**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN *ENSURING API SECURITY WITH HMAC (HASH-BASED MESSAGE AUTHENTICATION CODE)***

*Laporan ini disusun untuk memenuhi tugas 2 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

Septyana Agustina 221524058

**Sarah 221524059**

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](#_Toc175471873)

[A. IDENTIFIKASI PROBLEM 1](#_Toc175471874)

[B. DESKRIPSI PROBLEM 1](#_Toc175471875)

[C. METODOLOGI EKSPERIMEN 1](#_Toc175471876)

[D. PELAKSANAAN EKSPERIMEN 2](#_Toc175471877)

[E. ANALISIS HASIL EKSPERIMEN 2](#_Toc175471878)

# **IDENTIFIKASI PROBLEM**

Masalah yang diidentifikasi adalah bagaimana memastikan integritas dan keaslian pesan dalam komunikasi API, serta bagaimana mendeteksi dan mencegah replay attack dan modifikasi pesan yang mungkin dilakukan oleh pihak ketiga (MITM).

# **DESKRIPSI PROBLEM**

Dalam komunikasi API, memastikan bahwa data yang dikirim dan diterima tidak dimodifikasi oleh pihak ketiga dan berasal dari sumber yang sah merupakan tantangan penting. HMAC (Hash-based Message Authentication Code) adalah metode yang digunakan untuk menangani masalah ini dengan memberikan cara untuk memverifikasi integritas dan keaslian pesan. Eksperimen ini bertujuan untuk menguji efektivitas HMAC dalam melindungi komunikasi API, termasuk kemampuannya dalam mendeteksi replay attack dan modifikasi pesan yang dilakukan oleh MITM.

# **METODOLOGI EKSPERIMEN**

1. **Desain Eksperimen**

Eksperimen dilakukan dengan menggunakan HMAC untuk memastikan keamanan pesan API. Komunikasi API diuji dengan dan tanpa HMAC untuk mengevaluasi efektivitasnya dalam melindungi pesan dan mendeteksi serangan.

1. **Variabel Eksperimen**

* Variabel Independen: Implementasi HMAC dan keberadaan MITM.
* Variabel Dependen: Integritas pesan, deteksi replay attack, dan penanganan modifikasi pesan oleh MITM.

1. **Prosedur Eksperimen**
2. Implementasi Sistem: Setup server dan klien dengan HMAC untuk verifikasi pesan.
3. Pengujian Tanpa Modifikasi: Uji komunikasi antara klien dan server dengan HMAC tanpa adanya modifikasi pesan.
4. Pengujian Replay Attack: Simulasikan replay attack dengan mengirim ulang pesan yang sama.
5. Pengujian Modifikasi Pesan: Gunakan MITM untuk memodifikasi pesan dan uji kemampuan sistem dalam mendeteksi perubahan tersebut.
6. **Alat dan Bahan**

* Node.js untuk server dan klien
* Library crypto untuk HMAC
* Library axios untuk pengiriman HTTP requests
* MITM Proxy (seperti mitmproxy) untuk simulasi modifikasi pesan

1. **Pengukuran dan Pengamatan**

* Keberhasilan verifikasi HMAC (apakah pesan valid atau tidak)
* Deteksi replay attack (apakah nonce sudah digunakan sebelumnya)
* Respons server terhadap pesan yang dimodifikasi oleh MITM

1. **Analisis Data**

* Bandingkan hasil antara sistem dengan HMAC dan tanpa HMAC
* Evaluasi efektivitas dalam mendeteksi replay attack dan modifikasi pesan
* Identifikasi kesalahan dan respons sistem dalam setiap kasus.

# **PELAKSANAAN EKSPERIMEN**

1. Instalasi dan Konfigurasi Server dan Klien, buatlah tiga file utama: server.js, client.js, dan mitm.js.
2. Instalasi axios untuk permintaan HTTP dan crypto untuk HMAC.
3. Pengujian Tanpa Modifikasi: Kirimkan pesan dari klien ke server menggunakan HMAC dan verifikasi integritas pesan.
4. Pengujian Replay Attack: Kirim ulang pesan yang sama dari klien dan amati apakah server mendeteksi dan menolak replay attack.
5. Pengujian Modifikasi Pesan: Gunakan MITM untuk mengubah pesan yang dikirim dan verifikasi apakah server mendeteksi perubahan dan menolak pesan.

# **ANALISIS HASIL EKSPERIMEN**

1. **Hasil Pengamatan**

|  |  |  |
| --- | --- | --- |
| Pengujian | Hasil yang Diharapkan | Hasil yang Diperoleh |
| Tanpa Modifikasi | Pesan diterima dengan valid HMAC | Pesan diterima, HMAC valid |
| Replay Attack | Server menolak pesan duplikat dengan error | Pesan kedua ditolak dengan error replay attack |
| Modifikasi Pesan (MITM) | Server menolak pesan dengan HMAC tidak valid | Pesan dimodifikasi ditolak dengan error HMAC tidak valid |

1. **Analisis Pros and Cons**

**Pros**

* HMAC secara efektif memastikan integritas dan keaslian pesan.
* Dapat mendeteksi dan mencegah replay attack jika nonce dikelola dengan baik.
* Relatif mudah diimplementasikan dengan dukungan pustaka yang luas.

**Cons**

* Memerlukan manajemen secret key yang aman.
* Sedikit menambah latensi dalam komunikasi API.
* Implementasi nonce yang buruk dapat mengurangi efektivitas deteksi replay attack.

1. **Kesimpulan dan Saran**

Hasil eksperimen menunjukkan bahwa penggunaan HMAC efektif dalam memastikan integritas dan keaslian pesan, serta mampu mendeteksi replay attack dan modifikasi pesan oleh pihak ketiga. HMAC sangat cocok diterapkan pada aplikasi yang memerlukan keamanan tinggi, seperti API finansial atau komunikasi data sensitif, di mana integritas dan autentikasi pesan krusial. Namun, penerapan HMAC dapat menambah latensi karena proses perhitungan HMAC pada setiap pesan. Untuk aplikasi dengan kebutuhan responsivitas tinggi, ini bisa menjadi kendala, sehingga diperlukan pertimbangan antara tingkat keamanan yang dibutuhkan dan performa aplikasi. Saran terbaik adalah menggunakan HMAC di lingkungan di mana keamanan lebih diutamakan daripada sedikit tambahan latensi, dan memastikan bahwa kunci rahasia dijaga dengan baik untuk menghindari potensi serangan.