KAKAS UNTUK MENDETEKSI LUBANG KEAMANAN PADA APLIKASI WEB BERBASIS PHP DAN MYSQL YANG DAPAT DI EKSPLOITASI OLEH SERANGAN SQL INJECTION

Teguh Wiharko, MT.
teguhwhk@hotmail.com

ABSTRAK

Aplikasi web yang ditulis menggunakan bahasa pemrograman PHP yang menggunakan MySQL sebagai basisdata rentan terhadap serangan SQL Injection. SQL Injection merupakan jenis serangan yang mengeksploitasi kelemahan pada komponen-komponen masukan yang terdapat pada aplikasi web jika data yang dimasukan pada komponen-komponen masukan tersebut tidak divalidasi dengan benar. Bahasa pemrograman PHP memiliki fungsi mysql_real_escape_string() yang dapat dipergunakan untuk mencegah serangan SQL Injection. Fungsi mysql_real_escape_string() tersebut dapat diimplementasikan kesemua komponen masukan yang terdapat pada aplikasi web.

Akan dibangun suatu kakas yang dapat digunakan untuk melakukan pendeteksian terhadap penggunaan fungsi mysql_real_escape_string() pada semua komponen masukan yang terdapat pada aplikasi web. Kakas juga akan menampilkan informasi mengenai total komponen masukan yang terdapat pada aplikasi web, berapa banyak komponen masukan yang telah terproteksi oleh fungsi mysql_real_escape_string(), serta seberapa banyak masukan yang belum terproteksi dan rentan terhadap serangan SQL injection.

Kata Kunci : Lubang Keamanan, Aplikasi Web, PHP, SQL Injection

A. PENDAHULUAN

Latar Belakang Masalah

Mengamankan sistem perangkat lunak berbasis *web* merupakan tantangan bagi setiap pengembang aplikasi. Aplikasi *web* dituntut untuk tidak hanya dapat berjalan sesuai rancangan, tetapi juga harus dapat digunakan seaman mungkin.

Aplikasi *web* yang tidak dapat memenuhi unsur keamanan, hampir dipastikan tidak akan dapat digunakan sebagaimana mestinya.

Salah satu teknik penyerangan terhadap aplikasi *web* yang umum adalah serangan *SQL Injection*, dimana penyerang akan memasukan kode-kode SQL yang telah dimodifikasi ke dalam komponen-komponen masukan pada suatu aplikasi *web*. Tingkat kerusakan yang terjadi akibat serangan *SQL Injection* sangat bervariasi, dimulai dari pencurian data, perusakan data, atau bahkan penyerang dapat membuat aplikasi web tersebut tidak dapat melayani permintaan dari *client*.

Oleh sebab itu diperlukan kakas yang dapat mendeteksi kelemahan setiap komponen masukan pada aplikasi *web*. Kakas tersebut harus dapat memberikan hasil analisis kepada pengguna, dimana letak kelemahan pengamanan komponen masukan serta persentasi komponen-komponen yang tidak aman dibandingkan dengan jumlah komponen masukan keseluruhan.

Rumusan Masalah

Berdasarkan kepada latar belakang masalah di atas, maka masalah yang akan dibahas adalah bagaimana merancang kakas yang dapat mendeteksi lubang keamanan terhadap jenis serangan *SQL Injection* pada aplikasi *web* yang ditulis menggunakan bahasa pemograman PHP?

Tujuan Penelitian

Tujuan dari penelitian ini adalah :

(1) Mengembangkan kakas untuk mendeteksi lubang keamanan terhadap *SQL Injection* pada aplikasi *web*.

(2) Mengetahui prosedur pencegahan terhadap serangan *SQL Injection* pada aplikasi berbasis *web*.

Batasan Masalah

Batasan masalah pada penelitian ini adalah:

(1) Kakas hanya dapat mendeteksi aplikasi *web* yang ditulis menggunakan bahasa pemograman PHP.

(2) Basisdata yang digunakan oleh aplikasi *web* yang dapat dideteksi oleh kakas hanya MySQL.

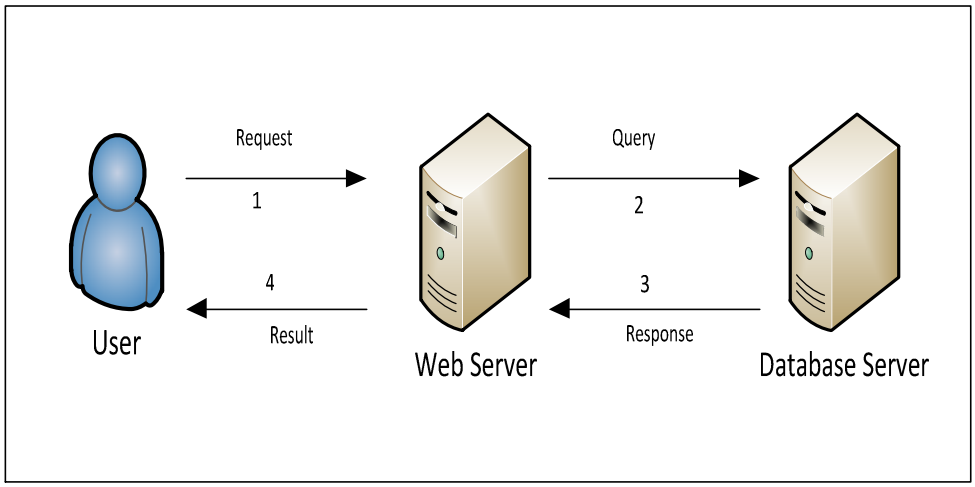
(3) Situs yang dapat diperiksa adalah situs pada domain lokal sebelum situs tersebut di-*hosting* pada penyedia layanan *hosting* internet.

B. TINJAUAN PUSTAKA

Aplikasi Web

Aplikasi *web* merupakan jenis aplikasi komputer yang berjalan di atas infrastruktur internet. Aplikasi *web* berjalan pada komputer *server* yang menjalankan aplikasi *web server* dan akan melayani permintaan klien yang meminta layanan menggunakan *browser*. Keuntungan menggunakan aplikasi *web* adalah sifatnya yang *cross-platform* dan kemampuannya dalam melakukan distribusi ke ribuan komputer klien tanpa instalasi aplikasi. Sebuah aplikasi *web* adalah suatu sistem perangkat lunak yang berbasiskan teknologi dan standar dari konsorsium *world wide web* (W3C) yang menyediakan sumber yang bersifat spesifik seperti konten atau layanan melalui sebuah *user interface* yang disebut *web browser* [8].

Aplikasi *web* dapat ditulis dalam beberapa bahasa pemograman yang mendukung seperti: HTML (*Hypertext Markup Language*), PHP (*PHP: Hypertext Preprocessor*), ASP (*Active Server Page*), CGI (*Common Gateway Interface*), Perl, Phyton dan Java.



Gambar 1. Aliran informasi pada aplikasi *web* [3].

Aliran informasi seperti pada gambar 1, pada aplikasi berbasis *web* dinamis adalah sebagai berikut [3]:

(1) Pengguna (*user*) mengirimkan permintaan ke *web server*.

(2) *Web server* mendapatkan data pengguna, membuat sebuah perintah SQL sesuai permintaan pengguna dan mengirimkannya ke *server* basisdata.

(3) *Server* basisdata akan mengeksekusi perintah SQL dan mengembalikan hasilnya ke *web server*.

(4) *Web server* kemudian akan secara dinamis membuat halaman HTML berdasar kepada respon dari basisdata.

SQL Injection

SQL Injection merupakan jenis serangan dimana kode SQL (*Structured Query Language*) disisipkan atau ditambahkan ke dalam parameter masukan aplikasi/pengguna yang nantinya akan diteruskan ke *server* SQL untuk dilakukan parsing dan dieksekusi. Bentuk serangan pada *SQL Injection* terdiri dari penyisipan kode secara langsung ke dalam parameter yang kemudian akan digabungkan dengan perintah-perintah SQL dan kemudian dieksekusi.

Ketika seorang penyerang dapat melakukan modifikasi suatu perintah SQL, perintah tersebut akan dieksekusi dengan hak akses yang sama dengan pengguna aplikasi yang sebenarnya, yang terkadang memiliki hak akses/otoritas yang tinggi dalam sistem tersebut. Serangan *SQL Injection* dapat terjadi dikarenakan pengembang aplikasi *web* tidak melakukan validasi terhadap nilai yang diterima dari form *web*, *cookie*, parameter masukan, dan sebagainya sebelum diteruskan ke *query* SQL yang kemudian akan dieksekusi oleh *server* basisdata.

Salah satu contoh serangan *SQL Injection* yang mungkin terjadi adalah [4]:

(1) Terdapat perintah SQL berikut

```
Select id, forename, surname from authors
```

(2) Perintah tersebut akan mengambil data ‘id’, ‘*forename’*, dan ‘*surname’* dari tabel ‘*authors’*. Untuk hasil pencarian yang lebih spesifik dapat juga digunakan perintah SQL berikut

```
Select id, forename, surname from authors where
forename = ‘john’ and surname = ‘smith’
```

(3) Serangan *SQL Injection* dapat dilakukan misalnya dengan memasukan variabel sebagai berikut

```
Forename: jo’; drop table authors--
Surname:
```

(4) Dengan masukan seperti di atas maka perintah SQL *select* akan dihentikan prosesnya oleh karakter ‘;’ dan dilanjutkan dengan eksekusi perintah SQL berikutnya yaitu ‘*drop table authors*--' yang akan menghapus tabel *authors* dari basisdata.

Cara yang dapat dilakukan untuk pencegahan serangan *SQL Injection* adalah dengan melakukan validasi masukan dari pengguna. Validasi data yang dapat dilakukan sebagai berikut [4]:

(1) Berusaha untuk membuat data masukan menjadi valid.

Misalnya dengan menghilangkan karakter kutip satu dari masukan.

```
function escape(input)
  input = replace(input,"'","''")
  escape = input
end function
```

Fungsi *escape* di atas digunakan untuk menghilangkan karakter kutip satu dari masukan.

(2) Menolak masukan yang diketahui berpotensi digunakan dalam serangan.

Misalnya dengan mendefinisikan kata-kata yang diketahui dapat digunakan dalam serangan *SQL Injection*.

```
function validate_string( input )
  known_bad = array( "select", "insert",
     "update", "delete", "drop",
     "--", "'" )
  validate_string = true
  for i = lbound( known_bad ) to ubound(
known_bad )
    if ( instr( 1, input, known_bad(i),
vbtextcompare
        ) <> 0 ) then
       validate_string = false
       exit function
    end if
  next
end function
```

Fungsi *validate_string* di atas digunakan untuk menolak masukan yang tidak baik.

(3) Hanya menerima masukan yang baik.

Dengan mendefinisikan karakter atau kata yang valid saja.

```
function validatepassword( input )
  good_password_chars =
"abcdefghijklmnopqrstuvwxyz
    ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
  validatepassword = true
  for i = 1 to len( input )
    c = mid( input, i, 1 )
    if ( InStr( good_password_chars, c ) = 0 )
then
      validatepassword = false
      exit function
    end if
   next
end function
```

Fungsi *validate_password* di atas digunakan untuk hanya memperbolehkan masukan yang valid.

Pencegahan Terhadap SQL Injection

Tabel 1. Klasifikasi dan Pemetaan Masalah Keamanan[7].

	Aspek Keamanan dan Integritas Sistem *Web*				
	Access Control	***Confidentiality***	***Message Integrity***	**Serangan**	
				SQL Injection	**XSS**
Jaringan	Konfigurasi hak akses pengguna jaringan	Secure Socket Layer (SSL)	SSL	-	-
Basisdata	Pada level Basisdata	Enkripsi	Fungsi Hash	-	-
Aplikasi *Web*	Fungsi *access control* pada kode program	Enkripsi	*Digital Signature*	Fungsi validasi data masukan	- Validasi data masukan - Validasi data keluaran

Pada Tabel 1 terlihat bahwa serangan *SQL Injection* merupakan salah satu jenis serangan yang mengeksploitasi kelemahan di bagian masukan data pada aplikasi *web*. Serangan *SQL Injection* yang berhasil dilakukan dapat mengakibatkan kerusakan pada aplikasi *web* dengan tingkat kerusakan yang beragam. Untuk menentukan jenis pengamanan terhadap serangan *SQL Injection*, maka dilakukan analisis terhadap teknik proteksi terhadap kedua jenis serangan tersebut[7].

Tabel 2. Teknik proteksi terhadap *SQL Injection* pada PHP[7].

Teknik Proteksi	***SQL Injection***
Fungsi validasi data masukan	✔
Fungsi *encoding* data keluaran	-

Fungsi validasi terhadap semua data masukan dapat digunakan untuk melakukan proteksi terhadap serangan *SQL Injection*. Kelemahan dari penggunaan fungsi validasi data masukan adalah mengurangi kemampuan dari aplikasi *web* untuk menerima semua variasi masukan[7].

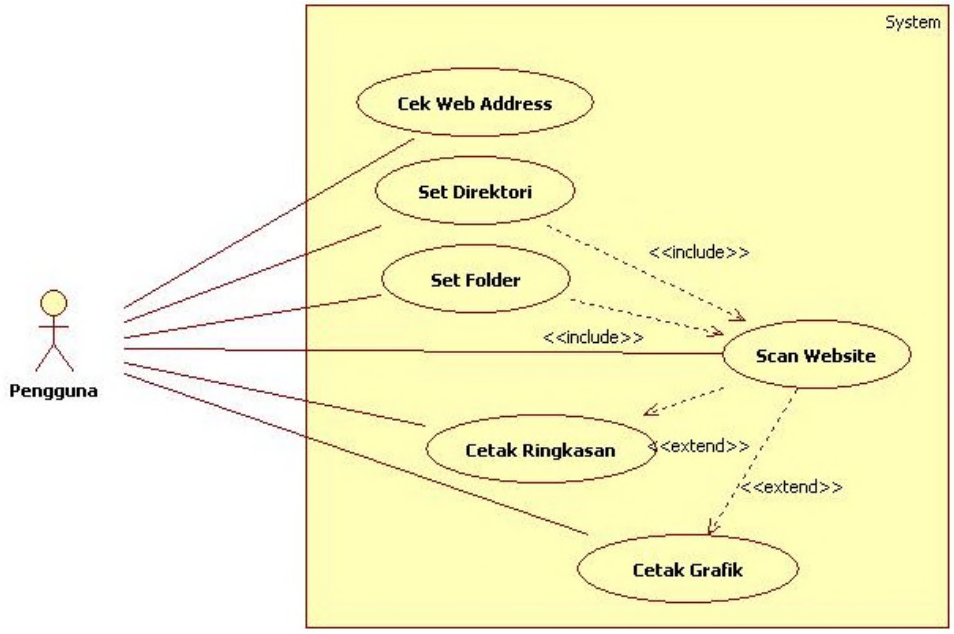
Pada bahasa pemrograman PHP yang menggunakan MySQL sebagai basisdatanya, terdapat fungsi *mysql_real_escape_string*() yang bertujuan sebagai fungsi validasi data masukan. Walaupun demikian, pengembang aplikasi juga dapat membuat sendiri fungsi validasi sendiri. Fungsi *mysql_real_escape_string*() akan digunakan untuk memproteksi semua komponen masukan yang ada pada aplikasi *web*. Contoh penggunaan dari fungsi *mysql_real_escape_string*() pada tiga jenis masukan pada aplikasi *web* adalah[7]:

```
mysql_real_escape_string($_POST[variabel_masukan]);
mysql_real_escape_string($_GET[variabel_masukan]);
mysql_real_escape_string($_REQUEST[variabel_masukan]);
mysql_real_escape_string($_SESSION[variabel_masukan]);
```

C. ANALISIS DAN PERANCANGAN

Pada tahap analisis, *Use case diagram* digunakan sebagai alat bantu menentukan scenario penggunaan dari kakas *SQL Injection Scanner* yang akan dibangun. Sedangkan pada tahap perancangan, *class* diagram digunakan untuk menentukan berapa banyak *class* pembentuk serta hubungan antar *class* pada sistem yang akan dibangun. *Sequence diagram* digunakan untuk menentukan aliran proses pada sistem yang akan dibangun yang berkaitan dengan waktu proses akan dilaksanakan.

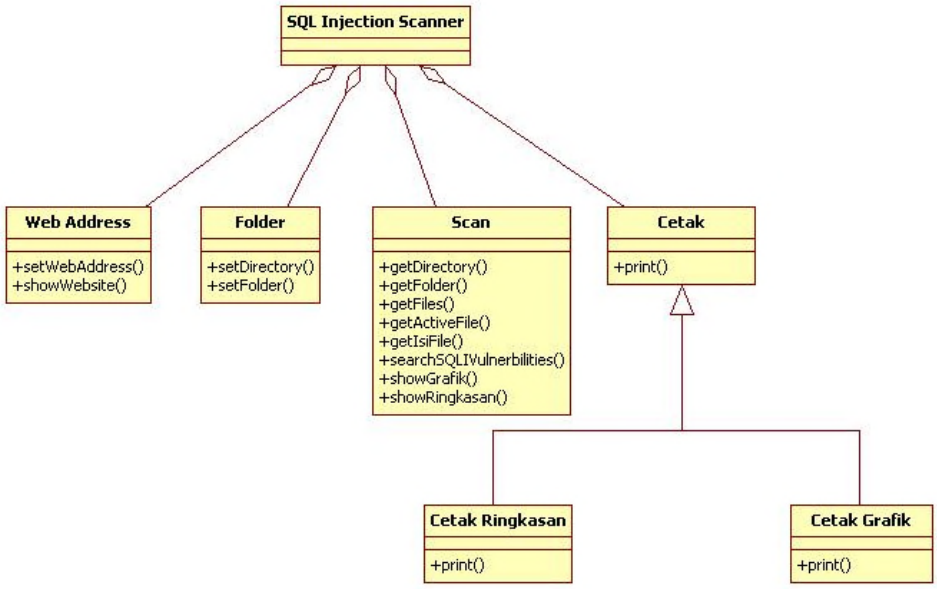
Use Case Diagram



Gambar 2. ***Use case diagram*** **kakas** ***SQL Injection Scanner*****.**

Berikut ini skenario yang akan digunakan pada proses pengembangan kakas *SQL Injection Scanner*. Pengguna kakas akan melakukan pengecekan terhadap aplikasi *web* untuk melihat apakah aplikasi *web* tersebut aktif atau tidak. Jika aplikasi *web* tersebut aktif, maka pengguna kakas dapat melakukan setting terhadap direktori dan *folder* tempat kode program aplikasi tersebut berada. Setelah itu, pengguna kakas dapat melakukan pemeriksaan terhadap semua file pada *folder* aktif yang telah dipilih. Hasil dari pemeriksaan terhadap kelemahan aplikasi *web* tersebut dapat dicetak ke *printer*, baik berupa cetakan teks maupun grafik.

Class Diagram

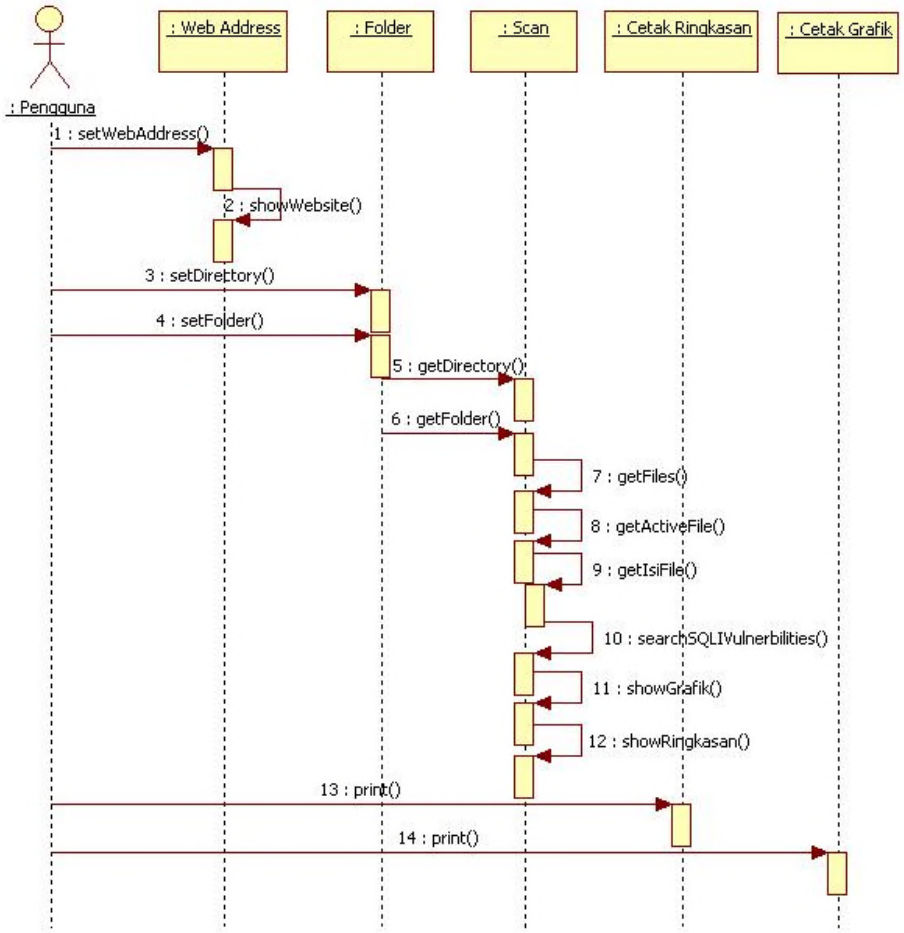


Gambar 3. Class diagram kakas *SQL Injection Scanner*.

Class diagram pada gambar 3 (tiga) di atas menggunakan pola hubungan agregasi dimana kakas *SQL Injection Scanner* terdiri dari beberapa kelas pembentuk, yaitu: (1) *Class* Web Address; (2) *Class* Folder; (3) *Class* Scan; dan (4) *Class* Cetak. Sedangkan *Class* Cetak di turunkan menjadi dua *Class* turunan, yaitu: (1) *Class* Cetak Ringkasan; dan (2) *Class* Cetak Grafik.

Class Web Address terdiri dari metode: (1) setWebAddress(); dan (2) showWebsite().*Class Folder* terdiri dari metode: (1) setDirectory(); dan (2) setFolder().*Class Scan* terdiri dari metode: (1) getDirectory(); (2) getFolder(); (3) getFiles(); (4) getActiveFile(); (5) getIsiFile(); (6) searchSQLIVulnerabilities(); (7) showGrafik(); dan (8) showRingkasan. *Class* Cetak Ringkasan dan *Class* Cetak Grafik memiliki masing-masing satu metode print().

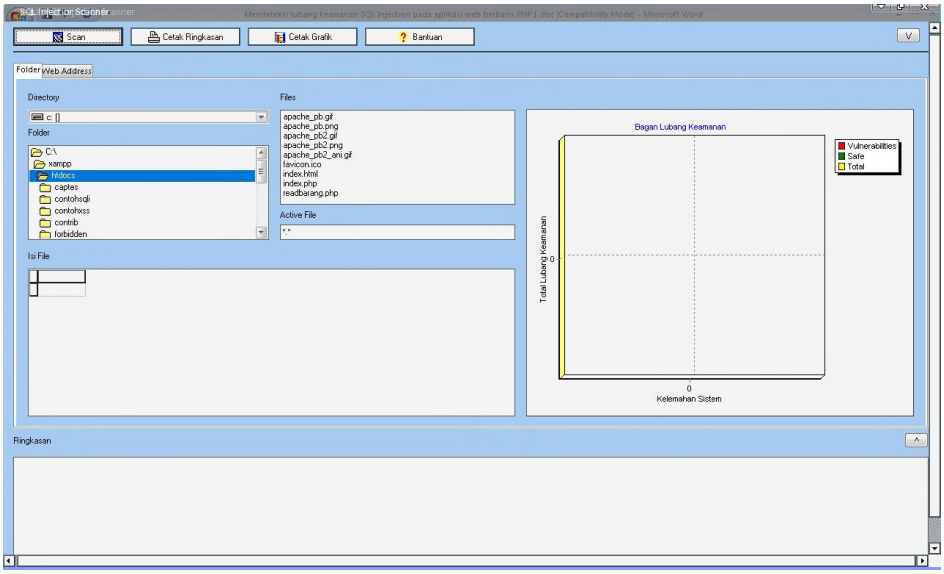
Sequence Diagram



Gambar 4. *Sequence diagram* kakas *SQL Injection Scanner*.

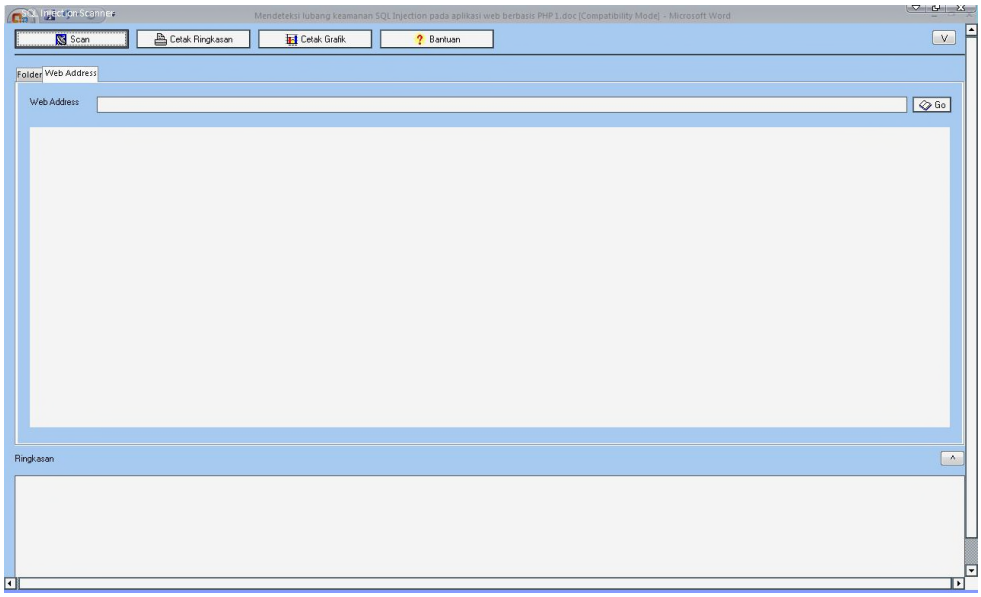
Sequence diagram pada gambar 4, menggambarkan aliran proses pada kakas sesuai dengan *use case diagram* yang telah dibuat. Data masukan, proses, dan keluaran pada *sequence diagram* merupakan penjabaran lebih detail dari *use case diagram*.

D. IMPLEMENTASI



Gambar 5. Antarmuka kakas *SQL Injection Scanner*.

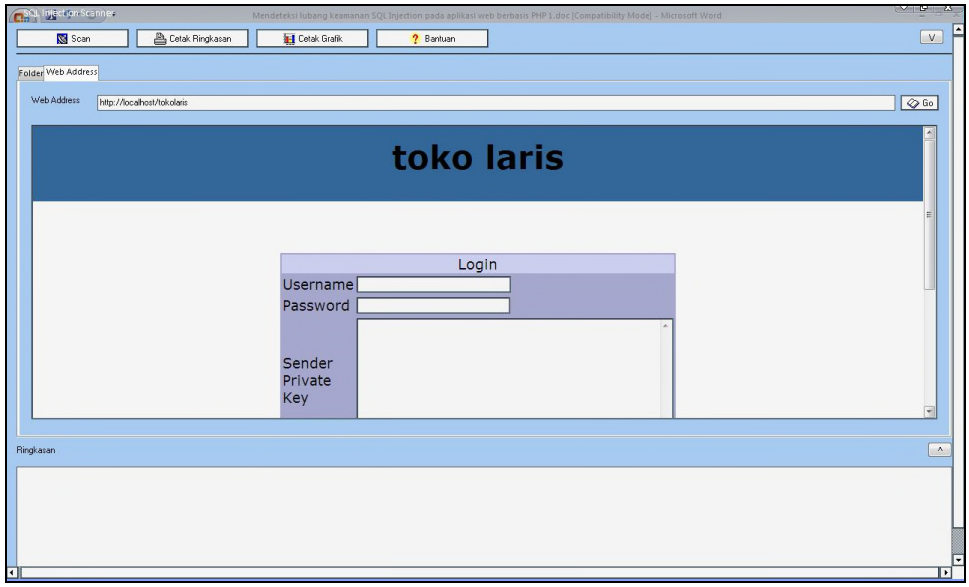
Antarmuka kakas *SQL Injection Scanner* dibuat sesederhana mungkin untuk memudahkan proses penggunaannya. Antarmuka penggunanya terdiri dari bagian tombol-tombol utama, yang terdiri dari: tombol *Scan* untuk mulai melakukan proses pendeteksian kelemahan sistem, tombol Cetak Ringkasan untuk mencetak data hasil pendeteksian dalam bentuk teks, tombol Cetak Grafik untuk mencetak data hasil pendeteksian dalam bentuk grafik, serta tombol *Help* yang digunakan untuk menampilkan jendela bantuan.



Gambar 6. Antarmuka *web browser* pada kakas *SQL Injection Scanner*.

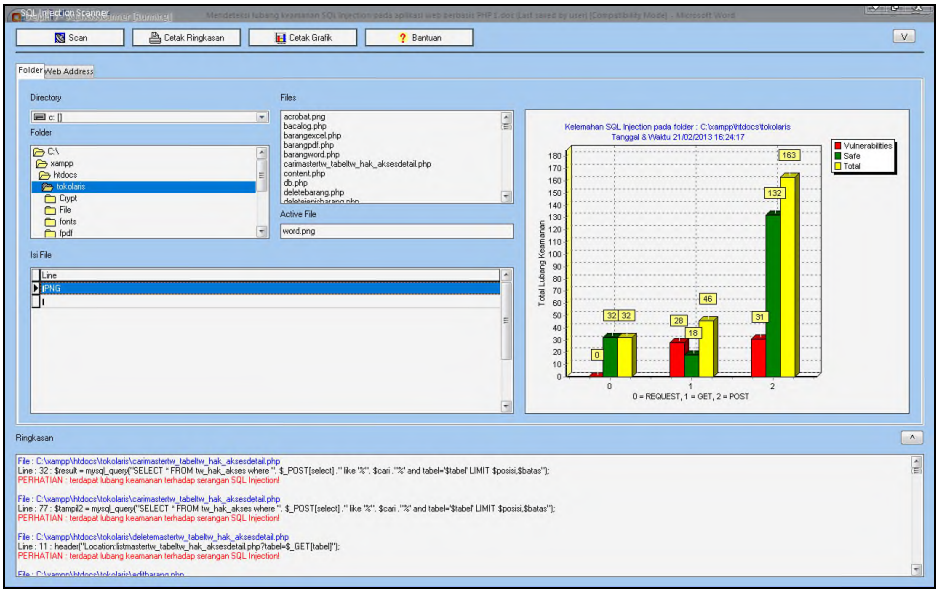
Antarmuka utama terdiri dari 2 (dua) tab, yaitu tab *Folder* dan tab *Web Address*. Tab *Folder* terdiri dari dua komponen masukan, yaitu kotak pilihan direktori dan kotak pilihan *folder*. Tab *web address* terdiri dari kotak teks masukan alamat aplikasi *web*, tombol *Go* untuk mulai mengakses aplikasi *web* yang ditulis pada kota teks masukan alamat aplikasi, serta bagian *web browser* yang digunakan untuk menampilkan aplikasi *web*.

E. EVALUASI KAKAS



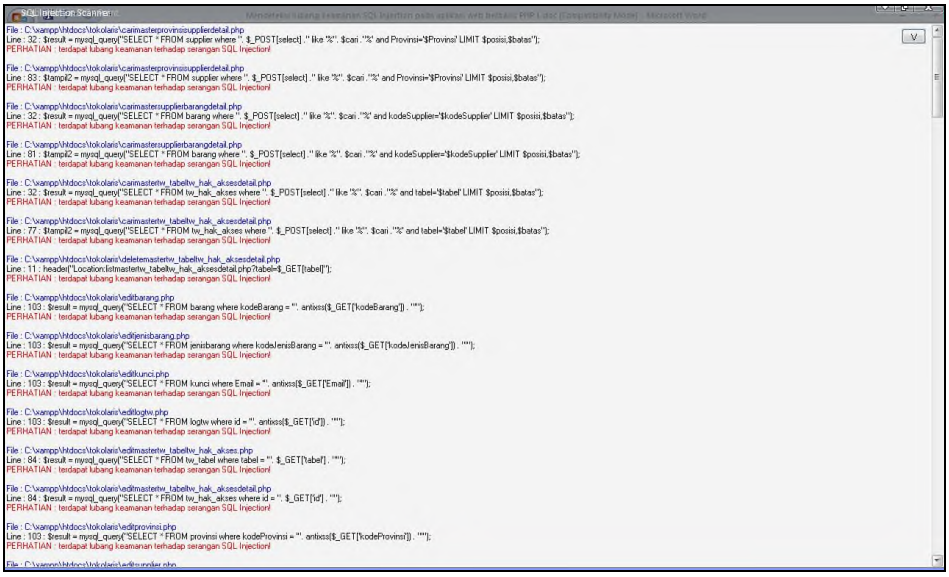
Gambar 7. Cek ketersediaan layanan (*website availability*) situs internet yang akan diuji oleh kakas *SQL Injection Scanner*.

Gambar 7, memperlihatkan kakas *SQL Injection Scanner* yang sedang melakukan akses terhadap aplikasi *web* "toko laris" yang disimpan pada domain lokal "http://localhost/toko laris". Aplikasi *web* "toko laris" ditampilkan pada bagian *web browser* untuk memastikan bahwa aplikasi *web* tersebut aktif dan dapat diakses.



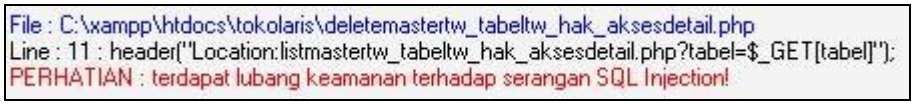
Gambar 8. Kakas *SQL Injection Scanner*.

Setelah tombol *Scan* ditekan, maka proses pendeteksian kelemahan sistem pada aplikasi *web* "toko laris" dimulai dan hasilnya ditampilkan pada bagian grafik dan bagian ringkasan teks. Daftar file pada *folder* aktif ditampilkan pada bagian File, sedangkan nama file yang sedang dideteksi ditampilkan pada bagian *Active File*. Isi dari setiap file aktif yang sedang dideteksi, ditampilkan pada bagian Isi File.



Gambar 9. Contoh gabungan pesan peringatan terhadap adanya kelemahan pada situs internet yang telah diuji menggunakan kakas *SQL Injection Scanner*.

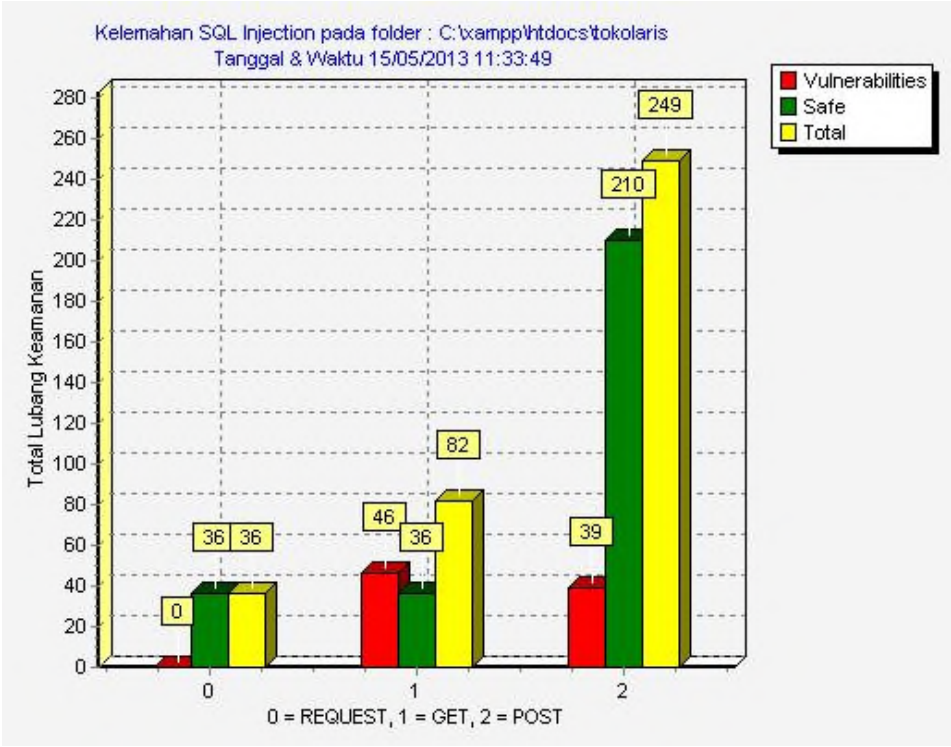
Gambar 10. Contoh pesan peringatan terhadap kelemahan pada metode pengiriman data POST.



Gambar 11. Contoh pesan peringatan terhadap kelemahan pada metode pengiriman data GET.

Ringkasan teks dapat digunakan untuk menganalisis data kelemahan sistem terhadap serangan *SQL Injection* dengan lebih detail. Setiap kelemahan yang

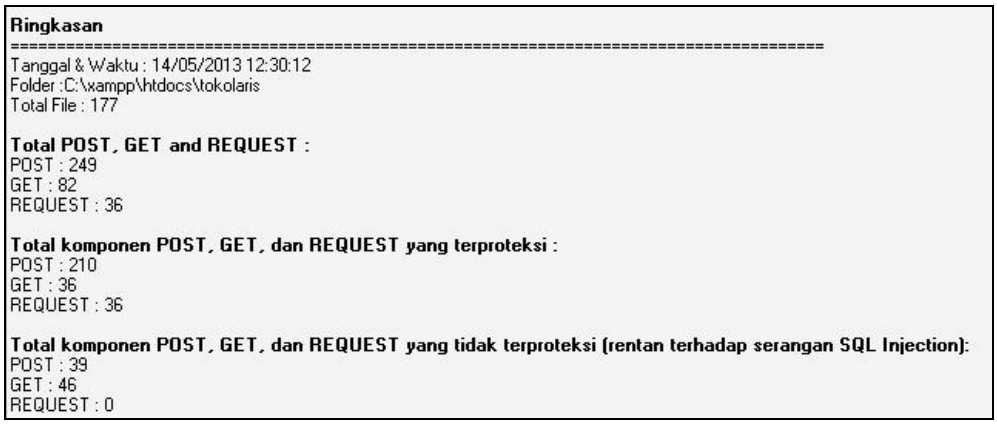
ditemukan akan disajikan dalam 3 (tiga) bagian, yaitu *File, Line*, dan PERHATIAN. Bagian *File* menunjukan posisi *file* yang dideteksi. Bagian *Line* menunjukan baris keberapa dari kode program yang memiliki kelemahan. Bagian PERHATIAN digunakan untuk menarik perhatian dari pengguna mengenai adanya kelemahan sistem pada file dan baris di atasnya.



Gambar 12. Ringkasan Grafik hasil pendeteksian kelemahan sistem.

Hasil kelemahan aplikasi *web* secara umum dapat dibaca dari ringkasan grafik. Terdapat tiga komponen masukan yang dideteksi oleh kakas *SQL Injection Scanner*, yaitu komponen masukan yang mempergunakan metode REQUEST, GET, dan POST. Masing-masing metode akan ditampilkan dalam bentuk batang dengan warna yang berbeda. Batang berwarna merah menunjukan jumlah

komponen masukan yang disinyalir memiliki kelemahan terhadap serangan *SQL Injection*. Batang berwarna hijau menunjukan jumlah komponen masukan yang aman, dan batang berwarna kuning menunjukan jumlah komulatif komponen masukan.



Gambar 13. Ringkasan hasil pengujian oleh kakas *SQL Injection Scanner*.

Ringkasan teks merupakan bentuk lain dari grafik, yang menyatakan bahwa proses pendeteksian pada aplikasi *web* telah selesai. Waktu deteksi, *folder* aplikasi, dan total file yang dideteksi akan dipresentasikan. Jumlah masing-masing komponen masukan dari dengan metode yang berbeda juga dipresentasikan.

F. KESIMPULAN DAN SARAN

Kesimpulan

Penggunaan kakas *SQL Injection Scanner* ternyata dapat digunakan untuk melakukan pendeteksian terhadap kelemahan sistem *web* yang ditulis menggunakan bahasa pemograman PHP yang menggunakan basisdata MySQL. Jenis serangan yang dapat dideteksi oleh kakas *SQL Injection Scanner* ini adalah serangan *SQL* (*Structured Query Language) Injection* yang merupakan salah satu

jenis serangan populer yang ditujukan untuk menyerang aplikasi *web*/situs internet.

Kakas *SQL Injection Scanner* dapat mempercepat dan mempermudah pendeteksian kelemahan aplikasi *web* terhadap serangan *SQL Injection*. Kecepatan pendeteksian menggunakan kakas *SQL Injection Scanner* jauh lebih cepat dibandingkan jika pendeteksian dilakukan secara manual, satu persatu pada puluhan/ratusan/ribuan komponen masukan pada aplikasi *web*.

Saran

Kakas *SQL Injection Scanner* ini dapat dikembangkan untuk melakukan penetrasi langsung ke dalam aplikasi *web* dengan cara melakukan serangan *SQL Injection*. Dengan cara tersebut, maka kelemahan sistem yang dapat dideteksi bukan hanya kelemahan sistem yang berasal dari kode program PHP saja tetapi juga kelemahan sistem yang berasal dari basisdata yang digunakan, *web server* atau komponen dari pihak ketiga.

Kakas *SQL Injection Scanner* ini dapat dikembangkan untuk mendeteksi kelemahan aplikasi *web* yang ditulis bukan hanya menggunakan bahasa pemrograman PHP saja. *Active Server Pages* (ASP) yang mengakses basisdata *Microsoft SQL Server* merupakan salah satu bahasa pemrograman dan basisdata *server* yang juga banyak digunakan untuk menulis aplikasi *web*, direkomendasikan juga untuk dapat didukung oleh kakas *SQL Injection Scanner* ini.

REFERENSI

http://www.php.net/manual/en/security.database.sql-injection.php, diakses pada 13 Maret 2013.

Suprianto, Dodit (2008) : Buku Pintar Pemograman PHP, *OASE Media Bandung*.

Clarke, Justin (2009) : SQL Injection attacks and defense, *Syngress*.

Anley, Chris (2002) : Advanced SQL Injection in SQL Server Applications, *NGSSoftware Insight Security Research (NISR) Publication*.

http://www.mysql.com/products/enterprise/techspec.html, diakses pada 13 Maret 2013.

Hakim, Lukmanul (2009) : Trik Rahasia Master PHP Terbongkar Lagi, *Lokomedia Yogyakarta*.

Wiharko, Teguh (2012) : Application Generator Untuk Meningkatkan Kualitas Keamanan Dan Integritas Pada Aplikasi Basisdata Berbasis Web, *Tesis, Institut Teknologi Bandung*.

Adri, Muhammad (2008) : Konsep Dasar Web Engineering, Modul Pemograman Web, *Universitas Negeri Padang*.

Fowler, Martin (2005) : UML Distilled Edisi 3, Panduan Singkat Bahasa.