

<https://github.com/ace231/CS380-EX6>

### Setup:

```
GNU nano 2.2.6      File: /etc/php5/apache2/php.ini

; Development Value: On
; Production Value: On

; magic_quotes_gpc
; Default Value: Off
; Development Value: Off
; Production Value: Off

; max_input_time
; Default Value: -1 (Unlimited)
; Development Value: 60 (60 seconds)
; Production Value: 60 (60 seconds)

; output_buffering
; Default Value: Off
; Development Value: 4096
; Production Value: 4096

; register_argc_argv

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

```
GNU nano 2.2.6      File: /etc/php5/apache2/php.ini

; escape any character sequences in GET, POST, COOKIE and ENV data which might
; otherwise corrupt data being placed in resources such as databases before
; making that data available to you. Because of character encoding issues and
; non-standard SQL implementations across many databases, it's not currently
; possible for this feature to be 100% accurate. PHP's default behavior is to
; enable the feature. We strongly recommend you use the escaping mechanisms
; designed specifically for the database your using instead of relying on this
; feature. Also note, this feature has been deprecated as of PHP 5.3.0 and is
; scheduled for removal in PHP 6.
; Default Value: Off
; Development Value: Off
; Production Value: Off
; http://php.net/magic-quotes-gpc
magic_quotes_gpc = Off

; Magic quotes for runtime-generated data, e.g. data from SQL, from exec(), etc.
; http://php.net/magic-quotes-runtime
magic_quotes_runtime = Off

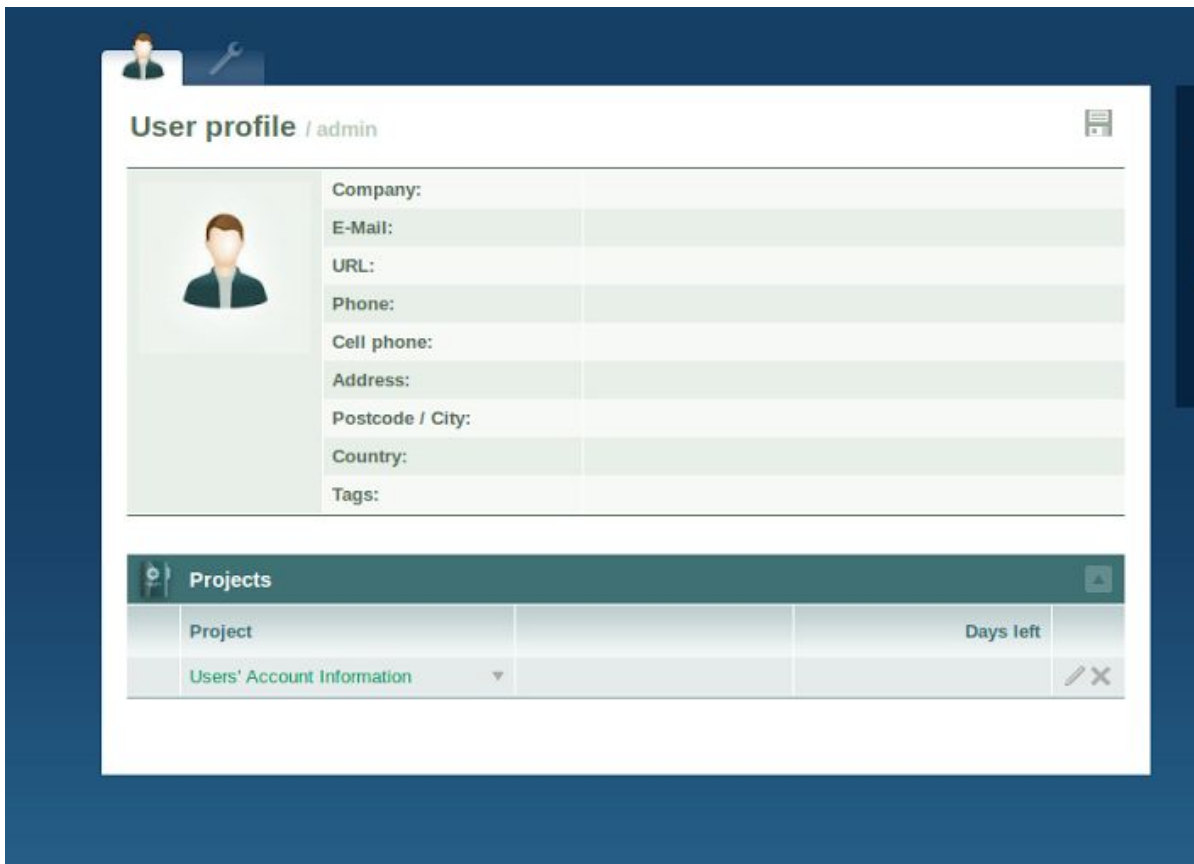
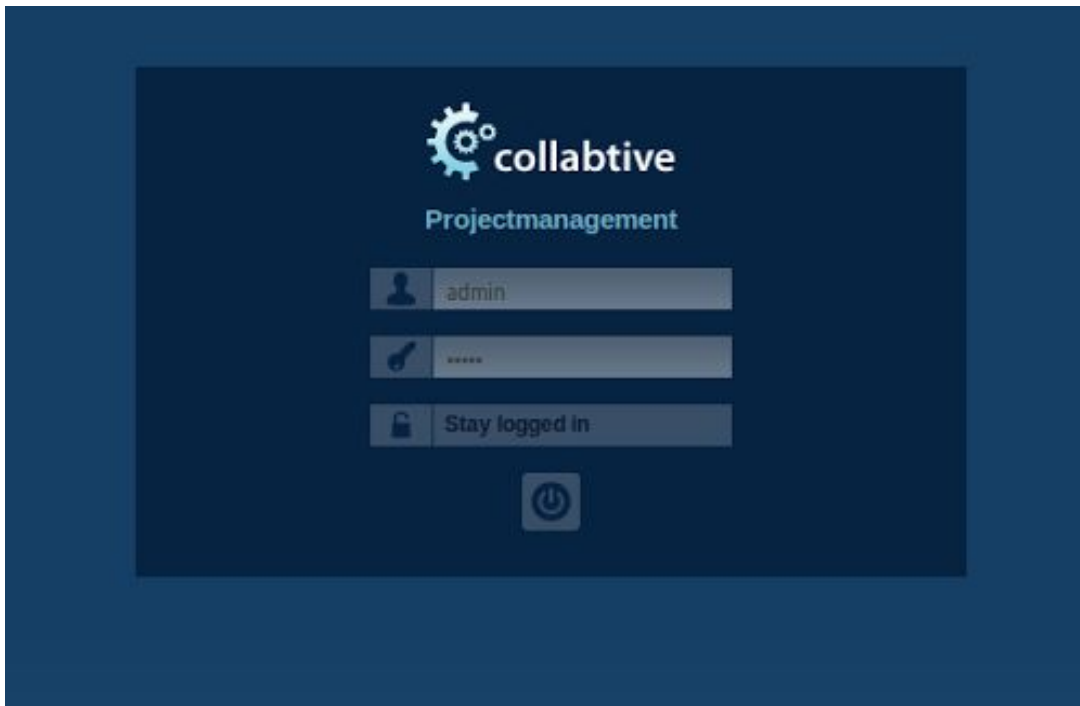
; Use Sybase-style magic quotes (escape ' with '' instead of \').
; http://php.net/magic-quotes-sybase
magic_quotes_sybase = Off

; Automatically add files before PHP document.
; http://php.net/auto-prepend-file

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Magic\_quotes\_gpc value switched to Off throughout the php.ini file.

Ran Apache command restart server and logged in as user admin with password admin



Account page of admin user

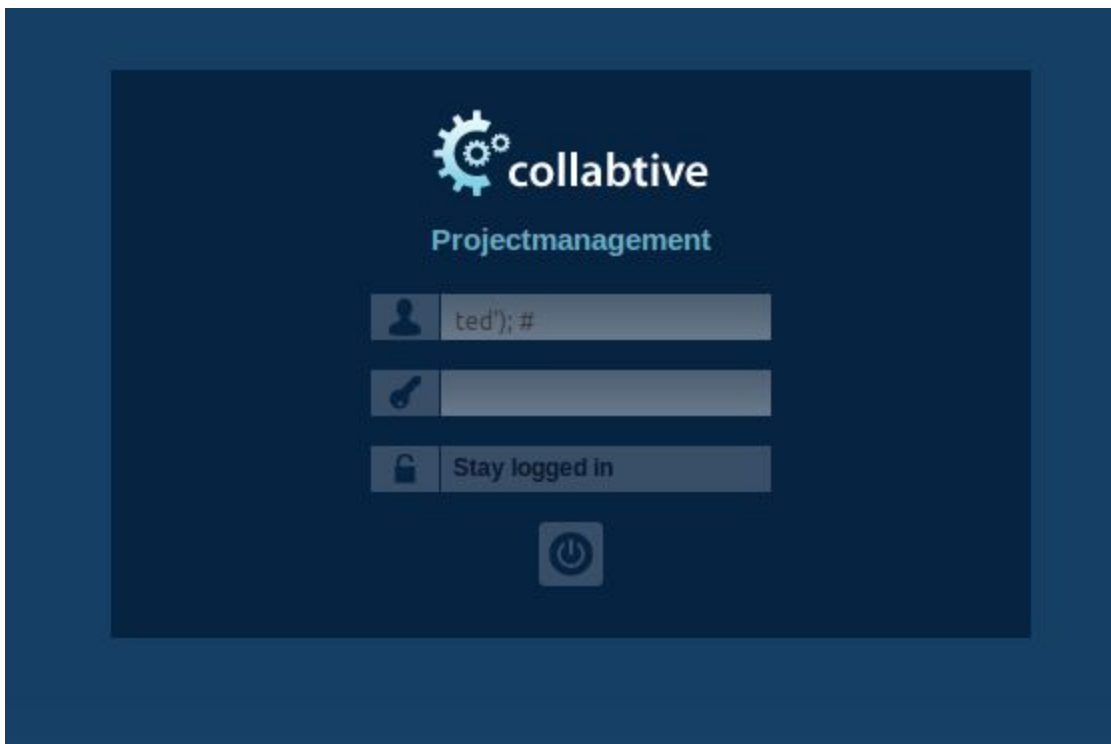
## SQL Injection in User Authentication:

The most important line in the login code has to be `WHERE (name = '$user' OR email = '$user') AND pass = '$pass'");`

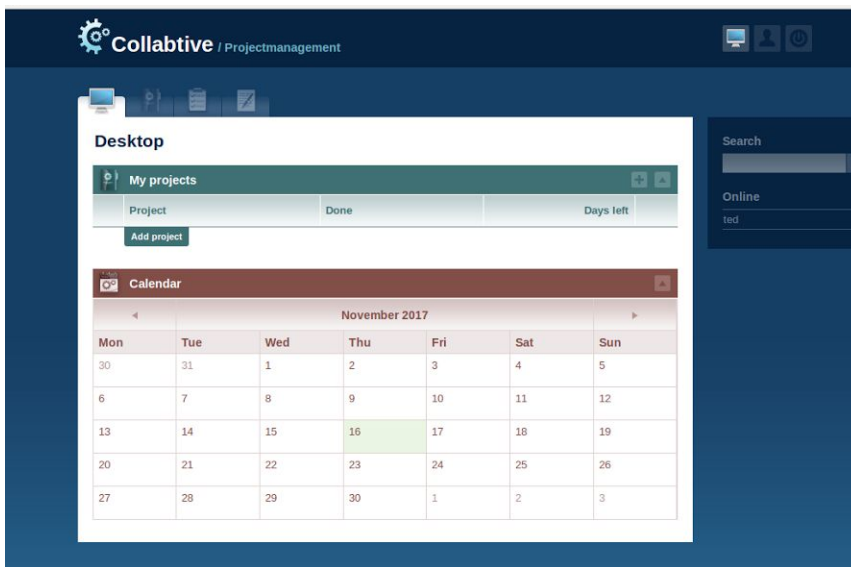
The exploit can be triggered by inputting `<username>'); #`

What happens is the PHP code will treat the input as actual code since it is not sanitizing or running any sort of check to prevent outside code from being injected into the query. When a valid username is used, the SQL query becomes `WHERE (name = '<username>'); # OR email = '$user') AND pass = '$pass'");` everything from `#` to the end of the line is ignored as a comment. The query will fetch the record of the inputted username without even needing a password. The record will be seen to exist and the login proceeds.

Here it is working for the username "ted"



Query becomes `WHERE (name = 'ted'); #` (Remember the rest becomes a comment)



Login was allowed

More proof

The screenshot shows the 'User profile' page for a user named 'ted'. The page has a dark blue header with the 'Collabtive / Projectmanagement' logo. Below the header, there's a user profile card with a placeholder image and a list of fields for user information. The fields are: Company, E-Mail, URL, Phone, Cell phone, Address, Postcode / City, Country, and Tags. Each field has a light green input area.

Company:	
E-Mail:	
URL:	
Phone:	
Cell phone:	
Address:	
Postcode / City:	
Country:	
Tags:	

The same logic works with the username "admin"

The screenshot shows the login page of the Collabtive Projectmanagement system. The page has a dark blue background with the 'collabtive Projectmanagement' logo at the top. Below the logo, there are three input fields: a username field containing 'admin'); #', a password field, and a 'Stay logged in' checkbox. A power button icon is located at the bottom center.

Query becomes `WHERE (name = 'admin'); #` (This example shows a more dangerous scenario)

The screenshot shows the desktop dashboard of the Collabtive Projectmanagement system. The dashboard has a dark blue header with the 'Collabtive / Projectmanagement' logo and a navigation bar. The main content area is divided into two sections: 'My projects' and 'Calendar'. The 'My projects' section shows a table with columns for Project, Done, and Days left. The 'Calendar' section shows a calendar for November 2017. On the right side, there is a search bar and a dropdown menu for 'Project'.

Project	Done	Days left
Users' Account Information	0%	

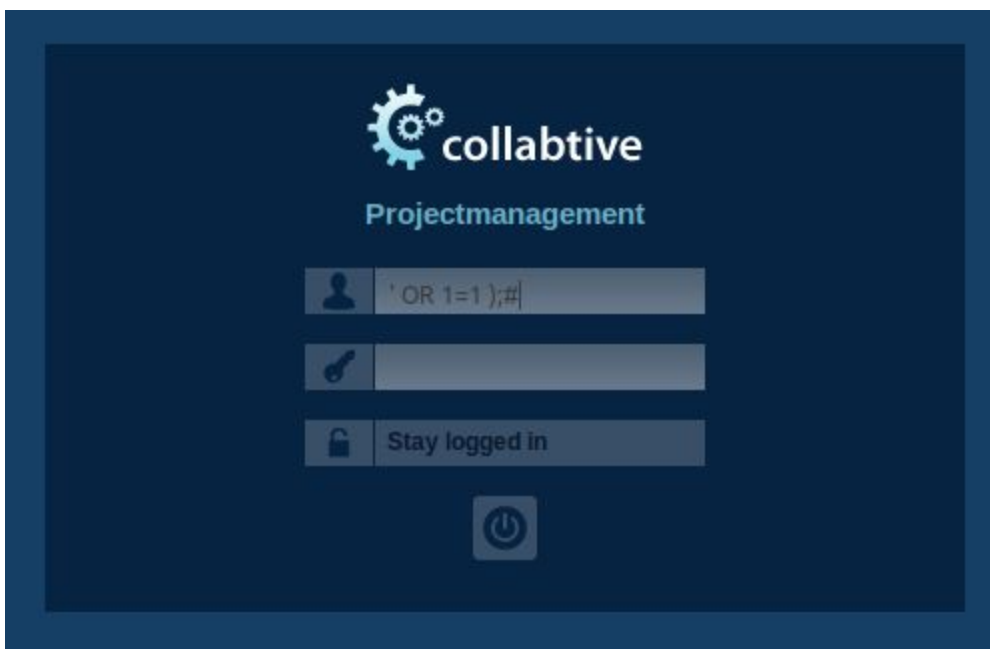
Mon	Tue	Wed	Thu	Fri	Sat	Sun
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

Now logged in as an admin user

The screenshot shows the Collabtive Projectmanagement interface. At the top, there's a dark blue header with the Collabtive logo and 'Projectmanagement' text. Below the header, there's a user profile section for an admin user. The profile includes a placeholder image and a form with fields for Company, E-Mail, URL, Phone, Cell phone, Address, Postcode / City, Country, and Tags. Below the profile, there's a 'Projects' section with a table. The table has columns for Project, Days left, and an action column. The first row shows 'Users' Account Information' with a dropdown arrow and edit/delete icons.

Project	Days left	
Users' Account Information		

These examples worked only because the inputted usernames actually existed. Would it be possible to login into the website without providing a username at all? The answer is yes.



Here the query becomes `WHERE (name = '' OR 1=1 );# OR email = '$user') AND pass = '$pass' )`; and again remember, because of the pound sign/hashtag the only valid part of the query becomes `WHERE (name = '' OR 1=1 );` this returns true no matter what, skipping the login page's user check and allowing the login process to continue... As an admin sadly enough

**Collabtive** / Projectmanagement

**Desktop**

**My projects**

Project	Done	Days left
Users' Account Information	0%	

**Calendar**

November 2017

Mon	Tue	Wed	Thu	Fri	Sat	Sun
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

**Search**

**Project**

Please choose

**Online**

admin

**Collabtive** / Projectmanagement

**User profile / admin**

**Company:**

**E-Mail:**

**URL:**

**Phone:**

**Cell phone:**

**Address:**

**Postcode / City:**

**Country:**

**Tags:**

**Projects**

Project	Days left
Users' Account Information	

And there it is, SQL may make managing data and tables easy, but if user input is not sanitized and checked properly, it opens the door to rather simple yet potentially catastrophic code injections.