**WRITE UP : CHALLENGE LOGIN! CỦA ACSC ASIA 2024**

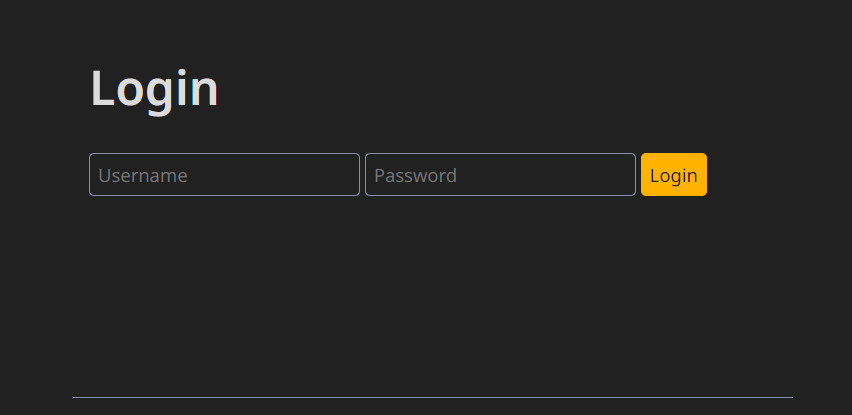
Hôm 29/03 đang ngồi giải lab WEB PENTEST của Cyberjutsu, thì thấy thông báo thầy Luật nhắn tin trong 1 channel discord nguyên văn : “Chơi đi các em” thì mình chơi luôn và có đường link dẫn vào link đăng ký cuộc thi ASC Asia.

Thế là cũng tò mò đó giờ chưa chơi CTF lớn bao giờ và tìm hiểu đăng ký chơi thử để có thêm “kiến thức, kinh nghiệm và trải nghiệm”.

Các challenge đều cho source code và docker-compose.yml như các bài lab của Cyberjutsu. Và nhờ sự trợ giúp, hỗ trợ rất nhiệt tình đó là anh DmKnght (Ông hoàng meme, top 1 lớp wpt03,… của lớp Cyberjutsu) thế là có writeup này.

**# Phân tích challenge**

Link : <http://login-web.chal.2024.ctf.acsc.asia:5000/>



Thử thách này là một bài toán whitebox, nơi mà có quyền truy cập vào mã nguồn bên trong môi trường Docker. Mã nguồn cho phép chúng ta đi sâu vào việc phân tích và debug:

Cách hoạt đông của ứng dụng:

const express = require('express');

const crypto = require('crypto');

const FLAG = *process*.*env*.FLAG || 'flag{this\_is\_a\_fake\_flag}';

const app = express();

app.use(express.urlencoded({ extended: true }));

const USER\_DB = {

    user: {

        username: 'user',

        password: *crypto*.randomBytes(32).toString('hex')

    },

    guest: {

        username: 'guest',

        password: 'guest'

    }

};

app.get('/', (*req*, *res*) => {

*res*.send(`

    <html><head><title>Login</title><link rel="stylesheet" href="https://cdn.simplecss.org/simple.min.css"></head>

    <body>

    <section>

    <h1>Login</h1>

    <form action="/login" method="post">

    <input type="text" name="username" placeholder="Username" length="6" required>

    <input type="password" name="password" placeholder="Password" required>

    <button type="submit">Login</button>

    </form>

    </section>

    </body></html>

    `);

});

app.post('/login', (*req*, *res*) => {

    const { username, password } = *req*.body;

    if (username.length > 6) return *res*.send('Username is too long');

    const user = USER\_DB[username];

    if (user && user.password == password) {

        if (username === 'guest') {

*res*.send('Welcome, guest. You do not have permission to view the flag');

        } else {

*res*.send(`Welcome, ${username}. Here is your flag: ${FLAG}`);

        }

    }else {

*res*.send('Invalid username or password');

    }

});

**#Phân tích vấn đề**

Ta có đoạn credential của user là hardcoded :

const USER\_DB = {

    user: {

        username: 'user',

        password: *crypto*.randomBytes(32).toString('hex')

    },

    guest: {

        username: 'guest',

        password: 'guest'

    }

};

Sau đó, chức năng đăng nhập:

app.post('/login', (*req*, *res*) => {

    const { username, password } = *req*.body;

    if (username.length > 6) return *res*.send('Username is too long');

    const user = USER\_DB[username];

    if (user && user.password == password) {

        if (username === 'guest') {

*res*.send('Welcome, guest. You do not have permission to view the flag');

        } else {

*res*.send(`Welcome, ${username}. Here is your flag: ${FLAG}`);

        }

    }else {

*res*.send('Invalid username or password');

    }

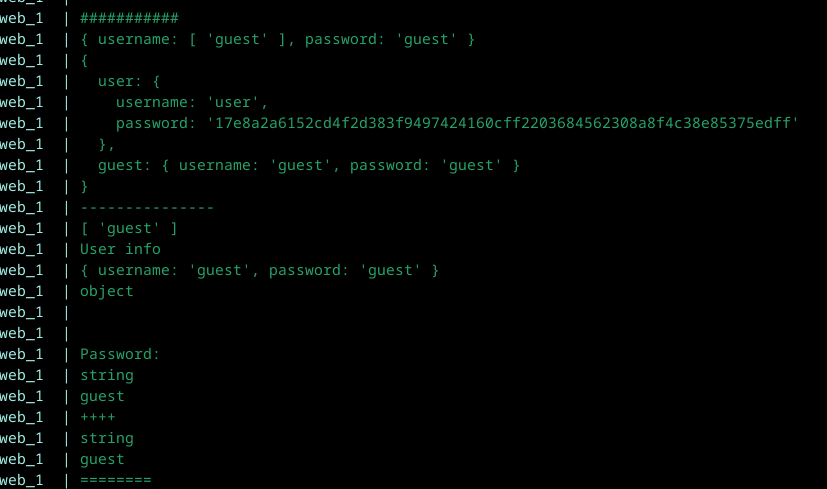
});

**# Đặt giả thuyết và chứng minh**

**Giả thuyết 1:** Khai thác **authorization** để lấy flag sau khi **authenticate**

**Chứng minh giả thuyết :**

Nhìn vào dòng if (user && user.password == password) đã nghĩ ngay tới sử dụng “Type Juggling” ( tự động chuyển đổi kiểu dữ liệu của giá trị trong biểu thức để phù hợp với ngữ cảnh ) nó sẽ giống như PHP nhưng sau đọc và tìm hiểu [<https://medium.com/infofl00d1ng/l00sy-c0mpar-s0n-ctf-write-up-guess-the-token-3-8f70fbf86bdf>] giả thuyết này đã thất bại vì password là string password: crypto.randomBytes(32).toString('hex')



🡪 Type juggling khi attack vào parameter password không thành công.

Vậy thì liệu có cách nào attack type juggling nhưng không vào param password không?

if (username === 'guest') {

*res*.send('Welcome, guest. You do not have permission to view the flag');

        } else {

*res*.send(`Welcome, ${username}. Here is your flag: ${FLAG}`);

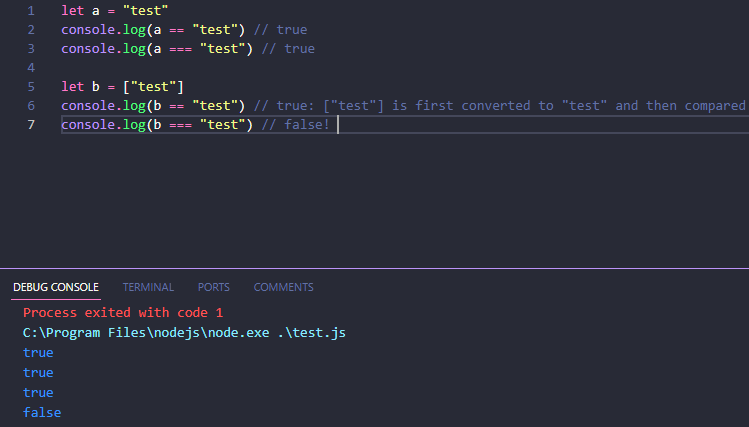
        }

Ta có 1 đoạn code cũng là “Type Juggling” mà mình đã bỏ quên và lí do mình bỏ quên là cho rằng bài cho 2 user mà để lấy flag thì phải sử dụng thằng user còn lại.Nhìn kỹ thì có phải **chỉ cần điều kiện nó trả về false** là có flag đúng không ?.

**Giả thuyết 2:** Liệu có cách nào attacker type juggling nhưng không vào param password không ?

**Chứng minh giả thuyết :**

Sau khi kham khảo [[JavaScript type confusion: Bypassed input validation (and how to ... - Snyk](https://snyk.io/blog/remediate-javascript-type-confusion-bypassed-input-validation/)]:



Theo như tài liệu ở trên thì == sẽ cố gắng ép kiểu để so sánh còn === (sẽ không ép kiểu) và có kết quả như hình trên.

Đến đây thì chúng ta đã đi được 50% sắp ra flag rồi. Làm tới đây thì chỉ cần mở Burp Suite lên gõ vào username=[‘guest’] là **xong tadaaaa** …. **NHƯNG** <input type="text" name="username" placeholder="Username" length="6" required> >>> **Username đã bị giới hạn dưới 6 chữ** ☹

Vậy câu hỏi tiếp theo làm sao để nó trở thành mảng lại phải đi search google thôi.

Sau khi bí quá, search không ra mình đã hỏi anh DmKnght và manh mối đó là **Rule Decode Query**. Từ manh mối đó hỏi con AI, google thì biết được những trang web mà sử dụng ngôn ngữ JavaScript nó sẽ xử lý chuỗi truy vấn như PHP [3] (<https://www.php.net/manual/en/function.parse-str.php>)

Theo tài liệu thi nó sẽ là username[]=’guest’ sau đó nó sẽ trở thành username=[‘guest’] điều này khá lạ với mình vì đây là lần đầu biết đến.

Hóa ra trong source code đã gợi ý cho mình nhưng mình không chú ý 2 dòng sau:

const app = express();

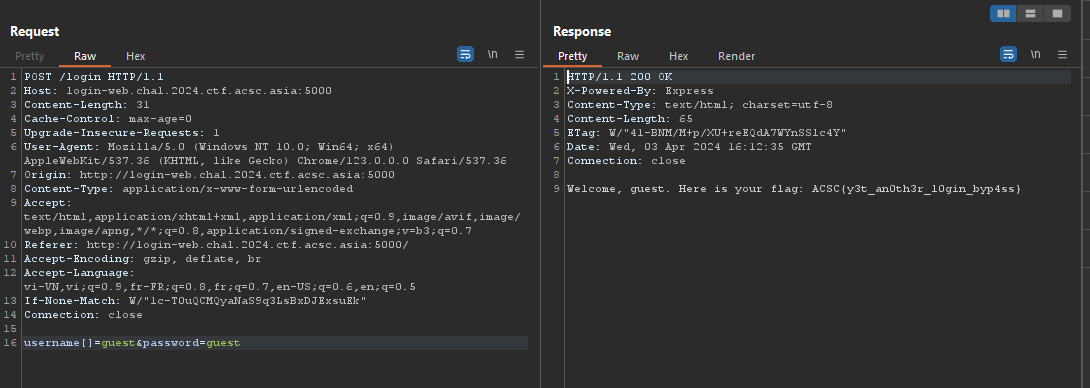
app.use(express.urlencoded({ extended: true }));

>> Khi thuộc tính extended được đặt thành true, dữ liệu được mã hóa URL sẽ được phân tích với thư viện qs để xử lý dữ liệu có thể là các object hoặc array lồng nhau

[<https://stackoverflow.com/questions/29960764/what-does-extended-mean-in-express-4-0>]

[<https://www.npmjs.com/package/qs>]

VÀ BÙM!!! :



**# KẾT LUẬN**

Mình đã học rất nhiều thứ ở challenge này, những cái mà không ngờ ngôn ngữ nó lại xử lý như thế.

Đây là 1 challenge khá hay vì không đánh đố và thuần về logic code

Mình là newbie nên có thể cách đặt giả thuyết mình hơi non và cách chứng minh chưa đầy đủ, có gì góp ý hoặc bỏ qua cho mình ạ 😊.

**# THAM KHẢO**

**[1]** <https://medium.com/infofl00d1ng/l00sy-c0mpar-s0n-ctf-write-up-guess-the-token-3-8f70fbf86bdf>

**[2]** [JavaScript type confusion: Bypassed input validation (and how to ... - Snyk](https://snyk.io/blog/remediate-javascript-type-confusion-bypassed-input-validation/)

**[3]** <https://www.php.net/manual/en/function.parse-str.php>

**[4]** <https://stackoverflow.com/questions/29960764/what-does-extended-mean-in-express-4-0>

**[5]** <https://www.npmjs.com/package/qs>