semanTIK, Vol.5, No.2, Jul-Des 2019, pp. 273-282
ISSN: 2502-8928 (Online)
DOI : 10.5281/zenodo.3510178
Terakreditasi "Peringkat 4 (Sinta 4)" oleh Kemenristekdikti
Website : http://ojs.uho.ac.id/index.php/semantik

IMPLEMENTASI TEKNOLOGI ENKRIPSI URL (*UNIFORM RESOURCE LOCATOR*) DAN *LOGIN FORM* MENGGUNAKAN ALGORITMA *BLOWFISH* UNTUK MENCEGAH SERANGAN SQL *INJECTION*

Wa Ode Hardianas Shalawati*[1], Muh. Yamin[2], Natalis Ransi[3]
[*1,2,3]Jurusan Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari
e-mail: *[1]hardianas.waode25@gmail.com, [2]putra0683@gmail.com, [3]natalis.ransi@uho.ac.id

Abstrak

Keamanan *website* menjadi aspek yang sangat penting. Karena *website* yang tidak aman akan mudah dirusak integritas data dan informasinya melalui berbagai serangan yang ada. SQL *Injection* adalah salah satu jenis serangan yang sering terjadi pada situs *web*. SQL *Injection* adalah sebuah metode untuk memasukan perintah SQL sebagai *input* melalui sebuah *web* guna mendapatkan akses *database*. URL dan *Login Form* yang tidak aman sering menjadi target serangan SQL *Injection* Banyak cara untuk mencegah serangan SQL *Injection* dan salah satunya meggunakan teknik kriptografi untuk mengenkripsi URL dan *Login Form website* menjadi kode atau sandi yang tidak dapat dimengerti. Metode kriptografi yang digunakan adalah algoritma *blowfish* yang sudah sering digunakan untuk mengamankan *file*. Penelitan ini bertujuan untuk menguji performa *blowfish* untuk mencegah serangan SQL *Injection* melalui penerapan enkripsi URL dan *Login Form*. Keamanan *web* dari serangan SQL *Injection* diuji sebelum dan sesudah penerapan enkripsi. Pengujian serangan dilakukan secara manual dan otomatis. Pengujian otomatis menggunakan aplikasi SQL *Map* dan JSQL *Injection*. Hasil pengujian membuktikan bahwa penerapan enkripsi URL dan *login form* menggunakan algoritma *blowfish* efektif untuk mencegah serangan SQL *Injection* karena serangan SQL *Injection* selalu gagal setelah penerapan enkripsi URL dan *Login Form*.

Kata kunci—Algoritma *Blowfish*, Enkripsi, SQL *Injection*, URL, *Login Form*

Abstract

Security of a website is a very important aspect. Because unsafe websites will easily be damaged the integrity of the data and information through various attack. SQL Injection is one type of attack that often occurs on websites. SQL Injection is a method for entering SQL commands as an input through a web to get database access. Insecure URLs and Login forms are often targeted by SQL Injection attacks to exploit web databases. There are many ways to prevent SQL Injection attacks and one of them uses cryptographic techniques to encrypt URLs and Login Forms of websites into an incomprehensible codes or passwords. The cryptographic method used is blowfish algorithm. This research aims to tests the performance of blowfish to prevent SQL Injection attacks through the implementation of URL and Login Form encryption. Web security from SQL Injection attacks is tested before and after encryption. Attack testing is done manually and automatically. Automatic testing using SQL Map and JSQL Injection applications. The test results prove that the implementation of URL and login form encryption uses blowfish algorithm is effective to prevent SQL Injection attacks because SQL Injection attacks always fail after implementation of URL and Login Form encryption.

Keywords*— Blowfish Algorithm, Encryption, SQL Injection, URL, Login Form*

1. PENDAHULUAN

Sebagai media penyampaian informasi, keamanan sebuah *website* menjadi hal yang penting. Karena *website* yang tidak aman akan mudah diserang untuk merusak, mengubah, ataupun memanipulasi sistem yang bekerja pada *website.* Salah satu jenis serangan yang sering terjadi dilihat dari sisi keamanan *website* adalah serangan SQL *Injection*. Berdasarkan laporan dari OWASP (*Open Web Aplication Security Project*), SQL *Injection* menempati urutan teratas daftar 10 celah keamanan pada aplikasi *web* yang paling berbahaya [1].

SQL *Injection* merupakan sebuah kerentanan yang menyebabkan seorang penyerang memiliki kemampuan untuk mempengaruhi *query* SQL yang dikirimkan melalui aplikasi ke *database*. Dengan kemampuan tersebut seorang penyerang dapat mempengaruhi *syntax* SQL, kekuatan, dan fleksibilitas dari *database* sehingga memungkinkan penyerang untuk masuk ke dalam sistem *website* yang harus dijaga keamananya [2].

Salah satu cara untuk mengamankan *website* dari serangan SQL *Injection* adalah dengan mengenkripsi URL dan *login form* pada *website*. URL *web* dan *login form* sering digunakan sebagai media untuk melakukan tindak kejahatan terhadap suatu *web* dengan cara memberikan berbagai macam perintah terhadap basis data pada server *website* tersebut. Dengan melakukan enkripsi URL dan *login form* pada *website* maka akan menyamarkan URL dan *login form website* menjadi kode atau sandi yang tidak dapat dibaca oleh sembarang orang sehingga dapat mencegah serangan SQL *Injection* pada *website*.

Metode enkripsi yang digunakan adalah algoritma *blowfish*. Penggunaan Algoritma *blowfish* didasarkan pada kelebihan sifatnya yang mudah, cepat, tersusun dengan rapi dan terjamin keamanannya. Algoritma *Blowfish* memiliki desain yang sangat sederhana dan mudah dipahami. Algoritma ini mudah untuk diimplementasikan karena menggunakan modifikasi Feistel *Cipher* serta *S-box* dan *P-array* yang sederhana. Telah banyak penelitian yang mengangkat algoritma *blowfish* sebagai metode enkripsi data dan telah teruji tingkat keamanannya. Namun belum ada yang menerapkan algoritma *blowfish* untuk keamanan URL dan *login form website.* Olehnya itu penulis tertarik menguji keamanan *website* dari serangan SQL *Injection* dengan cara menerapkan enkripsi URL dan *login form* menggunakan algoritma *blowfish.*

2. METODE PENELITIAN

2.1 Kriptografi

Kriptografi (*cryptography*) berasal dari dua kata yaitu *Crypto* dapat diartikan rahasia (*secret*) dan *graphy* dapat diartikan tulisan (*writing*) jadi kriptografi dapat diartikan sebagai suatu ilmu atau seni untuk mengamankan pesan agar tidak diketahui oleh pihak yang tidak diinginkan atau kriptografi adalah seni dari penulisan rahasia atau membaca sandi atau tulisan-tulisan rahasia [3].

2.2 Enkripsi dan Dekripsi

Proses menyandikan *plaintext* menjadi *ciphertext* disebut enkripsi (*encryption*). Sedangkan proses mengembalikan *ciphertext* menjadi *plaintext* semula dinamakan dekripsi (*decryption*). Enkripsi dan dekripsi dapat diterapkan baik pada pesan yang dikirim maupun pada pesan tersimpan.

2.3 Algoritma *Blowfish*

Blowfish menggunakan sub kunci berukuran besar. Kunci-kunci tersebut harus dikomputasikan pada saat awal, sebelum pengkomputasian enkripsi dan dekripsi data [4].

Untuk alur algoritma enkripsi dengan metode *blowfish* dijelaskan sebagai berikut:

1. Terdapat kotak permutasi (P-*box*) yang terdiri dari 18 buah 32-*bit* sub kunci: P1,P2,P3….,P18. P-*box* ini sudah ditetapkan sejak awal , 4 buah P-*box* awal adalah sebagai berikut:
   P1=0x243f6a88
   P2=0x85a308d3
   P3=0x13198a2e
   P4=0x03707344
2. XOR kan P1 dengan 32-*bit* awal kunci, XOR kan P2 dengan 32 *bit* berikutnya dari kunci, dan teruskan hingga seluruh panjang kunci telah ter XOR kan

(kemungkinan sampai P14, 14x32=448, panjang maksimum kunci).
3. Terdapat 64 *bit* dengan isi kosong, *bit-bit* tersebut dimasukkan ke langkah 2.
4. Gantikan P1 dan P2 dengan keluaran dari langkah 3.
5. Enkripsikan keluaran langkah 3 dengan langkah 2 kembali, namun kali ini dengan subkunci yang berbeda (sebab langkah 2 menghasilkan subkunci baru).
6. Gantikan P3 dan P4 dengan keluaran dari langkah 5
7. Lakukan seterusnya hingga seluruh P-*box* teracak sempurna.
8. Total keseluruhan terdapat 521 iterasi untuk menghasilkan subkunci-subkunci yang dibutuhkan. Aplikasi hendaknya menyimpannya dari pada menghasilkan ulang subkunci-subkunci tersebut. Kunci-kunci yang digunakan antara lain terdiri dari 18 buah 32-*bit subkey* yang tergabung dalam P-*array* (P1,P2,P3,…,P18). Selain itu ada pula empat 32-*bit* S-box yang masing-masingnya memiliki 256 entri:
   S1,0,S1,1,…,S1,255;
   S2,0,S2,1,…,S2,255;
   S3,0,S3,1,…,S3,255;
   S4,0,S4,1,…,S4,255;

Pada jaring feistel, *blowfish* memiliki 16 iterasi, masukannya adalah 64 *bit* elemen data, X. Untuk melakukan proses enkripsi:
1. Bagi X menjadi dua bagian yang masing-masing terdiri dari 32 *bit* : XL, XR.
2. For i= 1 to 16:
   XL = XL xor Pi
   XR = F(XL) xor XR
   Tukar XL dan XR
3. Setelah iterasi ke –enam belas, tukar XL dan XR lagi untuk melakukan *undo* pertukaran terakhir
4. Lalu lakukakan
   XR = XR XOR P17
   XL = XL XOR P18
   Terakhir, gabungkan kembali XL dan XR untuk mendapatkan *ciphertext*.

Algoritma *blowfish* memiliki keunikan dalam hal proses dekripsi, yaitu proses dekripsi dilakukan sama persis dengan prose enkripsi, hanya saja pada proses dekripsi P1,P2,….,P18 digunakan dalam urutan terbalik.

2.4 SQL *Injection*

SQL *Injection* adalah sebuah aksi *hacking* yang dilakukan di aplikasi *client* dengan cara memodifikasi perintah SQL yang ada di aplikasi *client*. SQL *Injection* merupakan teknik mengeksploitasi aplikasi berbasis yang didalamnya menggunakan basis data untuk penyimpanan data [5].

2.5 URL (*Uniform Resource Locator*)

URL (*Uniform Resource Locator*) adalah sebuah alamat yang menunjukan rute ke *file* pada *web* atau pada fasilitas *internet* yang lain. URL diketikkan pada *browser* untuk mengakses suatu situs *web* [6].

2.6 Sistem *Login*

Sistem *login* (*login*, juga biasa disebut *log in* , *log on*, *sign in*, *signin*, *sign on*, *sign on*) adalah proses untuk mengakses komputer dengan memasukkan identitas dari akun pengguna dan kata sandi untuk mendapatkan hak akses menggunakan sumber daya komputer tujuan [7]

2.6 Rancangan Pengujian

Pengujian sistem dari serangan SQL *Injection* dilakukan dengan 3 tahap pengujian. Tahap pertama adalah pengujian dengan melakukan serangan SQL *Injection* secara manual. Tahap kedua melakukan serangan SQL *Injecton* menggunakan aplikasi SQL *Map* dan terakhir pengujian menggunakan aplikasi JSQL *Injection*.

3. HASIL DAN PEMBAHASAN

3.1 Analisis Sistem

Sistem yang diajukan merupakan sistem yang berbasis *website*. Aplikasi ini merupakan aplikasi yang digunakan untuk SIUP (Surat Izin Usaha Perdagangan) yang mengimplementasikan teknologi enkripsi URL (*Uniform Resource Locator*) dan *login form* dari serangan SQL *Injection* menggunakan algoritma *blowfish*. Berikut adalah alur kerja yang ditempuh dalam sistem yang diajukan:
1. *User* melakukan registrasi.
2. *User* melakukan *login* ke sistem.
3. *User* melakukan serangan SQL *injection* ke sistem.

3.2 Analisis *Requirement*

Kebutuhan sistem atau *System Requirement* pada sistem ini dibagi atas dua kelompok, yaitu *Functional Requirement* dan *Non-Functional Requirement*. Penjelasan dari masing-masing kelompok akan dijelaskan sebagai berikut:

1. *Functional Requirement* / Kebutuhan Fungsional

Functional Requirement, adalah kebutuhan atau proses yang harus dikerjakan atau informasi yang harus dimuat oleh sistem yang berkaitan dengan fungsi sistem. *Functional Requirement* pada sistem ini, akan ditampilkan pada Tabel 1.

Tabel 1 Tabel *Functional Requirement*

No.	***Functional Requirement***
R1	Sistem harus dijalankan dengan membuat akun terlebih dahulu
R2	Sistem harus dijalankan dengan melakukan proses *login*
R3	Sistem memiliki menu *dashboard*, permohonan, profil dan data *user*

2. *Non-Functional Requirement* / Kebutuhan Non Fungsional

Non-Functional Requirement mengacu pada kinerja pada sebuah sistem maupun penggunaannya.Bagian ini menjelaskan tentang kebutuhan dari *software/tools* dan *hardware* pada tahap pengembangan dan implementasi.

Tabel 2 Kebutuhan *Software* pada Tahap Pengembangan

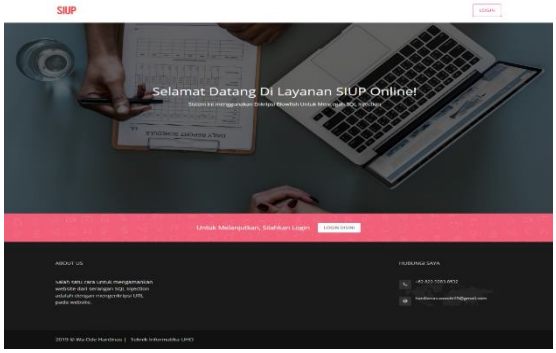
No	Nama Perangkat	Spesifikasi
1.	Sistem Operasi	*Windows 10*
2.	*Code Igniter*	*3.1.9*
3.	*Visual Studio Code*	*V 1.34.0*
4.	XAMPP	*3.2.2*
5.	*Mozilla Firefox*	*66.0.3*
6.	SQL *Map*	*1.2.5*
7.	*Python*	*3.7.2*
8.	JSQL *Injection*	*0.81*
9.	*Command Prompt* (CMD)	

Tabel 3 Kebutuhan *Hardware* pada Tahap Pengembangan

No	Nama Perangkat	Spesifikasi
1.	*Processor*	Intel Core i3
2.	Memori	RAM 4 GB
3.	*Harddisk*	500 GB

3.3 Implementasi

Implementasi sistem merupakan sebuah perancangan sistem yang dibuat dalam bentuk nyata. Sistem ini dibuat dengan bahasa Pemrograman Berbasis PHP. Aplikasi SIUP *Online* dirancang dengan mengimplementasikan enkripsi *blowfish* pada URL dan *Login Form* sistem untuk mencegah serangan SQL *Injection*.



Gambar 4 Halaman Utama Sistem

3.3 Pengujian

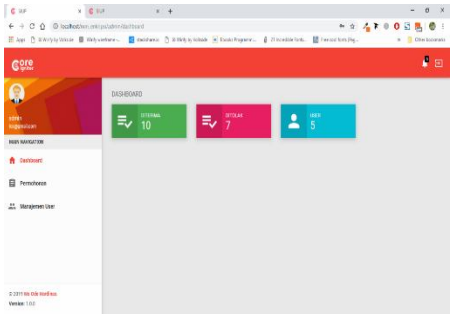
Pada pengujian ini dilakukan serangan SQL *Injection* pada sistem yang telah dienkripsi dan tidak terenkripsi (Non-enkripsi) dengan menggunakan 3 Metode/*Tools* yaitu Manual, SQL *Map*, dan JSQL *Injection*. Ada 2 Kondisi dari hasil Pengujian, yaitu *PASS* dan *FAIL*. Kondisi *PASS* terjadi jika proses injeksi berhasil dilakukan, dan kondisi *FAIL* terjadi jika proses injeksi gagal dilakukan. Berikut merupakan tabel pengujian yang akan digunakan untuk pengujian pada aplikasi *web* yang terenkripsi dan tidak terenkripsi.

Tabel 5 Skenario Pengujian *SQL Injection*

No.	SQL *Injection*	Skenario		
1	Testing SQL *Injection* dapat di lakukan pada target	Target	:	*Login Form*
		SQL Target	:	SELECT * FROM table_user WHERE Username='$username' AND Password='$password'
		SQL *Injection* 1	:	username input: $username = Random text'
		SQL *Injection* 2	:	user_password input: $password = Random text'
		Injection Result	:	SELECT * FROM table_user WHERE user_name='$username'' AND user_password='$password'' ''
		Hasil yang diharapkan	:	Muncul *error database* dengan *table name* dan *table column*
2	SQL *Injection* Klasik. Untuk menghasilkan nilai *true* dalam *query*	Target	:	*Login Form*
		SQL Target	:	SELECT * FROM table_user WHERE Username='$username' AND Password='$password'
		SQL *Injection* 1	:	username input: $username = 1' or '1' = '1
		SQL *Injection* 2	:	user_password input: $password = 1' or '1' = '1
		Injection Result	:	SELECT * FROM table_user WHERE user_name='1' OR '1' = '1' AND user_password='1' OR '1' = '1'
		Hasil yang diharapkan	:	*Login* Berhasil di lakukan
3	SQL Injeksi Klasik, *password* terenkripsi. Untuk menghasilkan nilai dalam *query* dengan menjadikan *field password* sebagai komen	Target	:	*Login Form*
		SQL Target	:	SELECT * FROM table_user WHERE Username='$username' AND Password='$password'
		SQL *Injection* 1	:	username input: $username = 1' or '1' = '1' -- /*
		SQL *Injection* 2	:	user_password input: $password = Random text
		Injection Result	:	SELECT * FROM table_user WHERE ((user_name='1' or '1' = '1'))/*') AND (user_password=enkripsi('$password')))
		Hasil yang diharapkan	:	*Login* Berhasil di lakukan
4	*Simple Select Statement*	Target	:	URL
		SQL Target	:	SELECT * FROM table_permohonan WHERE id_produk = '$id' *AND* 1=1 -- /*'
		URL	:	http://www.example.com/product.php?id=10 *AND* 1=1
		SQL *Injection* 1	:	id=1 ' AND 1=1 -- /*
		SQL *Injection* 2	:	id=1' AND 1=2 -- /*
		Injection Result	:	SELECT * FROM table_permohonan WHERE id_produk = '$id' AND 1=1 -- /*
		Hasil yang diharapkan	:	AND 1=1 menampilkan data, AND 1=2 menampilkan *error*
5	Teknik *Exploitation Union* Step1. Untuk mendapatkan jumlah kolom	Target	:	URL
		SQL Target	:	SELECT * FROM table_permohonan WHERE id_produk = $id
		URL	:	http://www.example.com/product.php?id=1 ' ORDER BY 10 -- /*
		SQL *Injection*	:	' ORDER BY 10 -- /*
		Injection Result	:	SELECT * FROM table_permohonan WHERE id_produk = '$id' ORDER BY 10 -- /*
		Hasil	:	Jika angka lebih tinggi dari jumlah kolom, akan tampil halaman *error*. Kalau angka lebih rendah atau sama dengan jumlah kolom, halaman tampil seperti biasa
6	Teknik *Exploitation Union Step* 2	Target	:	URL
		SQL Target	:	SELECT * FROM table_permohonan WHERE id_produk = $id
		URL	:	http://www.example.com/product.php?id=1 ' UNION ALL SELECT user_name,user_password,1,1,n..n FROM table_user WHERE user_name = 'admin' LIMIT 1,1 -- /*
		SQL *Injection*	:	' UNION ALL SELECT 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 FROM table_user WHERE user_name = 'admin' LIMIT 1,1 -- /*
		Injection Result	:	SELECT * FROM table_permohonan WHERE id_produk = '$id' UNION ALL SELECTuser_name,user_password,1,1,n..n FROM table_user WHERE user_name = 'admin' LIMIT 1,1 -- /*
		Hasil yang diharapkan	:	Data yang di *exploit* dapat di tampilkan ke tempat di dmana *field* data di tampilkan dengan menggati *value* pada saat UNION

1. Pengujian SQL *Injection* pada Aplikasi Nonenkripsi
   a. Pengujian Manual
      Berikut pengujian manual dengan target *login form*. Inputan injeksi sesuai pada skenario 2 pada table 5 Skenario Pengujian SQL *Injection*

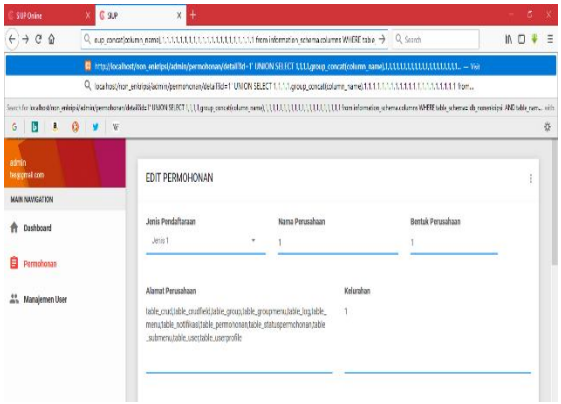
Gambar 5 Pengujian Manual *Login Form* (Aplikasi Nonenkripsi)



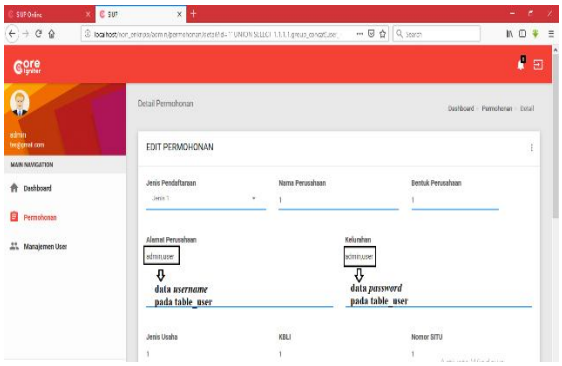
Gambar 6 Hasil Pengujian Manual *Login Form* (Aplikasi Nonenkripsi)

SQL *Injection* berhasil dilakukan atau *PASS*. Untuk melewati proses *login* masing-masing dari *username* dan *password* harus bernilai *True*. Penyerang dapat masuk kedalam sistem tanpa mengetahui *username* dan *password* karena penyerang menggunakan *input*-an logika *OR* pada *username* dan *password*. Apabila *input*-an *username* dan *password* bernilai *False* maka agar *input*-an bernilai *True* di *OR* kan dengan 1=1 yang bernilai *True*. Pada logika *OR* ketika *False* di *OR* kan dengan *True* maka hasilnya *True*. Sehingga system akan membaca *input*-an *username* dan *password* nya bernilai *True* dan penyerang dapat masuk kedalam sistem.

Berikut pengujian manual dengan target URL. Inputan injeksi sesuai skenario ke- 6 pada Tabel 5.



Gambar 7 Pengujian Manual URL (Aplikasi Nonenkripsi)

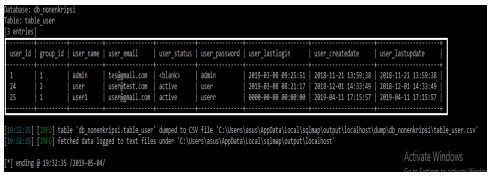


Gambar 8 Hasil Pengujian Manual URL (Aplikasi Nonenkripsi)

SQL *Injection* berhasil dilakukan atau *PASS*. Adapun hasil dari pengujian tersebut adalah diperoleh ekstraksi data dari kolom. Pada perintah kali ini digunakan tabel *table_user* kolom *user_name* dan kolom *user_password* yang diekstak datanya.

b. Pengujian Menggunakan SQL *Map*

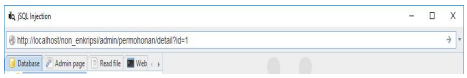
Gambar 9 Perintah Pengujian Menggunakan SQL *Map* (Aplikasi Nonenkripsi)



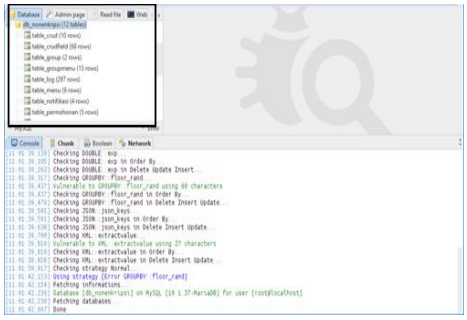
Gambar 10 Hasil Pengujian Menggunakan SQL *Map* (Aplikasi Nonenkripsi)

Penyerang dapat memperoleh seluruh informasi dari *database* menggunakan SQL *Map* menunjukan bahwa serangan SQL *Injection* berhasil dilakukan atau *PASS*.

c. Pengujian Menggunakan JSQL *Injection*



Gambar 11 Pengujian Menggunakan JSQL *Injection*



Gambar 12 Hasil Pengujian Menggunakan JSQL *Injection*

Adapun hasil dari pengujian menggunakan JSQL *Injection* adalah berhasil dilakukan serangan SQL *Injection* atau *PASS*. Diperoleh seluruh informasi *database* dari sistem.

2. Pengujian SQL *Injection* pada Aplikasi Terenkripsi

a. Pengujian Manual

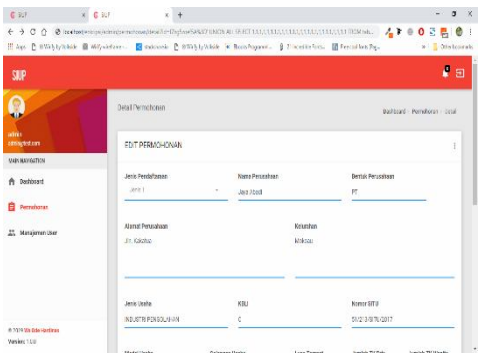
Berikut pengujian manual target *login form*. *Input*-an injeksi sesuai pada skenario 2 pada Tabel 5 Skenario Pengujian SQL *Injection*

Gambar 13 Pengujian Manual *Login Form* (Aplikasi Terenkripsi)

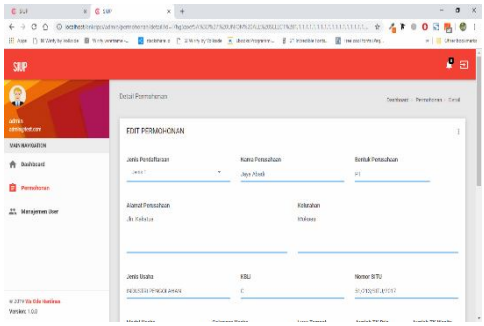
Gambar 14 Hasil Pengujuan Manual *Login Form* (Aplikasi Terenkripsi)

Adapun hasil dari pengujian tersebut yaitu terjadi *error* / *username* dan *password* salah. Jadi serangan SQL *Injection* menggunakan *input*-an ini tidak berhasil atau *FAIL* pada aplikasi terenkripsi menggunakan *blowfish*.

Berikut pengujian manual dengan target URL. Inputan injeksi sesuai pada skenario 6 pada table 5 Skenario Pengujian SQL *Injection*



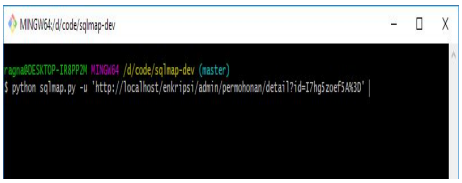
Gambar 15 Pengujian Manual URL (Aplikasi Terenkripsi)



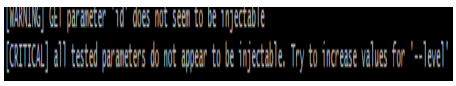
Gambar 16 Hasil Pengujian Manual URL (Aplikasi Terenkripsi)

Adapun hasil dari pengujian tersebut yaitu Data yang di *exploit* tidak dapat di tampilkan ke tempat dimana *field* data di tampilkan dengan mengganti *valu**e.*** Jadi serangan SQL *Injection* gagal dilakukan atau *FAIL* pada aplikasi terenkripsi menggunakan *blowfish*.

b. Pengujian Menggunakan SQL *Map*



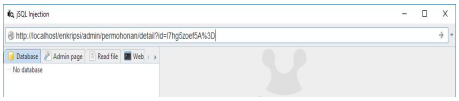
Gambar 18 Perintah Pengujian Menggunakan SQL *Map* (Aplikasi Terenkripsi)



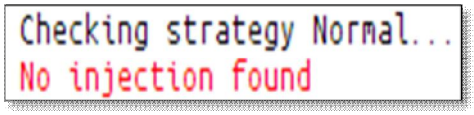
Gambar 19 Hasil Pengujian Menggunakan SQL *Map* (Aplikasi Terenkripsi)

Adapun hasil dari pengujian tersebut yaitu serangan SQL *Injection* tidak bisa dilakukan atau *FAIL*.

c. Pengujian Menggunakan JSQL *Injection*



Gambar 20 Pengujian Menggunakan JSQL *Injection*(Aplikasi Terenkripsi)



Gambar 22 Detail Hasil Pengujian Menggunakan JSQL *Injection* (Aplikasi Terenkripsi)

Adapun hasil dari pengujian menggunakan JSQL *Injection* untuk aplikasi terenkripsi adalah gagal dilakukan serangan SQL *Injection* atau *FAIL*.

Hasil dari pengujian lengkap SQL *Injection* pada Aplikasi yang telah terenkripsi dan tidak terenkripsi tampil pada Tabel 6 dibawah ini.

Tabel 6 Hasil Pengujian Aplikasi yang Terenkripsi dan Tidak Terenkripsi Menggunakan *Blowfish*

No	SQL *Injection*	Non-Enkripsi			Enkripsi		
		Manual	SQL *Map*	JSQL Injection	Manual	SQL *Map*	JSQL *Injection*
1	Testing SQL *Injection* dapat dilakukan pada target	*Fail*	*Fail*	*Fail*	*Pass*	*Pass*	*Pass*
2	SQL *Injection* Klasik. Untuk menghasilkan nilai *true* dalam *query*	*Fail*	*Fail*	*Fail*	*Pass*	*Fail*	*Pass*
3	SQL *Injection* Klasik, *password* terenkripsi. Untuk menghasilkan nilai dalam *query* dengan menjadikan *field password* sebagai komen	*Fail*	*Fail*	*Fail*	*Pass*	*Pass*	*Fail*
4	*Simple Select Statement*	*Fail*	*Fail*	*Fail*	*Pass*	*Pass*	*Pass*
5	Teknik *Exploitation Union Step* 1. Untuk mendapatkan jumlah kolom	*Fail*	*Fail*	*Fail*	*Pass*	*Pass*	*Pass*
6	*Teknik Exploitation Union Step* 2	*Fail*	*Fail*	*Fail*	*Pass*	*Pass*	*Pass*

Dari Tabel 6 Hasil Pengujian Aplikasi yang Terenkripsi dan Tidak Terenkripsi diketahui

bahwa pengujian SQL *Injection* secara manual, menggunakan SQL *Map* dan JSQL *Injection* membuktikan bahwa Aplikasi Terenkripsi *Blowfish* lebih aman dari serangan SQL *Injection* daripada Aplikasi yang tidak terenkripsi.

4. KESIMPULAN

Berdasarkan penelitian dan hasil pengujian yang dilakukan pada aplikasi SIUP dengan menguji keamanan *web* menggunakan SQL *Injection*, maka disimpulkan dengan sebagai berikut :

1. Algoritma *Blowfish* dapat mencegah terjadinya SQL *Injection* pada *Web*
2. Tingkat keamanan sistem yang menggunakan algoritma enkripsi *blowfish* jauh lebih aman daripada sistem yang tidak menggunakan algoritma *blowfish* (Non Enkripsi).

5. SARAN

Saran yang dapat diberikan dalam pengembangan selanjutnya adalah sebagai berikut :

1. Perlu dikembangkan dengan menggunakan jenis serangan yang lain agar dapat melihat celah suatu *web*, sehingga informasi tidak bocor atau diretas.
2. Diharapkan dapat dilakukan pengembangan pada aplikasi yang dapat mengamankan *web* dari serangan SQL *Injection* dengan menerapkan enkripsi selain *blowfish.*

DAFTAR PUSTAKA

[1] owasp.org, "OWASP Top 10 Application Security Risks - 2017," *www.owasp.org*, 2017. [Online]. Available: https://www.owasp.org/index.php/Top_10-2017_Top_10.

[2] Yulianingsih, "Menangkal Serangan SQL Injection Dengan Parameterized Query," *J. Edukasi dan Penelit. Inform.*, vol. 2, no. 1, pp. 46–49, 2017.

[3] R. Munir, *Kriptografi*. Bandung: Informatika Bandung, 2006.

[4] C. A. Sutanto, "Penggunaan Algoritma Blowfish Dalam Kriptografi," 2010.

[5] A. Yudistira, "Analisis Keamanan Otentikasi dan Basis Data Pada Web Simple-O Menggunakan SQL Injection," Universitas Indonesia, 2012.

[6] N. Afiyana, "Perancangan Aplikasi Tryout Bimbingan Belajar Berbasis Web," Akademi Manajemen Informatika dan Komputer Bina Sarana Informatika, 2014.

[7] D. M. Khairina, "Analisis Keamanan Sistem Login," *J. Inform. Mulawarman*, vol. 6, no. 2, pp. 64–67, 2011.