Peningkatan Keamanan Web Terhadap Serangan *Cross Site Scripting* (XSS).

Herman Tolle, ST. MT. , Tri Astoto Kurniawan, ST. MT, Andi Zakaria, ST.

ABSTRAK: Teknologi *dynamic web page* kini telah menjadi bagian yang tidak terpisahkan dari kehidupan dunia maya. Teknologi ini membawa perubahan yang signifikan dalam proses pembangunan sistem penyedia layanan dalam jaringan internet. Teknologi ini memampukan penyedia layana untuk memberikan layanan yang lebih inovatif. Efek yang diharapkan tentu saja peningkatan dari segi ekonomi. Namun dibalik keuntungan-keuntungan tersebut, teknologi ini memiliki permasalahan dari segi keamanan. Permasalahan keamanan tersebut dinamakan *Cross Site Scripting*, atau juga dikenal sebagai XSS. Kebocoran informasi penting merupakan hal yang dapat terjadi jika kelemahan ini tidak ditangani dengan baik.Penelitian ini bertujuan mempelajari bagaimana cara kerja *cross site scripting (XSS)* dan kemudian mengembangkan suatu metode pencegahan dan pengamanan website terhadap serangan XSS. Sebagai bahan studi kasus dipilih sebuah *vulnerable* CMS (*Content Management System*) yaitu AuraCMS yang banyak digunakan sebagai aplikasi portal tetapi masih memiliki kelemahan. Beberapa aplikasi pada AuraCMS memiliki *vulnerable* terhadap serangan *Cross Site Scripting,* yaitu aplikasi kirim artikel dan aplikasi buku tamu. Melalui pengkajian penelitian, maka dikembangkan suatu modul yang berfungsi sebagai filter untuk mencegah serangan XSS tersebut. Modul tersebut berisi beberapa fungsi yang berguna untuk menyaring masukan berupa kode atau injeksi kode. Dengan adanya modul *filter* tersebut, masukan berupa kode pada kedua aplikasi tersebut dapat dihindari sehingga aplikasi yang dibuat dapat berjalan sesuai dengan yang diharapkan dan terhindar dari serangan XSS yang merusak.

Kata kunci : *Cross Site Scripting*, *Web*, CMS (*Content Management System*), dan keamanan aplikasi web..

Halaman Web yang dinamis meru-pakan teknologi yang memberi per ubahan penyediaan informasi, laya-nan, dan tampilan secara signifikan. Halaman Web yang dinamis me-mungkinkan interaksi yang lebih baik antara penyedia layanan dengan penggunanya. Dengan mengguna-kan teknologi ini, halaman Web akan terlihat lebih manusiawi. Penyedia layanan dapat menambahkan *content-content* yang sebelumnya masih merupakan impian belaka.

Saat mengakses sebuah halaman web, maka akan berhubungan de-ngan *cookies*. *Cookies* adalah *data file* yang ditulis ke dalan *harddisk* oleh *Web Server* untuk mengidentifi-kasikan *user* pada *site* tersebut sehingga sewaktu *user* tersebut kembali mengunjugi *site* tersebut, *site* itu sudah akan mengenalinya.

Jadi bisa dikatakan bahwa *cookies* itu semacam *ID card* untuk di *site* tersebut yang telah dikunjungi. Dan tiap-tiap *Web site* pada umum-nya mengeluarkan/membuat *cookies* itu masing-masing. Ada Web yang menyapa tiap kali *user* mengunjungi *site* tersebut selayaknya teman lama, itu berkat *cookies*.

Cookies membantu *Web site* untuk "mengingat" siapa *user* dan meng-*set preferences* sesuai untuk *user* sehingga apabila *user* kembali mengunjungi *Web site* tersebut akan langsung dikenali. Menghilangkan kebutuhan untuk me-*register* ulang di *Web site* tersebut yang memerlukan *user* untuk me-*register* supaya bisa mengakses *Web site* tersebut (*site* tertentu saja), *cookies* membantu meng-*log-in user* ke dalam *Web server* tersebut. Memungkinkan *Web site* untuk meng-*track* pola *Web surfing user* dan *site-site* favorit yang sering dikunjungi. Meskipun sekilas bahwa *cookies* itu seakan banyak

Herman Tolle, Tri Astoto Kurniawan, Andi Zakaria, adalah Dosen Teknik Elektro Universitas Brawijaya Malang

gunanya, akan tetapi sampai seka-rang masih menjadi bahan perdebat an mengenai keberadaaan *cookies* ini, karena selain membuat sebuah *Web site* terlihat *user friendly*, tetapi juga menghadirkan isu melanggar privasi pengakses Web.

RUMUSAN MASALAH

Penelitian ini akan mengkaji implementasi Web dinamis terhadap serangan *Cross Site Scripting* serta bagaimana mencegah dan meng-amankannya dari serangan tersebut, yaitu pengkajian terhadap :

1. Cara kerja serangan *Cross Site Scripting* terhadap aplikasi web.
2. Eksploitasi yang dilakukan akibat *Cross Site Scripting.*
3. Pencegahan terhadap serang-an *Cross Site Scripting.*

BATASAN MASALAH

Untuk memberikan penekanan khusus sesuai dengan judul peneliti-an ini maka dilakukan pembatasan pada penelitian ini :

1. Sistem operasi yang digunakan adalah Microsoft Windows XP Professional, Version 2002, Service Pack 2.
2. Bahasa pemrograman yang dipakai adalah PHP, MySQL, dan JavaScript.
3. *Web Server* yang digunakan adalah xampp-win32-1.4.14.
4. Komputer yang digunakan ada-lah komputer yang *stand alone*.
5. Studi kasus pada sebuah *Content Management System* (CMS) yaitu AuraCMS.
6. *Browser* Mozilla Firefox 4.42.

TUJUAN

Tujuan penelitian ini adalah mempelajari metode serangan *Cross Site Scripting* (XSS) terhadap suatu aplikasi Web dan menemukan suatu teknik pencegahan dan pengamanan terhadap serangan XSS.

LANDASAN TEORI

Cross Site Scripting (XSS)

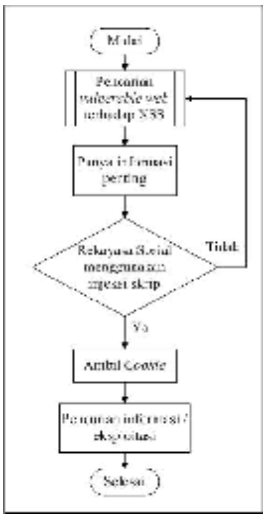
Cross Site Scripting dapat diarti-kan sebagai kelemahan yang terjadi akibat ketidakmampuan *server* dalam memvalidasi *input* yang diberi-kan oleh *user*. Algoritma, yang digu nakan untuk pembuatan halaman yang diinginkan, tidak mampu mela-kukan penyaringan terhadap *input* tersebut. Hal ini memungkinkan halaman yang dihasilkan menyerta-kan perintah yang sebenarnya tidak diperbolehkan. *Cross Site Scripting* merupakan kelemahan yang populer untuk dieksploitasi. Namun sayang-nya, banyak penyedia layanan yang tidak mengakui kelemahan tersebut dan melakukan perubahan pada sistem yang digunakan.

Cara Kerja *Cross Site Scripting* (XSS)

Cross Site Scripting bekerja bak penipu dengan kedok yang mampu mengelabui orang yang tidak was-pada. Elemen penting dari keberha silan *Cross Site Scripting* adalah *social engineering* (rekayasa sosial) yang baik dari si penipu. *Social engineering* merupakan elemen ter-penting yang menentukan keberhasil an penipuan yang akan dilakukan. *Cross Site Scripting* memampukan seseorang yang tidak bertanggung-jawab melakukan penyalahgunaan informasi penting.

Sebelum sampai pada proses penyalahgunaan tersebut, *attacker* mengambil langkah-langkah dengan mengikuti pola tertentu. Langkah pertama, *attacker* melakukan penga-matan untuk mencari web-web yang memiliki kelemahan *Cross Site Scripting*. Langkah kedua, *attacker* mencari tahu apakah web tersebut menerbitkan informasi yang dapat digunakan untuk

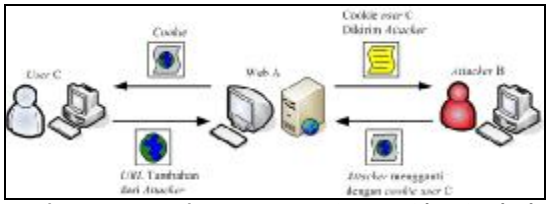
melakukan pencuri-an infomasi lebih lanjut. Informasi tersebut biasanya berupa *cookie*. Langkah kedua ini tidak selalu dijalankan.



Gambar 1 Diagram alir serangan *Cross Site Scripting*

Langkah ketiga, *attacker* membu-juk korban untuk mengikuti sebuah *link* yang mengandung kode, dituju-kan untuk mendapatkan informasi yang telah disebutkan sebelumnya. Kemampuan *social engineering* dari *attacker* diuji disini. Setelah menda-patkan informasi tersebut, *attacker* melakukan langkah terakhir, pencuri an maupun pengubahan informasi vital.

Berikut merupakan contoh kasus dari *Cross Site Scripting*. Anggap sebuah penyedia layanan *message board*, Web A, memiliki kelemahan *Cross Site Scripting*. Web tersebut juga menghasilkan *cookie*. *Cookie* tersebut bertujuan agar *user* dapat membuka jendela *browser* baru tanpa memasukkan *username* dan *password* lagi. B, mencoba untuk mengeksploitasi kelemahan ini, meletakkan sebuah *link* yang me-ngandung kode yang “jahat” pada *message board* tersebut. C, seorang *user*, tertarik pada *link* tersebut menekan *link* tersebut. Tanpa disa-dari C, *cookie* yang terdapat pada komputernya telah dikirimkan ke komputer B. Kini B dapat mengakses *message board* sebagai C hanya dengan menggantikan *cookie*-nya dengan *cookie* yang ia dapatkan dari C. Gambar 1 menunjukkan alur dari kejadian tersebut.



Gambar 2 Contoh serangan *Cross Site Scripting* (XSS)
Sumber : [OLL-03:2]

Pengamanan terhadap serangan *Cross Site Scripting* (XSS)

Pencegahan jauh lebih baik daripada pengobatan. Pencegahan *Cross Site Scripting* sebenarnya merupakan bagian dari proses perancangan sistem yang dikem-bangkan. Jika sistem tersebut meng-gunakan teknologi *dynamic web page*, berbagai pertimbangan perlu dilakukan. Pengamanan disini dilaku-kan dari sisi *developer, user*, dan *browser*.

Developer

Para *developer* (pengembang) aplikasi web dan para *vendor* (pemi-lik) web perlu memastikan bahwa semua masukan dari *user* perlu diu-raikan dan disaring lagi dengan baik. Masukan dari *user* meliputi berbagai hal yang disimpan pada metode *GET*, *POST*, *cookies*, *URLs*, dan data yang seharusnya ditampilkan antara *browser* dan *server*. Bebera-pa petunjuk yang baik dalam melaku kan penyaringan terhadap beberapa karakter yang dimasukkan oleh *user* dapat ditemukan pada dokumen per-syaratan OWASP ”OWASP Panduan untuk Membangun aplikasi web dan pelayanan web yang aman”. Pandu-an dapat dikunjungi pada alamat situs owasp yaitu (http://www.owasp. org/requirements).

Teknologi *Static Web Page*

Cara terbaik dan efektif untuk menghindari terjadinya *cross site scripting* adalah menghindari peng-gunaan teknologi *dynamic web page*. Halaman yang statis tentu saja memberikan kontrol yang lebih di sisi *server* dibandingkan dengan hala-man web yang dinamis. Halaman web yang dihasilkan secara statis akan memberikan kelakukan yang lebih pasti dibandingkan halaman web yang dihasilkan secara dinamis. Konsekuensi yang ditanggung ada-lah penyedia layanan harus merela kan sifat interaktif yang mungkin diinginkan.

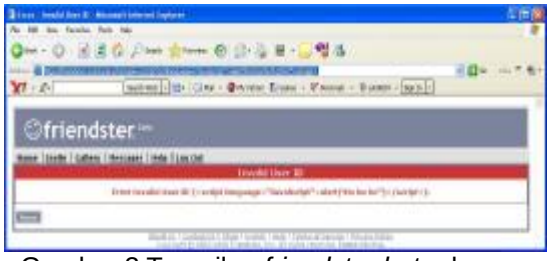
Metoda *POST*

Metoda *POST* adalah metoda pengirimana data dimana variabel yang dikirimkan tidak disertakan pada link yang digunakan. Metoda *POST* menyembunyikan variabel yang dikirimkan dari *user*. Metoda ini menjamin kode tidak dapat diinjeksi-kan melalui *link* yang telah didesain oleh *attacker*. *Link* merupakan satu satunya cara yang dapat digunakan oleh *attacker* untuk mengeksploitasi *Cross Site Scripting*. Oleh karena itu, metoda ini ampuh untuk mengatasi *Cross Site Scripting*. Kekurangan metoda ini, *user* tidak dapat menyim-pan *link* favorit untuk mempermudah navigasi.

Pengkodean Karakter *Special* pada *Link*

Untuk me-nonaktifkan kode *script* yang diinjeksikan, kita perlu mem-buat aplikasi yang mampu mengko dekan karakter tersebut, sehingga karakter tersebut tidak dapat dime-ngerti oleh *browser* yang digunakan. Proses pengkodean juga harus men-cakup *HTML escape code (%hexnumber)*. Gambar 2 menun-jukkan menunjukkan web dengan proses pengkodean yang baik. *Link* yang dimasukkan pada *field address* dari *browser* adalah:

http://friendster.com/user.php?uid=<script%20language="JavaScript">alert('Ho%20h o%20ho")</script>.



Gambar 3 Tampilan *friendster beta* dengan menggunakan *HTML escape code*
Sumber : *http://friendster.com*

Penggalan *soure code* dari halaman yang dihasilkan oleh *server* berikut, menunjukkan pengkodean yang dilakukan oleh penyedia layanan *friendster beta*.

```
<div class="error">
<h1> Invalid User ID </h1>
Error Invalid User ID
(&lt;script language="JavaScript"&gt;alert('Ho
ho ho")&lt;/script&gt;).
</div>
<div>
<p class="buttonbox">
<a
class="submitbutton"href="/home.php">Home</
a>
</p>
</div>
```

Proses pengkodean diatas mengunakan format UTF-8 untuk mengkodekan karakter spesial. Pada contoh penggalan kode diatas ‘<’ dikodekan menjadi &ltd dan ‘>’ dikodekan menjadi &gt. Dengan pengkoden tersebut *script* yang ada tidak akan dijalankan. Rutin sederhana yang dapat melakukan proses pengkodean diatas adalah *Server.HTMLEncode(string)*.

Injeksi sering kali tidak berhasil dilakukan. Ini dapat terjadi akibat proses pem-*filter*-an di sisi *server*. Pem-*filter*-an ini biasanya dilakukan dengan menghilangkan karakter-karakter spesial yang penting dalam pengkodean dan juga karakter-karakter ekuivalen yang

dikodekan. Pengkodean yang biasa digunakan adalah *HTML escape encoding*.

METODE PENELITIAN

Pada tahap ini dijelaskan mengenai langkah-langkah yang akan dilakukan untuk menganalisis aplikasi web yang *vulnerable* ter-hadap serangan *Cross Site Scripting* (XSS) serta pencegahannya. Ada-pun langkah–langkah yang dilakukan yaitu :

Studi Literatur

Studi literatur yang dilakukan meliputi:

a. Konsep *Cross Site Scripting*.
b. Teori dasar web meliputi :
    1. Struktur dan bahasa pemro-graman HTML (*Hyper Text MarkUp Language*).
    2. Struktur dan bahasa pemro-graman JavaScript.
    3. Stuktur dan bahasa pemro-graman PHP (*PHP Hypertext Preprocessor*).
c. Basis data meliputi :
    1. Bahasa basis data SQL (*Structure Query Language*).
    2. Koneksi basis data MySQL.
    3. SQL (*Structure Query Language*) dalam MySQL.

Studi Kasus terhadap *Vulnerable* CMS (AuraCMS)

AuraCMS adalah hasil karya anak bangsa yang merupakan *software* CMS (*Content Management System*) untuk *website* yang berbasis PHP & MySQL berlisensi GPL (*General Public License*). Dengan bentuk yang sederhana dan mudah ini diha-rapkan dapat digunakan oleh pema kai yang masih pemula sekalipun. *Software* ini digunakan untuk mem-bangun *website* dengan *content* dinamis yang berlisensi GNU/GPL, *free-simple-easy to use*.

Perancangan dan Implementasi

Pada tahap perancangan ini yang dilakukan adalah melakukan analisis kebutuhan terhadap serangan *Cross Site Scripting* dan perancangan sis-tem keamanan terhadap serangan *Cross Site Scripting*.

Kemudian untuk implementasi yang dilakukan yaitu mengenai ling-kungan implementasi, algoritma sis-tem, injeksi dengan *Cross Site Scripting* terhadap beberapa aplikasi yang *vulnerable* terhadap serangan *Cross Site Scripting* pada AuraCMS, eksploitasi, pencegahan, serta per-baikan kode program penyusun aplikasi.

Pengujian dana Analisis

Pengujian dan analisis merupa-kan tahap untuk menguji tiap blok sistem yang dibuat dan dibandingkan dengan teori dan perancangan. Pe-ngujian dan analisis ini dilakukan untuk mendapatkan kesimpulan me-ngenai bagaimana serangan *Cross Site Scripting* itu bisa terjadi serta bagaimana cara mencegah serang-an tersebut.

Pengambilan Kesimpulan dan Saran

Tahap berikutnya dari penulisan penelitian ini adalah pengambilan kesimpulan dari analisis *vulnerable web* terhadap serangan C*ross Site Scripting*. Pengambilan kesimpulan dilakukan setelah semua tahapan analisis dan pengujian telah selesai dilakukan dan didasarkan pada kesesuian antara teori dan praktek. Kesimpulan ini merupakan informasi akhir dari analisis *vulnerable web* yang berisi mengenai cara mence-gah dan mengamankan suatu web dinamis terhadap serangan *Coss Site Scripting*.

PERANCANGAN SISTEM

Analisis kebutuhan sistem

Analisis kebutuhan sistem dibutuh-kan untuk menggambarkan kebutuh an yang dibutuhkan oleh pengguna, untuk mengetahui aplikasi web apa saja yang *vulnerable* dan bagaimana serangan *Cross Site Scripting* terjadi serta

pencegahan yang perlu dilaku-kan terhadap serangan *Cross Site Scripting* ini.

Analisis kelemahan pada AuraCMS

Sebuah serangan *cross site scripting* dilakukan dengan menye-diakan alamat khusus yang dikemas oleh *attacker* untuk calon korbannya. Dalam konteks XSS, penyerang mengundang korbannya untuk me-ngeksekusi alamat URL yang diberi kan dan membiarkan korban mengi kuti *link* tersebut dengan menjalan-kan *script* yang sebelumnya beraksi di komputer klien untuk mendapat-kan informasi yang diinginkannya.

Penemuan titik rawan aplikasi

Beberapa titik dimana biasanya XSS terjadi adalah pada halaman konfirmasi (seperti mesin pencari dimana memberikan keluaran dari masukan pengguna dalam aktvi-tas pencarian) dan halaman kesa lahan (*error page*) yang memban-tu pengguna dengan mengisi ba gian *form* untuk memperbaiki kesalahan. Pada AuraCMS ini beberapa aplikasi yang *vulnerable* adalah pada aplikasi bukutamu dan aplikasi kirim artikel.

Perancangan sistem

Perancangan ini dilakukan untuk mengetahui aplikasi sistem yang akan dibuat secara umum. Peran-cangan sistem meliputi perancangan pencegahan serangan *Cross Site Scripting* (XSS), diagram blok sistem, dan pengamanan terhadap aplikasi web.

Perancangan pencegahan serangan *Cross Site Scripting* (XSS)

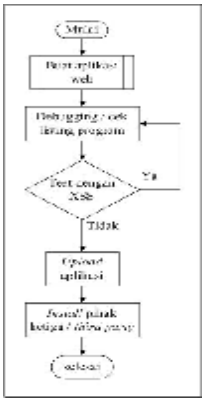
Perancangan pencegahan serang an *Cross Site Scripting* (XSS), tahap ini dibahas mengenai bagaimana mencegah serangan *Cross Site Scripting* dan titik-titik apa saja yang menyebabkan aplikasi web itu *vulnerable* terhadap serangan *Cross Site Scripting ini*. Membuat sebuah *web site* yang tidak *vulnerable* terha-dap serangan *Cross Site Scripting* melibatkan usaha dari pengembang (*developer*), pengguna (*user*), dan *browser*.

Diagram alir pencegahan serangan *Cross Site Scripting*

Salah satu aspek yang sangat penting dalam menemukan *bug* dari *Cross Site Scripting* adalah pada *web site* yang bersifat dinamis, dimana *attacker* dapat melakukan serangan ini. Pengem-bang/*Developer* perlu menghilang kan lubang keamanan *Cross Site Scripting* ini selama proses pe-ngembangan web yang dibuat. Untuk memeriksa aplikasi yang dibuat apakah *vulnerable* atau tidak terhadap serangan injeksi kode ini, lakukan hal berikut ini. Karena masing-masing format yang tampak pada masukan sebuah *form* yang dilewatkan pada sebuah *URL* dapat dicoba format salah satu *script* berikut :

```
<script>alert('xss')</script>
<img csstest =
javascript:alert('xss')>&{alert('XSS')};
```

Diagram alir untuk pencegahan serangan *Cross Site Scripting* ditunjukkan dalam Gambar 4.

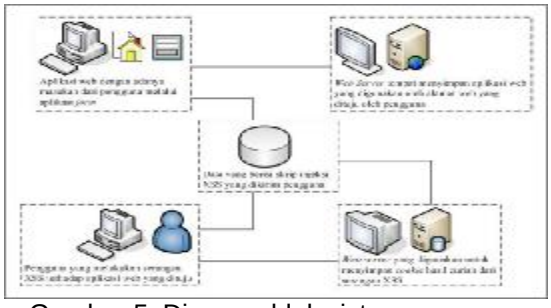


Gambar 4 Diagram alir pencegahan serangan *Cross Site Scripting*
Sumber : *Perancangan*

Diagram blok sistem

Diagram blok sistem menggam-barkan setiap blok atau bagian dari sistem aplikasi. Bagaimana serang-an

Cross Site Scripting itu terjadi. *Cross Site Scripting* itu terjadi dapat digambarkan dengan diagram blok yang ditunjukkan dalam Gambar 5.



Gambar 5. Diagram blok sistem serangan *Cross Site Scripting*
Sumber : *Perancangan*

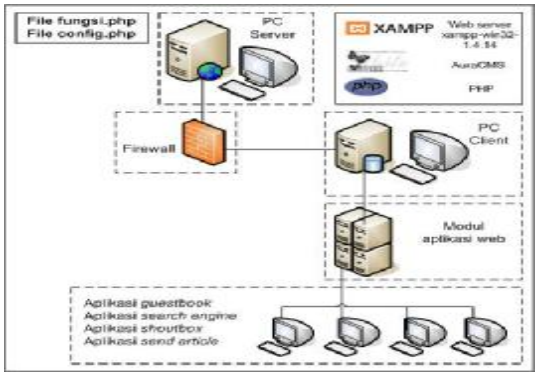
Blok diagram sistem serangan *Cross Site Scripting* meliputi :

- *Attacker* (penyerang)
- Aplikasi web
- Skrip injeksi
- *Web server* aplikasi web
- *Web server cookie*

Pengamanan terhadap aplikasi web

Perancangan sistem pengaman modul aplikasi web merupakan perancangan yang dilakukan untuk mengetahui apa saja modul pendu-kung aplikasi web serta bagaimana melakukan pengamanan terhadap aplikasi web ini.

Modul yang diperlukan sebagai solusi pencegahan terhadap serang-an *Cross Site Scripting* ini ditunjuk kan dalam sebuah diagram blok, seperti ditunjukkan dalam Gambar 6..



Gambar 6. Diagram blok *file* fungsi pencegahan *Cross Site Scripting*
Sumber : *Perancangan*

Blok diagram sistem pencegahan terhadap serangan *Cross Site Scripting* (XSS) meliputi :

§ Pada PC Server terdapat *web server* xampp-win32-1.4.42, AuraCMS, dan bahasa pemro-graman php. Untuk meghindari adanya lubang keamanan maka pada AuraCMS ini perlu dibuat sebuah modul yaitu modul *filter* yang berperan dalam melakukan *filter* terhadap masukan yang diberikan oleh *user* dan modul untuk melakukan pengaturan terhadap isi dari CMS tersebut yaitu :

Ø File filter.php

File inilah yang menentukan aman atau tidaknya sebuah aplikasi web yang dibuat ter-hadap serangan *Cross Site Scripting*, karena pada *file* ini terdapat beberapa fungsi yang berguna untuk melakukan *filter* seperti :

1. htmlspecialchars(), fungsi untuk menentukan karak-ter html tertentu saja yang diperbolehkan untuk digu-nakan pada aplikasi web.
2. stripslashes(), merupakan fungsi yang digunakan un-tuk mengabaikan *back-slash* yang ditimbulkan dari *input form*, kebalikan dari fungsi ini adalah fungsi addslashes().
3. strip_tags(), untuk men-strip atau menghilangkan tag php serta html, dan mengembalikan *string* ha-sil. Fungsi ini tepat sekali jika digunakan untuk me-ngatur tag-tag yang diper-bolehkan dan tag yang akan diabaikan.
4. htmlentities(), merupakan fungsi yang mengkonversi karakter khusus ke dalam tag html.

Ø File config.php

Pada *file* config.php ini terda-pat beberapa variabel yang digunakan untuk mengatur isi dari web tersebut. *File* ini didalamnya juga menyertakan *file* filter.php. Pengaturan yang dimaksud disini seperti :

1. Konfigurasi untuk *data-base*.
2. Konfigurasi situs dan *e-mail*.
3. Konfigurasi data situs.
4. Konfigurasi untuk *file image*.

§ *Firewall*
§ PC Client
§ Modul aplikasi web
§ Beberapa aplikasi web yang membutuhkan interaksi dengan *user* pada web ini, meliputi:
   1. *Guestbook*
   2. *Send article*

IMPLEMENTASI SISTEM

Lingkungan Implementasi

Sistem dibuat dengan menggu-nakan aplikasi pemrograman web PHP dan *database* MySQL. Sistem diimplementasikan dengan menggu-nakan spesifikasi sebagai berikut :

- Sistem operasi yang digunakan adalah Windows XP Professional, *Version* 2002, Service Pack 2.
- *Web server* yang dipakai adalah xampp-win32-1.4.14.
- *Content Management System* (CMS) AuraCMS.
- *Browser* Mozilla firefox 2.0.0.1.
- Aplikasi web pada AuraCMS.
- Bahasa pemrograman web yaitu PHP dan JavaScript.

Implementasi anatomi serangan *Cross Site Scripting* (XSS)

Implementasi *Cross Site Scripting* ini dilakukan terhadap *open source* CMS (*Content Management System*) yaitu AuraCMS, hal ini dilakukan untuk mengetahui apakah CMS tersebut telah melakukan pem*filter*an terhadap skrip-skrip injeksi serangan *Cross Site Scripting* ini. Tampilan awal dari AuraCMS ini dapat dilihat pada Gambar 7.

Gambar 7. Tampilan awal dari AuraCMS
Sumber : *Implementasi*

Solusi pencegahan terhadap serangan *Cross Site Scripting*

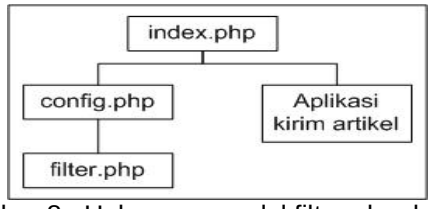
Pencegahan terhadap serangan *Cross Site Scripting* ini dilakukan dengan membuat sebuah *file* fungsi *filter* terhadap masukan yang diberi-kan oleh *user, file* fungsi yang dimak-sud disini adalah *file* fungsi.php. Dari *file* inilah *filter* terhadap masukkan yang diberikan oleh *user* dilakukan.

Modul pencegahan *Cross Site Scripting* (XSS)

Modul ini dibuat untuk mencegah adanya lubang keamanan pada aplikasi web yang dibuat dalam hal lubang keamanan yang dimaksud adalah kelemahan terhadap serang-an XSS. Modul ini saling terhubung dengan modul aplikasi lain yang membutuhkan interaksi dengan *user*, modul ini juga terhubung dengan modul *administrator* yang berfungsi sebagai verifikasi *user* sebagai *administrator* atau bukan.

Implementasi pada aplikasi kirim artikel

Implementasi modul filter.php pa-da aplikasi kirim artikel ini ditunjuk kan dalam Gambar 8.



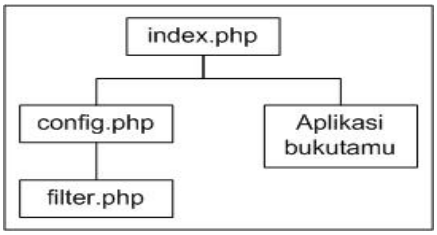
Gambar 8. Hubungan modul filter.php dengan modul aplikasi kirim artikel

Sumber : *Implementasi*

Pada saat *user* mengakses halaman kirim artikel, maka secara tidak langsung ada beberapa modul yang ikut berjalan yaitu modul filter.php, config.php, dan index.php.

Implementasi pada aplikasi bukutamu

Implementasi modul filter.php pa-da aplikasi bukutamu ini ditunjukkan dalam Gambar 9.



Gambar 9 Hubungan modul filter.php dengan modul aplikasi bukutamu
Sumber : *Implementasi*

Sama seperti pada aplikasi kirim artikel yang telah dijelaskan sebe-lumnya, secara tidak langsung ketika *user* mengakses aplikasi bukutamu maka *user* tersebut juga mengakses beberapa modul pendukung seperti modul filter.php, config.php, dan index.php.

PENUTUP

Kesimpulan

1. *Cross Site Scripting* merupakan kelemahan yang dapat dieksploi-tasi dengan mudah.
2. Metode serangan *Cross Site Scripting* dilakukan dengan cara memberikan *input* berupa kode/ skrip, terhadap aplikasi web yang membutuhkan interaksi dengan user.
3. Akibat yang ditimbulkan oleh *Cross Site Scripting* adalah peng-gunaan *cookie* oleh *user* yang tidak berkepentingan dalam hal ini *attacker* sehingga *attacker* bisa mengakses dan *log in* dengan menggunakan *cookie* curian ter-sebut.
4. Pemilihan metoda pencegahan disesuaikan dengan kebutuhan dari penyedia layanan yang ada. Yang terpenting, penyedia layan-an mampu menjamin keamanan data dari pengguna yang ada.
5. Para pengembang web hendak-nya memperhatikan program apli kasi yang dibuat sebelum diguna-kan. Beberapa hal yang perlu dilakukan adalah :
   - § Sebaiknya perlu dilakukan *debugging* sebelum program aplikasi yang dibuat digunakan secara *online*.
   - § Pembuatan sebuah modul *filter* sangat diperlukan dalam mem-bangun sebuah aplikasi web.
   - § Pada pemrograman PHP terdapat beberapa fungsi yang bisa digunakan untuk menga-mankan aplikasi yang dibuat, fungsi-fungsi tersebut yaitu:
     - ü htmlspecialchars()
     - ü stripslashes()
     - ü strip_tags()
     - ü htmlentities()

Saran

Ada beberapa saran pengamanan yang cukup efektif, saran tersebut antara lain:

- § Para pengembang web, hendak-nya melakukan pengujian untuk tiap halaman yang dibangun, ju-ga dilakukan pengecekan ulang untuk tiap masukan dari penggu-na, untuk menghindari celah serangan *Cross Site Scripting* (XSS).
- § Untuk *user* hendaknya tidak memberikan *input* berupa skrip atau kode terhadap aplikasi web yang membutuhkan interaksi dengan *user*.
- § *Setting browser* perlu dilakukan untuk memastikan bahwa peng-aturan *cookies* pada *browser* yang digunakan untuk sebuah web telah aman.

DAFTAR PUSTAKA

[ANN-02] Anonim, November 2002,

"Internet Computing", Akses : http://computer.org/internet/.

[END-02] Endler, David, May 2002, "The Evolution of Cross-Site Scripting Attacks", iDEFENSE Labs. Akses : http://www.idefense.com/advisories/xss/.

[HEN-03] Hendrickx, Michael, 2003, "XSS : Cross Site Scripting, Detection and Prevention", Scanit.

[IRA-06] Irawan, Taufik, Desember 2006, "Konsep Dasar Basis Data", Komunitas Mahasiswa Informatika Independen UAD Yogyakarta.

[KAD-02] Kadir, Abdul, 2002, "Dasar Pemrograman Web Dinamis Menggunakan PHP", Penerbit ANDI Yogyakarta.

[KLE-03] Klein, Amit, August 2003, "Cross Site Scripting Explained", Sanctum Inc.Akses : http://www.SanctumInc.com/advisories/cross site scripting/.

[MCC-03] Mcclure, Stuart, Saumil Shah, Sheeraj Shah, 2003, "Web Hacking : Serangan dan Pertahanannya", Penerbit ANDI Yogyakarta.

[OLL-03] Ollman, Gunter, 2003, "HTML Code Injection and Cross-site scripting, Understanding the cause and effect of CSS (XSS) Vulnerabilities", ISS Advisor.

[PRA-02] Prasetyo, Didik Dwi, Oktober 2002, "Belajar Sendiri Administrasi Database Server MySQL", PT Elex Media Komputindo.

[PRA-05] Prasetyo, Didik Dwi, Februari 2005, "Solusi Menjadi Web Master Melalui Manajemen Web Dengan PHP", PT Elex Media Komputindo.

[RAF-01] Rafail, Jason, November 2001, "Cross Site Scripting Vulnerabilities", CERT Coordination Center.

[RAH-02] Rahardjo, Budi, September 2002, "Keamanan Sistem Infromasi Berbasis Internet", PT Insan Infonesia – Bandung & PT INDOCISC – Jakarta.

[SPE-03] Spett, Kevin, September 2003, "Cross Site Scripting - are your web application vulnerable", SPILabs.Akses : http://www.spydinamics.com/white papers/xss/.

[ZUC-03] Zuchlinski, Gavin, Oktober 2003, "Advanced Cross Site Scripting", Akses : http://libox.net/advisories/cross site scripting/.

[HAR-02] Hartanto, Antonius Aditya, 2002, "Teknologi e-Learning Berbasis PHP dan MySQL", PT Elex Media Komputindo.