Security Advisory

TypeORM
Prototype Pollution Leading To SQL Injection

Created by Norbert Szetei, Viktor Chuchurski
09/21/2022

Overview

This document summarizes the results of a vulnerability research activity in the TypeORM Object-relational mapping tool used by our customer as a third-party library.

We have identified a critical vulnerability which allows to pollute a parameter used to compose SQL queries. An attacker can easily exploit the finding as a SQL injection or Denial of Service.

About Us

Doyensec is an independent security research and development company focused on vulnerability discovery and remediation. We work at the intersection of software development and offensive engineering to help companies craft secure code.

Research is one of our founding principles and we invest heavily in it. By discovering new vulnerabilities and attack techniques, we constantly improve our capabilities and contribute to secure the applications we all use.

SQL Injection / Denial of Service Via Prototype Pollution	
Vendor	TypeORM
Severity	**Critical**
Vulnerability Class	Injection Flaws (SQL, XML, Command, Path, etc)
Component	TypeORM Library, affected versions 0.2.35 - 0.3.9
Status	Closed
CVE	CVE-2022-36531
Credits	Norbert Szetei, Viktor Chuchurski

Summary

The baseline expectation for any ORM is to avoid the possibility of SQL injection and to ensure that all the SQL queries are safely passed to the database. Since TypeORM is one of the most popular ORM solutions, currently having 960,405 weekly downloads on npmjs, guaranteeing its safety by sanitizing all potentially malicious user-supplied input is crucial.

Versions older than 0.2.24 were affected by a critical severity vulnerability, making it possible to inject directly into SQL queries. Successful exploitation of a SQL injection vulnerability can lead to the disclosure or modification of sensitive information to unauthorized users, compromising the confidentiality of the system.

Moreover, exploitation of the finding using incorrect SQL syntax leads to denial of service, affecting the user experience of its users and the general availability of the system.

Functions which internally called `mergeDeep`, such as `connection.manager.save`, were affected by prototype pollution. This allowed changing properties of different functions, resulting in a SQL injection.

Due to code refactoring, the issue was reintroduced approximately one year after the initial fix.

Technical Description

The bug was fixed in release 0.2.25 by ensuring that the user-supplied property with the prototype keyword `__proto__` is not processed:

```
$ git clone https://github.com/typeorm/typeorm
$ cd typeorm

$ git diff 0.2.24 0.2.25 src/util/OrmUtils.ts
[ .. SNIP .. ]
-                if (value instanceof Promise)
                if (key === "__proto__" || value instanceof Promise)
[ .. SNIP ..]
```

However, the refactoring of the code from the release 0.2.34 to 0.2.35 reintroduced the same bug by removing the `__proto__` check:

```
$ git diff 0.2.34 0.2.35 src/util/OrmUtils.ts
     static mergeDeep(target: any, ...sources: any[]): any {
-        if (!sources.length) return target;
-        const source = sources.shift();
-
-        if (this.isObject(target) && this.isObject(source)) {
-            for (const key in source) {
-                const value = source[key];
-                if (key === "__proto__" || value instanceof Promise)
-                    continue;
```

The reproduction steps are the same as for the previously reported vulnerability, based on the example code https://github.com/typeorm/typescript-example.

Additionally, if we add a printing statement into https://github.com/typeorm/typeorm/blob/0659ec395298390a2ec3e39ecae1ab4764c4e41a/src/util/OrmUtils.ts#L119 to output the `"value"` argument, we can dynamically confirm that the parameters are polluted.

For instance, injecting

```
const post = JSON.parse(`{"text":"a","title":{"__proto__":
{"where":{"id":2,"where":null}}}}`)
```

causes pollution of the object with

```
{ __proto__: { where: { id: 2, where: null } } }
```

Consequently, only the entry with `id = 2` is outputted.

Remediation

Reimplement the fix from version 0.2.25 to prevent merging objects with the prototype pollution payload.

We strongly recommend implementing a unit test to prevent regressions.

References

- SQL Injection or Denial of Service due to a Prototype Pollution
  https://hackerone.com/reports/869574

Disclosure Timeline

07/07/2022	Issue responsible disclosed to TypeORM dev team (Umed K.)
09/19/2022	TypeORM deployed a fix in TypeORM 0.3.10
09/21/2022	Advisory public release