```
+--------------------------------------------------------+
|                Vulnerability Scan Report               |
+--------------------------------------------------------+
```

[1] Missing security header: Referrer-Policy

Risk Level: 1 (Low)

Vulnerability Details:

Evidence 1:

URL: https://respected-vehicles-comments-plots.trycloudflare.com/

Evidence: Response headers do not include the Referrer-Policy HTTP security header as well as the <meta> tag with name 'referrer' is not present in the response. Request / Response

Description: We noticed that the target application's server responses lack the <code>Referrer-Policy</code> HTTP header, which controls how much referrer information the browser will send with each request originated from the current web application.

Recommendation: The Referrer-Policy header should be configured on the server side to avoid user tracking and inadvertent information leakage. The value no-referrer of this header instructs the browser to omit the Referer header entirely.

SOLUSI:

Menambahkan Header Referrer-Policy, dengan isi middleware SecurityHeadersnya seperti berikut:

```
namespace App\Http\Middleware;


use Closure;
use Illuminate\Http\Request;


class SecurityHeaders
{
    public function handle(Request $request, Closure $next)
    {
        $response = $next($request);

        $response->headers->set('Referrer-Policy', 'no-referrer');

        return $response;
    }
}
```

Dan isi file bootstrap/app.php seperti berikut supaya middleware SecurityHeaders aktif:

```
->withMiddleware(function (Middleware $middleware) {
 $middleware->append(SecurityHeaders::class);
})
```

[2] Missing security header: Content-Security-Policy

- Risk Level: 1 (Low)

Vulnerability Details:

- Evidence 1:

  - URL: https://rf-boots-wars-throws.trycloudflare.com/

  - Evidence: Response does not include the HTTP Content-Security-Policy security header or meta tag Request / Response

- Description: We noticed that the target application lacks the Content-Security-Policy (CSP) header in its HTTP responses. The CSP header is a security measure that instructs web browsers to enforce specific security rules, effectively preventing the exploitation of Cross-Site Scripting (XSS) vulnerabilities.

- Recommendation: Configure the Content-Security-Header to be sent with each HTTP response in order to apply the specific policies needed by the application.

SOLUSI:

menambahkan header Content-Security-Policy, dengan isi middleware SecurityHeadersnya seperti berikut:

```
class SecurityHeaders
{
    public function handle(Request $request, Closure $next): Response
    {
        $response = $next($request);


        $response->headers->set('Referrer-Policy', 'no-referrer');
        $response->headers->set('Strict-Transport-Security', 'max-age=31536000; includeSubDomains; preload');
        $response->headers->set('X-Content-Type-Options', 'nosniff');


        $response->headers->set('Content-Security-Policy',
            "default-src 'self'; " .
            "script-src 'self'; " .
            "style-src 'self' 'unsafe-inline'; " .
            "img-src 'self' data:; " .
            "font-src 'self'; " .
            "object-src 'none'; " .
            "base-uri 'self'; "
        );


        return $response;
    }
}
```

Tetapi entah kenapa masih warning.

[3] Missing security header: Strict-Transport-Security

- Risk Level: 1 (Low)

Vulnerability Details:

- Evidence 1:

    - URL: https://rf-boots-wars-throws.trycloudflare.com/

    - Evidence: Response headers do not include the HTTP Strict-Transport-Security header Request / Response

- Description: We noticed that the target application lacks the HTTP Strict-Transport-Security header in its responses. This security header is crucial as it instructs browsers to only establish secure (HTTPS) connections with the web server and reject any HTTP connections.

- Recommendation: The Strict-Transport-Security HTTP header should be sent with each HTTPS response. The syntax is as follows: `Strict-Transport-Security: max-age=&lt;seconds>[; includeSubDomains]` The parameter `max-age` gives the time frame for requirement of HTTPS in seconds and should be chosen quite high, e.g. several months. A value below 7776000 is considered as too low by this scanner check. The flag `includeSubDomains` defines that the policy applies also for sub domains of the sender of the response.

SOLUSI:

menambahkan header **Strict-Transport-Security, dengan isi middleware SecurityHeadersnya seperti berikut:**

```
class SecurityHeaders

{

   public function handle(Request $request, Closure $next): Response

   {

       $response = $next($request);


       $response->headers->set('Referrer-Policy', 'no-referrer');

       $response->headers->set('Content-Security-Policy', "default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'; img-src 'self' data:; font-src 'self';");
```

        $response->headers->set('Strict-Transport-Security', 'max-age=31536000; includeSubDomains; preload');

```
        return $response;
    }
}
```

[4] Missing security header: X-Content-Type-Options

    - Risk Level: 1 (Low)

    Vulnerability Details:

    - Evidence 1:

        - URL: https://rf-boots-wars-throws.trycloudflare.com/

        - Evidence: Response headers do not include the X-Content-Type-Options HTTP security header Request / Response

    - Description: We noticed that the target application's server responses lack the <code>X-Content-Type-Options</code> header. This header is particularly important for preventing Internet Explorer from reinterpreting the content of a web page (MIME-sniffing) and thus overriding the value of the Content-Type header.

    - Recommendation: We recommend setting the X-Content-Type-Options header such as `X-Content-Type-Options: nosniff`.

SOLUSI:

menambahkan header Header **X-Content-Type-Options**, **, dengan isi middleware SecurityHeadersnya seperti berikut:**

```
class SecurityHeaders
{
    public function handle(Request $request, Closure $next): Response
    {
        $response = $next($request);


        $response->headers->set('Referrer-Policy', 'no-referrer');
        $response->headers->set('Content-Security-Policy', "default-src 'self'; script-src 'self'; style-src 'self'
'unsafe-inline'; img-src 'self' data:; font-src 'self';");
        $response->headers->set('Strict-Transport-Security', 'max-age=31536000; includeSubDomains;
preload');
        $response->headers->set('X-Content-Type-Options', 'nosniff');


        return $response;
    }
}
```

[5] Server software and technology found

- Risk Level: 1 (Low)

Vulnerability Details:

- Evidence 1:
    - Software / Version: Alpine.js
    - Category: JavaScript frameworks

- Evidence 2:
    - Software / Version: Bunny
    - Category: CDN

- Evidence 3:

- Software / Version: Bunny Fonts
- Category: Font scripts

- Evidence 4:
    - Software / Version: Livewire
    - Category: Web frameworks, Miscellaneous

- Evidence 5:
    - Software / Version: Laravel
    - Category: Web frameworks

- Evidence 6:
    - Software / Version: PHP
    - Category: Programming languages

- Evidence 7:
    - Software / Version: Cloudflare
    - Category: CDN

- Description: We noticed that server software and technology details are exposed, potentially aiding attackers in tailoring specific exploits against identified systems and versions.

- Recommendation: We recommend you to eliminate the information which permits the identification of software platform, technology, server and operating system: HTTP server headers, HTML meta information, etc.

SOLUSI:

[6] Robots.txt file found

- Risk Level: 1 (Low)

Vulnerability Details:

- Evidence 1:

  - URL: https://rf-boots-wars-throws.trycloudflare.com/robots.txt

- Description: We found the robots.txt on the target server. This file instructs web crawlers what URLs and endpoints of the web application they can visit and crawl. Website administrators often misuse this file while attempting to hide some web pages from the users.

- Recommendation: We recommend you to manually review the entries from robots.txt and remove the ones which lead to sensitive locations in the website (ex. administration panels, configuration files, etc).

SOLUSI:

Mengahapus file Robots.txt & membuat middleware RestrictRobotsTxt supaya hanya mengizinkan Googlebot, blokir yang lain via .htaccess atau middleware Laravel.

Dengan Isi Middleware RestrictRobotsTxt.php sebagai berikut:

```
class RestrictRobotsTxt
{
  public function handle(Request $request, Closure $next): Response
  {
    if ($request->is('robots.txt')) {
      $userAgent = $request->header('User-Agent');

      // Cek apakah user-agent mengandung Googlebot, Bingbot, dll
      if (
        !str_contains($userAgent, 'Googlebot') &&
        !str_contains($userAgent, 'Bingbot')
      ) {
        // Blokir akses
        return response('Forbidden', 403);
```

```
            }
        }

        return $next($request);
    }
}
```