Trabajo Práctico 1° Cuatrimestre 2022
Investigación de vulnerabilidades y contramedidas
Ciberseguridad K3011

Grupo N° 6	
Apellido y Nombre	**Legajo**
Antelo Facundo	171.477-6
Arakaki Adrian Dario	156.165-0
Crespi Ian	167.397-0
Laureano Enrique	171.477-6

Control de cambios.

Entrega 1 - Selección de la aplicación

Selección de la aplicación

- Aplicación Suitecrm, proveedor Salesagility

  En las versiones anteriores a 7.12.2 la aplicación es vulnerable:

  1. mediante XSS lo que permite al atacante introducir JavaScript a través de la carga de archivos adjuntos.
  2. mediante inyección de SQL
  3. mediante la ejecución de código remoto a través del archivo Log File Name

  Links:

  https://www.cvedetails.com/cve/CVE-2021-45903/

  https://www.cvedetails.com/cve/CVE-2021-42840/

  https://www.cvedetails.com/cve/CVE-2021-45041/

- Aplicación Rosario Student Information System (RosarioSiS), proveedor RosarioSiS

En las versiones anteriores a 8.1.1 la aplicación es vulnerable mediante inyección de SQL (también presenta Broken Access Control) en la cual los atacantes pueden correr comandos PostgreSQL.

En versiones anteriores y con CVE reportados en 2020, la aplicación era vulnerable a través de distintos XSS

Links: https://www.cvedetails.com/cve/CVE-2021-44427/

- Aplicación SO harmonyos, proveedor Huawei

  La aplicación posee las siguientes vulnerabilidades:

  1. mediante SQL Injection

2. tiene una mala gestion de privilegios por lo cual, los atacantes puede generar un bypass y explotar esta vulnerabilidada
3. los proveedores no verificados de aplicaciones para telefonos huawei, pueden obtener datos del dispositivo y falsificar interfaces de usuario para ejecutar comandos maliciosos

Links:

https://www.cvedetails.com/cve/CVE-2021-39978/

https://www.cvedetails.com/cve/CVE-2021-22376/

https://www.cvedetails.com/cve/CVE-2021-22403/

Selección de la vulnerabilidad

Introducción

La vulnerabilidad elegida es en la aplicación Rosario Student Information System, es de tipo SQL Injection y nos permite ejecutar código de PostgreSQL (SELECT, INSERT, DELETE, UPDATE) en el Side.php a través del parámetro “syear”.

Contramedida: La aplicación no tenía controles de acceso y los usuarios no autenticados podrán ejecutar código a través del navegador.
La contramedida para solucionar esto (No se conoce si su implementación fue realizada por dicha aplicación en un futuro release) fue crear dichos controles de acceso (los IFs necesario que debe pasar cada usuario) para poder ejecutar dicho código y que no sea accesible para cualquier usuario.

Infraestructura

Requisitos mínimos para instalar RosarioSIS:

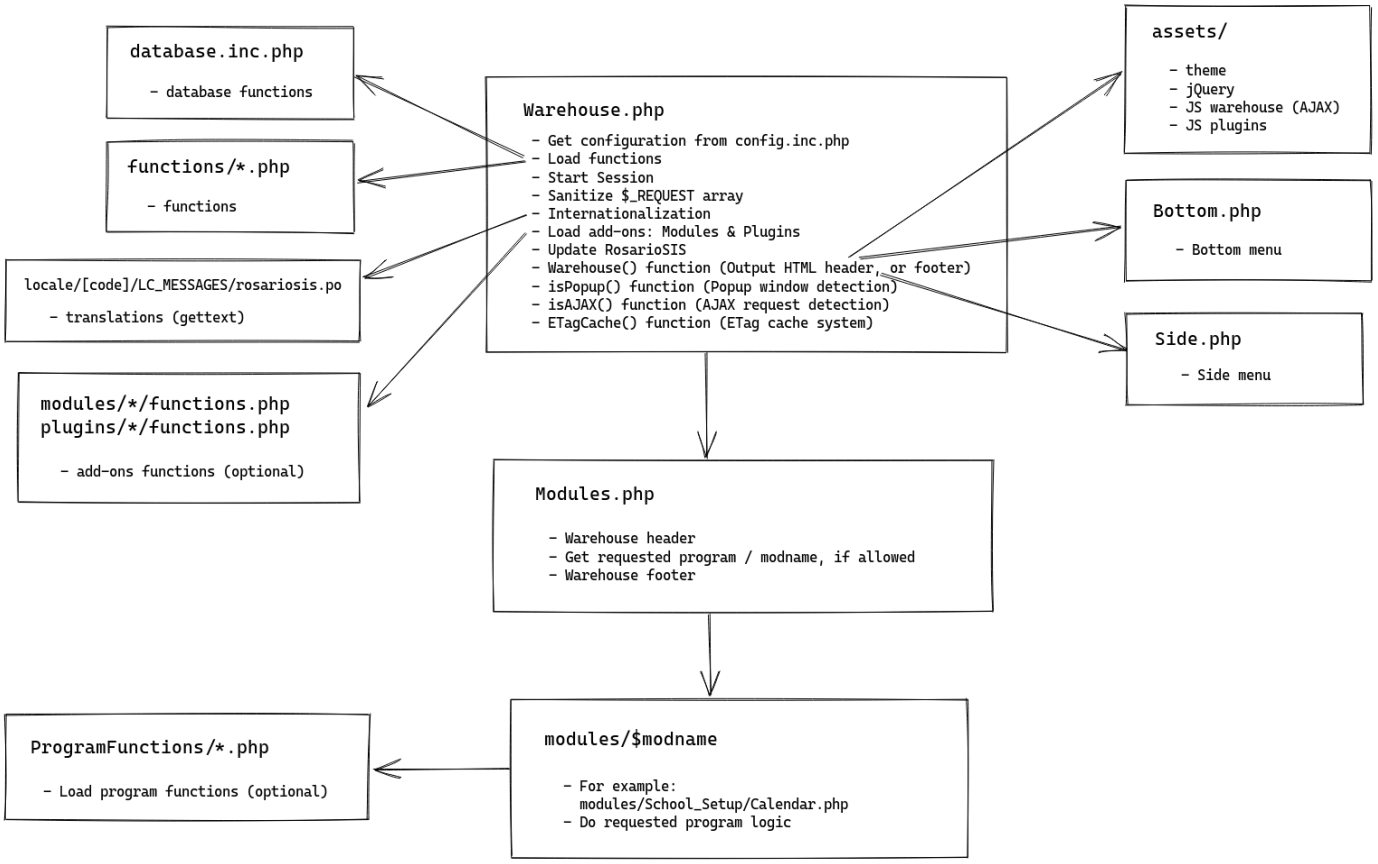
- PHP 5.4.45
- PostgreSQL 8.4
- Servidor web con apache y nginx.

Las tecnologías web que utiliza son:

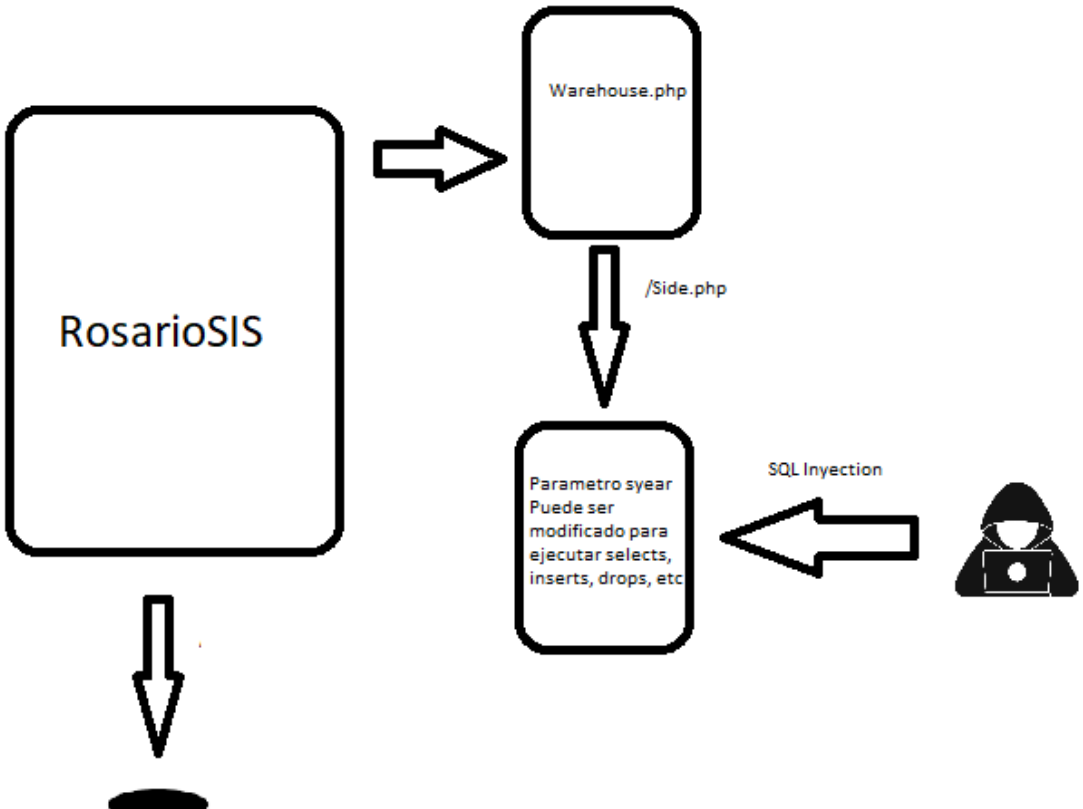
- HTML5
- CSS3
- Aplicación en AJAX (JQuery).

Para la realización del trabajo práctico, se instalará Rosariosis 8.1, PHP 7.4.21 y PostgreSQL 13.3.

Arquitectura de las carpetas y archivos



Escenario de ataque

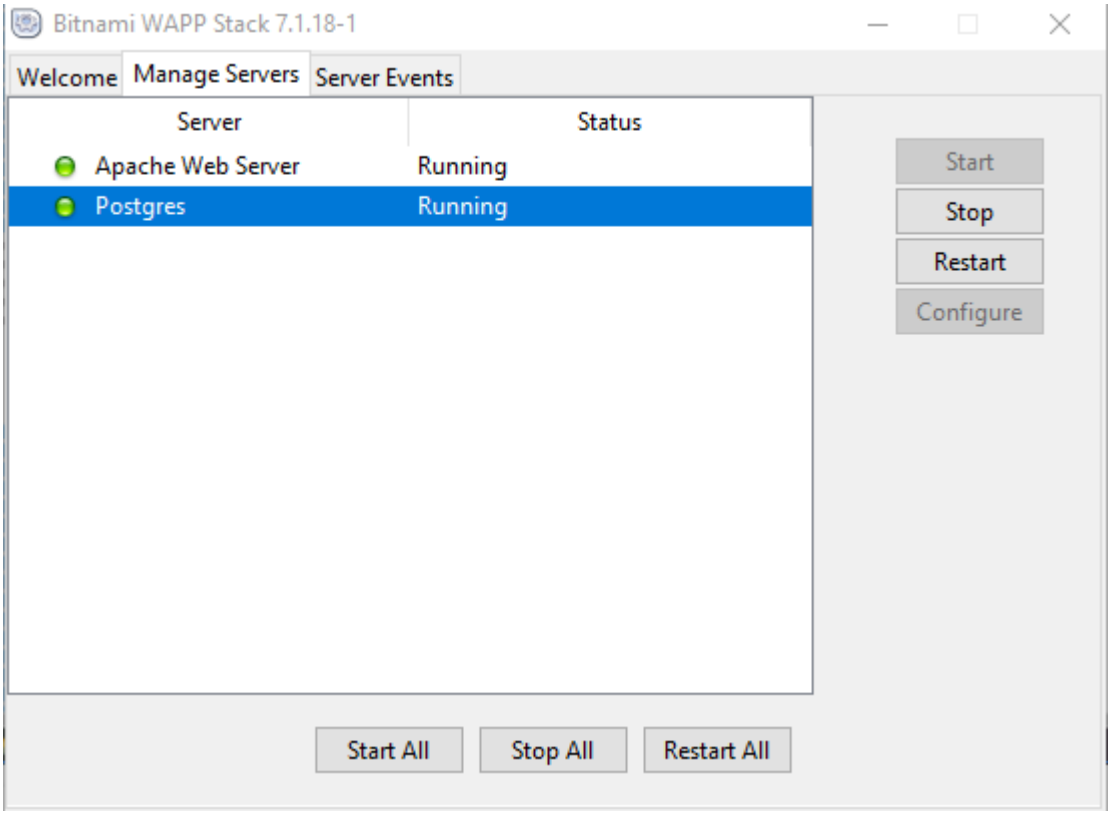


Entrega 3

Preparación del entorno de trabajo:

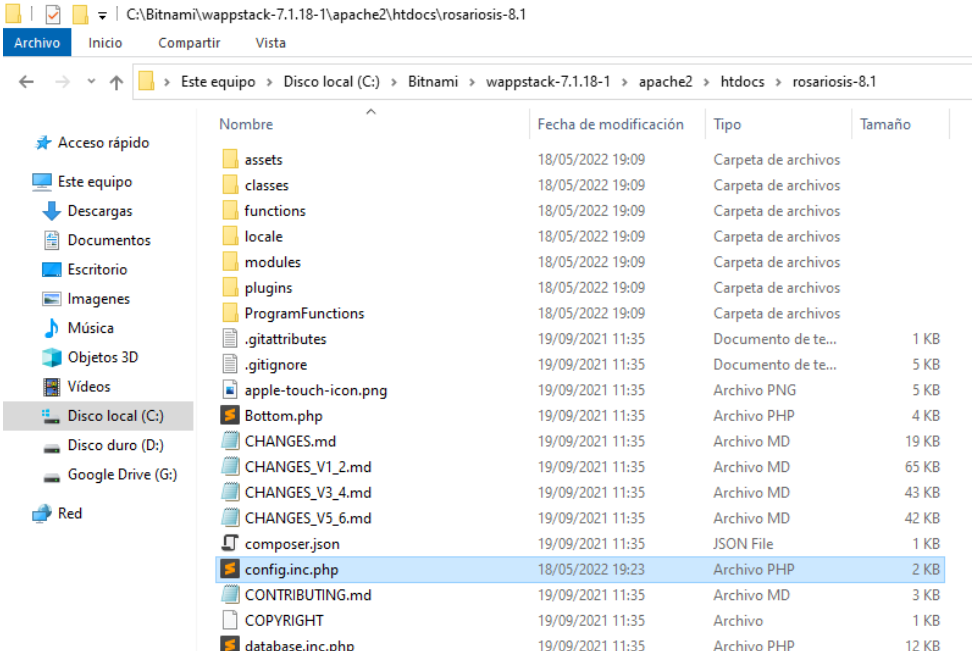
Capturas de pantalla del procedimiento y el entorno de trabajo listo:

Paso 1:



Se levanta el servidor de la base de datos PostgresQL.

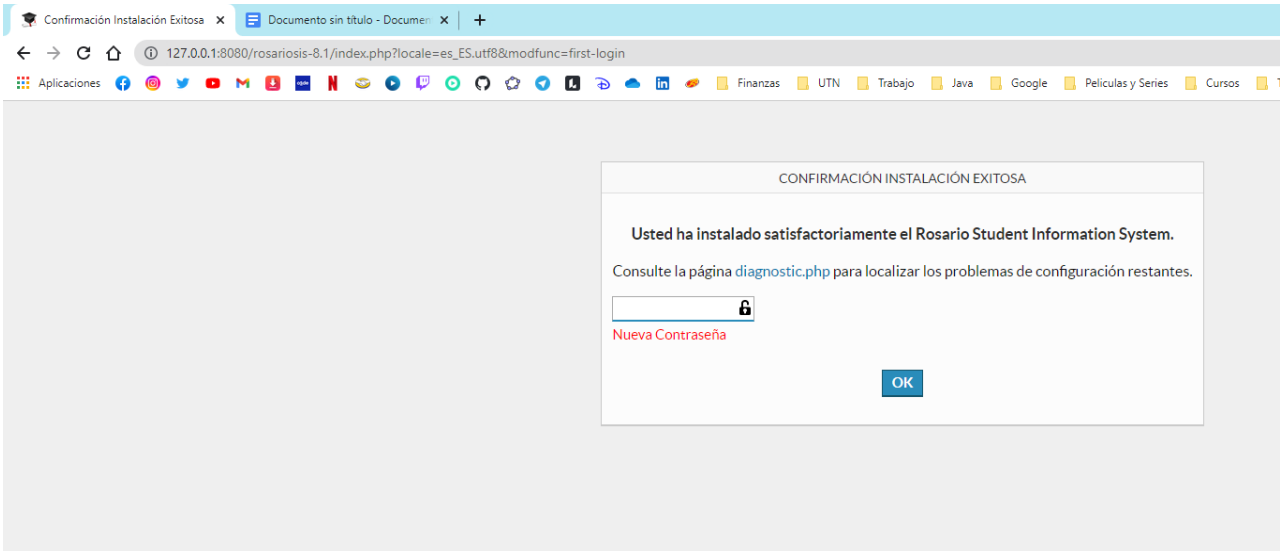
Paso 2: Se configura el archivo php con los puertos, nombre de la base de datos, usuario y password, para ingresar luego.

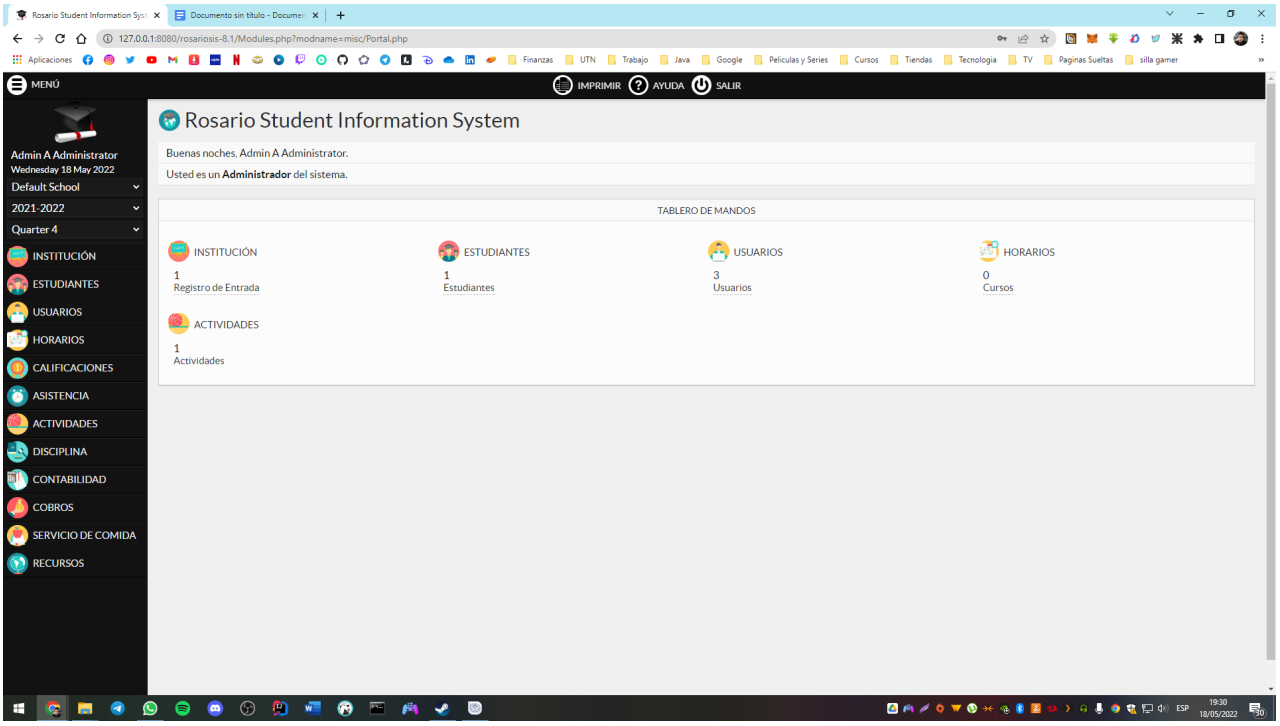


config.inc.php

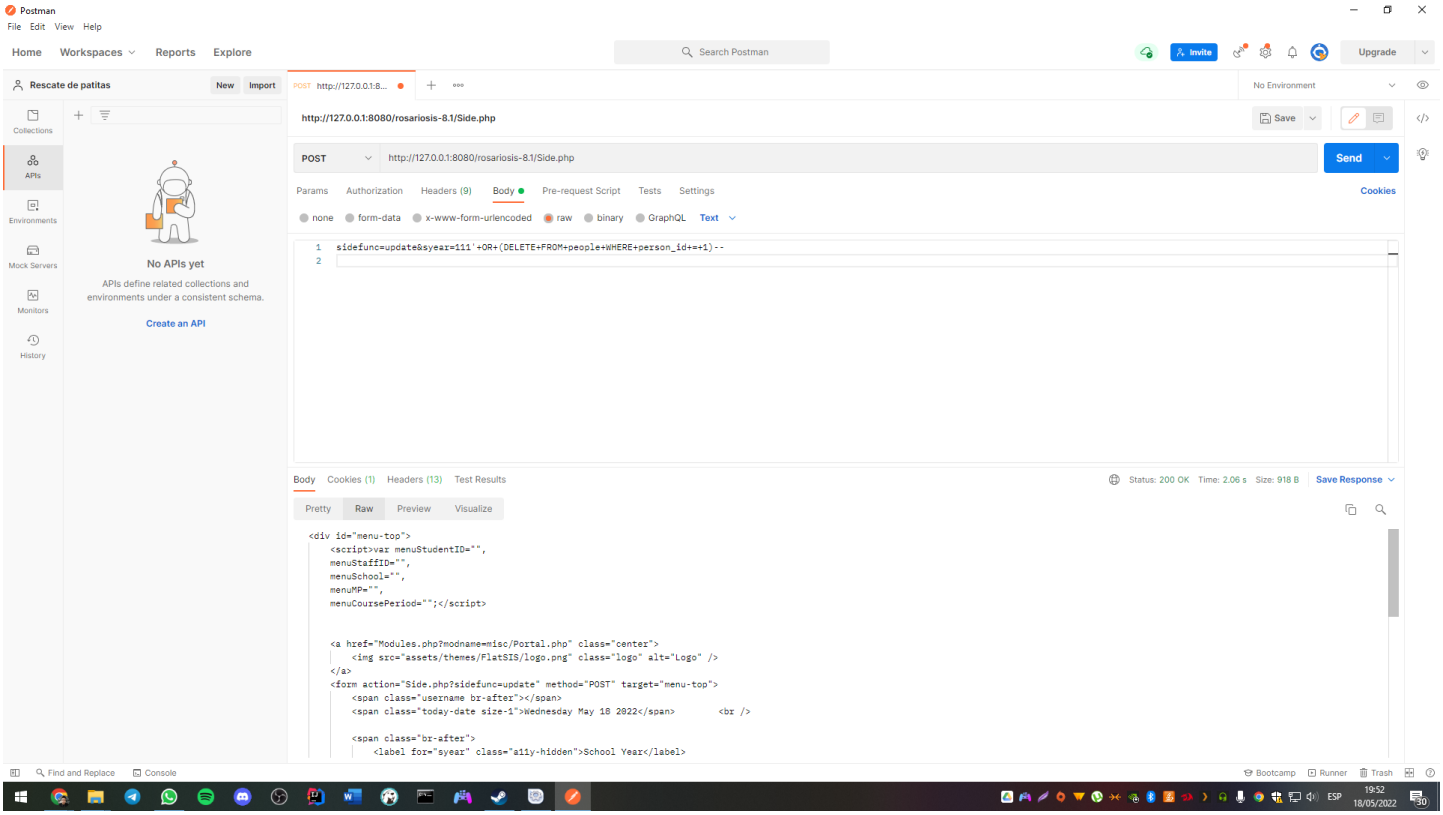
```php
<?php
/**
 * The base configurations of RosarioSIS
 *
 * You can find more information in the INSTALL.md file
 *
 * @package RosarioSIS
 */

/**
 * PostgreSQL Database Settings
 *
 * You can get this info from your web host
 */

// Database server hostname: use localhost if on same server.
$DatabaseServer = 'localhost';

// Database username.
$DatabaseUsername = 'postgres';

// Database password.
$DatabasePassword = 'root';

// Database name.
$DatabaseName = 'rosariosis';

// Database port: default is 5432.
$DatabasePort = '5432';
```

Paso 3: Se loguea y se ingresa a la aplicación Rosario Student System information

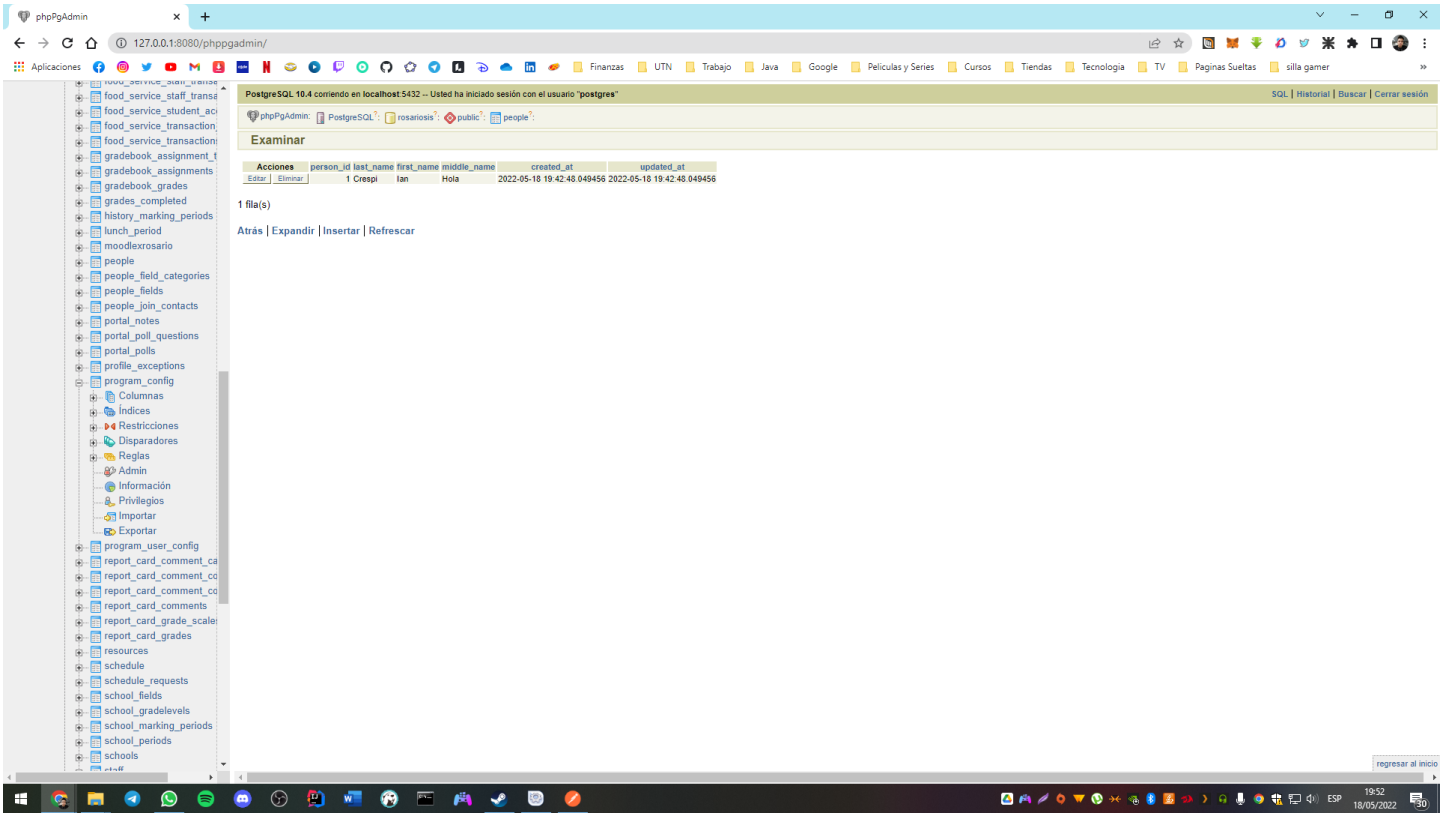




Paso 4: Se deja preparado el post man para realizar peticiones POST a la dirección de la pagina en el parámetro /Side.php



Paso 5: Se ingresa a la base de datos pertinente del sistema y se crean plantillas de ejemplo para observar si hay cambios luego de la inyección de código SQL.



Entrega 4

Contramedidas SQL Injection:

Existen dos contramedidas principales a la hora de combatir el SQL Injection:

- Desinfectar la entrada del usuario

Al estar utilizando una versión anterior de PHP (5.4.45) es aconsejable ejecutar toda la entrada del usuario a través de una función llamada mysql_real_escape_string(). Básicamente, lo que hace es eliminar todos los caracteres especiales de una cadena para que pierdan su significado cuando los utilice la base de datos.

Por ejemplo, si tiene una cadena como " I'm a string ", un atacante puede utilizar el carácter de comilla simple (') para manipular la consulta de la base de datos que se está creando y provocar una inyección SQL. Ejecutarlo a través mysql_real_escape_string() produce " I\'m a string ", que agrega una barra invertida a la comilla simple, evitándola. Como resultado, la cadena completa ahora se pasa como una cadena inofensiva a la base de datos, en lugar de poder participar en la manipulación de consultas.

Hay un inconveniente con esta contramedida: es una técnica realmente antigua que acompaña a las formas más antiguas de acceso a bases de datos en PHP. A partir de PHP 7, esta función ya no existe, lo que nos lleva a nuestra próxima solución…

- Utilizar declaraciones preparadas y parámetros enlazados

Una declaración preparada es una característica que se utiliza para ejecutar las mismas (o similares) declaraciones SQL repetidamente con alta eficiencia.

Las declaraciones preparadas básicamente funcionan así:

1- Preparar: se crea una plantilla de declaración SQL y se envía a la base de datos. Ciertos valores se dejan sin especificar, llamados parámetros (etiquetados como "?"). Ejemplo: INSERT INTO MyGuests VALUES(?, ?, ?)
2- La base de datos analiza, compila y realiza la optimización de consultas en la plantilla de declaración SQL y almacena el resultado sin ejecutarlo.
3- Ejecutar: en un momento posterior, la aplicación vincula los valores a los parámetros y la base de datos ejecuta la instrucción. La aplicación puede ejecutar la sentencia tantas veces como quiera con diferentes valores.

En comparación con la ejecución directa de sentencias SQL, las sentencias preparadas tienen tres ventajas principales:

-Las declaraciones preparadas reducen el tiempo de análisis ya que la preparación de la consulta se realiza solo una vez (aunque la declaración se ejecuta varias veces)

-Los parámetros vinculados minimizan el ancho de banda al servidor, ya que necesita enviar solo los parámetros cada vez, y no toda la consulta.

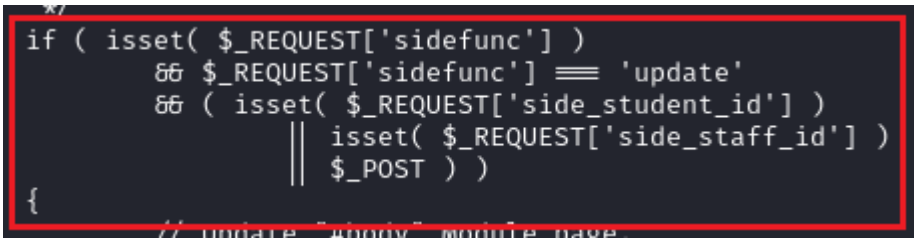
-Por ultimo y mas relevante en relación a nuestro TP, las declaraciones preparadas son muy útiles contra las inyecciones de SQL, porque los valores de los parámetros, que se transmiten más tarde usando un protocolo diferente, no necesitan secuencias de escape ('\'). Si la plantilla de declaración original no se deriva de una entrada externa, no se puede producir la inyección SQL.
Resumiendo, las declaraciones preparadas son una forma de realizar consultas a bases de datos de manera más segura y confiable. La idea es que en lugar de enviar la consulta sin procesar a la base de datos, primero le decimos a la base de datos la estructura de la consulta que enviaremos. Esto es lo que queremos decir con "preparar" una declaración. Una

vez que se prepara una declaración, pasamos la información como entradas parametrizadas para que la base de datos pueda "llenar los vacíos" conectando las entradas a la estructura de consulta que enviamos antes. Esto elimina cualquier SQL Injection que puedan tener las entradas, lo que hace que se trate solamente como variables en todo el proceso.

Detalles técnicos de la vulnerabilidad:

El archivo Side.php (además de otros archivos en RosarioSIS), no implementa un control de acceso o gestiones de comprobación de sesiones para prevenir que cualquier usuario sin autenticación pueda ejecutar código accediendo directamente a través de un navegador.

Dado que no se implementan verificaciones de administración de sesión, visitar ***/Side.php*** a través de un navegador hace que las ***líneas 85 - 90*** de ***/Warehouse.php*** se ejecuten desde ***la línea 10*** de ***/Side.php*** , que es un fragmento de código que carga archivos php dentro de las ***funciones. /*** directorio del servidor web



el if en verifica los parámetros controlados por el usuario ($_REQUEST, $_POST), un usuario no autenticado podría crear y proporcionar una solicitud que pasará la verificación condicional y, en última instancia, controlará el flujo de ejecución.

Esto se puede lograr a través de una petición post del siguiente tipo:

POST /rosariosis-v8.1/Side.php
sidefunc=update&syear=111';INSERT+INTO+STAFF+(SYEAR,STAFF_ID,"first_name","last_name","username","password","profile","profile_id")+values('2021','4','Hack','Erist','hackerist','$6$6c0b492b4c64bc9e$sgHUN45tj7jyn72f94o4w/7rSh)

Esta petición nos permite ingresar en la tabla de “staff” un campo nuevo que nos permitirá logearnos como staff dentro del sistema.