BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Pada penelitian ini digunakan empat buah tinjauan pustaka, yang pertama ditulis oleh Feri Setiyawan. Web server pada penelitian ini menggunakan nginx yang dilengkapi dengan naxsi sebagai web application firewall. Pembahasan mengenai perbandingan antara sebelum dan sesudah implementasi naxsi terhadap serangan *SQL injection* (Feri Setiyawan, 2014).

Pustaka kedua ditulis oleh Albi Alamsyah mengenai pengujian keamanan setelah implementasi *ModSecurity* terhadap empat jenis serangan yaitu *SQL Injection, Cross Site Scripting (XSS), Local File Inclusion (LFI) dan Remote File Inclusion (RFI)* (Albi Alamsyah, 2016).

Pustaka ketiga ditulis oleh Gilang Ramadhan, web server yang digunakan adalah nginx yang dilengkapi dengan naxsi dan apache yang dilengkapi dengan modsecurity. Pembahasan mengenai studi kasus pada sebuah instansi yang membandingan instalasi nginx yang dilengkapi naxsi dengan apache yang dilengkapi dengan *ModSecurity* (Gilang Ramadhan, 2014).

Pustaka keempat ditulis oleh Aditya, web server yang digunakan adalah apache yang dilengkapi dengan *ModSecurity*. Pembahasan mengenai pembacaan log oleh aplikasi jwall auditconsole untuk dijadikan alat pelapor adanya insiden (Aditya Noor Sandy, 2014).

Detail dari tinjauan pustaka disajikan dalam bentuk tabel yang terlihat pada tabel 2.1 Tabel Perbandingan.

Tabel 2 1 Tabel Perbandingan

Parameter / Penulis	**Objek**	**Metode/Alat**	**Bahasa Pemrograman**	**Interface**
Feri Setiyawan (UIN SUKA) tahun 2014	Pencegah *SQL Injection*	Naxsi, Nginx	PHP (Joomla)	-
Albi Alamsyah (Universitas Muhammadiyah Jember) tahun 2016	Pengujian keamanan terhadap *SQL Injection, Cross Site Scripting (XSS), Local File Inclusion (LFI) dan Remote File Inclusion (RFI)*	ModSecurity, Apache	PHP (DVWA)	-
Gilang Ramadhan (Unikom) tahun 2014	Perbandingan keamanan dua jenis *Web Application Firewall (WAF)*	Naxsi, Nginx, ModSecurity, Apache	PHP (website XecureIT)	-
Aditya Noor Sandy (UPN Veteran Jawa Timur) tahun 2014	Pelaporan dugaan tindakan intrusi pada website	ModSecurity, Apache	PHP	Jwall Auditconsole
Usulan	Pengembangan filter *ModSecurity*	ModSecurity, Apache	PHP (Wordpress dan Joomla)	-

	untuk *File Upload, PHP Code Injection* dan *PHP Object Injection*			

2.2 Dasar Teori

Beberapa pendekatan yang diambil pada penelitian ini diuraikan secara urut penomoran seperti berikut.

2.2.1 Tipe Ancaman Pada Aplikasi Web

Beberapa jenis serangan yang digunakan pada penelitian ini diuraikan seperti berikut.

1. File Upload

   File yang diunggah mewakili risiko yang signifikan terhadap aplikasi. Langkah pertama serangan pada umumnya adalah mendapatkan beberapa kode pada sistem yang akan diserang. Kemudian serangan hanya perlu mencari jalan untuk mendapatkan kode yang dieksekusi. Menggunakan upload file membantu penyerang menyelesaikan langkah pertama.

   Konsekuensi dari upload file yang tidak terbatas dapat bervariasi, termasuk pengambilalihan sistem yang lengkap, sistem berkas atau database yang kelebihan beban, serangan penerusan ke sistem back-end, serangan sisi klien, atau penghindaran sederhana. Itu tergantung pada apa aplikasi yang dilakukan dengan file upload dan terutama di tempat penyimpanannya.

2. Code Injection

Code Injection adalah istilah umum untuk jenis serangan dengan mengirim kode program kemudian dieksekusi oleh aplikasi. Jenis serangan ini memanfaatkan penanganan data yang tidak sempurna. Jenis serangan ini biasanya dimungkinkan karena kurangnya validasi data input / output yang tepat, misalnya: karakter yang diizinkan, format data dan jumlah data yang diharapkan. *Code Injection* berbeda dengan *Command Injection* karena penyerang hanya dibatasi oleh fungsi bahasa pemrograman yang digunakan.

3. PHP Object Injection

PHP Object Injection adalah tingkat kerentanan aplikasi yang memungkinkan penyerang melakukan berbagai jenis serangan berbahaya, seperti *Code Injection, SQL Injection, Path Traversal* dan *Denial of Service*, tergantung pada konteksnya. Kerentanan terjadi ketika masukan yang diberikan pengguna tidak disterilkan dengan baik sebelum dikirim ke fungsi PHP *unserialize()*. Karena PHP memungkinkan serialisasi objek, penyerang bisa melewati string serial ke panggilan *unserialize()* yang rentan, sehingga menghasilkan sembarang objek PHP ke dalam lingkup aplikasi.

Dari uraian tiga buah ancaman pada aplikasi web di atas, terdapat perbedaan metode yang dilakukan oleh penyerang dalam memanfaatkannya. Berikut ini merupakan tabel beberapa metode penyerangan yang biasa digunakan pada masing – masing jenis kelemahan.

Tabel 2 2 Metode Penyerangan

Kelemahan	Metode
File Upload	Pengiriman file php secara langsung
	Pengiriman file php dengan ekstensi ganda

PHP Code Injection	Manipulasi parameter untuk bypass seleksi
	Penyisipan kode ke dalam database
PHP Object Injection	Pengiriman serialisasi objek secara langsung
	Pengiriman serialisasi objek dengan enkripsi

Beberapa aplikasi website yang memiliki kelemahan seperti yang telah diuraikan dan akan diteliti ditampilkan dalam bentuk tabel berikut ini.

Tabel 2 3 Informasi Kelemahan

Nama	Kelemahan	Jenis Kelemahan
Wordpress Plugin Mac Photo Gallery 2.7	Tidak terdapat verifikasi terhadap file yang dikirimkan.	*File Upload*
Wordpress Plugin Asset Manager 0.2	Tidak terdapat verifikasi terhadap file yang dikirimkan.	
Wordpress Plugin AllWebMenus < 1.1.9	Fitur update dengan menggunakan file kompresi zip dapat dieksekusi oleh siapa saja.	
Wordpress Plugin LearnDash 2.5.3	Fitur upload melakukan pemotongan string ekstensi file yang bisa dimanfaatkan dengan ekstensi ganda.	
Wordpress Plugin Insert PHP < 3.3.1	Program gagal melakukan validasi terhadap hak pengguna terhadap id posting.	*PHP Code Injection*
Wordpress Plugin W3 Total Cache 0.9.2.3	Tag khusus dengan nama “mfunc” mempunyai fungsi spesial untuk penambahan kode php.	
Joomla Component com_civicrm 4.2.2	Pembuatan file dengan fungsi *fwrite* yang bisa dieksekusi siapa saja.	
Wordpress Plugin Ultimate Product Catalog <= 4.2.24	Fungsi *unserialize* langsung diberi parameter nilai cookie tanpa validasi.	*PHP Object Injection*
Joomla! 1.5 < 3.4.5 ‘x-forwarded-for’	Tidak terdapat verifikasi terhadap data browser yang dimasukkan ke	

	dalam database.	
Joomla! 3.0.2 – 'highlight.php'	Fungsi *unserialize* diberi parameter nilai dari variabel *highlight* tanpa validasi.	
Joomla! 3.0.3 – 'remember.php'	Fungsi *unserialize* diberi parameter nilai cookie tanpa validasi.	

2.2.2 Web Application Firewall

Web Application Firewall (WAF) adalah *firewall* untuk aplikasi HTTP. Menerapkan seperangkat aturan terhadap permintaan atau keluaran HTTP. Umumnya, aturan ini mencakup serangan umum seperti *cross-site scripting (XSS)* dan *SQL injection*. Sementara proxy umumnya melindungi klien, *WAFs* melindungi server. *WAF* digunakan untuk melindungi aplikasi web atau kumpulan aplikasi web tertentu. *WAF* bisa dianggap sebagai reverse proxy. *WAF* bisa berbentuk alat, plugin server, atau filter, dan mungkin disesuaikan dengan aplikasi. Upaya untuk melakukan kustomisasi dapat menjadi signifikan dan perlu dipertahankan menyesuaikan dengan perubahan yang dilakukan pada aplikasi.

2.2.3 ModSecurity

ModSecurity adalah open source, cross platform *web application firewall (WAF)* yang dikembangkan oleh Trustwave's SpiderLabs. *ModSecurity* memiliki bahasa pemrograman berbasis event yang kuat yang memberikan perlindungan dari berbagai serangan terhadap aplikasi web dan memungkinkan pemantauan lalu lintas HTTP, logging dan analisis secara real-time. Dengan lebih dari 10.000

penyebaran di seluruh dunia, *ModSecurity* adalah *WAF* yang paling banyak digunakan. Skenario paling penting penggunaan *ModSecurity* adalah :

1. Real-time application security monitoring and access control

   ModSecurity memberi akses ke arus lalu lintas HTTP secara real-time, beserta kemampuan untuk memeriksanya. Dengan tambahan mekanisme penyimpanan permanen *ModSecurity*, dapat dilakukan pelacakan elemen sistem dari waktu ke waktu dan melakukan korelasi peristiwa. Karena *ModSecurity* menggunakan permintaan dan penyangga respons penuh, maka dapat dilakukan pemblokiran.

2. Full HTTP traffic logging

   Web server secara tradisional menyimpan sangat sedikit log yang mengandung informasi untuk keamanan dan bahkan dengan melakukan konfigurasi khusus tidak bisa didapatkan semua informasi yang dibutuhkan. *ModSecurity* menyediakan kemampuan untuk mencatat apapun yang dibutuhkan, termasuk data transaksi mentah, yang penting untuk forensik. Selain itu, bisa dipilih transaksi yang akan disimpan, bagian mana dari transaksi yang masuk, dan komponen mana yang disterilkan.

3. Continuous passive security assessment

   Penilaian keamanan aktif sebagian besar merupakan kegiatan yang terjadwal, di mana tim independen bersumber untuk mencoba melakukan serangan simulasi. Penilaian keamanan pasif terus menerus adalah variasi pemantauan real-time, di mana berfokus pada perilaku pihak eksternal, dengan mengamati perilaku sistem itu sendiri. Hal ini adalah sistem peringatan dini yang bisa

mendeteksi jejak kelainan perilaku sistem dan kelemahan keamanan sebelum dieksploitasi.

4. Web application hardening

   Salah satu kegunaan *ModSecurity* adalah digunakan untuk mempersempit data HTTP yang diterima, misalnya metode, header, jenis konten dan lain – lain. Dengan *ModSecurity* mungkin juga dilakukan perbaikan masalah manajemen sesi, serta kerentanan pemalsuan data pada lalu lintas situs.