The WebOrion Software Solutions

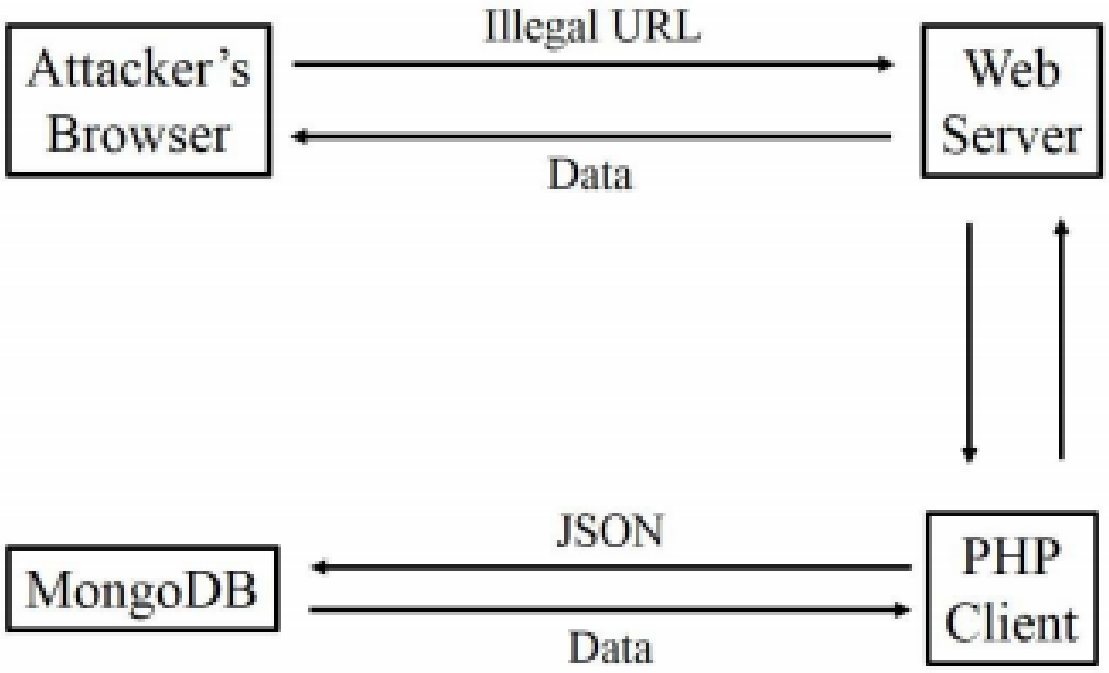
NoSQL Injection Attacks and Prevention Techniques

www.theweborion.com info@theweborion.in

NoSQL database stands for “Not Only SQL” or “Not SQL”. It is a non-relational Database Management System, that have a dynamic schema for document type or unstructured data. MongoDB, BigTable, Redis, etc. are the example of NoSQL database.

NoSQL Injection is security vulnerability that lets an attacker to inject code into the query that would be executed by the database. Five types of NoSQL attacks like Tautologies, Union queries, JavaScript injections, Piggybacked queries and Cross origin violation.

Below Figure show a scenario of NoSQL injection attack on MongoDB.



Types of NoSQL attack

1) Tautology: In the tautology attack the attacker tries to use a conditional query statement to be evaluated always true. These attacks allow bypassing authentication or access mechanisms by injecting code in conditional statements, generating expressions that are always true.

2) Union Queries: Union based injection allows an attacker to extract information from the database by extending the results returned by the original query. The most common uses of union queries are to bypass authentications pages and extract data.

3)JavaScript injections: JavaScript enables complicated transactions and queries on the database engine. Passing unsanitized user input to these queries might allow for injection of arbitrary JavaScript code, which could result in illegal data extraction or alteration.

Types of NoSQL attack

4) Piggybacked Queries: An attacker injects additional queries into the original query used by the Grade Central site to add, modify or delete student accounts. Attackers exploit assumptions in the interpretation of escape sequences special characters to insert additional queries to be executed by the database, which could lead to arbitrary code execution by attackers.

5) Origin Violation: “HTTP REST APIs” are a popular module in NoSQL databases. A new class of vulnerabilities that lets attackers target the database even from another domain. In cross origin attacks, attackers exploit legitimate users and their web browsers to perform an unwanted action. Such violations in the form of a cross-site request forgery attack in which the trust that a site has in a user’s browser is exploited to perform an illegal operation on a NoSQL database. By injecting an HTML form into a vulnerable website or tricking a user into the attacker’s own website. An attacker can perform a post action on the target database, thus compromising the database.

Example of NoSQL Injection using Node.js and MongoDB

Create a simple code when the inputs are not sanitized.

Inject Code:

```
app.post('/user', function (req, res){
var query = {
        username: req.body.username,
        password: req.body.password
}
db.collection('users').findone(query, function (err, user){
        console.log(user);
});
}) ;
```

Suppose that receive the following request

POST http://www.example.com/user HTTP/1.1

Content-Type: application/json

```
{
        “username”: {“$ne”: null},
        “password”: {“$ne”: null}
}
```

As **$ne** is the not equal operator, this request would return the first user without knowing its name or password.

Prevent NoSQL Injection

To prevent NoSQL injections, it is required to validate the user input or escape it properly. A very first and basic step is to validate user input, with regards to the following rules to confirm the expected type being received in the request is valid:

1) Validate length and type of the data

2) Validate and sanitize the input to an expected type (i.e. type casting)

Safe Code

```
var sanitize = require('mongo-sanitize');
app.post('/user', function (req, res){
        var query = {
                username: sanitize(req.body.username),
                password: sanitize(req.body.password)
        }
        db.collection(''users').findOne(query, function (err, user) {
                console.log(user);
        });
});
```

In this case solution is to sanitize the input before using them. A good option is mongo-sanitize. There is second solution, if you are using Mongoose, you don't need to sanitize the inputs. There is just need to set the properties to be typed as string. If someone passes an object like {$ne: null}, Mongoose will convert it to a string and no harm will be done.

About TheWeborion

- WebOrion™ – Trusted brand since 2012 for Cyber Security
- Our experts convert ideas into reality and add value to our customers by providing quality Cyber Security solutions.
- We thrive in providing security to all types of applications focusing on preventing cyber attacks and data clean-up after cyber incident.

Learn more:

Phone: +1-(202)-765-7053

Email: info@theweborion.com

Website: www.theweborion.com