PENINGKATAN KESELAMATAN PANGKALAN DATA MELALUI PENCEGAHAN SUNTIKAN SQL

MAYSHARA KARIM
KAMSURIAH AHMAD

Fakulti Teknologi & Sistem Maklumat, Universiti Kebangsaan Malaysia

ABSTRAK

Suntikan SQL merupakan salah satu ancaman yang kritikal terhadap keselamatan pangkalan data. Kesan daripada serangan suntikan SQL menyebabkan data yang terkandung di dalam pangkalan data berisiko untuk dieksploitasi oleh pihak tidak bertanggungjawab, menjejaskan integriti data, mengganggu operasi pelayan dan seterusnya menjejaskan imej organisasi. Walaupun suntikan SQL merupakan serangan yang dilakukan di aras aplikasi, pencegahan suntikan SQL memerlukan kawalan keselamatan di semua aras maklumat iaitu aras aplikasi, aras pangkalan data dan aras rangkaian. Ketiadaan langkah pencegahan suntikan SQL di aras aplikasi menjadikan pangkalan data terdedah kepada serangan. Dalam projek ini, aplikasi web di Agensi Kelayakan Malaysia (MQA) dijadikan sebagai kajian kes untuk mengkaji kelemahan terhadap suntikan SQL. Kelemahan yang dikenal pasti adalah berpunca daripada kod aturcara yang menggunakan SQL dinamik, ketiadaan pengesahsahihan input dan pengendalian ralat yang tidak konsisten. Bagi mengatasi kelemahan ini, kaedah prosedur tersimpan yang ditambah baik dengan pertanyaan berparameter bagi menggantikan SQL dinamik, digabungkan dengan pengesahsahihan input dan pengendalian ralat telah dibangunkan dan diuji ke atas aplikasi web semu. Hasil simulasi eksperimen membuktikan bahawa gabungan kaedah tersebut mampu mencegah suntikan SQL daripada berlaku. Penilaian statistik masa juga mendapati prosedur tersimpan diproses menggunakan tempoh masa yang lebih singkat berbanding SQL dinamik.

1. PENGENALAN

Kecanggihan teknologi maklumat memudahkan kehidupan tetapi dalam masa yang sama memberikan ancaman sekiranya aspek keselamatan tidak diberi perhatian khusus. Internet yang menghubungkan manusia di serata dunia membawa bersama ancaman penggodam. Penggodam mempunyai pelbagai cara untuk mencuri dan mengeksploitasi maklumat sesuatu pihak. Salah satunya adalah dengan melakukan suntikan SQL terhadap pangkalan data. Suntikan SQL berupaya menjejaskan keselamatan data organisasi; maka semakin ramai menyedari kepentingan mencegah serangan ini sebelum ia berlaku. Pencegahan suntikan SQL menjadi fokus kepada pihak yang terdiri daripada pentadbir pangkalan data, pentadbir rangkaian, pengaturcara kod aplikasi, pembekal sistem pangkalan data, pembangun perisian dan tidak terkecuali pengurusan tertinggi organisasi. Ini kerana masalah yang dibawa oleh suntikan SQL telah lama berlaku, iaitu sejak tahun 2002 (Horner & Hyslip 2017) dan masih menjadi topik hangat dalam isu keselamatan maklumat.

Keselamatan aplikasi web dan keselamatan pangkalan data adalah saling bergantungan antara satu sama lain. Sekiranya aplikasi web dimanipulasi penggodam, pangkalan data yang telah dilengkapi ciri-ciri keselamatan juga masih boleh diceroboh. Malik & Patel (2016) menyenaraikan ancaman terhadap keselamatan pangkalan data; antaranya ialah keistimewaan berlebihan *(excessive privilege)* yang merujuk kepada pengguna yang diberikan keizinan untuk mengakses atau menjalankan pelbagai transaksi di pangkalan data dan menyalahgunakan keizinan tersebut dan suntikan SQL *(SQL injection)* iaitu kemasukan input yang tidak dibenarkan ke

pangkalan data bagi menjalankan sebarang arahan yang tidak sah. Selain itu, kelemahan jejak audit *(weak audit trail)* atau perekodan transaksi pangkalan data secara automatik yang tidak dilakukan dengan baik dan pendedahan sandaran *(exposure of backup)* atau media yang digunakan sebagai storan kepada sandaran seperti cakera keras dan pita terdedah kepada kecurian.

Projek Keselamatan Aplikasi Web Terbuka (OWASP 2017) pula menyenaraikan ancaman terhadap keselamatan aplikasi web; antaranya ialah pemecahan pengesahan *(broken authentication)* atau penggodaman proses memastikan identiti pengguna dan *cross-site scripting (XSS)* yang membenarkan penggodam melaksanakan skrip yang berbahaya di pelayar web pengguna. Suntikan SQL turut disenaraikan dan begitu juga dengan pemecahan kawalan akses *(broken access control)* yang merujuk kepada kawalan akses kepada kandungan tertentu diberikan kepada pihak yang tidak sepatutnya.

Melalui maklumat daripada kedua-dua sumber di atas, terdapat persamaan di antara kedua-dua ancaman iaitu suntikan SQL. Suntikan SQL merupakan cara yang licik bagi penggodam untuk mendapatkan maklumat yang lebih lanjut mengenai pangkalan data supaya eksploitasi dapat dilakukan. Penggodam memainkan peranan seperti seorang pengguna dengan menaip input di borang yang terdapat di aplikasi web. Namun begitu input tersebut mengandungi kod hasad hasil daripada pengetahuan yang luas dalam bahasa pengaturcaraan SQL. Melalui input yang direka khas untuk mencari kelemahan suntikan SQL, penggodam akan memanipulasi arahan SQL tersebut sehingga mendapat maklumat yang diingininya.

1.1. Serangan Suntikan SQL

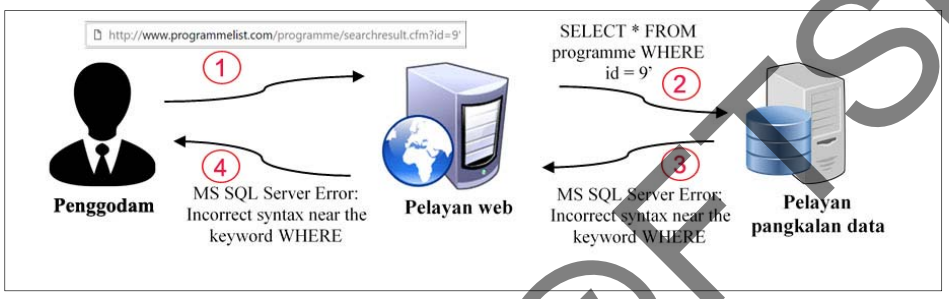
Structured Query Language (SQL) ialah bahasa pengaturcaraan standard yang digunakan untuk mengakses pangkalan data. Suntikan SQL pula ialah tindakan memasukkan kod hasad sebagai input kepada aplikasi web dan dihantar ke pangkalan data untuk melaksanakan pelbagai arahan (Clarke 2012). Serangan berlaku apabila penggodam mengeksploitasi suntikan SQL bagi melaksanakan arahan yang tidak dibenarkan. Antara contoh eksploitasi yang sering dilakukan ialah mencuri maklumat sulit yang boleh membawa keuntungan seperti nombor kad kredit, nombor akaun simpanan bank, kata laluan, rekod transaksi perniagaan dan rekod perubatan. Selain itu, eksploitasi juga boleh melibatkan pengubahsuaian data. Contohnya; pelajar yang cuba mengubah gred atau markah peperiksaan. Ada juga eksploitasi yang tidak bersifat peribadi; seperti sabotaj ke atas pangkalan data dengan memadam jadual tertentu dengan sengaja, penutupan operasi pangkalan data dan mengganggu trafik rangkaian.

Istilah 'suntikan' digunakan kerana kod hasad dimasukkan ke dalam pernyataan SQL tersebut. Dalam erti kata lain, pangkalan data melaksanakan arahan tersebut kerana ia tidak dapat membezakan antara pernyataan SQL dan input yang tidak sah. Penggodam memasukkan kod SQL yang mengandungi perintah yang diingini dan mendapat hasilnya kerana perintah tersebut adalah sah daripada segi sintaks dan aturan kod SQL. Istilah serangan dalam konteks suntikan SQL pula difahami sebagai akses tidak sah kepada aplikasi atau sistem, yang didapati melalui mekanisme suntikan SQL yang didapati daripada pengubah suaian pertanyaan SQL (Pullagura & Gokilavani 2014).

Aplikasi web yang mempunyai ruangan input pengguna boleh diserang suntikan SQL seperti yang dinyatakan di atas kerana kod aplikasi tersebut mempunyai kelemahan tertentu yang mendedahkannya kepada serangan. Kelemahan dalam sistem aplikasi didefinisikan sebagai pepijat, lohong, kerentanan atau kecacatan yang tidak disengajakan, yang terdapat di dalam

aplikasi dan boleh disalah guna oleh pengguna secara tanpa izin untuk mendapatkan akses tidak terhad kepada data yang disimpan (Kindy, & Pathan 2013). Kelemahan ini terjadi kerana aplikasi web tersebut tidak mengesahsahih terlebih dahulu parameter input daripada pengguna (Deepa & Thilagam 2016; Jumaa & Omar 2017; Mahapatra & Khan 2012; Minhas & Kumar 2013; Nagpal Chauhan & Singh 2015; Namdev, Hasan & Shrivastav 2012; Shehu, Xhuvani & Ahmetaj 2012; Tajpour, Massrum & Heydari 2010)

Melalui pelayar web, sesebuah aplikasi boleh dikenal pasti mempunyai kelemahan terhadap suntikan SQL dengan mudah (Clarke 2012).Seperti yang dapat dilihat dalam Rajah 1.1, alamat URL aplikasi web memaparkan parameter 'id' yang menerima input jenis angka integer. Penggodam seterusnya akan melaksanakan aktiviti berikut:



Rajah 1.1 Aliran maklumat ketika berlaku ralat akibat suntikan SQL (Clarke 2012).

1) Meletakkan tanda petikan tunggal (') selepas angka parameter tersebut dan menekan kekunci 'ENTER'.

2) Pelayan web menghantarkan pertanyaan SQL SELECT * FROM programme WHERE id = 9' kepada pelayan pangkalan data. Pertanyaan SQL ini pada asalnya bertujuan menyenaraikan semua data daripada semua lajur di dalam jadual yang bernama programme; yang mana lajur idnya bernilai 9.

3) Pangkalan data mendapati pertanyaan SQL tersebut tidak dapat diproses kerana adanya tanda petikan tunggal (') yang tidak sepatutnya dilakukan. Pelayan pangkalan data berkemungkinan akan melakukan satu daripada dua perkara ini iaitu; menghantar mesej ralat yang tersuai atau menghantar mesej ralat sintaks secara terus daripada pangkalan data.

4) Sekiranya mesej ralat tersuai dihantar, maka halaman tersebut tidak mempunyai kelemahan terhadap suntikan SQL. Sekiranya mesej ralat sintaks atau apa-apa mesej ralat dihantar secara terus daripada pangkalan data, maka halaman tersebut dikategorikan mempunyai kelemahan untuk dieksploitasi suntikan SQL. Penggodam seterusnya akan dapat merangka strategi serangan dengan lebih mudah apabila mengetahui aplikasi web ini mempunyai kelemahan tersebut.

Setelah mendapati aplikasi web mengandungi kelemahan, penggodam boleh melakukan apa sahaja tindakan untuk memanipulasi data yang tersimpan dalam pangkalan data seterusnya menjejaskan kerahsiaan dan integriti data, serta menyebabkan kekeliruan pengesahan dan aras keizinan pengguna.

1.2. Pernyataan Masalah

Kebanyakan organisasi tidak menyedari bahawa aplikasi web mereka terdedah kepada suntikan SQL ataupun telah diserang oleh suntikan SQL (Vikholm, & Flodström 2013). Kesedaran dan pengetahuan tentang suntikan SQL juga masih rendah, maka sebarang langkah yang diambil hanya di peringkat asas sahaja seperti pemasangan tembok api rangkaian, penggunaan *Secure Socket Layer (SSL)* dan kawalan akses rangkaian. Namun begitu, langkah tersebut hanya memberikan perlindungan di aras rangkaian, dan bukan di aras aplikasi (Huang et al. 2004). Xie & Aiken (2006) menyatakan; disebabkan kelemahan aplikasi web berada di dalam logik program, maka tembok api rangkaian tidak dapat melindungi sepenuhnya daripada serangan. Alazab (2016) juga bersetuju dengan menyatakan bahawa tembok api dan maklumat protokol tidak melindungi aplikasi web daripada penggodam kerana posisinya yang bertempat di belakang infrastruktur aplikasi web. Janot & Zavarsky (2008) menegaskan bahawa pendedahan aplikasi web yang bersifat universal, protokol HTTP yang mesra tembok api dan pengabaian keselamatan pangkalan data berikutan kebergantungan kepada keselamatan pelayan juga meningkatkan risiko serangan. Melalui pernyataan ini, dapat disimpulkan bahawa satu mekanisme pertahanan khas di aras aplikasi perlu diwujudkan bagi memberi perlindungan kepada pangkalan data.

Walau bagaimanapun, pencegahan di aras aplikasi dan pangkalan data sering tidak diberi perhatian kerana ramai menyangkakan ancaman keselamatan hanya wujud di aras rangkaian sahaja. Dengan adanya pengetahuan tentang ancaman suntikan SQL, organisasi perlu memastikan jurang kawalan keselamatan di aras aplikasi dan pangkalan data diatasi. Agensi Kelayakan Malaysia (MQA) yang dijadikan domain kajian ini juga masih tidak mempunyai kaedah yang khas untuk mencegah serangan suntikan SQL di aras aplikasi dan pangkalan data. Perkara ini disahkan oleh Ketua Penolong Pengarah Unit Pembangunan Sistem di organisasi tersebut. Pada masa ini, keselamatan kod aturcara dan pangkalan data hanya bersandarkan kepada langkah keselamatan asas yang lebih tertumpu di aras rangkaian seperti tembok api rangkaian, kawalan akses pangkalan data dan saringan permintaan pelayan web. Kesemua langkah tersebut masih tidak mencukupi dan tidak melindungi kod aturcara dan pangkalan data sepenuhnya. Ini juga disebabkan langkah keselamatan tersebut bertujuan mengatasi masalah lain, dan bukannya masalah suntikan SQL.

Berdasarkan kajian kesusasteraan, pencegahan di aras pangkalan data iaitu prosedur tersimpan merupakan satu kaedah yang kerap digunakan, namun perlu ditambah baik dengan komponen keselamatan lain iaitu pertanyaan berparameter. Ini adalah bagi mengurangkan kelemahan yang disebabkan oleh pertanyaan SQL dinamik yang berfungsi menjalankan logik fungsian seperti carian data, log masuk pengguna dan sebagainya. Manakala pencegahan di aras aplikasi iaitu pengesahsahihan input bertujuan menghalang kod hasad memasuki pelayan pangkalan data; dan pengendalian ralat bertujuan menyembunyikan maklumat sensitif pangkalan data daripada dipaparkan oleh mesej ralat lalai.

Ketiadaan kaedah pencegahan di aras aplikasi dan pangkalan data di organisasi dan penemuan kelemahan kaedah pencegahan yang dilakukan oleh umum ini menjadi motivasi bagi kajian. Terdapat keperluan untuk menambah baik kaedah prosedur tersimpan dengan menggabungkan pertanyaan berparameter sebagai mekanisme pencegahan suntikan SQL di aras pangkalan data dan membangunkan pengesahsahihan input dan pengendalian ralat sebagai mekanisme pencegahan di aras aplikasi.

2. LIPUTAN KESUSASTERAAN

2.1. Kaedah Pencegahan Suntikan SQL

Langkah meningkatkan keselamatan aplikasi web dan pangkalan data boleh dilakukan di aras pelayan, rangkaian, pangkalan data dan aplikasi. Para pengkaji sependapat mengatakan kawalan keselamatan di semua aras amat membantu; sekiranya tidak berjaya menghalang semua serangan, sekurang-kurangnya dapat membantu melambatkan dan menyukarkan usaha penyerang. Walau bagaimanapun, kebanyakan organisasi hanya mengambil langkah keselamatan yang paling asas iaitu di aras pelayan dan rangkaian sahaja. Ini adalah kerana aras pelayan dan rangkaian adalah aras fizikal yang juga adalah aset yang tertakluk kepada prosedur keselamatan yang telah ditetapkan oleh dasar-dasar yang sedia ada. Sebagai contoh, organisasi dalam sektor awam perlu mematuhi Dasar Keselamatan ICT yang telah menggariskan peraturan tertentu bagi kawalan capaian internet, penggunaan tembok api, tetapan konfigurasi pelayan, kawalan akses pengguna dan sebagainya.

Dengan menjadikan saranan para pengkaji lepas sebagai panduan, kajian ini menumpukan kepada pendekatan memperbaiki kod aturcara aplikasi web sebagai pencegahan daripada serangan suntikan SQL. Kod aturcara bagi aplikasi web yang dikaji mengandungi pernyataan SQL dinamik bagi menjalankan fungsi carian. Oleh itu, teknik pertanyaan berparameter perlu digunakan bagi menggantikan pernyataan SQL dinamik. Selain daripada itu, kajian ini mendapati pertanyaan berparameter boleh digunakan bersekali dengan prosedur tersimpan bagi meningkatkan keselamatan kod aturcara aplikasi kerana penggunaan prosedur tersimpan dapat menambahkan satu lagi lapisan abstraksi sebelum interaksi dengan pangkalan data. Dengan menempatkan fungsi carian di dalam prosedur tersimpan, aplikasi web menjadi lebih selamat kerana logik perniagaan tidak terdapat di dalam kod aturcara.

Teknik pengesahsahihan parameter input digunakan kerana bagi memastikan parameter input yang dihantar ke pelayan pangkalan data tidak mengandungi kod hasad. Dengan cara ini, kod hasad tidak akan dapat memasuki dan membahayakan prosedur tersimpan. Teknik ini juga bersesuaian dengan aplikasi web yang dikaji kerana aplikasi web tersebut masih belum mewujudkan proses pengesahsahihan input. Begitu juga dengan pengendalian ralat; ini kerana aplikasi web yang dikaji didapati sering mengemukakan mesej ralat lalai yang boleh mendedahkan maklumat sulit pangkalan data. Jadual 2.1 berikut memberikan perbandingan di antara kaedah pencegahan suntikan SQL di kedua-dua aras aplikasi dan pangkalan data yang telah dikaji.

Jadual 2.1 Perbandingan teknik pencegahan suntikan SQL.

Bil	Kaedah yang dicadangkan oleh kajian lepas	Isu	Kesesuaian dengan aplikasi web kajian kes
1	Pertanyaan berparameter	Melibatkan pengubahsuaian kod; maka hanya sesuai bagi aplikasi berskala kecil	√
2	Pengesahsahihan input	Pengesahsahihan di pihak klien masih boleh dipintas; maka perlu turut dibuat di pihak pelayan	√
3	Pengendalian ralat	Perlu dilakukan secara konsisten	√
4	Tembok api aplikasi web	Masih boleh dipintas	×
5	Prosedur tersimpan	Masih boleh disuntik sekiranya menggunakan SQL dinamik	√

6	Tembok api pangkalan data	Tidak terdapat dalam sumber terbuka bagi pangkalan data MS SQL Server	×
7	Analisis statik dan dinamik	Mempengaruhi masa tindak balas aplikasi web kepada pengguna kerana menggunakan masa pemprosesan yang tinggi.	×

Melalui jadual ini, kelebihan dan kekurangan setiap teknik dikenal pasti bagi membantu memetakan kaedah yang sesuai dengan keperluan aplikasi web yang digunakan sebagai kajian kes. Disebabkan aplikasi web yang dijadikan kajian kes menggunakan pangkalan data proprietari iaitu Microsoft SQL Server, maka tembok api pangkalan data tidak dipilih sebagai kaedah pencegahan dalam kajian ini. Tembok api pangkalan data versi sumber terbuka hanya terdapat untuk pangkalan data daripada jenis sumber terbuka juga. Teknik tembok api web aplikasi (WAF) juga tidak dipilih disebabkan keadaan WAF masih boleh dipintas oleh penggodam. Panduan memintas WAF juga banyak tersedia di ruangan forum dan tutorial penggodam di internet. Dengan menggunakan WAF, pengaturcara mungkin menganggap aplikasi web telah selamat tanpa menyedari bahawa fitur saringan WAF masih boleh dilolosi. Kajian lampau juga menyarankan agar organisasi tidak bergantung sepenuhnya kepada kedua-dua teknik ini tanpa memperbaiki kod aturcara aplikasi yang jelas mempunyai kelemahan.

Teknik analisis statik dan dinamik juga tidak sesuai dengan aplikasi web yang dikaji kerana bakal menyebabkan masa tindak balas aplikasi web kepada pengguna meningkat. Sebagai organisasi yang sentiasa berurusan dengan pengguna melalui aplikasi web, masa tindak balas yang pantas adalah penting. Selain daripada itu, aplikasi web yang dikaji juga adalah berskala kecil, maka tidak ada keperluan untuk menggunakan teknik pencegahan automatik.

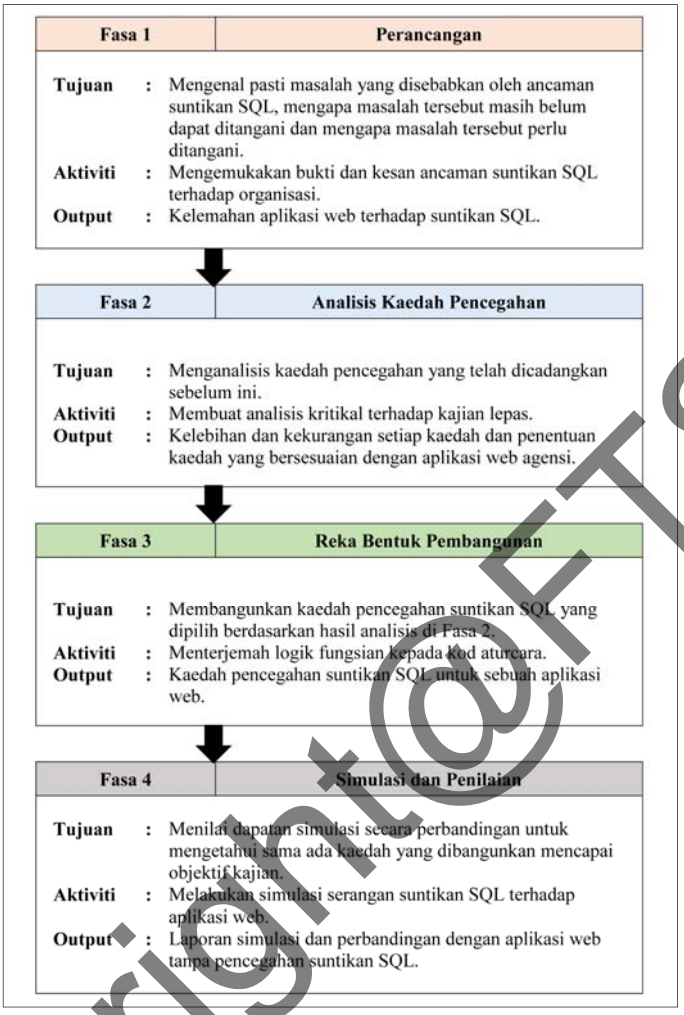
3. METODOLOGI

3.1. Metodologi Kajian

Kajian ini dilakukan secara kuantitatif kerana keberkesanan kaedah pencegahan hanya dapat diperhatikan melalui keputusan pengujian menggunakan perisian komputer. Pendekatan yang digunakan adalah secara simulasi eksperimen dan aktiviti yang dijalankan sepanjang kajian dalam mencapai objektif yang ditetapkan adalah seperti yang ditunjukkan dalam Rajah 3.1. Kesemua aktiviti kajian dibahagikan kepada empat fasa iaitu perancangan, analisis kaedah pencegahan, reka bentuk pembangunan serta simulasi dan penilaian.

3.2. Penyediaan Aplikasi Web Semu

Dalam fasa ini, aplikasi web yang menjadi kajian kes diwakili oleh aplikasi web versi semu. Aplikasi web semu dibangunkan untuk membuktikan wujudnya kelemahan kepada suntikan SQL dan seterusnya menjadi medium pengujian kaedah pencegahan. Aplikasi web semu dianggap sebagai replika asas kepada aplikasi web asal kerana dibangunkan dengan mengandungi kod aturcara dan logik fungsian yang sama dengan aplikasi web asal. Walau bagaimanapun, semua nama lajur, nama jadual, nama pangkalan data dan nama parameter telah diubah untuk melindungi maklumat sulit organisasi. Untuk melancarkan proses yang bakal dijalankan, aplikasi web semu hanya mengandungi halaman yang berkaitan sahaja iaitu halaman carian. Penampilan antara muka aplikasi web semu juga diringkaskan dan diubah kepada kandungan paling minimum.



Rajah 3.1 Metodologi kajian

3.3. Penggunaan SQLMap Sebagai Pengesan Kelemahan

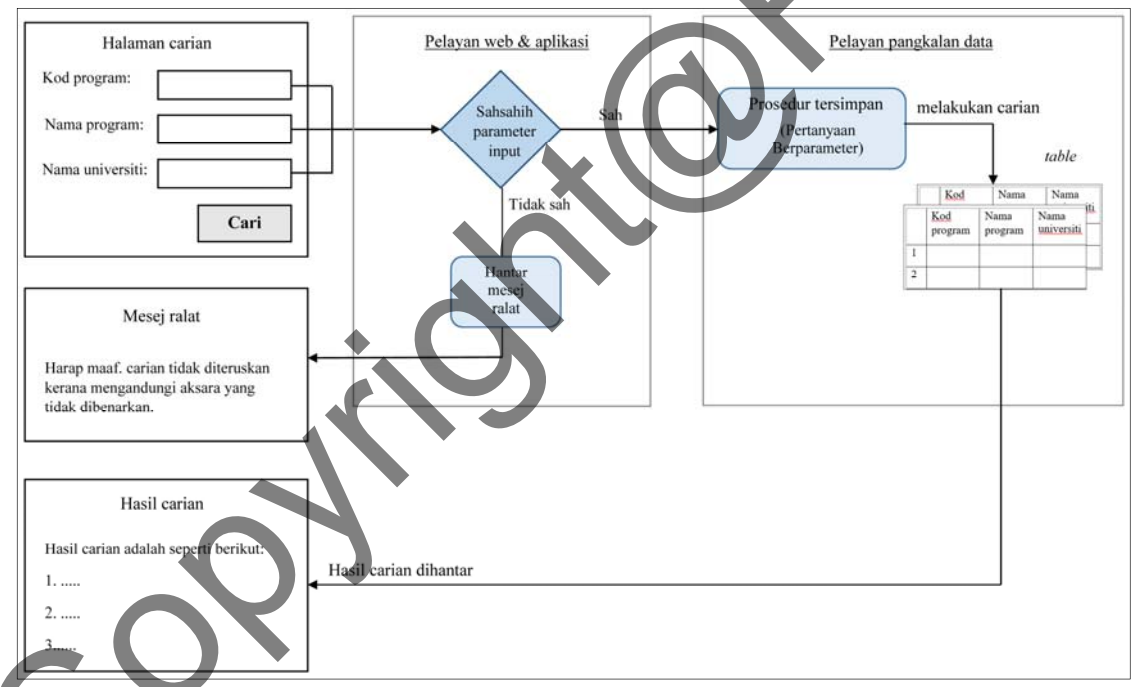
SQLMap dipilih sebagai instrumen pengimbas kelemahan suntikan SQL dalam kajian ini kerana ketersediaannya sebagai perisian sumber terbuka. Perisian sumber terbuka ini boleh dimuat turun dengan percuma daripada internet tanpa sebarang kos. Kedua, instrumen ini bersifat automatik dan dapat menjalankan proses mengesan kelemahan aplikasi web dalam tempoh yang singkat. Ketiga, instrumen ini adalah paling efektif dan popular di kalangan penggodam dan penguji penembusan aplikasi web menurut kajian yang dilakukan oleh (Zhu 2016).

Melalui fungsinya yang automatik, kerja-kerja mengesan kelemahan dapat dipermudahkan dan dipercepatkan. Projek Terbuka Keselamatan Aplikasi Web (OWASP 2017) menyarankan kepada para pengaturcara untuk mengaudit aplikasi web masing-masing sebagai salah satu langkah berjaga-jaga sebelum mendedahkan aplikasi web untuk kegunaan umum. Kelemahan-kelemahan yang dikesan menggunakan alatan ini seterusnya menjadi panduan kepada pengaturcara untuk memperkuatkan pertahanan aplikasi web daripada serangan suntikan SQL.

3.4. Pembangunan Kaedah Pencegahan Suntikan SQL

Prosedur tersimpan merupakan kaedah utama yang dibangunkan dan mengandungi pertanyaan berparameter. Pertanyaan berparameter ialah pertanyaan SQL yang mempunyai satu atau lebih parameter terbenam di dalamnya. Parameter dihantar kepada pertanyaan ini sebaik sahaja pengguna menekan butang 'Hantar' atau menekan pautan di aplikasi web. Parameter yang mengandungi input pengguna yang terbenam tidak akan ditafsirkan sebagai perintah untuk dilaksanakan, maka tidak akan membenarkan kod disuntik oleh suntikan SQL. Pertanyaan berparameter merupakan ciri keselamatan kod aturcara yang digunakan sekiranya terdapat keperluan untuk membina pertanyaan SQL yang dinamik. Dengan penggantian pertanyaan SQL dinamik kepada pertanyaan berparameter, kod aplikasi menjadi lebih selamat kerana logik fungsian ditakrifkan terlebih dahulu dan parameter input dimasukkan selepas logik fungsian diproses.

Manakala pengesahsahihan parameter dan pengendalian ralat merupakan ciri tambahan untuk membantu mengurangkan kelemahan terhadap suntikan SQL. Reka bentuk kaedah secara keseluruhan telah diterjemahkan di dalam Rajah 3.4 bagi menunjukkan aliran kerja yang berlaku ketika kaedah ini digunakan.



Rajah 3.4 Reka bentuk pelaksanaan prosedur tersimpan, pengesahsahihan parameter dan pengendalian ralat untuk mencegah suntikan SQL.

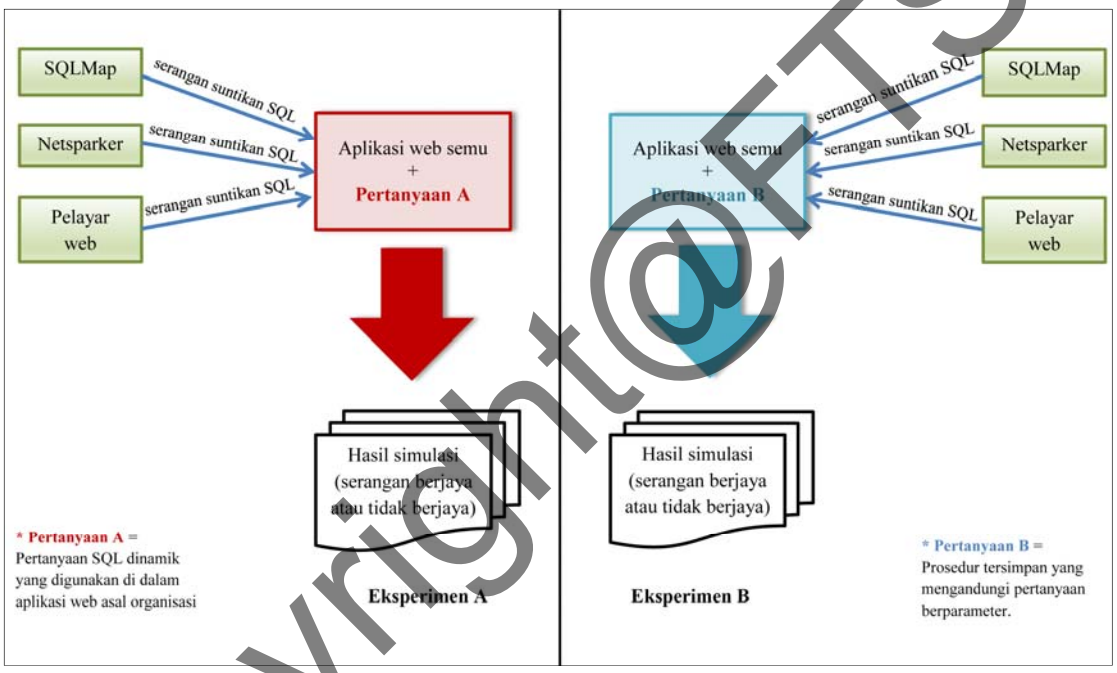
3.5. Perisian dan Perkakasan

Persekitaran simulasi dijalankan di dalam komputer yang menggunakan sistem pengoperasian Microsoft Windows 7 Enterprise Edition, 4GB RAM, cip pemprosesan Intel(R) Core(TM) i5-2320 CPU @ 3.00GHz. Aplikasi web semu dibangunkan menggunakan menggunakan pangkalan data Microsoft SQL Server 2008. Sistem pengoperasian pelayan web yang digunakan ialah Windows 2008 R2 dengan teknologi aplikasi web ASP.Net, Coldfusion dan Microsoft IIS 7.5,

dengan tetapan alamat http://localhost/programme/search.cfm atau http://127.0.0.1/programme/search.cfm. Prosedur tersimpan ditulis dalam bahasa Transact-SQL di dalam Microsoft SQL Server. Perisian sumber terbuka yang digunakan dalam simulasi ialah SQLMap yang berfungsi bersama-sama *library* Python dan perisian Netsparker Trial Edition.

3.6. Simulasi Serangan Dalam Persekitaran Semu

Simulasi dijalankan dalam dua versi iaitu simulasi serangan terhadap aplikasi web dengan pertanyaan SQL dinamik dan simulasi serangan terhadap aplikasi web dengan prosedur tersimpan. Kedua-dua eksperimen menggunakan alatan dan perisian yang sama iaitu SQLMap, Netsparker dan pelayar web. Setelah simulasi dilakukan, setiap output yang dijana direkodkan melalui paparan skrin perisian tersebut atau skrin aplikasi web semu. Kedua-dua versi output kemudiannya dibandingkan.



Rajah 3.6 Reka bentuk simulasi serangan suntikan SQL.

4. DAPATAN KAJIAN

4.1. Penyekatan Serangan

Jadual 4.1 Perbandingan penyekatan serangan dalam eksperimen A dan B.

Simulasi serangan	Dapatan	
	Eksperimen A	**Eksperimen B**
1) SQLMap	Serangan berjaya	Serangan tidak berjaya
2) Netsparker	Serangan berjaya	Serangan tidak berjaya
3) Pelayar web	Serangan berjaya	Serangan tidak berjaya

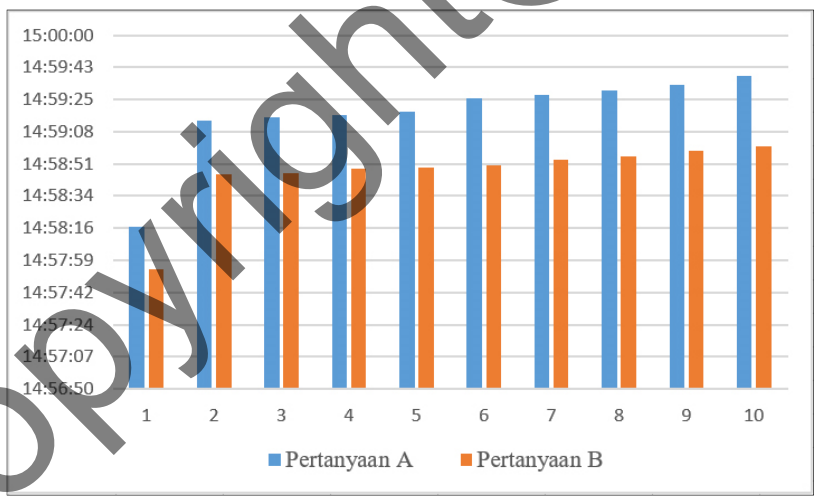
Hasil bagi kedua-dua eksperimen dicatatkan di dalam Jadual 4.1. Didapati aplikasi web yang tidak mempunyai kaedah pencegahan suntikan SQL berjaya diserang oleh suntikan SQL, manakala aplikasi web yang dilengkapi dengan prosedur tersimpan berparameter, pengendalian ralat dan pengesahsahihan input tidak membenarkan sebarang serangan berlaku. Ini membuktikan bahawa kaedah pencegahan yang dibangunkan telah dapat mempertahankan aplikasi web daripada serangan suntikan SQL.

4.2. Penilaian Tempoh Pemprosesan

Untuk menilai prestasi prosedur tersimpan berparameter, kajian ini menggunakan dua instrumen penilaian iaitu Statistik Klien Microsoft SQL Server dan SQLQueryStress. Statistik Klien merupakan kemudahan yang tersedia di dalam Microsoft SQL Server manakala SQLQueryStress merupakan alat mandiri yang didapati sebagai sumber terbuka.

4.2.1. Penilaian Menggunakan Statistik Klien MS SQL Server

Masa pelaksanaan klien ialah amaun masa kumulatif yang digunakan bagi pihak klien ketika pertanyaan SQL dilaksanakan. Bagi mendapatkan masa pelaksanaan secara purata, sepuluh percubaan dilaksanakan. Merujuk kepada Rajah 4.2.1, seperti yang dapat dilihat di setiap percubaan, Pertanyaan A menjana masa pelaksanaan yang lebih banyak berbanding Pertanyaan B. Perbezaan ini berlaku kerana prosedur tersimpan yang berada di pelayan pangkalan data melaksanakan pertanyaan SQL terlebih dahulu sebelum aplikasi web mengemukakan panggilan daripada pengguna. Oleh yang demikian, pihak klien telah dapat menjimatkan masa pelaksanaan dengan memanggil pertanyaan SQL yang telah sedia dilaksanakan di pelayan pangkalan data.



Rajah 4.2.1 Carta yang menunjukkan perbezaan masa pelaksanaan klien bagi sepuluh percubaan pelaksanaan Pertanyaan A dan Pertanyaan B.

4.2.2. Penilaian Menggunakan SQLQueryStress

Tujuan SQLQueryStress digunakan adalah untuk mengetahui impak pertanyaan SQL (sama ada pertanyaan SQL dinamik, pertanyaan berparameter atau prosedur tersimpan) terhadap prestasi sistem keseluruhan dengan mudah dan pantas. Berdasarkan jadual, didapati masa yang diambil oleh pertanyaan B adalah lebih rendah daripada masa yang diambil oleh pertanyaan A.

Jadual 4.2.2 Perbandingan kriteria SQLQueryStress ke atas aplikasi web dengan kedua-dua pertanyaan SQL dinamik (Pertanyaan A) dan prosedur tersimpan (Pertanyaan B).

Kriteria (purata)	Pertanyaan A	Pertanyaan B
Masa berlalu *(elapsed time) (ms)*	1.3600	0.7350
Saat CPU *(CPU seconds) (ms)*	0.0996	0.0167
Saat sebenar *(actual seconds) (ms)*	0.2834	0.0401
Saat klien *(client seconds) (ms)*	0.1009	0.0475
Bacaan logik *(logical reads)*	1620.0	1547.0

Selain itu, didapati pertanyaan B mempunyai bacaan logik sebanyak 1547 berbanding 1620 oleh pertanyaan A. Pertanyaan SQL yang mempunyai bacaan logik yang rendah adalah lebih efisien daripada pertanyaan SQL yang menghasilkan bacaan logik yang tinggi. Ini adalah kerana bacaan logik yang tinggi memberi implikasi negatif kepada memori sistem. Dengan erti kata lain, bacaan logik yang tinggi memberi bebanan yang lebih kepada sistem komputer. Dengan ini, disimpulkan bahawa prosedur tersimpan menghasilkan pertanyaan SQL yang lebih efisien daripada pertanyaan SQL dinamik.

5. KESIMPULAN

Aplikasi web semu yang dilengkapi dengan prosedur tersimpan berparameter telah diuji melalui tiga cara simulasi serangan iaitu menggunakan perisian SQLMap, Netsparker dan pelayar web. Dapatan simulasi memberikan kesimpulan bahawa suntikan SQL tidak berjaya menembusi aplikasi web dan pangkalan data setelah kaedah prosedur tersimpan dilaksanakan. Ini menandakan kaedah pencegahan suntikan SQL yang dibangunkan telah berjaya mencegah suntikan SQL daripada berlaku.

Kajian ini juga menilai kaedah prosedur tersimpan daripada perspektif masa yang digunakan dan impaknya kepada sistem keseluruhan. Penilaian menggunakan Statistik Klien Microsoft SQL Server dan perisian SQLQueryStress telah memberikan kesimpulan bahawa meskipun terdapat perbezaan masa yang kecil, prosedur tersimpan sentiasa menggunakan tempoh masa yang lebih singkat berbanding pertanyaan SQL dinamik. Ini adalah penting dalam memastikan kaedah pencegahan suntikan SQL yang digunakan tidak menghasilkan overhed tinggi seterusnya meningkatkan masa tindak balas aplikasi web kepada pengguna.

RUJUKAN

Alazab, A. 2016. New Strategy for Mitigating of SQL Injection Attack *154*(11), 1–10.

Clarke, J. 2012. SQL Injection Attacks and Defense. *SQL Injection Attacks and Defense*, 1–473. doi:10.1016/B978-1-59749-424-3.X0001-1

Deepa, G. & Thilagam, P. S. 2016. Securing web applications from injection and logic vulnerabilities: Approaches and challenges. *Information and Software Technology*, *74*, 160–180. doi:10.1016/j.infsof.2016.02.005

Horner, M. & Hyslip, T. 2017. SQL Injection: The Longest Running Sequel in Programming History. *Journal of Digital Forensics*, *12*(2), 10. Retrieved from http://commons.erau.edu/jdfsl/vol12/iss2/10/

Huang, Y., Yu, F., Hang, C., Tsai, C., Lee, D.-T. & Kuo, S. 2004. Securing web application code by static analysis and runtime protection. *Proceedings of the 13th conference on World Wide Web - WWW '04*, hlm.40. doi:10.1145/988672.988679

Janot, E. & Zavarsky, P. 2008. Preventing SQL Injections in Online Applications : Study , Recommendations and Java Solution Prototype Based on the SQL DOM.

Jumaa, A. & Omar, A. 2017. Online Database Intrusion Detection System Based on Query Signatures. *Journal of University of Human Development*, *3*(1), 282–287. doi:10.21928/juhd.20170315.14

Kindy, D. A. & Pathan, A. K. 2013. A Detailed Survey on various aspects of SQL Injection in Web Applications : Vulnerabilities , Innovative Attacks and Remedies. *International Journal of Communication Networks and Information Security*, *5*(2), 80–92.

Mahapatra, R. P. & Khan, S. 2012. A Survey Of Sql Injection Countermeasures *3*(3), 55–74.

Malik, M. & Patel, T. 2016. Database Security - Attacks and Control Methods. *International Journal of Information Sciences and Techniques*, *6*(1/2), 175–183. doi:10.5121/ijist.2016.6218

Minhas, J. & Kumar, R. 2013. Blocking of SQL Injection Attacks by Comparing Static and Dynamic Queries (February), 1–9. doi:10.5815/ijcnis.2013.02.01

Nagpal, B., Chauhan, N. & Singh, N. 2015. A Survey on the Detection of SQL Injection Attacks and Their Countermeasures. *Journal of Information Processing Systems*,. doi:10.3745/JIPS.03.0024

Namdev, M., Hasan, F. & Shrivastav, G. 2012. Review of SQL Injection Attack and Proposed Method for Detection and Prevention of SQLIA. *International Journal of Advanced Research in Computer Science and Software Engineering*, *2*(7), 24–28.

OWASP. 2017. OWASP Top 10 - The Ten Most Critical Web Application Security Risks. *Owasp*, 22. Retrieved from https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf%0Ahttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:OWASP+Top+10+-2010#1

Pullagura, P. S. P. & Gokilavani, A. 2014. THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE Defeating SQL Injection on Preventing Run Time Attacks Abstract : *2*(5), 93–96.

Shehu, B., Xhuvani, A. & Ahmetaj, S. 2012. Methods of Identifying and Preventing SQL Attacks. *International Journal of Computer Science Issues*, *9*(6), 403–406.

Tajpour, A., Massrum, M. & Heydari, M. Z. 2010. Comparison of SQL Injection Detection and Prevention Techniques. *International Conference on Education Technology and Computer*, 174–179.

Vikholm, O. & Flodström, M. 2013. SQL-Injections: A wake-up call for developer: A study about a major threat and issue for companies and organizations worldwide 36. Retrieved from http://uu.diva-portal.org/smash/record.jsf?pid=diva2:630946%5Cnhttp://uu.diva-portal.org/smash/get/diva2:630946/FULLTEXT01

Xie, Y. & Aiken, A. 2006. Static detection of security vulnerabilities in scripting languages. *15th USENIX Security Symposium*, 179–192. Retrieved from http://www.usenix.org/event/sec06/tech/full_papers/xie/xie_html/

Zhu, Y. C. 2016. *Exploring Defense of SQL Injection Attack in Penetration Testing*.