Analisis keamanan menggunakan web aplikasi bwapp terhadap serangan XSS (Cross Site Scripting) dan SQL Injection

Reza Rafsanjani Prayogo (1410652019) [1] , *Victor Wahanggara, S. Kom*[2]

Jurusan Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Jember

rreza.prayogo@gmail.com

ABSTRAK

Perkembangan internet yang pesat dijadikan sebagai media dan sumber informasi dengan kenyamanan akses internet ini, keamanan merupakan salah satu faktor penting yang harus diperhatikan dalam membangun sebuah website oleh pengembang web. SQL Injection adalah kerentanan yang terjadi ketika penyerang memiliki kemampuan untuk mempengaruhi SQL yang melewati suatu aplikasi ke database back-end. XSS (Cross Site Scripting) merupakan serangan dengan cara memasukan kode html atau client script code lainnya. metode yang akan penulis gunakan yaitu simulasi SQL Injection dan XSS (Cross Site Scripting) yang nantinya dapat menjadi pembelajaran bagi developer website untuk lebih mengedepankan aspek keamanan. Dalam uji coba ini dapat disimpulkan bahwa menganalisis celah keamanan yang ada pada web pada simulasi SQL Injection dan XSS (Cross Site Scripting)

Kata kunci : *Cross Site Scripting , keamanan web , SQL injection*

Analysis Web Application Security Using Bwapp Against XSS (Cross Site Scripting) And SQL Injection Attack

Reza Rafsanjani Prayogo (1410652019) [1] , *Victor Wahanggara, S. Kom*[2]

Departement of Informatics Faculty of Engineering,

University of Muhammadiyah Jember

rreza.prayogo@gmail.com

ABSTRACT

Fast development of Internet media and serve as resources to the convenience of this internet access, security is one of the important factors that should be considered in building a website by web developers. SQL Injection is a vulnerability that occurs when an attacker has the ability to influence the SQL that passes through an application to the back-end database. XSS (Cross Site Scripting) is an attack by entering the client script code html or other code. The method to be the authors use the simulation SQL Injection and XSS (Cross Site Scripting), which will be a lesson learned for website developers to more advanced aspects of security. In this trial can be concluded that analyzing the vulnerabilities that exist on the web at simulation SQL Injection and XSS (Cross Site Scripting)

Keyword : *XSS ,Cross Site Scripting, web security, SQL Injection*

BAB I

PENDAHULUAN

1.1. Latar Belakang

Internet sebagai jaringan komunikasi global dapat dijadikan sebagai media dan sumber informasi terkini, seperti ilmu pengetahuan, teknologi, hiburan, bisnis dan sumber informasi lainnya. Kemudahan serta kenyamanan seperti ini menyebabkan internet selalu digunakan. Namun dibalik kemudahan dan kenyamanan internet, ternyata ada satu aspek yang saat ini masih kurang diperhatikan oleh pengguna internet, yaitu keamanan yang merupakan salah satu aspek penting pada aplikasi web. Sampai saat ini tidak ada website yang dapat dikatakan benar-benar aman.

SQL Injection adalah kerentanan yang terjadi ketika penyerang memiliki kemampuan untuk mempengaruhi *Structured Query Language* (SQL) *Query* yang melewati suatu aplikasi ke database back-end.dengan mampu mempengaruhi apa yang akan diteruskan ke database, penyerang dapat memanfaatkan sintaks dan kemampuan dari SQL, serta kekuatan dan fleksibilitas untuk mendukung operasi dan fungsionalitas sistem yang tersedia ke database. Injeksi SQL bukan merupakan kerentanan yang eksklusif mempengaruhi aplikasi web, kode yang menerima masukan dari sumber yang tidak dipercaya dan kemudian menggunakan input yang membentuk SQL dinamis bisa rentan. Kasus SQL Injection terjadi ketika penyerang dapat memasukan serangkaian pernyataan SQL ke query dengan memanipulasi data input ke aplikasi.

Berdasarkan penjelasan diatas dapat dikatakan bahwa serangan SQL Injection sangat berbahaya karena penyerangan yang telah berhasil memasuki database sistem dapat melakukan manipulasi data yang ada pada database sistem. Proses manipulasi data yang tidak semestinya oleh penyerang dapat menimbulkan kerugian bagi pemilik website yang terinjeksi kebocoran data dan informasi merupakan hal yang fatal. Data-data tersebut dapat disalahgunakan oleh pihak yang tidak bertanggung jawab.

Keamanan data dan informasi menjadi aspek penting dalam menjaga ketahanan website. Oleh karena itu penulis mencoba membuat simulasi pencegahan dilakukan untuk menguji sebuah website. Berdasarkan latar belakang diatas penulis berniat untuk mengambil tema tentang keamanan website yang berjudul "ANALISIS KEAMANAN MENGGUNAKAN WEB APLIKASI BWAPP TERHADAP SERANGAN XSS DAN SQL INJECTION".

1.2. Rumusan Masalah

Berdasarkan pada latar belakang yang dijelaskan sebelumnya maka rumusan masalah dalam penelitian ini adalah sebagai berikut :

1. Bagaimana Simulasi serangan XSS (Cross Site Scripting)
2. Bagaimana simulasi serangan SQL Injection berjalan
3. Pengujian simulasi serangan XSS dan SQL Injection
4. Pencegahan terhadap serangan XSS dan SQL Injection

1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Metode serangan yang dipakai pada serangan XSS adalah reflected GET
2. Metode serangan yang dipakai pada serangan SQL Injection adalah SQL Injection Get/Search
3. Simulasi disini menggunakan web aplikasi bwapp

1.4. Tujuan

Tujuan yang ingin dicapai penulis adalah

1. Mengetahui bagaimana mekanisme simulasi serangan yang terjadi pada aplikasi web.
2. Menganalisa web yang terserang XSS dan SQL Injection
3. Bagaimana mencegah web yang terserang SQL Injection
4. Bagaimana mencegah web yang terserang XSS (Cross Site Scripting)

1.5. Manfaat

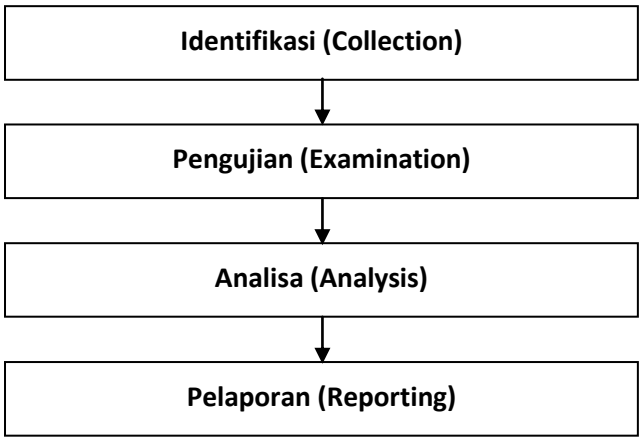
Manfaat yang didapat dari penelitian ini adalah

1. Dapat mengetahui mekanisme serangan SQL injection
2. Dapat mengetahui mekanisme serangan XSS (Cross Site Scripting)
3. Dapat mengetahui langkah atau tindakan pencegahan berdasarkan analisa keamanan suatu website

BAB III
METODOLOGI PENELITIAN

3.1. Metode atau teknik pengerjaan

Metode atau teknik pengerjaan tugas akhir ini adalah dengan :



a. Identifikasi (Collection)

   Pada tahap ini dilakukan identifikasi terhadap kebutuhan, tahapan identifikasi ini penulis berhasil mengidentifikasi kebutuhan alat dan bahan, identifikasi website target yang diteliti, jangka waktu penelitian

   1. Alat dan bahan : 1 buah laptop sebagai media simulasi, sebuah web server lokal dan browser untuk melakukan serangan XSS dan SQL Injection
   2. Waktu penelitian : Mei – Juli 2016

b. Pengujian (Examination)

   Pada tahap ini mulai dilakukan serangan terhadap aplikasi web bwapp, peneliti mulai melakukan ujicoba serangan XSS dan SQL Injection.

c. Analisa (Analysis)

   Pada tahapan ini, dilakukan analisa terhadap hasil serangan XSS dan SQL Injection, hal ini berguna untuk menemukan kelemahan-

kelemahan pada aplikasi web bwapp. Berdasarkan hasil analisa diharapkan dapat diperoleh solusi untuk serangan SQL Injection.

d. Pelaporan (Reporting)

Pada tahap pelaporan, mulai dilakukan dokumentasi terhadap hasil penelitian beserta analisisnya.

3.2. Skenario penelitian

3.2.1. SQL Injection

Skenario pengujian ini dilakukan dengan memasukkan string-string di tempat input yang nantinya akan memodifikasi source code dari aplikasi berbasis web yang akan menjadi target. Berikut adalah tahapan-tahapan yang dilakukan dalam skenario pengujian ini

a. Injeksi URL

Aplikasi berbasis web yang menjadi target adalah web aplikasi bwapp yang sudah terinstall di laptop. Hal yang pertama dilakukan adalah membuka alamat web aplikasi bwapp dari browser. Setelah terbuka maka disinilah proses injeksi dijalankan. Proses yang dilakukan adalah menambahkan string-string yang dapat mengganggu jalannya web aplikasi bwapp. String-string tersebut antara lain dapat berupa tanda petik satu (‘) tanda minus (-), dsb. Penjelasan mengenai proses SQL Injection akan dijelaskan lebih rinci pada bab IV

b. Injeksi di form search

Pada saat membuka web aplikasi bwapp localhost/bwapp, maka alamat tersebut akan menampilkan login form. Login dan memilih SQL Injection Get/Search. Disinilah proses SQL Injection dapat dilakukan. Input form pada search bar dapat ditambahkan string-string yang bukan input seharusnya seperti tanda petik satu (‘), dsb. String ini yang akan memanipulasi query yang ada dalam source code web aplikasi bwapp. Proses SQL Injection akan dijelaskan lebih lengkap pada bab IV.

c. Memberi pencegahan

   Setelah web aplikasi bwapp diuji keamanannya, maka baru source code dari web ini diberi pencegahan. Pencegahan ini dimaksudkan agar proses SQL Injection ini tidak berhasil dijalankan. Pencegahan yang diberikan tergantung dari pengujian yang sudah dilakukan, seperti apakah SQL Injection yang berhasil dilakukan. Bentuk pencegahan ini bisa berupa pembatalan atau pelarangan perintah masukan input yang seharusnya tidak boleh dimasukkan, seperti string-string atau integer-integer yang dapat mengganggu source code web

d. Injeksi setelah diberi pencegahan

   Injeksi yang dilakukan sama seperti kedua point diatas, hanya saja pada proses ini aplikasi bwapp sudah diberi pencegahan. Sehingga injeksi ini sebenarnya hanya untuk menguji apakah pencegahan yang dipasang sudah aman.

3.2.2. XSS (Cross Site Scripting)

Skenario pengujian ini dilakukan dengan cara memasukkan kode HTML atau client script lainnya ke suatu web. Serangan ini seolah-olah berasal dari web resmi tersebut. Jenis serangan ini umumnya terjadi dikarenakan web aplikasi tidak melakukan validasi dan encoding terhadap input yang diberikan oleh user dan langsung mengeneratenya kembali. Berikut adalah tahapan-tahapan dalam skenario pengujian ini

a. Input URL

   Pada tahap ini memasukkan URL sebagai target serangan, disini menggunakan web app bwapp untuk melakukan simulasi serangan Cross Site Scripting (XSS), pertama kita buka browser dan masukkan alamat URL bwapp disini menggunakan localhost/bwapp, setelah terbuka proses menambahkan script kode pada web akan dilakukan. Penjelasan mengenai proses XSS (Cross Site Scripting) akan dijelaskan pada bab VI

b. Melakukan Injeksi

   Pada tahap ini disini penulis menyisipkan script kode ke dalam URL, atau pada kolom inputan gunanya adalah untuk mengetes apakah script yang disisipkan langsung dieksekusi atau tidak. Proses XSS (Cross Site Scripting) akan dijelaskan pada bab VI

c. Memberikan pencegahan

   Setelah web aplikasi bwapp diuji keamanannya, maka baru source code dari web ini diberi pencegahan. Pencegahan ini dimasudkan agar proses XSS (Cross Site Scripting) ini tidak berhasil dijalankan. Pencegahan yang diberikan tergantung dari pengujian yang sudah dilakukan, seperti apakah XSS (Cross Site Scripting) yang berhasil dilakukan

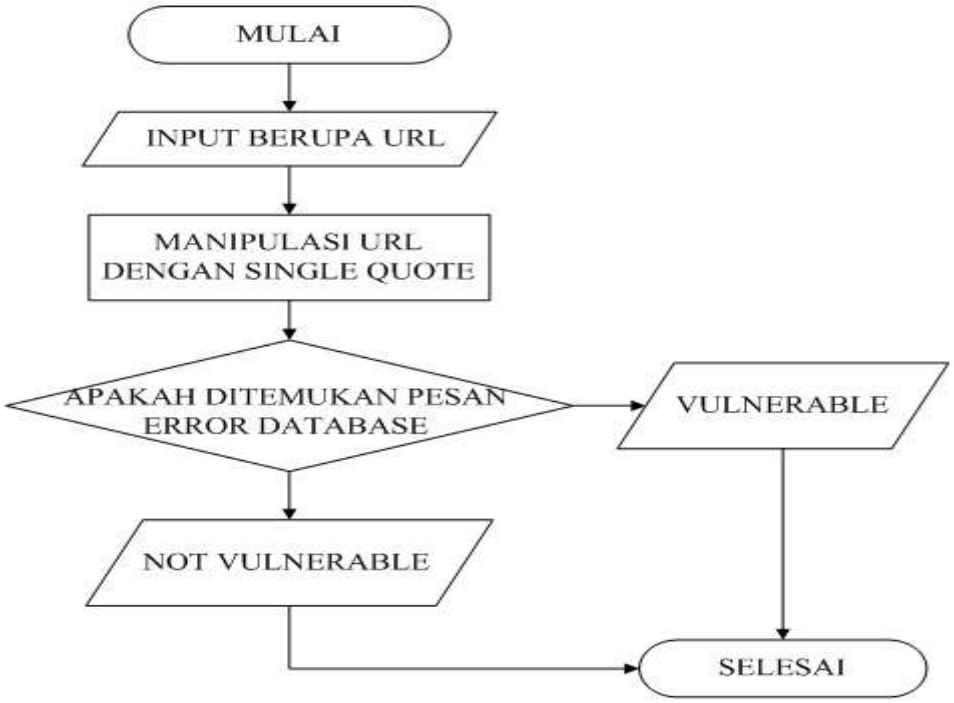
d. Injeksi setelah diberi pencegahan

   Injeksi yang dilakukan sama seperti kedua point diatas, hanya saja pada proses ini aplikasi bwapp sudah diberi pencegahan. Sehingga injeksi ini sebenarnya hanya untuk menguji apakah pencegahan yang dipasang sudah aman.

3.3. Kerangka Penelitian

3.3.1. SQL Injection

Berikut adalah diagram alir SQL Injection



Gambar 3.1. Diagram alir SQL Injection

Deskripsi :

a. Input URL

   Memasukkan URL sebagai target serangan SQL Injection

b. Manipulasi URL dengan singel quote

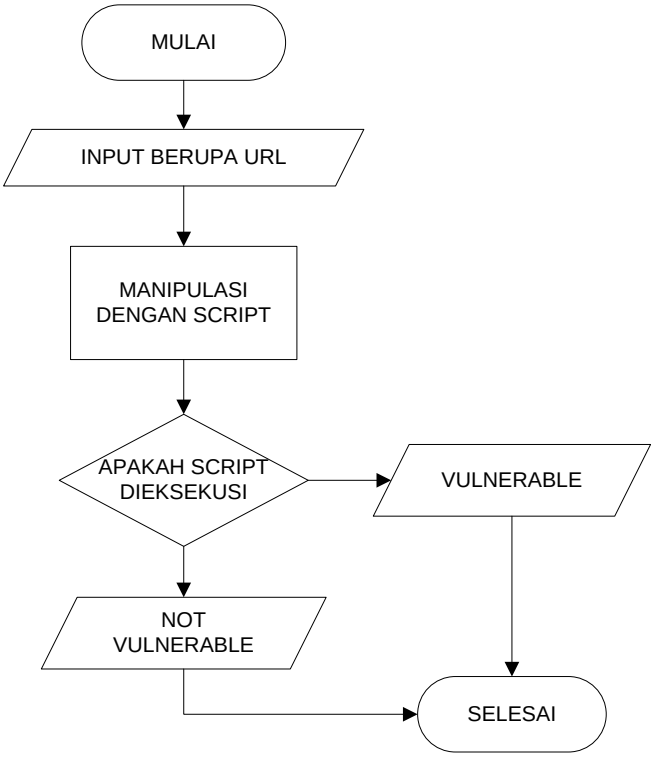
   URL dimanipulasi dengan kutip satu (‘) gunanya untuk menguji apakah tingkatan filter hanya sebatas pembatasan karakter tanda kutip satu

c. Error request

   Setelah hasil dari manipulasi URL menunjukkan error maka aplikasi web rentan / vulnerable untuk dilakukan injeksi

3.3.2. XSS (Cross Site Scripting)

Berikut adalah diagram alir XSS (Cross Site Scriping)



Gambar 3.2. Diagram alir XSS (Cross Site Scripting)

Deskripsi :

a. Input URL

   Memasukkan URL sebagai target serangan Cross Site Scripting (XSS)

b. Manipulasi URL dengan script

   Menambahkan script untuk mengetahui apakah script yang disisipkan berjalan atau tidak

c. Eksekusi script

   Setelah hasil menambahkan script dan script tersebut dijalankan maka aplikasi web rentan atau vulnerable

BAB IV
HASIL DAN PEMBAHASAN

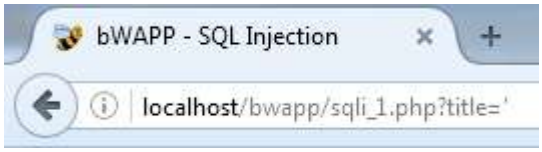
Dalam penelitian ini akan dibuat simulasi serangan, pemberian pencegahan, dan injeksi setelah diberi pencegahan pada serangan XSS dan SQL Injection. Pada bab sebelumnya sudah dijelaskan perancangan simulasi pengujian yang akan dilakukan. Pada bab ini akan dijelaskan mengenai pengujian tersebut, hasil ujicoba dan analisisnya. Pengujian ini memiliki 2 skenario yaitu serangan pertama dilakukan dengan serangan SQL Injection dan yang ke dua adalah XSS (Cross Site Scripting). Pengujian serangan SQL Injection dilakukan dengan memasukkan input berupa string-string ke dalam tempat input yang tersedia pada web bwapp, seperti URL, search bar dan sebagainya. Sedangkan pengujian serangan XSS menyisipkan kode injeksi seperti HTML atau client script lainnya ke suatu website.

4. 1. Mekanisme SQL Injection

Pada dasarnya SQL Injection merupakan cara mengeksploitasi celah keamanan yang muncul pada level atau layer database dan aplikasinya. Celah keamanan tersebut ditunjukkan berupa respon dari request data yang salah ke server yang sengaja dikirim oleh penyerang. Berikut simulasi serangan pada web app bwapp.

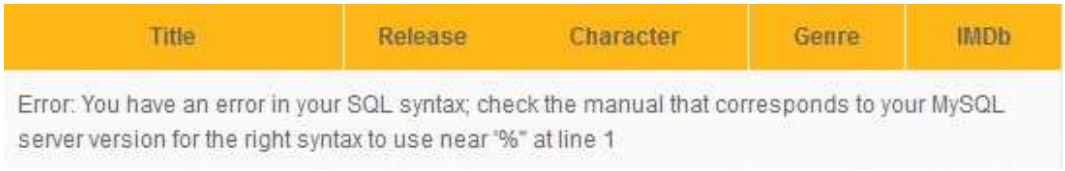
Beberapa celah kerentanan yang kerap menjadi korban SQL Injection adalah :

1. Karakter-karakter kendali, kontrol atau filter tidak didefinisikan dengan baik dan benar (*Incorrectly Filtered Escape Characters*)
2. Tipe pemilihan dan penanganan variabel maupun parameter program yang keliru (*Incorrect Type Handling*)
3. Celah keamanan berada dalam database server (*Vulnerability Inside the Database Server*)
4. Dilakukan mekanisme penyamaran SQL Injection (*Blind SQL Injection*)



Gambar 4.1. manipulasi dengan single quote

Pengujian pertama yaitu menambahkan single quote (‘) pada URL target, ini ditujukan untuk mencari tahu apakah ada celah pada website target. Berikut hasil keluaran setelah diberi single quote pada akhir URL



Gambar 4.2. hasil error setelah menambahkan single quote

Pada pengujian single quote, browser menampilkan pesan error, ini menandakan bahwa penggunaan single quote memiliki celah. Ini artinya web tersebut vulnerability. Error disini dikarenakan pada web aplikasi bwapp tidak memfilter penggunaan single quote. Setelah tahu bahwa web vulnerability lalu mencari tau nama tabel dan kolom, tapi sebelum itu harus mencari tahu pada kolom ke berapa bisa melakukan Injection, cara yang digunakan adalah dengan memodifikasi alamat URL dengan perintah “order by” , order by adalah fungsi untuk menampilkan semua data secara urut berdasarkan abjadnya, sedangkan untuk angka berfungsi untuk menguji banyaknya tabel yang dicari . perhatikan URL berikut ini

```
http://localhost/bwapp/sqli_1.php?title=1' order by 1-- -
http://localhost/bwapp/sqli_1.php?title=1' order by 2-- -
http://localhost/bwapp/sqli_1.php?title=1' order by 3-- -
http://localhost/bwapp/sqli_1.php?title=1' order by 4-- -
http://localhost/bwapp/sqli_1.php?title=1' order by 5-- -
http://localhost/bwapp/sqli_1.php?title=1' order by 6-- -
http://localhost/bwapp/sqli_1.php?title=1' order by 7-- -
http://localhost/bwapp/sqli_1.php?title=1' order by 8-- -
```

Gambar 4.3. mencari kolom yang memiliki celah SQL Injection

URL diatas jika dicoba satu persatu akan menampilkan situsnya secara normal. Tapi ketika diinputkannya dengan “order by 8-- -“ browser akan menampilkan pesan error. Dari pesan error tersebut dapat dipastikan ada 7 kolom yang dapat dilakukan injeksi. Berarti yang diambil sampai “order by 7-- -“ saja. Setelah mengetahui terdapat 7 buah kolom pada target, selanjutnya yaitu mencari kolom yang bisa dilakukan injeksi, untuk itu menggunakan fungsi union select agar kolom yang bisa diinjeksi tampil, akan memodifikasi URL lagi, tampilannya akan menjadi sebagai berikut.

Gambar 4.4. fungsi union select

Fungsi union select pada dasarnya untuk menggabungkan dua atau lebih query select dalam satu hasil keluaran. Setelah memodifikasi URL dengan fungsi union select maka dibawah form search akan menampilkan angka ajaib, karena angka tersebut adalah letak kolom dimana attacker bisa melakukan injeksi lebih lanjut.

Gambar 4.5. kolom yang bisa diinjeksi

Dari angka diatas terdapat 4 celah yang bisa diinjeksi, yaitu kolom 2,3,4, dan 5,

Tabel 4.1. celah keamanan

Celah yang bisa diinjeksi	Keterangan
Kolom 1	Not Vulnerable
Kolom 2	Vulnerable
Kolom 3	Vulnerable
Kolom 4	Vulnerable
Kolom 5	Vulnerable
Kolom 6	Not Vulnerable
Kolom 7	Not Vulnerable

Dari tabel diatas, terdapat beberapa celah pada SQL Injection, yaitu pada kolom ke 2,3,5, dan 4. Pada tabel diatas dengan modifikasi fungsi union select, bisa dilihat terdapat 4 celah yang bias diinjeksi, dari angka tersebut bisa di intip beberapa informasi seperti nama database, user database, versi database, dll. Untuk melihat nama database, versi database dll lalu selanjutnya memodifikasi URL sebagai berikut

berikutnya akan menampilkan kolom yang ada pada database, dengan perintah yang sama untuk menampilkan tabel, modifikasi URL persis seperti menampilkan tabel dengan perintah column_name.

Gambar 4.6. modifikasi URL menampilkan kolom database

Dari perintah modifikasi pada gambar diatas, browser akan menampilkan nama kolom yang ada pada database, tampilannya sebagai berikut

Setalah itu mencari kolom yang ada pada nama tabel “users” yang ada pada database, dengan memodifikasi URL sebagai berikut

Gambar 4.7. URL untuk menampilkan kolom pada tabel users

Dari perintah diatas untuk menampilkan kolom yang ada pada tabel users hasilnya adalah sebagai berikut

Search for a movie: Search

Title	Release	Character	Genre	IMDb
id	3	5	4	Link
login	3	5	4	Link
password	3	5	4	Link
email	3	5	4	Link
secret	3	5	4	Link
activation_code	3	5	4	Link
activated	3	5	4	Link
reset_code	3	5	4	Link
admin	3	5	4	Link
USER	3	5	4	Link
CURRENT_CONNECTIONS	3	5	4	Link

Gambar 4.8. kolom pada tabel users

Dari tampilan diatas, seluruh nama tabel users sudah diketahui, selanjutnya melihat informasi lebih dalam pada kolom login dan password karena dari sini bisa didapatkan user dan password

Untuk melihat informasi pada tabel login dan password modifikasi URL sebelumnya dan menyelipkan kata login pada URL sehingga menjadi seperti berikut

Gambar 4.9. menampilkan form login pada tabel users

Pada perintah diatas browser akan menampilkan beberapa user yang ada pada tabel user, berikut tampilan yang akan dimunculkan oleh browser.

Title	Release	Character	Genre	IMDb
A.I.M.	3	5	4	Link
bee	3	5	4	Link
reza	3	5	4	Link

Gambar 4.10. users pada database

Dari gambar diatas bisa dilihat beberapa user login yang terdapat pada web bwapp ini, terdapat 3 user yaitu A.I.M. , bee dan juga reza selanjutnya akan melihat informasi lainnya yaitu password, dengan perintah yang sama pada modifikasi URL sebelumnya bisa melihat password pada database, berikut adalah modifikasi URL untuk menampilkan password pada database.

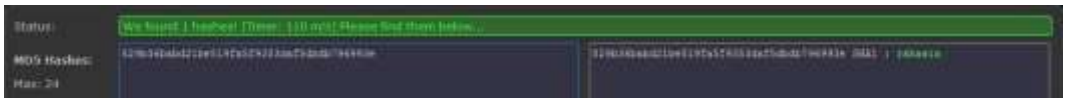
Gambar 4.11. URL untuk menampilkan password

Pada perintah modifikasi URL diatas browser akan menampilkan informasi password pada database, browser akan menampilkan sebagai berikut

Title	Release	Character	Genre	IMDb
6885858486f31043e5839c735d99457f045affd0	3	5	4	Link
829b36babd21be519fa5f9353daf5dbdb796993e	3	5	4	Link

Gambar 4.12. browser menampilkan password

bisa dilihat apa yang ditampilkan browser pada gambar diatas, terlihat bahwa password yang terlihat terenkripsi dalam bentuk md5 atau enkripsi yang lainnya. Untuk menterjemahkannya bisa menggunakan website yang menyediakan cracking password. Disini menggunakan website hashkiller.co.uk untuk



Gambar 4.13. Enkripsi password

Terlihat bahwa md5 yang terenkripsi menampilkan hasil enkripsi password yaitu “rahasia”. Dari sini serangan untuk SQL Injection telah selesai lalu bisa dicoba user dan password yang didapat dari serangan sebelumnya untuk login pada web bwapp ini.

Berikut ini tabel hasil ujicoba penetrasi serangan SQL Injection

Tabel 4.2. Hasil Ujicoba serangan SQL Injection

No.	Script serangan	Skenario serangan	Hasil
1.	localhost/bwapp/sqli_1.php?title='	SQL Injection	Berhasil
2.	localhost/bwapp/sqli_1.php?title=1' union select 1,2,3,4,5,6,7-- -	SQL Injection	Berhasil
3.	localhost/bwapp/sqli_1.php?title=1' union select 1,2,3,4,5,6,7 from information_schema-- -	SQL Injection	Berhasil
4.	localhost/bwapp/sqli_1.php?title=1' union select login from users-- -	SQL Injection	Berhasil
5.	localhost/bwapp/sqli_1.php?title=1' union select password from users-- -	SQL Injection	Berhasil

4. 2. Memberi Pencegahan SQL Injection

Pengujian yang sudah dilakukan adalah pengujian SQL Injection di form search, disini akan diberikan pencegahan untuk penangkal SQL Injection. Pada umumnya contoh sintak yang dikirim sebagai berikut

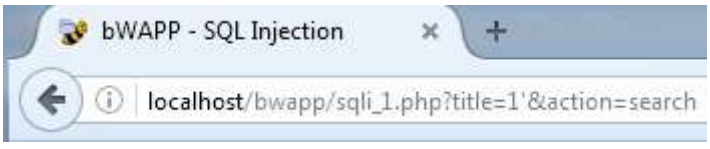
Select *from movies where title action search

maka pada penulisan sintak php akan menjadi seperti berikut

```
$sql = "SELECT * FROM movies WHERE title LIKE '%" . sqli($title) . "%'";
```

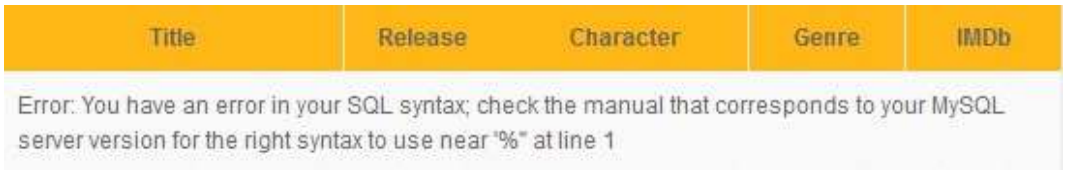
Gambar 4.14. script php

Pada proses eksekusi normal sintak tersebut, database akan memberikan balikan hasil yang sesuai parameter yang dikirimkan. Namun bila kita meracuni parameter yang dikirim melalui URL dengan sebuah karakter khusus yaitu single quote (‘) seperti ini



Gambar 4.15. modifikasi URL

Maka SQL query tersebut tidak akan bisa dieksekusi dan database server akan memberikan balikan berupa pesan error seperti berikut



Gambar 4.16. hasil balikan dari database

Mengapa? Karena yang dikirimkan ke database terdapat 2 buah single quote, hal inilah yang menyebabkan error, dan hal inilah yang menjadi celah sebuah situs dan dengan mudah dieksploitasi dengan metode SQL Injection. Bagaimana cara mengatasinya, dengan cara melakukan sanitasi pada data yang dikirimkan dari URL. PHP sendiri telah menyediakan method khusus untuk menangani hal ini, yaitu *myql_real_escape* atau *mysql_real_escape_string.* Saat penggunaan method itu maka sintak pengiriman SQL Query ke database akan menjadi seperti berikut.

```
$sql = "SELECT * FROM movies WHERE title LIKE '%" . mysql_real_escape_string(sqli($title)) . "%'";
```

Gambar 4.17. method anti single quote

Dan dengan demikian database hanya membaca 1 single quote. Dengan tambahan method diatas maka single quote tidak terbaca dan dibaca sebagai tanda slash (/)

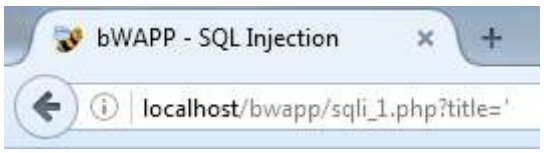
4. 3. Pengujian setelah diberi pencegahan

Pada tahap ini akan dicoba melakukan serangan SQL Injection lagi pada web aplikasi bwapp yang telah diberi pencegahan. Masukkan method tambahan pada source code, sisipkan seperti gambar dibawah ini

```
$sql = "SELECT * FROM movies WHERE title LIKE '%" . mysql_real_escape_string(sqli($title)) . "%'";
```

Gambar 2.18. source code anti SQL Injection

Setelah ditambahkan source code diatas akan dicoba kembali melakukan serangan SQL Injection pada web aplikasi bwapp. Tambahkan single quote pada URL atau pada search form pada web aplikasi bwapp seperti tampilan dibawah ini



Gambar 4.19. modifikasi URL dengan single quote

Lalu setelah pada URL ditambahkan single quote web tidak akan menampilkan error lagi setelah ditambahkan *mysql_real_escape_string* seperti tampilan berikut ini

Gambar 4.20. hasil setelah diberi pencegahan

Dengan menambahkan *mysql_real_escape_string* fungsi single quote tidak lagi berfungsi pada web aplikasi bwapp

Berikut tabel pengujian setelah diberi pencegahan SQL Injection

Tabel 4.3. Hasil Ujicoba setelah diberi pencegahan

No.	Script serangan	Skenario serangan	Hasil
1.	localhost/bwapp/sqli_1.php?title='	SQL Injection	Gagal
2.	localhost/bwapp/sqli_1.php?title=1' union select 1,2,3,4,5,6,7-- -	SQL Injection	Gagal
3.	localhost/bwapp/sqli_1.php?title=1' union select 1,2,3,4,5,6,7 from information_schema-- -	SQL Injection	Gagal
4.	localhost/bwapp/sqli_1.php?title=1' union select login from users-- -	SQL Injection	Gagal
5.	localhost/bwapp/sqli_1.php?title=1' union select password from users-- -	SQL Injection	Gagal

4. 4. Analisis SQL Injection

a. Analisis method

Saat pertama kali membuka alamat localhost/bwapp/ di mozilla, tampilan yang terbuka adalah tampilan loginform aplikasi bwapp. Namun saat alamat tersebut dibuka, alamat URL dimozilla berubah menjadi localhost/bwapp/login.php. ini artinya saat pertama kali membuka alamat localhost/bwapp halaman yang diakses berdasarkan source code web app bwapp adalah login.php. setelah diketahui alamat URL berubah dapat dilihat bahwa metode pengiriman aplikasi bwapp menggunakan method GET. Hal ini juga dapat dilihat dari source code aplikasi bwapp seperti gambar 4.22.

```
<h1>SQL Injection (GET/Search)</h1>

<form action="<?php echo($_SERVER["SCRIPT_NAME"]); ?>" method="GET">
```

Gambar 4.21. Source Code Method GET

Methode atau metode adalah suatu proses bagaimana suatu data dikirimkan ke web server. Pada pemrograman web dikenal 2 metode yaitu method POST dan method GET. Keduanya memiliki perbedaan, dimana perbedaan yang paling mencolok adalah variabel yang dikirimkan. Untuk mengetahui perbedaan method GET dan method POST lebih rinci dapat dilihat perbedaan berikut

- Method GET
    a. Variabel terlihat pada URL
    b. Dibatasi oleh panjang string sebanyak 2047 karakter
    c. Memungkinkan pengunjung memasukkan variabel pada form proses
- Method POST
    a. Nilai variabel tersembunyi
    b. Tidak dibatasi oleh panjang string

Dilihat dari perbedaan diatas, diketahui bahwa mehod GET lebih rentan terhadap serangan SQL Injection. Sehingga dapat dikatakan bahwa web aplikasi bwapp ini sangat rentan terhadap serangan SQL Injection

b. Analisis operasi basis data

Pada saat pengujian yang telah dijelaskan diatas, terlihat bahwa saat bagian akhir alamat web bwapp diberi tanda petik satu (‘) muncul sebuah pesan error. Ini adalah cara untuk menguji apakah MySQL memiliki bug atau tidak. Apabila saat string tanda petik satu (‘) tersebut dimasukkan muncul pesan error, itu artinya MySQL memiliki bug, itu artinya ada kemungkinan web tersebut dapat diinjeksi.
Uji coba dilakukan dengan menggunakan 2 skenario. Skenario yang pertama dilakukan serangan SQL Injection pada aplikasi web sebelum diberi pencegahan. Sedangkan pada skenario ke dua dilakukan serangan SQL Injection pada aplikasi web yang telah diberi pencegahan.

4. 5. Mekanisme XSS (Cross Site Scripting)

Cross Site Scripting (XSS) adalah suatu serangan dengan menggunakan mekanisme injection pada aplikasi web dengan memanfaatkan metode HTTP Get atau HTTP Post. XSS biasa digunakan oleh pihak-pihak yang berniat tidak baik dalam upaya mengacaukan naskah program sebagai bagian text input melalui mekanisme inputan yang tesedia.

Pada form input, langsung coba serangan XSS, coba isi dengan normal sebelum kita coba melakukan injeksi, inputkan firstname dan lastname. Contoh firstname : reza last name : rafsanjani ,berikut tampilan setelah kita inputkan seperti contoh diatas

Gambar 4.22. form input pada XSS

Selanjutnya setelah dieksekusi akan menjadi seperti berikut

Gambar 4.23. Hasil input

Dari hasil diatas hasil input pada form terlihat normal, sekarang mulai lakukan injeksi dengan menambahkan kode script seperti HTML atau Javascript, sisipkan seperti kode berikut pada salah satu form

Gambar 4.24. script Js pada form input

Script pada form input adalah <script>alert(“hacking XSS”)</script> dan hasil setelah diinputkan script kode itu hasilnya akan menjadi berikut

Gambar 4.25. hasil input script

Pada gambar diatas bisa dilihat setelah disisipkan script terdapat box alert dan didalamnya terdapat kata yang diinputkan pada form input tadi, kenapa keluar box

alert setelah script di proses, karena validasi input tidak difilter dengan baik sebelum menampilkan data yang di generate oleh pengguna

Berikut tabel hasil ujicoba serangan XSS

Tabel 4.4. Hasil Ujicoba serangan XSS

No.	Script Serangan	Skenario Serangan	Hasil
1.	<script>alert(‘XSS)</script>	XSS	Berhasil
2.	<script>window.open(“http://localhost/log/index php”</script>	XSS	Berhasil
3.	“>><marquee XSS </marquee>	XSS	Berhasil
4.	<iframe<?php echo(11)?>onload=alert(‘XSS’)> </iframe>	XSS	Berhasil

4. 6. Memberi pencegahan XSS (Cross Site Scripting)

Dari pengujian serangan XSS (Cross Site Scripting) diatas dapat diketahui bahwa form input mempunyai kelemahan, tidak adanya filter pada validasi input dan mengakibatkan data yang ditampilkan langsung digenerate oleh pengguna. Salah satu cara sederhana untuk menghindari XSS (Cross Site Scripting) dan HTML Injection adalah dengan membuat karakter-karakter yang memiliki ‘makna’ di dalam HTML dan Javascript untuk diubah menjadi named entity, yaitu mengkonversi karakter khusus seperti < menjadi *<* , dan karakter > menjadi *&gt*; , atau cara lainnya adalah dengan menghilangkan sama sekali seluruh tag HTML atau script dari inputan user.

Untuk kedua keperluan ini, PHP memiliki fungsi **htmlspecialchars()** dan fungsi **strip_tagsa()**. Fungsi htmlspecialchars() akan mengkonversi 4 karakter khusus HTML menjadi named entity sehingga tidak akan diproses oleh web browser. Keempat karakter tersebut adalah : < , > , & , “ . keempat karakter khusus inilah yang membuat web browser akan menerjemahkan sebuah string menjadi kode HTML atau Javascript. Sedangkan fungsi strip_tags() akan menghapus seluruh tag HTML dari inputan user.

Berikut tampilan script kode sebelum diberi pencegahan

```php
if(isset($_GET["firstname"]) && isset($_GET["lastname"]))
{

    $firstname = $_GET["firstname"];
    $lastname = $_GET["lastname"];
```

Gambar 4.26. Source code tanpa pencegahan

Dan hasilnya bisa dilihat pada percobaan sebelumnya, semua inputan tidak difilter dengan baik oleh browser, inilah yang menjadi suatu lubang kelemahan yang bisa dimanfaatkan para penyerang. Setelah tahu bahwa form inputan tidak difilter dengan baik, disini akan diberikan pencegahan yang akan memfilter inputan pengguna, dengan fungsi htmlspecialchars() dan strip_tags() , berikut tampilan setelah diberi pencegahan

```php
if(isset($_GET["firstname"]) && isset($_GET["lastname"]))
{

    $firstname = $_GET["firstname"];
    $lastname = $_GET["lastname"];
    $firstname = htmlspecialchars($firstname);
    $lastname = strip_tags($lastname);
```

Gambar 4.27. memberi pencegahan XSS

Dengan tambahan fungsi htmlspecialchars dan strip_tags maka serangan XSS (Cross Site Scripting) tidak bisa dilakukan lagi, pada sub bab berikutnya akan dilakukan pengujian setelah diberi pencegahan

4. 7. Pengujian setelah diberi pencegahan

Sebelumnya telah diberi script tambahan untuk menangkal serangan XSS (Cross Site Scripting) berikut bisa diuji lagi setelah diberi pencegahan, masuk ke web bwapp dengan URL localhost/bwapp, login dengan username : bee dan password : bug , lalu pilih XSS reflected Get, coba seperti pada pengujian sebelumnya, masukkan script pada salah satu form input yang disediakan

Gambar 4.28. Script serangan pada form input

Jalankan script diatas dan bisa dilihat hasilnya, script yang diinputkan tidak berefek apapun tidak seperti pada serangan sebelumnya , berikut hasil dari inputan setelah diberi pencegahan

Gambar 4.29. hasil form input setelah diberi pencegahan

Dari sini pengujian setelah diberi pencegahan telah berhasil dilakukan untuk menangkal serangan XSS.

Berikut hasil pengujian setelah diberi pencegahan

Tabel 4.5. hasil pengujian setelah diberi pencegahan

No.	Script Serangan	Skenario Serangan	Hasil
1.	<script>alert(‘XSS)</script>	XSS	Gagal
2.	<script>window.open(“http://localhost/log /index php”</script>	XSS	Gagal
3.	“>><marquee XSS </marquee>	XSS	Gagal
4.	<iframe<?phpecho(11)?>onload=alert (‘XSS’)> </iframe>	XSS	Gagal

4. 8. Analisis XSS (Cross Site Scripting)

Pada hasil percobaan diatas bisa diketahui bahwa serangan XSS (Cross Site Scripting) pada web aplikasi bwapp bisa dilakukan, dari inputan code script pada form inputan browser langsung mengenerate apa yang diinputkan oleh pengguna, inilah yang menjadi celah keamanan serangan XSS, tetapi setelah diberi pencegahan dengan menambahkan script pada program, serangan XSS tidak bisa dilakukan karena semua inputan tidak difilter dengan baik, inilah kelemahan XSS (Cross Site Scripting)

4. 9. Analisa Keseluruhan

Hasil pengujian SQL Injection dan XSS pada simulasi diatas terdapat beberapa celah keamanan, berikut tabel analisanya

a. SQL Injection sebelum diberi pencegahan

Tabel 4.6. celah keamanan yang bisa diinjeksi

Penetrasi serangan	Keterangan
SQL Injection single quote	Vulnerable
SQL Injection double quote	Not Vulnerable

Pada simulasi serangan SQL Injection menggunakan single quote, browser menampilkan error setelah ditambahkan single quote. berbeda dengan double quote, browser tidak menampilkan error sama sekali

b. SQL Injection setelah diberi pencegahan

Namun pada pengujian setelah diberi pencegahan, semua kolom yang tersedia untuk penetrasi tidak bisa dilakukan, karena script sudah ditambahkan. Berikut tabel setelah diberi pencegahan

Tabel 4.7. tabel celah keamanan tidak bisa diinjeksi

Penetrasi serangan	Keterangan
SQL Injection single quote	Not vulnerable
SQL Injection double quote	Not vulnerable

c. XSS (Cross Site Scripting) sebelum diberi pencegahan

Tabel 4.8. celah keamanan pada XSS (Cross Site Scripting)

Celah keamanan	Keterangan
Script HTML	Vulnerable
Script JavaScript	Vulnerable

Pada pengujian keamanan XSS (Cross Site Scripting) terdapat 2 celah keamanan pada XSS yaitu input script HTML dan input script JavaScript, sebelum diberi pencegahan, kedua script itu bisa diinputkan dan langsung dieksekusi oleh web. Ini dikarenakan validasi inputan tidak difilter dengan baik.

d. XSS (Cross Site Scripting) setelah diberi pencegahan

Setelah diberi pencegahan pada XSS (Cross Site Scripting) dengan menambahkan script pada program, penetrasi XSS tidak bisa dijalankan karena form input sudah diberi pencegahan untuk memfilter karakter-karakter khusus yang tidak bisa diinputkan pada form inputan. Berikut tabel setelah diberi pencegahan pada XSS.

Tabel 4.9. tabel celah keamanan XSS tidak bisa diinjeksi

Celah keamanan	Keterangan
Script HTML	Not vulnerable
Script JavaScript	Not vulnerable

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

a. Dari hasil pengujian simulasi SQL Injection dan XSS (Cross Site Scripting) terdapat 4 celah keamanan yang bisa dipenetrasi, sedangkan pada XSS terdapat 2 celah keamanan .

b. Setelah SQL Injection dan XSS diberi pengaman, hasil penetrasi tidak bisa dilakukan karena sudah diberi script pengaman anti SQL Injection dan XSS (Cross Site Scripting)

5.2. Saran

a. Pada web bwapp lebih diberi pengaman agar tidak mudah dilakukan injeksi

b. Untuk pengembangan selanjutnya dapat diberikan solusi, bukan saja SQL Injection dan XSS (Cross Site Scripting). Mengingat serangan SQL Injection tidak hanya penambahan single quote dan XSS tidak hanya menyisipkan script.

DAFTAR PUSTAKA

Andrea, Adelphia. 2016. “Cepat Belajar Hacking”. Elex Media Komputindo

AQua, Onix. 2013. “Cara Hacking Website Dengan Teknik Manual SQL Injection”, http://indocyberarmy.blogspot.com/2013/02/cara-hacking-website-dengan-teknik-sql.html.

DuPaul, Neil. 2016. “SQL Injection Cheat Sheet & Tutorial : Vulnerabilities & How to Prevert SQL Injection Attack”, http://www.veracode.com/security/sql-injection.

Clarke, J., 2009, SQL Injection Attacks and Defense. Burlington: Syngress Publishing and Elseiver.

Chandraleka, Happy. “Siapa Bilang nge-Hack Itu Susah”. Elex Media Komputindo

Digdo, Pringgo, Girindro. 2012. “Analisis Serangan dan Keamanan pada Aplikasi Web”. Elex Media Komputindo

Kristanto, Andri. 2010. “Kupas Tuntas PHP & My SQL”. Cable Book

Kurniawan, Dedik. 2013. “Buku Pintar Teknik Hacking”. Elex Media Komputindo

Muammar, Ahmad. 2013. “Web Hacking (basic)”. OSCP

McCluer, Struart. 2009. “Web Hacking-Serangan dan Pertahanannya”. Andi Publisher

Sitorus, Eryanto. 2006. “Hacker dan Keamanan”. Andi Publisher

Zam, Efvy. 2015 “Teknik Hacking dengan SQL Injection”. Elex Media Komputindo

Zam, Efvy. 2015 “Hacking aplikasi web : Uncensored”. Jasakom