**IF4033 – Keamanan dan Penjaminan Informasi**

**Security *Website***



oleh :

13510064 Muhammad Iqbal

13510084 Jaisyalmatin Pribadi

13510102 Dedy Prasetiady

**TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2014**

# Deskripsi

*Website* yang dibangun pada tugas ini adalah sebuah *website* sederhana dengan tiga modul sederhana yang menerapkan sistem keamanan. Ketiga modul tersebut adalah modul *log in,* modul *forget password,* modul *upload file.* Modul *log in* memiliki fungsi sebagai pintu masuk *user* ke dalam *website*. Dibutuhkan *email* dan *password* sebagai tanda masuk. Modul *forget password* digunakan untuk *user* yang lupa terhadap *password*. Sedangkan modul *upload file* digunakan untuk mengunggah *file* ke dalam *website*. Tujuan dari pembangunan *website* ini adalah menjawab bagaimana cara melakukan keamanan terhadap serangan yang mungkin terjadi atau umumnya ada di dalam realitas saat ini.

# *Readme*

Langkah penggunaan pada *websitei* ini adalah:

1. Lakukan *login* pada halaman yang muncul pertama kali. Masukkan *email*  dan *password*.
2. Jika *user* lupa *password* yang dimiliki, pilih menu *forget password*. Pada langkah ini, *user* diminta memasukkan *email*. Selanjutnya *website* akan mengirimkan *token* melalui *email*  yang dicantumkan sehingga *user* dapat masuk ke dalam *webstite*.
3. Setelah *login*, *user* dapat melakukan *upload*. *User* dapat memilih *file* yang dinginkan untuk di-*upload*. Perlu diingat bahwa batasan *upload file* adalah 10MB. Saat *file* berhasil di-*upload*, *user* dapat melakukan *download* file tersebut dengan memilih *link* yang tersedia di bawah.

# *Security of the System*

Berikut akan dijelaskan langkah yang dilakukan pada setiap modul dan sistem keamanan yang telah dibangun,

## 3.1 Modul *Log in*

Pada modul ini, *user* harus menuliskan *email* dan *password* untuk membuktikan bahwa mereka memiliki *privilege* untuk masuk ke dalam *website*. Saat *user* melakukan *submit,* maka *password* tersebut akan di-*concat* dengan *salt*. Setelah itu, akan dilakukan *hash* menggunakan MD5 dengan penggunaan *salt* global.

Saat melakukan *login*, akan dilakukan fungsi *login\_attemp* berdasarkan *email* yang dimasukkan. Jika terjadi lima kali *login*, akan dilihat waktu terakhir percobaan *login*. Jika percobaan terakhir *login* ditambah lima ratus detik masih lebih besar dari waktu saat ini, mata *user* tidak dapat melakukan *login* selama lima ratus detik ke depan. Sedangkan jika waktu terakhir percobaan ditambah lima ratus detik masih lebih kecil dari waktu saat ini, maka *login\_attemp* akan direset menjadi nol.

Pada proses *login* juga dilakukan pengecekan apakah *email* tersebut ada di dalam *database*. Jika tidak ada, maka tidak akan dilakukan pengecekan *password*, namun percobaan *login* tetap dihitung. Jika *email* tersebut terdaftar pada *database* maka akan dilakukan pengecekan *password*. Diawali dengan mengambil *salt* dan *password*. Kemudia keduanya akan di-*concat* dan di-*hash* dengan *salt* global. Hasil dari langkah ini akan dibandingkan dengan *password* yang ada di dalam *database*. Jika sukses, *login\_attemp* akan dikembalikan menjadi nol. Sedangkan jika gagal maka *login\_attemp* akan ditambahkan satu dan waktu *last try* akan di-*update*.

## 3.2 Modul *Forget Password*

Saat *user* menggunakan modul *forget password*, *user* akan diminta untuk memasukkan *email* yang dimiliki *user*. Jika *email* yang dimasukkan tidak ada pada *database*, proses akan dihentikan. Jika *email* tersebut ada di dalam *database* maka akan dilanjutkan ke proses berikutnya berupa pengiriman kepada *email* tersebut. Isi dari *email* berupa *link* *reset password* beserta token. Token tersebut didapat dari hasil *generate* nilai *time stamp* yang di-*concat* dengan *salt* yang sebelumnya telah didapat dengan fungsi *get*. Semua itu akan di-*hash* dengan *salt* global. Tidak lupa dilakukan langkah pengecekan, apakah ada token pada *email* tersebut. Jika ada, token yang lama akan dihapus. Setelah itu semua yang telah dilakukan di atas akan di-*concat* dengan ID *user*. Hal ini dimaksudkan agar token bersifat identik, tidak ada kesamaan antar token yang ada. Token akan disimpan di dalam *database* beserta *email* dan *time stamp*.

Saat *user* membuka *email* dan memiliki *link* yang telah dikirim, akan dilakukan *redirect* menuju *form* yang meminta *user* memasukkan *password* yang baru. Sebelumnya akan dilakukan pengecekan apakah ada token tersebut di dalam *database*. Jika tidak, tidak akan dilakukan proses. Jika ada, akan dilakukan verifikasi terhadap *time stamp* yang ada.

## 3.3 Modul *Upload File*

Pada modul *upload* ini, setiap *user* akan memiliki *directory* masing-masing. *Directory*  ini didapat dari langkah *generate*  yang dilakukan dari ID serta *salt* yang saling di-*concat*.

Saat melakukan *upload*, akan dilakukan pengecekan *session* untuk melihat apakah *user* telah melakukan *login* atau tidak. Jika *user* belum melakukan *login* maka proses *upload* akan ditolak. Jika *user* telah melakukan *login*, maka akan diambil ID dan *salt* yang dimiliki *user* untuk menentukan *directory* yang akan digunakan untuk menyimpan *file* yang di*-upload*. Jika *directory* belum ada, maka akan dilakukan langkah *generate* untuk membuat *directory*.

Pada fungsi *upload* ini, terdapat pembatasan jumlah *file* yang di-*upload*, yaitu 10MB. Namun fungsi *upload* ini memperbolehkan *user* melakukan *upload* semua tipe *file*. Setelah *upload* dilakukan, pada halaman *upload* akan ditampilkan *link* yang berisi *file* yang telah di-*upload* oleh *user* bersangkutan. Daftar *file* yang ada disesuaikan dengan *session* yang dimiliki *user* saat itu. *Link* tiap *file* tersebut didapat dari *generate* yang dilakukan dari *salt* dan ID *user*.

# Kesimpulan

Selama pengerjaan tugas pembangunan keamanan dalam *website*, terdapat banyak pemahaman mengenai implementasi keamanan yang dilakukan terhadap *website*. Berikut merupakan kesimpulan yang dapat diambil dari pengerjaan tugas ini,

1. Pada modul *login*, keamanan dari serangan *brute force* dapat ditangani dengan adanya *login\_attemp*, sehingga batasan percobaan kegagalan dapat dijadikan idikator dalam menangkal serangan *login* secara berulang kali dalam beberapa waktu.
2. Proses *hash* yang dilakukan terhadap *password* akan membuat *attacker* tidak dapat melihat *password* seorang *user* meskipun berhasil masuk ke dalam *database*.
3. Penambahan *salt* pada *password* yang akan di-*hash* membuat sistem enkripsi berjalan lebih baik dan sulit untuk ditebak oleh *attacker*.
4. Pada modul *upload*, *user* yang tidak memiliki *privilege* terhadap *file* yang di-*upload* tidak dapat mengakses *link* yang telah disediakan karena *link* dari *file* yang telah di-*upload* akan menyesuaikan dengan *session* setiap *user* saat itu.
5. Setiap *user* akan disediakan *directory* masing-masing untuk penyimpanan *file* yang telah di-*upload*, sehingga dapat menangani pencurian *file* dari *user* asing.
6. Pembatasan jumlah *file* pada proses *upload* ditujukan agar