Fundamental of Information Security and Blockchain (TCS332)

B. Tech CSE III Semester

Instructor:

Dr. Mohammad Wazid

Professor, Department of CSE

Head of Cyber security and IoT research group

Graphic Era (Deemed to be University), Dehradun, India

Email: wazidkec2005@gmail.com

Homepage: https://sites.google.com/site/mwazidiiith/home

SQL injection

SQL injection-Overview

- SQL injection is a code injection mechanism which is used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution.
- SQL injection technique 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-Mechanism

- 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
- Look at the example above again. The original purpose of the code was to create an SQL statement to select a user, with a given user id.
- 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 105 OR 1=1 is always TRUE.

- Does it 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.**

Types of SQL Injections

- SQL injections typically fall under three categories: In-band SQLi (Classic), Inferential SQLi (Blind) and Out-of-band SQLi.
- We can classify SQL injections types based on the methods they use to access backend data and their damage potential.

In-band SQLi

- The attacker uses the same channel of communication to launch their attacks and to gather their results.
- In-band SQLi's simplicity and efficiency make it one of the most common types of SQLi attack. There are two sub-variations of this method:

In-band SQLi-Types

- **Error-based SQLi-** attacker performs actions that cause the database to produce error messages.
- The attacker can potentially use the data provided by these error messages to gather information about the structure of the database.
- **Union-based SQLi-** this technique takes advantage of the UNION SQL operator, which fuses multiple select statements generated by the database to get a single HTTP response.
- UNION Syntax

SELECT column_name(s) FROM table1

UNION

SELECT column_name(s) FROM table2;

- This response may contain data that can be required by the attacker.

Inferential (Blind) SQLi

- The attacker sends data payloads to the server and observes the response and behavior of the server to learn more about its structure.
- This method is called blind SQLi because the data is not transferred from the website database to the attacker, thus the attacker cannot see information about the attack in-band.
- Blind SQL injections rely on the response and behavioral patterns of the server so they are typically slower to execute but may be just as harmful.
- Blind SQL injections can be classified as follows:

Inferential (Blind) SQLi

- **Boolean-** attacker sends a SQL query to the database prompting the application to return a result.
- The result will vary depending on whether the query is true or false.
- Based on the result, the information within the HTTP response will modify or stay unchanged.
- The attacker can then work out if the message generated a true or false result.
- **Time-based**-attacker sends a SQL query to the database, which makes the database wait (for a period in seconds) before it can react.
- The attacker can see from the time the database takes to respond, whether a query is true or false.
- Based on the result, an HTTP response will be generated after a waiting period.

Out-of-band SQLi

- The attacker can only carry out this form of attack when certain features are enabled on the database server used by the web application.
- This form of attack is primarily used as an alternative to the in-band and inferential SQLi techniques.
- Out-of-band SQLi is performed when the attacker can’t use the same channel to launch the attack and gather information, or when a server is too slow or unstable for these actions to be performed.
- These techniques count on the capacity of the server to create DNS or HTTP requests to transfer data to an attacker.

Solutions

- The first step is input validation (sanitization), which is the practice of writing code that can identify illegitimate user inputs.
- While input validation should always be considered best practice, it is rarely a foolproof solution.
- The reality is that, in most cases, it is simply not feasible to map out all legal and illegal inputs.
- For this reason, a web application firewall (WAF) is commonly employed to filter out SQLI, as well as other online threats.
- To do so, a WAF typically relies on a large, and constantly updated, list of meticulously crafted signatures that allow it to surgically filter out malicious SQL queries.

Solutions

- Usually, such a list holds signatures to address specific attack vectors and is regularly patched to introduce blocking rules for newly discovered vulnerabilities.
- Modern web application firewalls are also often integrated with other security solutions.
- For example, a web application firewall that encounters a suspicious, but not outright malicious input may cross-verify it with IP data before deciding to block the request.
- It only **blocks the input if the IP itself has a bad reputational history (attacker's URL).**

Example of SQL injection

- How to install sqlmap in windows
- https://www.youtube.com/watch?v=CdR7pw_FBj8
- Steps to launch the SQL injection
- C:\sqlmap>sqlmap.py -u http://testweb11.com/artists.php?artist=1 --dbs
- C:\sqlmap>sqlmap.py -u http://testweb11.com/artists.php?artist=1 -D acuart --tables
- C:\sqlmap>sqlmap.py -u http://testpweb11.com/artists.php?artist=1 -D acuart -T users --columns
- C:\sqlmap>sqlmap.py -u http://testweb11.com/artists.php?artist=1 -D acuart -T users -C uname --dump
- Output: test
- C:\sqlmap>sqlmap.py -u http://testpweb11.com/artists.php?artist=1 -D acuart -T users -C pass --dump
- Output: test
- Therefore we have discovered that username and password both are “test”.

References

- Textbook: Security in Computing, 5th Edition by C. P. Pfleeger, S. L. Pfleeger, J. Margulies
- SQL Injection attack, information available at: https://www.acunetix.com › websitesecurity › sql-injection