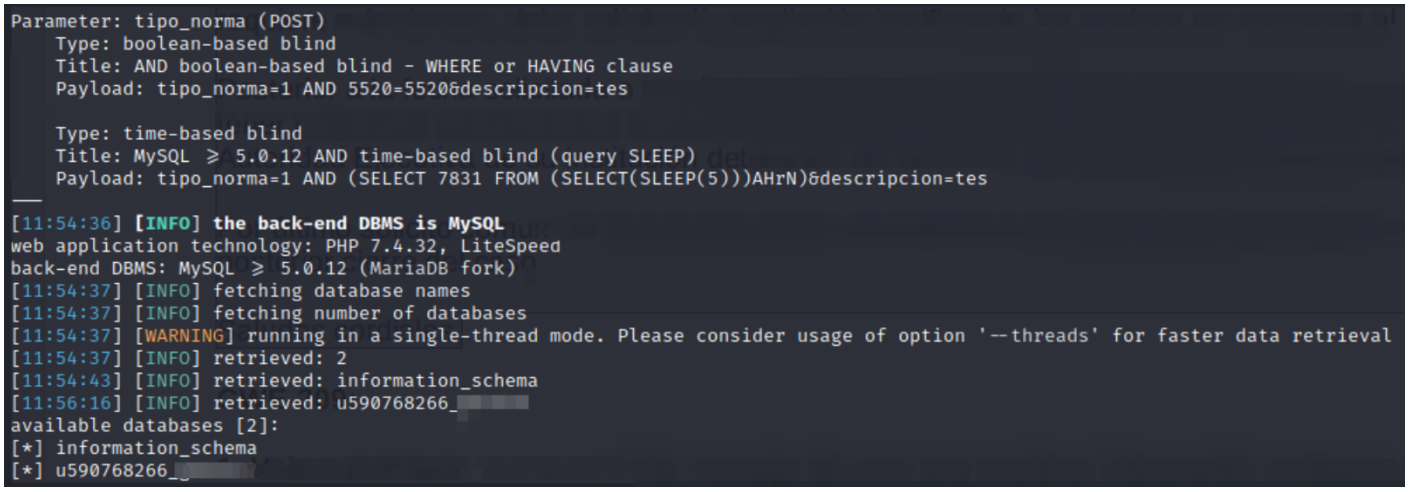
Aplicaciones web

Capítulo destinado a la solución de vulnerabilidades en aplicaciones web.

Inyección SQL

Sin la eliminación o citación suficiente de la sintaxis SQL en las entradas controlables por el usuario, la consulta SQL generada puede hacer que esas entradas se interpreten como SQL en lugar de datos de usuario ordinarios. Esto se puede usar para alterar la lógica de consulta para eludir las comprobaciones de seguridad o para insertar declaraciones adicionales que modifican la base de datos de back-end, posiblemente incluyendo la ejecución de comandos del sistema.

La inyección SQL se ha convertido en un problema común con los sitios web basados en bases de datos. La falla se detecta y explota fácilmente y, como tal, es probable que cualquier sitio o paquete de software con una base mínima de usuarios esté sujeto a un intento de ataque de este tipo. Esta falla depende del hecho de que SQL no hace una distinción real entre los planos de control y de datos.



Inyección SQL PHP - Solución

La solución aplica para aplicaciones que no usan ningún framework como base de desarrollo.

Opción a)

Usar esta opción en versiones de PHP 4 > = 4.3.0 hasta versiones anteriores a PHP 5.5.0.

```
$id=mysqlrealescapestring($GET["id"]); // sanitizando el parámetro id (ejemplo)
```

Opción b)

Ejemplo, valida el parámetro id

```
$id=$GET["id"];
$id=str_replace("'","",$id);
$id=str_replace('"','',$id);
$id=str_replace(')','',$id);
$id=str_replace('-','',$id);
$id=str_replace('%','',$id);
```

Opción c)

Usar consultas parametrizadas
https://www.php.net/manual/es/mysqli.quickstart.prepared-statements.php

También es importante realizar la verificación y validación en el resto de campos de entrada que existan en formularios o parámetros en URL de su sitio web.

Inyección SQL ASP.net - solución

Comience restringiendo la entrada en el código del lado del servidor para sus páginas web ASP.NET. No confíe en la validación del lado del cliente porque se puede omitir fácilmente. Use la validación del lado del cliente solo para reducir los viajes de ida y vuelta y mejorar la experiencia del usuario.

Si usa controles de servidor, use los controles de validación de ASP.NET, como los controles RegularExpressionValidator y RangeValidator para restringir la entrada. Si usa controles de entrada HTML regulares, use la clase Regex en su código del lado del servidor para restringir la entrada.

Opción a)

Puede restringir su entrada usando un control RegularExpressionValidator como se muestra a continuación:

```
<%@ language="C#" %>
<form id="form1" >
    <asp: TextBox ID="SSN" />
    <asp: RegularExpressionValidator ID="regexpSSN"
                                     ErrorMessage="Incorrect SSN Number"
                                     ControlToValidate="SSN"
                                     ValidationExpression="^\d{3}-\d{2}-\d{4}$" />
```

```
</form>
```

Si la entrada de SSN proviene de otra fuente, como un control HTML, un parámetro de cadena de consulta o una cookie, puede restringirla usando la clase Regex del espacio de nombres System.Text.RegularExpressions . El siguiente ejemplo asume que la entrada se obtiene de una cookie.

Opción b)

Usando System.Text.RegularExpressions;

```
if (Regex.IsMatch(Request.Cookies["SSN"], "^\d{3}-\d{2}-\d{4}$"))
{
    // access the database
}
else
{
    // handle the bad input
}
```

Opción c)

Usar parámetros con procedimientos almacenados:

El uso de procedimientos almacenados no impide necesariamente la inyección de SQL. Lo importante es usar parámetros con procedimientos almacenados. Si no usa parámetros, sus procedimientos almacenados pueden ser susceptibles a la inyección SQL si usan entrada sin filtrar como se describe en la sección "Descripción general" de este documento.

El siguiente código muestra cómo usar SqlParameterCollection al llamar a un procedimiento almacenado.

```
using System.Data;
using System.Data.SqlClient;


using (SqlConnection connection = new SqlConnection(connectionString))
{
  DataSet userDataset = new DataSet();
  SqlDataAdapter myCommand = new SqlDataAdapter(
              "LoginStoredProcedure", connection);
```

```
  myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;
  myCommand.SelectCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11);
  myCommand.SelectCommand.Parameters["@au_id"].Value = SSN.Text;

  myCommand.Fill(userDataset);
}
```

En este caso, el parámetro @au_id se trata como un valor literal y no como un código ejecutable. Además, se comprueba el tipo y la longitud del parámetro. En el ejemplo de código anterior, el valor de entrada no puede tener más de 11 caracteres. Si los datos no se ajustan al tipo o la longitud definidos por el parámetro, la clase SqlParameter genera una excepción.

Opción d)

Usar parámetros con SQL dinámico:

Si no puede usar procedimientos almacenados, aún debe usar parámetros cuando construya sentencias SQL dinámicas. El siguiente código muestra cómo usar SqlParametersCollection con SQL dinámico.

```
using System.Data;
using System.Data.SqlClient;

using (SqlConnection connection = new SqlConnection(connectionString))
{
  DataSet userDataset = new DataSet();
  SqlDataAdapter myDataAdapter = new SqlDataAdapter(
         "SELECT au_lname, au_fname FROM Authors WHERE au_id = @au_id",
         connection);
  myCommand.SelectCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11);
  myCommand.SelectCommand.Parameters["@au_id"].Value = SSN.Text;
  myDataAdapter.Fill(userDataset);
}
```

Opción e)

Uso de procesamiento por lotes de parámetros:

Un concepto erróneo común es que si concatena varias sentencias SQL para enviar un lote de sentencias al servidor en un solo viaje de ida y vuelta, no puede usar parámetros. Sin embargo, puede usar esta técnica si se asegura de que los nombres de los parámetros no se repitan. Puede hacer esto fácilmente asegurándose de usar nombres de parámetros únicos durante la concatenación de texto SQL, como se muestra aquí.

```
using System.Data;
using System.Data.SqlClient;
. . .
using (SqlConnection connection = new SqlConnection(connectionString))
{
  SqlDataAdapter dataAdapter = new SqlDataAdapter(
       "SELECT CustomerID INTO #Temp1 FROM Customers " +
       "WHERE CustomerID > @custIDParm; SELECT CompanyName FROM Customers " +
       "WHERE Country = @countryParm and CustomerID IN " +
       "(SELECT CustomerID FROM #Temp1);",
       connection);
  SqlParameter custIDParm = dataAdapter.SelectCommand.Parameters.Add(
                                          "@custIDParm", SqlDbType.NChar, 5);
  custIDParm.Value = customerID.Text;

  SqlParameter countryParm = dataAdapter.SelectCommand.Parameters.Add(
                                      "@countryParm", SqlDbType.NVarChar, 15);
  countryParm.Value = country.Text;

  connection.Open();
  DataSet dataSet = new DataSet();
  dataAdapter.Fill(dataSet);
}
. . .
```

Inyección SQL JAVA - solución

La solución más simple es usar PreparedStatement en lugar de Statement para ejecutar la consulta.

En lugar de concatenar el nombre de usuario y la contraseña en la consulta, los proporcionamos para consultar a través de los métodos de establecimiento de PreparedStatement.

Ahora, el valor del nombre de usuario y la contraseña recibidos de la solicitud se tratan solo como datos, por lo que no se producirá una inyección SQL.

Veamos el código del servlet modificado.

```
String query = "select * from tbluser where username=? and password = ?";
Connection conn = null;
PreparedStatement stmt = null;
//Las credenciales utilizadas son de ejemplo, se recomienda utilizar contraseñas robustas
try {
    conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/user", "root", "root");
    stmt = conn.prepareStatement(query);
    stmt.setString(1, username);
    stmt.setString(2, password);
    ResultSet rs = stmt.executeQuery();
    if (rs.next()) {
        // Login Successful if match is found
        success = true;
    }
    rs.close();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        stmt.close();
        conn.close();
    } catch (Exception e) {
    }
}
```

Entendamos lo que está pasando en este caso.

Consulta : seleccione * de tbluser donde nombre de usuario =? y contraseña = ?

El signo de interrogación (?) en la consulta anterior se denomina parámetro posicional. Hay 2 parámetros posicionales en la consulta anterior. No concatenamos nombre de usuario y contraseña para consultar. Usamos métodos disponibles en PreparedStatement para proporcionar información de usuario.

Hemos configurado el primer parámetro usando stmt.setString(1, username) y el segundo parámetro usando stmt.setString(2, password). La API de JDBC subyacente se encarga de desinfectar los valores para evitar la inyección de SQL.

Deshabilitar phpinfo

La función phpinfo() en PHP muestra información detallada sobre la configuración de PHP instalada en el servidor, incluyendo la versión de PHP, módulos cargados, variables de entorno y configuración del servidor. Si esta función se deja activa en un servidor web público, puede ser una vulnerabilidad de seguridad.

Esto se debe a que los atacantes pueden utilizar la información expuesta en la función phpinfo() para encontrar vulnerabilidades conocidas en la versión de PHP, los módulos instalados o la configuración del servidor. Los atacantes pueden utilizar esta información para desarrollar ataques específicos que exploten las vulnerabilidades conocidas.

Por lo tanto, es una buena práctica deshabilitar la función phpinfo() en servidores web públicos o restringir el acceso a esta función solo a usuarios de confianza. Se recomienda también mantener actualizada la versión de PHP y sus módulos instalados para reducir el riesgo de vulnerabilidades conocidas.

phpinfo()

PHP Version 5.5.9-1ubuntu4.29

System	Linux SitioWebBackup20221230 4.15.18-10-pve #1 SMP PVE 4.15.18-32 (Sat, 19 Jan 2019 10:09:37 +0100) x86_64
Build Date	Apr 22 2019 18:33:42
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-readline.ini, /etc/php5/apache2/conf.d/mcrypt.ini
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212
Zend Extension Build	API220121212,NTS
PHP Extension Build	API20121212,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, mcrypt.*, mdecrypt.*

Deshabilitar phpinfo en Linux

Abrir el archivo de configuración de PHP (php.ini).

Rutas por defecto en servidores

1. En servidores apache el archivo php.ini se encuentra en la siguiente ruta: `/etc/php/[VERSION]/apache2/`
2. En servidores Nginx el archivo php.ini se encuentra en la siguiente ruta: `/etc/php/[VERSION]/fpm`

Una vez encontrado el archivo php.ini se debe realizar lo siguiente:

Buscar la línea que contiene la directiva `"disable_functions"` y agregue `"phpinfo"` a la lista separada por comas de funciones deshabilitadas.

```
disable_functions = phpinfo
```

Guarde el archivo php.ini y reinicie su servidor web para que los cambios surtan efecto.

Reiniciar Apache:

Abre una línea de comandos en el servidor.
Ejecuta el siguiente comando para reiniciar el servicio de Apache:

```
//para versiones modernas de Linux que usan systemd

sudo systemctl restart apache2

//para versiones antiguas de Linux

sudo service apache2 restart
```

Reiniciar Nginx:

Abre una línea de comandos en el servidor.
Ejecuta el siguiente comando para reiniciar el servicio de Nginx:

```
//para versiones modernas de Linux que usan systemd
```

```
sudo systemctl restart nginx


//para versiones antiguas de Linux


sudo service nginx restart
```

Deshabilitar phpinfo en Windows

Ubicación del archivo php.ini en Windows utilizando XAMPP

1. Abre el cmd.
2. Luego dirigirse a la carpeta donde esta instalada PHP en su unidad c. Por ejemplo:

```
cd c: xamppphp
```

Una vez encontrado el archivo php.ini se debe realizar lo siguiente:

Buscar la línea que contiene la directiva `"disable_functions"` y agregue `"phpinfo"` a la lista separada por comas de funciones deshabilitadas.

```
disable_functions = phpinfo
```

Guarde el archivo php.ini y reinicie su servidor web para que los cambios surtan efecto.

Reiniciar WIndows con XAMPP instlado:

1. Abre la consola de XAMPP desde el menú de inicio o buscando la aplicación en la carpeta de instalación.
2. En la consola de XAMPP, detén los servicios de Apache y MySQL haciendo clic en el botón "Stop" para cada uno de ellos. Asegúrate de que ambos servicios estén detenidos antes de continuar.
3. Una vez que se hayan detenido los servicios, cierra la consola de XAMPP.
4. Abre el menú Inicio de Windows y haz clic en el botón de "Apagar" para mostrar las opciones de apagado.
5. Selecciona "Reiniciar" para reiniciar el servidor.
6. Espera a que el servidor se reinicie completamente y vuelve a abrir la consola de XAMPP.
7. En la consola de XAMPP, inicia los servicios de Apache y MySQL haciendo clic en el botón "Start" para cada uno de ellos. Asegúrate de que ambos servicios estén iniciados antes de continuar.
8. Verifica que tus sitios web o aplicaciones estén funcionando correctamente.

Cross-Site Scripting XSS

XSS Cross-Site Scripting (o "Inyección de scripts entre sitios", en español). Es un tipo de vulnerabilidad de seguridad en aplicaciones web, donde un atacante puede insertar código malicioso (como JavaScript) en una página web, que luego se ejecutará en el navegador de un usuario que visite esa página.

Los ataques XSS ocurren cuando una aplicación web no valida correctamente las entradas de los usuarios, permitiendo que un atacante inserte código malicioso en una página web que se servirá a otros usuarios. El código malicioso puede ser diseñado para robar información personal, redirigir a los usuarios a sitios web maliciosos, mostrar anuncios no deseados, o incluso para tomar control del navegador del usuario.

Existen dos tipos principales de ataques XSS:

Reflejado: El ataque XSS reflejado ocurre cuando el código malicioso se ejecuta en la página web después de que un usuario hace una solicitud a la aplicación web. El código malicioso se "refleja" de vuelta al usuario a través de la respuesta de la aplicación web.

Almacenado: El ataque XSS almacenado ocurre cuando el código malicioso se almacena en la base de datos de la aplicación web y se ejecuta cada vez que se solicita la página web afectada.

Los desarrolladores pueden prevenir ataques XSS mediante la validación y filtrado de las entradas de los usuarios, utilizando bibliotecas de seguridad como Content Security Policy (CSP) y asegurándose de que todas las entradas de los usuarios se escapen de forma adecuada antes de ser utilizadas en una página web.

Sanitización de XSS en PHP

Aquí hay un ejemplo de cómo mitigar el riesgo de XSS en PHP utilizando la función `"htmlspecialchars()"` para escapar las salidas de usuario:

```
<?php
$param = "<a href='test'>Test</a>";
$valid = htmlspecialchars($param, ENT_QUOTES, 'UTF-8');

echo $valid // <a href='test'>Test</a>
```

```
?>
```

En este ejemplo, estamos utilizando la función `"htmlspecialchars()"` para escapar los caracteres especiales HTML en la entrada del usuario. La función toma tres argumentos: la cadena de entrada a escapar, la opción `ENT_QUOTES` para escapar tanto comillas dobles como comillas simples, y la codificación de caracteres 'UTF-8'.

Sanitización XSS en Laravel

Este método fue probado para Laravel 7, y consiste en crear un middleware para sanitizar las entradas.

El middleware proporciona un mecanismo conveniente para inspeccionar y filtrar las solicitudes HTTP que ingresan a su aplicación.

Para crear un middleware se debe aplicar el siguiente comando en la raíz del proyecto de Laravel:

```
php artisan make: middleware XssSanitizer
```

Editar el archivo app/Http/Middleware/XssSanitizer.php para que quede de la siguiente manera:

```
<?php
namespace App\Http\Middleware;
use Closure;
use Illuminate\Http\Request;

class XssSanitizer
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle(Request $request, Closure $next)
    {
        $input = $request->all();
```

```
        array_walk_recursive($input, function(&$input) {
            $input = strip_tags($input);
        });
        $request->merge($input);
        return $next($request);
    }
}
```

Ahora es necesario agregar la ruta de XssSanitizer.php al vector $routeMiddleware ubicado en app/Http/Kernel.php:

```
protected $routeMiddleware = [

'auth' => \App\Http\Middleware\Authenticate::class,

....

'XssSanitizer' => \App\Http\Middleware\XssSanitizer::class,
];
```

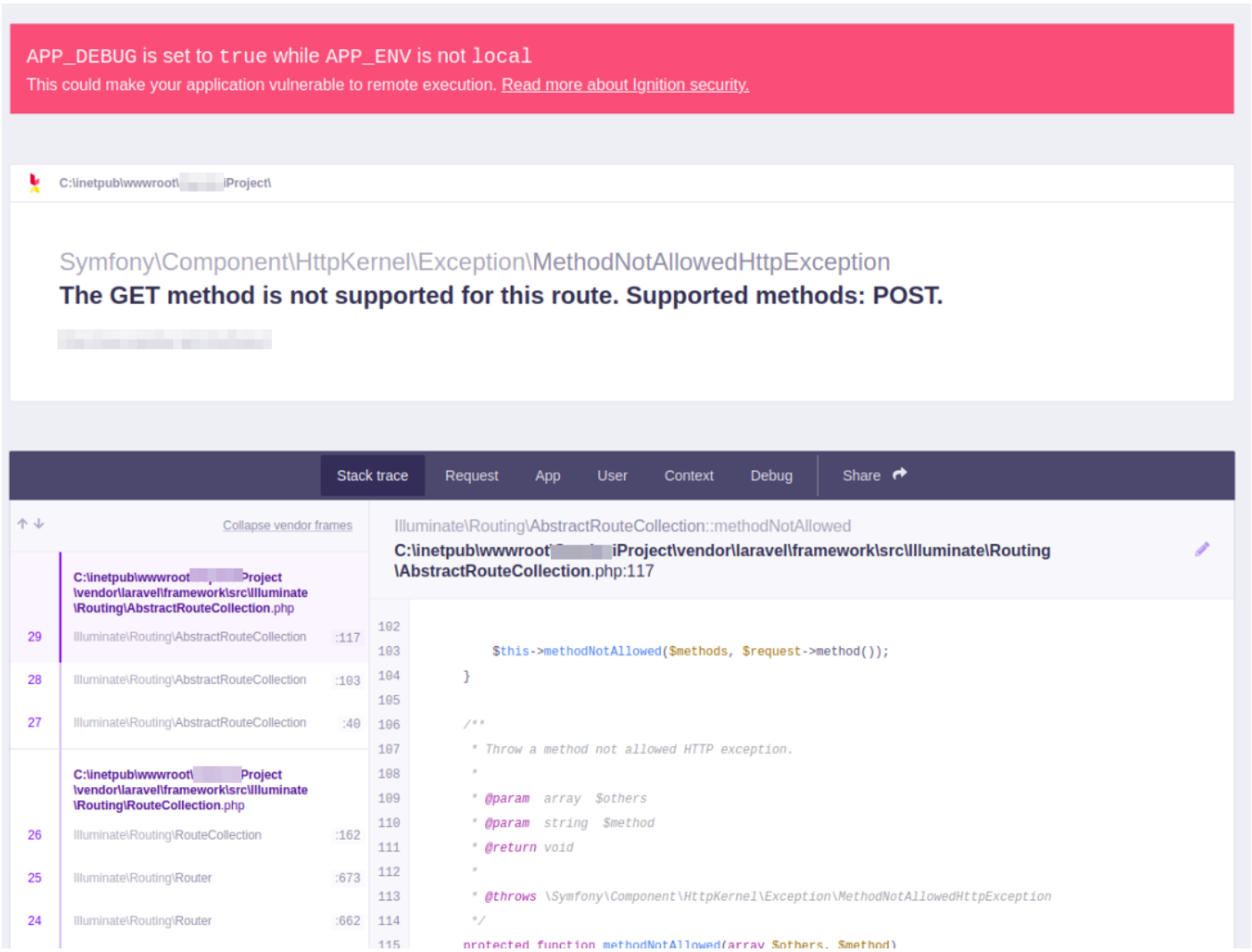
Una vez realizado esto, ya se puede utilizar XssSanitizer middleware en las rutas para realizar la sanitización:

```
Route::group(['middleware' => ['XssSanitizer']], function () {

Route::get('/', function () {
        return view('welcome');
    });
    Route::get('/formulario',function () {
        return view('formulario');
    });

    Route::get('form-get', function (Illuminate\Http\Request $request)
    {
        return  $request->input('buscar');
    })->name('form-get');

});
```

Despliegue de mensajes de error

El software genera un mensaje de error que incluye información confidencial sobre su entorno, usuarios o datos asociados.

La información confidencial puede ser información valiosa por sí misma (como una contraseña, nombres de usuario), o puede ser útil para lanzar ataques dirigidos.



Un atacante puede usar el contenido de los mensajes de error para ayudar a lanzar otro ataque más enfocado.

Deshabilitar el despliegue de errores en PHP

El manual de PHP recomienda deshabilitar "display_errors" en servidores expuestos a Internet.

Para PHP 5.2.4 y superior, la configuración `"display_errors"` en el archivo de configuración `"php.ini"` debe establecerse en `"stderr"` (flujo de salida de error), en lugar de "`stdout`" (flujo de salida enviado a los clientes).

```
display_errors = stderr
```

Para versiones anteriores, `"display_errors"` es un tipo booleano y se puede establecer en `"False"` para desactivarlo. La configuración también se puede deshabilitar en tiempo de ejecución usando `ini_set()` desde dentro de un script PHP.

```
display_error = False
```

Deshabilitar el despliegue de errores en Framework Yii 2.0

Yii incluye embebido un Error Exception Handler que hace del manejo de errores una experiencia mucho más llevadera.

El manejo de excepciones está habilitado por defecto. Se puede deshabilitar definiendo la constante global `"YII_ENABLE_ERROR_HANDLER"` a “`"false"` en el script de entrada (Entry Scripts) de la aplicación.

```
YII_ENABLE_ERROR_HANDLER = false
```

Variables de entorno en PHP

Una variable de entorno es un valor dinámico que se almacena en el sistema operativo y que puede ser utilizado por diferentes aplicaciones y procesos en un sistema informático. Estas variables contienen información que puede ser utilizada por programas y scripts para personalizar su comportamiento y configuración.

Las variables de entorno se establecen y se gestionan a nivel del sistema operativo, y están disponibles para cualquier programa que se ejecute en el sistema. Estas variables pueden contener información como la ubicación de ciertos archivos o directorios, el nombre del usuario actual, la configuración regional o cualquier otra información que se considere útil para el funcionamiento de las aplicaciones.

Las variables de entorno son útiles para personalizar y automatizar el comportamiento de los programas y scripts, y también pueden ser utilizadas para compartir información entre diferentes aplicaciones y procesos en el sistema.

Variables de entorno en Apache

Opción a)

Puede agregar la siguiente instrucción desde el archivo de configuración que se encuentra en `/etc/apache2/sites-available`

```
<Files archivo.extension>
order allow,deny
deny from all
</Files>
```

Opción b)

Puede agregar la siguiente instrucción desde archivos .htaccess:

```
<FilesMatch
"\.(engine|inc|install|make|module|profile|po|sh|.*sql|theme|twig|tpl(\.php)?|xtmpl|yml)(~|\.sw[o
p]|\.bak|\.orig|\.save)?$|^(\.(?!well-
```

```
known).*|Entries.*|Repository|Root|Tag|Template|composer\.(json|lock)|web\.config)$|^#.*#$|\.php(
~|\.sw[op]|\.bak|\.orig|\.save)$">
  <IfModule mod_authz_core.c>
    Require all denied
  </IfModule>
  <IfModule !mod_authz_core.c>
    Order allow,deny
  </IfModule>
</FilesMatch>
```