**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN MENGENAI PROBLEM SERANGAN SESSION FIXATION DENGAN PENYELESAIAN MENGGUNAKAN SOLUSI REGENERASI SESSION ID SETELAH LOGIN**

*Laporan ini disusun untuk memenuhi tugas 1 mata kuliah Pengembangan Web (Teori)*



Disusun oleh kelompok B4:

Asri Husnul Rosadi 221524035

Faris Abulkhoir 221524040

Mahardika Pratama 221524044

Muhamad Fahri Yuwan 221524047

Najib Alimudin Fajri 221524053

**Sarah 221524059**

Septyana Agustina 221524060

Dosen Pengampu:

Joe Lian Min, M.Eng.

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2024**

# **DAFTAR ISI**

[DAFTAR ISI i](#_Toc175628822)

[A. IDENTIFIKASI PROBLEM 1](#_Toc175628823)

[B. DESKRIPSI PROBLEM 1](#_Toc175628824)

[C. METODOLOGI EKSPERIMEN 1](#_Toc175628825)

[D. PELAKSANAAN EKSPERIMEN 3](#_Toc175628826)

[E. ANALISIS HASIL EKSPERIMEN 5](#_Toc175628827)

# **IDENTIFIKASI PROBLEM**

Dalam pengembangan aplikasi web, keamanan autentikasi pengguna adalah aspek yang sangat kritis. Salah satu ancaman serius yang sering diabaikan adalah serangan session fixation, di mana penyerang dapat memaksa pengguna untuk menggunakan session ID yang sudah diketahui atau ditetapkan oleh penyerang. Jika serangan ini berhasil, penyerang dapat memperoleh akses ilegal ke akun pengguna dan data sensitif, tanpa memerlukan kredensial pengguna. Eksperimen ini bertujuan untuk menguji efektivitas regenerasi session ID setelah login sebagai solusi untuk mencegah serangan session fixation, dengan fokus pada aplikasi web yang dibangun menggunakan Node.js dan Express.

# **DESKRIPSI PROBLEM**

Session fixation adalah jenis serangan di mana penyerang menetapkan atau mengetahui session ID tertentu sebelum korban melakukan autentikasi. Ketika korban melakukan login, mereka secara tidak sadar menggunakan session ID yang telah diketahui atau ditetapkan oleh penyerang. Tanpa mekanisme keamanan seperti regenerasi session ID setelah login, penyerang dapat menggunakan session ID tersebut untuk mengakses sesi korban secara tidak sah.

# **METODOLOGI EKSPERIMEN**

1. **Desain Eksperimen**

Eksperimen ini dirancang untuk mengevaluasi efektivitas regenerasi session ID dalam mencegah serangan session fixation. Eksperimen dilakukan dengan membandingkan dua versi aplikasi web: satu yang menerapkan regenerasi session ID setelah login, dan satu lagi yang tidak. Perbandingan ini akan membantu menilai sejauh mana regenerasi session ID dapat mengurangi risiko penyerangan.

1. **Variabel Eksperimen**

* Variabel Independen:
* Penerapan regenerasi session ID setelah login.
* Tidak adanya regenerasi session ID setelah login.
* Variabel Dependen:
* Kemampuan penyerang untuk mengambil alih sesi pengguna (diukur dengan akses ke halaman dashboard setelah serangan).
* Keamanan sesi (dinilai dari keberhasilan atau kegagalan penyerang dalam mengakses sesi pengguna).

1. **Prosedur Eksperimen**
2. Persiapan Lingkungan:

* Siapkan dua server Node.js yang menjalankan aplikasi web yang sama, namun dengan konfigurasi berbeda untuk sesi (dengan dan tanpa regenerasi session ID).
* Siapkan server palsu untuk menetapkan session ID yang telah diketahui (fixed-session-id).

1. Langkah-langkah Eksperimen:

* Langkah 1: Penyerang mengarahkan korban ke halaman palsu yang menetapkan session ID (fixed-session-id).
* Langkah 2: Korban diarahkan ke aplikasi web asli dan melakukan login.
* Langkah 3: Penyerang mencoba mengakses halaman dashboard di aplikasi asli menggunakan session ID yang telah ditetapkan sebelumnya.

1. Pengukuran dan Pengamatan:

* Pantau apakah penyerang dapat mengakses halaman dashboard menggunakan session ID yang ditetapkan oleh server palsu.
* Catat perbedaan antara aplikasi yang menerapkan regenerasi session ID dan yang tidak.
* Gunakan alat debugging seperti Developer Tools pada browser untuk memantau cookie dan sesi yang aktif.

1. **Alat dan Bahan**
2. Software:

* Node.js dan Express untuk menjalankan server aplikasi web.
* Curl untuk melakukan serangan session fixation dari terminal.
* Browser dengan Developer Tools untuk memeriksa cookie dan sesi.

1. Hardware:

* Komputer atau server lokal untuk menjalankan aplikasi.
* Koneksi jaringan lokal atau virtual untuk menguji serangan dalam lingkungan terkontrol.

1. **Pengukuran dan Pengamatan**
2. Pengukuran Utama:

* Keberhasilan penyerang dalam mengakses halaman dashboard tanpa harus melakukan login ulang.
* Log server untuk mencatat aktivitas sesi, termasuk session ID yang digunakan.

1. Pengamatan:

* Amati apakah sesi yang diperoleh penyerang memungkinkan akses penuh ke halaman dashboard.
* Periksa apakah regenerasi session ID berhasil mencegah penyerang menggunakan sesi yang sama

1. **Analisis Data**

Data yang dikumpulkan akan dianalisis untuk mengevaluasi dampak penerapan regenerasi session ID terhadap keamanan sesi pengguna. Analisis ini akan mencakup perbandingan antara tingkat keberhasilan serangan session fixation sebelum dan sesudah penerapan regenerasi session ID. Selain itu, penilaian terhadap peningkatan keamanan yang dihasilkan oleh regenerasi session ID akan dilakukan dengan memperhatikan apakah penyerang dapat atau tidak dapat mengakses halaman dashboard setelah login korban.

# **PELAKSANAAN EKSPERIMEN**

1. Membuat Aplikasi Web Sederhana:

* Aplikasi dibuat menggunakan Node.js dan Express dengan halaman login dan dashboard sederhana.
* Dummy data pengguna dibuat untuk simulasi login.

1. Konfigurasi Aplikasi dengan dan Tanpa Regenerasi Session ID:

* Dua versi aplikasi dikonfigurasi:
* Versi 1: Regenerasi session ID setelah login.
* Versi 2: Tidak ada regenerasi session ID setelah login.

1. Membangun Server Palsu untuk Menetapkan Session ID:

* Server palsu dibuat untuk menetapkan session ID yang sudah diketahui (fixed-session-id).

1. Pelaksanaan Serangan:

* Penyerang mengarahkan korban ke server palsu yang menetapkan fixed-session-id.
* Korban kemudian diarahkan ke server asli untuk login.
* Penyerang mencoba mengakses sesi korban dengan menggunakan session ID yang telah diketahui (fixed-session-id).

1. Pengujian dengan curl:

* Penyerang menggunakan curl untuk mencoba mengakses halaman dashboard dengan session ID yang diketahui.

Kode untuk Server dengan Regenerasi Session ID:

|  |
| --- |
| // Proses login  app.post('/login', (req, res) => {  const { username, password } = req.body;  const user = users.find(u => u.username === username && u.password === password);  if (!user) {  return res.status(401).send('Invalid credentials');  }  // Regenerasi session ID setelah login untuk mencegah session fixation  req.session.regenerate((err) => {  if (err) {  console.error('Session regeneration failed:', err);  return res.status(500).send('Internal Server Error');  }  // Simpan informasi pengguna di sesi  req.session.user = { id: user.id, username: user.username };  res.redirect('/dashboard');  });  }); |

Kode untuk Server tanpa Regenerasi Session ID:

|  |
| --- |
| // Proses login  app.post('/login', (req, res) => {  const { username, password } = req.body;  const user = users.find(u => u.username === username && u.password === password);  if (!user) {  return res.status(401).send('Invalid credentials');  }  // Tidak ada regenerasi session ID  req.session.user = { id: user.id, username: user.username };  res.redirect('/dashboard');  }); |

# **ANALISIS HASIL EKSPERIMEN**

1. **Hasil Pengamatan**

* Aplikasi dengan Regenerasi Session ID: Ketika session ID diganti setelah login, penyerang tidak dapat menggunakan session ID yang sudah diketahui (fixed-session-id) untuk mengakses halaman dashboard. Sesi yang dimiliki korban tidak dapat diambil alih oleh penyerang.
* Aplikasi tanpa Regenerasi Session ID: Ketika session ID tidak diganti setelah login, penyerang seharusnya berhasil menggunakan session ID yang sudah diketahui (fixed-session-id) untuk mengakses halaman dashboard dan dapat mengambil alih sesi korban. Akan tetapi, ada permasalahan dalam proses curl sehingga hasil tidak sesuai harapan meskipun penyerang sudah mencoba masuk melalui session yang sama.

1. **Analisis Pros dan Cons:**

**Pros:**

1. Mencegah serangan session fixation dengan memastikan bahwa session ID yang digunakan setelah login adalah unik dan tidak dapat diprediksi oleh penyerang.
2. Meningkatkan keamanan secara signifikan pada aplikasi web.

**Cons:**

1. Memerlukan overhead tambahan di server karena perlu regenerasi session ID.
2. Potensi masalah kompatibilitas jika aplikasi lain atau klien menggunakan session ID yang bergantung pada sesi tertentu.
3. **Kesimpulan dan Rekomendasi**

Eksperimen ini dirancang untuk menguji efektivitas regenerasi session ID dalam mencegah serangan session fixation. Hasilnya menunjukkan bahwa pada aplikasi tanpa regenerasi session ID, penyerang diharapkan dapat menggunakan session ID yang sudah diketahui untuk mengambil alih sesi korban dan mengakses halaman dashboard. Namun, karena terdapat kendala teknis dalam proses curl, penyerang tidak berhasil mengakses sesi tersebut meskipun telah menggunakan session ID yang sama. Ini menunjukkan bahwa meskipun regenerasi session ID adalah praktik yang sangat dianjurkan untuk meningkatkan keamanan, dalam beberapa kasus, faktor lain seperti konfigurasi server atau alat yang digunakan oleh penyerang juga dapat mempengaruhi hasil. Oleh karena itu, direkomendasikan untuk tetap menerapkan regenerasi session ID setelah login dan melakukan pengujian lebih lanjut untuk memastikan bahwa mekanisme keamanan berfungsi dengan baik di berbagai kondisi.