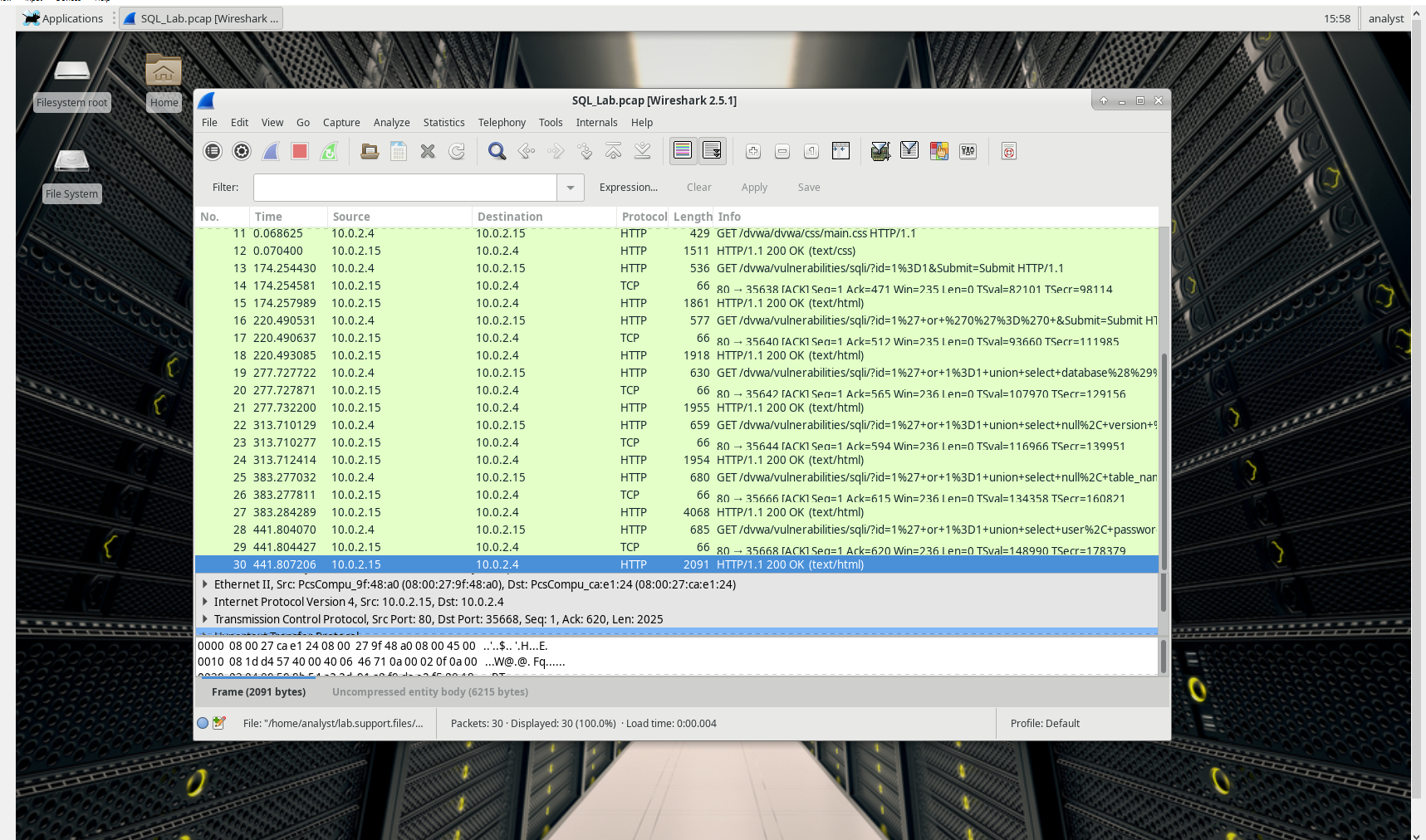
**Lab – Attacking a mySQL Database**

**Objectives In this lab, you will view a PCAP file from a previous attack against a SQL database.**

**Part 1: Open the PCAP file and follow the SQL database attacker**

**Step 1: Open Wireshark and load the PCAP file.**



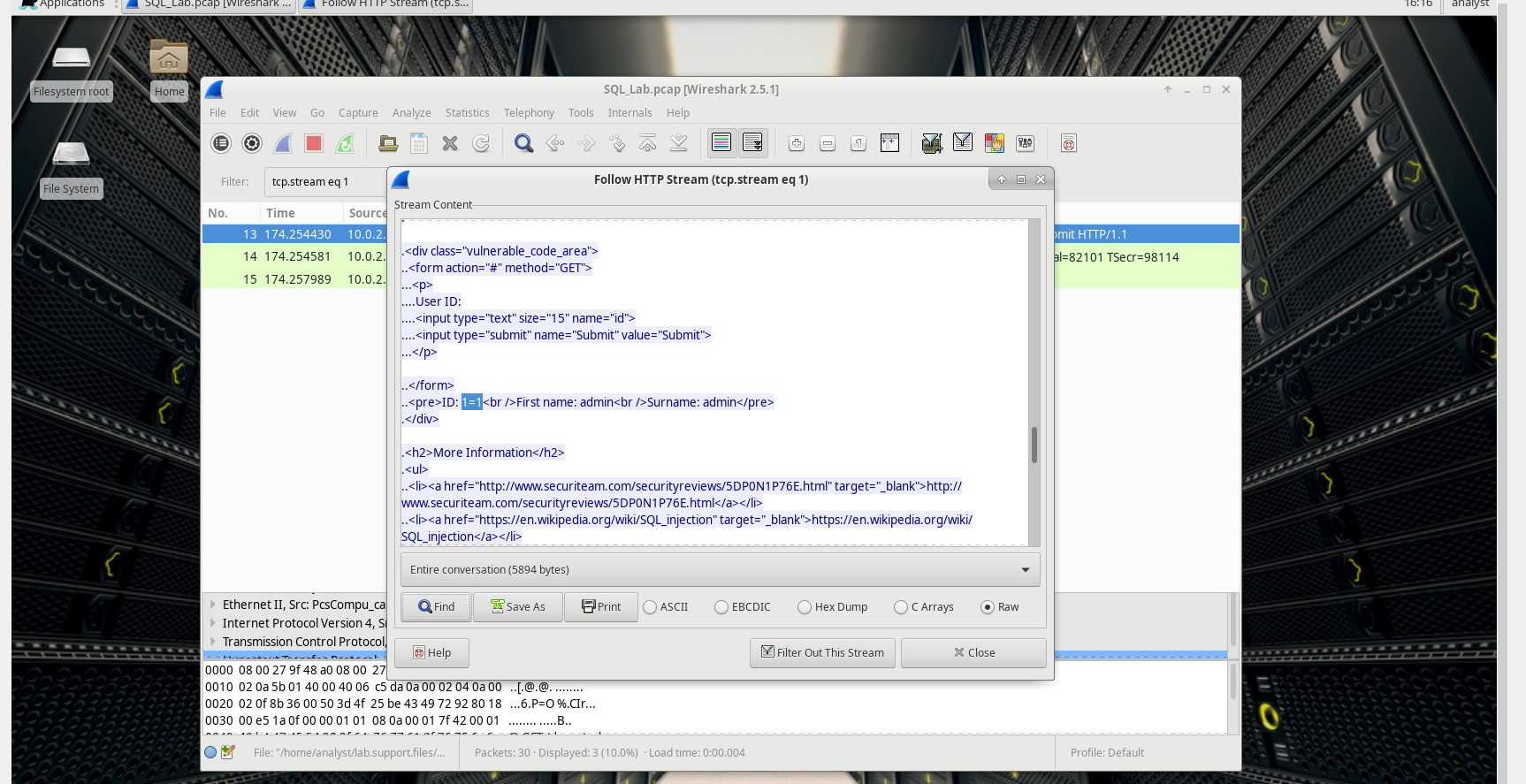
What are the two IP addresses involved in this SQL injection attack based on the information displayed?

10.0.2.15 and 10.0.2.4

**Step 2: View the SQL Injection Attack.**

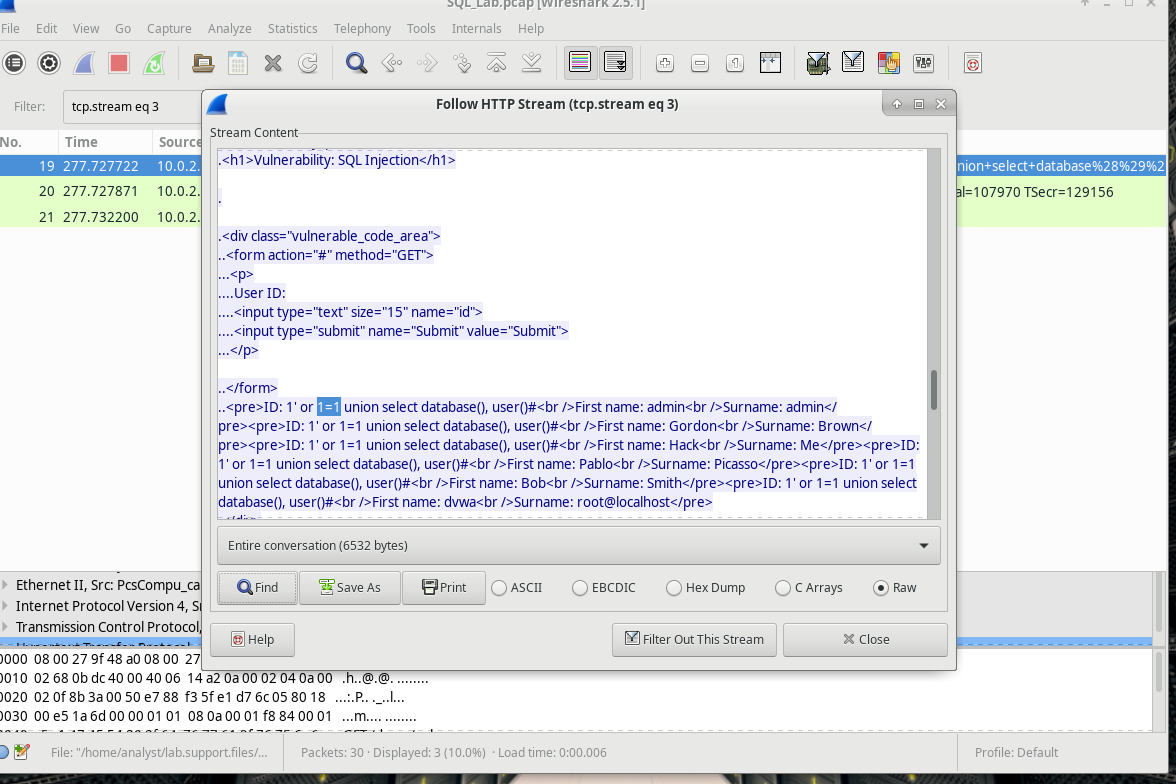


HTTP stream



query (1=1) into a UserID search box

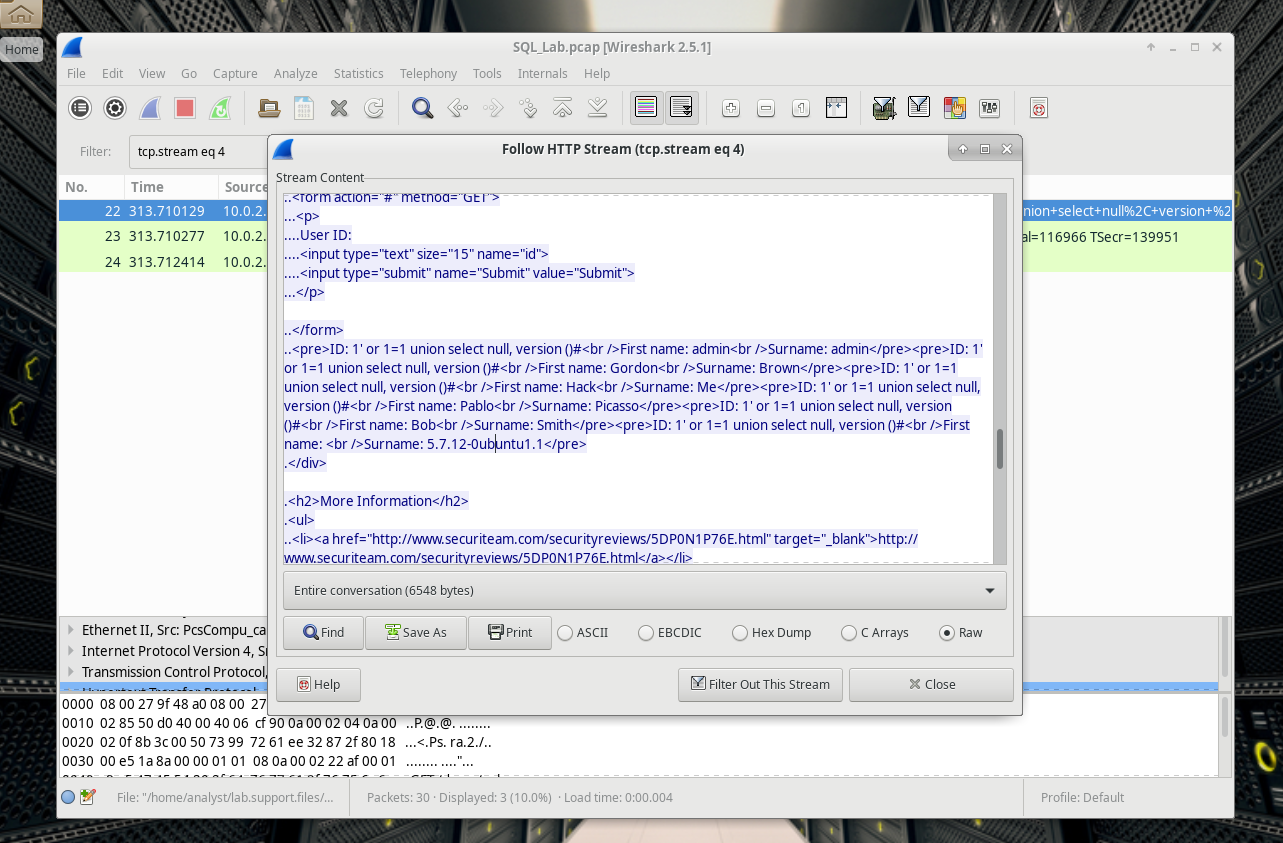
**Step 3: The SQL Injection Attack continues...**



The attacker has entered a query (1’ or 1=1 union select database(), user()#) into a UserID search box on the target 10.0.2.15.

The database name is dvwa and the database user is root@localhost. There are also multiple user accounts being displayed.

Step 4: The SQL Injection Attack provides system information.

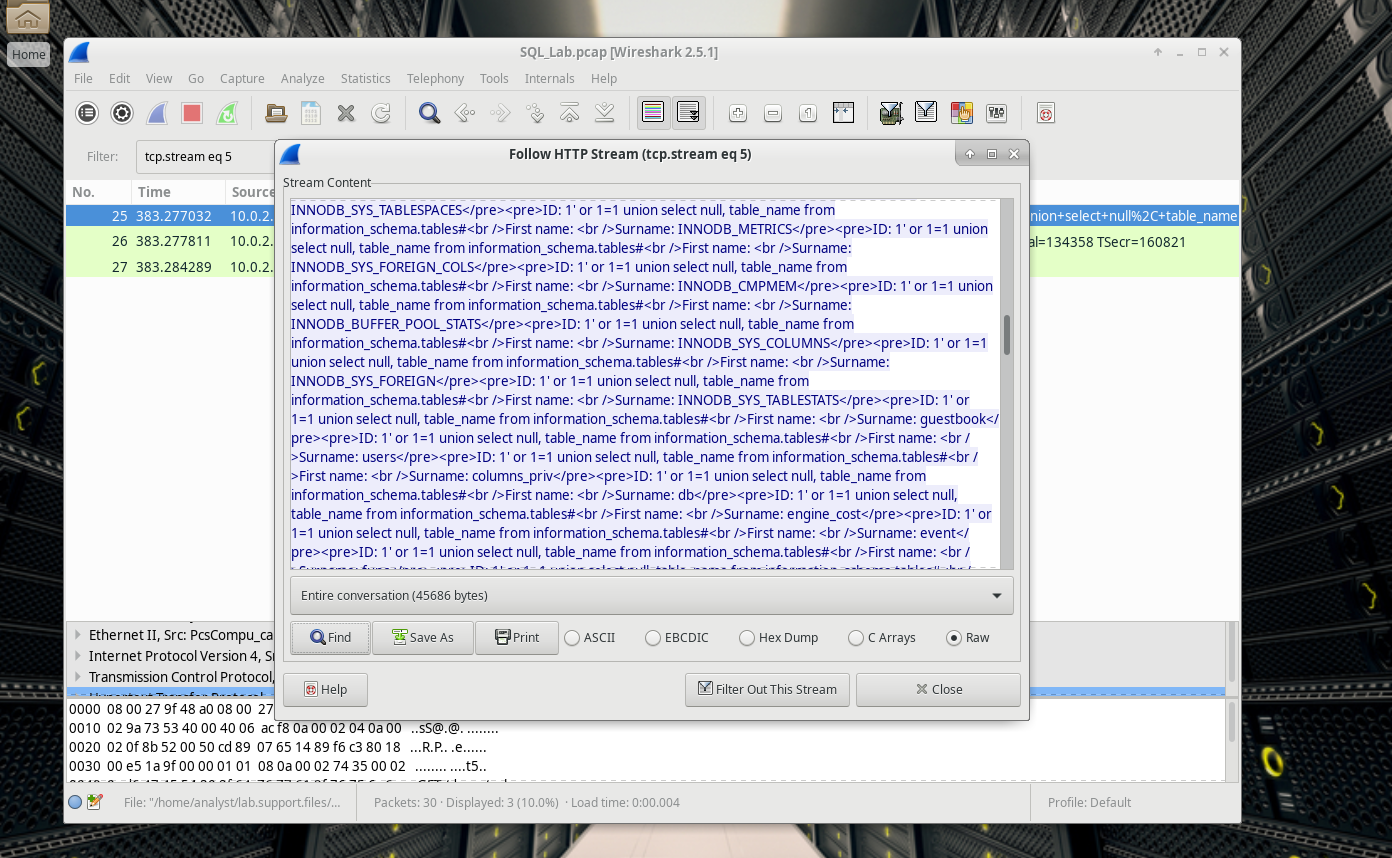


The attacker has entered a query (1’ or 1=1 union select null, version ()#) into a UserID search box

What is the version?

5.7.12-0ubuntu1.1

**Step 5: The SQL Injection Attack and Table Information.**



The attacker has entered a query (1’or 1=1 union select null, table\_name from information\_schema.tables#) into a UserID search box

**What would the modified command of (1' OR 1=1 UNION SELECT null, column\_name FROM INFORMATION\_SCHEMA.columns WHERE table\_name='users') do for the attacker?**

It will display only data where table\_name is equal to users in INFORMATION\_SCHEMA.columns table.

**Step 6: The SQL Injection Attack Concludes.**

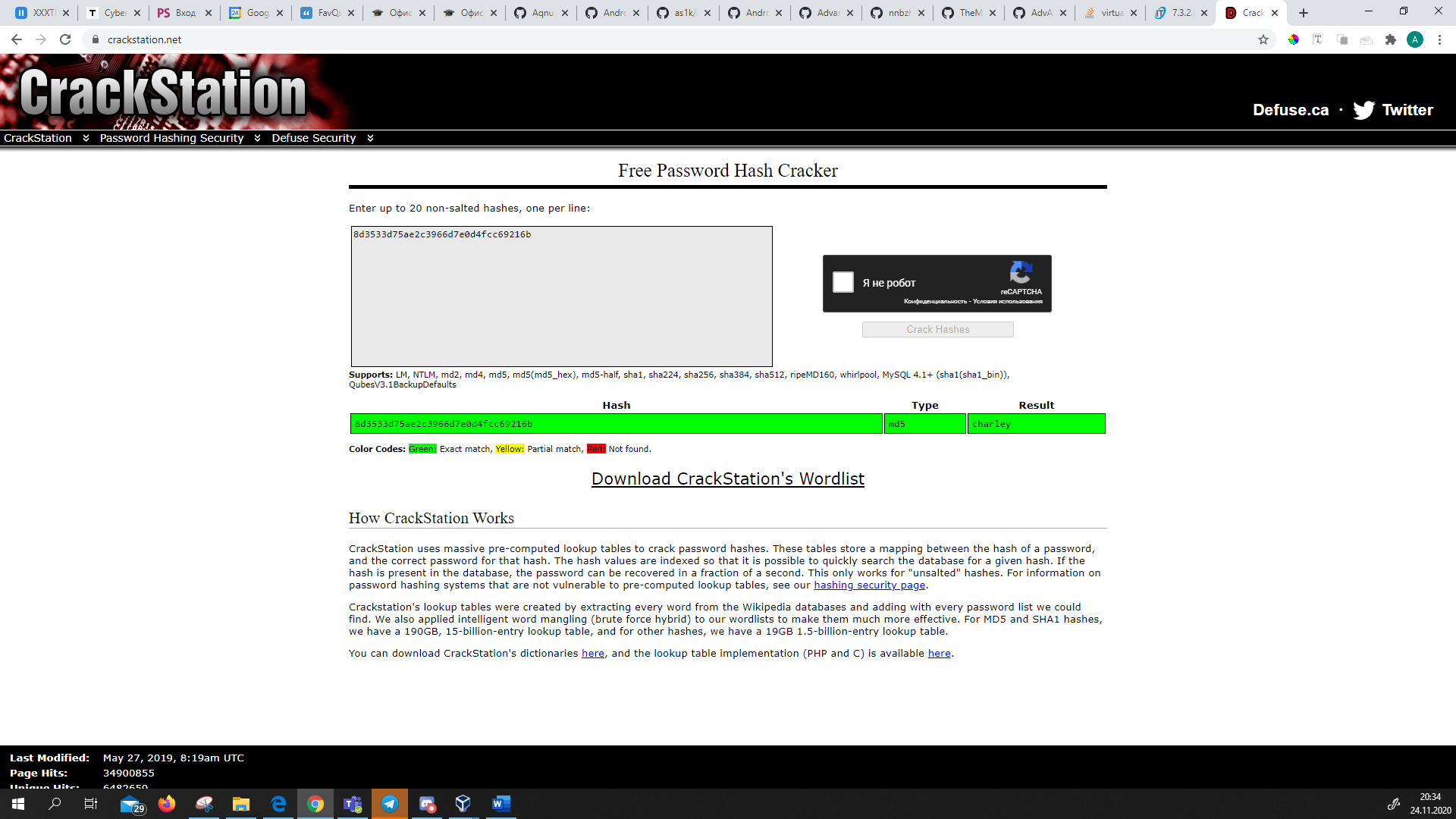


query (1’or 1=1 union select user, password from users#) into a UserID

**Which user has the password hash of 8d3533d75ae2c3966d7e0d4fcc69216b?**

Users: 1337

**Using a website such as https://crackstation.net/, copy the password hash into the password hash cracker and get cracking.**



**What is the plain-text password?**

Charley

Reflection

1. What is the risk of having platforms use the SQL langauge?

An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more.

1. Browse the Internet and perform a search on “prevent SQL injection attacks”. What are 2 methods or steps that can be taken to prevent SQL injection attacks?

The only sure way to prevent SQL Injection attacks is input validation and parametrized queries including prepared statements. The application code should never use the input directly. The developer must sanitize all input, not only web form inputs such as login forms. They must remove potential malicious code elements such as single quotes. It is also a good idea to turn off the visibility of database errors on your production sites. Database errors can be used with SQL Injection to gain information about your database.

If you discover an SQL Injection vulnerability, for example using an Acunetix scan, you may be unable to fix it immediately. For example, the vulnerability may be in open source code. In such cases, you can use a web application firewall to sanitize your input temporarily.