Reporte de Vulnerabilidad

Archivo: ClassUsuario.php

Código Analizado:

```
<?php

 class Usuario{

  private $strNombre;
  private $strEmail;
  private $strTipo;
  private $strClave;
  protected $strFechaRegistro;
  static $strEstado = "Activo";

  function __construct(string $nombre, string $email, string $tipo)
  {
   $this->strNombre = $nombre;
   $this->strEmail = $email;
   $this->strTipo = $tipo;
   $this->strClave = rand();
   $this->strFechaRegistro = date('Y-m-d H:m:s');
  }

  public function getNombre():string
  {
   return $this->strNombre;
  }

  public function getEmail():string
  {
   return $this->strEmail;
  }

  public function gerPerfil() {
   echo "<div style='border: 1px solid #ccc; padding: 10px; font-family: sans-serif;'>"; //
Contenedor con estilos
   echo "<h2 style='color: #333;'>Datos del usuario</h2>"; // Encabezado

   echo "<p><strong>Nombre:</strong> " . $this->strNombre . "</p>";
   echo "<p><strong>Email:</strong> " . $this->strEmail . "</p>";

   // ¡Nunca muestres la clave real!
   echo "<p><strong>Clave:</strong> " . $this->strClave . "</p>";

   echo "<p><strong>Fecha de registro:</strong> " . $this->strFechaRegistro . "</p>";
   echo "<p><strong>Estado:</strong> " . self::$strEstado . "</p>";

   echo "</div>"; // Cierre del contenedor
  }

  public function setCambiarClave(string $pass){
   $this->strClave = $pass;
  }

 }//End class usuario

 ?>
```

Análisis: ```html
Vulnerabilidades de Seguridad

Revelación de Información Sensible (Clave)

Tipo: Revelación de Información Sensible / Hardcoded Value / Debug Information Leakage.

Línea Aproximada: Línea 39: `echo "<p><strong>Clave:</strong> " . $this->strClave . "</p>";`

Descripción: El método `gerPerfil()` muestra la clave del usuario directamente en el HTML. Esto es una grave vulnerabilidad de seguridad, ya que expone la clave a cualquiera que pueda ver la página. Además, la clave se genera de forma predeterminada con `rand()` en el constructor, que no es criptográficamente seguro.

Mitigación:

- **Nunca** mostrar la clave directamente. En su lugar, considere mostrar un hash de la clave (si es necesario) o, mejor aún, no mostrar nada relacionado con la clave.
- Almacenar la clave de forma segura usando un algoritmo de hash robusto como `password_hash()` de PHP y sal.
- Generar claves usando una función segura como `random_bytes()` y codificarlas usando `bin2hex()` o `base64_encode()`.

Mejora de Métricas: N/A (Esta es una vulnerabilidad de seguridad, no un problema de calidad de código en sí mismo, aunque contribuye a una mala práctica).

Métricas de Calidad del Código

Complejidad

Descripción: La complejidad ciclomática del código es baja. No hay lógica compleja ni ramas anidadas.

Mejora: Por el momento, no se requiere mejorar la complejidad. Si se agregara lógica más compleja (validación, etc.), considere refactorizar en métodos más pequeños y enfocados.

Acoplamiento

Descripción: El acoplamiento es bajo. La clase `Usuario` no depende fuertemente de otras clases externas (excepto las funciones nativas de PHP).

Mejora: Mantener el acoplamiento bajo es bueno. Si se agregaran dependencias, considere usar la inyección de dependencias para mejorar la testabilidad y el desacoplamiento.

Legibilidad

Descripción: El código es generalmente legible. Los nombres de las variables y métodos son descriptivos. El formato es consistente.

Mejora: Se puede mejorar la consistencia utilizando un estándar de codificación (PSR-12). Asegurarse de que todos los métodos tengan comentarios PHPDoc.

Duplicación

Descripción: No hay duplicación significativa en el código proporcionado.

Mejora: N/A

Formato de Fecha

Descripción: Existe un error sutil en el formato de fecha.

Línea Aproximada: Línea 17: `$this->strFechaRegistro = date('Y-m-d H:m:s');`

Mejora: El formato correcto para los minutos es `i`, no `m`. Se debe cambiar a `$this->strFechaRegistro = date('Y-m-d H:i:s');`

Solución Propuesta

Código Modificado (gerPerfil):

```
public function gerPerfil() {
    echo "<div style='border: 1px solid #ccc; padding: 10px; font-family: sans-serif;'>";
    echo "<h2 style='color: #333;'>Datos del usuario</h2>";

    echo "<p><strong>Nombre:</strong> " . htmlspecialchars($this->strNombre) . "</p>";
    echo "<p><strong>Email:</strong> " . htmlspecialchars($this->strEmail) . "</p>";

    //  ¡Nunca muestres la clave!
    echo "<p><strong>Clave:</strong>  ******** (oculto) </p>";

    echo "<p><strong>Fecha de registro:</strong> " . $this->strFechaRegistro . "</p>";
    echo "<p><strong>Estado:</strong> " . self::$strEstado . "</p>";

    echo "</div>";
}

//En el constructor, cambiar la generación de la clave por algo seguro
function __construct(string $nombre, string $email, string $tipo)
{
    $this->strNombre = $nombre;
    $this->strEmail = $email;
    $this->strTipo = $tipo;
    $this->strClave = bin2hex(random_bytes(16)); // Genera una clave aleatoria segura
    $this->strFechaRegistro = date('Y-m-d H:i:s'); //Corregido el formato de minutos
}

public function setCambiarClave(string $pass){
  //Hay que hashear la clave antes de guardarla
  $this->strClave = password_hash($pass, PASSWORD_DEFAULT);
}
```

Explicación de Cambios:

- La clave ya no se muestra en el perfil del usuario. Se reemplaza por un texto que indica que está oculta.
- Se usa `htmlspecialchars()` para escapar los nombres y emails, previniendo XSS.
- Se corrigió el formato de la fecha a `H:i:s`.
- Se usa `random_bytes()` y `bin2hex()` para generar una clave segura en el constructor.
- Se usa `password_hash()` para hashear la clave cuando se cambia.

```
```