

STT	Tỷ đánh giá
Câu 1	100%
Câu 2	100%
Câu 3	100%
Câu 4	100%
Câu 5	100%
Câu 6	100%
Câu 7	100%
Câu 8	100%
Câu 9	100%
Câu 10	100%

Câu 1:

1. Thông tin chung

- Tên challenge: Admin has the power
- Mức độ: easy

2. Phân tích ban đầu

Web có chức năng đăng nhập phân quyền user và supporter

Đọc qua source của web thì thấy để lộ mật khẩu ở phần index

```
for maintenance purpose use this info (user:support password:x34245323)-->  
aUserStyle"></style>
```

Cookie cũng có thể thấy được và chỉnh sửa trong application

Dùng devtool

Cookie bị lộ và có thể chỉnh sửa, bằng cách này để lấy quyền admin

3. Quá trình khai thác

Chỉnh sửa cookie thành role admin

Name	Value	Do
PHPSESSID	f7tas042kq4hdq3fbitjs443ec	wc
role	support	wc

Sử dụng mật khẩu và username đã bị lộ để đăng nhập

Flag:

hiadminyouhavethepower

Hi admin

Admin Secret flag : **hiadminyouhavethepower**

4. Mức độ ảnh hưởng

Kiểu lỗ hổng này ảnh hưởng cực kì cao trong bất cứ web app nào áp dụng cơ chế phân quyền bằng cookie mà không verify chặt chẽ ở server. Chỉ cần thay đổi giá trị cookie (có sẵn thông tin đăng nhập “support” đọc được trong source comment), kẻ tấn công có thể:

- Toàn quyền xem, sửa, xóa dữ liệu nhạy cảm.
- Chiếm quyền admin để dựng backdoor, truy cập vào mọi khu vực bảo mật.
- Hạ thấp tính toàn vẹn và tính sẵn sàng của hệ thống (ngghiêm trọng hơn, có thể dùng payload để inject lệnh, xóa log, v.v.).

5. Cách khắc phục

- Không tin và không để client quyết định role hay permission.
- Lưu role/permission trên server (hoặc JWT có ký, mã hóa).
- Check quyền chặt chẽ ở server mỗi khi user gọi API.
- Bật HttpOnly/Secure/SameSite cho cookie.

Câu 2

1. Thông tin chung

Tên challenge: This is sparta

Mức độ: easy

2. Phân tích ban đầu

Web có chức năng đăng nhập

Đọc script thấy có đoạn bị làm rối

```
<script>
***
    var _0xae5b=
    [ "\x76\x61\x6C\x75\x65", "\x75\x73\x65\x72", "\x67\x65\x74\x45\x6C\x6D\x65\x6E\x74\x42\x79\x49\x61",
    check(){var _0xeb80x2=document[_0xae5b[2]](_0xae5b[1]][_0xae5b[0]];var _0xeb80x3=document[_0xae5b[2]]
    </script>
```

3. Quá trình khai thác

Thử decode đoạn bị làm rối thì thấy đó là các trường password, username,...
Ta thấy trong hàm check thì lại kiểm tra password và username trong chính frontend

[illegible]

ta biết được cả username và password là vì đó là phần tử thứ 4 trong mảng 0xae5b

```
> console.log("\x43\x79\x62\x65\x72\x2d\x54\x61\x6c\x65\x6e\x74")
Cyber-Talent
< undefined
>
```

Flag

A screenshot of a terminal window with a dark background. The text displayed is:
10.10.10.10
http://10.10.10.10:8080
Congratz
FLAG: {J4V4_Scr1Pt_1S_Aw3s0me}
In the bottom right corner, there is a blue button with the text 'OK' in white.

{J4V4 Scr1Pt 1S Aw3s0me}

4. Mức độ ảnh hưởng

Loại lỗ hổng: Client-side authentication (Broken Authentication) – ứng dụng tin tưởng hoàn toàn vào JS ở trình duyệt để xác thực người dùng. Theo OWASP Top 10:2021, đây thuộc mục A07:2021 – Identification and Authentication Failures.

Impact:

- Toàn bộ cơ chế login vô tác dụng, attacker có thể bypass dễ dàng và truy xuất mọi tính năng (unauthorized access).
- Mất confidentiality (flag, data nhạy cảm), phá vỡ tính toàn vẹn và có thể ảnh hưởng đến tính sẵn sàng nếu attacker spam request.
- Theo thống kê OWASP, Avg Weighted Impact ≈ 6.50 (High)

5. Cách khắc phục

Một số cách khắc phục:

- Đẩy hoàn toàn logic auth về server:
 - Client chỉ gửi username/password qua HTTPS tới API; server kiểm tra, so sánh với mật khẩu đã hash (bcrypt/Argon2) trong DB.
 - Server sinh session cookie (HttpOnly, Secure, SameSite) hoặc JWT đã ký; client không còn lưu bất kỳ secret nào trong code.
- Không hard-code hay obfuscate secret trong frontend:
 - Dẫu có obfuscate, JS vẫn chạy trên trình duyệt nên attacker decode rất dễ.
 - Mọi thông tin nhạy cảm (credentials, token cấp đặc quyền) chỉ lưu trên server-side.
- Áp dụng chính sách bảo mật của OWASP:
 - Thực hiện rate-limit, CAPTCHA cho endpoints login để ngăn brute-force.
 - Dùng multi-factor authentication nếu cần nâng cao bảo mật.
- Kiểm thử định kỳ & code review: Đưa challenge kiểu này vào scope pentest: tìm mọi logic auth còn sót lại ở client.

Câu 3

1. Thông tin chung

Tên challenge: lam Legend

Mức độ: easy

2. Phân tích ban đầu

Web có chức năng đăng nhập,

Thấy có 1 đoạn script bất thường trong source:

```
</form>
<script>
  [][[] + [][+[]]]+[[]][++][++][++][++][[]][+[]][+[]][+[]][+[]] + [[] + [
</script>
```

Sau khi tìm hiểu bằng internet có thể đây là JSFuck

3. Quá trình khai thác

Giải mã thì ra được

```

0.
String.fromCharCode(102,117,110,99,116,105,111,110,32,99,104,101,99,107,40,41
,123,10,10,118,97,114,32,117,115,101,114,32,61,32,100,111,99,117,109,101,110,
116,91,34,103,101,116,69,108,101,109,101,110,116,66,121,73,100,34,93,40,34,11
7,115,101,114,34,41,91,34,118,97,108,117,101,34,93,59,10,118,97,114,32,112,97
,115,115,32,61,32,100,111,99,117,109,101,110,116,91,34,103,101,116,69,108,101
,109,101,110,116,66,121,73,100,34,93,40,34,112,97,115,115,34,41,91,34,118,97,
108,117,101,34,93,59,10,10,105,102,40,117,115,101,114,61,61,34,67,121,98,101,
114,34,32,38,38,32,112,97,115,115,61,61,32,34,84,97,108,101,110,116,34,41,123
,97,108,101,114,116,40,34,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
32,32,32,32,32,67,111,110,103,114,97,116,122,32,92,110,32,70,108,97,103,58,32
,123,74,52,86,52,95,83,99,114,49,80,116,95,49,83,95,83,48,95,68,52,77,78,95,7
0,85,78,125,34,41,59,125,32,10,101,108,115,101,32,123,97,108,101,114,116,40,3
4,119,114,111,110,103,32,80,97,115,115,119,111,114,100,34,41,59,125,125)

1. function check(){ var user = document["getElementById"]("user")["value"];
var pass = document["getElementById"]("pass")["value"]; if(user=="Cyber" &&
pass== "Talent"){alert(" Congratz \n Flag: {J4V4_Scr1Pt_1S_S0_D4MN_FUN}");}
else {alert("wrong Password");}}

```

Ta biết được flag là {J4V4_Scr1Pt_1S_S0_D4MN_FUN}

4. Mức độ ảnh hưởng

Loại lỗi: Information Exposure (CWE-200) vì app tin vào “bảo mật qua tầm bông” – chỉ obfuscate JS chứ không hề mã hóa hoặc ký số, client có thể bóc ra plaintext credential

Theo OWASP Top 10: Hạng mục này trước là A3:2017-Sensitive Data Exposure, nay đổi tên thành A02:2021-Cryptographic Failures, nhấn mạnh việc lộ key/secret do crypto/cấu hình kém

Mức CVSS: thường rơi vào khoảng 6.5–8.0 (High), vì “exposed credentials” trực tiếp dẫn đến unauthorized access – attacker có thể login dưới quyền bất kỳ user nào (thậm chí admin nếu có flag tương ứng), xâm phạm toàn bộ dữ liệu và chức năng của hệ thống.

5. Cách khắc phục

Tách hẳn phần auth về server

Không hard-code bất kỳ secret nào trong front-end: Dùng biến môi trường (process.env) trên server hoặc trong quá trình build CI/CD, inject vào server chứ không nhét vô bundle.

Mã hóa và ký số nếu cần truyền dữ liệu nhạy cảm: Nếu client vẫn phải nhận một số data nhạy, hãy đóng gói trong JWT hoặc payload đã được HMAC/RSA ký.

Dùng HTTPS + header bảo mật: Bật Strict-Transport-Security để chặn downgrade. Thêm Content-Security-Policy để ngăn việc load script lạ hay sniff code.

Quét secret trước khi deploy: Trong pipeline (CI), tích hợp tool như git-secrets, truffleHog để phát hiện vô tình commit API key, credentials.

Thiết kế role/permission rõ ràng ở server: Mỗi API endpoint check token/session, verify quyền mới cho phép truy cập.

Câu 4

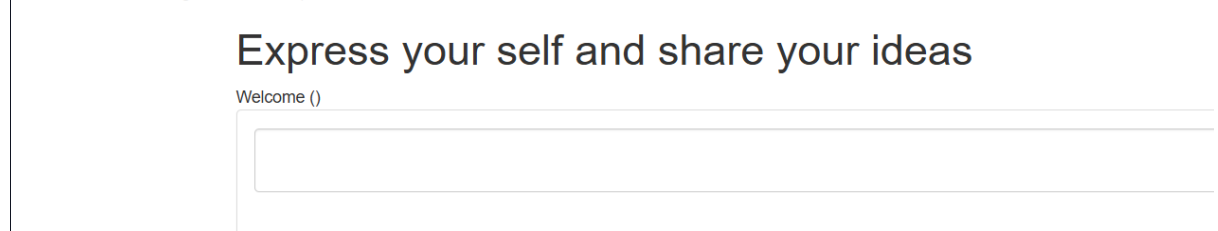
1. Thông tin chung

2. Phân tích ban đầu

Web có chức năng đăng các ideas, có thể là lấy dữ liệu từ database

Thử nhập đầu "" xem có thể là lỗi SQLi không

Error : HY000 1 unrecognized token: """)"

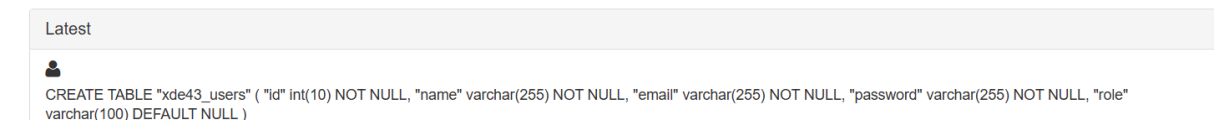
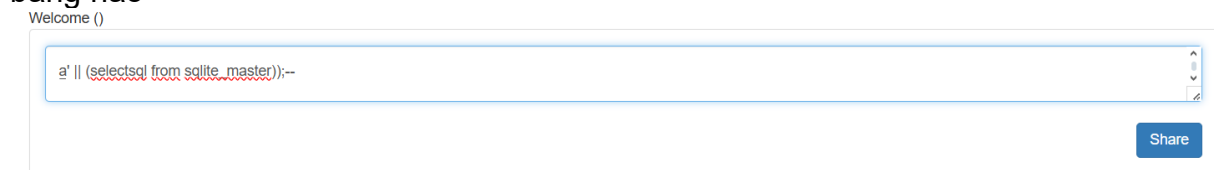


có thể khai thác SQLi

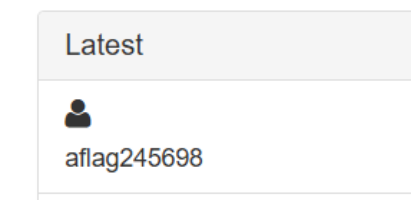
thêm vào đó khi báo về lỗi near "thông điệp nhập vào": syntax error ta biết được hệ thống dùng sqlite

3. Quá trình khai thác

Dùng lệnh ' || (select sql from sqlite_master));-- để biết được database có các bảng nào



Nhập câu lệnh SQL a' || (select password from xde43_users where role='admin')); để lấy được mật khẩu admin



4. Mức độ ảnh hưởng

- Loại lỗ hổng: SQL Injection (CWE-89), thuộc OWASP A03:2021-Injection, với mức average weighted impact ~7.15/10 – clearly High severity

- Impact thực tế: attacker có thể

- Dump toàn bộ database (user list, password hash,...).
- Lấy luôn password của admin, leo thang quyền.
- Xâm phạm tính bảo mật (confidentiality), phá vỡ tính toàn vẹn (integrity) và có thể ảnh hưởng đến tính sẵn sàng (availability) của hệ thống.

5. Cách khắc phục

Parameterized Queries / Prepared Statements:

- Tuyệt đối không build SQL bằng string concat.
- Dùng API như PDO::prepare() (PHP), sqlite3_prepare_v2() (C) để user input luôn được xử lý là data, không phải code

Input Validation & Sanitization:

- Áp whitelist (chỉ cho phép ký tự cần thiết), reject các ký tự đặc biệt nguy hiểm.
- Kết hợp escaping nếu bắt buộc phải dùng dynamic SQL

Giảm quyền của DB user: Tạo user DB chỉ có quyền SELECT, INSERT trên bảng comment/chức năng bình thường; không grant quyền đọc bảng user/admin.

Không tiết lộ chi tiết lỗi: Tắt stack trace và error message gốc khi query lỗi, tránh leak thông tin DB schema hay engine.

Áp dụng WAF / RASP: Chặn các request chứa pattern injection phổ biến (--, ||, v.v.), giám sát và cảnh báo sớm.

Câu 5

1. Thông tin chung

- Tên challenge: Cool Name Effect
- Mức độ: Easy

2. Phân tích ban đầu

Khi đọc src, ta thấy đoạn script khả nghi và có chữ eval

```
... eval(function(p,a,c,k,e,d){e=function(c){return(c<a?'':e(parseInt(c/a)))+(c=c%a)>35?St
    $e=$(4),14=$e.1o(R);8.B+=14;$e.k(\`11\`,8.B-14/2));7 1c=4.B/2;b(4.5.9<1c)4.5.9=1c;4.1X=4.E
    {$e.k(4,\`c\`,2Q $e.l(5,4))}})d 4}}(1f);(6(j,w){7 24=1U;7 U=6(){7 z=[\`y\`,\`o\`,\`u\`,\`r\`
    (1f,X);\`62,200,\`|||this|options|function|var|_self|radius|if|arctext|return|letter|}|}|
    {})); == $0
```

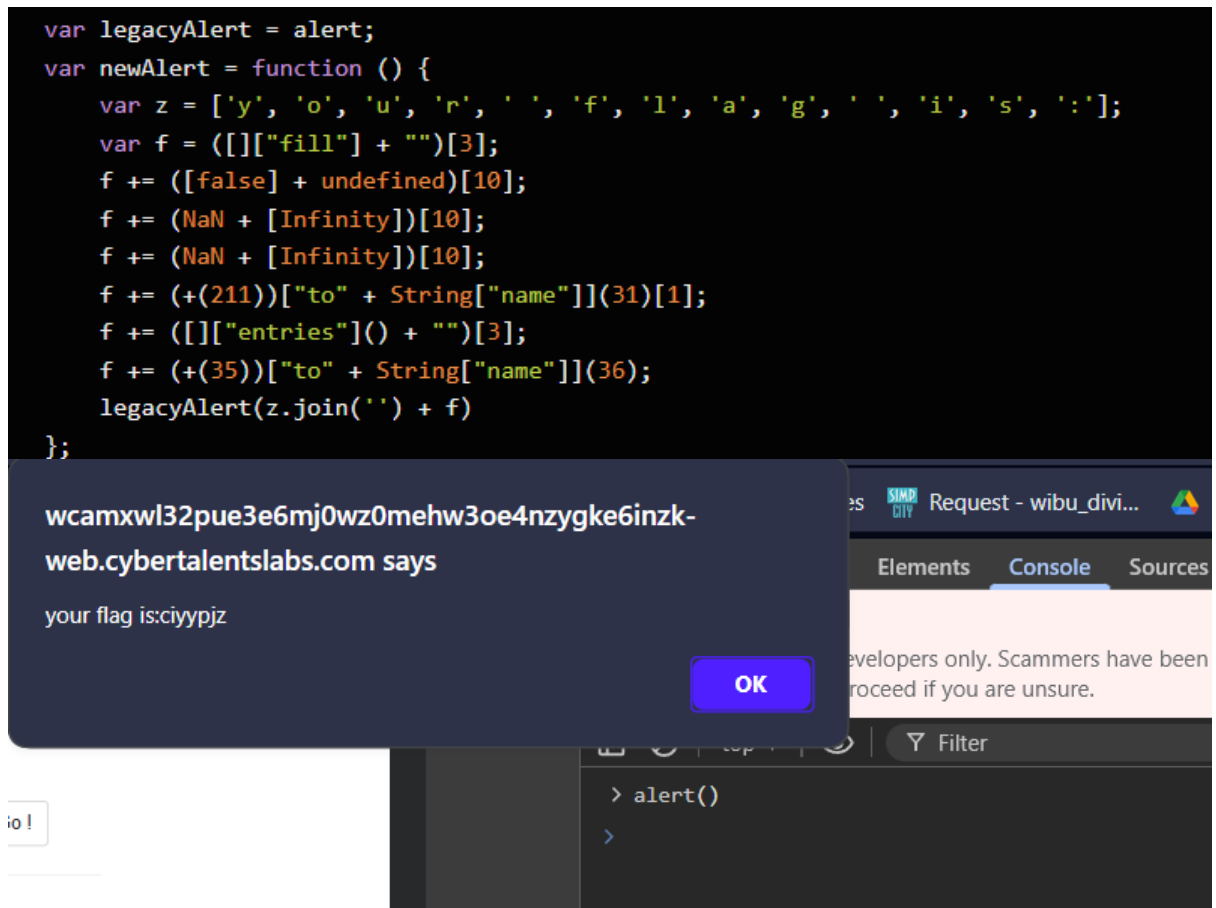
Dùng de4js để hiểu được đoạn script

```
(function ($, undefined) {
    $.fn.fitText = function (kompressor, options) {
        var settings = {
            'minFontSize': Number.NEGATIVE_INFINITY,
            'maxFontSize': Number.POSITIVE_INFINITY
        };
        return this.each(function () {
            var $this = $(this);
            var kompressor = kompressor || 1;
            if (options) {
                $.extend(settings, options)
            }
            var resizer = function () {
                $this.css('font-size', Math.max(Math.min($this.width() / (kompressor * 10), parseFloat(settings.maxFontSize)),
                    settings.minFontSize));
            };
            resizer();
            $(window).resize(resizer)
        })
    };

    function injector(t, splitter, klass, after) {
        var a = t.text().split(splitter),
            inject = '',
            emptyclass;
        if (a.length) {
            $(a).each(function (i, item) {
                emptyclass = '';
                if (item === ' ') {
                    emptyclass = ' empty';
                    item = '&nbsp;';
                }
            });
        }
    }
})
```

3. Quá trình khai thác

Khi đọc code bản rõ thì ta thấy được chỉ cần gọi alert() là hiện ra flag
your flag is:ciyypjz



4. Mức độ ảnh hưởng

Loại lỗ hổng: Reflected XSS (CWE-79), thuộc nhóm Injection theo OWASP Top 10 (XSS nằm trong A03:2021-Injection)

CVSS 3.x điển hình cho reflected XSS (AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N) thường ~8.8 (High)

Impact:

- Kẻ tấn công có thể chạy JS tùy ý trong context của nạn nhân → cướp cookie (HttpOnly nếu không bật), đọc localStorage, thay đổi UI, redirect, phishing, CSRF,...
- Phá vỡ confidentiality (cookie, token), integrity (thay UI, dữ liệu), đôi khi ảnh hưởng availability (spam alert, redirect loop).

5. Cách khắc phục

Output Encoding / Escaping:

- a. Khi render tên, phải escape mọi ký tự đặc biệt (<, >, ", &, ...) thành HTML entities.
- b. Dùng template engine/framework có auto-escaping (React/Vue/Angular binding, Thymeleaf, Handlebars, v.v.) để client không chèn được thẳng HTML.

Input Validation (Whitelist): Chỉ cho phép chữ cái, số, dấu cách... trên trường tên. Bất kỳ ký tự "<>" hoặc dấu ngoặc kép nào khác phải bị reject hoặc encode.

Content Security Policy (CSP): Thiết chính CSP chặt chẽ, chỉ cho phép JS từ các nguồn tin cậy (script-src 'self'), chặn inline script/onerror...
 Cookie Flags & Secure Defaults: Luôn set HttpOnly; Secure;
 SameSite=Strict/Lax cho session cookie, giảm nguy cơ JS ăn cắp cookie.
 Sử dụng OWASP XSS Prevention Cheat Sheet: Áp dụng các kỹ thuật trong cheatsheetseries.owasp.org: output encoding theo context (HTML, JS, URL, CSS), avoid innerHTML, dùng textContent, v.v.
 Kiểm thử & Code Review: Thêm test case XSS vào CI, tự động thử payload kiểu hay <svg/onload> để đảm bảo filter không bypass được.

Câu 6

1. Thông tin chung

Tên challenge: who am i?

Mức độ: Easy

2. Phân tích ban đầu

Web có chức năng đăng nhập, phân quyền người dùng
 Web cho ta tài khoản guest và không còn gì hết

```
<center> == $0
  > <form method="POST"> ... </form>
  <!--
      Guest Account:
      -----
      Username:Guest
      Password:Guest
  -->
</center>
</font>
</center>
</body>
```

Ý tưởng, thử sửa đổi cookie để leo quyền

3. Quá trình khai thác

Request Cookies ☐ show filtered out request cookies

Name	Value	Domain	Path	Ex...	Size	Htt...	Sec...	Sa...	Par...	Cro...	Pri...
Authentication	bG9naW49R3Vlc3Q%3D	wcamx...	/	Se...	32						Me...

Khi decode cookie ta thấy nó có nghĩa login=Guest

Ta thử đổi thành login=Admin thì encode ra bG9naW49QWRtaW4=

Dimensions: Responsive 846 x 743 72% Custom

Welcome, Administrator !

Congratulation. Your Flag is :

Flag{B@D_4uTh1Nt1C4T10n}

Application: Manifest, Service workers, Storage

Storage: Local storage, Session storage, Extension storage, IndexedDB, Cookies

Filter: Name, Value

Authentication: bG9naW49QWRtaW4=

Flag là FFlag{B@D_4uTh1Nt1C4Ti0n}

4. Mức độ ảnh hưởng

Đây là Broken Authentication kết hợp Broken Access Control (CWE-287, CWE-200): ứng dụng tin tuyệt đối vào cookie client-side, không verify server-side

Kẻ tấn công có thể leo thang đặc quyền (Privilege Escalation) lên admin chỉ bằng vài dòng Base64, bypass hoàn toàn cơ chế auth.

Ảnh hưởng chủ yếu là mất tính bảo mật (confidentiality) và mất tính toàn vẹn (integrity) của hệ thống; tính sẵn sàng ít chịu tác động.

Theo tiêu chuẩn CVSS, lỗ hổng dạng này thường được đánh giá ở mức High (7.0–9.0).

5. Cách khắc phục

Session Management an toàn: Không lưu role/permission dưới dạng plaintext/Base64 trong cookie.

Kiểm tra quyền server-side: Mỗi request chỉ dựa vào session/token trên server để xác thực role; không tin bất kỳ trường “role” nào từ client.

Cookie flags: Bật HttpOnly, Secure và SameSite=Strict/Lax để ngăn client-side script hoặc kẻ MITM truy cập/sửa cookie

Loại bỏ debug/comment nhạy cảm: Xóa hết thông tin user/pass hay flag trong source code, comment trước khi deploy.

Giám sát và test thường xuyên: Tích hợp CI pipeline quét secret (git-secrets), pentest check privilege escalation, hoặc dùng WAF chặn cookie bất thường

Câu 7

1. Thông tin chung

Tên challenge: Newsletter

Mức độ: Easy

2. Phân tích ban đầu

Ý tưởng: lấy các file từ folder nên ta có thể thử tấn công bằng payload với câu lệnh lấy file name

3. Quá trình khai thác

Chỉnh sửa định dạng nhập vào

```
<h1>Super NewsLetter</h1>
<form action method="post">
  <input type="text" name="email" value> == $0
  <input type="submit" value="enter">
</form>
::after
</div>
<!-- jQuery (necessary for Bootstrap's JavaScript plugin
```

Dùng lệnh ;ls #@. Để lấy được tên file

```
emails_secret_1337.txt hgdr64.backup.tar.gz index.php
```

Your email inserted successfully

hgdr64.backup.tar.gz

4. Mức độ ảnh hưởng
 - Loại lỗ hổng: OS Command Injection (CWE-78), nằm trong OWASP Top 10 A03:2021-Injection
 - Kẻ tấn công có thể thực thi bất cứ lệnh hệ thống nào với quyền của ứng dụng (đọc/xóa/ghi file, escalate, mở reverse shell...), dẫn đến RCE hoàn toàn.
 - CVSS (Base) kiểu này thường ở mức 10.0 CRITICAL (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)
5. Cách khắc phục

Tuyệt đối không gọi shell command trực tiếp: Dùng API/ngôn ngữ tích hợp (ví dụ: PHP mail(), Python smtplib) thay vì system() hay exec()
Nếu bắt buộc phải gọi shell, phải escape input
Validate & whitelist: Chỉ chấp nhận đúng định dạng email chuẩn qua regex kiểu /^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\$/.
Chạy tiến trình dưới user ít quyền nhất: Không cho web server chạy dưới root; chỉ đủ quyền đọc ghi thư mục cần thiết.
Harden & giám sát: Thêm WAF/RASP để chặn pattern “;”, “&&” trong tham số, kiểm tra log, alert ngay khi thấy output bất thường từ lệnh hệ thống.

Câu 8

1. Thông tin chung

Tên challenge: Encrypted Database
Mức độ: Easy
2. Phân tích ban đầu

Khi đọc src ta thấy để lộ endpoint admin
3. Quá trình khai thác

Khi vào endpoint admin thì lỗi 403, không có manh mối về mật khẩu và tài khoản nhưng có để lộ 1 endpoint khác

```
<!-- jQuery (necessary for Bootstrap's JavaScript plug
<script src="admin/assets/app.js"></script>
</body>
```

Ý tưởng sẽ là tấn công, khai thác vào endpoint này

```
</div>
  <div class="form-group">⋮ </div>
  <input type="hidden" name="db" value="secret-database/db.json"> == $0
  <div class="form-group">⋮ </div>
</form>
::after
</div>
</div>
```

Truy cập vào end point mới này thì thấy được flag có vẻ đã mã hóa

"flag": "ab003765f3424bf8e2c8d1d69762d72c"} }

Thử giải mã ta có được

MD5 Hash	Text
ab003765f3424bf8e2c8d1d69762d72c	badboy
»	

4. Mức độ ảnh hưởng

Đây là **Broken Access Control** (A01:2021) vì endpoint nhạy cảm (file DB) được phơi bày cho client mà không có xác thực/ủy quyền server-side. Đồng thời là **Cryptographic Failures** (A02:2021) khi dùng MD5 – thuật toán đã bị bẻ khóa dễ dàng với rainbow tables, không phù hợp để bảo vệ dữ liệu nhạy cảm.

Impact: attacker có thể truy xuất toàn bộ cơ sở dữ liệu hoặc thông tin nhạy cảm, giải mã trong chớp mắt, dẫn đến mất hoàn toàn tính **confidentiality** và **integrity**. CVSS Base ~7.5–9.0 (High to Critical).

5. Cách khắc phục

Giữ chặt access control server-side: Tắt directory listing, loại bỏ file JSON khỏi web-root công khai hoặc chỉ cho phép truy cập sau khi đã xác thực và phân quyền rõ ràng.

Thay MD5 bằng hashing/bảo mật mạnh: Không dùng MD5 để hash dữ liệu quan trọng. Chuyển sang Argon2/scrypt/bcrypt/PBKDF2 có salt và work factor, đảm bảo chống GPU cracking.

Loại bỏ mọi đầu mối nhạy cảm trên client: Không để các đường dẫn quan trọng (/admin, /secret-database) hoặc thông tin DB trong code client.

Câu 9

1. Thông tin chung

Tên challenge: Easy Message

- Mức độ: Easy

2. Phân tích ban đầu

Đọc qua src không thấy có gì bất thường, thử khai thác robot.txt

3. Quá trình khai thác

Khi truy cập robots.txt thì lộ /?source

```
User-agent: *
Disallow: /?source
```

Thử truy cập /?source

<?php

```
$user = $_POST['user'];  
$pass = $_POST['pass'];
```

```
include('db.php');
```

```
if ($user == base64_decode('Q3liZXItVGFsZW50') && $pass == base64_decode('Q3liZXItVGFsZW50'))
{
    success_login();
}
else {
    failed_login();
}
```

?

Ta thấy để lộ user và pass ở dạng base64

Giải mã ra ta được Cyber-Talent

Khi đăng nhập với các thông tin thu thập đc

I Have a Message For You

[illegible]

Thử decode message thấy đây là mã morse: FLAG(I-KN0W-Y0U-AR3-M0RS3)

4. **Mức độ ảnh hưởng**
Information Disclosure (CWE-200) do lộ source cùng credentials client-side encoded bằng Base64.
Security Misconfiguration (A05:2021) khi để lộ endpoint nhạy cảm qua robots.txt mà không require auth phía server.
Cryptographic Failures (A02:2021) vì dùng Base64 làm “mã hoá” chứ không phải crypto thực thụ, dẫn đến sensitive data bị giải mã dễ dàng
Impact: attacker bypass auth, đọc hết dữ liệu, lấy flag, thậm chí chiếm tài khoản admin nếu có; confidentiality và integrity bị phá; CVSS tầm 5–7 (Medium–High) tùy kịch bản.

5. Cách khắc phục

Đặt auth bắt buộc cho mọi endpoint nhạy cảm (kể cả `/?source`): server phải kiểm tra session/token trước khi trả code

Loại bỏ secrets client-side: không hard-code username/password hay flag trong code, không đưa vào HTML/JS.

Dùng crypto đúng nghĩa: thay Base64 thành JWT/HMAC-signed token hoặc session ID random, verify signature server-side.

Xóa hoặc chặn robots.txt với những đường dẫn nhạy cảm, hoặc chỉ cho user đã auth xem.

Apply Principle of Least Privilege: các file, folder chứa logic nội bộ chỉ cho user có đúng quyền mới truy cập được.

Review & Test định kỳ: tích hợp CI/CD quét secrets (git-secrets), pentest kiểm tra Security Misconfiguration.

Câu 10

1. Thông tin chung

Tên challenge: Cheers

Mức độ: Easy

2. Phân tích ban đầu

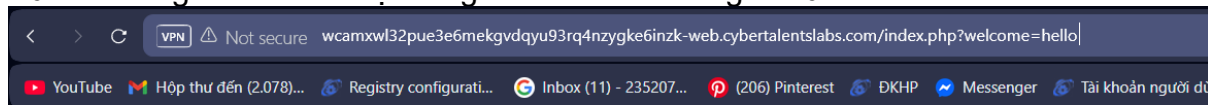
Ops!! Can You Fix This Error For Me Please ??!

Notice: Undefined index: welcome in /var/www/html/index.php on line 14

Khai thác từ thông tin trên lỗi đã hiển thị

3. Quá trình khai thác

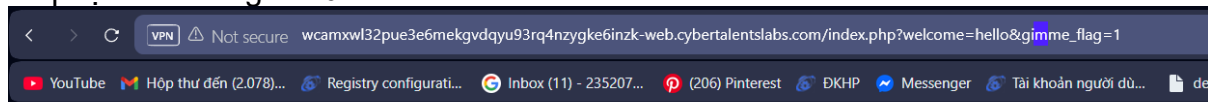
Sửa lỗi bằng cách thêm nội dung vào url theo thông tin từ lỗi



Hello !!!

Notice: Undefined index: gimme_flag in /var/www/html/index.php on line 19

Tiếp tục với thông tin từ lỗi mới



Hello !!!

FLAG{k33p_c4lm_st4rt_c0d!ng}

4. Mức độ ảnh hưởng

Loại lỗ hổng: Information Exposure Through an Error Message (CWE-209) – ứng dụng tiết lộ quá nhiều thông tin triển khai (tên biến, vị trí file, dòng lệnh) qua lỗi hiển thị công khai

OWASP Top 10: A05:2021 – Security Misconfiguration (verbose error messages)

Impact:

- a. Mất hoàn toàn **confidentiality**, flag (hoặc bất kỳ dữ liệu nhạy cảm nào) bị lộ ngay lập tức.
- b. Thông tin đường dẫn, tên biến giúp attacker dễ dàng tìm thêm lỗ hổng khác.

CVSS 3.x: thường ở mức **5.3 (Medium)** cho lỗi lộ thông tin qua error message

5. Cách khắc phục

Tắt hiển thị lỗi ra client

Kiểm tra tồn tại biến trước khi dùng

Xử lý lỗi nội bộ

- a. Không echo trực tiếp stack trace hay thông tin triển khai.
- b. Ghi chi tiết lỗi vào log server, hiển thị cho user thông báo chung chung (“Đã xảy ra lỗi, vui lòng thử lại sau”)

Thiết lập trang lỗi mặc định: Dùng .htaccess hoặc cấu hình web server để chuyển hướng tất cả lỗi 500, 404 về trang tĩnh an toàn.

Áp dụng principle “fail-safe”

- c. Mặc định giả sử giá trị đầu vào không hợp lệ → không trả dữ liệu quan trọng.
- d. Chỉ khi đã validate xong và cấp phép rõ ràng mới reveal bất cứ thông tin nhạy cảm nào.