“SQL Injection and Prevention”

A Seminar Report
Submitted in partial fulfillment of the
Requirements for the award of the Degree of

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY)

Submitted by
Kedar Sitaram Jadhav (Roll No. 30)

DEPARTMENT OF INFORMATION TECHNOLOGY
RAMNIRANJAN JHUNJHUNWALA COLLEGE
(Affiliated to University of Mumbai)
GHATKOPAR (WEST), 400086
MAHARASHTRA
2021-2022

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the seminar entitled, "**SQL Injection and Prevention**", is bonafied work of name Kedar Jadhav bearing Seat. No: 30 submitted in partial fulfillment of the requirements for the award of degree of MASTERS OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide

Coordinator

External Examiner

Date: 07-04-2022

College Seal

ACKNOWLEDGEMENT

I have great pleasure in presenting this seminar entitled "**SQL Injection and Prevention**" and I grab this opportunity to convey my immense regards to all people who with their invaluable contributions made this seminar successful.

It gives me great pleasure in presenting this Seminar Report. Its justification will never complete if I don't express my vote of thanks **RAMNIRANJAN JHUNJHUNWALA COLLEGE** AND **Dr. HIMANSHU DAWDA**.

I sincerely thank and express my profound gratitude to our Seminar Guide **Mrs. BHARTI BHOLE** for timely and prestigious guidance required for the completion.

INDEX

1. Introduction

In this Seminar I will show you how SQL injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input.

SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

2. SQL Injection

2.1 What is SQL?

SQL (Structured Query Language) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables.

SQL is used to communicate with a database. it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database.

There are five types of SQL commands:

1. DDL
2. DML
3. DCL
4. TCL
5. DQL

Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Microsoft Access, Ingres, etc.

How Does it work?

SQL is the most common language for extracting and organising data that is stored in a relational database. A database is a table that consists of rows and columns.

SQL manages a large amount of data, especially if there is a lot of data that is being written simultaneously and there are too many data transactions.

There are different versions and frameworks for SQL, the most commonly used is MySQL. MySQL is an open-source solution that helps facilitate SQL's role in managing back-end data for web applications.

When an SQL query is written & run (or parsed), it is processed by a query optimizer. The query reaches SQL server, where it compiles in three phases; Parsing, Binding and Optimization.

1. Parsing – A process to check the syntax
2. Binding – A process to check the query semantics
3. Optimization – A process to generate the query execution plan

In the third step, all possible permutations and combinations are generated to find the most effective query execution plan in a reasonable time. The shorter the query takes, the better it is.

What is SQL Used for?

- Execute queries against a database
- Retrieve data from a database
- Insert, update and delete records into a database
- Create new databases, or new tables in a database
- Create stored procedures & views in a database
- Set permissions on tables, procedures, and views

2.2 What is SQL Injection?

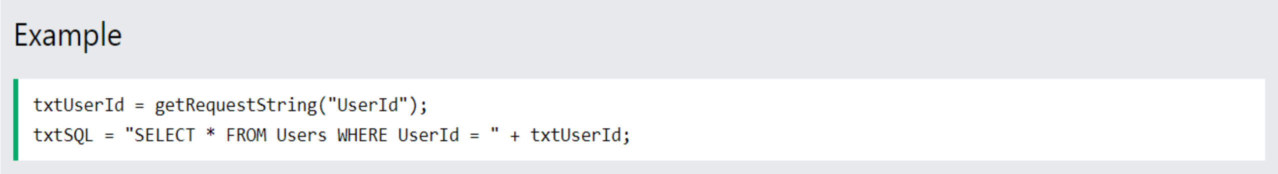
SQL injection is a code injection technique that might destroy your database.

SQL injection is one of the most common web hacking techniques.

SQL injection is the placement of malicious code in SQL statements, via web page input.

SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will unknowingly run on your database.

Look at the following example which creates a SELECT statement by adding a variable (txtUserId) to a select string. The variable is fetched from user input (getRequestString):



Example

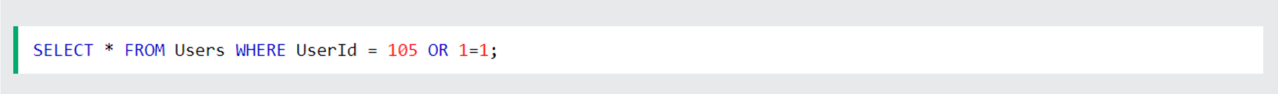
```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

SQL Injection Based on 1=1 is Always True

If there is nothing to prevent a user from entering "wrong" input, the user can enter some "smart" input like this:

UserId: 105 OR 1+1

Then, the SQL statement will look like this:

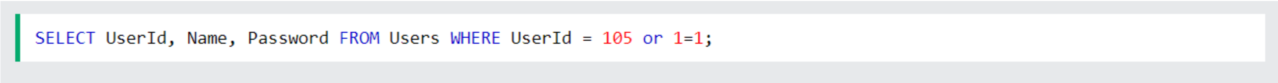


```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

The SQL above is valid and will return ALL rows from the "Users" table, since OR 1=1 is always TRUE.

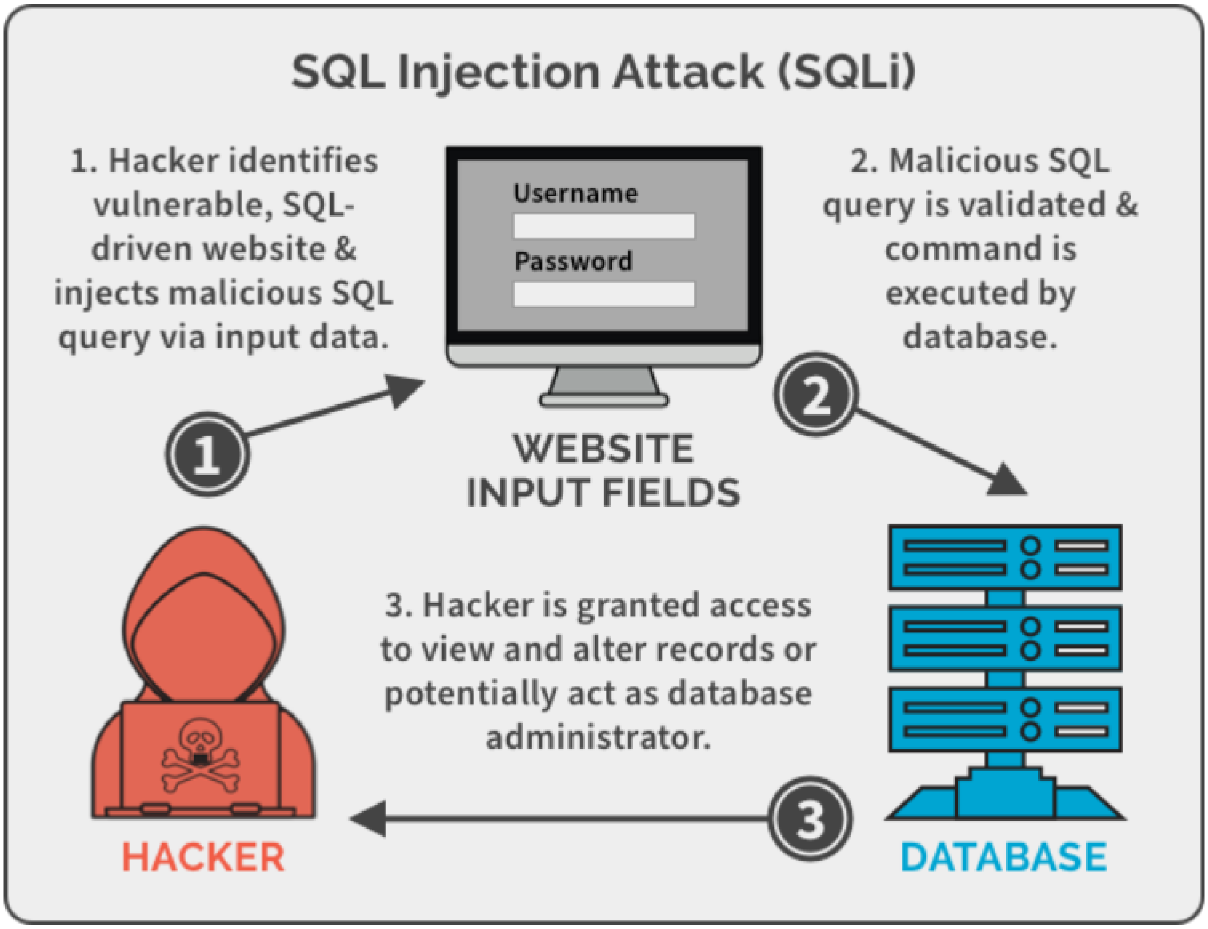
Does the example above look dangerous? What if the "Users" table contains names and passwords?

The SQL statement above is much the same as this:



```
SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1;
```

A hacker might get access to all the user names and passwords in a database, by simply inserting 105 OR 1=1 into the input field.


SQL Injection Attack (SQLi)
1. Hacker identifies vulnerable, SQL-driven website & injects malicious SQL query via input data.
Username
Password
2. Malicious SQL query is validated & command is executed by database.
1
2
WEBSITE INPUT FIELDS
3. Hacker is granted access to view and alter records or potentially act as database administrator.
3
HACKER
DATABASE

2.3 Application Security Weakness

To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application.

A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.

- Attackers can use SQL Injections to find the credentials of other users in the database.

- SQL lets you select and output data from the database. An SQL Injection vulnerability could allow the attacker to gain complete access to all data in a database server.
- You can use SQL to delete records from a database, even drop tables. Even if the administrator makes database backups, deletion of data could affect application availability until the database is restored.
- In some database servers, you can access the operating system using the database server. This may be intentional or accidental.

2.4 Types of SQL Injection

SQLi is a common and well-documented attack strategy whose success has far-reaching business consequences such as unauthorized viewing of credentials and gaining administrative access to the application's database.

SQLi attacks are categorized based on the following methods used to gain database access:

1. **In-band SQLi**- The attacker gathers their results using the same communication channel they use to launch attacks.
2. **Inferential SQLi**- In a Blind SQL injection technique, the hacker sends malicious data payloads, then reconstructs the database server's structure using the web application's response.
3. **Out-of-Band SQLi**- The attacker uses the same channel to launch the attack and gather results. While this attack is uncommon since it relies on certain database server features being enabled.

- Injection involves -
    - Injection through user input
    - Injection through cookies
    - Injection through server variables

- Inferential SQLi injection involves – Content based and Time-based SQLi. The attacker submits a second (different) request. To handle the second request, causing the attackers injected SQL query to execute.

2.5 Attack Intent

1. **Adding or Modifying data**– Add new data to the database, Perform an INSERT in the injected SQL

2. **Bypassing authentication**– Alter passwords or databases permissions

3. **Extracting data**– Existing database or server objects, such as tables or user's details

4. **Determining database schema**– Modify stored procedures, functions, or other database schemas

2.6 Other Injection type

1. **Code Injection** – Code injection is one of the most common types of injection attacks. If attackers know the programming language, the framework, the database or the operating system used by a web application, they can inject code via text input fields to force the webserver to do what they want.

2. **Command Injection** – These attacks are also possible, mainly due to insufficient input validation. They differ from code injection attacks in that the attacker inserts system commands instead of pieces of programming code or scripts.

   Therefore, hacker doesn't need to know the programming language in which the application is based or the language used by the database. But they need to know the operating system used by the hosting server.

3. **Cross side scripting** – Whenever an application inserts input from a user within the output it generates, without validating or encoding it, it gives the opportunity to an attacker to send malicious code to a different end-user.

   Cross-Site Scripting (XSS) attacks take these opportunities to inject malicious scripts into trusted websites, which is ultimately sent to other users of the application, which become the attacker's victims.

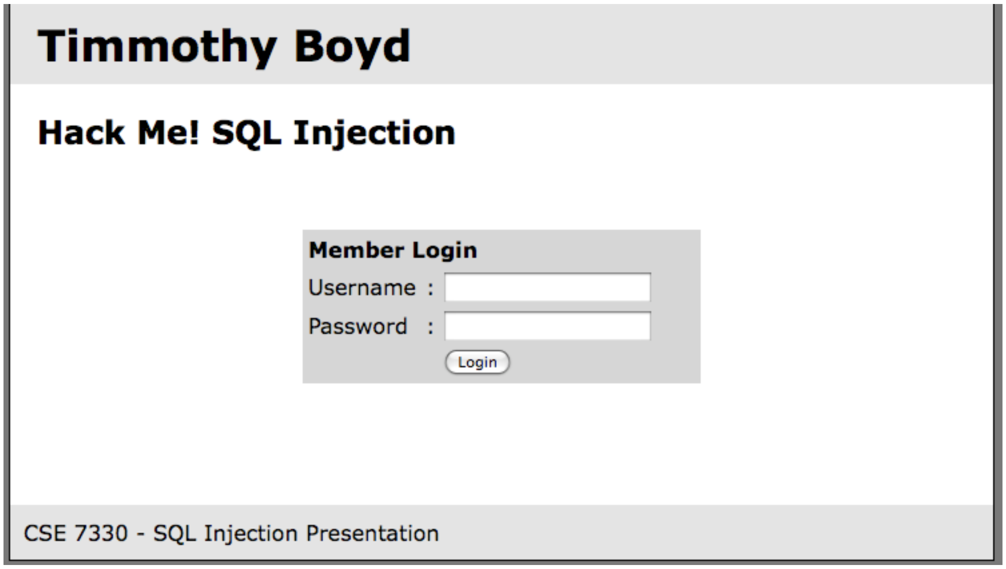
4. **xPath Injection**– This type of attack is possible when a web application uses information provided by a user to build an XPath query for XML data.

   The way these attack works is similar to SQL injection: attackers send malformed information to the application in order to find out how the XML data is structured, and then they attack again to access that data.

5. **LDAP Injection**– LDAP is a protocol designed to facilitate the search of resources (devices, files, other users) in a network. It is very useful for intranets, and when used as part of a single sign-on system, it can be used to store usernames and passwords.

   LDAP queries involve the use of special control characters that affect its control. Attackers can potentially change the intended behavior of an LDAP query if they can insert control characters in it.

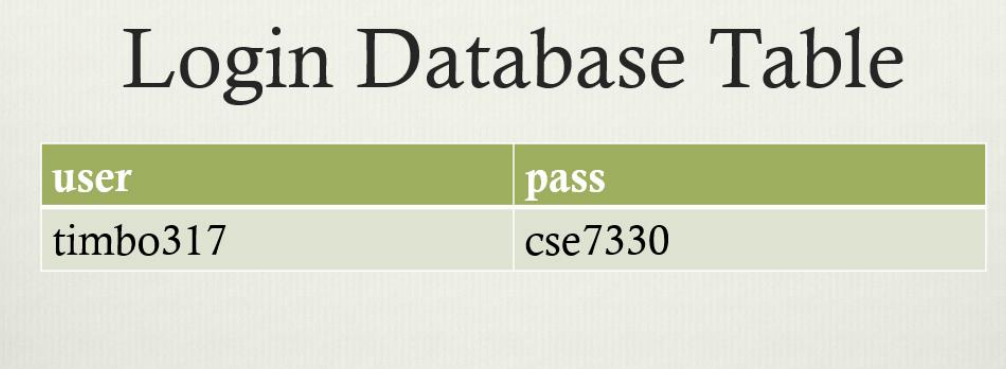
3. Example Website



Timmothy Boyd

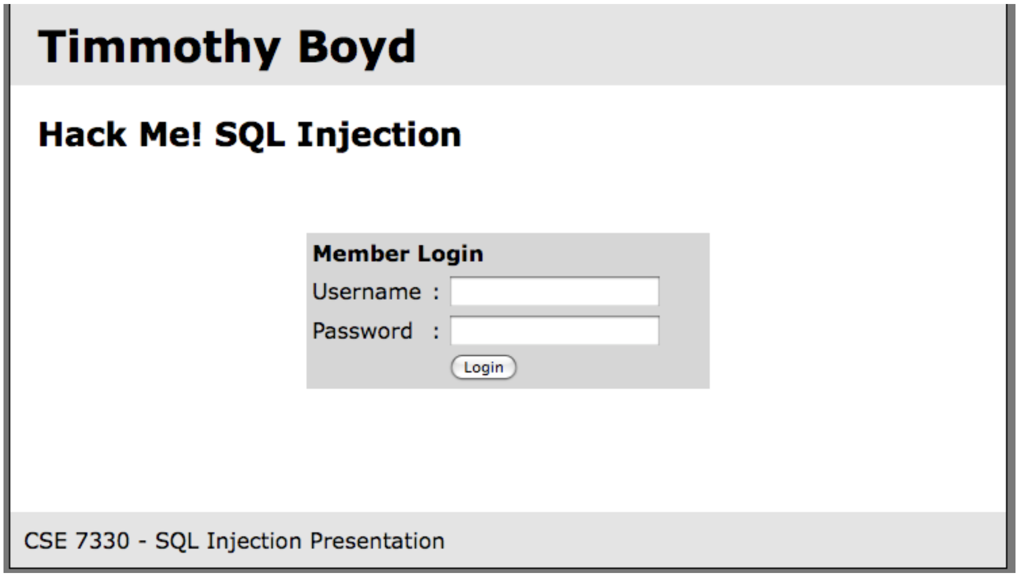
Hack Me! SQL Injection

CSE 7330 - SQL Injection Presentation



Login Database Table

user	pass
timbo317	cse7330



Timmothy Boyd

Hack Me! SQL Injection

CSE 7330 - SQL Injection Presentation

SELECT * FROM `login` WHERE `user`=‘’; DROP TABLE `login`; --’ AND `pass`=‘’

3.1 More Malicious Example

1. SELECT * FROM `login` WHERE `user`=‘’; INSERT INTO `login` ('user','pass') VALUES ('haxor','whatever');--’ AND `pass`=‘’

2. SELECT * FROM `login` WHERE `user`=‘’; UPDATE `login` SET `pass`=‘pass123’ WHERE `user`=‘timbo317’;--’ AND `pass`=‘’

3. SELECT * FROM `login` WHERE `user`=‘’; DROP TABLE `login`; --’ AND `pass`=‘’

4. Real World Example

1. **GhostShell attack**—hackers from APT group Team GhostShell targeted 53 universities using SQL injection, stole and published 36,000 personal records belonging to students, faculty, and staff.

2. **Turkish government**—another APT group, RedHack collective, used SQL injection to breach the Turkish government website and erase debt to government agencies.

3. **7-Eleven breach**—a team of attackers used SQL injection to penetrate corporate systems at several companies, primarily the 7-Eleven retail chain, stealing 130 million credit card numbers.

4. **HBGary breach**—hackers related to the Anonymous activist group used SQL Injection to take down the IT security company’s website. The attack was a response to HBGary CEO publicizing that he had names of Anonymous organization members.

5. An SQL injection vulnerability resulted in an urgent June bugfix release of Ruby on Rails 3.x.

6. Yahoo! Voices was hacked in July. The attack acquired 453,000 user email addresses and passwords. The perpetrators claimed to have used union-based SQL injection to break in. LinkedIn.com leaked 6.5 million user credentials in June. A class action lawsuit alleges that the attack was accomplished with SQL injection.

7. SQL injection was documented as a security threat in 1998, but new incidents still occur every month. Making honest mistakes, developers fail to defend against this means of attack, and the security of online data is at risk for all of us because of it.

5. How does this prevent an attack?

1. The root cause of SQL injection vulnerabilities is insufficient input validation.
2. The SQL statement you pass to prepare is parsed and compiled by the database server.
3. By specifying parameters (either a ? or a named parameter like :name) you tell the database engine what to filter on.
4. Don't store password in plain text in the DB
   - salt them and hash them
5. Check syntax of input for validity
   - Many classes of input have fixed languages
   - Verify that the input is a valid string in the language
   - Exclude quotes and semicolons
6. By specifying Have length limits on input
   - Many SQL injection attacks depend on entering long strings
   - Code reviews

Least Privilege

- To minimize the potential damage of a successful SQL injection attack, you should minimize the privilege assigned to every database account in your environment.
- Do not assign DBA or admin type access rights to your application accounts.
- Don’t run your DBMS as root or system.
- Always enforce the constraint – Server side.
- READ ONLY database access.

Quick Fixes

- For companies that have a large setup or a lot of legacy code that will take a long time to audit and fix, put some SQL injection detection pattern in your Load Balancer itself.
- Enable mod_security on Apache.
- Run the RIPS scanner on your code for detecting vulnerabilities.
    - https://sourceforge.net/projects/rips-scanner/
    - https://resources.infosecinstitute.com/topic/rips-finding-vulnerabilities-php-application/

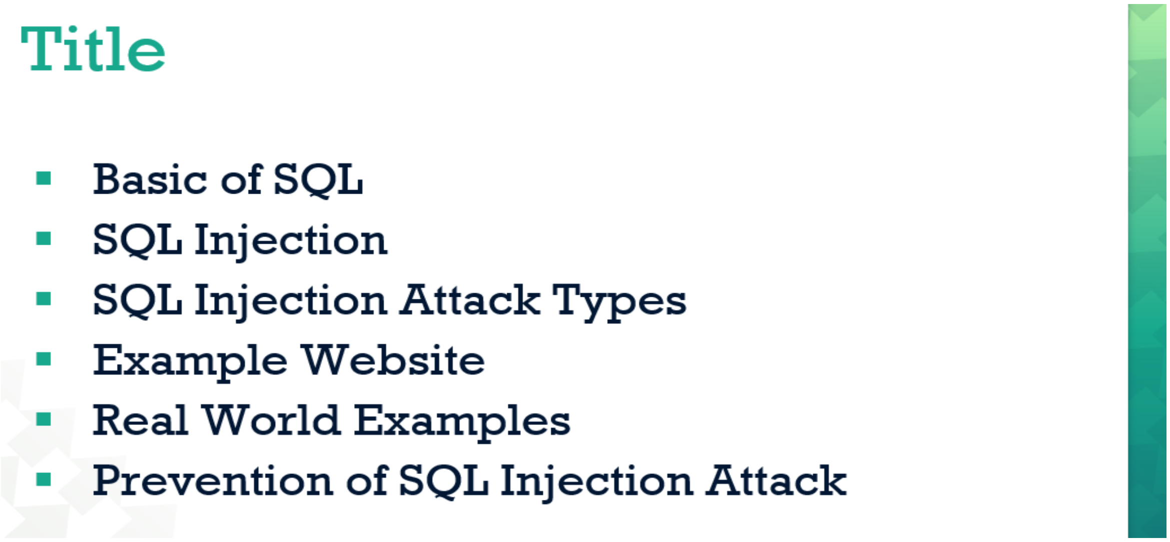
5. Presentation information

5.1 Presentation Snap Shots

1. Start Page

SQL Injection and Prevention.

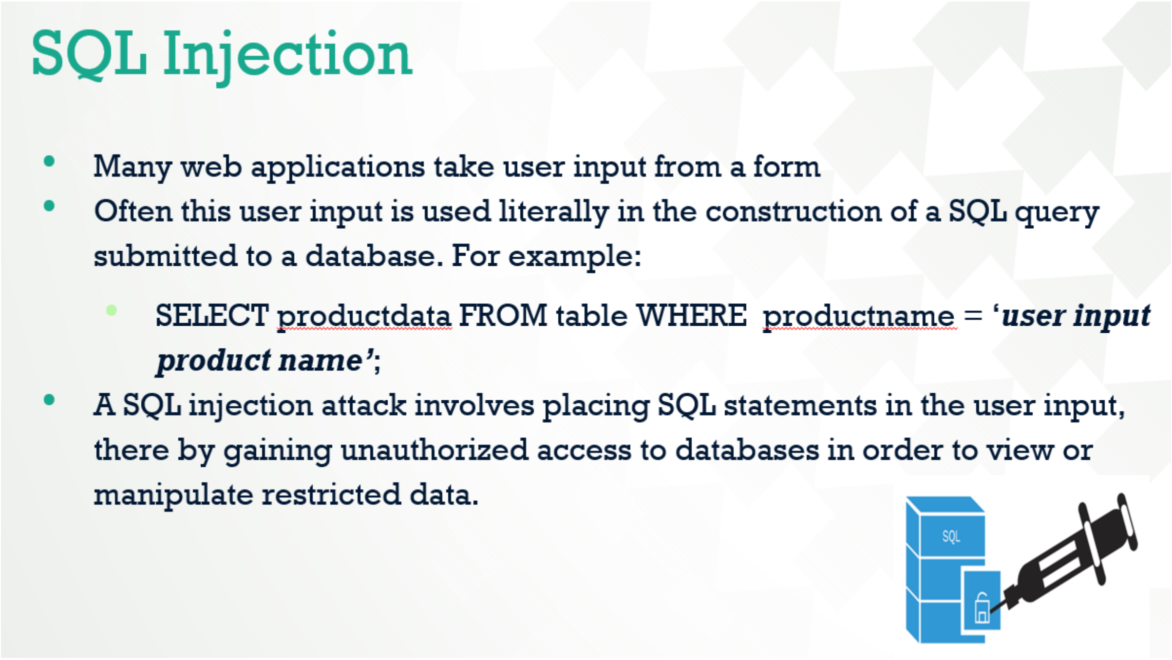
2. Topic cover



Title

- Basic of SQL
- SQL Injection
- SQL Injection Attack Types
- Example Website
- Real World Examples
- Prevention of SQL Injection Attack

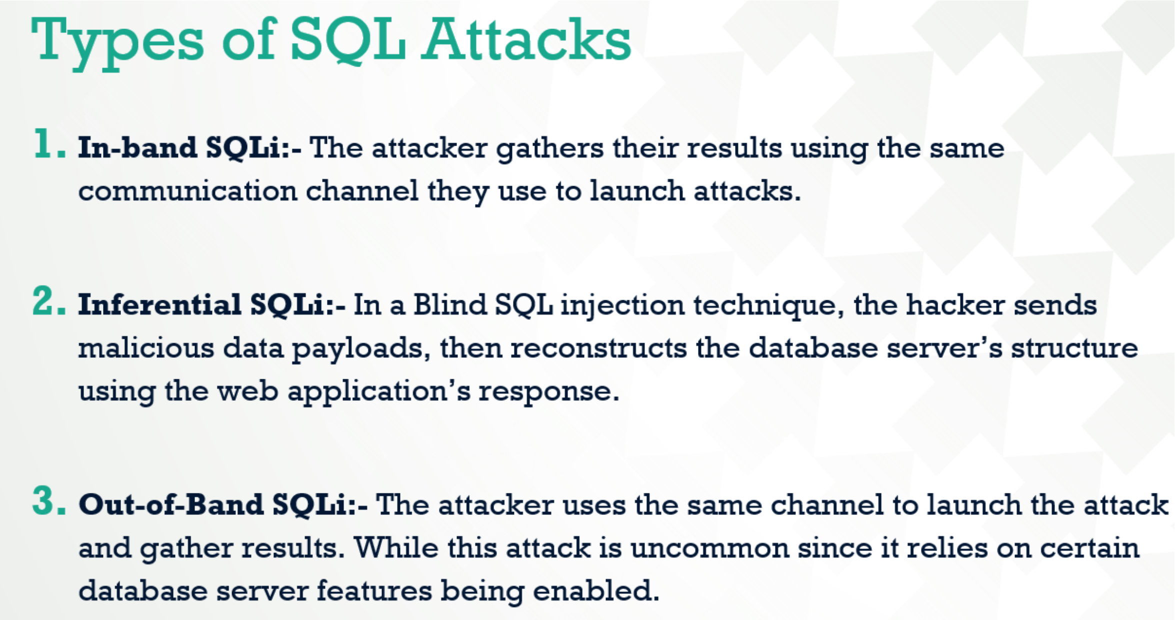
3. SQL Injection



SQL Injection

- Many web applications take user input from a form
- Often this user input is used literally in the construction of a SQL query submitted to a database. For example:
  - SELECT productdata FROM table WHERE productname = '***user input product name***';
- A SQL injection attack involves placing SQL statements in the user input, there by gaining unauthorized access to databases in order to view or manipulate restricted data.

4. Types of attacks



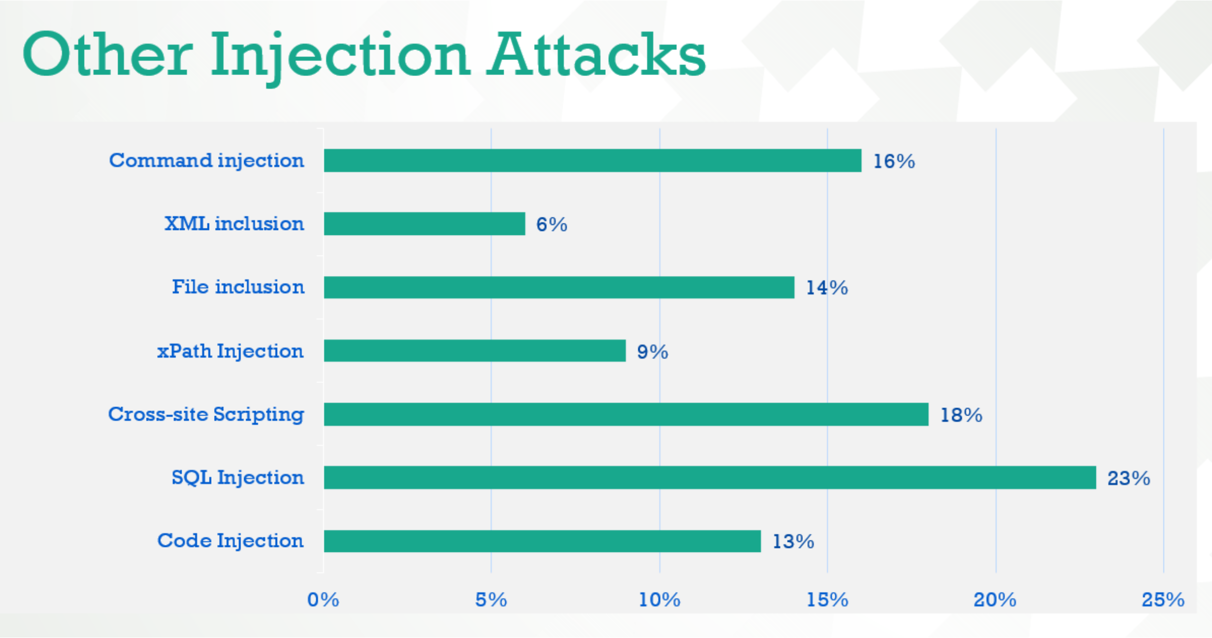
Types of SQL Attacks

1. **In-band SQLi:-** The attacker gathers their results using the same communication channel they use to launch attacks.

2. **Inferential SQLi:-** In a Blind SQL injection technique, the hacker sends malicious data payloads, then reconstructs the database server's structure using the web application's response.

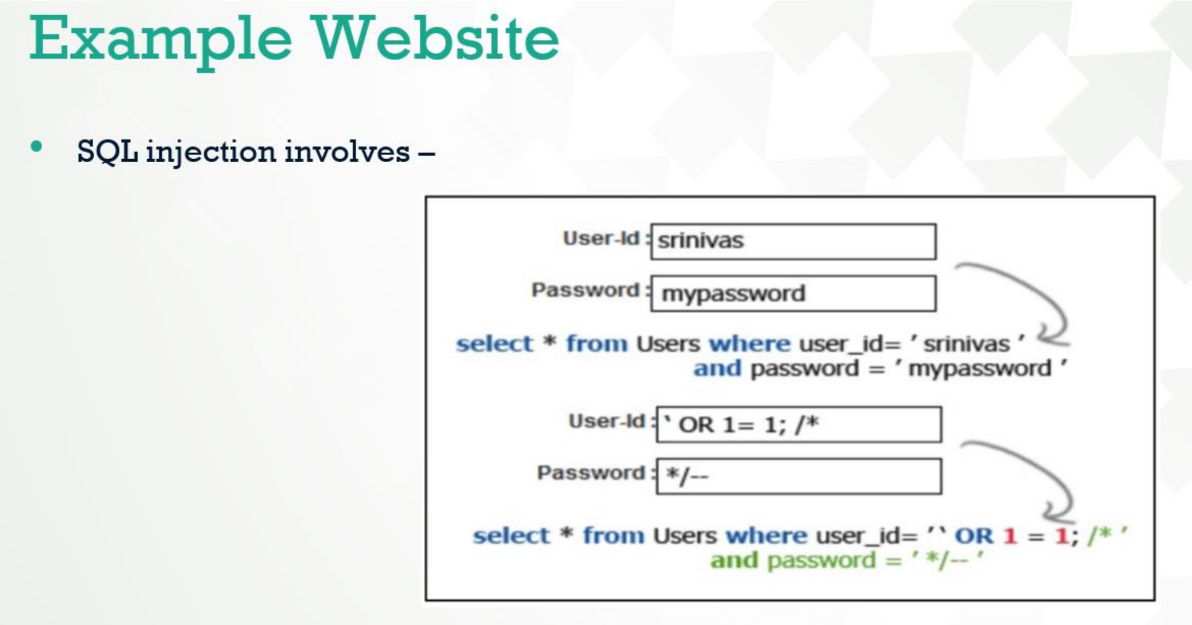
3. **Out-of-Band SQLi:-** The attacker uses the same channel to launch the attack and gather results. While this attack is uncommon since it relies on certain database server features being enabled.

5. Other Injection attack



Other Injection Attacks

6. Example website



Example Website

- SQL injection involves –

User-Id : srinivas

Password : mypassword

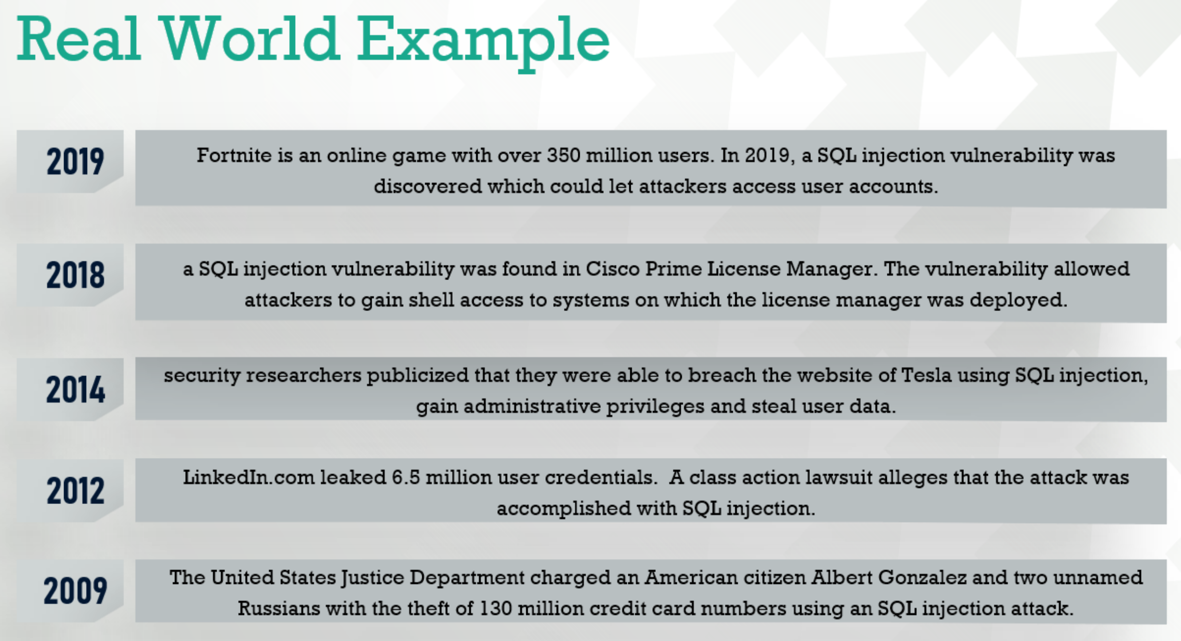
```
select * from Users where user_id= ' srinivas '
                  and password = ' mypassword '
```

User-Id : ` OR 1= 1; /*

Password : */--

```
select * from Users where user_id= '` OR 1 = 1; /* '
                  and password = ' */-- '
```

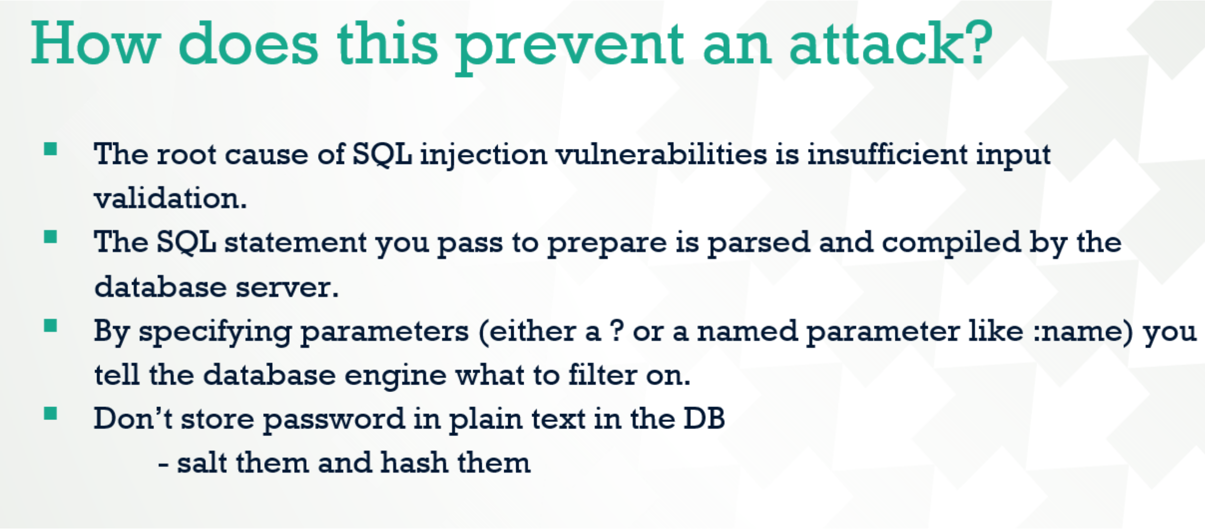
7. Real World Example



Real World Example

2019	Fortnite is an online game with over 350 million users. In 2019, a SQL injection vulnerability was discovered which could let attackers access user accounts.
2018	a SQL injection vulnerability was found in Cisco Prime License Manager. The vulnerability allowed attackers to gain shell access to systems on which the license manager was deployed.
2014	security researchers publicized that they were able to breach the website of Tesla using SQL injection, gain administrative privileges and steal user data.
2012	LinkedIn.com leaked 6.5 million user credentials. A class action lawsuit alleges that the attack was accomplished with SQL injection.
2009	The United States Justice Department charged an American citizen Albert Gonzalez and two unnamed Russians with the theft of 130 million credit card numbers using an SQL injection attack.

8. Prevent an attack



How does this prevent an attack?

- The root cause of SQL injection vulnerabilities is insufficient input validation.
- The SQL statement you pass to prepare is parsed and compiled by the database server.
- By specifying parameters (either a ? or a named parameter like :name) you tell the database engine what to filter on.
- Don't store password in plain text in the DB
  - salt them and hash them

5.2 References

1. SQL and SQL injection

https://en.wikipedia.org/wiki/SQL
https://www.imperva.com/learn/application-security/sql-injection-sqli/

2. SQL Injection type

https://crashtest-security.com/sql-injections/

3. Example Website

https://www.rapid7.com/fundamentals/sql-injection-attacks/

4. Real World Example

https://brightsec.com/blog/sql-injection-attack/

5. Prevent an SQL Injection attack

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

https://www.ptsecurity.com/ww-en/analytics/knowledge-base/how-to-prevent-sql-injection-attacks/

6. Conclusion

SQL injection is one of the more common and more effective forms of attack on a system. Controlling the malicious SQL code/script on the web application and maintaining the end privacy is still a key challenge for the web developer.

These issues must be considered seriously by the web developers involved in developing websites using databases.

SQL injection attacks are a growing criminal threat to your web applications, especially those that access sensitive data.

Other defences require much greater investment in deployment and support and should be used only on the most important or sensitive applications. With that in mind, here are the two most important things you can do to protect your applications from SQL injection attacks: Code defensively and monitor your most important applications.