

ASSIGNMENT 3



Ryan Hanif Dwihandoyo
Bootcamp CS Batch 3

Pendahuluan

Session hijacking atau pembajakan sesi adalah serangan keamanan di mana seorang penyerang mengambil alih sesi pengguna yang valid pada sebuah aplikasi web. Ketika pengguna melakukan login, server akan membuat sebuah ID sesi unik yang disimpan di browser pengguna dalam bentuk cookie. Penyerang berusaha untuk mencuri atau memanipulasi ID sesi ini untuk mendapatkan akses tidak sah ke akun pengguna tanpa memerlukan kredensial login.

Risiko dari serangan ini sangat signifikan, mulai dari pencurian data pribadi dan finansial, hingga pengambilalihan akun secara penuh yang memungkinkan penyerang bertindak sebagai pengguna sah. Oleh karena itu, memahami dan menerapkan teknik pencegahan session hijacking adalah elemen fundamental dalam menjaga keamanan aplikasi web.

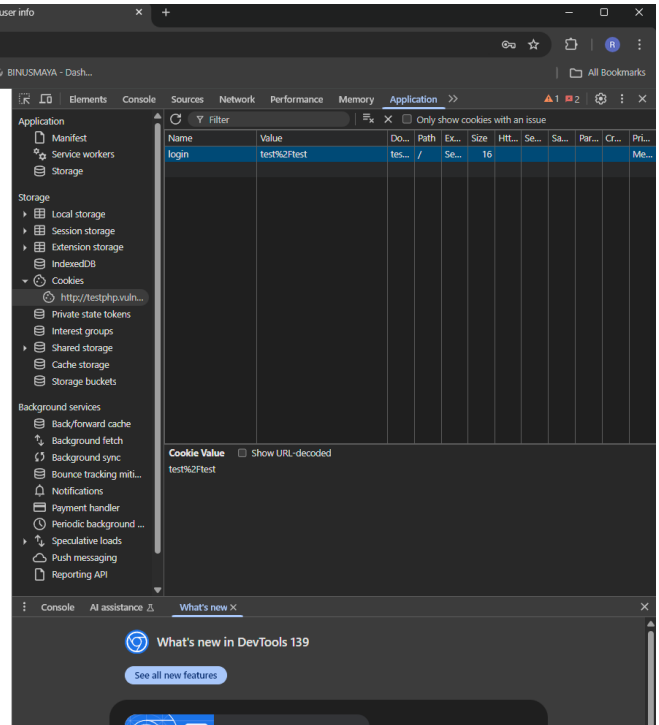
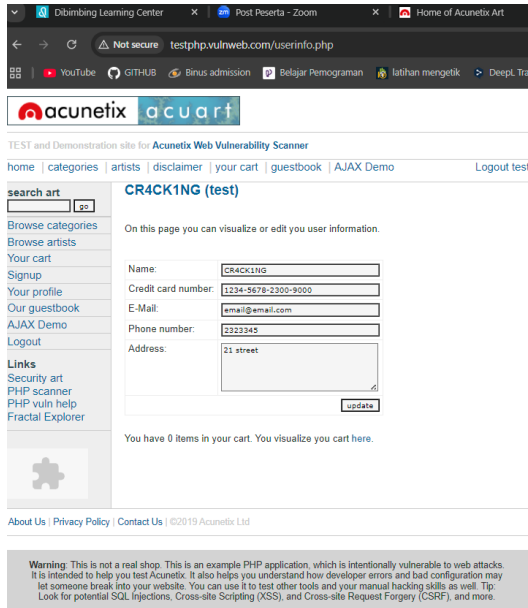
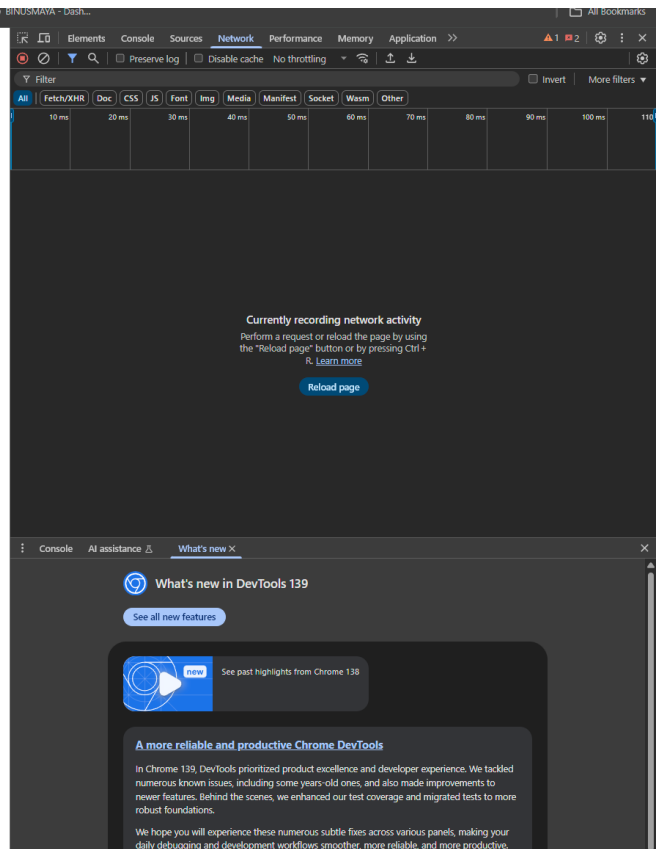
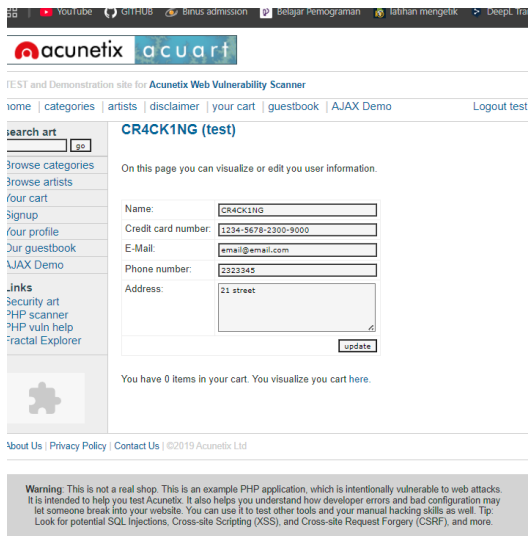
Analisis Kerentanan pada testphp.vulnweb.com

Situs testphp.vulnweb.com merupakan sebuah platform yang sengaja dibuat rentan untuk tujuan pembelajaran dan pengujian. Berdasarkan analisis, situs ini memiliki beberapa kerentanan kritis yang membuka celah untuk serangan session hijacking.

Transmisi ID Sesi Tanpa Enkripsi: Kerentanan paling mendasar adalah penggunaan protokol HTTP. Ini berarti semua komunikasi antara browser pengguna dan server, termasuk cookie yang berisi ID sesi, dikirim dalam bentuk teks biasa. Penyerang yang berada di jaringan yang sama (misalnya, jaringan WiFi publik) dapat dengan mudah menyadap lalu lintas ini menggunakan tools seperti Wireshark atau Bettercap untuk mencuri cookie sesi. Serangan ini dikenal sebagai **Man-in-the-Middle** (MITM) ketiadaan **Secure Cookie Flags**: Cookie sesi pada testphp.vulnweb.com tidak dikonfigurasi dengan atribut keamanan yang memadai:

- **Tanpa flag Secure:** Memungkinkan cookie untuk dikirim melalui koneksi HTTP yang tidak aman, membuatnya rentan terhadap penyadapan.
- **Tanpa flag HttpOnly:** Memungkinkan skrip sisi klien, seperti JavaScript, untuk mengakses cookie. Jika situs ini juga rentan terhadap **Cross-Site Scripting** (XSS), penyerang dapat menyuntikkan skrip untuk mencuri cookie sesi pengguna.
- **Tanpa flag SameSite:** Meningkatkan risiko serangan **Cross-Site Request Forgery** (CSRF), di mana penyerang dapat menipu browser pengguna untuk mengirimkan permintaan ke situs web beserta cookie sesinya.

Potensi ID Sesi yang Lemah: Banyak aplikasi web lama menggunakan algoritma pembuatan ID sesi yang dapat diprediksi. Jika seorang penyerang mampu menebak atau memprediksi ID sesi pengguna lain, mereka dapat dengan mudah membajak sesi tersebut.



Burp Suite Community Edition v2025.8.3 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Intercept HTTP history WebSockets history Match and replace Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length
42	https://www.google.com	POST	/gen_204?atyp=i&ei=QNNaEaODCU6a4-EP6sH20Q8&ct...	✓		204	694
41	http://testphp.vulnweb.com	GET	/			200	5180
40	https://www.google.com	POST	/gen_204?atyp=i&ei=QNNaEaODCU6a4-EP6sH20Q8&ct...	✓		204	694

Request

Pretty Raw Hex

```
17 Sec-Ch-Prefers-Color-Scheme: dark
18 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
  Safari/537.36
19 Rtt: 0
20 Sec-Ch-Ua-Platform-Version: ""
21 Accept: */*
22 Origin: https://www.google.com
23 X-Client-Data: CjV2ygE=
24 Sec-Fetch-Site: same-origin
25 Sec-Fetch-Mode: no-cors
26 Sec-Fetch-Dest: empty
27 Referer: https://www.google.com/
28 Accept-Encoding: gzip, deflate, br
29 Priority: u=4, i
30
31
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 204 No Content
2 Content-Type: text/html; charset=UTF-8
3 Content-Security-Policy: object-src 'none';base-uri
  'self';script-src 'nonce-SW0qJtcZ9vJLsxazJ_WZgg'
  'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline'
  https: http:report-uri https://csp.withgoogle.com/csp/gws/other
4 Cross-Origin-Opener-Policy: same-origin-allow-popups;
  report-to="gws"
5 Report-To:
  [{"group":"gws","max_age":2592000,"endpoints":[{"url":"https://cs
  p.withgoogle.com/csp/report-to/gws/other"}]}]
6 Permissions-Policy: unload=()
7 Date: Sat, 13 Sep 2025 02:40:33 GMT
8 Server: gws
9 Content-Length: 0
10 X-Xss-Protection: 0
11 X-Frame-Options: SAMEORIGIN
```

Event log (4) All issues

Memory: 218.2MB Disabled

Burp Suite Community Edition v2025.8.3 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Intercept HTTP history WebSockets history Match and replace Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length
---	------	--------	-----	--------	--------	-------------	--------

Request

Pretty Raw Hex

```
1 GET / HTTP/1.1
2 Host: dibimbing.id
3 Accept-Language: en-US,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
  Safari/537.36
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
  f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
  =b3;q=0.7
7 Sec-Ch-Ua: "Not=A?Brand";v="24", "Chromium";v="140"
8 Sec-Ch-Ua-Mobile: ?0
9 Sec-Ch-Ua-Platform: "Windows"
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Priority: u=0, i
16 Connection: keep-alive
17
18
```

Response

Pretty Raw Hex Render

```
9 Content-type: text/html; charset=utf-8
10 Date: Sat, 13 Sep 2025 02:37:15 GMT
11 Etag: "8f577faa573182e40ef7ff2da96537a0e"
12 Last-Modified: Fri, 12 Sep 2025 22:05:20 GMT
13 Server: Vercel
14 Strict-Transport-Security: max-age=63072000
15 X-Vercel-Cache: HIT
16 X-Vercel-Id: sin1::vz6m2-1757731035254-895075020d4d
17 Content-Length: 186714
18
19 <!DOCTYPE html><html lang="id">
  <head>
    <meta charset="UTF-8">
    <script type="text/javascript" async src="
      https://www.googletagmanager.com/gtag/js?id=G-4EMPR7MB94">
    </script>
    <script type="text/partytown">
      window.dataLayer = window.dataLayer || [];
      function gtag() {
        dataLayer.push(arguments);
      }
      gtag('js', new Date());
      gtag('config', 'G-4EMPR7MB94');
    </script><script type="text/javascript">
      (function (w, d, s, l, i) {

```

Event log (4) All issues

Memory: 224.1MB Disabled

Teknik Pencegahan

Untuk mengatasi kerentanan yang telah diidentifikasi, berikut adalah tiga metode pencegahan utama yang dapat diterapkan.

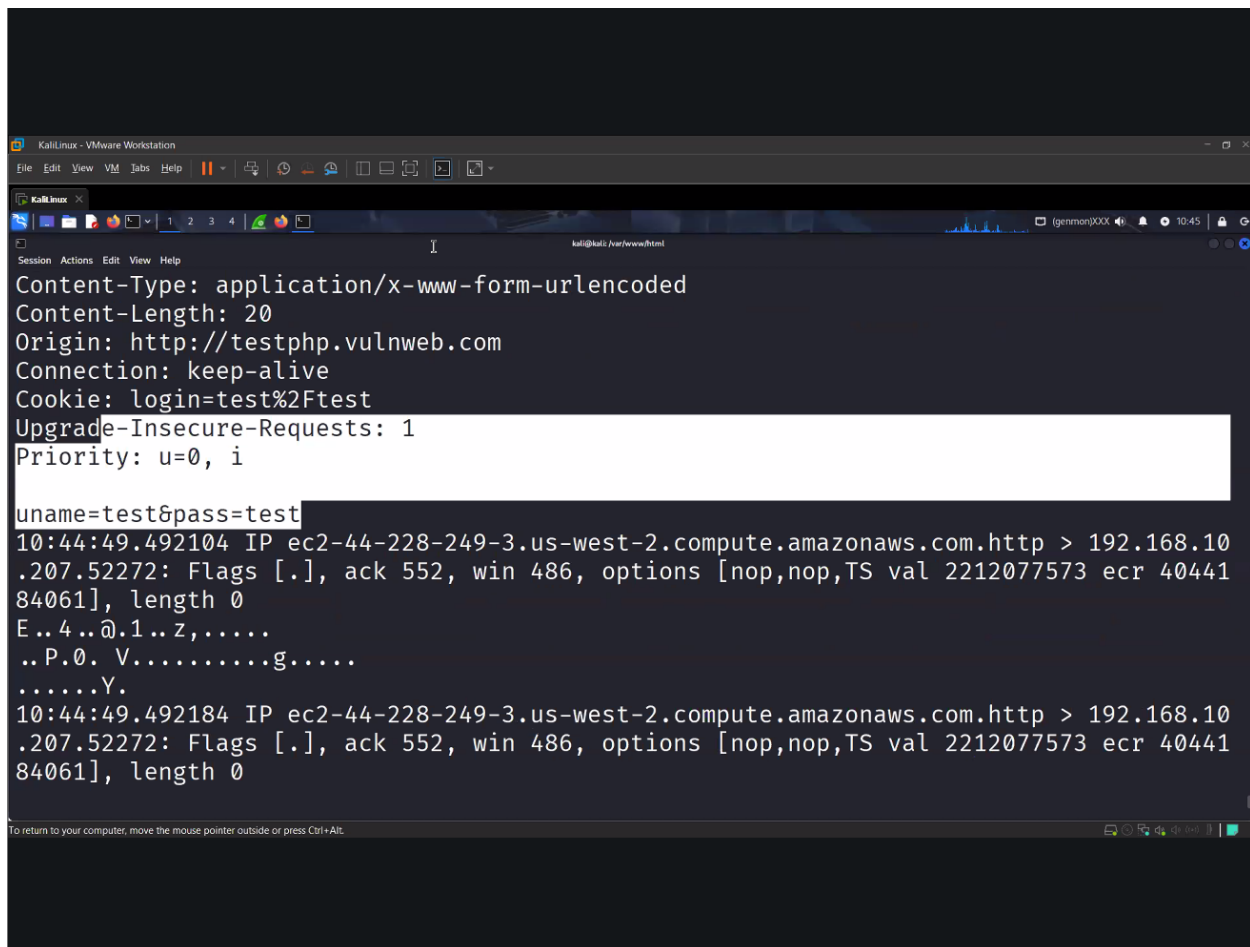
a. Implementasi HTTPS

HTTPS (***Hypertext Transfer Protocol Secure***) adalah versi aman dari HTTP yang menggunakan enkripsi SSL/TLS untuk melindungi data yang dikirimkan.

Cara Penerapan:

- Memperoleh sertifikat SSL/TLS dari Otoritas Sertifikat (CA) terpercaya.
- Menginstal dan mengkonfigurasi sertifikat tersebut pada server web.
- Mengkonfigurasi server untuk secara otomatis mengalihkan semua permintaan HTTP ke HTTPS.

Dengan ini, semua data, termasuk cookie sesi, akan terenkripsi dan aman dari penyadapan.



```
KaliLinux - VMware Workstation
File Edit View VM Tabs Help
Kali Linux
kali@kali: /usr/www/html
Session Actions Edit View Help
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Origin: http://testphp.vulnweb.com
Connection: keep-alive
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1
Priority: u=0, i
uname=test&pass=test
10:44:49.492104 IP ec2-44-228-249-3.us-west-2.compute.amazonaws.com.http > 192.168.10.207.52272: Flags [..], ack 552, win 486, options [nop,nop,TS val 2212077573 ecr 40441 84061], length 0
E..4..@.1..Z,.....
..P.0. V.....g.....
.....Y.
10:44:49.492184 IP ec2-44-228-249-3.us-west-2.compute.amazonaws.com.http > 192.168.10.207.52272: Flags [..], ack 552, win 486, options [nop,nop,TS val 2212077573 ecr 40441 84061], length 0
To return to your computer, move the mouse pointer outside or press Ctrl+Alt.
```

b. Penerapan Secure Cookie Flags

Mengatur atribut keamanan pada cookie adalah lapisan pertahanan penting untuk melindungi ID sesi.

Cara Penerapan: Atribut ini ditambahkan pada header respons Set-Cookie dari server.

- **Secure**: Menambahkan flag Secure akan memastikan cookie hanya dikirim melalui koneksi HTTPS. Contoh: Set-Cookie: PHPSESSID=...; Secure.

- **HttpOnly**: Flag HttpOnly akan mencegah JavaScript mengakses cookie, sehingga mitigasi risiko pencurian cookie melalui serangan XSS. Contoh:

- **Set-Cookie**: PHPSESSID=...; HttpOnly.

- **SameSite**: Atribut ini mengontrol pengiriman cookie pada permintaan lintas situs, efektif untuk mencegah serangan CSRF. Contoh: Set-Cookie: PHPSESSID=...; SameSite=Strict.

c. Implementasi HSTS (HTTP Strict Transport Security)

HSTS adalah mekanisme kebijakan keamanan yang memaksa browser untuk berkomunikasi dengan server hanya melalui koneksi HTTPS.

Cara Penerapan: **HSTS** diaktifkan dengan mengirimkan header respons Strict-Transport-Security dari server. Contoh: Strict-Transport-Security: max-age=31536000; includeSubDomains. Header ini akan memberitahu browser untuk "mengingat" bahwa situs tersebut hanya boleh diakses melalui **HTTPS** untuk periode waktu yang ditentukan (max-age).

4. Demonstrasi dan Pengujian Menggunakan Burp Suite

Berikut adalah simulasi sederhana untuk mendemonstrasikan kerentanan dan pengujian pencegahannya menggunakan Burp Suite di lingkungan lab terkontrol.

Langkah 1: Mendemonstrasikan Kerentanan

Konfigurasi browser untuk menggunakan Burp Suite sebagai proxy.

Aktifkan mode "Intercept is on" pada tab Proxy. Buka <http://testphp.vulnweb.com> di browser dan lakukan proses login. Burp Suite akan menangkap permintaan HTTP yang dikirimkan. Pada header permintaan, Anda dapat melihat dengan jelas cookie sesi (**contoh: Cookie: PHPSESSID=abcdef123456**) dikirim sebagai teks biasa. Ini membuktikan kerentanan terhadap penyadapan.

Langkah 2: Menguji Metode Pencegahan

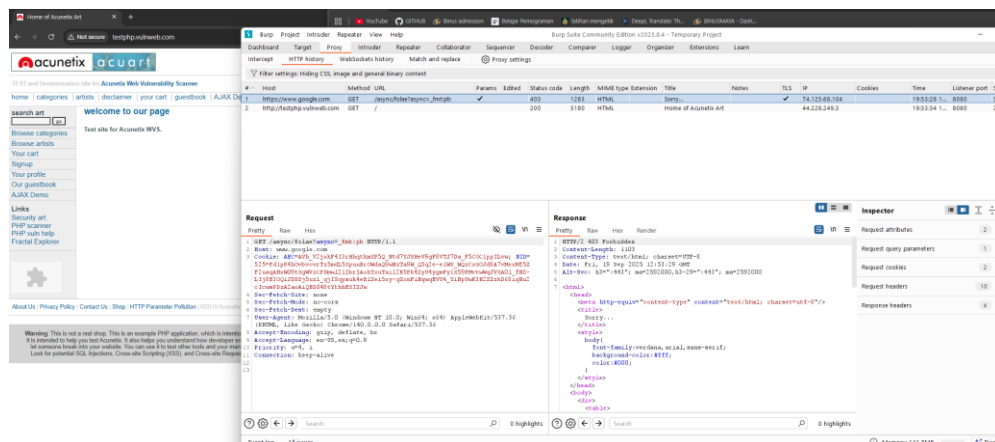
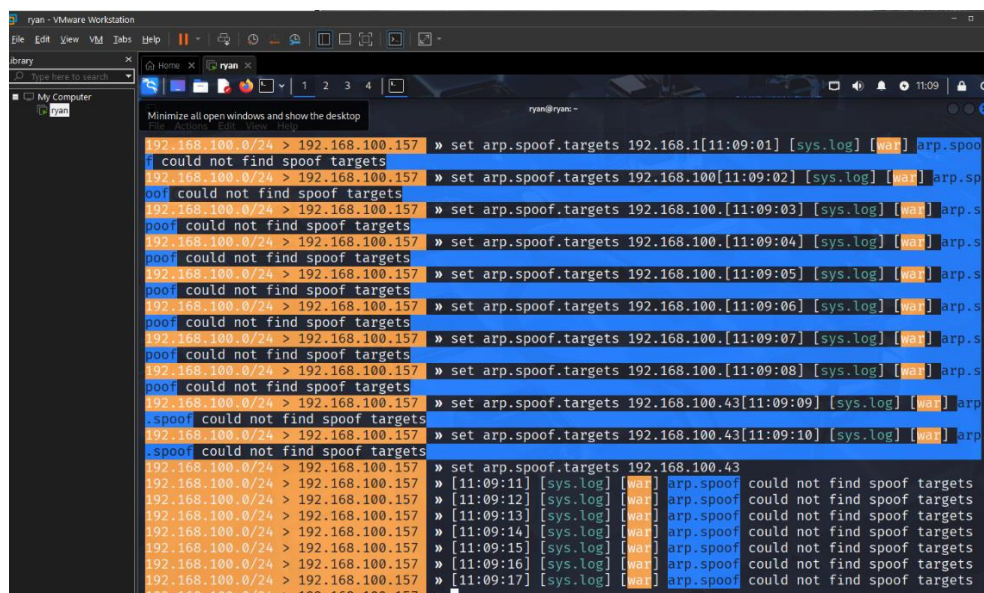
Meskipun kita tidak bisa mengubah konfigurasi server testphp.vulnweb.com, kita dapat mensimulasikan bagaimana pencegahan akan terlihat.

- **Pengujian HTTPS:** Jika situs telah menerapkan HTTPS, data yang ditangkap di Burp Suite akan terenkripsi dan tidak dapat dibaca secara langsung.

- **Pengujian HttpOnly:** Jika situs rentan terhadap XSS, kita bisa mencoba menyuntikkan payload seperti `<script>alert(document.cookie)</script>`.

- **Tanpa HttpOnly:** Payload ini akan berhasil dan menampilkan cookie sesi dalam sebuah kotak dialog.

- **Dengan HttpOnly:** Payload yang sama akan gagal mengakses cookie, dan kotak dialog akan muncul kosong, membuktikan efektivitas HttpOnly dalam mencegah pencurian cookie melalui XSS.



Evaluasi Efektivitas

Evaluasi terhadap berbagai teknik pencegahan menunjukkan bahwa tidak ada satu solusi tunggal yang dapat mengatasi semua ancaman session hijacking. Sebaliknya, efektivitas tertinggi dicapai melalui pendekatan keamanan berlapis yang menggabungkan beberapa metode.

Implementasi HTTPS adalah fondasi utama yang sangat efektif dalam melindungi data dari serangan penyadapan jaringan, seperti Man-in-the-Middle (MITM). Dengan mengenkripsi seluruh sesi komunikasi, HTTPS memastikan kerahasiaan ID sesi selama transit. Namun, kekuatannya terbatas pada perlindungan data saat bergerak di jaringan dan tidak memberikan perlindungan terhadap serangan pada level aplikasi, seperti Cross-Site Scripting (XSS) atau Cross-Site Request Forgery (CSRF).

Penerapan Secure Cookie Flags memberikan lapisan pertahanan krusial pada level aplikasi. Flag HttpOnly sangat efektif untuk memitigasi risiko pencurian cookie melalui serangan XSS dengan mencegah akses ke cookie dari skrip sisi klien. Sementara itu, flag Secure memastikan cookie hanya dikirim melalui koneksi yang terenkripsi, dan flag SameSite menjadi pertahanan yang kuat terhadap serangan CSRF. Kelemahan utamanya adalah ketergantungan pada implementasi HTTPS yang benar agar flag Secure dapat berfungsi secara optimal.

HTTP Strict Transport Security (HSTS) berfungsi sebagai penguat kebijakan keamanan dengan memaksa browser untuk selalu menggunakan koneksi HTTPS. Ini secara efektif mencegah serangan downgrade protocol, di mana penyerang mencoba memaksa browser kembali ke koneksi HTTP yang tidak aman untuk menyadap data. Keterbatasannya adalah HSTS tidak efektif pada kunjungan pertama pengguna, karena browser perlu menerima header HSTS dari server setidaknya sekali sebelum kebijakan tersebut diberlakukan.

Secara keseluruhan, kombinasi dari ketiga teknik ini menciptakan sistem pertahanan yang solid. HTTPS melindungi data di jaringan, secure cookie flags mengamankan cookie di level browser, dan HSTS memastikan kebijakan koneksi aman selalu ditegakkan setelah kunjungan pertama.

6. Rekomendasi Mitigasi

Berdasarkan analisis yang telah dilakukan, berikut adalah rekomendasi strategi mitigasi yang spesifik dan dapat ditindaklanjuti. Prioritaskan Migrasi ke HTTPS: Langkah paling krusial adalah segera beralih ke HTTPS untuk mengenkripsi seluruh komunikasi.

Terapkan Secure Cookie Flags secara Menyeluruh: Konfigurasi server untuk selalu mengirimkan flag Secure, HttpOnly, dan SameSite (Lax atau Strict) untuk semua cookie sesi. Aktifkan HSTS: Setelah HTTPS berjalan stabil, terapkan header HSTS untuk memastikan koneksi yang aman secara konsisten. Regenerasi ID Sesi: Selalu buat ID

sesi baru setiap kali tingkat otentikasi pengguna berubah (misalnya, setelah login berhasil) untuk mencegah serangan session fixation.

Implementasikan Timeout Sesi: Terapkan timeout sesi yang wajar (misalnya, 15-30 menit tidak aktif) untuk membatasi rentang waktu bagi penyerang untuk mengeksploitasi sesi yang dicuri.

7. Kesimpulan

Session hijacking adalah ancaman nyata yang dapat diatasi dengan praktik keamanan web yang baik. Situs testphp.vulnweb.com menjadi contoh nyata bagaimana ketiadaan lapisan keamanan dasar seperti HTTPS dan secure cookie flags dapat membahayakan pengguna. Dengan menerapkan pendekatan keamanan berlapis yang mencakup enkripsi HTTPS, pengamanan cookie yang ketat, dan penegakan kebijakan HSTS, risiko pembajakan sesi dapat dimitigasi secara efektif, sehingga menciptakan lingkungan yang lebih aman bagi pengguna aplikasi web.