



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

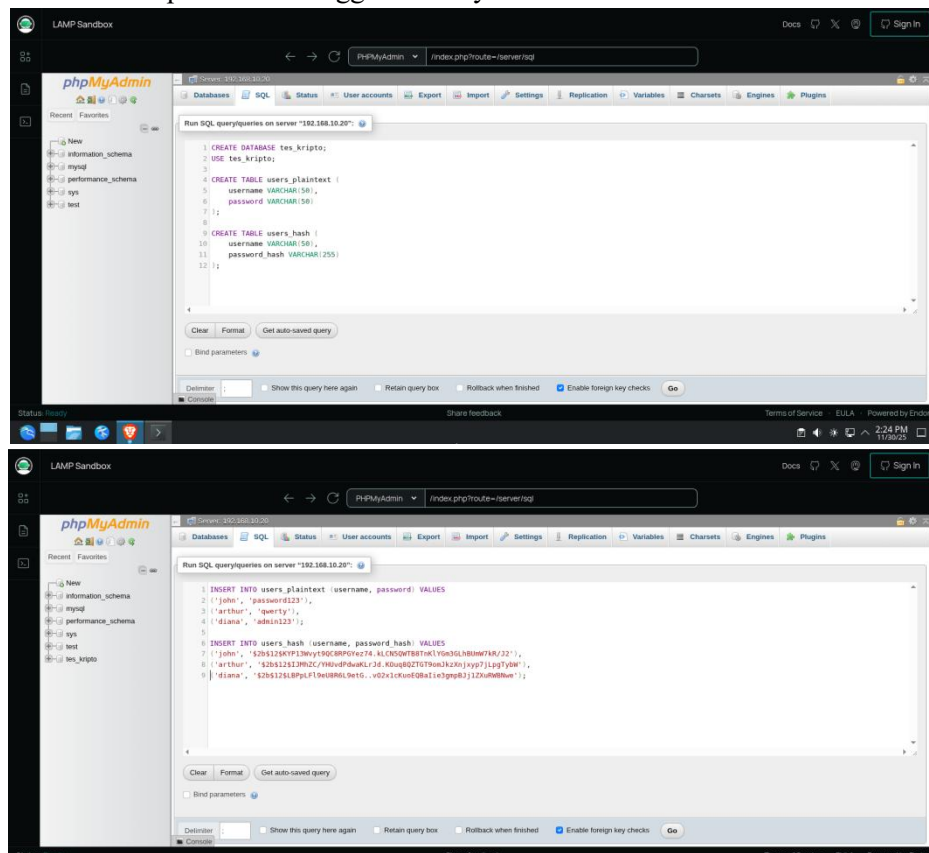
Membuat Database dan Table;

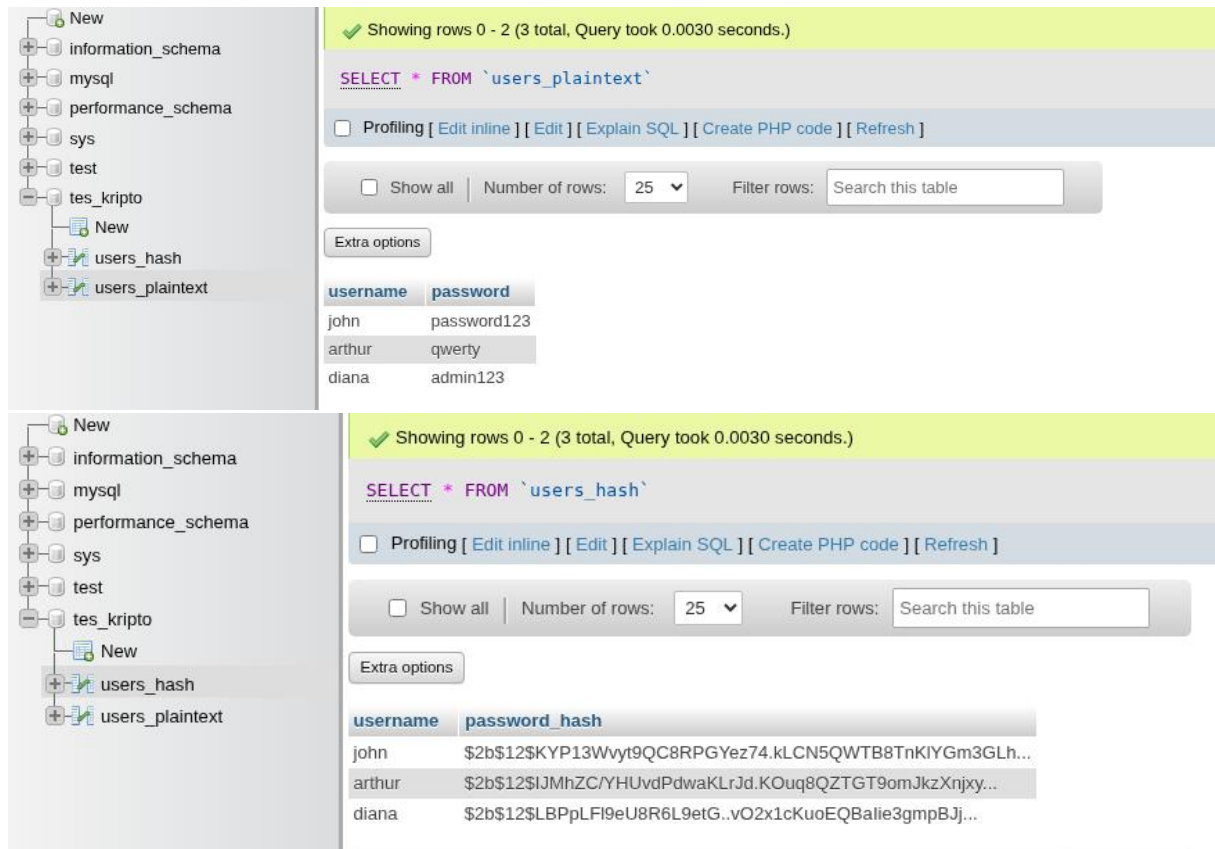
Disini saya menggunakan MySQL dari XAMPP for Browser (LAMP Sandbox) karena sederhana dan tidak memerlukan instalasi server. Buat 2 tabel:

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.





Showing rows 0 - 2 (3 total, Query took 0.0030 seconds.)

```
SELECT * FROM `users_plaintext`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

username	password
john	password123
arthur	qwerty
diana	admin123

Showing rows 0 - 2 (3 total, Query took 0.0030 seconds.)

```
SELECT * FROM `users_hash`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

username	password_hash
john	\$2b\$12\$KYP13Wvyt9QC8RPGYez74.kLCN5QWTB8TnKIYGm3GLh...
arthur	\$2b\$12\$IJmhzC/YHUvdPdwaKLrJd.KOuq8QZTGT9omJkzXnjxy...
diana	\$2b\$12\$LBPPpLFi9eU8R6L9etG..vO2x1cKuoEQBalie3gmpBJj...

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 kelemahan 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.

```
(ragna13@ragna)-[~]
$ nano generate_hash.py

(ragna13@ragna)-[~]
$ python3 generate_hash.py
Password: 100%secure
Bcrypt Hash: $2b$12$gVfa114SMBJbp7WE89wbSeNtqlyTW/EHV7zGo80nv9lONI90KZC6G

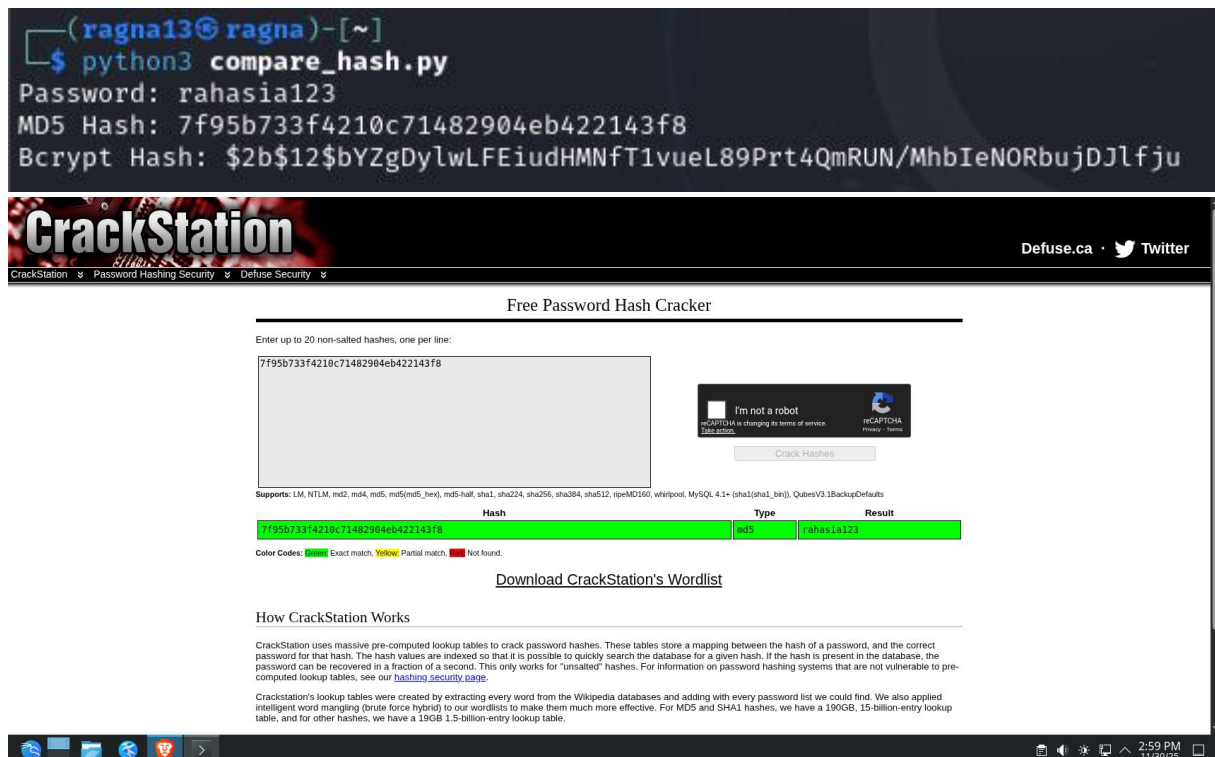
(ragna13@ragna)-[~]
$
```

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.



The screenshot shows the CrackStation website, a free password hash cracker. The interface includes a terminal window at the top showing the execution of a Python script to generate a bcrypt hash. Below the terminal, the website's header and navigation menu are visible. The main content area is titled "Free Password Hash Cracker" and contains a form for entering hashes. A sample hash is entered: 7f95b733f4210c71482904eb422143f8. The website also features a CAPTCHA and a "Crack Hashes" button. Below the form, there is a table showing the results of the hash cracking process.

Hash	Type	Result
7f95b733f4210c71482904eb422143f8	md5	100%secure

Color Codes: ■ Exact match, ■ Partial match, ■ Not found.

[Download CrackStation's Wordlist](#)

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

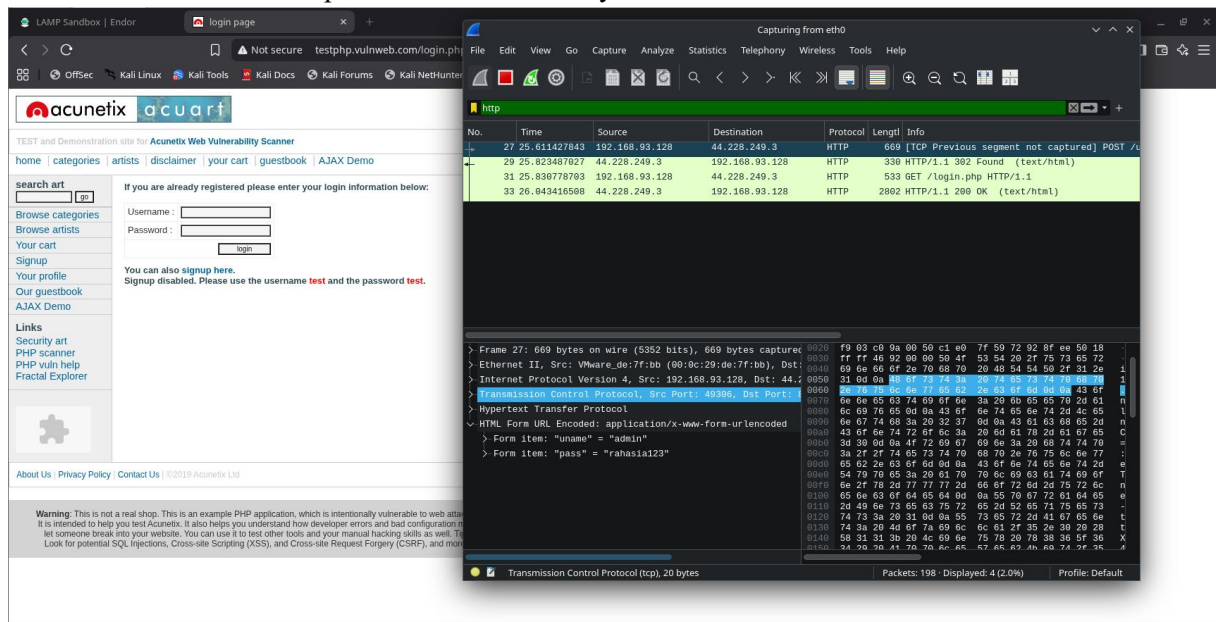
CrackStation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 100GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

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 bagaimana HTTPS melindunginya.

A. Proses Praktikum

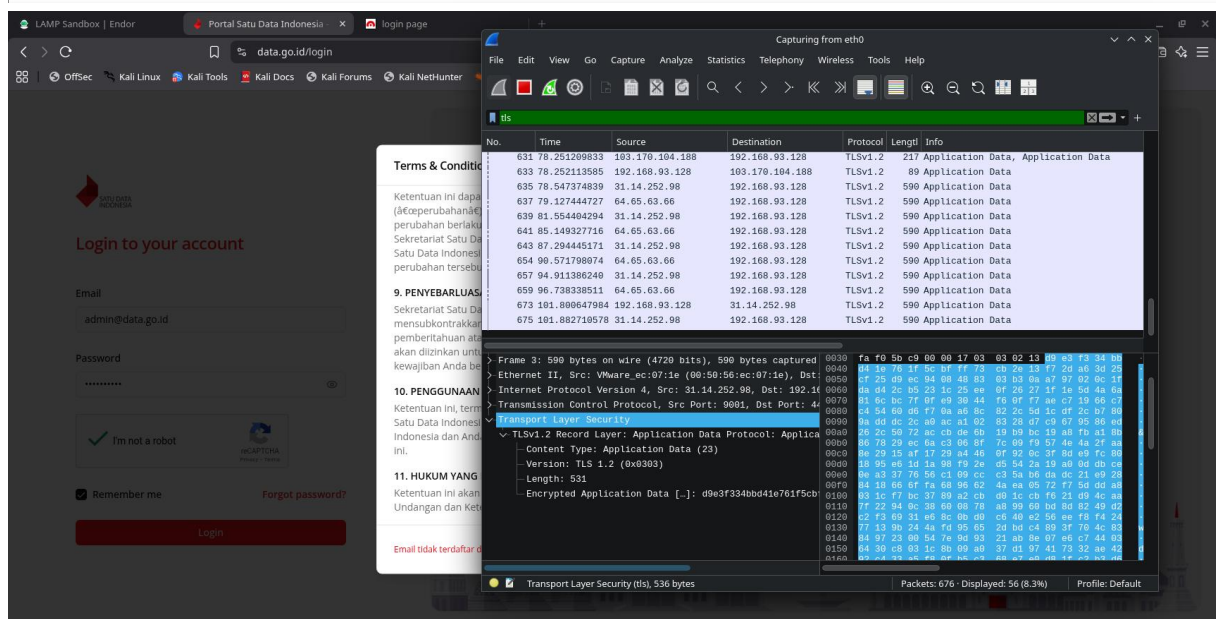
1. Siapkan tools: Wireshark atau Tcpdump
2. Cari target: temukan sebuah website yang masih menggunakan form login HTTP. Disini saya menggunakan website yang sengaja vulnerable seperti <http://testphp.vulnweb.com>
3. Buka wireshark dan mulai capture data traffic pada interface jaringan yang digunakan (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. Masukkan username dan password palsu (misal admin / password123), lalu klik login.
6. Hentikan capture data traffiknya di Wireshark 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.



The screenshot shows a web browser window with the URL `testphp.vulnweb.com/login.php`. The login form has fields for Username and Password, and a Login button. Below the form, a warning message states: "Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration can let someone break into your website. You can use it to test other tools and your manual hacking skills as well. To look for potential SQL injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more..."

Overlaid on the browser is the Wireshark network traffic capture window. The filter is set to `http`. The packet list shows several HTTP packets. The selected packet (No. 27) is an HTTP POST request to `192.168.93.128` on port 80. The packet details pane shows the following structure:

- Frame 27: 669 bytes on wire (5352 bits), 669 bytes captured
- Ethernet II, Src: VMware_de:7f:bb (00:0c:20:de:7f:bb), Dst: 192.168.93.128
- Internet Protocol Version 4, Src: 192.168.93.128, Dst: 192.168.93.128
- Transmission Control Protocol, Src Port: 49385, Dst Port: 80
- Hypertext Transfer Protocol
 - HTML Form URL Encoded: application/x-www-form-urlencoded
 - Form item: "uname" = "admin"
 - Form item: "pass" = "rahasia123"



The screenshot shows a web browser window with the URL `data.go.id/login`. The login form has fields for Email and Password, and a Login button. Below the form, there is a "Remember me" checkbox and a "Forgot password?" link. A "Terms & Conditions" link is also visible.

Overlaid on the browser is the Wireshark network traffic capture window. The filter is set to `tls`. The packet list shows several TLS packets. The selected packet (No. 3) is a TLSv1.2 record. The packet details pane shows the following structure:

- Frame 3: 590 bytes on wire (4720 bits), 590 bytes captured
- Ethernet II, Src: VMware_ec:07:1e (00:50:56:ec:07:1e), Dst: 192.168.93.128
- Internet Protocol Version 4, Src: 192.168.93.128, Dst: 192.168.93.128
- Transmission Control Protocol, Src Port: 9001, Dst Port: 443
- TLSv1.2 Record Layer: Application Data Protocol: Application Data
 - Content Type: Application Data (23)
 - Version: TLS 1.2 (0x0303)
 - Length: 531
 - Encrypted Application Data [...]: d9e3f334b0d41e761f5cb...

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 stripping

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, penerapan enkripsi yang kuat baik untuk data at rest maupun data in transit adalah pilar fundamental dalam mengurangi risiko dalam keamanan siber.