**Lab 5: Attacking a Vulnerable Web Application and Database**

**By**

**Natnael Kebede**



**University of Dallas**

**2020**

This Paper Submitted in Partial Fulfillment of the Requirements for

CYBS 7359 – Penetration Testing and Vulnerability Assessment

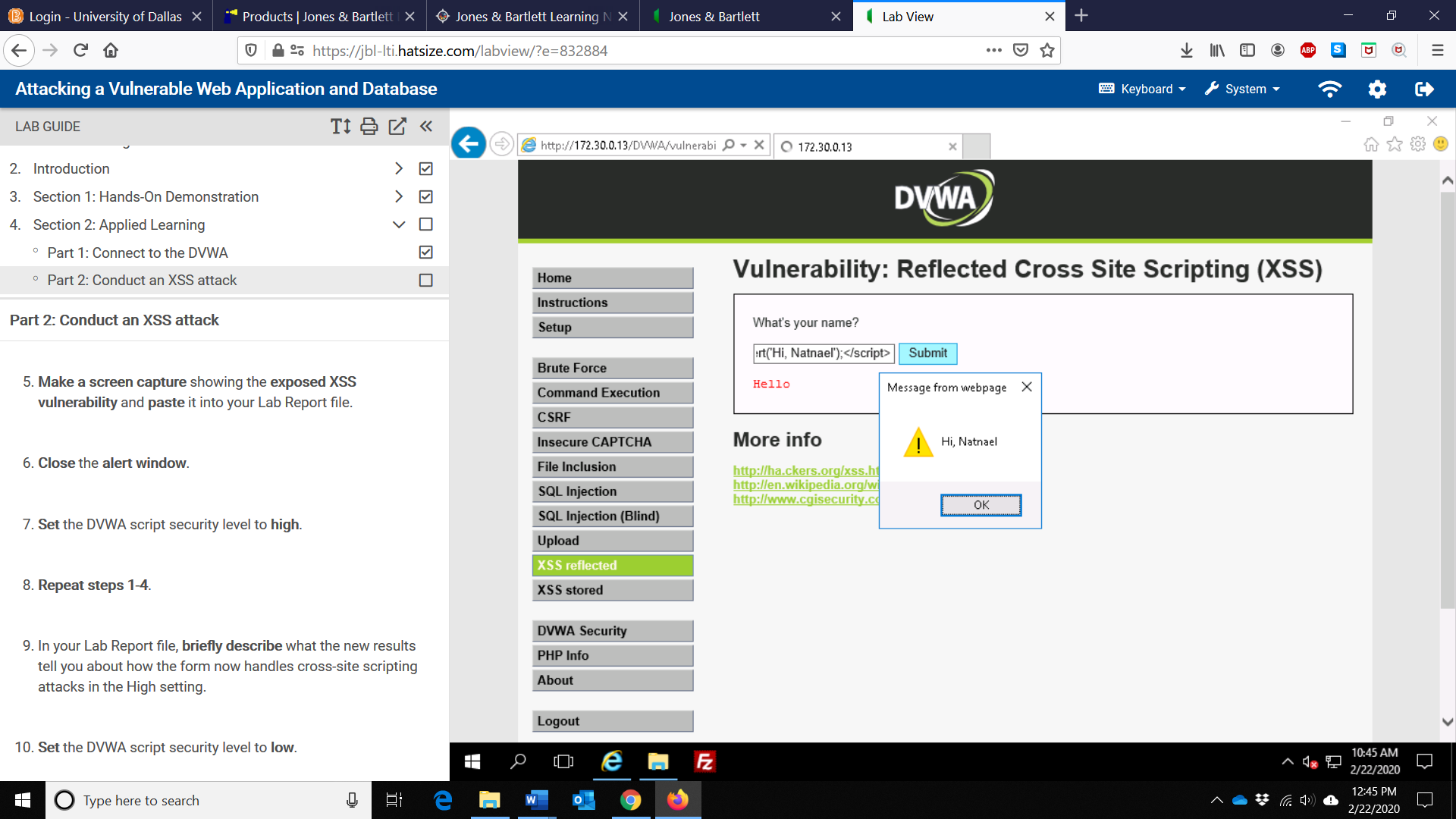
Spring 2020

Presented to Dr. Renita Murimi

**Section 2**

**Part 2**

1. Make a screen capture showing the exposed XSS vulnerability and paste it into your Lab Report file.



1. In your Lab Report file, briefly describe what the new results tell you about how the form now handles cross-site scripting attacks in the High setting.

Setting the DVWA script security level to high displays results that differ from when the level was set to low.

First, entering <Natnael> into the form displayed “Hello Natnael”. That is, adding the scripts to a webpage using the greater and less than arrows surrounding my name demonstrates that the form is not allowing scripts to run and is not vulnerable anymore.

Second, entering <script>alert(‘Hi, Natnael’);</script> into the form displayed Hello <script>alert(‘Hi Natnael’);</script> instead of a command alert pop0up window which would have indicated the a vulnerability still exists and was able to be tested. Additionally, the fact that this result was displayed proves that the form no longer allows scripts to run. Since this simple script wasn’t also processed correctly, there isn’t a good chance that any type of malicious script can be run.

**Part 3**

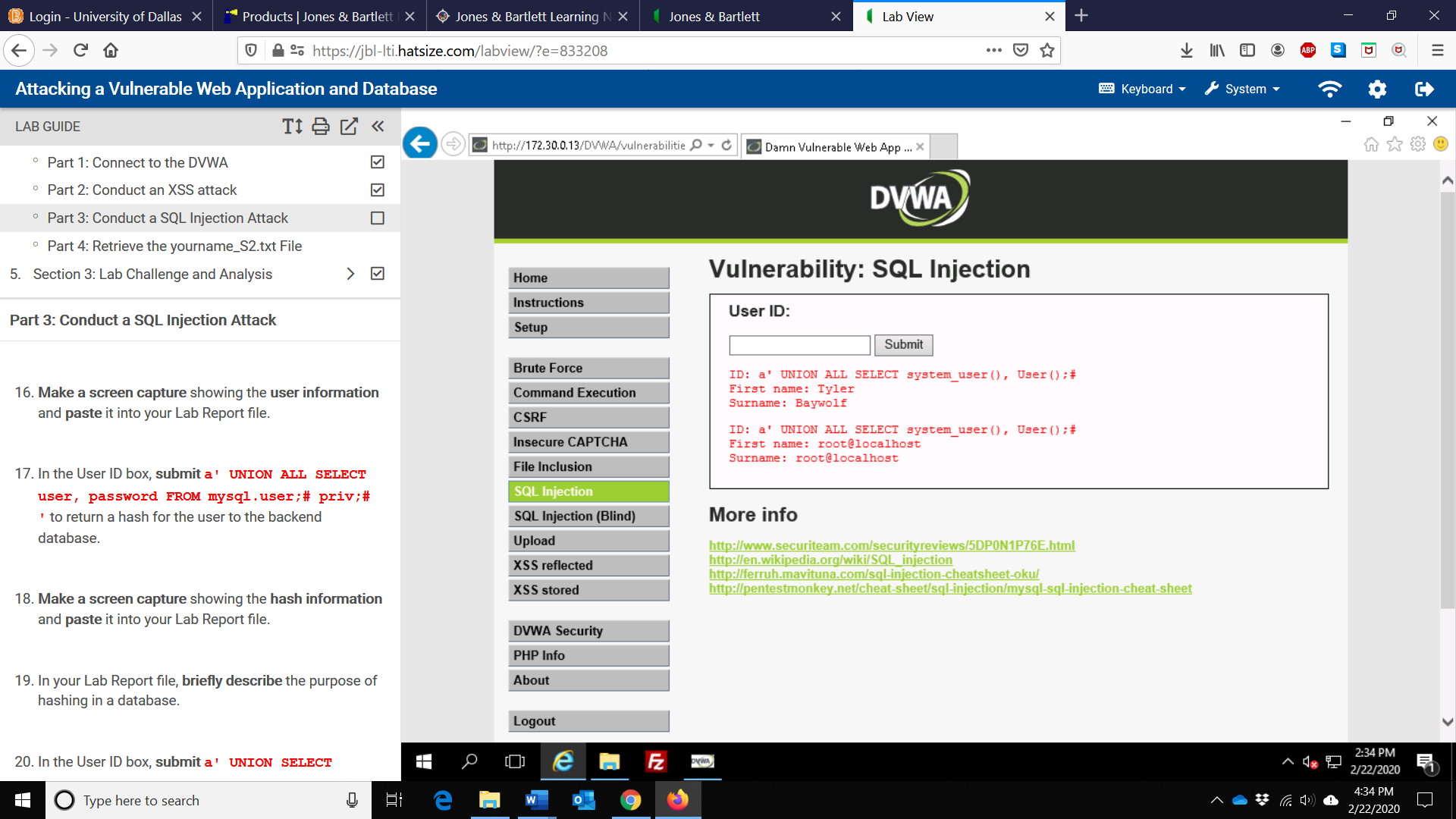
1. In your lab report file, briefly describe the results of your SQL injection attempt (steps 4-6)

The SQL injection a’ ORDER BY 1; # displayed the first and last name of a user in the database (Tyler Baywolf). The lack of errors from the injection shows that there is an existence of vulnerabilities and that we were able to order the output by the first (1) column, or field.

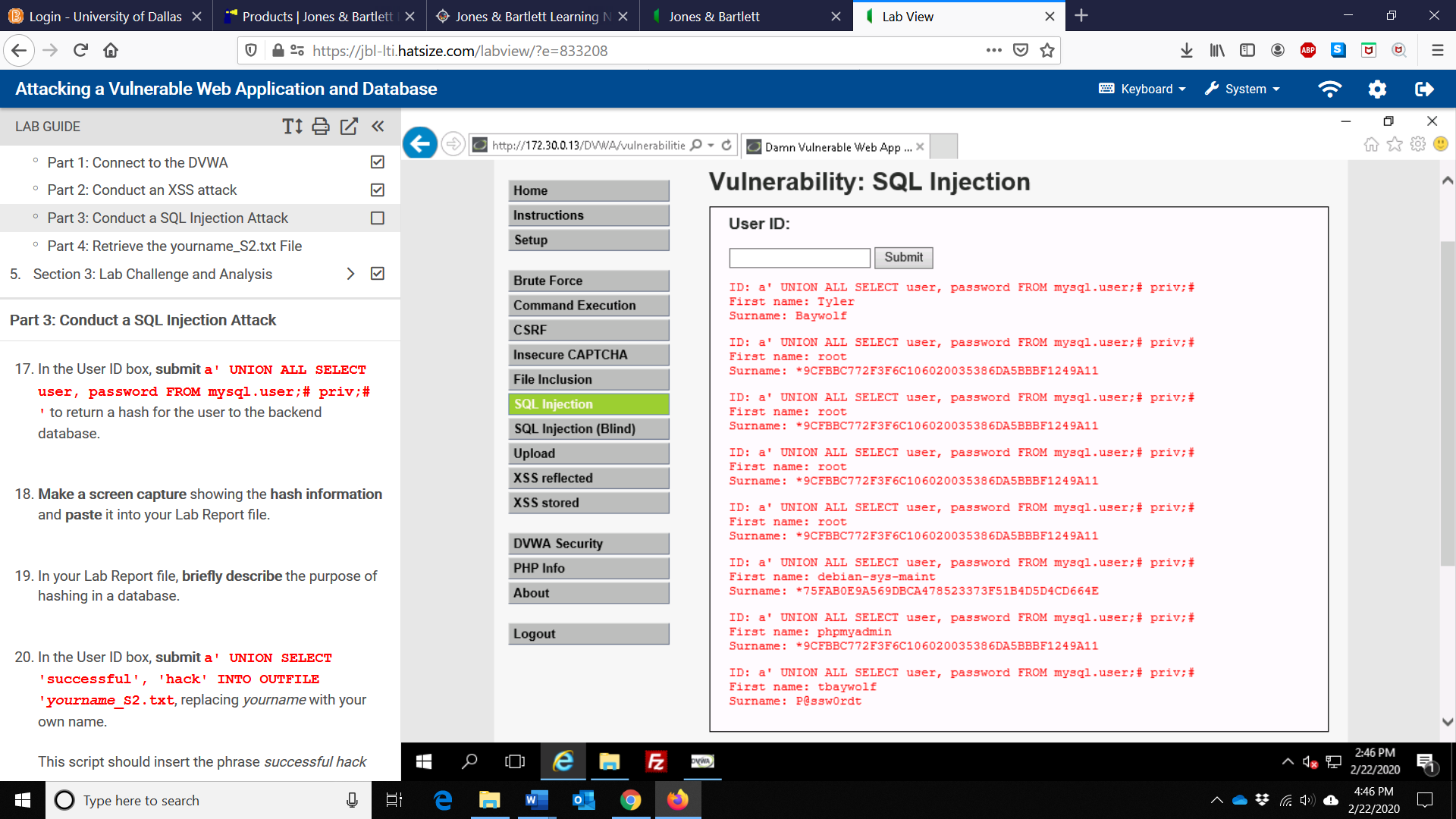
Similarly, the SQL injection a’ ORDER BY 2; # displayed the first and last name of a user in the database (Tyler Baywolf). The lack of errors from the injection shows that there is an existence of vulnerabilities and that we were able to order the output by the second (2) column, or field.

However, the SQL injection a’ ORDER BY 3; # displayed the error: Unknown column ‘3’ in ‘order clause’. The presence of errors shows that there isn’t a third (3) column.

1. Make a screen capture showing the user information and paste it into your Lab Report file.



1. Make a screen capture showing the hash information and paste it into your Lab Report file.

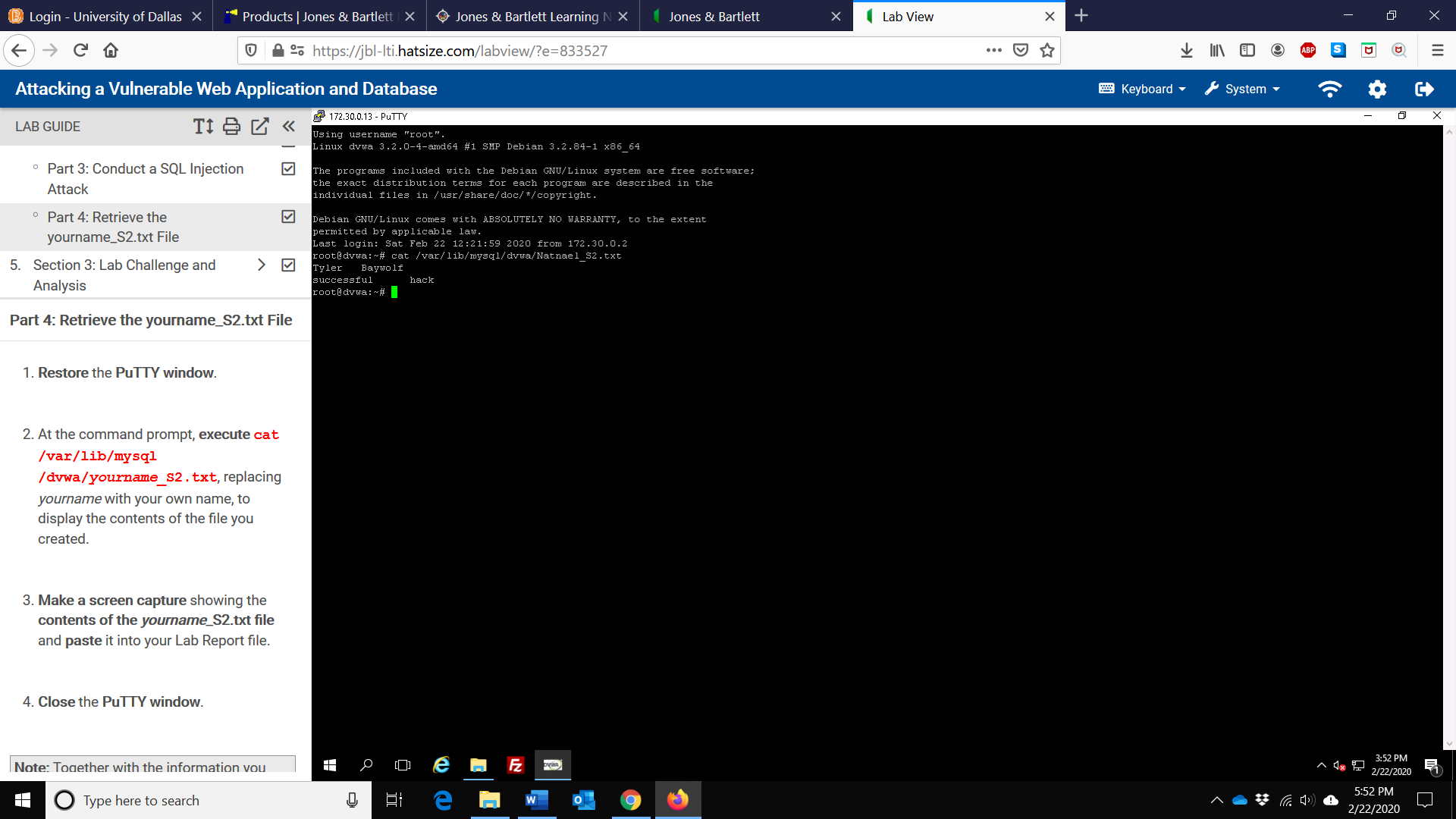


1. In your Lab Report file, briefly describe the purpose of hashing in a database.

The purpose of using hashing in a database is often to index and retrieve data from a database as it is faster to find the data using the shorter fixed-length value or hashed key than to find it using the original value. Additionally, hashing is also used to encrypt and decrypt digital signatures by indexing the original value or key and then using it every time the data associated with the value or key needs to be retrieved. When performing an SQL injection such as a’ UNION ALL SELECT user, password FROM mysql.user;# priv; #’ and getting a result, the hashed value from a database is an output of the data contained in the backend database about users.

**Part 4**

1. Make a screen capture showing the contents of the Natnael\_S2.txt file and paste it into your Lab Report file.



1. In your Lab Report file, briefly describe the security countermeasures you recommend to mitigate the risk from compromise and exploitation.

Some security countermeasure to mitigate the risk from compromise and exploitation include safeguards such as encrypting the data elements that reside in long-term storage of the SQL database. Additionally, organizations need to incorporate penetration testing and web application testing as part of its implementation procedures.

Furthermore, to prevent SQL injection attacks, applications should constrain and sanitize input data before allowing any SQL queries from executing. For instance, users’ entry to online forms should be validated such that SQL tags such as < and >, %, and wildcard signs are not allowed as valid characters or are treated as normal text without capability of being executed as SQL queries.