# ANALISIS KEAMANAN SERANGAN *SQL INJECTION* BERDASARKAN METODE KONEKSI DATABASE

Syafrizal S. Ardiansyah[1], Suwanto Raharjo[2], Joko Triyono[3]

[123]Teknik Informatika, FTI, IST AKPRIND Yogyakarta.
Email : rizal.ex99@gmail.com [1], wa2n@akprind.ac.id [2], jack@akprind.ac.id [3]

***ABSTRACT***

*The rapid development of information technology enabling everyone to access the needed information instantly. Security becomes a very important factor if the information are accessed globaly, one of the vulnerabilities that are common and very dangerous for the web is a SQL Injection. It therefore requires analysis of guaranteeing the security of the data connection on web applications become more secure. Analysis process of the SQL Injection attack to compare several types of data connections on PHP is Mysql Connect, Mysql-improved OOP and Procedural and PDO in RDBMS MySQL with use of the role-based and different data servers. According to results the research obtained significant differences lies in the data connection using the function bindParam and non-bindParam. The final result in analysis study obtained an understanding of how an attack can happen to the data connection and the results obtained from the research in SQL Injection attacks are useful for comparing a better connection method used by web developers to maintain security.*

***Keywords****: Security Database, SQL Injection, Data Connection, RBAC.*

**INTISARI**

Pesatnya perkembangan dunia teknologi informasi memudahkan setiap orang dalam mengakses informasi yang dibutuhkan dengan instan. Keamanan menjadi faktor yang sangat penting jika informasi yang diakses bersifat global, salah satu celah keamanan yang sering terjadi dan sangat berbahaya bagi *web* adalah *SQL Injection*. Oleh sebab itu diperlukan analisis yang menjamin keamanan koneksi data pada aplikasi web menjadi lebih aman. Proses analisis serangan *SQL Injection* membandingkan beberapa jenis koneksi data pada *PHP* yaitu *Mysql Connect*, *Mysql-improved OOP & Procedural* dan *PDO* pada *RDBMS MySQL* dengan penggunaan hak akses dan server data yang berbeda. Menurut hasil penelitian yang diperoleh perbedaan signifikan terletak pada koneksi data menggunakan fungsi *bindParam* dan *non-bind Param*. Hasil akhir pada penelitian analisis ini didapatkan pemahaman bagaimana serangan dapat terjadi pada koneksi data dan hasil yang diperoleh dari hasil penelitian serangan *SQL Injection* berguna untuk membandingkan metode koneksi yang lebih baik dipakai oleh pengembang untuk menjaga keamanan web.

**Kata kunci**: Keamanan Basis Data, *SQL Injection*, Koneksi Data, *RBAC*.

## PENDAHULUAN

Pesatnya perkembangan teknologi komunikasi dan informasi yang memudahkan setiap orang dalam mengakses informasi yang dibutuhkan dengan instan. Namun pada pekembangan teknologi terdapat kelemahan pada keamanan sistem sehingga perlu adanya perbaikan agar membuatnya menjadi lebih aman, kelemahan keamanan sistem pada teknologi tersebut umumnya ditemukan oleh hacker. Hacker adalah seseorang orang yang mempelajari, menganalisis, memodifikasi suatu program atau menerobos masuk ke suatu sistem jaringan untuk mendapatkan keuntungan atau hanya sekedar tertantang. Hacker sendiri dapat dibagi menjadi 2 (dua) jenis yaitu: hacker adalah seorang golongan profesional memiliki pengetahuan tinggi dan mendalam pada sistem komputer yang mengenal pasti kelemahan suatu sistem lalu memberitahukan kepada pemiliknya dan cracker adalah kebalikan dari pada hacker yaitu bersifat merusak suatu sistem guna kepentingannya pribadi.

Website pada umumnya masih banyak mempunyai celah kerentanan keamanan yang bisa disusupi salah satu contohnya dengan SQL Injection. SQL Injection termasuk daftar top 10 Open Web Application Security Project (OWASP) pada tahun 2013, seperti dijelaskan oleh situs

resminya di https://www.owasp.org/index.php/Top_10_2013-Top_10, termasuk di antaranya OS Injection dan LDAP Injection. SQL Injection adalah teknik exploitasi penyalahgunaan celah keamanan yang terjadi pada lapisan database sebagai menyimpan data aplikasinya. Metode ini menggunakan celah keamanan pada sebuah aplikasi dimana cracker dapat menyisipkan karakter-karakter berupa string atau kode ASCII pada bagian pernyataan yang berhubungan langsung untuk mengeksekusi query SQL di database. Hal tersebut terjadi karena adanya kelemahan developer pada pembuatan coding program yang menyebabkan aplikasi menjadi bermasalah pada segi keamanan sehingga pernyataan dapat langsung di akses ke database tanpa penyaringan pada aplikasi.

Penelitian pada skripsi ini akan menganalisisa keamanan data pada *database* berdasarkan metode koneksi pembuatan *coding* yang berguna mengamankan informasi penting data pada *database* dengan tujuan untuk mendapatkan hasil nyata perbandingan keamanan metode koneksi pada pembuatan program yang umunya dipakai oleh pengembang aplikasi sehingga dapat diputuskan metode koneksi manakah yang lebih baik dan membuat keamanan data pada *database* terjaga.

## TINJAUAN PUSTAKA

Pada penelitian ini, beberapa referensi yang telah diteliti dan berhubungan dengan obyek penelitian.Aplikasi *web* berinteraksi dengan *back-end database* untuk mengambil data sebagai dan ketika diminta oleh pengguna. Aplikasi *web* (Seperti *e-commerce*, perbankan, belanja, perdagangan, blog dll) merupakan tulang punggung dari industri bisnis *online* saat ini. Untuk kegiatan seperti membayar tagihan & informasi barang yang harus disimpan aman dengan aplikasi *web* ini tapi sayangnya tidak ada jaminan integritas dan kerahasiaan informasi. Ekspose secara global pada aplikasi ini membuat mereka rentan terhadap serangan karena kehadiran kerentanan pada keamanan. kerentanan keamanan ini terus menginfeksi aplikasi *web* melalui serangan injeksi. *SQL Injection Attack* (*SQLIA*) adalah salah satu ancaman yang paling atas dalam *database* aplikasi *web centric* dan *SQL Injection Vulnerabilities* (*SQLIV*) adalah tipe kerentanan keamanan yang sangat serius. *SQLIA* memungkinkan penyerang untuk mendapatkan kontrol atas *database* dari sebuah aplikasi yang mengakibatkan penipuan keuangan, kebocoran data rahasia, *hacking* pada jaringan, menghapus *database*, pencurian dan banyak lagi. Dalam makalah ini akan membahas klasifikasi serangan *SQL Injection* dan analisis juga yang harus dilakukan atas berdasarkan risiko data yang terkait dengan masing-masing serangan termasuk melalui *URL* atau *input* pengguna (Sharma & Jain, 2014).

Serangan *SQL Injection* adalah salah satu yang paling serius kerentanan keamanan dalam sistem aplikasi *web*, sebagian besar kerentanan disebabkan oleh kurangnya validasi *input* dan pengunaan parameter *SQL*. Jenis serangan *SQL Injection* dan teknologi pencegahannya dijelaskan pada *paper* ini. Metode mendeteksi tidak hanya memvalidasi *input* pengguna, tetapi juga menggunakan jenis parameter *SQL* yang aman. Model pertahanan *SQL Injection* dibuat menurut proses deteksi, yang dimana lebih efektif terhadap serangan *SQL Injection* (Qian, Zhu, Hu, & Liu, 2015). Serangan *SQL Injection* adalah cara yang populer untuk menyerang dalam hal struktur dokumen dan ancaman umum lainnya sekarang. Ada beberapa cara mendeteksi serangan sesuai penelitian dan juga metode pencegahan yang telah dibahas dalam beberapa *paper* penelitian. Tujuan akhir *paper* ini yaitu untuk menembus serangan itu. Untuk itu kami telah mengusulkan sebuah kerangka efisien dari mana otoritas pusat dapat mengendalikan semua *IP* pengunjung dan membatasi *IP* tersebut. Jika *IP* dibatasi oleh zona maka isi dari konten juga akan dibatasi menggunakan pembaruan perintah *SQL* dengan status dimodifikasi berdasarkan negara. Jika yang dibatasi *IP* ingin mengakses data dari sumber yang tidak terpercaya, maka akan segera memberitahukan ke *admin* dan saat penyerangan akan ada peringatan yang disimpan ke dalam *log* daerah serangan berasal. Akhirnya dengan membandingkan hasil kita akan mendapatkan kejelasan hasil yang lebih baik dibandingkan dengan tes dan metodologi (Kaushik & Ojha, 2014).

Keamanan data adalah tugas yang paling penting dalam dunia saat ini. Selama bertahun-tahun berbagai skema enkripsi dikembangkan dalam rangka untuk melindungi data dari berbagai serangan oleh para penyusup. Makalah ini membahas pentingnya enkripsi pada *database* dan membuat tinjauan yang mendalam dari berbagai teknik enkripsi *database* dan membandingkannya berdasarkan keuntungan dan kerugian mereka (Singh & Kaur, 2015).

*Role-Based Access Control* (*RBAC*) adalah model *de facto* akses kontrol yang digunakan sistem informasi saat ini, juga dalam lingkungan militer. Hal ini karena *RBAC* dapat digunakan untuk menggambarkan keamanan multi-level kebijakan *AC* begitu umum di lingkungan militer, penanganan informasi dari berbagai tingkat klasifikasi. *Cyriptographic Access Control* (*CAC*), di sisi lain, merupakan pelaksanaan paradigma yang dimaksudkan untuk menegakkan kebijakan kriptografi *AC*. Metode *CAC* yang juga menarik dalam *cloud* militer dan lingkungan taktis mereka karena didistribusikan dan operasi secara *offline*. Menggabungkan kemampuan *RBAC* dan *CAC* tampaknya sepenuhnya sulit dipahami. *Paper* ini mempelajari kelayakan pelaksanaan *RBAC* sehubungan dengan hak akses menggunakan jenis terbaru dari skema kriptografi disebut *Attribute-Based Encryption* (*ABE*). penyajian sebuah model implementasi berdasarkan *Extensible Control Access Markup Language* (*XACML*) referensi arsitektur dan mengevaluasi bagaimana keadaan *ABE* saat ini dan dapat mewujudkan berbeda dasar *RBAC* dari komponen model. Penelitian ini akan menunjukkan bahwa hal tersebut layak untuk diterapkan setidaknya inti *RBAC* dengan standar arsitektur *XACML* dan *ABE* model, dan bahwa ekspresi dari skema *ABE* dapat mencapai hampir keseluruhan hal simetris perintah *RBAC* dan fungsi, seperti *Dynamic Separation of Duty* (Kiviharju, 2013).

*Role mining* berfungsi untuk menentukan peran mengatur implementasi *Role-Based Access Control* (*RBAC*) sistem dan dianggap sebagai salah satu dari fase implementasi yang paling penting dan paling mahal. Sementara berbagai macam model *mining* telah diusulkan, dan menemukan pengalaman / persepsi pengguna - salah satu tujuan utama untuk sistem informasi - yang mengejutkan yaitu diabaikan oleh karya-karya yang ada. Salah satu keuntungan *RBAC* adalah untuk mendukung beberapa tugas hak akses dan memungkinkan pengguna untuk mengaktifkan hak akses yang diperlukan untuk melakukan tugas pada setiap sesi. Sebuah sistem *RBAC* yang *user-friendly* diharapkan untuk menetapkan beberapa hak akses untuk setiap pengguna. Jadi dalam makalah ini mengusulkan untuk menggabungkan proses *role mining*, penetapan penggunaan hak akses kendala yang mengamanatkan jumlah maksimum peran setiap pengguna yang dapat dimiliki. Dibawah ini alasan, penelitian ini merumuskan *role mining* berorientasi pengguna sebagai masalah penggunaan *role mining*, di mana semua pengguna memiliki tugas hak akses yang sama maksimal, masalah pribadi *role mining* di mana pengguna dapat memiliki tugas dan hak akses maksimal yang berbeda, dan versi perkiraan dari dua masalah, yang mentolerir sejumlah penyimpangan dari rekonstruksi yang lengkap. Kendala ekstra pada tugas hak akses maksimal menimbulkan tantangan besar untuk *role mining*, yang pada umumnya adalah masalah yang sulit. Beberapa metode *role mining* yang telah ada, dapat dilihat penerapannya untuk masalah yang dihadapi. Mengingat insufisiensi, dengan menyajikan algoritma baru, yang didasarkan pada dinamika strategi generasi hak akses kandidat terbaru, disesuaikan dengan masalah. Percobaan pada *set data benchmark* menunjukkan efektivitas algoritma yang diusulkan (Lu, Hong, Yang, Duan, & Badar, 2015).

Penelitian yang telah disebutkan di atas menjadi referensi dalam penyunsunan dan pembuatan Analisi Keamanan Serangan *SQL Injection* Berdasarkan Metode Koneksi Database dengan landasan teori hasil dari perbandingan metode koneksi data pada bahasa pemrograman PHP dengan RDBMS MySQL.

Menurut dokumen resmi *PHP*, *PHP* adalah singkatan dari *PHP Hypertext Preprocessor*, merupakan bahasa berbentuk *script* yang ditempatkan dalam *server* dan diproses di *server*. Hasilnya yang dikirimkan ke *client*, tempat pemakai menggunakan *browser* (Kadir, 2001). *MYSQL* adalah sebuah perangkat lunak pembuat *database* yang bersifat terbuka atau *open source* dan berjalan di semua *platform* baik Linux maupun Windows, *MySQL* merupakan program pengakses *database* yang bersifat *network* sehingga dapat digunakan untuk aplikasi *multi user* (banyak pengguna). *MySQL* merupakan *Relational Database Management System* (*RDBMS*) yang didistribusikan secara gratis dibawah lisensi *GPL* (*General Public License*), dimana setiap orang bebas untuk menggunakan *MySQL*, namun tidak boleh dijadikan produk turunan yang bersifat komersial (Nugroho, 2004).

*SQL Injection* adalah kerentanan kemanan terjadi ketika memberikan penyerang kemampuan dalam merubah *Structured Query Language* (*SQL*) *query* melewati *back-end database* aplikasi. Dengan dapat menginfeksi melewati *database*, penyerang dapat mempengaruhi *sytax* dan kapabilitas *SQL* itu sendiri, sebagaimana kemampuan menguasai dan fungsional fleksibilitas *database* dan fungsional sistem operasi yang menyimpan

*database*. Kerentanan keamanan pada *SQL Injection* tidak langsung berakibat langsung aplikasi *web*, setiap kode diterima sebagai masukkan dari sumber yang tidak terpercaya dan digunakan sebagai masukkan pada *form* dinamis pada *statement SQL* dapat saja berbahaya (Clarke, 2009).

*RBAC* merupakan multi tingkat keamanan, dengan konseptual sederhana dimana akses ke objek pada sistem komputer didasarkan pada peranan pengguna dalam sebuah organisasi. Peranan dengan hak akses / tanggung jawab yang berbeda telah lama dikenal dalam organisasi bisnis, dan aplikasi komputer yang telah beredar secara komersial setidaknya pada 1970-an mempunyai keterbatasan pada kendala akses tersebut. Sebagai contoh, aplikasi perbankan *online* yang ada pada periode tersebut baik *teller* dan *teller* pengawas mempunyai hak akses yang dapat mengeksekusi perintah yang berbeda dari transaksi seharusnya, sekaligus pengguna *ATM* mampu mengeksekusi perintah transaksi terhadap *database* yang sama pada *teller* (Ferraiolo, Kuhn, & Chandramouli, 2007)

## PEMBAHASAN

Hasil proses pengujian dan pengambilan data pada aplikasi *web* untuk mengetahui hasil yang didapatkan telah sesuai dan dapat dipresentasikan kedalam susunan tabel yang telah dikategorikan menurut hak akses, koneksi data dan jenis *server*. Penjelasan secara detail jenis server, hak akses, dan koneksi data yang digunakan dan hasil yang didapat dari percobaan serangan *SQL Injection* dengan *sqlmap*.

Dari sisi server pengujian menggunakan 4 (empat) jenis yaitu *Localhost Server*, *Virtual Private Server*, *Server Hosting* Prabayar dan *Server Hosting* Gratis. Bagian *hardware* untuk *Localhost* dan *VPS* yang dipakai adalah komputer milik kampus IST Akprind yang berada di Laboratorium 6 Komputer Jaringan & Multimedia, sedangkan *hardware* untuk *Hosting* Prabayar dan Hosting Gratis yang dipakai tidak dapat diketahui oleh peneliti karena kebijakan privasi pada pemilik *hosting*, untuk keperluan *hosting* prabayar menggunakan jasa jaringan dari https://www.domainesia.com/ dengan pembelian domain premium securitylook.net dengan arsitektur jaringan *two tier* dan hosting gratis menggunakan jasa jaringan dari http://www.freehostingeu.com/ dengan domain gratis seclook.eu.pn dengan arsitektur jaringan *three tier*. Penjelasan spesifikasi *hardware* dan *software server* dijelaskan seperti berikut:

| No. | Jenis Server | Hardware | Software | IP / Domain |
|---|---|---|---|---|
| 1 | Server Localhost | CPU Intel® core2duo E7500 @2.93GHz | OS Debian 8.0 jessie | 192.168.2.105 |
| | | RAM 1 GHz | Web Server Apache 2 | (lokal) |
| | | | Database Server MySQL 5.5.47 | |
| | | | PHP 5.6.17 | |
| 2 | Virtual Private Server | CPU Intel® core2duo E7500 @2.93GHz | OS Debian 8.0 jessie | 202.91.10.211 |
| | | RAM 1 GHz | Web Server Apache 2 | (publik) |
| | | | Database Server MySQL 5.5.47 | |
| | | | PHP 5.6.17 | |
| 3 | Server Hosting Prayabar | X | Web Server Apache 2 | 45.114.118.228 |
| | | | Database Server MySQL 5.6.30 | (publik & basis data) |
| | | | PHP 5.5 | |
| 4 | Server Hosting Gratis | X | Web Server Apache 2 | 83.125.22.215 |
| | | | Database Server MySQL 5.5 | (publik) |
| | | | PHP 5.5.35 | 82.197.130.53 |
| | | | | (basis data) |

Dari sisi hak akses pengujian menggunakan 8 (delapan) jenis yaitu *root*, *user1*, *user2*, *user3*, *user4*, *user5*, *user6*, dan *user7*. Hak akses berperan penting pada *RBAC* program analisis serangan *SQL Injection* dengan fungsi utama mengatur keamanan dan batasan pada level sehingga setiap pengguna yang mempunyai akses harus mempunyai batas akses terhadap *RDBMS MySQL*. Berikut adalan penjelasan *RBAC* pengguna dari sisi database dan nama yang dipakai termasuk deskripsi isi dari hak akses yang digunakan.

| No. | Hak Akses & Database Table | Jenis Akses | Keterangan |
|---|---|---|---|
| 1 | **root** | Data | SELECT, INSERT, UPDATE, DELETE, FILE |
| | GLOBAL | Structure | CREATE, ALTER, INDEX, DROP, CREATE TEMPORARY TABLES, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE, CREATE VIEW, EVENT, TRIGGER |
| | | Administration | GRANT, SUPER, PROCESS, RELOAD, SHUTDOWN, SHOW DATABASES, LOCK TABLES, REFERENCES, REPLICATION CLIENT, REPLICATION SLAVE, CREATE USER |
| | ALL | Data | SELECT, INSERT, UPDATE, DELETE |
| | | Structure | CREATE, ALTER, INDEX, DROP, CREATE TEMPORARY TABLES, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE, CREATE VIEW, EVENT, TRIGGER |
| | | Administration | GRANT, LOCK TABLES, REFERENCES |
| 2 | **user1** | Data | SELECT, INSERT, UPDATE, DELETE |
| | "bio" | Structure | CREATE, ALTER, INDEX, DROP, CREATE TEMPORARY TABLES, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE, CREATE VIEW, EVENT, TRIGGER |
| | | Administration | GRANT, LOCK TABLES, REFERENCES |
| 3 | **user2** | Data | SELECT, INSERT, UPDATE, DELETE |
| | "bio" | Structure | CREATE, ALTER, INDEX, DROP, CREATE TEMPORARY TABLES, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE, CREATE VIEW, EVENT, TRIGGER |
| 4 | **user3** "bio" Table "form" | Data | SELECT, INSERT, UPDATE, DELETE |
| 5 | **user4** "bio" Table "form" | Data | INSERT |
| 6 | **user5** "bio" Table "form" | Data | UPDATE |
| 7 | **user6** "bio" Table "form" | Data | DELETE |
| 8 | **user7** "bio" Table "form" | Data | SELECT |

Dan terakhir dari sisi metode koneksi data pengujian menggunakan 7 (tujuh) jenis sub-sub koneksi dengan 3 (tiga) diantaranya adalah jenis koneksi *parsing* utama data yaitu dengan *mysql*, *mysql-improved* (*mysqli*) dan *PHP Data Object* (*PDO*). Berikut adalah contoh skrip dengan fungsi SELECT pada masing-masing sub-sub metode koneksi yang dipakai.

1. *Mysql Connect*

```
1 $query = mysql_query("SELECT * FROM form WHERE id=$_GET[id]") or die(mysql_error());
2 $data = mysql_fetch_assoc($query);
```

Skrip 1. Fungsi SELECT pada *Mysql Connect*

2. *Mysqli Object Oriented PHP*

```
1 $query = $mysqli->query("SELECT * FROM form WHERE id=$_GET[id]") or die($mysqli->error);
2 $data = $query->fetch_assoc();
```

Skrip 2. Fungsi SELECT pada *Mysqli Object Oriented PHP*

3. *Mysqli Object Oriented PHP + Bindparam*

```
1 $stmt = $mysqli->prepare("SELECT * FROM form WHERE id =?") or die($mysqli-
```

```
    >error);
2   $stmt->bind_param("i", $_GET['id']);
3   $stmt->execute();
4   $stmt->bind_result($id, $nama, $tanggal_lahir, $agama, $jenis_kelamin, $alamat,
    $no_telp);
5   $stmt->fetch();
```

Skrip 3 .Fungsi SELECT pada *Mysqli Object Oriented PHP + Bindparam*

4. *Mysqli Procedural*

```
1   $query = mysqli_query($mysqli, "SELECT * FROM form WHERE id=$_GET[id] ORDER
    BY id DESC") or die(mysqli_error($mysqli));
2   $data = mysqli_fetch_assoc($query);
```

Skrip 4. Fungsi SELECT pada *Mysqli Procedural*

5. *Mysqli Procedural + Bindparam*

```
1   $query = mysqli_prepare($mysqli, "SELECT * FROM form WHERE id=? ORDER BY id
    DESC") or die(mysqli_error($mysqli));
2   mysqli_stmt_bind_param($query, "i", $_GET['id']);
3   mysqli_stmt_execute($query);
4   mysqli_stmt_bind_result($query, $id, $nama, $tanggal_lahir, $agama, $jenis_kelamin,
    $alamat, $no_telp);
5   mysqli_stmt_fetch($query);
```

Skrip 5. Fungsi SELECT pada *Mysqli Procedural + Bindparam*

6. *PDO Traditional*

```
1   try {
2   $query = $pdo->prepare("SELECT * FROM form WHERE id=$_GET[id]");
3   $query->execute();
4   $data = $query->fetch(PDO::FETCH_ASSOC);
5   } catch (PDOException $exc) {
6   echo $exc->getMessage();
7   }
```

Skrip 6. Fungsi SELECT pada *PDO Traditional*

7. *PDO Bindparam*

```
1   try {
2   $query = $pdo->prepare("SELECT * FROM form WHERE id=:id");
3   $query->bindparam(":id", $_GET['id']);
4   $query->execute();
5   $data = $query->fetch(PDO::FETCH_ASSOC);
6   } catch (PDOException $exc) {
7   echo $exc->getMessage();
8   }
```

Skrip 7. Fungsi SELECT pada *PDO Bindparam*

Hasil akhir pengujian penelitian dengan *sqlmap* maka dibagi berdasarkan 4 (empat) kategori jenis *server* yang terdiri dari *Localhost Server*, *Virtual Private Server*, *Server Hosting* Prabayar dan *Server Hosting* Gratis. Hasil data yang peroleh kemudian dirangkum dalam bentuk susunan tabel hasil analisa untuk membandingkan hak akses dan jenis koneksi data mana sajakah yang dapat memungkinkan terjadinya serangan *SQL Injection*. Pada hasil analisa dalam bentuk tabel jika serangan memungkinkan dilakukan atau berhasil maka pada tabel hasil peneliti akan memberikan tanda"√" yang mengartikan bahwa pada *file log* terdapat keseluruhan hasil yang hendak dicari, jika serangan tidak memungkinkan untuk dilakukan atau gagal maka peneliti cukup memberikan tanda "X" yang mengartikan bahwa hasil isi *file log* tidak didapatkan atau kosong. Berikut adalah hasil analisa dan susunan kategori hasil yang didapatkan oleh *sqlmap* berdasarkan jenis *server* yang dipakai.

1. *Localhost Server*

Hasil pertama dari pengujian pada jenis *Localhost Server*, data analisa yang didapatkan dengan keseluruhan jenis koneksi data dapat diterapkan meliputi *mysql connect, mysqli object oriented PHP, mysqli object oriented PHP + bindParam, mysqli procedural, mysqli procedural + bindParam, PDO Traditional* dan *PDO bindParam*, dengan seluruh jenis hak akses dapat diterapkan pada *server* tersebut yaitu *root*, *user1*, *user2*, *user3*, *user4*, *user5*, *user6* dan *user7*. Hasil analisa yang didapatkan tersusun dalam bentuk tabel seperti berikut.

| No | Jenis Koneksi | Jenis User Database | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | root | user1 | user2 | user3 | user4 | user5 | user6 | user7 |
| 1 | mysql_connect | √ | √ | √ | √ | X | X | X | √ |
| | mysqli | | | | | | | | |
| 2 | object oriented | √ | √ | √ | √ | X | X | X | √ |
| 3 | object oriented | X | X | X | X | X | X | X | X |
| 4 | procedural | √ | √ | √ | √ | X | X | X | √ |
| 5 | procedural + | X | X | X | X | X | X | X | X |
| | PDO | | | | | | | | |
| 6 | traditional | √ | √ | √ | √ | X | X | X | √ |
| 7 | bindparam | X | X | X | X | X | X | X | X |

2. *Virtual Private Server*

Hasil kedua dari pengujian pada jenis *Virtual Private Server*, data analisa yang didapatkan dengan keseluruhan jenis koneksi data dapat diterapkan meliputi *mysql connect, mysqli object oriented PHP, mysqli object oriented PHP + bindParam, mysqli procedural, mysqli procedural + bindParam, PDO Traditional* dan *PDO bindParam*, dengan seluruh jenis hak akses dapat diterapkan pada *server* tersebut yaitu *root*, *user1*, *user2*, *user3*, *user4*, *user5*, *user6* dan *user7*. Hasil analisa yang didapatkan tersusun dalam bentuk tabel seperti berikut.

| No | Jenis Koneksi | Jenis User Database | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | root | user1 | user2 | user3 | user4 | user5 | user6 | user7 |
| 1 | mysql_connect | √ | √ | √ | √ | X | X | X | √ |
| | mysqli | | | | | | | | |
| 2 | object oriented | √ | √ | √ | √ | X | X | X | √ |
| 3 | object oriented | X | X | X | X | X | X | X | X |
| 4 | procedural | √ | √ | √ | √ | X | X | X | √ |
| 5 | procedural + | X | X | X | X | X | X | X | X |
| | PDO | | | | | | | | |
| 6 | traditional | √ | √ | √ | √ | X | X | X | √ |
| 7 | bindparam | X | X | X | X | X | X | X | X |

3. *Server Hosting* Prabayar

Hasil ketiga dari pengujian pada jenis *Server Hosting* Prabayar, data analisa yang didapatkan hanya beberapa jenis koneksi data dapat diterapkan meliputi *mysql connect, mysqli object oriented PHP, mysqli object oriented PHP + bindParam, mysqli procedural, mysqli procedural + bindParam, PDO Traditional* dan *PDO bindParam*, dengan jenis hak akses dapat diterapkan hanya beberapa yaitu *user2*, *user3*, *user4*, *user5*, *user6* dan *user7*. Hasil analisa yang didapatkan tersusun dalam bentuk tabel seperti berikut.

| No | Jenis Koneksi | Jenis User Database | | | | | |
|---|---|---|---|---|---|---|---|
| | | user2 | user3 | user4 | user5 | user6 | user7 |
| 1 | mysql_connect | √ | √ | X | X | X | √ |
| | mysqli | | | | | | |
| 2 | object oriented | √ | √ | X | X | X | √ |
| 3 | object oriented | X | X | X | X | X | X |
| 4 | procedural | √ | √ | X | X | X | √ |
| 5 | procedural + | X | X | X | X | X | X |
| | PDO | | | | | | |
| 6 | traditional | √ | √ | X | X | X | √ |
| 7 | bindparam | X | X | X | X | X | X |

4. *Server Hosting* Gratis

Hasil terakhir dari pengujian pada jenis *Server Hosting* Gratis, data analisa yang didapatkan hanya beberapa jenis koneksi data dapat diterapkan meliputi *mysql connect, mysqli object oriented PHP, mysqli object oriented PHP + bindParam, mysqli procedural, mysqli procedural + bindParam, PDO Traditional* dan *PDO bindParam*, dengan jenis hak akses dapat diterapkan hanya *user2*. Hasil analisa yang didapatkan tersusun dalam bentuk tabel seperti berikut.

| No | Jenis Koneksi | Jenis User Database |
|---|---|---|
| | | user2 |
| 1 | **mysql_connect** | √ |
| | **mysqli** | |
| 2 | **object oriented** | √ |
| 3 | **object oriented** | X |
| 4 | **procedural** | √ |
| 5 | **procedural +** | X |
| | **PDO** | |
| 6 | **traditional** | √ |
| 7 | **bindparam** | X |

## KESIMPULAN

Kesimpulan yang dapat diambil dari skripsi ini adalah sebagai berikut:

1. Membuktikan dari proses penelitian bahwa hasil akhir yang diperoleh jika aplikasi yang dikembangkan menggunakan *Mysql-improved OOP* atau *Procedural* dengan *bindParam* atau *PHP Data Objects* dengan *bindParam*, serangan *SQL Injection* tidak dapat mungkin dilakukan untuk sampai saat penelitian ini dibuat.
2. Hasil sangat berbeda jika membandingkan *server* milik pribadi dengan jasa jaringan sebab kebebasan konfigurasi *server* pada jasa jaringan terbatas untuk setiap pengguna yang mempunyai akses terhadap *server shared hosting* untuk *hosting prayabar user* yang dapat diimplementasikan mulai *user2* sampai *user7*, sedangkan *hosting gratis* hanya *user2* dan untuk *server* pribadi keseluruhan hak akses dari *user root* sampai *user7*.
3. Tidak keseluruhan *API* pada *DBMS MySQL* mendukung eksplorasi metode serangan *SQL Injection*, sebagai contoh fungsi *stacked queries* pada *DBMS MySQL* tidak mendukung untuk bahasa pemrograman *PHP*.
4. Penelitian menggunakan sebuah *tools penetration system* pada *kali linux* yaitu *sqlmap* yang berguna mempercepat proses *SQL Injection* pada aplikasi *web* target.

Penyempurnaan dan pengembangan skripsi ini masih dapat dilakukan dalam hal sebagai berikut:

1. Pendalaman untuk pencegahan serangan *SQL Injection* masih mungkin dilakukan dengan penambahan fungsi-fungsi enkripsi dan algoritma yang mampu mengamankan data pada *RDBMS*.
2. Penelitian tentang pengaturan hak akses untuk setiap pengguna sangat berpengaruh pada manajemen *RBAC* sehingga dapat menjaga kerahasiaan data.
3. Perlunya penelitian yang juga membandingkan apakah serangan *SQL Injection* mempunyai dampak jika menyerang *RDBMS* lainnya seperti *Oracle*, *PostgreSQL*, *MS SQL* dan lain sebagainya.

## DAFTAR PUSTAKA

Clarke, J. (2009). *SQL Injection Attacks and Defense.* Burlington: Syngress Inc.

Ferraiolo, D. F., Kuhn, D. R., & Chandramouli, R. (2007). *Role-Based Access Control.* Boston: Artech House.

Kadir, A. (2001). *Dasar Pemrograman Web Dinamis dengan Menggunakan PHP.* Yogyakarta: ANDI.

Kaushik, M., & Ojha, G. (2014). Attack Penetration System for SQL Injection. *International* Journal *of Advanced Computer Research*, 4.

Kiviharju, M. (2013). Cryptographic Roles in the Age of Wikileaks. *IEEE Military Communications Conference*.

Lu, H., Hong, Y., Yang, Y., Duan, L., & Badar, N. (2015). Towards user-oriented RBAC model. Journal *of Computer Security*, 23.

Nugroho, B. (2004). *Aplikasi Pemrograman Web Dinamis dengan PHP dan MySQL.* Yogyakarta: Gava Media.

Qian, L., Zhu, Z., Hu, J., & Liu, S. (2015). Research of SQL Injection Attack and Prevention Technology. *International Conference on Estimation, Detection and Information Fusion*.

Sharma, C., & Jain, S. (2014). Analysis and Classification of SQL Injection Vulnerabilities and Attacks on Web Applications. *IEEE International Conference on Advance in* Engineering *& Technology Research, August 01-02*.

Singh, P., & Kaur, K. (2015). Database Security Using Encryption. *International Conference on* Futuristic *Trend in Computational Analysis and Knowledge Management*.