JURNAL DASI ISSN: 1411-3201
Vol. 11 No. 3 September 2010

PERBANDINGAN SISTEM KEAMANAN PENGEMBANGAN APLIKASI *WEBSITE* WEB 2.0 MENGGUNAKAN *FRAMEWORK* RUBY ON RAILS DAN CAKEPHP

Ema Utami, Sahid
STMIK AMIKOM Yogyakarta

Abtraksi

Web 2.0 merupakan istilah yang digunakan untuk menunjukkan berbagai layanan di web yang memungkinkan pemakai untuk berkolaborasi dan berbagi informasi secara online. Teknologi Web 2.0 menggunakan teknologi yang sejak lama dikembangkan untuk web, ditambah dengan berbagai teknologi yang memungkinkan kolaborasi serta berbagai informasi. Sehingga secara logis bisa dikatakan bahwa Web 2.0 akan mempuyai masalah keamanan yang sama dengan aplikasi web sebelumnya bahkan lebih. Pembuatan aplikasi website dapak dikatakan mudah atau sulit, disatu sisi bahasa pemrogramannya mudah dipelajari namun disisi lain kemudahan tersebut sering membuat web developer terlena. Faktor keamanan menjadi syarat mutlak dipenuhi dalam aplikasi Web 2.0. Tetapi tidak semua developer menyadari hal tersebut. Bahkan ada yang membangun aplikasi website tanpa di-optimasi dan tidak memperhitungkan keamanan data

Akibatnya aplikasi website tersebut mudah dirusak oleh pihak-pihak yang tidak bertanggung jawab. Tapi, ada pendekatan baru dalam pemrograman website yang bisa meng-handle masalah-masalah tersebut. Salah satunya adalah pemrograman web berbasis framework. Tulisan ini memaparkan lebih dalam tentang perbedaan sistem keamanan dalam pengembangan aplikasi web berbasis framework antara RoR dan CakePHP.

Hasil dari penelitian ini diharapkan dapat digunakan sebagai bahan pertimbangan bagi developer yang sudah atau akan menggunakan salah satu dari framework RoR dan CakePHP untuk membangun sebuah aplikasi website berbasis framework.

Keywords: *Web 2.0, framework, RoR, CakePHP*

Pendahuluan

Framework merupakan sekumpulan *library* yang diorganisasikan pada sebuah rancangan arsitektur untuk memberikan

ketepatan, kecepatan, kemudahan dan konsistensi dalam pengembangan aplikasi. *Framework* menyediakan berbagai *library* untuk pengembangan aplikasi *website* secara tepat dan mudah dengan tidak mengurangi dari segi keamanan *website* tersebut. Terdapat banyak *framework* untuk aplikasi *website* yang dikembangkan sesuai dengan bahasa pemrograman yang digunakan. Contoh *framework* untuk aplikasi *website* antara lain dalah Ruby on Rails (Ruby), CodeIgniter (PHP), Symphony (PHP), CakePHP (PHP), Zope (Python), DotNetNuke (VB.NET), Apache Struts (Java), Spring (Java), dan lain sebagainya.

Salah satu dari sekian banyaknya *framework* untuk aplikasi *website* tersebut adalah Ruby on Rails (RoR). RoR merupakan *framework* Ruby berbasis open source untuk pengembangan aplikasi web (*web application framework*). RoR versi *beta* pertama dilepaskan kepada publik pada bulan juli 2004. RoR menerapkan secara penuh pola perancangan MVC (Model View Controller) dan juga dilengkapi dengan dukungan Ajax, *internationalization /localization*, dan ORM (Object Relational Model). Pengembangan suatu aplikasi web dengan menggunakan *framework* Ruby on Rails dapat dilakukan secara lebih cepat, mudah dan aman dibandingkan dengan *framework* lainnya. Seorang *web developer* dapat menulis program menggunakan RoR dengan sedikit pengkodeaan, dengan menjaga ukuran kode program tetap kecil maka dapat mempercepat pengembangan aplikasi dan memperkecil *bug* yang akan membuat kode program para *developer* mudah dipahami, dipelihara dan ditingkatkan. Kesuksesan RoR ini kemudian menginspirasi para *developer* PHP untuk membangun *framework* serupa dalam bahasa *scripting* PHP. Kemudian, para *developer* PHP menciptakan *framework* CakePHP dan pertama diluncurkan pada tahun 2006 sebagai adaptasi dari *framework* RoR.

Framework RoR dan CakePHP banyak dikaji secara khusus untuk dibandingkan. Kajian yang dilakukan banyak membandingkan dari segi kinerja (*performance*), metodologi dan keamanan (*security*). Pada penelitian sebelumnya menghasilkan beberapa solusi untuk mengatasi ancaman pada sistem keamanan *framework* RoR semisal,

solusi untuk ancaman *SQL Injection* yaitu menghindari mekanisme *gererate* SQL berbasis *user-controller variables* (Hendrayana, 2007;5). Tetapi, pada penelitian tersebut mempunyai beberapa kelemahan. Salah satunya, kurangnya solusi untuk ancaman *Cross Site Scripting (XSS)*. Sehingga menimbulkan ketertarikan peneliti untuk melakukan penelitian pada *framework* tersebut. Karena kesuksesan RoR mengispirasi para *developer* PHP membuat *framework* CakePHP. Maka penelitian ini diadakan untuk mengkaji ulang lebih dalam mengenai perbedaan sistem keamanan pengembangan aplikasi website antara *framework* RoR dan CakePHP. Penelitian ini akan dilakukan dengan studi literature sebagai kegiatan utama dan pembangunan dua buah *prototype* aplikasi *website* Web 2.0 untuk masing-masing *framework* sebagai penguat dari hasil analisa literatur. Hasil akhir dari penelitian ini diharapkan dapat diketahui mana yang terbaik diantara *framework* RoR dan CakePHP dalam segi sistem keamanan pengembangannya.

Pembahasan

SQL Injection

SQL injection merupakan serangan dengan memanfaatkan cacat programming pada aplikasi website untuk mengeksploitasi database server. SQL injection biasanya dilakukan ketika memasukkan input melalui form dengan perintah atau kode tertentu. Pihak tidak bertanggung jawab bisa menginputkan meta karakter pada inputan form tersebut kemudian dieksekusi oleh website dan dikirim ke database. Database server memproses meta karakter tersebut sebagai *query* yang wajar. Proses SQL injection biasanya memanfaatkan meta karakter dan logika SQL sehingga pihak tidak bertanggung jawab bisa memodifikasi query SQL. Akibat SQL injection biasanya berupa pencurian *record* database yang cukup *confidential* seperti username, password, email, bahkan nomor kartu kredit. Selain itu SQL injection juga bisa dimanfaatkan untuk mencuri login admin dan melakukan *take over* sepenuhnya.

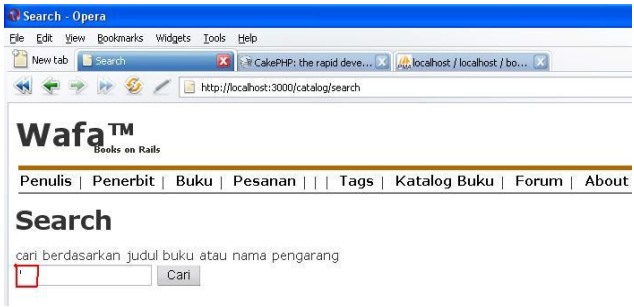
Pada tahap pengujian SQL Injection pada *framework* Ruby on Rails dan CakePHP dilakukan dengan dua cara yaitu pengujian dengan menambahkan logika AND pada URL (*Uniform Resource Locator*) dan mengisikan tanda quotes (‘) *pada form search.* Pengujian pertama dilakukan pada *framework* Ruby on Rails dengan penambahan logika AND pada *Url forum*. Contoh penambahan logika AND: `http://localhost:3000/forum/show/9 AND 1=0--`

Gambar 1 Pengujian Penambahan Logika AND pada RoR

Logika AND 1=0-- memberikan nilai False, sehingga browser tidak menampilkan hasil (*blank*). Setelah pengujian *framework* Ruby on Rails selesai diikuti pengujian kedua yaitu pengujian *sql injection* dengan memanfaatkan *URL forum* pada *framework* CakePHP. Pengujian masih menggunakan logika AND dengan tujuan supaya hasil yang didapat dapat sama. Contoh penambahan logika AND: `http://localhost/AplikasiCakePHP/application/posts/view/3 AND 1=0--`

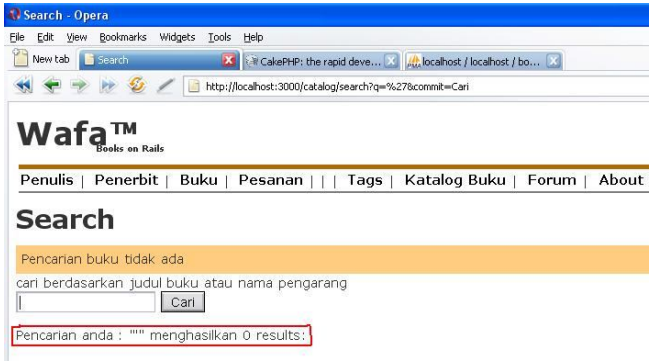
Gambar 2 Pengujian Penambahan Logika AND Pada CakePHP

Dari hasil pengujian dengan manfaatkan logika AND pada URL forum Ruby on Rails dan CakePHP dapat dikatakan bahwa *framework* tersebut tidak *vulnerable* terhadap serangan SQL *injection*. Untuk itu dilakukan pengujian kedua dengan mengisikan tanda quates (‘) pada *form search* supaya mendapatkan hasil yang lebih baik. Pengisian tanda *single quote* (‘) dilakukan di *form search.* Pengujian pertama dilakukan pada *framework* Ruby on Rails. Contoh penambahan *single quote* (‘) pada aplikasi framework Ruby on Rails :



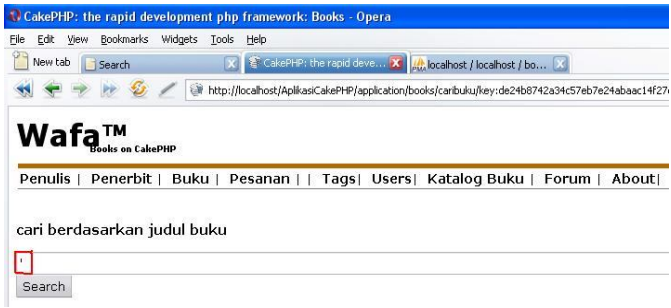
Gambar 3 Pengujian Penambahan Single Quote pada RoR

Penambahan *single quote* (‘) ditandai dengan warna merah, hasil penambahan *single quote* (‘) pada *framework* Ruby on Rails adalah terlihat seperti gambar 4 dibawah ini :



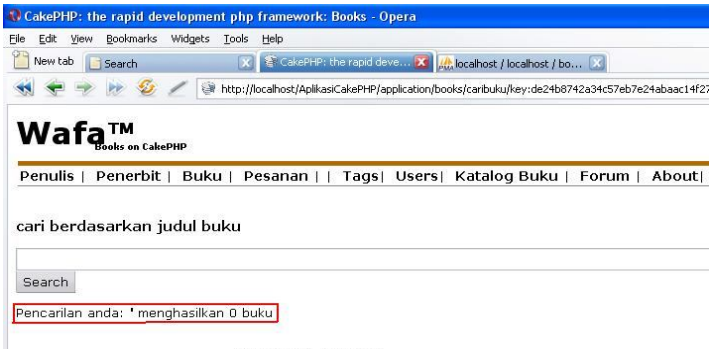
Gambar 4 Hasil Pengujian Penambahan Single Quote pada RoR

Pengujian kedua dilakukan pada *framework* CakePHP, langkah penambahan *single quote* (‘) sama seperti pengujian diatas yaitu terlihat pada gambar:



Gambar 5 Pengujian Penambahan Single Quote pada CakePHP

Dan setelah proses pencarian dilakukan, maka didapat hasil seperti terlihat pada gambar 6 dibawah ini:



Gambar 6 Hasil Pengujian Penambahan Single Quote pada CakePHP

Pengujian SQL injection dengan menggunakan logika AND dan penambahan single quote pada kedua *framework* Ruby on Rails dan CakePHP sudah dilakukan, maka dari hasil pengujian diatas dapat disimpulkan bahwa kedua *framework* tersebut tidak *vulnerable* terhadap serangan SQL *injection*. Dari hasil penelitian, *framework* Ruby on Rails dan CakePHP sudah memprotek serangan SQL injection dengan beberapa metode sesuai dengan system keamanan masing-masing :

Tabel 1 Tabel hasil analisis perbandingan SQL injection berdasarkan studi literatur

Framework	
Ruby on Rails	CakePHP
RoR memprotek serangan SQL injection dengan secara otomatis memfilter karakter khusus sql seperti tanda petik satu(‘), petik dua (“), karakter NULL dan line break (-). Tetapi terkadang juga harus diterapkan secara manual pada setiap penggalan (*fragment*) SQL, terutama dalam kondisi string	CakePHP sudah memprotek serangan SQL *injection* dengan menggunakan *Sanitize class*, *Sanitize* menyediakan empat metode untuk menerapkan berbagai tingkat keamanan data (Duane O’Brien, 2009) 1. *The sanitize paranoid method*

(*:conditions =>"…"*), *connection.execute()* atau fungsi *find_by_sql()*. Tidak dianjurkan untuk menambahkan string (*string1 + string2*), atau pada mekanisme konvensional Ruby untuk menggantikan string (*#{….}*) (Heiko Webers, 2008). Dan juga Hindari mekanisme generate SQL berbasis *user-contorller variables* (Hendrayana, 2007)	2. *The sanitize escape method* 3. *The sanitize html method* 4. *The sanitize clean method*

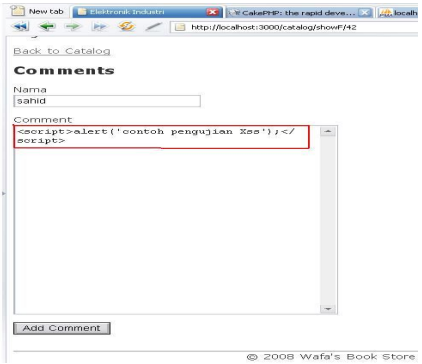
Cross Site Scripting

Cross site scripting adalah kelemahan keamanan yang terjadi pada penggunaan teknologi dynamic page. Cross site scripting dapat diartikan sebagai kelemahan yang terjadi akibat ketidakmampuan server dalam memvalidasi input yang diberikan oleh pengguna. Algoritma, yang digunakan untuk pembuatan halaman yang diinginkan, tidak mampu melakukan penyaringan terhadap masukan tersebut. Hal ini memungkinkan halaman yang dihasilkan menyertakan perintah yang sebenarnya tidak diperbolehkan. *Cross site scripting* bekerja menipu dengan kedok yang mampu mengelabui orang yang waspada. Elemen penting dari keberhasilan *cross site scripting* adalah *social engineering* yang baik dari si penipu. *Social engineering* merupakan elemen terpenting yang menentukan keberhasilan penipuan yang akan dilakukan.

Cross site scripting memampukan seseorang yang tidak bertanggung jawab melakukan penyalahgunaan informasi penting. Sebelum sampai pada proses penyalahgunaan tersebut, penyerang mengambil langkah-langkah dengan mengikuti pola tertentu. Langkah pertama, penyerang melakukakan pengamatan untuk mencari web-web yang memiliki kelemahan *crosssite scripting*. Langkah kedua, sang penyerang mencari tahu apakah web tersebut menerbitkan

informasi yang dapat digunakan untuk melakukan pencurian infomasi lebih lanjut. Informasi tersebut biasanya berupa *cookie*. Langkah kedua ini tidak selalu dijalankan. Langkah ketiga, sang penyerang membujuk korban untuk mengikuti sebuah link yang mengandung kode, ditujukan untuk mendapatkan informasi yang telah disebutkan sebelumnya. Kemampuan *social engineering* dari sang penyerang diuji disini. Setelah mendapatkan informasi tersebut, sang penyerang melakukan langkah terakhir, pencurian maupun pengubahan informasi vital (Richson Untung Tambun, 2004) .

Pengujian Cross-Site Scripting pada aplikasi Ruby on Rails dan CakePHP dilakukan dengan tujuan dapat mengetahui kelemahan server dalam memvalidasi input yang diberikan oleh pengguna. Pengujian Cross-Site Scripting dilakukan dengan menginputkan *java script* pada *form comment* dan *form search*. *Form comments* yang terdapat pada aplikasi Ruby on Rails dan CakePHP dimanfaatkan dalam pengujian *Cross-Site Scripting.* Pengujian pertama dilakukan pada *framework* Ruby on Rails dengan menginputkan bahasa *script* (*java script*) pada *form comments*. Contoh pengujian: `<script>alert('contoh pengujian Xss');</script>` Detail pengujian *Cross-Site Scripting* pada *framework* Ruby on Rails adalah :



Gambar 7 Pengujian Penginputan *Comment* pada RoR

Setelah proses penginputan *javascript* pada *form comments* dilakukan maka didapatkan hasil:

Gambar 8 Hasil Pengujian Penginputan *Comment* pada RoR

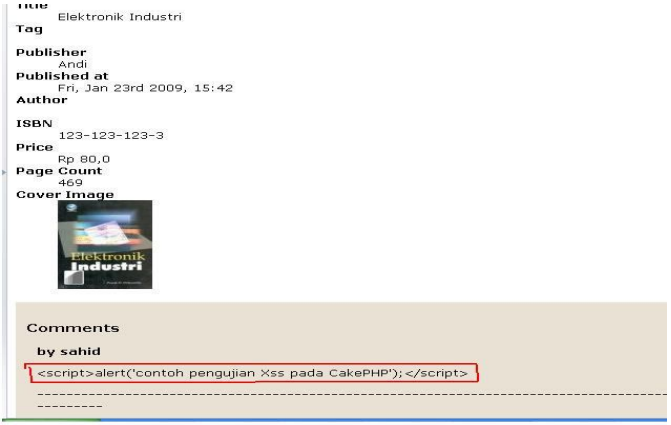
Pengujian *Cross-Site Scripting* yang kedua dilakukan pada *framework* CakePHP. Untuk mendapatkan hasil yang sama maka pengujian dilakukan dengan langkah yang sama seperti pengujian penginputan *javascript* pada *form comments* Ruby on Rails. Contoh pengujian:

```
<script>alert('contoh pengujian Xss pada CakePHP');</script>
```

Detail pengujian dapat dilihat pada gambar 9 dibawah ini :

Gambar 9 Pengujian Penginputan *Comment* pada CakePHP

Setelah proses penginputan *javascript* pada *form comments* dilakukan maka didapatkan hasil:



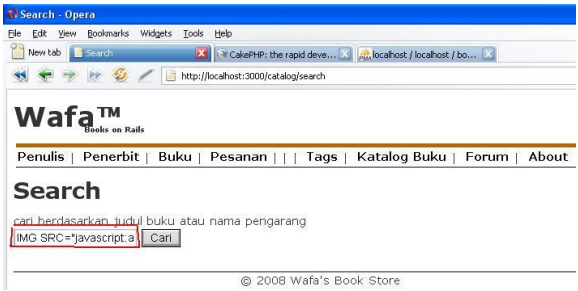
Gambar 10 Hasil Pengujian Penginputan *Comment* pada CakePHP

Pengujian pada *form comments* pada kedua aplikasi *framework* Ruby on Rails dan CakepHP sudah dilakukan, kemudian langkah selanjutnya untuk mendapatkan hasil yang lebih baik dilakukan pengujian pada *form seach*.

Pengujian *Cross-Site Scripting* pada *form search* dilakukan untuk memperkuat pengujian sistem keamanan pada *framework* Ruby on Rails dan CakePHP. Pengujian menggunakan bahasa yang sama seperti pengujian sebelumnya yaitu menggunakan bahasa script (*java script*). Pengujian pertama akan dilakukan pada aplikasi *framework* Ruby on Rails. Script yang diinputkan untuk pengujian adalah :

```
<IMG SRC="javascript:alert('Pengujian XSS Pada Ruby on Rails');">
```

Kemudian *script* diatas diinputkan kedalam *form search*. Detail pengujian *Cross-Site Scripting* pada *framework* Ruby on Rails dapat dilihat pada gambar 11 dibawah ini :



Gambar 11 Pengujian *Search* pada RoR

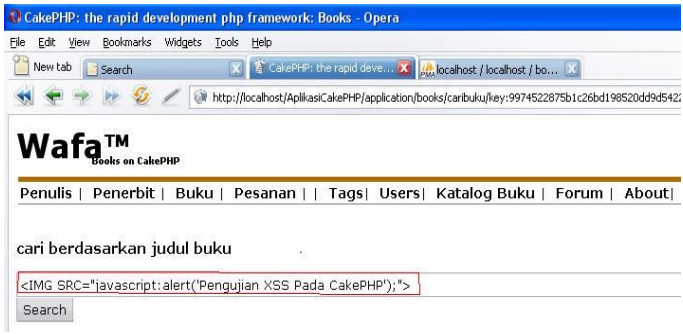
Setelah *script* dimasukkan pada *form search* kemudian dilakukan proses pencarian. Hasil dari proses pencarian dapat dilihat pada gambar 12.

Gambar 12 Hasil Pengujian *search* Pada RoR

Pengujian kedua dilakukan pada *framework* CakePHP. Untuk mendapatkan hasil yang sama maka pengujian dilakukan dengan langkah yang sama seperti proses pengujian menginputkan *search* pada RoR, yaitu :

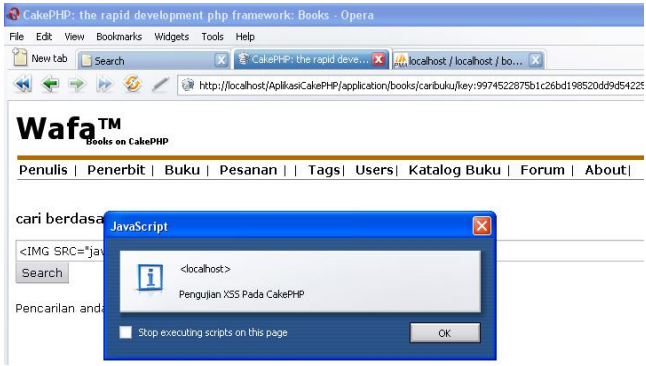
```
<IMG SRC="javascript:alert('Pengujian XSS Pada CakePHP');">
```

Kemudian *script* diinputkan kedalam *form search*. Detail pengujian *Cross-Site Scripting* pada *framework* CakePHP dapat dilihat pada gambar 13.



Gambar 13 Pengujian *Search* pada CakePHP

Setelah *script* dimasukkan pada *form search*, aplikasi CakePHP kemudian melakukan proses pencarian. Hasil dari proses pencarian dapat dilihat pada gambar 14 dibawah ini:



Gambar 14 Hasil Pengujian *Search* Pada CakePHP

Dua langkah pengujian *Cross-Site Scripting* dengan cara menginputkan *javascript* pada *form comment* dan *search* sudah dilakukan, maka dapat diambil kesimpulan bahwa kedua framework Ruby on Rails dan CakePHP *vulnerable* terhadap serangan *Cross-Site Scripting*. Teknik *Cross site scripting* sudah lama digunakan untuk melakukan eksploitasi dynamic website. Jika sebuah website dieksploitasi menggunakan metode *cross site scripting* pada halaman tertentu, maka sebenarnya bukan pelaku eksploitasinya yang hebat, tetapi lebih kepada *developer website* yang kurang memperhatikan *security hole* yang ada. Hingga sampai saat ini masih banyak website yang memiliki kelemahan mendasar tersebut, untuk mengatasinya web developer dituntut memiliki ketelitian dan mampu melihat celah keamanan yang bisa muncul dan dapat dieksploitasi oleh pengguna. *Framework* Ruby on Rails dan CakePHP dapat me-handle serangan tersebut, sehingga dapat mengurangi dan memudahkan web developer dalam memperhatikan *security hole* yang ada. Dalam mengantisipasi serangan *cross site scripting* masing-masing *framework* Ruby on Rails dan CakePHP mempunya teknik tersendiri.

Tabel 2 Tabel hasil analisis perbandingan XSS berdasarkan studi literatur

Framework	
Ruby on Rails	CakePHP
Untuk penanggulangan serangan *cross site scripting*, Ruby on Rails menyediakan beberapa alternatif (Heiko Webers,2008) : 1. Menggunakan *Rails sanitize() helper* 2. Gunakan *escapeHTML()* 3. Gunakan *plugin SafeErb*	CakePHP mengatasi serangan *cross site scripting* dengan menggunakan *Sanitize library* (Anonim, 2007) : *1. clean($data, $connection ='default')* *2. escape($data, $connection ='default')* *3. formatColumns(&$model)* *4. html($string, $remove = false)* *5. paranoid($string, $allowed = array())* *6. The strip methods* *a. stripImages($string)* *b. stripScripts($sting)* *c. stripWhitespace($string)* *d. stripTags($string)* e. *stripAl ($string)*

Cross Site Request Forgery

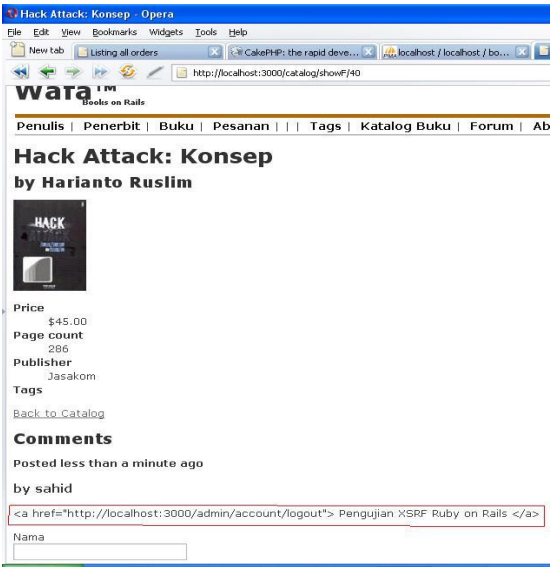
Cross-site request forgery, dikenal juga dengan *one click attack* atau *session riding* disingkat dengan CSRF atau XSRF, merupakan bentuk eksploitasi website. Cross-site request forgery menipu *Website* melalui *request* dari user yang dipercaya. Serangan bekerja melalui *link* atau *script* pada halaman site yang diakses user. *Link* tersebut dapat berupa gambar yang terhubung ke *website* tertentu. Jika *website* menyimpan informasi otentikasi dalam sebuah *cookie* yang belum *expire*, maka dengan melakukan klik ke *link* tersebut akan

menyebabkan website diakses menggunakan *cookie user* yang melakukan klik. Dengan kata lain, penyerang menipu browser user untuk mengirimkan *HTTP requests* ke *website* target (Berlianti, 2006).

Pengujian pertama dilakukan pada *framework* Ruby on Rails dengan menginputkan *link* pada *form comments*. Contoh serangan *Cross-Site Request Forgery* :

```
<a
href="http://localhost:3000/admin/account/logou
t"> Pengujian XSRF Ruby on Rails </a>
```

Ini adalah link yang digunakan untuk pengujian Cross-Site Request Forgery yang diinputkan kedalam *form comments*. Hasil dari penginputan link dapat dilihat pada gambar 15.

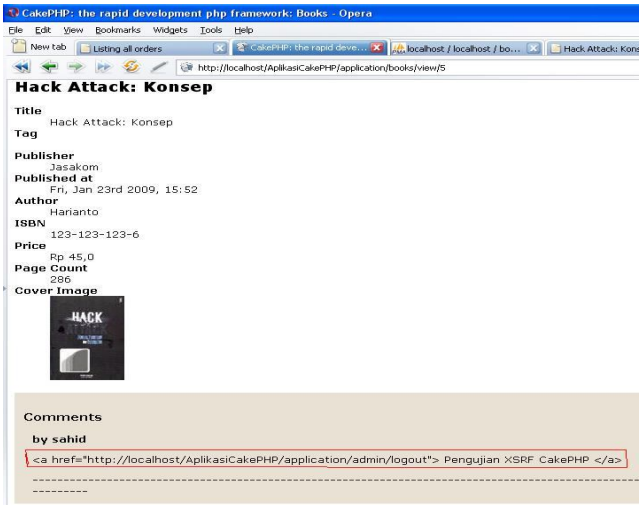


Gambar 15 Pengujian CSRF pada RoR

Pengujian kedua dilakukan pada *framework* CakePHP dengan mengginputkan *link* pada *form comments.* Contoh serangan *Cross-Site Request Forgery* :

```
<a
href="http://localhost/AplikasiCakePHP/applicat
ion/admin/logout"> Pengujian XSRF CakePHP </a>
```

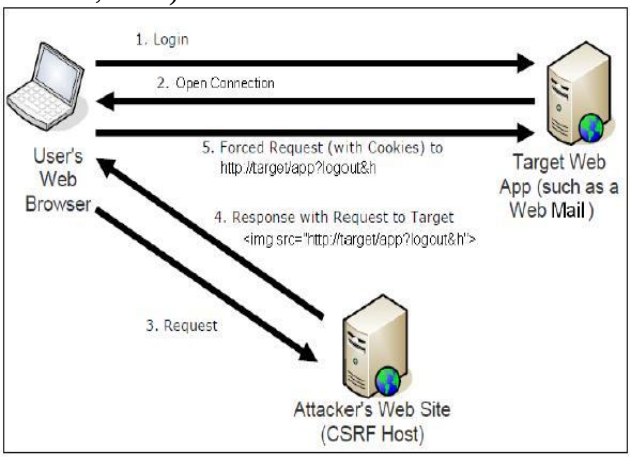
Ini adalah link yang digunakan untuk pengujian Cross-Site Request Forgery pada *framework* CakePHP yang diinputkan kedalam *form comments*. Hasil dari penginputan link dapat dilihat pada gambar 16.



Gambar 16 Pengujian CSRF Pada CakePHP

Pengujian *cross-site request forgery* pada kedua framework Ruby on Rails dan CakePHP sudah dilakukan, maka dapat dikatakan bahwa kedua framework tidak *vulnerable* dengan serangan *cross-site request forgery*. Serangan XSRF dikategorikan sebagai serangan *injection code*, dimana serangan berupa kode yang diinputkan pada halaman web di browser client lalu dieksekusi oleh aplikasi web.

Fokus dari serangan XSRF adalah bagaimana kode dapat dinputkan pada komputer korban oleh penyerang yang berada pada komputer yang berbeda. Caranya adalah menyamarkan atau menyembunyikan kode tersebut dibalik suatu objek, seperti gambar atau sekumpulan teks. Kemudian objek tersebut diberikan kepada korban. Selanjutnya penyerang meminta korban untuk melakukan suatu intruksi terhadap objek yang telah diberikan, biasanya menggunakan metode *Social Engineering*. Apabila instruksi dilaksanakan maka kode ikut tereksekusi tanpa diketahui oleh korban. Ilustrasinya adalah sebagai berikut (Anonim, 2009) :



Gambar 17 Ilustrasi Serangan CSRF

Keterangan pada gambar:

1. User login pada web server, misalnya pada webmail.
2. Server membuka sesi komunikasi dengan user atau client.
3. Penyerang telah menyediakan *link* agar dapat diakses oleh user. Misalnya dikirim ke email user atau diposting di suatu website. Penyerang meminta user untuk melakukan suatu intruksi dan user melakukannya.
4. Pada instruksi yang disediakan, penyerang menyimpan perintah yang tidak diketahui oleh user. Misalnya perintah berupa kode

https://mail.google.com/mail?logout&h yang disembunyikan pada link berupa image atau objek gambar.

5. Ketika user menjalankan instruksi maka kode tersebut ikut juga tereksekusi. Kode akan diterjemahkan oleh web server sebagai permintaan dari client. Misalnya kode *https://mail.google.com/mail?logout&h* akan menyebabkan user *sign out* atau keluar dari aplikasi webmail google.

Berdasarkan hasil penelitian, *framework* Ruby on Rails dan CakePHP memprotek serangan *cross site request forgery* dengan beberapa metode diantaranya :

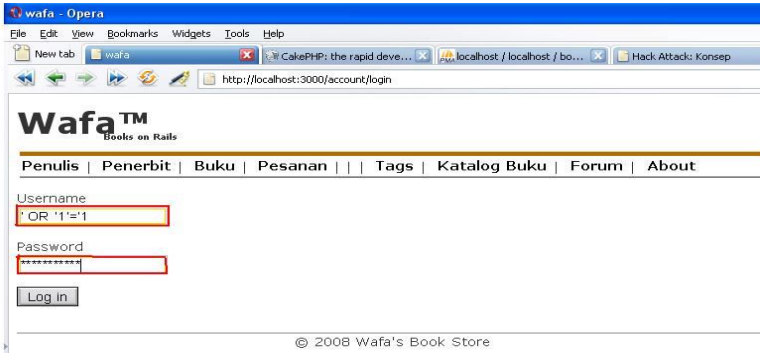
Tabel 3 Tabel hasil analisis perbandingan CSRF berdasarkan studi literature

Framework	
Ruby on Rails	CakePHP
Dari hasil analisa yang dilakukan untuk mengatasi masalah kelemahan CSRF pada website yang dibangun dengan RoR dapat diatasi dengan (Heiko Webers, 2008) : 1. Seperti yang diwajibkan oleh W3C, gunakan GET & POST 2. Permintaan *security token non-GET* pada sisi server (`protect_from_forgery :secret => "123456789012345678901234567890..."`).	Untuk memprotek serangan *cross site request forgery*, cakePHP menyediakan *SecureAction component* yang dapat di taruh dalam controller(s) : `var $components = array('SecureAction');` (Julien Perez, 2008). Bisa juga dengan menambahkan *security component* ke array *$components array* dalam controller(s): *public $components = array('Security');* (Dhofstet, 2009**).**

Authentication

Guna mencegah pihak yang tidak berhak memakai account user yang sah untuk akses kehalaman website, maka dalam website perlu untuk menerapkan sistem otentikasi yang tepat. Dalam menerapkan prinsip otentikasi pada web, sistem yang biasanya digunakan adalah *login system*. Otentikasi merupakan suatu bentuk komunikasi antara Web browser dan Web server. Web Browser berada pada sisi client (user) dan Web server terdapat pada sisi website. Kebanyakan web-site menyediakan account untuk usernya agar dapat melakukan akses, kontrol atau perubahan pada service yang disediakan web-site. Untuk itu user perlu login sebelum memasuki halaman tertentu. Permintaan login biasanya terdiri dari username dan password. User yang valid akan diizinkan untuk akses ke suatu halaman atau fungsi tertentu yang dimintanya (Berlianti, 2006).

Pengujian outentikasi pada penelitian ini akan menggunakan teknik SQL injection. Pengujian pertama dilakukan pada *framework* Ruby on Rails dengan menginputkan karakter illegal. Contoh pengujian otentikasi pada *form login* :



Gambar 18: Pengujian Otentikasi pada RoR

Pengujian otentikasi dilakukan dengan menginputkan `' OR '1'='1` pada username dan untuk password diisikan `' OR '2'>'1` maka SQL query akan membacanya sebagai:

```
SELECT * FROM users WHERE username = '' OR '1'='1'
AND password = '' OR '2'&gt;'1' LIMIT 1
```

artinya SQL query akan menemukan *record* pertama didalam database dan memberikan akses kepada pengguna. Pengujian kedua dilakukan pada *framework* CakePHP dengan menginputkan karakter illegal seperti diatas, yaitu menginputkan `' OR '1'='1` pada username dan untuk password diisikan `' OR '2'>'1`

Gambar 19 Pengujian Otentikasi pada CakePHP

Setelah karakter illegal dijalankan pada kedua *framework* Ruby on Rails dan CakePHP maka didapat hasil yang sama, dapat dilihat pada gambar 20.

Gambar 20 Hasil Pengujian Otentikasi Pada RoR dan CakePHP

Hasil pengujian otentikasi pada kedua *framework* dikembalikan pada form login kembali. Dari hasil pengujian otentikasi pada kedua *framework* Ruby on Rails dan CakePHP maka dapat disimpulkan bahwa *framework* tersebut aman. Otentikasi yang tepat diperlukan pada website untuk mengamankan data, mencegah pencurian identitas dan sebagainya. Berdasarkan hasil penelitian, *framework* Ruby on Rails dan CakePHP menyediakan system otentikasi untuk melindungi data pada aplikasi website.

Tabel 4: Tabel hasil analisis perbandingan otentikasi berdasarkan studi literatur

Framework	
Ruby on Rails	CakePHP
Ada beberapa plug-in otorisasi dan otentikasi yang tersedia pada RoR. Namun plug-in yang baik adalah plug-in yang menyimpan sandi terenkripsi seperti plug-in yang paling populer yaitu restful_authentication (Heiko Webers,2008): Ada sembilan ribu cara untuk melakukan authentication, namun Acts_as_authentication adalah pendekatan yang paling masuk akal (Hendrayana, 2007).	Sistem User Authentication adalah bagian dari *common* pada banyak aplikasi web. Dalam CakePHP ada beberapa system untuk otentikasi user, masing-masing memberikan pilihan yang berbeda. Pada intinya komponen otentikasi akan mengecek seorang user yang memiliki account pada website, kemudian user akan diberikan hak untuk mengakses website tersebut. CakePHP AuthComponent dapat digunakan untuk menciptakan system ini dengan mudah dan cepat (Anonim, diakses pada tanggal 28 Januari 2010).

Setelah dilakukan penelitian terhadap kedua *framework*, maka dapat disimpulkan bahwa masing-masing framework menyediakan modul-modul yang siap digunakan untuk mengatasi berbagai ancaman keamanan yang telah disebutkan diatas.

Penutup

Setelah dilakukan penelitian dapat diambil kesimpulan sebagai berikut :

1. Dilakukan dua tahapan analisa untuk membandingkan *framework* RoR dan CakePHP pada penelitian ini, yaitu analisa berdasarkan studi literatur dan analisa berdasarkan aplikasi yang dibuat. Tahapan analisis kedua dilakukan untuk memperkuat hasil dari analisa tahap pertama.
2. Kelebihan dan kekurang dari masing-masing *framework* dapat dilihat dari hasil analisa tahapan pertama yang disesuaikan dengan analisa tahapan kedua, hasil dari tahapan kedua memperkuat hasil dari hasil analisa tahapan pertama. Parameter yang digunakan berdasarkan fitur-fitur umum keamanan yang harus ada pada sebuah *framework*

Tabel 5: Tabel hasil pengujian

Fitur Umum Keamanan *Framework*	Hasil
Validasi data	Dukungan validasi data pada kedua *framework* sangat bagus, karena validasi data dipasang secara built-in
Enkripsi data	RoR dan CakePHP sama-sama didukung component enkripsi data, namun RoR lebih unggul karena dukungan plugin sehingga penggunaannya lebih mudah.
Dukungan faktor internal	CakePHP unggul dalam hal dukungan internal, karena dukungan library yang lebih lengkap.
Dukungan faktor eksternal	RoR lebih unggul dalam hal dukungan eksternal, karena dukungan plugin yang lebih lengkap.

3. Ancaman yang dapat menyerang sistem keamanan pada kedua *framework* adalah serangan *SQL injection, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF)* dan *authentication.*

4. Solusi untuk menghadapi ancaman pada sistem keamanan *framework* RoR dan CakePHP dapat dilihat dilihat dari hasil analisa tahapan pertama.
5. Tidak bisa ditarik kesimpulan secara garis besar alternatif *framework* terbaik diantara kedua *framework* ini. Hal ini dikarenakan kedua *framework* memiliki banyak solusi untuk mengatasi ancaman keamanan. Hasil ini pun dipengaruhi oleh *developer* dalam merancang aplikasinya serta versi dari masing-masing *framework* yang digunakan. Sehingga, hasil dari analisis ini bukanlah sesuatu yang mutlak dan tidak dapat dirubah.

Daftar Pustaka

Hendrayana. 2007. Study Keamanan Ruby on Rails (RoR). Sekolah Teknik Elektro Dan Informatika Institut Teknologi Bandung.

O'Brien, D., 2009, Cook up Web Sites Fast with CakePHP, Part 3: Use Sanitize for your Protection, IBM Corporation. Diakses pada tanggal 1 Desember 2009 (23:27).

Muchtar, S., W., 2009, Perbandingan Metodelogi Pengembangan Aplikasi Website Web 2.0 Dengan Menggunakan Framework Ruby on Rails Dan CakePHP. Yogyakarta : Jurusan Teknik Informatika STMIK AMIKOM Yogyakarta.

Aditya, Y.,2009, Open Source Framework, Bisnis Serius di Ranah Web 2.0 Dalam Pandangan Teknis Dan Bisnis, http://yodi.web.id/content/open_source_framework_bisnis_serius_di_ranah_web_20_dalam_pandangan_teknis_dan_bisnis , diakses pada tanggal 2 September 2009 (22:15).

Muhardin, E., 2006, Intro Framework, http://endy.artivisi.com/blog/java/intro-framework/, diakses pada tanggal 27 Januari 2010 (05.36).

Figiel, A., 2008, Introduction to Ruby on Rails, Blog.agnessa.eu, diakses pada tanggal 18 Juli 2008 (14:21).

Rajshekhar, A.P., 2008, Building Dynamic Web 2.0 Websites with Ruby on Rails. Packt publishing, Brimingham-Mumbai.

Yodi, 2008, Framework CakePHP dan peluang bisnis yang mengikutinya (updated!), http://yodi.web.id/content/framework_cakephp_dan_peluang_

bisnis_yang_mengikutinya_updated, diakses pada tanggal 27 Januari 2009 (18:02).

Galih, 2008, Celah keamanan web pemerintah klaten, http://dreamwork.web.id/ detail/celah-keamanan-web-pemerintah-klaten, diakses pada tanggal 25 Oktober 2009 (21:07).

Anonim, 2008, Celah Yang Ter-acuhkan, http://www.noee.co.cc/2008/07/celah-yang-ter-acuhkan.html, diakses pada tanggal 25 Oktober 2009 (21:38).

Webers, H., 2008, Ruby on Rails Security Guides, http://guides.rubyonrails.org/ security.html, diakses pada tanggal 14 Desember 2009 (23:24).

Tambun, R., U., 2004, Cross Site Scripting, Departemen Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Bandung.

Anonim, 2007, XSS prevention and general sanitization, http://myeasyscripts.com /loudbaking/xss-prevention-and-general-sanitization/, diakses pada tanggal 1 Januari 2010 (05:23).

Berlianti, 2006, Pengaturan Otentikasi dan Session Pada Web, Sekolah Teknik Elektro Dan Informatika Institut Teknologi Bandung

Anonim, 2009, Cross Site Request Forgery (XSRF), http://www.ittelkom.ac.id/ library/index.php?view=article&catid=6%3Ainternet&id=593%3Across-site-request-forgery--xsrf-&option=com_content&Itemid=15, diak- ses pada tanggal 3 September 2009 (01:52).

Perez, J., 2008, Protect your website against CSRF attacks, http://bakery.cakephp.org/articles/view/protect-your-website-against-csrf-attacks, diakses pada tanggal 28 Januari 2010 (09:27).

Dhofstet, 2009, CakePHP and CSRF, http://stackoverflow.com/ questions/1584420/cakephp-and-csrf/1584464#1584464, diakses pada tanggal 7 Januari 2010 (21:09).

Anonim, 5.2 Authentication, http://book.cakephp.org/view/172/Authentication, diakses pada tanggal 28 Januari 2010 (09:58).