

Nama : Latief Reswandana

Kelas : III RPLK

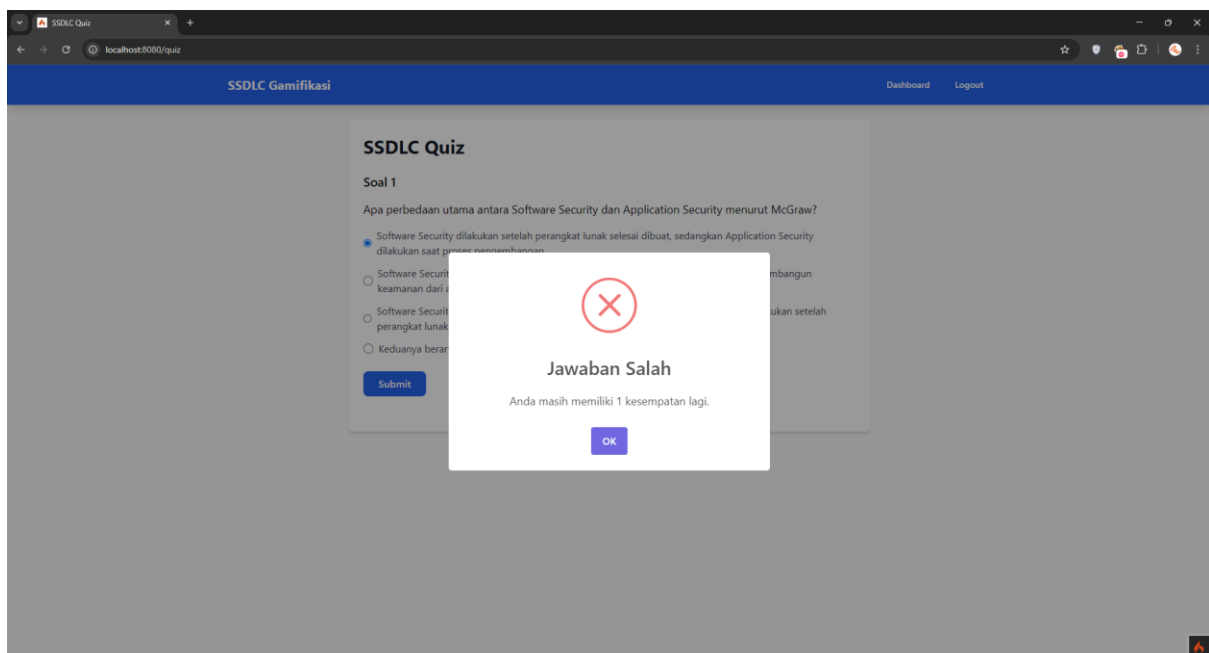
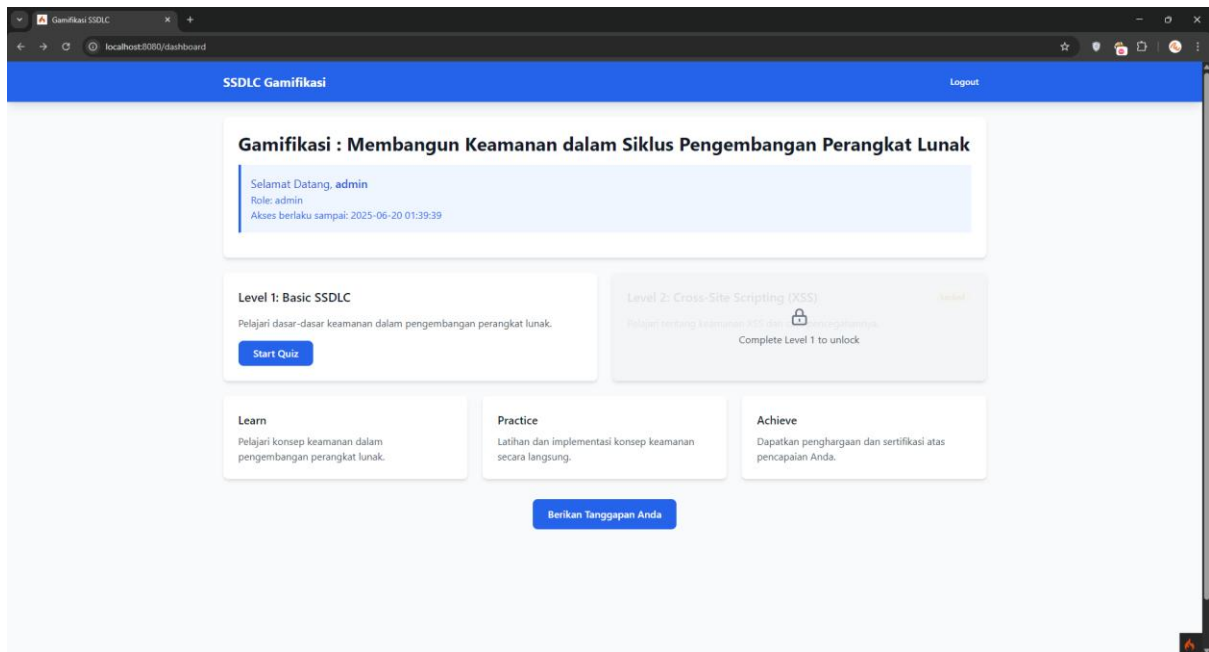
Soal : 3

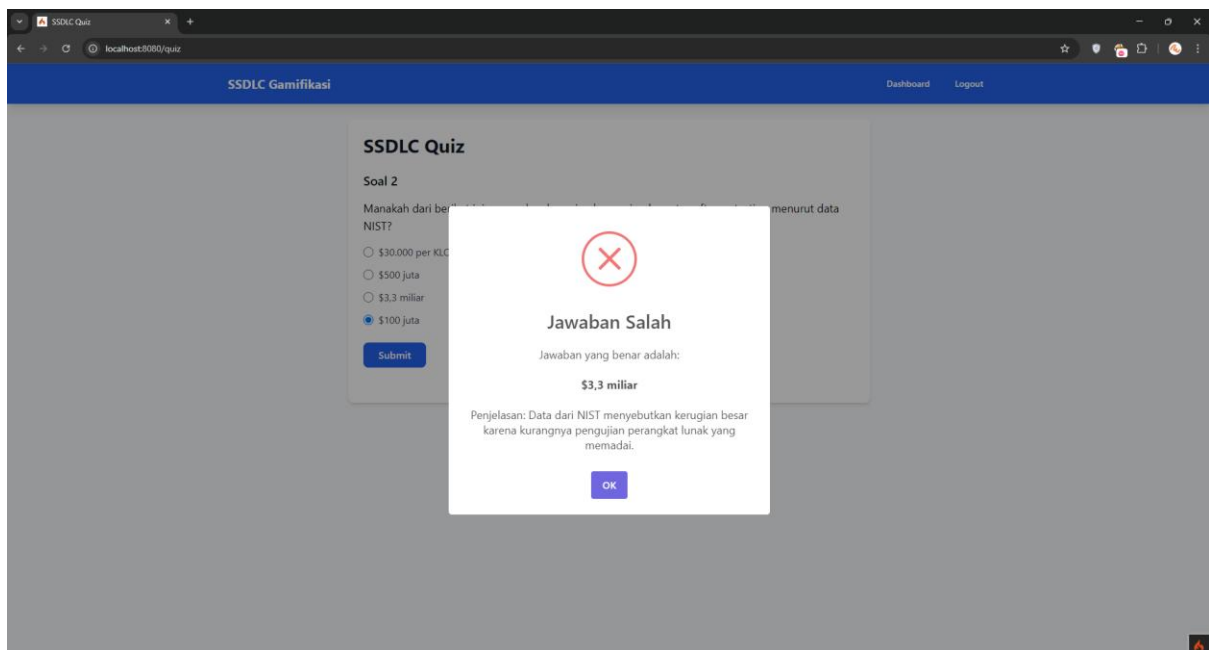
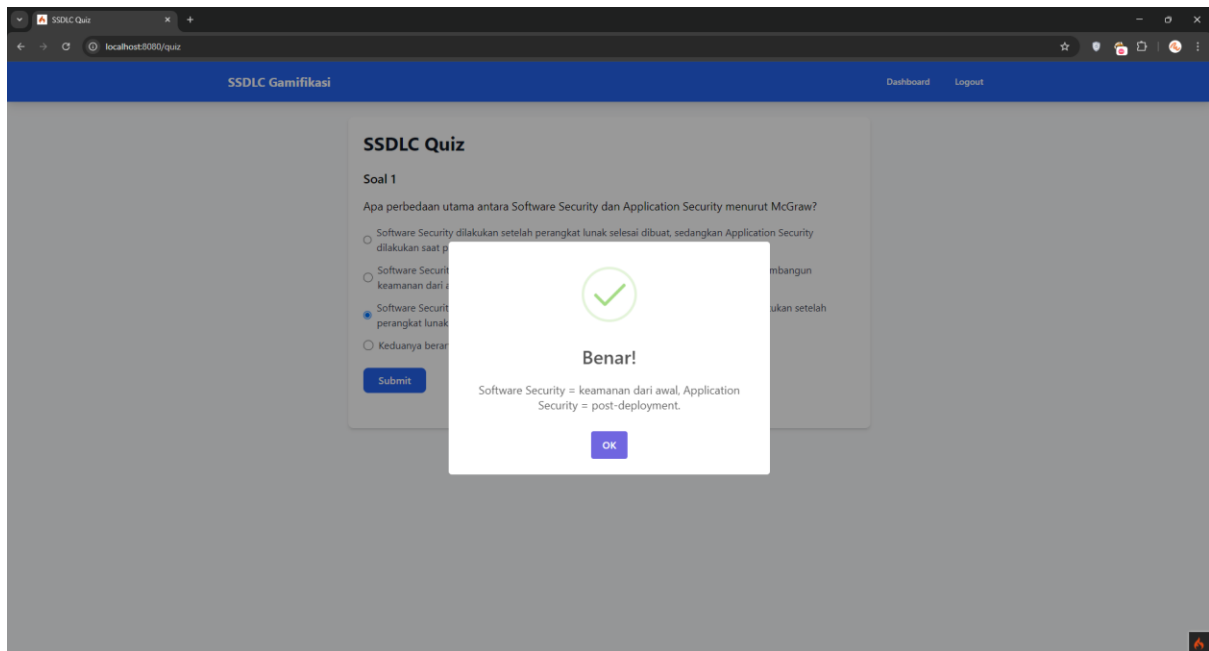
Aplikasi yang saya kembangkan berfokus pada edukasi interaktif mengenai keamanan dalam siklus hidup pengembangan perangkat lunak (Secure SDLC). Melalui pendekatan *gamifikasi*, pengguna diajak menjelajahi 2 level utama dan 1 level rahasia yang penuh tantangan.

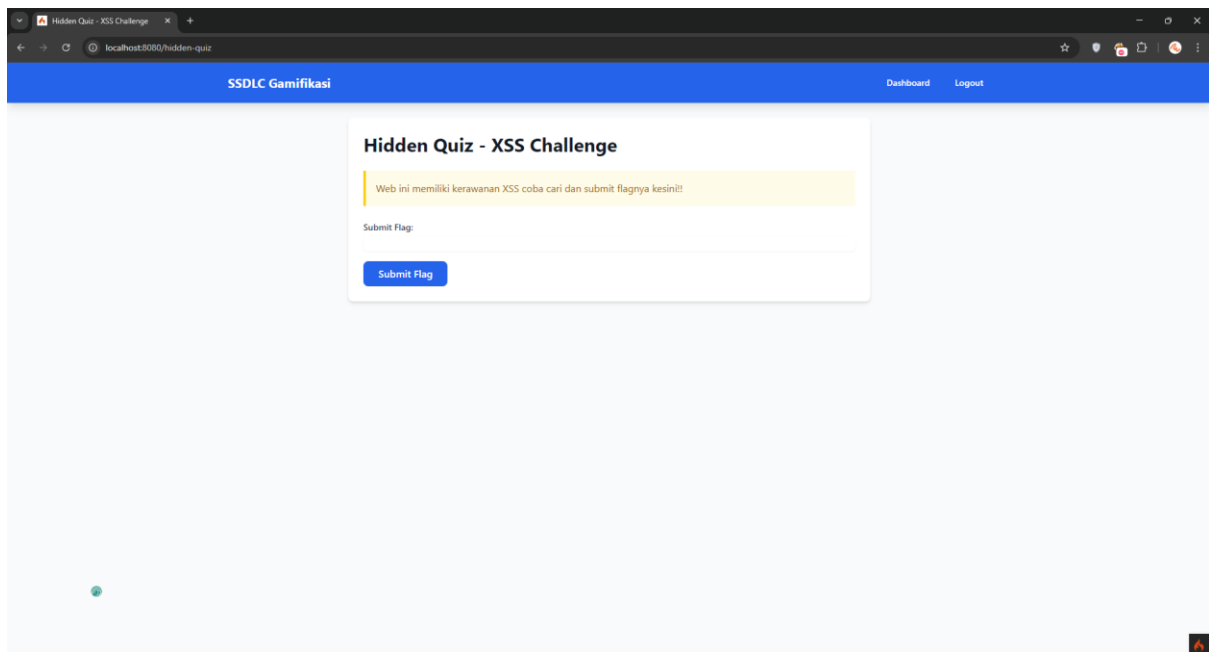
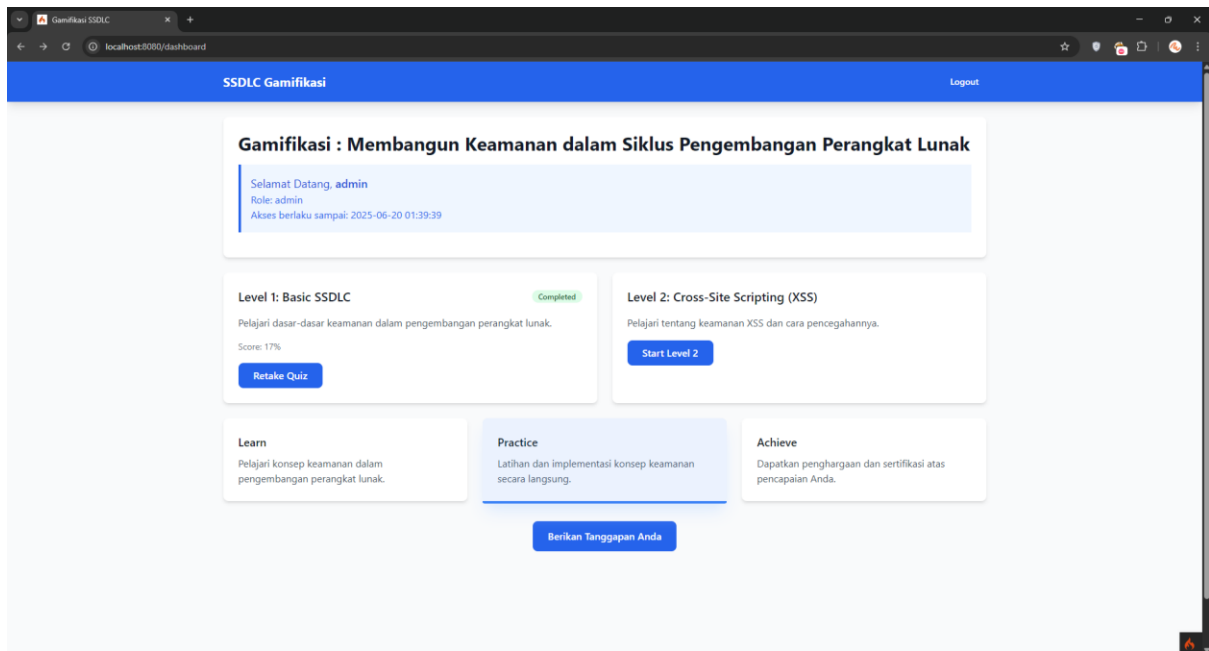
- Level 1 memperkenalkan konsep dasar keamanan dalam SDLC, memberikan pemahaman tentang pentingnya membangun keamanan sejak tahap awal pengembangan.
- Level 2 mengajak pengguna mengenal lebih dalam salah satu ancaman nyata dalam dunia aplikasi, yaitu Cross-Site Scripting (XSS), dengan skenario interaktif dan studi kasus sederhana.
- Level Tersembunyi menjadi tantangan spesial: pengguna diminta melakukan *pentest* sederhana untuk menemukan dan mengekstrak sebuah *flag* rahasia—uji nyali bagi mereka yang merasa cukup tangguh!

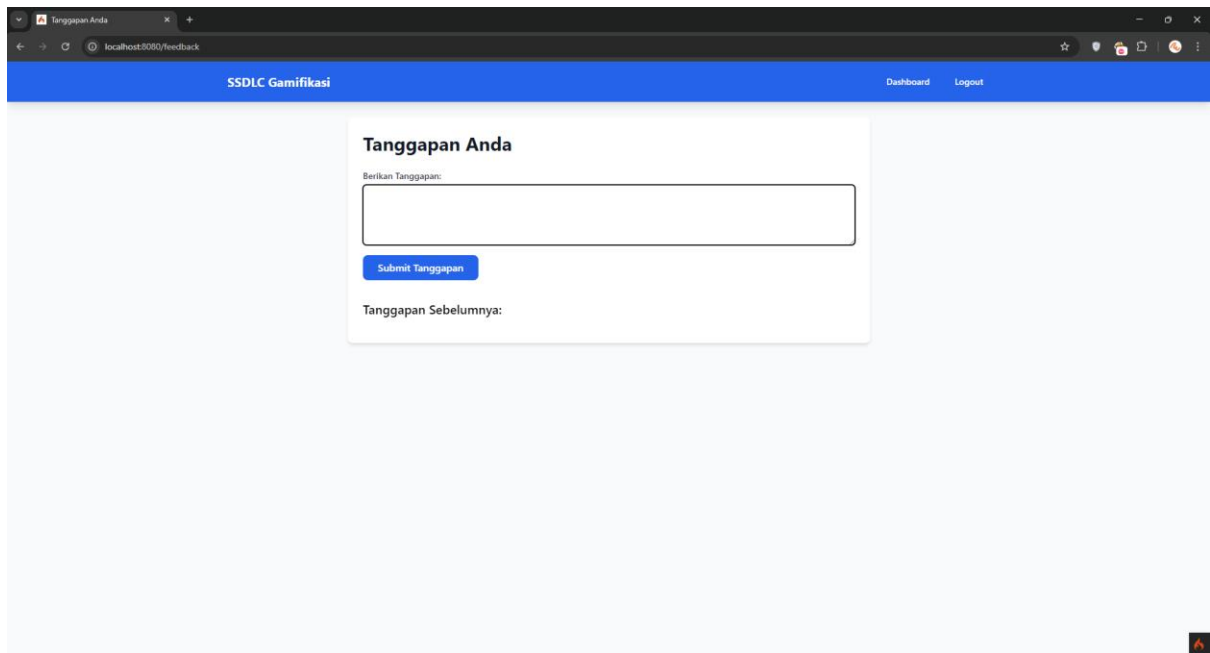
Dengan konsep ini, saya berharap pengguna tidak hanya memahami teori, tetapi juga dapat merasakan langsung pentingnya keamanan dalam setiap fase pengembangan perangkat lunak.

Dashboard :









Dari hasil dynamic analysis menggunakan ZAP ditemukan 1 kategori Medium warning dengan 1420 percobaan, dan 3 warning informal. Sebelumnya terdapat 1 kategori hard yaitu cloud metadata leak potential pada <http://localhost:8080/latest/meta-data/>. Tapi saya tangani dengan membatasi dengan `setAutoRoute(false)`;

```
Routes.php X Auth.php
app > Config > Routes.php
5  /**
8  $routes->get('/', 'Home::index');
9  $routes->get('/dashboard', 'Dashboard::index');
10 $routes->get('/quiz', 'Dashboard::quiz');
11 $routes->get('/quiz-level2', 'Dashboard::quizLevel2');
12 $routes->post('/complete-quiz', 'Dashboard::completeQuiz');
13 $routes->get('/hidden-quiz', 'Dashboard::hiddenQuiz');
14 $routes->get('/feedback', 'Dashboard::feedback');
15 $routes->get('login', 'Auth::login');
16 $routes->post('login', 'Auth::login');
17 $routes->get('logout', 'Auth::logout');
18
19 $routes->setAutoRoute(false);
```

```
// Set strict security headers (no wildcard * values)
$this->response->setHeader('X-Content-Type-Options', 'nosniff');
$this->response->setHeader('X-Frame-Options', 'SAMEORIGIN');
$this->response->setHeader('Referrer-Policy', 'strict-origin-when-cross-origin');
$this->response->setHeader('X-XSS-Protection', '1; mode=block');

// Content Security Policy without wildcards
$csp = "default-src 'self'; "
. "script-src 'self' https://cdn.tailwindcss.com https://cdn.jsdelivr.net https://code.jquery.com; "
. "style-src 'self' 'unsafe-inline' https://cdn.tailwindcss.com; "
. "img-src 'self' data:; "
. "font-src 'self' https://cdn.jsdelivr.net; "
. "connect-src 'self';";
$this->response->setHeader('Content-Security-Policy', $csp);
}
```

Setelah di patch hasil scan jadi seperti ini

The image displays two screenshots of the Burp Suite interface, showing the results of a security scan.

Top Screenshot: CSP: Wildcard Directive (1420)

- Alerts (5):** CSP: Wildcard Directive (1420)
- Parameter:** Content-Security-Policy
- Attack:** default-src 'none', script-src 'self', connect-src 'self', child-src 'self', img-src 'self' data, font-src 'self' data, style-src 'self'
- Evidence:** default-src 'none', script-src 'self', connect-src 'self', child-src 'self', img-src 'self' data, font-src 'self' data, style-src 'self'
- CWE ID:** 693
- WASC ID:** 15
- Source:** Passive (10055 - CSP)
- Alert Reference:** 10055-4
- Input Vector:** Content-Security-Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
- Other Info:** The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined: frame-ancestors, form-action
- Solution:** Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
- Reference:** <https://www.w3.org/TR/CSP/>, <https://canisuse.com/#search=content-security-policy>, <https://content-security-policy.com/>
- Alert Tags:**
- Key:** CSP: Wildcard Directive (1420)
- Value:** Content-Security-Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

Bottom Screenshot: Authentication Request Identified (2)

- Alerts (5):** Authentication Request Identified (2)
- Parameter:** username
- Attack:** password
- Evidence:** password
- CWE ID:** 287
- WASC ID:** 10
- Source:** Passive (10111 - Authentication Request Identified)
- Input Vector:** The given request has been identified as an authentication request. The 'Other info' field contains a set of key-value lines which identify any relevant fields. If the request is in a context which has an Authentication Method set to 'Auto-Detect' then this rule will change the authentication to match the request identified.
- Other Info:** useParam=username, useParam=password, passwordParam=password
- Solution:** This is an informational alert rather than a vulnerability and so there is nothing to fix.
- Reference:** <https://www.zaproxy.org/docs/desktop/addons/authentication-helper/auth-req-id/>
- Alert Tags:**
- Key:** Authentication Request Identified (2)
- Value:** The given request has been identified as an authentication request. The 'Other info' field contains a set of key-value lines which identify any relevant fields. If the request is in a context which has an Authentication Method set to 'Auto-Detect' then this rule will change the authentication to match the request identified.

1. CSP: Wildcard Directive (1420)

- URL: `http://localhost:8080/UI/ajaxSpider/`
- Risiko: Medium
- Confidence: High
- Masalah:
 - Header Content-Security-Policy (CSP) terlalu umum atau menggunakan wildcard
- Solusi:
 - Konfigurasi CSP secara lebih spesifik.
 - Hindari wildcard (*) dan gunakan domain yang terpercaya.

2. Authentication Request Identified (2)

- URL:
`http://localhost:8080/UI/network/action/setHttpProxy/override?apikey=ZAP&username=ZAP&password=ZAP`
- Risiko: Informational
- Confidence: Low
- Masalah:
 - ZAP mengidentifikasi request yang mengandung parameter username dan password.
 - Ini hanya peringatan untuk menunjukkan request autentikasi — bukan kerentanan langsung.
- Solusi:
 - Tidak perlu tindakan, kecuali jika dikombinasikan dengan kelemahan lain seperti credential leak.
 - Gunakan metode autentikasi yang aman dan hindari menyimpan kredensial di URL.

3. User Agent Fuzzer (12)

- URL: `http://localhost:8080/`
- Risiko: Informational
- Masalah:
 - ZAP melakukan pengujian dengan user-agent berbeda (misalnya, bot, crawler, atau browser lama).
 - Tidak ditemukan perbedaan perilaku yang signifikan dari server.
- Solusi:

- Tidak ada masalah besar yang perlu diperbaiki.
- Namun, pastikan tidak ada akses istimewa berdasarkan user-agent tanpa validasi tambahan.

4. User Controllable HTML Element Attribute (Potential XSS) (4259)

- URL:
http://localhost:8080/UI/alertView/alertCountsByRisk/...&url=https://zap.example.com
- Risiko: Informational
- Confidence: Low
- Masalah:
 - Parameter user-supplied (apikey, url, dll) bisa ditampilkan dalam atribut HTML.
 - Bisa berpotensi XSS jika tidak disanitasi dengan benar.
- Solusi:
 - Sanitasi dan validasi seluruh input dari user sebelum dimasukkan ke dalam HTML.
 - Gunakan library escaping seperti htmlspecialchars() di PHP atau encoding pada framework modern.