Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

Sección de alumnos: artículo original

Vulnerabilidad: SQLi

Vulnerability: SQLi

Integrantes:

Martín Nicolás Greco
Ingeniería Informática - Universidad Nacional de La Matanza
martgreco@alumno.unlam.edu.ar

Facundo Gabriel Jimenez
Ingeniería Informática - Universidad Nacional de La Matanza
fajimenez@alumno.unlam.edu.ar

Maximiliano Leonel Morales
Ingeniería En Informática. Universidad Nacional de La Matanza
maxmorales@alumno.unlam.edu.ar

Nicolás Federico Sierra
Ingeniería En Informática - Universidad Nacional de La Matanza
nsierra@alumno.unlam.edu.ar

Referentes Institucionales:

Mg. Jorge Eterovic
Universidad Nacional de La Matanza
eterovic@unlam.edu.ar

Martín Zeballos
Universidad Nacional de La Matanza
mzeballos@unlam.edu.ar

Aníbal Pose
Universidad Nacional de La Matanza
apose@unlam.edu.ar

Maximiliano Downar
Universidad Nacional de La Matanza
adownar@unlam.edu.ar

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

Resumen:

En este trabajo, analizaremos la vulnerabilidad SQL Inyección UNION Attack, la misma se basa en la inyección de código malicioso en sentencias SQL no correctamente generadas en nuestro programa con el fin de alterar el normal funcionamiento del mismo. Con la cláusula UNION podemos extender la sentencia y determinar información sensible de la base de datos como ser la cantidad de columnas y el tipo de datos de las mismas, con el fin de posteriormente, realizar nuestro ataque.

Abstract:

In this work, we will analyze the SQL Injection UNION Attack vulnerability. It is based on the injection of malicious code in SQL statements not correctly generated in our program to alter its normal functioning. With the UNION clause we can extend the statement and determine sensitive information about our database, such as the number of columns and their data type, to later carry out our attack.

Palabras Clave: *SQL, UNION, Vulnerabilidad SQLi, Base de Datos*

Key Words: *SQL, UNION, Vulnerability SQLi, Database*

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

I. CONTEXTO

El presente documento se enmarca en el área de la ciberseguridad, específicamente en la temática de las vulnerabilidades en aplicaciones web relacionadas con bases de datos. La técnica de **SQL Injection**, particularmente su variante conocida como **UNION Attack**, constituye una de las amenazas más comunes y críticas en el ámbito de la seguridad informática. El desarrollo de este documento está alineado con los objetivos de instituciones como la **Open Web Application Security Project (OWASP)**, que promueve la educación y concienciación sobre riesgos en aplicaciones web, y la **National Institute of Standards and Technology (NIST)**, que provee estándares de seguridad para la protección de sistemas informáticos. La investigación, se realiza como trabajo de la materia Seguridad en Redes de la carrera Ingeniería en Informática, dentro de la Universidad Nacional de La Matanza.

II. INTRODUCCIÓN

En el ámbito de la ciberseguridad, las vulnerabilidades en aplicaciones web representan un desafío persistente para desarrolladores y organizaciones. Una de las amenazas más significativas es el **SQL Injection**, técnica utilizada por atacantes para manipular consultas SQL a través de entradas no validadas. Este método permite el acceso no autorizado a bases de datos, comprometiendo la confidencialidad, integridad y disponibilidad de la información. Entre las variantes de este ataque, el **UNION Attack** destaca por su capacidad para combinar resultados de múltiples consultas y extraer información confidencial de manera discreta.

El **UNION Attack** aprovecha la cláusula UNION SELECT de SQL para unir resultados de consultas legítimas con datos adicionales provenientes de entradas maliciosas. Esta técnica requiere un conocimiento avanzado del esquema de la base de datos objetivo, lo que subraya la importancia de implementar medidas robustas de seguridad en las aplicaciones, tales como la validación de entradas y el uso de consultas parametrizadas. Según OWASP, el SQL Injection se encuentra entre las principales amenazas de seguridad para aplicaciones web desde hace varios años, lo que resalta su relevancia en el contexto actual [1].

Este trabajo explora en profundidad la técnica de **UNION Attack**, sus principios de funcionamiento, ejemplos prácticos y las estrategias de mitigación más efectivas. Además, se analizan casos documentados en la literatura, con base en los aportes de autores destacados en la

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

materia, como Halfond et al. [2], quienes enfatizan el impacto de estas vulnerabilidades en la integridad de los sistemas. Asimismo, se hace referencia a las guías prácticas propuestas por OWASP y NIST para abordar este tipo de ataques de manera eficaz.

La investigación tiene como objetivo no solo describir los aspectos técnicos del **UNION Attack**, sino también fomentar el desarrollo de aplicaciones web más seguras mediante el uso de prácticas recomendadas en la industria.

III. MÉTODOS

Para analizar y comprender en profundidad la técnica de SQL Injection UNION Attack, y en particular su aplicación en el contexto del laboratorio "Retrieve multiple values in a single column" de PortSwigger, se empleará una metodología basada en experimentación práctica, análisis técnico y revisión bibliográfica. Los métodos empleados se describen a continuación.

A. Configuración del Entorno de Pruebas

1. *Acceso al Laboratorio:*

   Se utilizará el laboratorio proporcionado por PortSwigger, disponible en el enlace https://portswigger.net/web-security/sql-injection/union-attacks/lab-retrieve-multiple-values-in-single-column. Este entorno simula una aplicación web vulnerable que permite realizar pruebas controladas de SQL Injection sin comprometer sistemas reales.

2. *Herramientas de Análisis:*

   - Navegador Web: Se usará un navegador compatible para interactuar con la aplicación y enviar payloads manualmente.

B. Exploración de la Vulnerabilidad

1. *Identificación del Punto de Inyección:*

   Se analizarán las solicitudes HTTP enviadas por la aplicación para identificar parámetros susceptibles a inyección SQL. Esta fase incluirá la inserción de caracteres especiales (', ", --, etc.) para determinar cómo la aplicación maneja las entradas de usuario.

2. *Determinación del Número y Tipo de Datos de Columnas:*

   Utilizando la cláusula ORDER BY o la técnica UNION SELECT NULL, se determinará el número de columnas en la consulta SQL que se ejecuta en el backend. Para determinar

el tipo de dato se utiliza la técnica de reemplazar alguno de los valores NULL por un string literal o algún número, por ejemplo:

'UNION Select NULL, 'test' --

3. *Inserción del Payload:*

   Una vez identificado el esquema de la consulta, se enviarán payloads que utilicen UNION SELECT para recuperar múltiples valores en una sola columna. Este enfoque permitirá extraer datos sensibles del sistema simulado, como nombres de usuarios, correos electrónicos u otros elementos representativos.

C. Documentación y Análisis de Resultados

1. *Registro de Evidencias:*

   Cada paso del ataque se documentará con capturas de pantalla, las cuales servirán para ilustrar claramente el proceso y los resultados obtenidos.

2. *Interpretación Técnica:*

   Se realizará un análisis técnico de las respuestas del servidor para explicar cómo la vulnerabilidad permite la recuperación de información, destacando las fallas de seguridad explotadas.

D. Propuesta de Mitigaciones

Se identificarán las mejores prácticas y medidas de seguridad para prevenir esta clase de ataques, basándose en recomendaciones de OWASP.

Con esta metodología, se espera no solo comprender el funcionamiento de la técnica UNION Attack aplicada en este laboratorio, sino también desarrollar propuestas para fortalecer la seguridad en aplicaciones web frente a este tipo de vulnerabilidades.

IV. RESULTADOS Y OBJETIVOS

Resultados Alcanzados

Durante la ejecución del laboratorio proporcionado por PortSwigger sobre la técnica **SQL Injection UNION Attack**, se obtuvieron los siguientes resultados:

A. Determinación del Número de Columnas

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

A través de la utilización de la cláusula UNION SELECT con valores NULL, se identificó que la consulta en la base de datos involucraba dos columnas. Este hallazgo se confirmó al probar con una cantidad creciente de columnas hasta recibir una respuesta sin errores desde el servidor.

Para poder determinar esto se realizaron los siguientes pasos:

Por empezar, se ingresó al laboratorio de PortSwigger y se seleccionó la categoría “Pets”. Esta acción actualizara la dirección web en el navegador. Luego, se identificará cuantas columnas tiene la tabla que se desea atacar.

Para ello se utiliza la operación de UNION y por cada columna potencial que puede llegar a tener la tabla de la base de datos se hace un select con el valor NULL, con una comilla simple al inicio y -- al final. El doble guion (--) es un indicador de comentario en SQL, por lo cual, lo que venga después de esa query SQL va a ser ignorado. El sentido de utilizar la operación de UNION es que justamente `une` los contenidos por mismo tipo de dato y misma cantidad de tabla (en su orden específico). A continuación, se especifica el código SQL que se debe agregar al final de la dirección del navegador:

‘UNION Select NULL --

Fig. 1. error en cantidad de columnas

Tal como se ilustra en la figura 1, el resultado de aplicar el código SQL fue un error. Esto indica que la tabla de la base de datos no tiene una sola columna, sino que posee más columnas. A continuación, se repitió la misma sentencia SQL pero con un nuevo null para contemplar la posibilidad de que haya dos columnas en la tabla:

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

'UNION Select NULL, NULL --

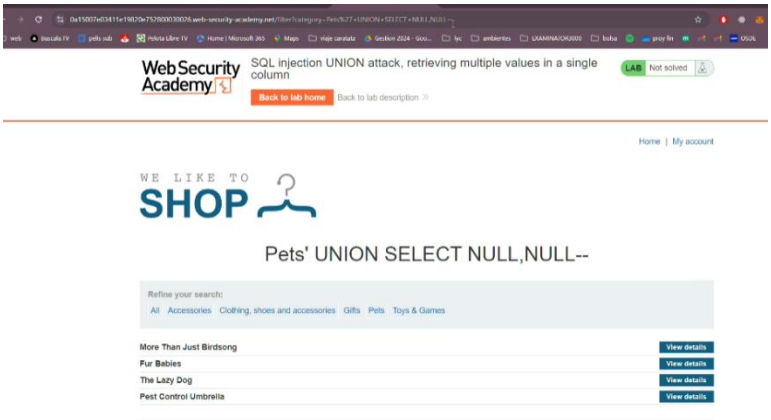

Fig. 2. Cantidad de columnas exactas de la tabla

Tal como se evidenció en la anterior ilustración, no se arrojó ningún error al aplicar el código SQL, con lo cual, se confirmó que la tabla posee dos campos.

B. *Identificación de Tipos de Datos*

- Se comprobó que la primera columna es de tipo numérico. Esto se verificó al reemplazar un valor NULL por un número, obteniendo una respuesta exitosa.
- Se determinó que la segunda columna es de tipo string, lo que permitió manipularla para mostrar información textual en la respuesta del servidor.

Para realizar esto, se reemplazó alguno de los NULL por texto.

En este caso primero se probó:

'UNION Select NULL, 'test' --

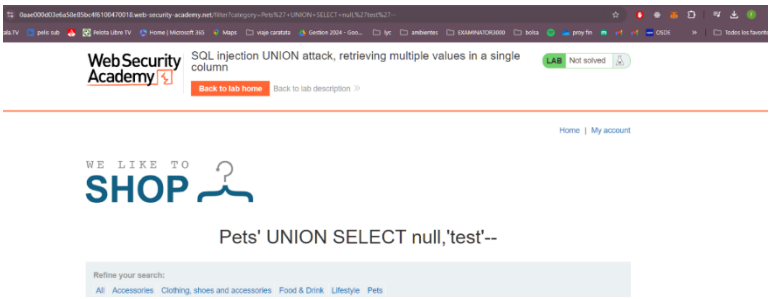

Fig 3. Prueba tipo de dato string segunda columna

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

Dado que la página devolvió información de forma correcta, se sabe que esa columna es de tipo string.

Posteriormente, se probó si la primera columna era string, pero resultó en error.

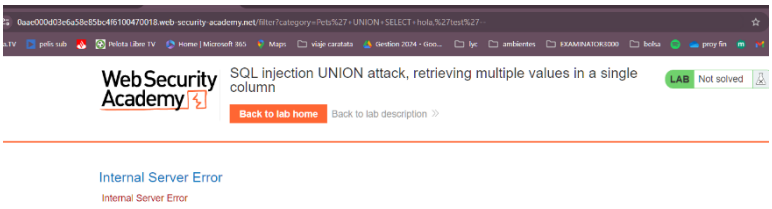

Fig. 4. Prueba tipo de dato string primer columna

A continuación, se comprobó si es el tipo de dato es numérico. Como se muestra en la figura **4**, el resultado fue correcto, lo que confirma que la primera columna es numérica.

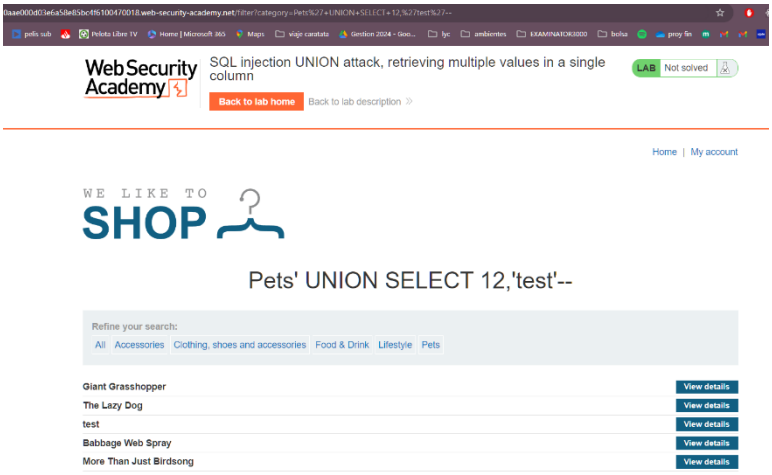

Fig. 5. Prueba tipo de dato número en primera columna y string en segunda columna

C. *Recuperación de Información Sensible*

Conociendo los nombres de las columnas username y password de la tabla users, se utilizó una consulta con el operador de concatenación || para unir los valores de ambas columnas en una sola respuesta. Al utilizar un carácter delimitador (~), se facilitó la visualización y extracción de la información sensible. Como resultado, se obtuvieron los datos de acceso de todos los usuarios, incluyendo el administrador, lo que permitió resolver el laboratorio.

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

Posteriormente, tras conocer los nombres de las columnas **username** y **password**, de la tabla users, se procedió a realizar una concatenación de dicha información.

Como es sabido, la segunda columna es de tipo de dato string, por lo que se puede armar un string específico.

Para eso se utiliza: el operador || (concatenación). En el caso expuesto se utiliza además un carácter de separación `~` para identificar usuario y contraseña.

'UNION Select NULL, username || ' ~ ' || password FROM users--

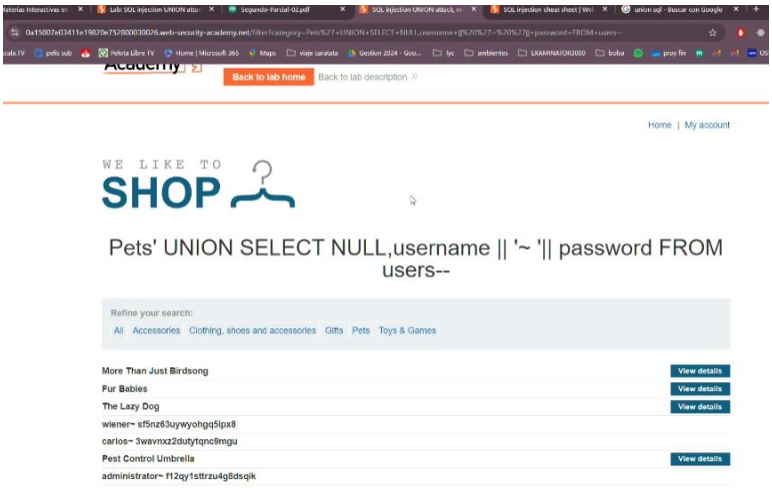

Fig. 6. Listado de credenciales recuperadas

El resultado de esa inyección SQL muestra un listado de todos los usuarios y sus respectivas contraseñas, las cuales se utilizan para resolver el laboratorio tras efectuar un inicio de sesión exitoso con alguna de las credenciales encontradas.
En nuestro caso, se ha utilizado el usuario **administrator**.

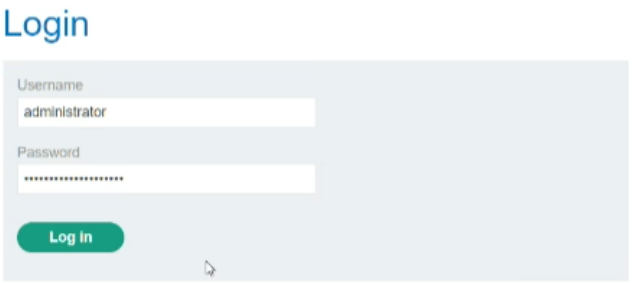

Fig. 6. Acceso a la plataforma con credencial de usuario administrator

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

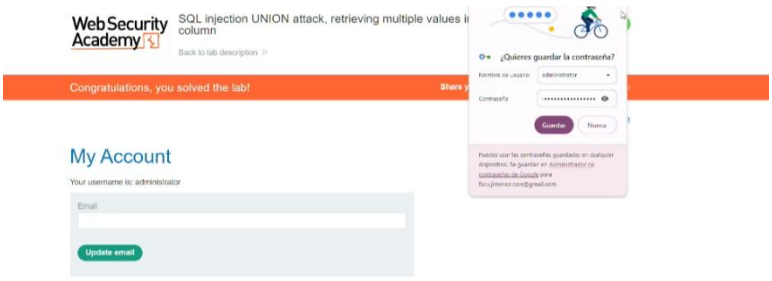

Fig. 7. Acceso exitoso a la plataforma con las credenciales provistas

D. Severidad

El tipo de vulnerabilidad analizada se ha clasificado como criticidad "critical". Tal como se vio anteriormente, un atacante externo puede extraer información sensible de la base de datos al punto de obtener las credenciales de administrador. De esta manera, existe la posibilidad de que pueda realizar cualquier acción maliciosa como borrar todas las tablas de la base de datos, afectar la integridad de los datos almacenados, eliminando y actualizando registros, entre otras acciones. La severidad no se considera "alta", sino más bien "critical", dado que esta clasificación depende mucho del contexto y los tipos de datos que almacena. Si estamos en presencia de una base de datos que almacena información personal es crítica debido a que deben cumplir con normativas como GDPR, PCI, etc. Además, esta vulnerabilidad expone totalmente el sistema y es altamente escalable en combinación con otros tipos de ataques.

E. Recomendaciones y herramientas de mitigación

En base en los resultados obtenidos, se desarrollaron guías prácticas para prevenir la explotación de vulnerabilidades mediante **UNION Attack**. Las medidas que se deberían implementar en el diseño y el desarrollo de las aplicaciones se describirán continuación:

1. *Utilizar consultas parametrizadas:*

   Es crucial evitar concatenar directamente los datos proporcionados por el usuario en textbox en las consultas SQL de nuestro sitio. Por eso, se deben utilizar consultas parametrizadas que separen el código SQL de los datos de entradas. Por ejemplo, para el caso particular que se vio, una posible solución sería implementar en el backend la siguiente sentencia:

Revista Digital del Departamento de
Ingeniería e Investigaciones Tecnológicas
de la Universidad Nacional de La Matanza
ISSN: 2525-1333

Vol.: 9 - Nro. 2 (Diciembre - 2024)

```
query = "SELECT * FROM users WHERE username = ? and password =?" cursor.execute(query, (username, password,))
```

2. *Validar los datos de entrada:*

   Los datos de entrada proporcionados por el usuario deben ser validados correctamente. Definir longitud máxima de caracteres, formato permitido, por ejemplo: solo se permiten números en identificadores. Además, se deben utilizar funciones de escape adecuadas en el caso de entradas de datos que permitan caracteres especiales.

3. *Restringir privilegios en la base de datos*

   Se deben configurar las cuentas de la base de datos con los privilegios mínimos necesarios.

4. *Controlar y limitar el uso de la cláusula UNION*

   Plantearse si realmente se necesita el uso de cláusulas UNION en nuestras consultas. Se pueden implementar restricciones como limitar el número de columnas que puede devolver una consulta.

5. *Utilizar ORMs para acceder a las bases de datos*

   Los Object-Relational Mappers (ORM) como SQLAlchemy, Entity Framework o Hibernate abstraen las consultas SQL, reduciendo significativamente el riesgo de inyecciones. Los ORM manejan automáticamente el escape de datos con lo cual son una buena opción.

6. *Configurar un WAF (Web Application Firewall)*

   Los WAF pueden detectar y bloquear patrones sospechosos de inyección SQL, incluyendo ataques de tipo UNION. Se pueden configurar reglas específicas para filtrar solicitudes maliciosas antes de que lleguen a la aplicación.

7. *Realiza pruebas de seguridad regularmente*

   Se recomienda el uso de herramientas de análisis estático y dinámico para detectar vulnerabilidades de inyección SQL. Algunas herramientas son:

   SQLMap: Para probar posibles vulnerabilidades

   Burp Suite: Para pruebas manuales o automatizadas

   OWASP ZAP: Un escáner de seguridad web gratuito

8. *Implementar limites en los resultados de las consultas*

   Se recomienda el uso de cláusulas como LIMIT para restringir el número de filas que pueden ser devueltas. Por ejemplo:

```
SELECT username, email FROM users WHERE activo = 1 LIMIT 10;
```

Los resultados alcanzados en este laboratorio proporcionan un entendimiento técnico clave para comprender la gravedad de las vulnerabilidades asociadas con **SQL Injection** y establecen la base para desarrollar estrategias de prevención más efectivas.

V. CONCLUSIONES

La investigación y experimentación realizadas en el contexto del laboratorio de PortSwigger sobre la técnica **SQL Injection UNION Attack** han permitido comprender con profundidad la manera en que las aplicaciones web pueden ser explotadas a través de entradas no validadas. Este trabajo ha evidenciado cómo una vulnerabilidad de este tipo puede comprometer la confidencialidad y la integridad de los datos almacenados en bases de datos, incluso en entornos simulados y controlados.

La determinación del número de columnas, la identificación de tipos de datos y la posterior extracción de información sensible mediante consultas SQL manipuladas demuestran la peligrosidad de no implementar medidas de seguridad adecuadas en aplicaciones web. En particular, este trabajo destacó la importancia de prácticas recomendadas como la validación

estricta de entradas y el uso de consultas parametrizadas para mitigar los riesgos asociados a este tipo de ataques.

A nivel práctico, el trabajo realizado ha demostrado cómo técnicas aparentemente simples, como el uso de la cláusula UNION SELECT, pueden ser combinadas para maximizar la efectividad de los ataques y extraer información clave.

Finalmente, se concluye que una correcta comprensión de los principios básicos de SQL Injection y sus variantes no solo permite a los profesionales de la seguridad anticiparse a las amenazas, sino que también habilita a los desarrolladores a diseñar aplicaciones más robustas frente a estos ataques. A futuro, será esencial seguir avanzando en la educación y divulgación de estas técnicas para fomentar una cultura de seguridad cibernética que priorice la prevención y el diseño seguro desde el inicio.

VI. REFERENCIAS Y BIBLIOGRAFÍA

[1] OWASP, "OWASP Top Ten 2021: The Ten Most Critical Web Application Security Risks." [En línea]. Disponible en: https://owasp.org.

[2] W. Halfond, A. Orso, y P. Manolios, "WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation," *IEEE Transactions on Software Engineering*, vol. 34, no. 1, pp. 65-81, 2008.

Recibido: 2024-08-29
Aprobado: 2024-11-06
Hipervínculo Permanente: https://doi.org/10.54789/reddi.9.2.5
Datos de edición: Vol. 9 -Nro. 2 - SA. 1
Fecha de edición: 2024-12-30