# SIMULASI KEAMANAN PADA APLIKASI WEB DENGAN WEB APPLICATION FIREWALL

Jamie Karisma Anggreana

Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail : jamiekarisma@gmail.com

## ABSTRAK

Aplikasi web adalah suatu aplikasi yang diakses menggunakan *browser* web melalui jaringan internet. Jenis-jenis aplikasi web sangat beragam, seperti toko *online* dan *website* informasi yang dalam beberapa tahun terakhir menjadi target serangan *hacker*. Adapun ancaman yang sering terjadi diantaranya merupakan ancaman *SQL Injection, Cross-Site Scripting* dan *Unrestricted File Upload.*

*SQL Injection* adalah jenis ancaman yang mengizinkan *query SQL* dapat di-*inject* oleh *client* kemudian diteruskan oleh *server* untuk dieksekusi. *Cross-Site Scripting* adalah ancaman yang mengizinkan kode (*client side script*) dimasukan ke dalam suatu *website* yang dapat dijalankan pada sisi *client. Unrestricted File Upload* adalah jenis ancaman yang mengizinkan *user* untuk menyisipkan *file*, biasanya *file* yang di-*upload* telah disediakan oleh *attacker* dan digunakan untuk mengeksekusi kode jarak jauh (*Remote Code Execution*).

Untuk mengatasi tipe ancaman tersebut dibutuhkan suatu mekanisme keamanan. *Web Application Firewall* adalah Suatu metode untuk pengamanan pada aplikasi web yang memiliki beberapa fungsi, mulai dari monitoring trafik, *secure directory*, pemfilteran *string* dan proteksi terhadap serangan seperti *SQL Injections*, *Cross-Site Scripting*, dan *Unrestricted File Upload*.

Berdasarkan pengujian yang dilakukan disimpulkan bahwa metode *Web Application Firewall* dapat menjadi mekanisme pertahanan terhadap ancaman *SQL Injection, Cross Site Scripting* dan *Unrestricted File Upload.*

Kata Kunci: Keamanan, Web Application Firewall, SQL Injections, Cross-Site Scripting, Unrestricted File Upload.

## 1. PENDAHULUAN

Aplikasi web adalah suatu aplikasi yang diakses menggunakan *browser* web melalui jaringan internet [1]. Jenis-jenis aplikasi web sangat beragam, seperti toko *online* dan *website* informasi yang dalam beberapa tahun terakhir menjadi target serangan *hacker* [2]. Berdasarkan data dari *The Open Web Application Security Project* (OWASP) pada tahun 2013, yang melakukan *survey* mengenai ancaman yang sering terjadi pada aplikasi web diantaranya merupakan ancaman *SQL Injection, Cross-Site Scripting* dan *Unrestricted File Upload* [3].

*SQL (Structure Query Language) Injection* adalah jenis ancaman yang mengizinkan *query SQL* dapat di-*inject* oleh *client* kemudian diteruskan oleh *server* untuk dieksekusi [4]. Ancaman ini terjadi pada *database* aplikasi web. *Cross-Site Scripting* atau sering dikenal dengan XSS adalah ancaman yang mengizinkan kode (*client side script*) dimasukan ke dalam suatu *website* yang dapat dijalankan pada sisi *client. Unrestricted File Upload* adalah jenis ancaman yang mengizinkan *user* untuk menyisipkan *file*, biasanya *file* yang di-*upload* telah disediakan oleh *attacker* dan digunakan untuk mengeksekusi kode jarak jauh (*Remote Code Execution*) [5].

Keamanan pada aplikasi web kurang mendapat perhatian dari *developer* sehingga menyebabkan serangan di internet dilakukan terhadap sebuah *web*, untuk itu diperlukan pengamanan khusus yakni dengan *Web Application Firewall* [6]. *Web Application Firewall* adalah Suatu metode untuk pengamanan pada aplikasi web, yang berupaya mencegah adanya ancaman dari *attacker* [7]. *Web Application Firewall* dapat bekerja dengan terlebih dahulu melakukan konfigurasi tambahan pada web *server* dan tidak perlu melakukan perubahan pada *script* pembangun aplikasi, sehingga dapat diterapkan pada aplikasi yang sudah berjalan.

*Web Application Firewall* (WAF) memiliki beberapa fungsi, mulai dari monitoring trafik, *secure directory*, pemfilteran *string* dan proteksi terhadap serangan seperti *SQL Injections*, *Cross-Site Scripting*, dan *Unrestricted File Upload*. *Web Application Firewall* membentuk lapisan keamanan yang dapat mendeteksi dan mencegah serangan pada aplikasi web. Adapun tindakan yang dapat dilakukan seperti menghentikan *request* dengan status 403 *forbidden* dan juga dapat melakukan *virtual patching.* Dimana *virtual patching* merupakan suatu

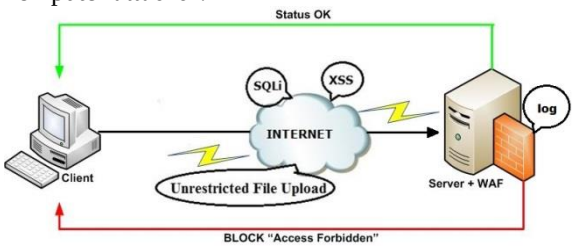
*rule* yang diterapkan untuk melakukan *patch* tanpa menyentuh aplikasi guna memblokir *request* yang berbahaya [8].

Berdasarkan uraian tersebut, pada pembangunan aplikasi web, segi keamanan kurang mendapat perhatian khusus dari *develeoper*. Dimana bentuk serangan yang sering terjadi berdasarkan data dari OWASP diantaranya *SQL Injection, Cross-Site Scripting*, dan *Unrestricted File Upload* masih menjadi suatu ancaman yang membahayakan pada aplikasi web. Salah satu metode pengamanan yang dapat diterapkan yaitu dengan menggunakan *Web Application Firewall.* Dimana metode tersebut akan diterapkan pada *website* penjualan dan untuk bentuk pengujian pertahanan, akan dicoba disimulasikan teknik serangan yang paling sering terjadi. Dengan sistem pertahanan ini, diharapkan dapat memberikan rekomendasi untuk meningkatkan segi keamanan, sehingga aplikasi web yang dibangun tidak hanya mempunyai desain yang baik namun juga terjaga integritas datanya. Penelitian yang mengambil topik *security* pada aplikasi web ini diberi judul "**SIMULASI KEAMANAN PADA APLIKASI WEB dengan *WEB APPLICATION FIREWALL*".**

## 2. ISI PENELITIAN

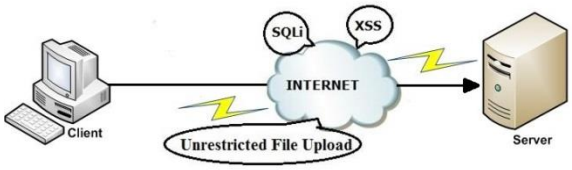
### 2.1 Simulasi

Perancangan jaringan yang akan digunakan terdiri dari sebuah komputer sebagai *server*, dan komputer *attacker.*




Gambar 1. Arsitektur Simulasi dengan WAF

Skenario yang dilakukan yaitu *attacker* yang terkoneksi dengan internet akan mengakses layanan pada *server* yaitu sebuah *e-commerce* dengan mekanisme keamanan menggunakan *Web Application Firewall*, kemudian *attacker* menginjeksikan kode tertentu pada aplikasi web dengan metode *SQL Injection, Cross Site Scripting* dan *Unrestricted File Upload*.




Gambar 2. Simulasi *non* WAF

Skenario yang dilakukan yaitu *attacker* yang terkoneksi dengan internet akan mengakses layanan pada *server* yaitu sebuah *e-commerce* tanpa menggunakan *Web Application Firewall*, kemudian *attacker* menginjeksikan kode tertentu pada aplikasi web dengan metode *SQL Injection, Cross Site Scripting* dan *Unrestricted File Upload*.

### 2.2 Pengujian

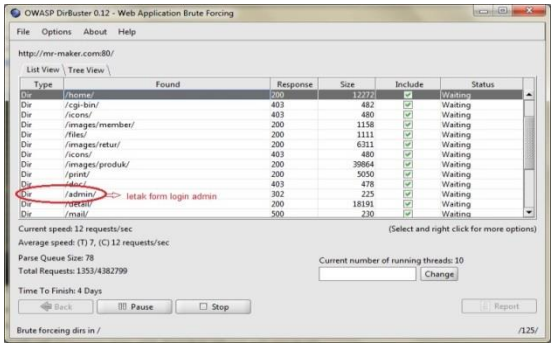
Pengujian sistem merupakan hal terpenting yang bertujuan untuk menemukan kesalahan-kesalahan atau kekurangan-kekurangan pada metode yang diuji. Pengujian bermaksud untuk mengetahui metode *Web Application Firewall* yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan.

Pengujian metode ini dilakukan pada aplikasi web tanpa metode keamanan dan pada aplikasi web dengan metode keamanan, dengan menginjeksikan tiga tipe serangan yang sering terjadi pada aplikasi web yaitu *SQL Injection, Cross Site Scripting* dan *Unrestricted File Upload.*

#### 2.2.1 Pengujian Ancaman SQL Injection

*Attacker* yang terkoneksi dengan internet akan mengakses layanan pada server yaitu sebuah e-commerce yang tidak menggunakan *Web Application Firewall* dengan metode *SQL Injection* untuk mem-*bypass form login* admin pada aplikasi web tersebut. Pada pengujian ancaman *SQL Injection* dilakukan dua langkah penting yaitu mencari *web form* dan manipulasi *input*-an *login.*

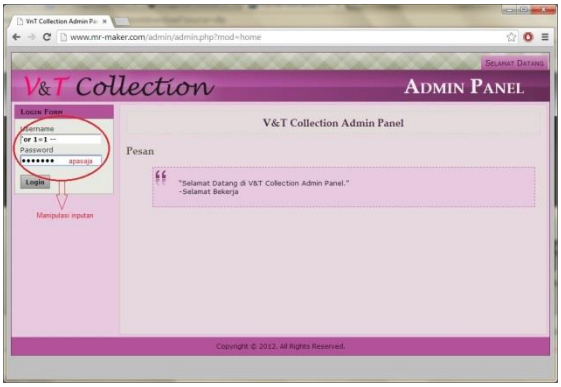
*Attacker* akan mencari letak dari *form login* yang terdapat pada aplikasi web dengan menggunakan *tool* pencarian direktori dan *file* sensitif pada web aplikasi.




Gambar 3 Pencarian Direktori Admin

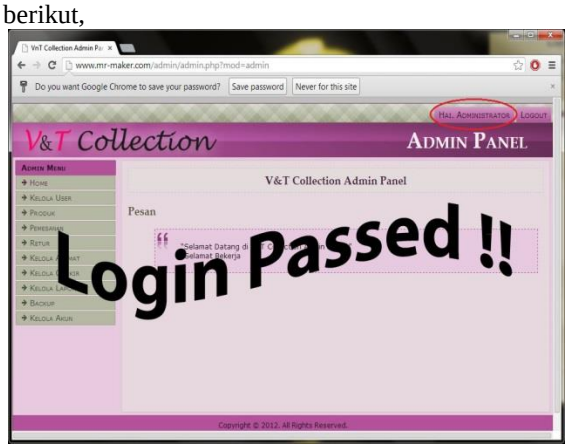
Setelah melakukan pencarian letak dari *form login* admin, *attacker* memanipulasi *input*-an *login* untuk mem-*bypass* proses *authentikasi*, misalnya *username* = ' or 1=1 -- dan *password* = apasaja.

.




Gambar 4 Manipulasi *Input*-an *Form Login*

Hasil dari manipulasi *input*-an *login* adalah sebagai berikut,




Gambar 5 Berhasil *Login*

Logikanya jika *password* diisi dengan nilai sembarang dan *username* diisi dengan nilai admin' OR 1=1 -- , maka eksekusi akan berubah. *Query* akan meminta *username* dan *password* ke *database* dengan kondisi *username*= admin atau 1=1, dan *password*=apasaja (*input*-an sembarang *password*). Kondisi ini mengakibatkan *query* yang dijalankan selalu dalam keadaan benar (*true*) dan *database* yang mengembalikan nilai benar tersebut, kemudian menampilkan halaman admin.

## 2.2.2 Pengujian Ancaman Cross Site Scripting

Pada pengujian ancaman *Cross Site Scripting* dilakukan pada *form* "hubungi kami" yakni dengan menginjeksi sebuah *script HTML* sehingga web terganggu.




Gambar 6 Ancaman XSS

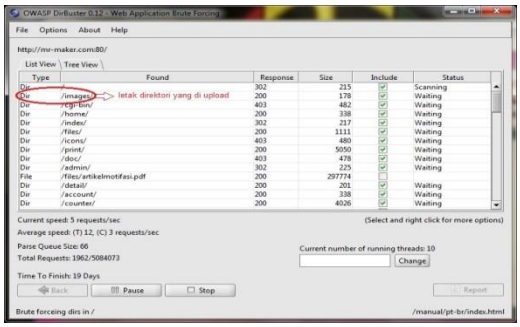
Berdasarkan gambar 4.8 bahwa ketika pesan diisi dengan sebuah *script HTML* maka *script* tersebut akan dieksekusi begitu saja, karena tidak adanya *filter* pada variabel pesan.

## 2.2.3 Pengujian Ancaman Unrestricted File Upload

Pada pengujian ancaman *Unrestricted File Upload* dilakukan dua langkah penting yaitu mencari direktori web tempat diletakannya *file* yang akan disisipkan dan meng-*upload file* yang telah disediakan untuk merubah halaman utama web aplikasi target (*deface*).

## 2.2.4 Pencarian Directori Web

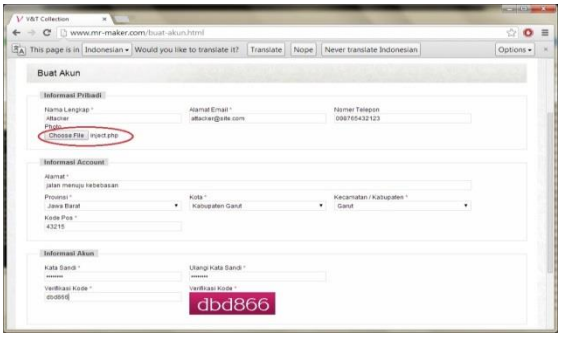
*Attacker* akan mencari letak dari direktori *file* yang di-*upload* menggunakan *tool* pencarian direktori dan *file* sensitif pada web aplikasi.




Gambar 7 Pencarian Direktori *Images*

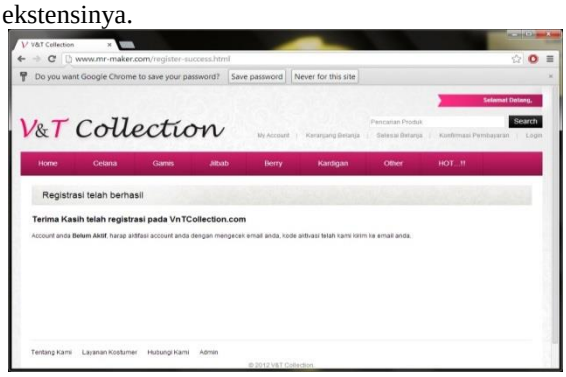
## 2.2.5 Eksekusi Form Upload

Setelah melakukan pencarian letak direktori *images*, pengujian ancaman *Unrestricted File Upload* dilakukan pada *form* "buat akun" dimana akan di-*upload* sebuah *file* untuk mengganti halaman utama *website* (*deface*).




Gambar 8 *Upload File Form Buat Akun*

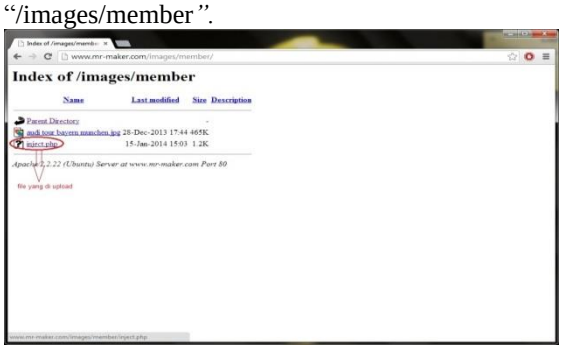
Ketika *inject.php* di-*upload* web aplikasi mengeksekusi begitu saja tanpa mem-*filter* ekstensinya.




Gambar 10 Registrasi Berhasil

### 2.2.6 Eksekusi File Upload

Setelah berhasil meng-*upload file inject.php*, lalu temukan *file* tersebut pada direktori "/images/member".




Gambar 11 Letak *File Upload*

Setelah *file inject.php* ditemukan seperti terlihat pada gambar, lalu eksekusi dengan cara meng-klik file tersebut.



Gambar 12 Eksekusi *Inject.php*

Logikanya ketika *file inject.php* dieksekusi, *file* tersebut membuat *file* baru yang bernama *redirect.php* pada direktori "/images/member" dan pada halaman utama serta me-*replace file index.php* sehingga tampilan awal web aplikasi berubah.



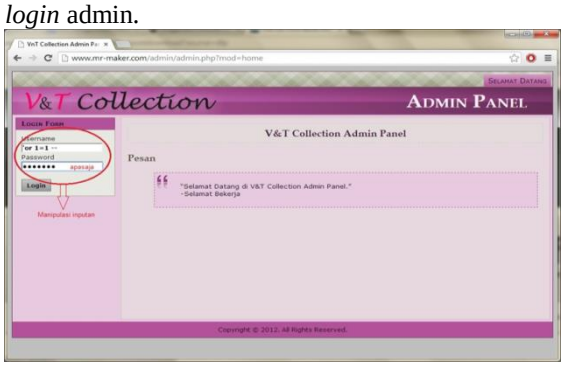

Gambar 13 Deface

## 2.3 Pengujian Keamanan

Pengujian keamanan meliputi pengujian *Web Application Firewall* dengan tiga tipe serangan yang sering terjadi pada aplikasi web, yaitu *SQL Injection*, *Cross Site Scripting*, *Unrestricted File Upload* sebagai tolak ukur yang menentukan apakah *Web Application Firewall* dapat meminimalisir ancaman yang sering terjadi pada aplikasi web.

### 2.3.1 Pengujian Metode WAF Pada SQL Injection

Pada pengujian ini, web aplikasi yang telah menggunakan metode *Web Application Firewall* diuji dengan tipe ancaman *SQL Injection* pada *form login* admin.



Gambar 14 Manipulasi Input-an Form Login (WAF)




Gambar 15 *Forbidden Login*

### 2.3.2 Pengujian Metode WAF Pada Cross Site Scripting

Pada pengujian ini, web aplikasi yang telah menggunakan metode *Web Application Firewall* diuji dengan tipe ancaman *Cross Site Scripting* pada *form* hubungi kami. Berikut menunjukan manipulasi *input*-an pada *form* hubungi kami yaitu <marquee> <h1 style="font-size:5em;"> TULISAN BERGERAK </h1></marquee>.

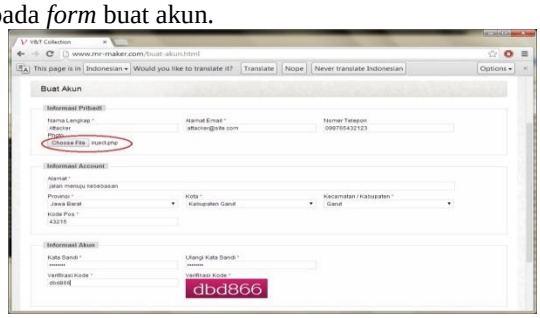



Gambar 16 Ancaman *Cross Site Scripting* (WAF)

Terlihat bahwa pesan yang diisikan tidak tampil dan terdapat *feedback* bahwa akses *forbidden*.

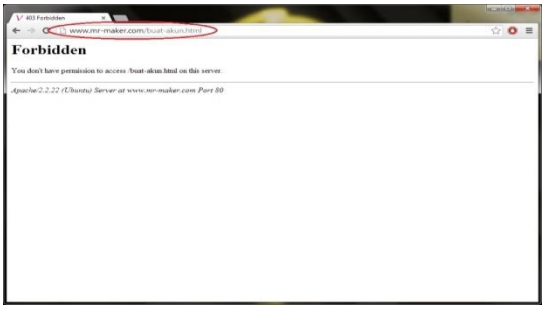
### 2.3.3 Pengujian Metode WAF Pada Unrestricted File Upload

Pada pengujian ini, web aplikasi yang telah menggunakan metode *Web Application Firewall* diuji dengan tipe ancaman *Unrestricted File Upload* pada *form* buat akun.




Gambar 17 *Form Upload* Buat Akun (WAF)

Ketika *inject.php* di-*upload* web aplikasi dengan *Web Application Firewall* tidak mengeksekusi begitu saja namun mem-*filter* ekstensinya.




Gambar 18 *Forbidden* Buat Akun

## 5 PENUTUP

Berdasarkan pengujian yang telah dilakukan pada aplikasi web yang tidak menggunakan dan yang menggunakan *Web application Firewall*, dapat ditarik hasil bahwa metode keamanan pada aplikasi web dengan metode *Web Application Firewall* dapat direkomendasikan sebagai metode keamanan pada *websiet*, karena:

1. Dapat menangani tiga tipe ancaman yang sering terjadi pada aplikasi web yaitu, *SQL Injection*, *Cross Site Scripting* dan *Unrestricted File Upload.*
2. Dapat diterapkan pada aplikasi web yang sudah berjalan dan tidak perlu melakukan perubahan pada *script* pembangun aplikasi.
3. *Rules* yang digunakan untuk menangani ancaman pada aplikasi web dapat disesuaikan dengan kebutuhan.

**Tabel 4.1 *Review* Pengujian**

| NO | Tipe Ancaman | Keterangan |
|---|---|---|
| 1 | SQL Injection | Dapat Ditangani WAF |
| 2 | Cross Site Scripting (XSS) | Dapat Ditangani WAF |
| 3 | Unrestricted File Upload | Dapat Ditangani WAF |

## 6 DAFTAR PUSTAKA

[1] Acunetix. (2013). *Acunetix Web Application Security*. Dipetik July 24, 2013, dari Acunetix Web Application Security Website: http://www.acunetix.com/websitesecurity/web-applications

[2] OWASP Foundation. (2008, July 17). *Best Practices Guide WAF*. Dipetik July 27, 2013, dari The Open Web Application Security Project Website: https://www.owasp.org/index.php/File:Best_Practices_Guide_WAF_v104.en.pdf

[3] OWASP Foundation. (2013, June 12). *Top Ten Attacks*. Dipetik August 10, 2013, dari The Open Web Application Security Project Website: https://www.owasp.org/index.php/Top_Ten

[4] Digdo, G. P. (2012). *Analisis Serangan dan Keamanan pada Aplikasi Web*. Jakarta: Elex Media Komputindo.

[5] OWASP Foundation. (2013). *Unrestricted File Upload*. Dipetik November 21, 2012, dari The Open Web Application Security Project Website: https://www.owasp.org/index.php/Unrestricted_File_Upload

[6] Bryan, S., & Vincent, L. (2011). *Web Application Security, A Beginner Guide.* United State: The McGraw-Hill Companies.

[7] OWASP Foundation. (2013). *Web Application Firewall*. Dipetik July 27, 2013, dari The Open Web Application Security Project Website: https://www.owasp.org/index.php/Web_Application_Firewall

[8] Kitt99. (2012, June 1). *RNDC*. Dipetik July 30, 2013, dari Research And Development Center Website: http://rndc.or.id/ezine/ezine-rndc-r2-6-2012/pengamanan-web-server-dengan-modsecurity.txt

[9] Wahana Komputer. (2001). *Penanganan Jaringan Komputer*. Yogyakarta: ANDI.

[10] McClure , S., Saumil, S., & Sheeraj, S. (2003). *Web Hacking, Serangan dan Pertahanannya*. Yogyakarta: ANDI.

[11] Simarmata, J. (2006). *Pengamanan Sistem Komputer*. Yogyakarta: ANDI.

[12] Indrajit, R. E. (2012). *Aneka Ragam Serangan di Dunia Maya*. Dipetik October 6, 2013, dari http://www.idsirtii.or.id/content/files/cyber6/1baf0eec40eab68e9257f24536ae808f.pdf

[13] L41n, a. 4. (2008, March 28). *ECHO*. Dipetik October 6, 2013, dari echo|zine, volume 6 issue 18: http://ezine.echo.or.id/ezine18/e18.006.txt

[14] Ravii, M. (2011). *Sistem Keamanan Komputer dan Jaringan*. Dipetik October 6, 2013, dari http://ravii.staff.gunadarma.ac.id/Downloads/files/35408/Keamanan+jaringan.pdf

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

[16] Grossman, J., Robert Hansen, Petko D.Petkov, Anton Rager, & Seth Fogie. (2007). *XSS Attacks: Cross Site Scripting Exploits and Defense.* Burlington: Syngress Publishing, Inc.

[17] Suprianto, D. (2008). *Buku Pintar Pemrograman PHP*. Bandung: Oase Media.

[18] Abdul Kadir. (2002). *Konsep & Tuntunan Praktis Basis Data.* Yogyakarta: ANDI.

[19] World Friend Indonesia. (2010). *Web Server*. Dipetik October 6, 2013, dari http://worldfriend.web.id/pengertian-web-server

[20] Entomy Network. (2013). *Virtual Private Server*. Dipetik December 10, 2013, dari www.entomy.com/VPS-Server-Configuration-Guide.pdf

[21] OWASP Foundation. (2009, October 22). *OWASP DirBuster Project*. Dipetik December 10, 2013, dari https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project

[22] Ivan Ristic. (2010). *Modsecurity Hanbook.* Feisty Duck.

[23] Ubuntu Manual Team. (2012). *Getting Started Ubuntu 12.04*. Dipetik December 10, 2013, from http://files.ubuntu-manual.org/manuals/getting-started-with-ubuntu/12.04/en_US/screen/Getting%20Started%20with%20Ubuntu%2012.04.pdf

[24] Webmin. (2013). *webmin kontrol panel*. Dipetik December 10, 2013, from http://www.webmin.com

[25] WinSCP. (2013). *open source file transfer client*. Dipetik December 10, 2013, from http://winscp.net/eng/index.php