Nathan is a security consultant for Tyler & Haley financial, a large mortgage lending company in New York City. He has been hired to raise the company DBA’s awareness about SQL injections.

1. Nathan is giving a speech on the four steps of exploitation. What four steps do you anticipate him including within his speech?

When it comes to exploitation, I anticipate Nathan to include the following in his speech.

* Reconnaissance – Sometimes called Footprinting or Fingerprinting, this is when the attacker learns about there victim. This can be passive like learning about the victim through a public domain, or active such as receiving a victim IP address through contact with the system.
* Scanning – Once reconnaissance is done, it is time to scan the victim’s network for vulnerabilities. Tools like netcat can prob the network for open ports and other network information.
* Enumeration – Once enough information is gathered, it is time to plan an attack. The assailant will then devise an attack using the information gathered and chose a specific tool stack. Examples of this are creating attacks for SQL-injection, Cross Site Scripting, or Social Engineering Attacks using a popular tool called Metasploit distributed on a Kali Linux OS.
* Penetration -This is the actual attack on the system, this is where all the previous steps come together, and the previously devised attack plan is now executed.

2. Nathan is planning to describe at least three ways a database management system can be identified. What three ways for identifying a DBMS should Nathan cover?

When it comes to three ways for identifying a DBMS, I anticipate Nathan will cover the following.

* Query the Database Version: Databases have prebuilt commands to display their versioning. If one of the queries are executed, it is the quickest path to reveal DBMS versioning.
* Using a Boolean String Query: Each database handles strings and how they concatenate them differently. By adding a specific Boolean query with the strings concatenated, it will reveal true or false. Depending on what Boolean query is performed, it will return true if that database could handle that query in the specific dialect. If it does not comprehend that query you know you are using the wrong language for that database.
* Reviewing Error messages: One can gather versioning information from error messages. An example of this is performing HTTP request to an endpoint could return information about the database in response. Another example is calling a query directly to the database and bypassing the application, this can lead to information gathered in an error message or response.

A fourth way that’s worth mentioning,

* The Data types : Every database does not use the same type of data types. If information about the data types were recovered one could narrow down a conclusion about the database. For example, MySQL does not have a datatype of “Array”, but the NoSQL database MongoDB does.

1. Nathan plans to provide a few examples of SQL statements that can be used to gather information externally. Provide two examples of SQL statements that Nathan can provide.

Two examples of SQL statements that Nathan could potential provide.

Example 1 – Versioning calls.

The known version call for MySQL is **SELECT version()**

An example of this would look like this for MySQL.

[**http://www.bank.com/myLogin.php?id=1%20UNION%20SELECT%20version()**](http://www.bank.com/myLogin.php?id=1%20UNION%20SELECT%20version())

The database would see this request as this query and return the versioning.

**SELECT \* FROM Users WHERE userId = 1 UNION SELECT version();**

Example 2 – Boolean String Concatenation Calls.

The known String concatenation call for MySQL and ORACLE is **CONCAT('a','b')**

An example of this would look like this for MySQL and ORACLE.

[**http://www.bank.com/myLogin.php?id=1%20AND%20%27ab%27=CONCAT%28%27a%27%2C%27b%27%29**](http://www.bank.com/myLogin.php?id=1%20AND%20%27ab%27=CONCAT%28%27a%27%2C%27b%27%29)

The database would see this request as this query and return true if it is MySQL or Oracle, and false for all others.

**SELECT \* FROM Users WHERE userId = 1 AND 'ab' = CONCAT('a','b');**

1. What suggestions do you expect Nathan to provide for securing the company databases against SQL injections?

When it comes to suggested ways for protecting against SQL injections, I anticipate Nathan will cover the following.

-Less User Supplied Input: The less options users must input, the less points of access the attacker has to inject code.

-Good User Input Validation: If user input is needed, there should be sufficient validation both on the front end before the request is sent and on the backend after the request is handled to validate the user typed the correct content that is needed to perform this query. It should verify that characters and patterns that are detrimental not get used.

-Using Prepared statements / Stored Procedures : Using a statement can leave the system vulnerable to SQL injection. Using a prepared statement to prepare the statement on the application side can prevent this or by using a callable statement to call the database who already houses the statement within itself. That way there is no exposed statement and the statement is already prepared on the application side or housed within the database already.

-Enforcing Least Privilege: By enforcing least privilege to users, assailants can not use their hacked user accounts to make queries out of that user’s scope. For example, if an attacker were to compromise an employees’ account but executed a query on that account that only a manager can execute. Then if the employee account is enforced by least privilege the query will not execute because they do not have that privilege and the database remains secure.