SQL Injection Attack

https://campus.barracuda.com/doc/42048250/

Description

SQL Injection Attacks take advantage of dynamically created SQL queries that use inadequately sanitized user input from web-based requests. By embedding SQL commands in user inputs, attackers can indirectly manipulate an SQL query issued by the web application to a database. Attackers can determine if an application is vulnerable and can view, modify, or delete application data by manipulating SQL queries dynamically. SQL Injection Attacks are simple to understand and attempt and allow access to sensitive or even critical application data, making them attractive to attackers.

Effects

Successful attacks compromise sensitive application data, can manipulate or delete the SQL database, can deface websites, can open reverse shells, or may even access the underlying OS. Injection attacks can be combined with Cross-Site Scripting (XSS) attacks.

Successful blind SQL injection attacks merely detect a variation in behavior, whether timing or response, based on an attempted injection to determine whether an application is vulnerable to SQL injection attacks.

Readily available off-the-shelf free tools (e.g., Havij, splmap) enable novice users to successfully execute injection attacks. In addition, highly sophisticated tools that intensely probe applications for obscure vulnerabilities are readily available.

Methods

The following input data can contain SQL Injection Attacks:

- URL Parameters Name or Value
- FORM parameters Name or Value
- Request Headers
- Cookies

Example

An attacker guesses that a web application uses the following SQL query to retrieve account details:

```
select accnt-detail from acct-master WHERE accnt=accntNumber
```

If `accntNumber` is a user input from an HTTP request, an attacker could make it:

```
5672' OR 'a'='a'
```

and the resulting query would be

```
select accnt-detail from acct-master WHERE accnt='5672' OR 'a'='a'
```

Since the query above always evaluates to true, every database row matches ('a'='a' is always TRUE) and the resulting query returns all `accnt-detail` entries from the database.

For SQL databases which allow semi-colon separated SQL commands to execute immediately, it is possible that input data containing arbitrary SQL commands could execute.

Prevention

SQL Injection Attacks can be prevented by secure coding, either by using prepared (parametrized) statements, or by properly escaping and sanitizing inputs. Either method can prevent execution of input data that contains commands. When using legacy or third-party code, or for tightly coupled interfaces where sanitizing inputs might not be possible, you can rely on Barracuda Web Application Firewall rule sets to inspect and block SQL injection across all input vectors. Inputs are first normalized to defeat obfuscation, such as encoding techniques (e.g., UTF8) or SQL specific techniques (e.g., embedded SQL comments to obfuscate the actual SQL query).

Tags

OWASP Top 10, PCI-DSS

See Also

CWE 89, OWASP, WASC