Теперь рассмотрим вторую уязвимость. Это тоже SQL-инъекция. Обнаружена она была в модуле Members в скрипте восстановления забытых паролей. Находится он в папке "modules/Members/" и называется "send_pass.php". С помощью уязвимости, о которой мы сейчас будем говорить, можно сменить пароль у любого аккаунта. Но отличие от первой уязвимости здесь в том что тогда мы меняли пароль на произвольный, а сейчас мы будем менять пароль на случайный.

Для наших экспериментов зарегистрируйте 2 аккаунта: аккаунт Warrior с email`ом wwwwwww@yandex.ru и аккаунт Via с email`ом via@rambler.ru (данный аккаунт будет играть роль цели). Регистрируйте аккаунты именно с этими почтовыми адресами, ниже я объясню для чего это нужно. Сначала подробно рассмотрим все действия скрипта который отвечает за восстановление пароля.

Что бы восстановить забытый пароль скрипт предлагает ввести либо логин либо email. Как только Вы нажимаете кнопку “Напомнить” управление передаётся скрипту send_pass.php . Что же он делает? Вот интересующие нас куски кода (строк 20-31):

```
$username = trim(strip_tags($_POST['username']));
$user_email = trim(strip_tags($_POST['user_email']));
if(($username != "") AND ($user_email == "")) {
$where_dat = "username = '$username'";
} else if(($username == "") AND ($user_email != "")) {
$where_dat = "user_email = '$user_email'";
} else if(($username != "") AND ($user_email != "")) {
$where_dat = "username = '$username' AND user_email = '$user_email'";
} else if(($username == "") AND ($user_email == "")) {
header("Location: index.php?go=Members&in=lost_pass");
exit;
}
```

Здесь происходит следующее: сначала в переменные $username и $user_email заносятся данные которые ввёл пользователь на страничке восстановления пароля. Фильтрации здесь практически никакой так как функция trim() убирает с начала и с конца строки пробелы и некоторые спец-символы, а функция strip_tags() удаляет только html/php-тэги. Если Вы указали имя пользователя и e-mail то в переменную $where_dat заносится текст
username=’введённое_имя_пользователя’ AND user_email=’введённый_email’
Если же введён только логин то в переменную $where_dat заносится
username=’введённое_имя_пользователя’
Если ввести только email то происходит почти то же самое только по отношению к email`у – в переменную $where_dat заносится текст “user_email=’введённый_email’”. Сразу видно что формируется конструкция условий для SQL-запроса. Сам же этот запрос находится в строке 32:

```
$sql = "SELECT * FROM ".$conf['member_prefix']."_members WHERE $where_dat";
```

Этот запрос возвратит все данные о пользователе которого Вы указали. Сначала может показаться что можно вывести какую-либо информацию при супециально сформированном e-mail`е (например добавить конструкцию UNION) но это не так. Дело в том что данные которые Вы ввели идут дальше и встраиваются в следующий запрос. А он уже изменяет таблицу пользователей. Если в первом или втором запросах произойдёт ошибка то скрипт прекратит работу и Вы ничего не увидите.

Рассмотрим всё по порядку. После выполнения первого запроса в переменную "$to" заносится email из полученного результата:

```
$to = $row['user_email'];
```

И далее идёт ещё один запрос, без которого восстановление не пройдёт. Строка 43:

```
$db->sql_query("UPDATE ".$conf['member_prefix']."_members SET user_actkey='$check_num',
user_newpasswd='$new_code_password'  WHERE $where_dat");
```

Вот здесь и находится главная загвоздка — во второй запрос тоже вставляется содержимое переменной $where_dat. Получается что мы должны сформировать такие данные которые бы подошли к первому запросу и выдали бы нужный результат, к тому же эти данные должны нормально обработаться и во втором запросе. То есть нам придётся подобрать такие данные которые бы свободно могли находиться и в SELECT и в UPDATE-запросах, а сделать это иногда крайне сложно.

Идём дальше. Во втором запросе переменные $check_num и $new_code_password генерируются произвольно и независимо от нас. Здесь в таблицу пользователей вставляется ключ активации нового пароля и сам новый пароль. То есть пользователь, прежде чем использовать новый пароль, должен его активировать. Получается, что если провалится этот запрос то ничего не сработает, и пароль не сменится. Либо не придёт письмо, либо не поменяется пароль. Но давайте взглянем на оба запроса в той ситуации, когда мы введём только email, например via@rambler.ru:

```
SELECT * FROM sn_members WHERE user_email=’via@rambler.ru’
UPDATE sn_members SET user_actkey='0', user_newpasswd='0'  WHERE user_email=’via@rambler.ru’
```

Вроде бы ничего особенного. Но как я уже говорил – с помощью специально сформированного запроса можно сменить пароль жертвы на произвольный. А жертва у нас – это аккаунт Via. Как же эксплуатировать данную уязвимость? Эксплуатируется она следующим образом: давайте представим что за место email`а мы введём следующий текст:
wwwwwww@yandex.ru’ OR username=’Via’/*
Код запросов станет следующий (код после знака комментария уже убран):

```
SELECT * FROM sn_members WHERE user_email=’ wwwwwww@yandex.ru’ OR username=’Via’/*
```

Этот запрос выдаст 2 email`а – первый от аккаунта где email wwwwwww@yandex.ru, а второй от аккаунта, у которого имя пользователя - Via.
Взглянем на второй запрос:
UPDATE sn_members SET user_actkey='0', user_newpasswd='0' WHERE user_email=' wwwwwww@yandex.ru' OR username='Via'/*
Этот запрос установит новый пароль и ключ активации сразу на 2 аккаунта – аккаунт с email`ом wwwwwww@yandex.ru и аккаунт с именем пользователя Via. Получается что после выполнения данного запроса у обоих аккаунтов будет одинаковый пароль и одинаковый ключ активации. А теперь я объясню зачем у аккаунта Warrior такой странный email. При первом запросе письмо отправится именно на тот email который будет первым в результате запроса. А теперь зайдите в MySQL и выполните следующий запрос:
SELECT user_email FROM sn_members WHERE user_email='wwwwwww@yandex.ru' OR username='Via';
Вы должны увидеть следующий результат:

```
+------------------+
| user_email       |
+------------------+
| wwwwwww@yandex.ru |
| via@rambler.ru    |
+------------------+
```

2 rows in set (0.00 sec)
Как видите – email атакующего вышел первым, потому что по алфавиту он стоит ниже чем email жертвы. Если же email атакующего будет выше по алфавиту то письмо отправится на email жертвы, а пароли всё равно сменятся. Итак. Давайте зайдём на страничку восстановления пароля и введём в поле email текст **wwwwwww@yandex.ru' OR username='Via'/***
Вы должны были увидеть сообщение о том что на email высланы инструкции для смены пароля. Так как в Denver`е стоит заглушка для sendmail`а то все письма, уходящие с сервера, хранятся в папке tmp/!sendmail/
Заглядываем туда и видим что появилось письмо:

Уважаемый Warrior, для Вашего аккаунта на сайте SmallPortal с IP адреса 127.0.0.1 был запрошен новый пароль!

Пожалуйста, запомните а в случае необходимости сохраните следующие данные:

Логин: Warrior
Пароль: [здесь новый пароль]

ВНИМАНИЕ! Для подтверждения смены пароля, Вам необходимо пройти по указанной ниже ссылке:
[ссылка для активации нового пароля]
Если Вы не запрашивали смену пароля, просто проигнорируйте данное сообщение.

С уважением
Администрация сайта SmallPortal
http://smallportal

Теперь дважды проходим по ссылке активации пароля потому что 2 одинаковых ключа активации установились на 2 аккаунта. После этого заходите под именем пользователя Via и паролем который пришёл Вам на Email:

Пароль сменён.