mysql/slow-queries.log
long_query_time = 2

# Log de errores
log_error = /var/log/mysql/error.log

# Log binario (para replicación y recuperación point-in-time)
log_bin = /var/log/mysql/mysql-bin.log
expire_logs_days = 7
max_binlog_size = 100M

# Log de conexiones fallidas
log_warnings = 2
```

#### 3. Logs de Autenticación (SSH, Sudo)

```
# Ya configurado por defecto en /var/log/auth.log

# Monitorizar intentos fallidos
grep "Failed password" /var/log/auth.log

# Monitorizar conexiones exitosas
grep "Accepted publickey" /var/log/auth.log

# Monitorizar uso de sudo
grep "sudo:" /var/log/auth.log
```

#### 4. Logs de Suricata IDS

**Configuración en `/etc/suricata/suricata.yaml`:**

```
outputs:
  - fast:
      enabled: yes
```

```
    filename: fast.log


  - eve-log:
      enabled: yes
      filetype: regular
      filename: eve.json
      types:
        - alert:
            payload: yes
            packet: yes
            metadata: yes
        - http:
            extended: yes
        - dns:
            enabled: yes
        - tls:
            enabled: yes
        - files:
            enabled: yes
        - ssh:
            enabled: yes


  - stats:
      enabled: yes
      filename: stats.log
      interval: 60
```

## 4.5.2 Análisis Automatizado de Logs

***Script de Análisis Diario***

```
#!/bin/bash

# /opt/scripts/analyze-security-logs.sh
```

```
DATE=$(date +%Y-%m-%d)

REPORT="/var/log/security-reports/daily_$DATE.txt"

mkdir -p /var/log/security-reports


cat > "$REPORT" << EOF

========================================

REPORTE DE SEGURIDAD DIARIO

Fecha: $DATE

========================================


EOF


# ================================================================================

# 1. ANÁLISIS DE INTENTOS DE AUTENTICACIÓN

# ================================================================================

echo "1. INTENTOS DE AUTENTICACIÓN SSH" >> "$REPORT"

echo "----------------------------------------" >> "$REPORT"


# Intentos fallidos

FAILED_SSH=$(grep "Failed password" /var/log/auth.log | grep "$(date
+%b\ %d)" | wc -l)

echo "Total de intentos fallidos: $FAILED_SSH" >> "$REPORT"


if [ $FAILED_SSH -gt 10 ]; then

    echo "⚠  ALERTA: Más de 10 intentos fallidos detectados" >> "$REPORT"


    # Top 5 IPs atacantes

    echo "" >> "$REPORT"

    echo "Top 5 IPs con intentos fallidos:" >> "$REPORT"

    grep "Failed password" /var/log/auth.log | grep "$(date +%b\ %d)" | \

        grep -oP 'from \K[0-9.]+' | sort | uniq -c | sort -rn | head -5 >>
"$REPORT"
```

```
fi

# ============================================================================

# 2. ANÁLISIS DE ACCESOS WEB SOSPECHOSOS

# ============================================================================

echo "" >> "$REPORT"

echo "2. ACCESOS WEB SOSPECHOSOS" >> "$REPORT"

echo "----------------------------------------" >> "$REPORT"


# Buscar patrones de ataque común

SQL_INJECTION=$(grep -i "select.*from\|union.*select\|'\s*or\s*'1'='1"
/var/log/apache2/*access.log | wc -l)

XSS_ATTEMPTS=$(grep -i "<script\|javascript:\|onerror="
/var/log/apache2/*access.log | wc -l)

PATH_TRAVERSAL=$(grep -i "\.\./\|\.\.%2f" /var/log/apache2/*access.log | wc -
l)

WEBSHELL=$(grep -i "c99\|r57\|shell\|cmd.php" /var/log/apache2/*access.log |
wc -l)


echo "Intentos de SQL Injection: $SQL_INJECTION" >> "$REPORT"

echo "Intentos de XSS: $XSS_ATTEMPTS" >> "$REPORT"

echo "Intentos de Path Traversal: $PATH_TRAVERSAL" >> "$REPORT"

echo "Búsqueda de Webshells: $WEBSHELL" >> "$REPORT"


if [ $((SQL_INJECTION + XSS_ATTEMPTS + PATH_TRAVERSAL + WEBSHELL)) -gt 20 ];
then

    echo "⚠  ALERTA: Alto número de intentos de ataque web" >> "$REPORT"

fi


# ============================================================================

# 3. ANÁLISIS DE ERRORES CRÍTICOS

# ============================================================================

echo "" >> "$REPORT"
```

```
echo "3. ERRORES CRÍTICOS DEL SISTEMA" >> "$REPORT"

echo "----------------------------------------" >> "$REPORT"


# Errores de Apache

APACHE_ERRORS=$(grep -i "error\|critical\|alert\|emergency"
/var/log/apache2/error.log | grep "$(date +%b\ %d)" | wc -l)

echo "Errores de Apache: $APACHE_ERRORS" >> "$REPORT"


# Errores de MySQL

MYSQL_ERRORS=$(grep -i "error\|crash\|corrupt" /var/log/mysql/error.log |
grep "$(date +%Y-%m-%d)" | wc -l)

echo "Errores de MySQL: $MYSQL_ERRORS" >> "$REPORT"


# ============================================================================

# 4. ALERTAS DE SURICATA IDS

# ============================================================================

echo "" >> "$REPORT"

echo "4. ALERTAS DE SURICATA IDS" >> "$REPORT"

echo "----------------------------------------" >> "$REPORT"


if [ -f /var/log/suricata/fast.log ]; then

    TOTAL_ALERTS=$(grep "$(date +%m/%d/%Y)" /var/log/suricata/fast.log | wc -
l)

    echo "Total de alertas: $TOTAL_ALERTS" >> "$REPORT"


    if [ $TOTAL_ALERTS -gt 0 ]; then

        echo "" >> "$REPORT"

        echo "Top 5 alertas más frecuentes:" >> "$REPORT"

        grep "$(date +%m/%d/%Y)" /var/log/suricata/fast.log | \

            grep -oP '\[Classification: .*?\]' | sort | uniq -c | sort -rn |
head -5 >> "$REPORT"

    fi

else
```

```
    echo "Log de Suricata no disponible" >> "$REPORT"
fi

# ================================================================================
# 5. CAMBIOS EN ARCHIVOS CRÍTICOS
# ================================================================================
echo "" >> "$REPORT"
echo "5. CAMBIOS EN ARCHIVOS CRÍTICOS" >> "$REPORT"
echo "----------------------------------------" >> "$REPORT"

# Verificar integridad de archivos críticos con AIDE o checksums
CRITICAL_FILES=(
    "/etc/passwd"
    "/etc/shadow"
    "/etc/sudoers"
    "/etc/ssh/sshd_config"
    "/etc/apache2/apache2.conf"
    "/var/www/html/index.php"
)

CHANGES_DETECTED=0
for file in "${CRITICAL_FILES[@]}"; do
    if [ -f "$file" ]; then
        LAST_MODIFIED=$(stat -c %y "$file" | cut -d' ' -f1)
        if [ "$LAST_MODIFIED" == "$DATE" ]; then
            echo "⚠  Modificado hoy: $file" >> "$REPORT"
            CHANGES_DETECTED=1
        fi
    fi
done
```

```
if [ $CHANGES_DETECTED -eq 0 ]; then
    echo "✓ No se detectaron cambios en archivos críticos" >> "$REPORT"
fi

# ================================================================================
# 6. USO DE RECURSOS
# ================================================================================
echo "" >> "$REPORT"
echo "6. USO DE RECURSOS DEL SISTEMA" >> "$REPORT"
echo "----------------------------------------" >> "$REPORT"

# Uso de disco
DISK_USAGE=$(df -h / | tail -1 | awk '{print $5}' | sed 's/%//')
echo "Uso de disco (/): $DISK_USAGE%" >> "$REPORT"
if [ $DISK_USAGE -gt 85 ]; then
    echo "⚠  ALERTA: Uso de disco alto" >> "$REPORT"
fi

# Uso de memoria
MEM_USAGE=$(free | grep Mem | awk '{print int($3/$2 * 100)}')
echo "Uso de memoria: $MEM_USAGE%" >> "$REPORT"
if [ $MEM_USAGE -gt 90 ]; then
    echo "⚠  ALERTA: Uso de memoria alto" >> "$REPORT"
fi

# Procesos sospechosos
SUSPICIOUS_PROCS=$(ps aux | grep -E "nc|netcat|python.*http.server|php.*-S" |
grep -v grep | wc -l)
if [ $SUSPICIOUS_PROCS -gt 0 ]; then
    echo "⚠  ALERTA: Procesos sospechosos detectados" >> "$REPORT"
    ps aux | grep -E "nc|netcat|python.*http.server|php.*-S" | grep -v
grep >> "$REPORT"
```

```
fi

# ============================================================================
# RESUMEN Y RECOMENDACIONES
# ============================================================================
echo "" >> "$REPORT"
echo "=======================================" >> "$REPORT"
echo "RESUMEN Y RECOMENDACIONES" >> "$REPORT"
echo "=======================================" >> "$REPORT"

# Generar recomendaciones basadas en hallazgos
if [ $FAILED_SSH -gt 50 ]; then
    echo "• Considerar implementar rate limiting para SSH" >> "$REPORT"
fi

if [ $SQL_INJECTION -gt 10 ]; then
    echo "• Revisar implementación de prepared statements" >> "$REPORT"
    echo "• Considerar desplegar WAF (ModSecurity)" >> "$REPORT"
fi

if [ $DISK_USAGE -gt 85 ]; then
    echo "• Limpiar logs antiguos y archivos temporales" >> "$REPORT"
    echo "• Considerar expandir almacenamiento" >> "$REPORT"
fi

echo "" >> "$REPORT"
echo "Reporte generado: $(date)" >> "$REPORT"
echo "=======================================" >> "$REPORT"

# Enviar reporte por email (opcional)
if command -v mail &> /dev/null; then
```

```
    mail -s "Security Report - $DATE" admin@company.com < "$REPORT"

fi


# Mostrar resumen en consola

cat "$REPORT"
```

**Automatizar con cron:**

```
# Ejecutar análisis diario a las 23:00

0 23 * * * /opt/scripts/analyze-security-logs.sh >> /var/log/security-analysis.log 2>&1
```

## 4.5.3 Detección de Fallos de Seguridad en Logs

***Patrones de Ataque Comunes***

| Tipo de Ataque | Patrón en Logs | Severidad | Acción |
|---|---|---|---|
| **SQL Injection** | `' OR '1'='1`, `UNION SELECT`, `; DROP TABLE` | CRÍTICA | Bloquear IP, revisar código |
| **XSS** | `<script>`, `javascript:`, `onerror=` | ALTA | Bloquear IP, sanitizar inputs |
| **Path Traversal** | `../`, `..%2f`, `/etc/passwd` | ALTA | Bloquear IP, actualizar Apache |
| **Command Injection** | `; ls`, `\| cat`, `&& whoami` | CRÍTICA | Bloquear IP, revisar código |
| **Brute Force SSH** | >10 `Failed password` en 5 min | MEDIA | Fail2ban, 2FA |
| **Directory Scanning** | `403` en `/admin`, `/.git`, `/.env` | MEDIA | Verificar permisos |
| **Webshell Upload** | `.php` en uploads, `eval(`, `base64_decode` | CRÍTICA | Cuarentena, escaneo |
| **DDoS** | >1000 requests/seg desde misma IP | ALTA | Rate limiting, CDN |

***Script de Alertas en Tiempo Real***

```
#!/bin/bash

# /opt/scripts/realtime-security-monitor.sh


# Monitorizar logs en tiempo real y alertar

tail -F /var/log/apache2/access.log | while read line; do

    # Detectar SQL Injection

    if echo "$line" | grep -qiE "union.*select|'\s*or\s*'1'='1|;\s*drop\s*table"; then

        IP=$(echo "$line" | awk '{print $1}')
```

```
        echo "[ALERTA] SQL Injection detectado desde $IP"
        logger -t SECURITY "SQL Injection attempt from $IP"

        # Bloquear IP automáticamente
        if ! iptables -L INPUT -n | grep -q "$IP"; then
            iptables -A INPUT -s "$IP" -j DROP
            echo "$IP bloqueada automáticamente"
        fi
    fi

    # Detectar Path Traversal
    if echo "$line" | grep -qiE "\.\./|\.\.%2f|/etc/passwd"; then
        IP=$(echo "$line" | awk '{print $1}')
        echo "[ALERTA] Path Traversal detectado desde $IP"
        logger -t SECURITY "Path Traversal attempt from $IP"
    fi

    # Detectar Webshell Upload
    if echo "$line" | grep -qiE "c99\.php|r57\.php|shell\.php|cmd\.php"; then
        IP=$(echo "$line" | awk '{print $1}')
        echo "[ALERTA CRÍTICA] Posible webshell upload desde $IP"
        logger -t SECURITY "Webshell upload attempt from $IP"

        # Enviar alerta inmediata
        echo "Webshell detectado desde $IP" | \
            mail -s "ALERTA CRÍTICA SEGURIDAD" admin@company.com
    fi
done
```

**Ejecutar como servicio systemd:**

```
# /etc/systemd/system/security-monitor.service
```

```
[Unit]

Description=Real-time Security Monitor

After=network.target


[Service]

Type=simple

ExecStart=/opt/scripts/realtime-security-monitor.sh

Restart=always

User=root


[Install]

WantedBy=multi-user.target

systemctl enable security-monitor

systemctl start security-monitor
```

# 4.6 PROCEDIMIENTOS DE RESPUESTA A INCIDENTES

## 4.6.1 Niveles de Incidentes

| Nivel | Descripción | Ejemplos | Tiempo Respuesta | Responsable |
|---|---|---|---|---|
| **P0 - CRÍTICO** | Compromiso activo del sistema | Ransomware, RCE activo, exfiltración de datos | < 15 min | CISO + Equipo completo |
| **P1 - ALTO** | Vulnerabilidad crítica detectada | SQLi con evidencia de explotación, malware | < 1 hora | Security Team Lead |
| **P2 - MEDIO** | Actividad sospechosa | Múltiples intentos de login, escaneo de puertos | < 4 horas | Analista de Seguridad |
| **P3 - BAJO** | Anomalía menor | Single failed login, página 404 inusual | < 24 horas | Monitorización |

## 4.6.2 Procedimiento de Respuesta (IR Playbook)

***Fase 1: DETECCIÓN (0-5 minutos)***

**Fuentes de detección:**

- Alertas de Suricata IDS
- Alertas de script de monitorización
- Reportes de usuarios
- Escaneos programados de OpenVAS
- Análisis de logs

**Acciones:**

1. Documentar alerta inicial (timestamp, fuente, tipo)
2. Verificar si es falso positivo
3. Clasificar severidad (P0, P1, P2, P3)
4. Activar cadena de notificación según severidad

***Fase 2: CONTENCIÓN (5-30 minutos)***

**Para P0/P1:**

```
#!/bin/bash

# incident-containment.sh


INCIDENT_ID="$1"

ATTACKER_IP="$2"


# 1. Aislar servidor comprometido

iptables -P INPUT DROP

iptables -P OUTPUT DROP

iptables -A INPUT -i lo -j ACCEPT

iptables -A OUTPUT -o lo -j ACCEPT


# 2. Bloquear IP atacante en todos los sistemas

echo "$ATTACKER_IP" >> /etc/blocked_ips.txt

iptables -A INPUT -s "$ATTACKER_IP" -j DROP


# 3. Capturar estado actual

mkdir -p /forensics/$INCIDENT_ID


# Memoria RAM

if command -v memdump &> /dev/null; then

    memdump > /forensics/$INCIDENT_ID/memory.dump

fi


# Conexiones activas
```

```
netstat -antp > /forensics/$INCIDENT_ID/netstat.txt
ss -tulpn > /forensics/$INCIDENT_ID/sockets.txt

# Procesos
ps auxf > /forensics/$INCIDENT_ID/processes.txt
lsof > /forensics/$INCIDENT_ID/open_files.txt

# Usuarios logueados
w > /forensics/$INCIDENT_ID/logged_users.txt
last > /forensics/$INCIDENT_ID/login_history.txt

# 4. Snapshot del disco (si hay espacio)
if [ $(df / | tail -1 | awk '{print $4}') -gt 10485760 ]; then
    dd if=/dev/sda of=/forensics/$INCIDENT_ID/disk.img bs=4M status=progress
fi

# 5. Notificar
echo "Incidente $INCIDENT_ID: Sistema contenido" | \
    mail -s "INCIDENT CONTAINMENT" admin@company.com

echo "Contención completada. Datos forenses en /forensics/$INCIDENT_ID"
```

***Fase 3: ERRADICACIÓN (30 minutos - 2 horas)***

**Checklist de erradicación:**

- [ ] Identificar vector de entrada
- [ ] Eliminar backdoors/webshells
- [ ] Eliminar malware/rootkits
- [ ] Cerrar vulnerabilidad explotada
- [ ] Cambiar todas las credenciales
- [ ] Rotar claves SSH
- [ ] Revisar cuentas de usuario
- [ ] Eliminar accesos no autorizados

**Herramientas:**

```
# Búsqueda de rootkits
apt install -y rkhunter chkrootkit
rkhunter --check
chkrootkit

# Búsqueda de webshells
grep -r "eval(" /var/www/
grep -r "base64_decode" /var/www/
grep -r "system(" /var/www/

# Búsqueda de archivos modificados recientemente
find /var/www -type f -mtime -1

# Verificar cuentas de usuario
awk -F: '$3 >= 1000 {print $1}' /etc/passwd
lastlog

# Verificar cron jobs sospechosos
crontab -l
cat /etc/crontab
ls -la /etc/cron.*
```

***Fase 4: RECUPERACIÓN (2-4 horas)***

**Ver secciones 4.4.2 y 4.4.3 para procedimientos de restauración**

Pasos clave:

1. Restaurar desde backup limpio
2. Aplicar todos los parches de seguridad
3. Implementar hardening adicional
4. Reiniciar servicios
5. Validar funcionamiento

***Fase 5: POST-INCIDENTE (24-48 horas)***

**Informe post-incidente (template):**

```
# INFORME POST-INCIDENTE
```

## Información General

- **ID Incidente:** INC-2025-001
- **Fecha/Hora:** 2025-11-10 14:35:00
- **Severidad:** P1 (Alta)
- **Duración:** 3 horas 25 minutos
- **Analista:** [Nombre]

## Resumen Ejecutivo

[Descripción breve del incidente]

## Timeline

| Tiempo | Evento |
|---|---|
| 14:35 | Detección inicial |
| 14:40 | Confirmación de incidente |
| 14:45 | Contención iniciada |
| 15:30 | Erradicación completada |
| 17:00 | Servicios restaurados |
| 18:00 | Validación final |

## Análisis de Causa Raíz

### Vector de Ataque

[Cómo entró el atacante]

### Vulnerabilidad Explotada

[Qué vulnerabilidad se usó]

### Impacto

- Sistemas afectados:
- Datos comprometidos:

```
- Tiempo de inactividad:


## Acciones Correctivas

1. [Acción inmediata tomada]

2. [Mejora de seguridad implementada]

3. [Actualización de procedimientos]


## Acciones Preventivas

1. [Medida para prevenir recurrencia]

2. [Mejora de monitorización]

3. [Training requerido]


## Lecciones Aprendidas

- ¿Qué funcionó bien?

- ¿Qué se puede mejorar?

- ¿Qué falló?


## Seguimiento

- [ ] Validar parches aplicados

- [ ] Revisar logs por 7 días

- [ ] Actualizar runbooks

- [ ] Comunicar a stakeholders
```

# 4.7 TIMELINE DE IMPLEMENTACIÓN

## Fase 1: Crítico (Semana 1)

| Día | Actividad | Responsable | Duración |
|---|---|---|---|
| 1-2 | Parchear SQL Injection + XSS | Dev Team | 16h |
| 2-3 | Configurar contraseña MySQL root | SysAdmin | 2h |
| 3-4 | Actualizar Apache (Path Traversal) | SysAdmin | 4h |
| 4-5 | Implementar backup automatizado | SysAdmin | 8h |
| 5 | Validación y testing | QA | 8h |

**Resultado esperado:** Vulnerabilidades P0 mitigadas

### Fase 2: Alta Prioridad (Semanas 2-3)

| Semana | Actividad | Duración |
|---|---|---|
| 2 | Deshabilitar TLS 1.0/1.1 | 4h |
| 2 | Implementar CSP headers | 8h |
| 2 | Configurar logging centralizado | 16h |
| 3 | Desplegar WAF (ModSecurity) | 16h |
| 3 | Configurar fail2ban avanzado | 8h |
| 3 | Testing y ajustes | 16h |

**Resultado esperado:** Vulnerabilidades P1 mitigadas

### Fase 3: Mejoras Continuas (Mes 1)

- Semana 4: Implementar monitorización avanzada
- Semana 4: Documentar procedimientos de recuperación
- Semana 4: Training del equipo en respuesta a incidentes

**Resultado esperado:** Sistema hardened y monitorizado

# 4.8 MÉTRICAS Y KPIs

### Indicadores de Éxito

| Métrica | Objetivo | Medición |
|---|---|---|
| Tiempo de detección (MTTD) | < 15 min | Tiempo entre compromiso y alerta |
| Tiempo de respuesta (MTTR) | < 4 horas | Tiempo entre alerta y mitigación |
| Vulnerabilidades críticas | 0 | Escaneos mensuales OpenVAS |
| Backups exitosos | 100% | Verificación diaria |
| RTO (Base de datos) | < 30 min | Pruebas trimestrales |
| RPO (Pérdida de datos) | < 6 horas | Configuración de backups |

# 4.9 COSTOS ESTIMADOS

| Concepto | Costo Único | Costo Mensual |
|---|---|---|
| Licencias WAF | - | $200 |
| Almacenamiento backup (500GB) | $500 | $50 |
| Servidor de logs (opcional) | $1,000 | $100 |
| Herramientas de monitorización | - | $150 |
| Training equipo | $2,000 | - |
| **TOTAL** | **$3,500** | **$500** |

**ROI:** Prevención de un solo incidente de ransomware (promedio $150,000) justifica la inversión.

**FIN DEL PLAN DE MITIGACIÓN**