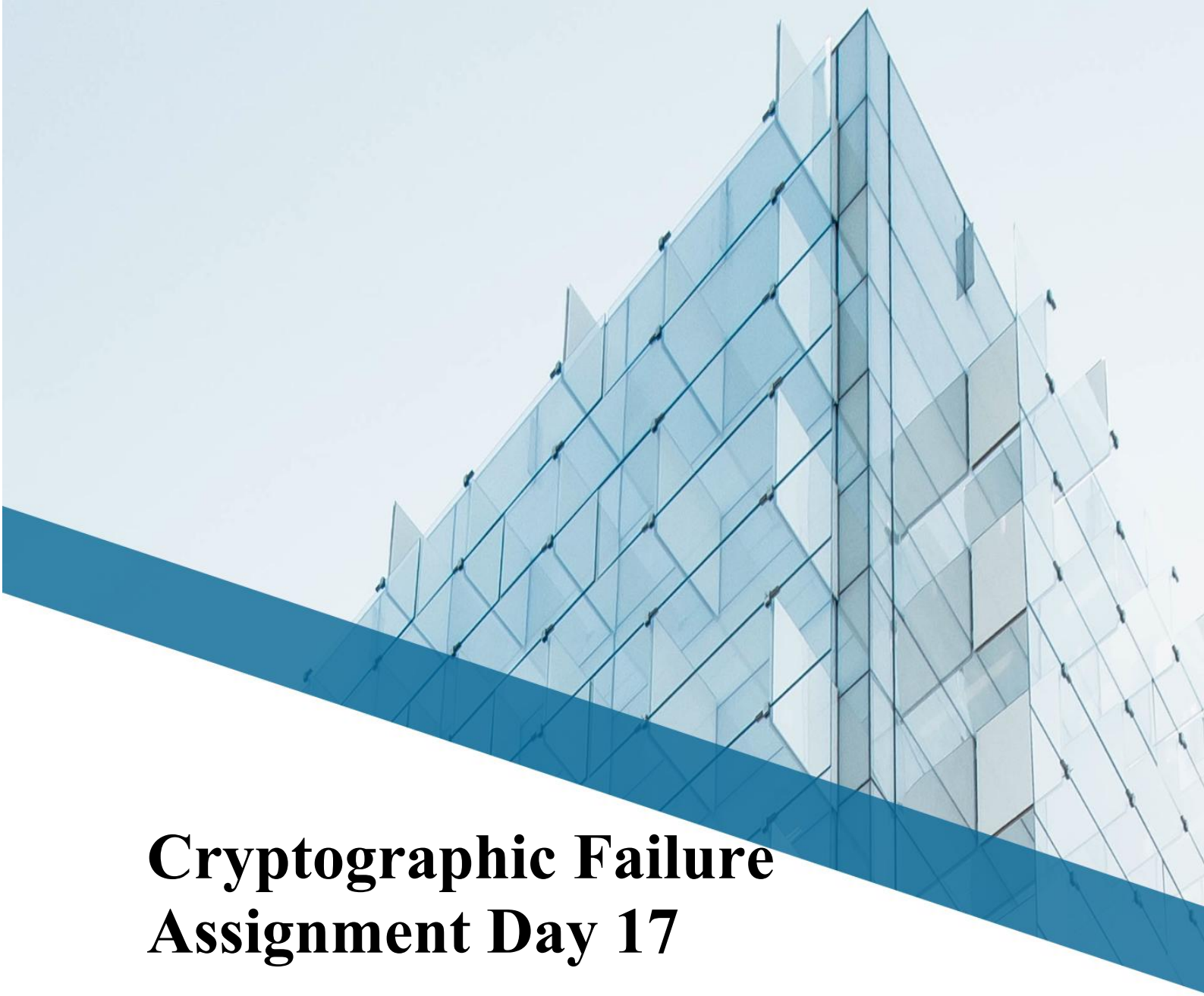


Cryptographic Failure Assignment Day 17

Bootcamp Cyber Security Batch 4

Disusun oleh: Sabilillah Ramaniya Widodo

!!Catatan: Diharapkan seluruh pengerjaan Assignment tidak sepenuhnya mengandalkan penggunaan AI!!

“Proses belajar ibarat menanam pohon. Jika hanya mengandalkan AI tanpa memahami esensinya, yang berkembang bukan kompetensimu, melainkan ketergantungan yang melemahkan.”
- Learning Design Dibimbing

1. Pendahuluan

Cryptographic Failure merupakan salah satu dari 10 risiko keamanan aplikasi web paling kritis menurut OWASP Top 10. Hal ini terjadi ketika data sensitif tidak dilindungi dengan baik, baik saat sedang disimpan (at rest) maupun saat sedang dikirim (in transit). Assignment ini bertujuan untuk mengidentifikasi, menganalisis, dan memberikan rekomendasi perbaikan terhadap beberapa contoh umum kegagalan kriptografi, seperti penyimpanan password dalam bentuk teks biasa/plaintext, penggunaan algoritma hash yang sudah usang/lemah, dan transmisi data melalui protokol HTTP yang tidak aman.

2. Penyimpanan password: plaintext vs hash

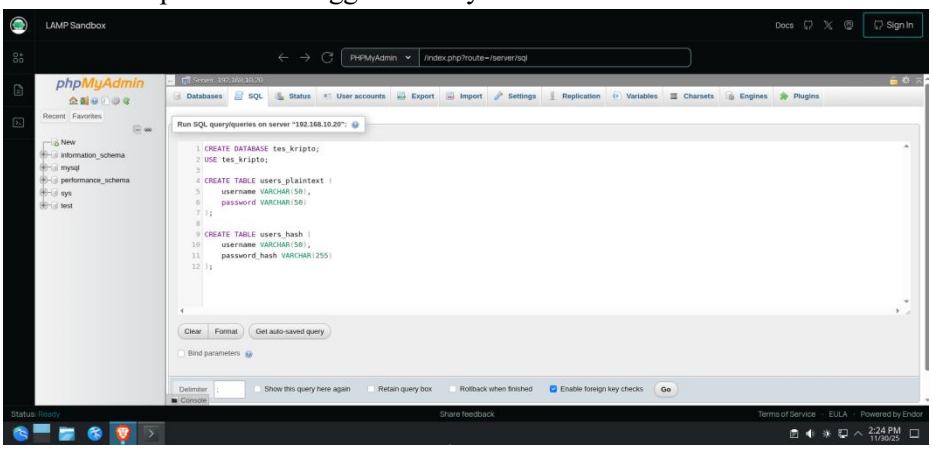
Tujuan dari bagian ini yaitu untuk membandingkan secara langsung risiko keamanan antara menyimpan password sebagai teks biasa/plaintext dengan menyimpannya dalam bentuk hash yang aman di dalam database.

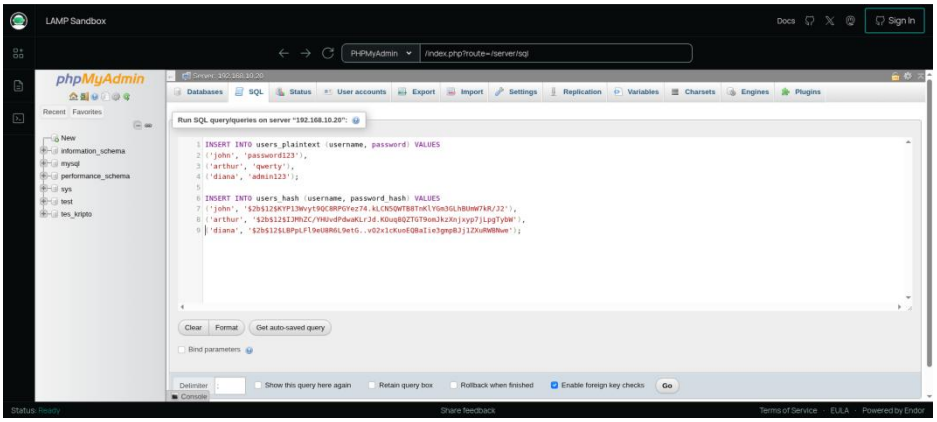
A. Proses Praktikum

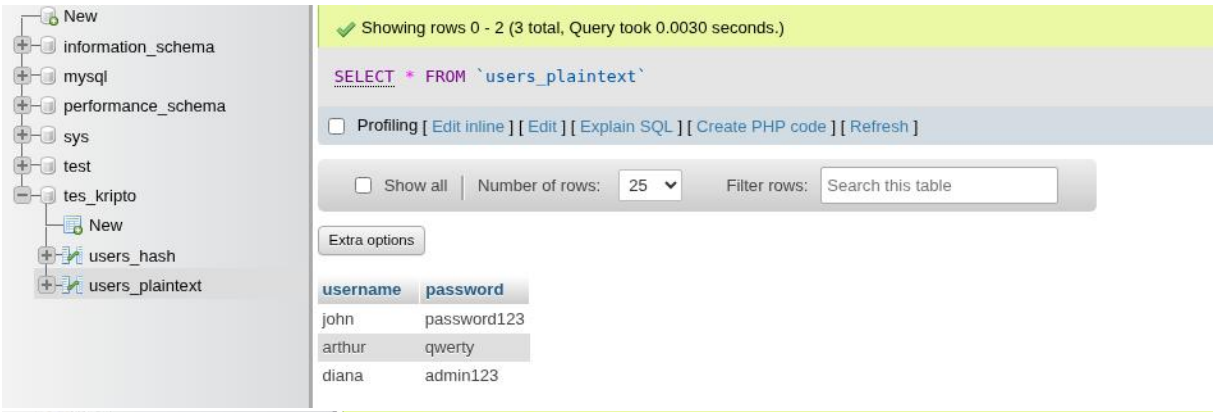
Membuat Database dan Table;
Disini saya menggunakan MySQL dari XAMPP for Browser (LAMP Sandbox) karena sederhana dan tidak memerlukan instalasi server. Buat 2 tabel:
user_plaintext untuk menyimpan password tanpa hash dan users_hash untuk menyimpan password yang sudah dihash menggunakan bcrypt.

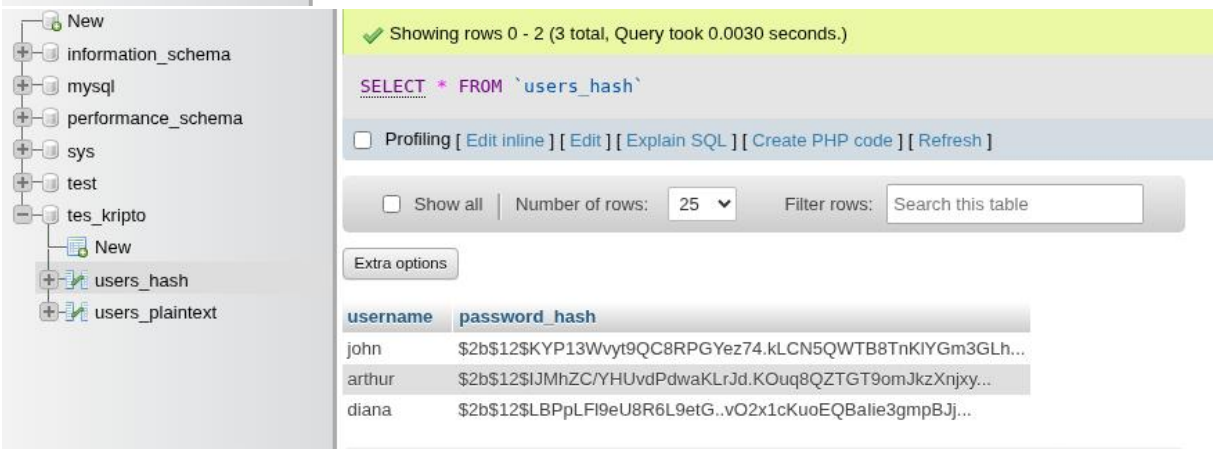
B. Menambahkan Data

Saya juga menambahkan tiga akun ke dalam masing-masing tabel. Untuk users_hash, saya membuat hash terlebih dahulu dari password menggunakan Python.









C. Analisis Risiko

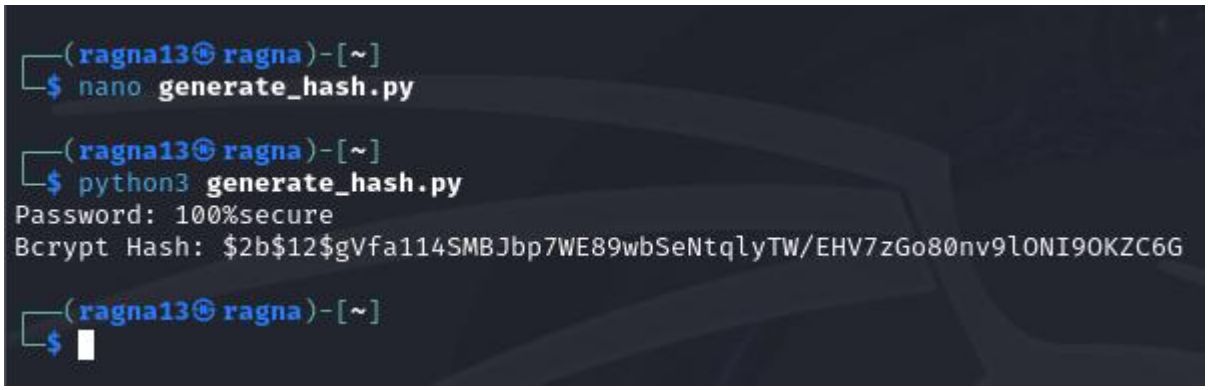
Kalau terjadi kebocoran database (Database Breach):
-Tabel users_plaintext memiliki risiko yang sangat tinggi.Attacker bisa langsung melihat dan menggunakan kombinasi username dan password semua pengguna. Ini bisa membuka pintu untuk pengambilalihan akun, pencurian identitas, dan penyerang bisa mencoba menggunakan kredensial yang sama di platform lain (credential stuffing)
-Tabel users_hash memiliki risiko yang jauh lebih rendah. Attacker hanya mendapat rentetan karakter acak/hash. Karena kita menggunakan bcrypt, yang mengandung salt unik setiap password, attacker tidak bisa menggunakan rainbow table untuk memecahkannya dengan cepat. Mereka harus melakukan serangan brute-force pada setiap hash 1 per 1, yang secara komputasi sangat mahal dan lambat.

3. Perbandingan Hashing: MD5 vs bcrypt

Bagian ini bertujuan untuk menunjukkan kelamahan algoritma hash lama yaitu MD5 dibandingkan dengan algoritma modern, bcrypt.

A. Proses Praktikum

Meng-generate Hash menggunakan Python untuk membuat hash dari sebuah password (misal: 100%secure) menggunakan MD5 atau bcrypt.

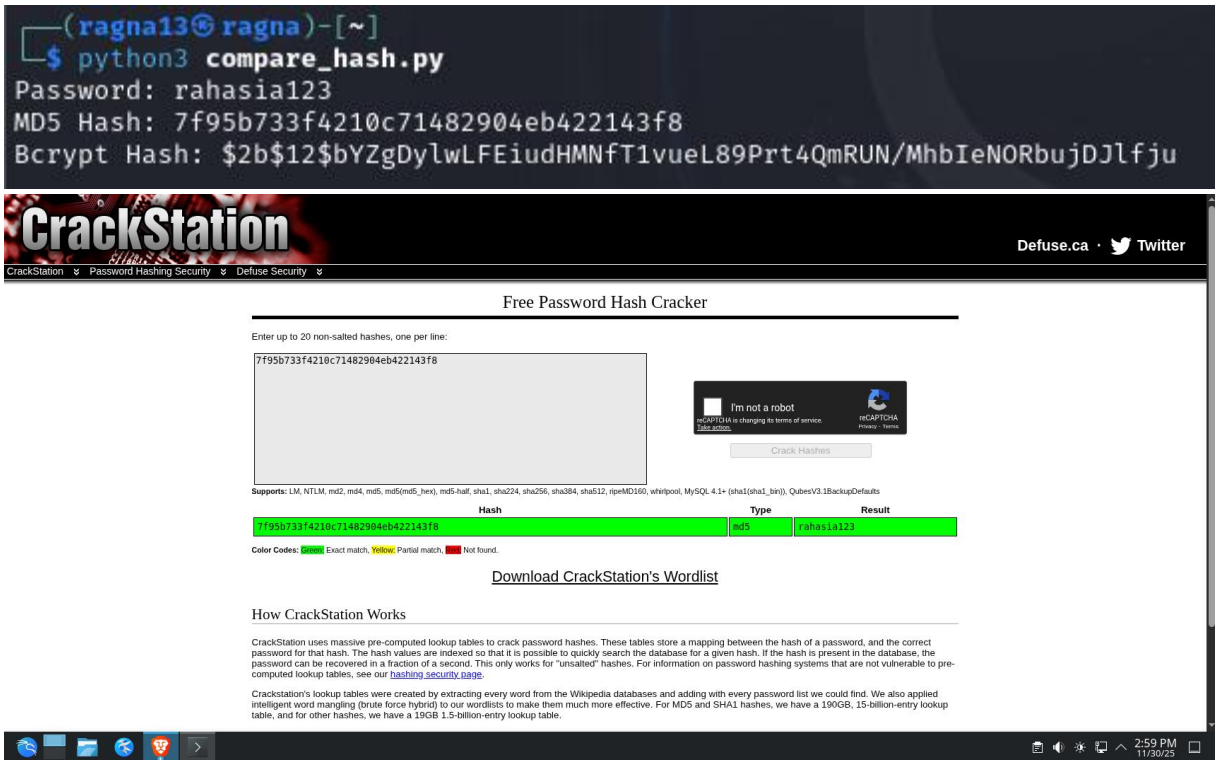


B. Analisis peran salt dan work factor pada bcrypt

Salt: salt sendiri merupakan data acak yang ditambahkan pada password sebelum proses hashing. Bcrypt disini secara otomatis menghasilkan salt yang unik untuk setiap password. Hal ini berfungsi untuk menggagalkan serangan rainbow table, karena setiap password memiliki salt yang berbeda, bahkan untuk password yang sama (misal terdapat 2 pengguna yang pakai password ‘qwerty123’), hasil hash-nya akan tetap berbeda. Hal ini membuat attacker tidak akan bisa membuat 1 table hash untuk semua kemungkinan password dan menggunakannya kembali.

Work Factor: work factor adalah parameter yang menentukan seberapa lambat dan intensif secara komputasi proses hashing bcrypt. Semakin tinggi angkanya (misal 13 atau 14), semakin lama waktu yang dibutuhkan untuk menghasilkan satu hash. Ini merupakan pertahanan yang sangat efektif melawan serangan brute-force. Kalau 1 percobaan hash membutuhkan waktu 100 milidetik, maka mencoba miliaran kombinasi akan memakan waktu bertahun-tahun, yang membuat serangan menjadi tidak praktis.

MD5 tidak memiliki 2 mekanisme ini, membuat algoritma tersebut cepat dan rentan terhadap rainbow table, sehingga sangat tidak aman untuk penyimpanan password.

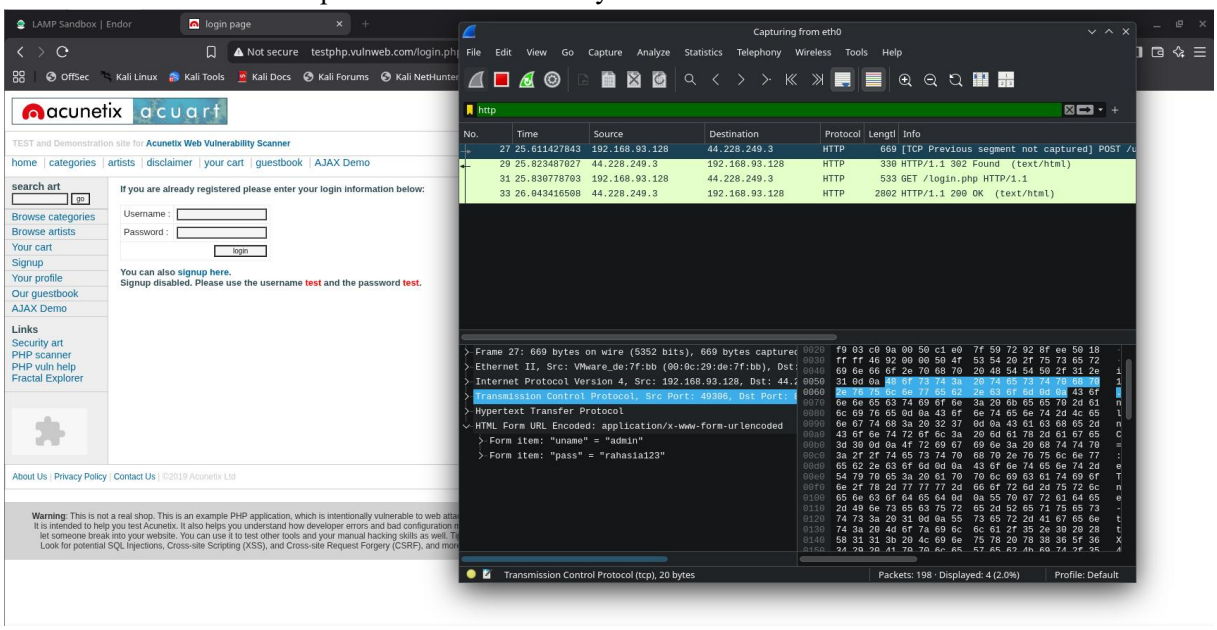


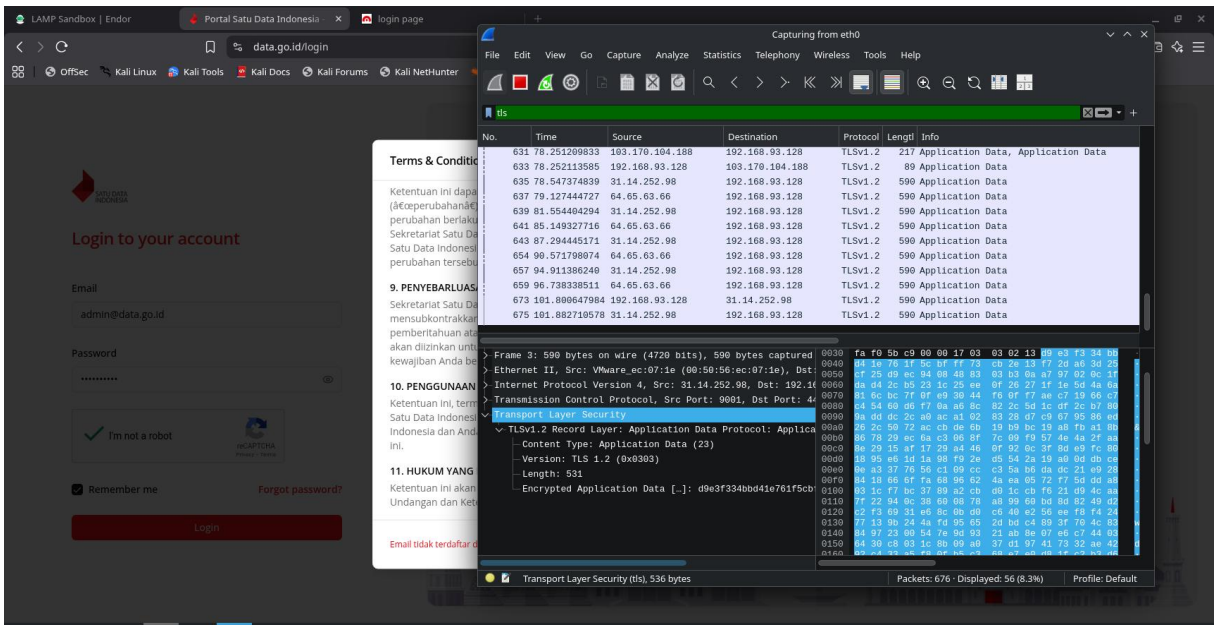
4. Analisis Trafik Login: HTTP vs HTTPS

Tujuan bagian ini adalah untuk menunjukkan secara praktis bagaimana data yang dikirim melalui HTTP bisa diintip (sniffing) dan bagaiman HTTPS melindunginya.

A. Proses Praktikum

1. Siapkan tools: Wireshark atau Tcpdump
2. Cari target: temukan sebuah website yang masih menggunakan form login HTTP. Disini saya menggunakan website yang sengaja vulnerable seperti http://testphp.vulnweb.com
3. Buka wireshark dan mulai capture data traffic pada interface jaringan yang digunakan (misal Wi-Fi atau Ethernet). Kemudian buka browser dan akses situs login HTTP.
4. Di Wireshark, bisa gunakan filter http atau http.request.method == “POST”.
5. Masukan username dan password palsu (misal admin / password123), lalu klik login.
6. Hentikan capture data trafficnya di Wiresharl dan filter paket POST atau “http” saja, lalu tekan enter. Akan terlihat kredensial dalam bentuk plaintext. Kalau misalnya website menggunakan HTTPS, Wireshark tidak akan memperlihatkan kredensialnya.





5. Rekomendasi Perbaikan

Berdasarkan temuan dari ketiga praktikum tersebut, berikut merupakan rekomendasi perbaikan yang spesifik dan praktiknya.

A. Password Plaintext

Solusi: Jangan pernah menyimpan password dalam bentuk plaintext. Semua password pengguna wajib di-hash sebelum disimpan ke database.
Praktik: Bisa gunakan algoritma hashing yang modern, adaptif, dan memiliki salt, seperti bcrypt atau Argon2.

B. Hashing Lemah - MD5

Solusi:Segera ubah semua hash password yang dibuat dengan algoritma usang seperti MD5, SHA1, dll. Dengan algoritma yang lebih kuat.
Praktik: Dengan mengimplementasikan strategi migrasi hash. Ketika pengguna login dengan password lama mereka, verifikasi menggunakan hash MD5. Jika berhasil, segera buat hash yang baru menggunakan bcrypt dan simpan di database, lalu hapus hash MD5 yang lama.

C. Transmisi via HTTP

Solusi: Mewajibkan penggunaan HTTPS di seluruh bagian website, bukan hanya pada bagian login.
Praktik:
a. Bisa dengan menginstal sertifikat SSL/TLS pada web server
b. Mengonfigurasi server untuk mengalihkan (redirect) semua trafik HTTP ke HTTPS secara otomatis (301 redirect)
c. Mengimplementasikan header HTTP Strict Transport Security (HSTS) untuk memaksa browser hanya berkomunikasi melalui HTTPS, mencegah serangan SSL striping

6. Kesimpulan

Praktikum ini menunjukkan dengan jelas akibat yang signifikan dari kegagalan kriptografi terhadap keamanan data sensitif.

Ringkasan Hasil:

- Menyimpan password sebagai plaintext sama saja dengan menyerahkan password langsung kepada attacker saat terjadi kebocoran data
- Algoritma hash yang sudah usang seperti MD5 sangat rentan dan bisa dipecahkan dalam hitungan detik menggunakan teknik modern
- Mengirim data melalui HTTP memungkinkan siapa saja di jaringan untuk mengintip dan mencuri informasi dengan mudah
- Dampaknya, kegagalan dalam mengelola dan menerapkan kriptografi bisa menyebabkan kompromi akun massal, pencurian data finansial, pelanggaran privasi, dan kerusakan reputasi yang parah untuk organisasi. Dengan demikian, penerakan enkripsi yang kuat baik untuk data at rest maupun data in transit adalah pilar fundamental dalam mengurangi risiko dalam keamanan siber.