3-1-2026

Análisis de una Vulnerabilidad

Ciberseguridad

A09:2025 - Errores de registro y alertas de seguridad

VÉRONIQUE GRUÉ

1. Descripción

Sin registro ni monitorización, no se pueden detectar ataques ni infracciones, y sin alertas es muy difícil responder con rapidez y eficacia ante un incidente de seguridad. La insuficiencia de registro, monitorización continua, detección y alertas para iniciar respuestas activas puede ocurrir en cualquier momento.

2. Ejemplo Práctico

Un atacante realiza 10,000 intentos de login fallidos contra diferentes cuentas durante 3 días. Sin registros adecuados ni alertas configuradas, nadie detecta el ataque de fuerza bruta en curso. Eventualmente el atacante compromete varias cuentas y extrae datos sensibles. Cuando finalmente se descubre la brecha, ha pasado un mes y no hay registros que permitan determinar qué datos fueron comprometidos.

3. Impacto Potencial

- Ataques no detectados que permanecen activos durante meses.
- Imposibilidad de realizar análisis forense tras incidentes.
- Incapacidad para cumplir requisitos regulatorios de auditoría.
- Tiempo de respuesta extremadamente largo ante incidentes.
- Desconocimiento del alcance real de brechas de seguridad.
- Repetición de ataques por falta de aprendizaje.

4. Medidas Preventivas

- Registrar todos los eventos de seguridad relevantes (logins, fallos de autenticación, cambios de privilegios).
- Implementar monitorización continua y análisis de logs.
- Configurar alertas automáticas para eventos sospechosos.
- Proteger logs contra alteración y garantizar su integridad.
- Centralizar logs en sistemas SIEM (Security Information and Event Management).
- Establecer tiempos de retención adecuados para logs. (Se aconseja guardar los registros de 30 a 90 días.
- Realizar revisiones periódicas de logs y patrones anómalos.
- Implementar correlación de eventos para detectar patrones de ataque.
- Crear procedimientos de respuesta ante alertas de seguridad.

5. Impacto Potencial en la Aplicación Final

En el contexto de la **Aplicación Final** desarrollada en PHP, la ausencia de una estrategia de registro y monitorización (A09) no solo es una debilidad teórica, sino un riesgo operativo real que afecta a los siguientes puntos:

- **Invisibilidad ante ataques de Fuerza Bruta:** La aplicación utiliza un sistema de login basado en el modelo UsuarioPDO. Si un atacante intenta adivinar contraseñas mediante scripts automatizados y no registramos los fallos de autenticación, el servidor procesará miles de peticiones sin que el administrador reciba notificación alguna, facilitando el compromiso de cuentas.

- **Falta de Trazabilidad en la Gestión de Departamentos:** Al realizar operaciones críticas como la "Baja Física" de un departamento (DepartamentoPDO::bajaFisicaDepartamento), si no existe un log de auditoría, es imposible determinar qué usuario eliminó la información, desde qué dirección IP y en qué momento exacto. Esto genera una pérdida total de responsabilidad.

- **Fuga de Información por Errores Expuestos:** Si la aplicación no está configurada para registrar errores en archivos internos y, en su lugar, los muestra por pantalla (display_errors), un atacante puede forzar errores en la base de datos para obtener información sobre la estructura de las tablas, rutas del servidor o versiones de software, facilitando un ataque posterior de inyección SQL.

- **Análisis Forense Nulo tras una Brecha:** En caso de que un atacante logre acceder a la Página de Inicio Privado o al panel de administración, la falta de logs impedirá saber si se extrajeron datos de los servicios REST (como los datos de la NASA) o si se modificaron perfiles de otros usuarios.

6. Medidas Preventivas para la Aplicación Final

Para mitigar los riesgos asociados al **A09**, implementaremos las siguientes soluciones técnicas en nuestra aplicación:

A. Configuración del Entorno de Ejecución (PHP.ini)

Estableceremos una política de "**Cero visibilidad externa, máxima visibilidad interna**":

En este caso, que se tienen los permisos para acceder al php.ini, se puede definir el log en la carpeta de la aplicación de esta forma: ruta del fichero /etc/php/8.3/fpm/php.ini

- **display_errors = Off**: Se evita que el usuario final vea errores técnicos.
- **log_errors = On**: Se activa el registro interno de fallos.
- Se define un archivo de registro fuera de la carpeta pública (www) para que no sea accesible vía navegador. Se descomenta la línea error_log = php_errors.log de php.ini, y se indica la ruta del archivo del log de la aplicación.

```
; Log errors to specified file. PHP's default beh
; empty.
; https://php.net/error-log
; Example:
;error_log = php_errors.log
; Log errors to syslog (Event Log on Windows).
;error_log = syslog
```

- **error_log = /var/www/html/VGDWESAplicacionFinal/tmp/php_errors.log**:

```
; Log errors to specified file. PHP's default behavior is to leave thi
; empty.
; https://php.net/error-log
; Example:
error_log = /var/www/html/VGDWESAplicacionFinal/tmp/php_errors.log
; Log errors to syslog (Event Log on Windows).
;error_log = syslog
```

Detalle de la configuración en este enlace:
https://github.com/verogmayo/VGDAWProyectoDAW/blob/master/ServidorDeDesarrollo.md#configuraci%C3%B3n-del-phpini-para-un-entorno-de-desarrollo

Si no se puede acceder al php.ini por falta de permisos se puede definir en el fichero de configuración de la base de datos de la aplicación(confDBPDODes.php), que es el fichero para desarrollo, ya que en explotación ya existe ese fichero.

ini_set("error_log", "/var/www/html/VGDWESAplicacionFinal/tmp/php-error.log");

VGDWESAplicacionFinal > config > confDBPDODes.php > ...

```php
<?php
//    define('DNS', 'mysql:host=' . $_SERVER['SERVER_ADDR'] . ';
dbname=DBVGDWESAplicacionFinalTema5');
// configuración para plesk
    define('DSN', 'mysql:host=localhost;dbname=DBVGDWESAplicacionFinal;charset=utf8');
    define('USUARIODB', 'userVGDWESAplicacionFinal');
    define('PSWD', 'paso');

    //Definicion del archivo de log
    ini_set('log_errors', 'On');
    ini_set('error_log', '/var/www/html/VGDWESAplicacionFinal/tmp/php-error.log');
?>

```

B. Implementación de Logs de Auditoría en el Modelo

Trazabilidad en la función borrarDepartamento de DepartamentoPDO.

```php
public static function borrarDepartamento($oDepartamento)
{
    $sql = <<<SQL
        DELETE FROM T02_Departamento
        WHERE T02_CodDepartamento = :codDepartamento
    SQL;
    try {
        $consulta = DBPDO::ejecutarConsulta($sql, [
            ':codDepartamento' => $oDepartamento->getCodDepartamento()
        ]);
        //rowCount() para ver si se borro la fila
        if ($consulta->rowCount() > 0) {
            // REGISTRO  (OWASP A09)
            // Obtenemos el usuario de la sesión para identidficar el usaurio que ha borrado el departamento
            $usuarioActivo = $_SESSION['usuarioVGDAWAplicacionFinal']->getCodUsuario();
            $codDpto = $oDepartamento->getCodDepartamento();
            //se recoge el fecha y la hora del borrado
            $oFechaHora = new DateTime();
            $fechaHoraFormateada = $oFechaHora->format('d-m-Y H:i:s');
            //se recoge el mensage de log infomrando de quien y cuando se ha borrado el departamento
            error_log("[AUDITORÍA] [$fechaHoraFormateada] El usuario '$usuarioActivo' ha BORRADO el
            departamento '$codDpto'.");
            return true;
        }
    } catch (Exception $e) {
        //se recoge el error de borrado fallido
        error_log("ERROR CRÍTICO: Intento de borrado fallido del depto " . $oDepartamento->getCodDepartamento
        ());
        return false;
    }
    return false;
```

Aquí se puede ver el log en el archivo php_error.log

php-error.log

```
[03-Feb-2026 18:32:42 UTC] [AUDITORÍA] [03-02-2026 18:32:42] El usuario 'vero' ha BORRADO el departamento 'JAP'.
```

C. Gestión de Excepciones Segura

Modificaremos los bloques try-catch de nuestra clase DBPDO para que, además de capturar la excepción, la registre de forma detallada:

```php
public static function ejecutarConsulta($sentenciaSQL, $aParametros = null) {
    try {
        // Conexión a la base de datos
        $conexion = new PDO(DSN, USUARIODB, PSWD);
        // Configurar el modo errores
        $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        //Preparar y ejecutar la consulta
        $consulta = $conexion->prepare($sentenciaSQL);
        $consulta->execute($aParametros);

        return $consulta;
    }catch (PDOException $e) {
        //Mensaje de error para el log interno
        error_log("ERROR DE BASE DE DATOS: " . $e->getMessage() . " en la consulta: " . $sentenciaSQL);
        $_SESSION['paginaAnterior']=$_SESSION['paginaEnCurso'];
        $_SESSION['paginaEnCurso']='error';
        //Se crea el objeto error para la vista del error para el usuario
        $_SESSION['error']= new AppError($e->getCode(),$e->getMessage(),$e->getFile(),$e->getLine(), $_SESSION
        ['paginaAnterior']);
        header('Location: index.php');
        exit;
    }

}
```

Log de Error de la base de Datos

```
[03-Feb-2026 19:01:00 UTC] ERROR DE BASE DE DATOS: SQLSTATE[42S02]: Base table or view not found: 1146 Table 'DBVGDWESAplicacionFinal.T03_Cuestion'
 doesn't exist en la consulta: SELECT * FROM T03_Cuestion
```

D. Monitorización de Intentos de Login

Se implementará una lógica que registre no solo el éxito, sino especialmente el fallo de autenticación, incluyendo el código de usuario intentado:

Si se detectan más de **5 intentos fallidos** en un periodo corto desde la misma IP, el sistema podría bloquear temporalmente dicha IP o requerir un factor adicional de autenticación (MFA/Captcha). En este caso se envía al fichero de error un mensaje en mayúsculas para recoger la incidencia.

Aquí se puede ver parte de la función validar usuario con el código para el log de errores:

```php
public static function validarUsuario($codUsuario, $password)
    $sql;

    $oFechaHora = new DateTime();
    $fechaHoraFormateada = $oFechaHora->format('d-m-Y H:i:s');

    try {
        // Ejecutar la consulta
        $consulta = DBPDO::ejecutarConsulta($sql, [
            ':codUsuario'  => $codUsuario,
            ':password' => $codUsuario . $password
        ]);
        // Obtener el resultado
        $usuarioDB = $consulta->fetchObject();
        // Si no existe el usuario o la contraseña es incorrecta
        if (!$usuarioDB) {
            // SE inicializa o incrementa el contador de intentos en la sesión
            $_SESSION['intentosLogin'] = ($_SESSION['intentosLogin'] ?? 0) + 1;
            //se crea el mensaje de intento fallido
            $mensaje = "[AUDITORÍA] [$fechaHoraFormateada] Intento de login FALLIDO para el usuario
            '$codUsuario'.";
            // Si llega a 5 intentos, se pone el mensaje en MAYÚSCULAS para que destaque
            if ($_SESSION['intentosLogin'] >= 5) {
                $mensaje = strtoupper("[ALERTA CRÍTICA] [$fechaHoraFormateada] SE HAN DETECTADO " . $_SESSION
                ['intentosLogin'] . " INTENTOS FALLIDOS CONSECUTIVOS PARA EL USUARIO '$codUsuario'. POSIBLE
                ATAQUE DE FUERZA BRUTA.");
            }
            //se envia el mensaje al fichero de error
            error_log($mensaje);
            return null;
        }
```

Log de errores de los intentos fallidos.

```
[03-Feb-2026 18:55:14 UTC] [AUDITORÍA] [03-02-2026 18:55:14] Intento de login FALLIDO para el usuario 'vero'.
[03-Feb-2026 18:55:20 UTC] [AUDITORÍA] [03-02-2026 18:55:20] Intento de login FALLIDO para el usuario 'vero'.
[03-Feb-2026 18:55:26 UTC] [AUDITORÍA] [03-02-2026 18:55:26] Intento de login FALLIDO para el usuario 'vero'.
[03-Feb-2026 18:55:31 UTC] [AUDITORÍA] [03-02-2026 18:55:31] Intento de login FALLIDO para el usuario 'vero'.
[03-Feb-2026 18:55:38 UTC] [ALERTA CRÍTICA] [03-02-2026 18:55:38] SE HAN DETECTADO 5 INTENTOS FALLIDOS CONSECUTIVOS PARA EL USUARIO 'VERO'. POSIBLE
 ATAQUE DE FUERZA BRUTA.
[03-Feb-2026 18:55:45 UTC] [ALERTA CRÍTICA] [03-02-2026 18:55:45] SE HAN DETECTADO 6 INTENTOS FALLIDOS CONSECUTIVOS PARA EL USUARIO 'VERO'. POSIBLE
 ATAQUE DE FUERZA BRUTA.
[03-Feb-2026 18:55:49 UTC] [AUDITORÍA] [03-02-2026 18:55:49] Login exitoso para el usuario 'vero'.
```

7. Conclusión

La vulnerabilidad A09:2025 representa uno de los riesgos más subestimados en el desarrollo de aplicaciones web, ya que su ausencia no genera fallos visibles inmediatos, pero imposibilita la detección temprana de ataques y el análisis forense posterior a incidentes de seguridad. Como se ha demostrado en este análisis, una aplicación sin un sistema robusto de registro y monitorización opera, en esencia, "a ciegas", permitiendo que atacantes comprometan cuentas, extraigan información sensible o modifiquen datos críticos sin dejar rastro alguno.