Jurnal Informatika Terpadu

https://journal.nurulfikri.ac.id/index.php/JIT
ISSN ONLINE : 2460-8998

PENERAPAN *HIGH AVAILABILITY WEB SERVER* MENGGUNAKAN NGINX DAN MODSECURITY

Taufiqul Hidayah[1], Henry Saptono[2]

[1, 2]Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Jakarta Selatan, DKI Jakarta, Indonesia 12640
bakarsapi@gmail.com , henry@nurulfikri.co.id

Abstract

Advances in information technology and the latest information systems have changed the way the public views application development or information systems from the point of view of developing desktop-based applications to web-based. In web application security, if security is not implemented in a web application, the web will lose data integrity and consumer trust in the company. Many web application owners ignore the security system on the website. Many crackers are not responsible and can damage files and look for security holes in these web applications. To secure these web applications, we can use the WAF (Web Application Firewall) method. However, the availability of web application services must be a concern because high availability will undoubtedly make the application or information system reliable. For this, the information system manager must think about increasing the high level of availability of the application or information system. By implementing a web application cluster system, can handle system failure issues.

Keywords: *Web Application, Firewall, High Available, Cluster*

Abstrak

Kemajuan dalam bidang teknologi informasi dan sistem informasi terkini telah mengubah cara pandang masyarakat mengenai pengembangan aplikasi atau sistem informasi dari cara pandang pengembangan aplikasi berbasis desktop menjadi berbasis web. Dalam keamanan aplikasi web jika tidak diterapkan suatu kemanan dalam suatu aplikasi web maka web tersebut akan rentan dalam kehilangan integritas data dan kepercayaan konsumen dalam perusahaan tersebut dan banyak pemilik aplikasi web mengabaikan security sistem pada website tersebut. Pada saat ini masih banyak cracker yang tidak bertanggung jawab dan dapat merusak file serta mencari celah keamanan pada aplikasi web tersebut, untuk mengamankan aplikasi web tersebut kita bisa menggunakan metode *WAF*(*Web Application Firewall*). Namun demikian ketersedian layanan aplikasi web harus menjadi perhatian, karena ketersediaan (*availability*) yang tinggi tentu akan menjadikan aplikasi atau sistem informasi dapat diandalkan. Untuk hal tersebut maka pengelola sistem informasi harus memikirkan bagaimana cara untuk meningkatkan tingkat ketersediaan yang tinggi dari aplikasi atau sistem informasi tersebut. Dengan menerapkan sistem *cluster* aplikasi web, persoalan kegagalan sistem dapat ditangani.

Kata kunci: *Web Application, Firewall, High Available, Cluster*

1. PENDAHULUAN

Kemajuan dalam bidang teknologi informasi dan sistem informasi terkini telah mengubah cara pandang masyarakat mengenai pengembangan aplikasi atau sistem informasi dari cara pandang pengembangan aplikasi berbasis desktop menjadi berbasis web. Web tidak hanya untuk menampilkan dan memberikan informasi tentang perusahaan atau organisasi. Web menjadi lingkungan dan platform berjalannya aplikasi atau sistem informasi yang cukup beragam, mulai dari sistem manajemen konten, atau informasi artikel, aplikasi atau informasi tutorial, aplikasi atau informasi forum jual beli dan lain sebagainya.

Peran penting dari aplikasi atau sistem informasi adalah mengubah bentuk bisnis proses ketentuan-ketentuan yang telah disepakati menjadi terkomputerisasi yang memberikan banyak keunggulan.

Diterima 2 Juli 2017, Direvisi 18 Agustus 2017, Diterima untuk publikasi 12 September 2017

Dalam keamanan aplikasi web jika tidak diterapkan kemanan dalam suatu aplikasi web maka web tersebut akan rentan dalam kehilangan integritas data dan kepercayaan konsumen dalam perusahaan tersebut dan banyak pemilik aplikasi web mengabaikan *security sistem* pada website tersebut. Pada saat ini masih banyak *cracker* yang tidak bertanggung jawab dan dapat merusak file serta mencari celah keamanan pada aplikasi web tersebut, untuk mengamankan aplikasi web tersebut kita bisa menggunakan metode WAF *(Web Application Firewall)* dengan *proxy Nginx* dan pengamanan menggunakan *modsecurity.*

Namun demikian ketersedian layanan aplikasi web harus menjadi perhatian, karena ketersediaan *(availability)* yang tinggi tentu akan menjadikan aplikasi atau sistem informasi dapat diandalkan . Untuk hal tersebut maka pengelola sistem informasi harus memikirkan bagaimana cara untuk meningkatkan tingkat ketersediaan yang tinggi dan keandalan dari aplikasi atau sistem informasi tersebut. Salah satu cara adalah dengan menerapkan sistem *cluster* aplikasi web, sehingga diharapkan persoalan kegagalan sistem dapat ditangani dengan adanya *cluster* aplikasi web.

Dalam beberapa kasus, aplikasi web atau sistem informasi berbasis web, dipasang pada *server* lokal, atau berjalan dalam lingkungan jaringan *private*. Dengan keadaan perangkat yang harus *standby* setiap hari maka seseorang tidaklah bisa untuk berbuat apa-apa jika sewaktu waktu *server* mengalami masalah seperti *server down* atau mati. maka kami akan mengimplementasikan suatu perangkat *clustering server* yang akan mengantisipasi jika *server* pusat mengalami masalah atau *down* maka *server clustering* yang langsung mengambil alih pusat data dan memberikan kesempatan untuk memperbaiki *server* pusat.

Berdasarkan latar belakang di atas yang tertera, maka penelitian yang akan diambil yaitu "Rancangan dan Implementasi *High Available web application security* berbasis *Nginx* dan *Modsecurity*" sebagai pembahasan utama.

1.1 Rumusan Masalah

Rumusan masalah yang diambil dalam penelitian ini adalah sebagai berikut:

1. Bagaimana Rancangan dan Implementasi *High Available Web Application Firewal*l berbasis *Nginx* dan *Modsecurity*?
2. Bagaimana efektifitas dari rancangan dan implementasi *Web Application Firewall* berbasis *Nginx* dan *Modsecurity*?
3. Bagaimana efektifitas sistem *High Availability* yang dirancang dan diterapkan?

1.2 Tujuan

Tujuan penelitian yang ingin dicapai dalam penelitian ini, antara lain:

1. Merancang dan menerapkan *High Available Web Application Firewall* berbasis *Nginx* dan *Modsecurity.*
2. Meningkatkan keamanan dari aplikasi web dengan menerapkan *Application Firewall.*
3. Meningkatkan ketersediaan *Web Application Firewall.*
4. Mengetahui efektifitas dari Rancangan dan Implementasi *High Available Web Application Firewall* berbasis *Nginx* dan *Modsecurity.*

1.3 Batasan Masalah

Batasan masalah yang diambil dalam penelitian ini adalah sebagai berikut:

1. *Platform* yang berkaitan dengan bagian *server backend* (*upstream server*) dalam penelitian ini dibatasi pada *platform* sistem linux dengan aplikasi *Web Server Apache* dan *Nginx.*
2. Pengujian hanya meliputi serangan *SQL injection, Brute Force, XSS.*
3. Pengujian *reverse proxy* hanya menggunakan *Nginx.*
4. Lingkungan penelitian hanya dilakukan dalam lingkungan jaringan komputer pengembangan (*development*) bukan *production.*

2. LANDASAN TEORI

2.1 *Web Application*

Aplikasi web merupakan aplikasi yang diakses mengunakan web browser melalui jaringan internet atau intranet. Aplikasi web juga merupakan suatu perangkat lunak komputer yang mendukung perangkat lunak berbasis web *seperti JavaScript, Ruby, Python, Php, Java* dan bahasa pemrograman lainnya.

2.2 *Web Application Firewall*

Web Application Firewall (WAF) adalah suatu proses untuk mengamankan suatu web dengan cara, memonitor, dan memblokir lalu lintas HTTP dan dari aplikasi web. WAF berbeda dengan *firewall* biasa karena WAF dapat memfilter konten aplikasi web tertentu sementara *firewall* biasa berfungsi sebagai gerbang keamanan antar *server*. Dengan memeriksa lalu lintas HTTP, itu dapat mencegah serangan yang berasal dari kelemahan keamanan aplikasi web.

2.3 *Web Attack*

Web Attack merupakan sebuah aktivitas yang dilakukan oleh seseorang dengan memanfaatkan sebuah celah atau sisi kerentanan pada aplikasi web maupun server dan ada 3 jenis serangan yang akan dijelaskan sebagai berikut:

2.3.1 *SQL Injection*

SQL Injection adalah sebuah teknik *hacking* untuk mendapatkan akses pada sistem *database* yang berbasis SQL. SQL merupakan singkatan dari *Structured Query Language* yaitu bahasa yang digunakan untuk membuat serta mengolah *database*. Dalam melakukan teknik SQL

injection, para *hacker* akan memanfaatkan celah keamanan pada web atau aplikasi.

2.3.2 *Brute Force*

Serangan *brute force* adalah sebuah teknik serangan terhadap sebuah sistem keamanan komputer yang menggunakan percobaan terhadap semua kunci yang mungkin. Pendekatan ini pada awalnya merujuk pada sebuah program komputer yang mengandalkan kekuatan pemrosesan komputer dibandingkan kecerdasan manusia.

2.3.3 *XSS (Cross Side Scripting)*

XSS adalah salah satu jenis serangan injeksi *code (code injection attack)*. XSS dilakukan oleh penyerang dengan cara memasukkan kode HTML atau *client script code* lainnya ke suatu situs. Serangan ini akan seolah-olah datang dari situs tersebut. Akibat serangan ini antara lain penyerang dapat mem-bypass keamanan di sisi klien, mendapatkan informasi sensitif, atau menyimpan aplikasi berbahaya.

2.4 *Web Server*

Web Server merupakan sebuah perangkat lunak dalam *server* yang berfungsi menerima permintaan (*request*) berupa halaman web melalui HTTP atau HTTPS dari klien yang dikenal dengan browser web dan mengirimkan kembali (*response*) hasilnya dalam bentuk halaman-halaman web yang umumnya berbentuk dokumen HTML salah satu web *server* yang paling dikenal yaitu bisa dilihat di bawah ini.

2.4.1 *Nginx*

Nginx adalah *server* HTTP gratis, yang berbasis *open-source*, berkinerja tinggi, dan *reverse proxy*, serta *server* proxy IMAP / POP3. NGINX dikenal karena kinerjanya yang tinggi, stabilitas, set fitur yang kaya, konfigurasi yang sederhana, dan konsumsi sumber daya yang rendah, dan di dalam *Nginx* terdapat *module* yang bernama *Modsecurity*.

2.4.2 *ModSecurity*

Modsecurity adalah pemantauan aplikasi web *real-time*, *logging*, dan kontrol akses , dan mesin *firewall* aplikasi web yang memberikan perlindungan yang sangat kecil. Agar menjadi berguna, *Modsecurity* harus dikonfigurasi dengan aturan. Tidak seperti deteksi intrusi dan sistem pencegahan, yang bergantung pada tanda tangan khusus untuk kerentanan yang diketahui.

2.5 *High Available*

High Avaibility Web Server adalah ketersediaan data informasi dari web *server* secara terus menerus untuk melayani dan memenuhi kebutuhan pengguna sistem tersebut. Tujuan dari *High Available* yaitu sebuah sistem dengan *High Availability Web Server* yang dapat memberikan suatu kenyamanan bagi pengguna atau penyedia layanan jika sewaktu-waktu terjadi gangguan pada sistem, penerapannya kita bisa menggunakan teknik *failover clustering*.

2.5.1 *Pacemaker*

Pacemaker adalah aplikasi yang berfungsi untuk menangani *failover*, yaitu pengecekan layanan yang berjalan di *server* dan kemudian memindahkan fungsi utama yang menyediakan layanan pada *client* ke *server* yang lainnya jika *server* utama mengalami kerusakan.

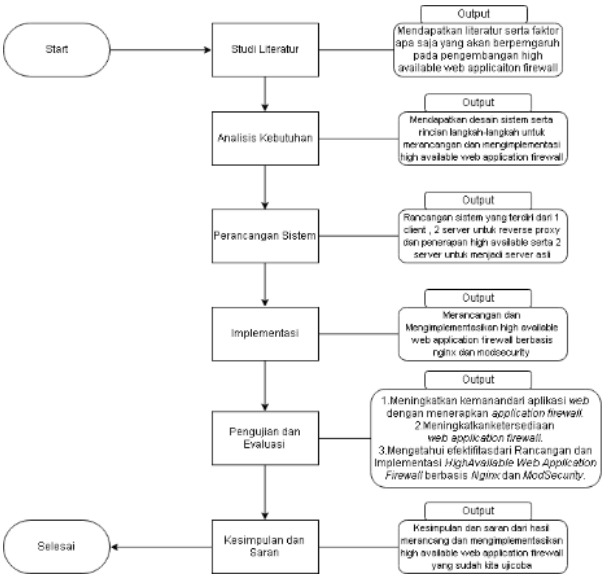
2.5.2 *Corosync*

Corosync merupakan bagian dari *pacemaker*, mirip seperti *heartbeat*. Fungsinya adalah sebagai *loadbalancer* dan *checker* pada *cluster environment*. *OpenSUSE 12.3* tidak memuat perangkat lunak ini pada repositori standar. Tapi bukan berarti tidak ada paket *RPM* yang bisa dengan mudah anda install. *OpenSUSE* menyediakan paket ini pada *HA:Clustering repository*. *Corosync* bertujuan untuk terus memeriksa *server* dalam konfigurasi *cluster* komputer untuk mengetahui keadaan sebenarnya dari server lainnya yang berada di dalam *cluster* yang sama.

3. METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Tahapan penelitian adalah proses penelitian yang terdiri dari tahapan studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan evaluasi, serta yang terakhir adalah kesimpulan dan saran. Berikut pada gambar tersebut adalah tahapan-tahapan penelitian:



Gambar 1. Tahapan Penelitian

1. Studi Literatur

Pada tahapan awal ini dilakukan dengan mencari, mengumpulkan, serta membaca artikel di website dan beberapa skripsi penelitian lainnya yang berhubungan

dengan rancangan dan implementasi *High Available web application security* berbasis *Nginx* dan *modsecurity*. Hasil dari studi literatur yaitu pembuatan design rancangan penelitian dan rujukan bagaimana penelitian harus dilakukan dan bahan apa saja yang diperlukan untuk tujuan penelitian ini agar dapat tercapai. Analisis yang dilakukan juga berpacu kepada studi literatur yang relevan dengan tema penelitian ini.

2. Analisis dan Kebutuhan

Pada tahapan ini dilakukan untuk menganalisis data apa saja yang di perlukan untuk perancangan dan pengimplementasian *High Available Web Application Firewall* sudah optimal atau belum dalam menangani serangan seperti *rules* yang ada di *OWASP*. Dengan mengetahui faktor-faktor yang mempengaruhi *High Available Web Application Firewall*, maka dapat dilakukan analisis kebutuhan sistem dan batasan masalah dari penelitian yang akan dibuat.

3. Perancangan Sistem

Pada tahapan ini akan dilakukan perancangan sistem yang akan dibuat dan diujikan, tentang rancangan dan implementasi *High Available Web Application Firewall.* Sistem operasi yang akan digunakan adalah Linux *Ubuntu* 16.04 LTS.

4. Implementasi

Pada tahapan ini, setelah dilakukan analisis kebutuhan dan perancangan sistem maka masuk ke fase imlementasi, dimana pada fase implementasi ini akan dilakukan proses konfigurasi dari rancangan dan implementasi *High Available Web Application Firewall* yang telah dibuat sehingga siap untuk dilakukan pengujian *High Available Web Application Firewall*. Melakukan implementasi dan analisis kinerja *High Available Web Application Firewall*, menggunakan sistem operasi *Ubuntu* 16.04, lalu instalasi dan konfigurasi berbagai perangkat lunak dan tools yang dibutuhkan seperti *Nginx*, *Modsecurity*, *Corosync* dan *Pacemaker*.

5. Pengujian dan Evaluasi

Pada tahapan ini akan dilakukan pengujian terhadap faktor-faktor kinerja *High Available Web Application Firewall* yang akan kita serang dengan menggunakan *SQL Injection, XSS , Brute Force*, serta jika kita mematikan proxy *server* akankah *server High Available* bekerja. Setelah menguji beberapa faktor maka peneliti akan menganalisis hasil yang di dapat dari pengujian yang telah dilakukan.

6. Kesimpulan dan Saran

Pada bagian sub bab ini berisi tentang kesimpulan yang diambil dari keseluruhan proses yang dilakukan dari penelitian ini, serta saran yang akan diberikan untuk menjadi masukan bagi pengembangan lebih lanjut.

4. ANALISIS DAN PERANCANGAN

4.1 Analisis Kebutuhan

Pada tahap ini akan dilakukan analisis terhadap kebutuhan-kebutuhan perangkat lunak dan perangkat keras sebagai berikut.

4.1.1 Analisis Kebutuhan *Software*

Peneliti menggunakan perangkat lunak berdasarkan kebutuhan yang umum digunakan dari studi literatur yang berkaitan dengan penerapan *High Available Web Application Firewall*. Perangkat lunak yang akan dibutuhkan adalah sebagai berikut:

- *Linux Ubuntu 16.04*
- *Virtual Box 5.2*
- *Nginx 1.13.4*
- *Modsecurity 2.9.3*
- *Corosync 2.4.5*
- *Pacemaker 2.0.2*

4.1.2 Analisis Kebutuhan *Hardware*

Pada tahapan ini dilakukan untuk menganalisis data apa saja yang diperlukan untuk perancangan dan pengimplementasian *High Available Web Application Firewall*. Dimana penggunaan spesifikasi *hardware* yang ditentukan berdasarkan pendekatan atau studi literatur yang peneliti dapatkan. Peneliti menggunakan spesifikasi *hardware* yang ada sekarang ini:

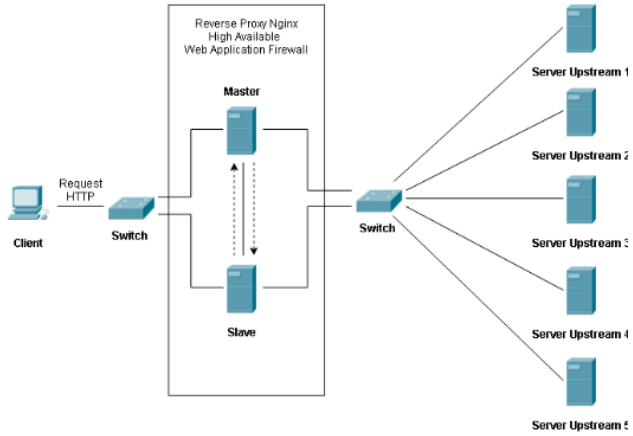
- Prosesor : Intel® Core™ i5-2450M CPU @ 2.50GHz (4 CPUs), ~2.5GHz
- RAM : 10GB
- Hardisk : 650GB

4.2 Perancangan Sistem

Dalam bagian perancangan sistem peneliti akan melakukan perancangan sistem yang terdiri dari perancangan arsitektur sistem dan rancangan pengujian.

4.2.1 Perancangan Arsitektur Sistem Fisik

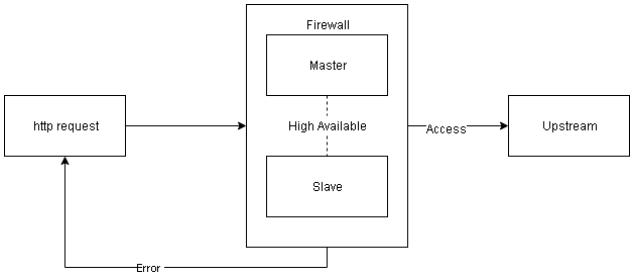
Dalam penelitian ini peneliti melakukan perancangan arsitektur sistem yang terdiri dari 1 *client* 1 *switch* untuk menghubungkan ke *master* dan *slave*, 1 *master server* untuk dipasangkan *firewall* dan *reverse proxy* , 1 *slave server* untuk *backup* dari *primary* dengan menggunakan metode *High Available*, 1 *switch* dan di belakang *switch* ada beberapa *Upstream server*. Pada gambar tersebut adalah rancangan arsitektur sistem yang peneliti rancangan yang akan digunakan untuk pengujian dan analisis.



Gambar 2. Rancangan Arsitektur Sistem Fisik

4.2.2 Perancangan Arsitektur Sistem *Logic*

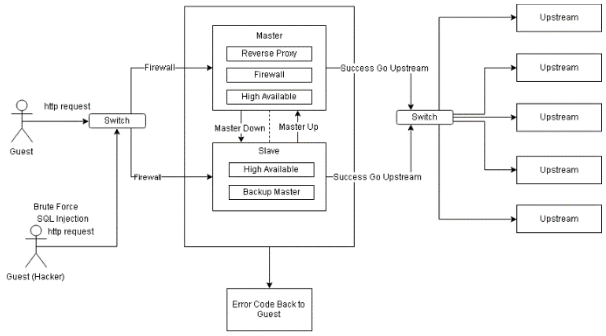
Secara konseptual, proses pengujian keamanan aplikasi dan dokumen-dokumen yang dihasilkan, serta keterkaitan siklus pengujian keamanan sistem berbasis web application, dapat dilihat pada gambar berikut.



Gambar 3. Rancangan Arsitektur Sistem *Logic*

4.3 Rancangan Pengujian

Berikut adalah rancangan pengujian yang akan peneliti terapkan:



Gambar 4. Rancangan Pengujian

Alur pengujian yang ingin peneliti uji adalah tentang *High Available Web Application Firewall* , yang dimana akan diuji dari segi keamanan yang akan kita uji cobakan dengan serangan-serangan umum yang dilakukan oleh *hacker* dan dari tingkat ketersediannnya yang bagaimana kita menguji bagaimana jika server *primary* mati dan dialihkan ke server *secondary*, yang menggunakan *Nginx* sebagai web servernya, *Modsecurity* sebagai module keamanan dan *Corosync Pacemaker* sebagai High Avalability atau *cluster* untuk *server*.

5. IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

Pada implementasi ini peneliti akan melakukan dari sebuah perancangan yang sudah dibuat yaitu bebarapa tahap antara lain melakukan tahap instalasi dan konfigurasi pada *Nginx* dan *Modsecurity* serta membuat tingkat ketersediaan yang tinggi dengan mengkonfigurasi *corosync* dan *pacemaker* untuk *Web Application.*

5.1.1 Instalasi dan Konfigurasi

Pada tahap instalasi dan konfigurasi ini, peneliti melakukan tahapan pemasangan *firewall* sistem menggunakan *Nginx* dan *Modsecurity* dan sistem *High Available* dengan menggunakan *corosync* dan *pacemaker*. Tujuannya agar terciptanya sebuah *Web Application Firewall* dan *Web Application* yang tingkat ketersedian tinggi. Setelah melakukan instalasi tersebut, peneliti juga akan mengkonfigurasi terhadap sistem yang akan dirancang.

5.2 Pengujian

Dalam tahap pengujian peneliti akan membahas tahap pengujian yang berdasarkan rumusan masalah, tahap pengujian untuk keamanan atau firewall menggunakan *Nginx* sebagai web server dan *Modsecurity* sebagai *module firewall* , serta *reverse proxy*. Dan tahap pengujian yang akan dilakukan lagi yaitu tahap untuk *High Available* atau tingkat ketersediaan yang tinggi untuk sebuah aplikasi web.

5.2.1 Pengujian *Web Application Firewall*

Pada tahap pengujian *Web Application Firewall*, peneliti akan menguji 3 serangan yang termasuk kedalam top 10 OWASP, yaitu serangan berupa *Brute Force*, *SQL Injection*, dan *XSS* Reflected, peneliti melakukan serangan tersebut dengan tujuan untuk mengetahui bagaimana keamanan sistem yang telah diterapkan bekerja atau tidaknya sebagai berikut:

A. *SQL Injection*

Pada tahap pengujian *SQL Injection* , peneliti akan menguji beberapa Pola serangan dari *SQL Injection* yang berbeda-beda yang didasari pada tabel berikut Dalam melakukan suatu pengujian terhadap sistem membutuhkan scenario pengujian terlebih dahulu. Skenario pengujian memiliki tujuan menjadi dasar pengembangan bagi pengerjaan sistem selanjutnya:

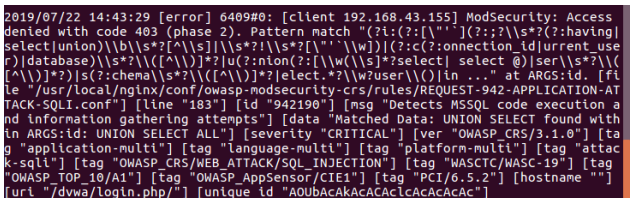
Tabel 1. Pola Serangan *SQL Injection*

Pola serangan	Http Response Code

http://websiteku.id/dvwa/?id=SELECT UNION SELECT ALL	403
http://websiteku.id/dvwa/?id=SELECT 1; #comment	403
http://websiteku.id/dvwa/?id=SELECT user();	403
http://websiteku.id/dvwa/?id=SELECT substr('abcd', 3, 1); # returns c	403
http://websiteku.id/dvwa/?id=CREATE USER test1 IDENTIFIED BY 'pass1'; — priv	403
http://websiteku.id/dvwa/?id=SELECT if(1=1,'foo','bar'); — returns 'foo'	403
http://websiteku.id/dvwa/?id=SELECT CASE WHEN (1=1) THEN 'A' ELSE 'B' END; # returns A	403
http://websiteku.id/dvwa/?id=SELECT 0×414243; # returns ABC	403
http://websiteku.id/dvwa/?id=SELECT @@hostname;	403
http://websiteku.id/dvwa/?id=DROP USER test1; — priv	403
http://websiteku.id/dvwa/?id=SELECT @@datadir;	403

Skenario pengujian yang dilakukan pada penelitian ini, yaitu dengan memasukkan *syntax database* pada URL *website* yang didasari pada tabel pola serangan dan alamat *website* yang digunakan adalah alamat *website dummy* (tidak ada).

Hasil didapatkan: Pada *website* akan dialihkan menjadi 403 forbidden dan akan terdeteksi pada log *ModSecurity* sebagai *SQL Injection* setelah 11 kali percobaan yang akan terlihat pada log berkut:



Gambar 5. Log Serangan *SQL Injection*

Pada gambar tersebut terlihat hasil log dari serangan SQL injection dengan status *Forbidden*.

B. *XSS (Cross Side Scripting)*

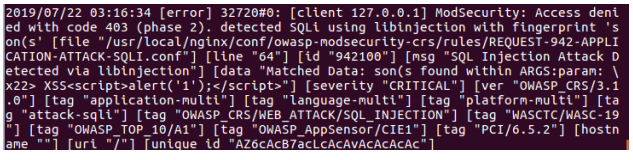
Pada tahap pengujian serangan *XSS*, peneliti akan menguji beberapa pola serangan dari *XSS* yang berbeda-beda yang didasari pada tabel berikut:

Tabel 2. Pola Serangan XSS

Pola serangan	Http Response Code
http://websiteku.id/dvwa/?p= "><script>alert(document.cookie)</script>	403
http://websiteku.id/dvwa/?p=user=<script>alert(123)</script>	403
http://websiteku.id/dvwa/?p=user=<script>window.onload = function() script>	403
http://websiteku.id/dvwa/?p=<IMG SRC="javascript:alert('*XSS* ');">	403
http://websiteku.id/dvwa/?p=<IMG SRC=javascript:alert("*XSS* ")>	403
http://websiteku.id/dvwa/?p=perl -e 'print "<IMG SRC=java\0script:alert(\"*XSS* \")>";' > out	403
http://websiteku.id/dvwa/?p=<iframe src=http://*XSS* .rocks/scriptlet.html <	403
http://websiteku.id/dvwa/?p=<BODY onload!#$%&()*~+-_.,:;?@[/\]^`=alert("*XSS* ")>	403
http://websiteku.id/dvwa/?p=\";alert('*XSS*');//	403
http://websiteku.id/dvwa/?p=</TITLE><SCRIPT>alert("*XSS* ");</SCRIPT>	403
http://websiteku.id/dvwa/?p=<BODY BACKGROUND="javascript:alert('*XSS* ')">	403

Skenario pengujian yang dilakukan pada penelitian ini, yaitu dengan memasukkan syntax *client script code* yang didasari pada tabel pola serangan dan alamat website yang digunakan adalah alamat website *dummy* (tidak ada):

Hasil didapatkan: Pada website akan dialihkan menjadi 403 *forbidden* dan akan terdeteksi pada log *ModSecurity* sebagai XSS setelah 11 kali percobaan yang akan terlihat pada log berikut:



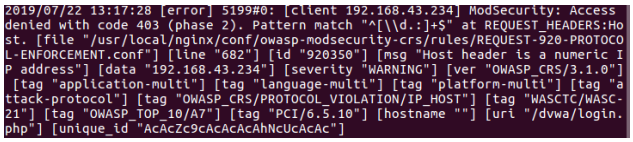
Gambar 6. Log Serangan XSS

Pada gambar tersebut terlihat hasil log dari serangan XSS dengan status *Forbidden*.

C. *Brute Force*

Pada tahap penguijan serangan Brute Force, peneliti akan menguji menyerang dengan menggunakan tools hydra dan metode Brute Force *password*.

Log dari serangan *Brute Force password*, setelah 15 kali percobaan log yang dihasilkan sebagai berikut:



Gambar 7. Log Serangan Brute Force

Pada gambar tersebut terlihat hasil log dari serangan Brute Force dengan status *Forbidden.*

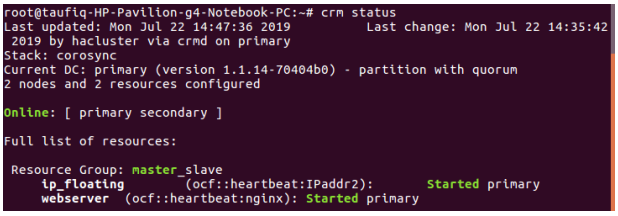
5.2.2 Pengujian *High Available Web Application*

Pada tahap pengujian *High Available Web Application*, peneliti akan menguji menguji dengan cara mematikan dan menyalakan server *primary* dan *secondary*, serta menguji melakukan perpindahan IP antara *primary* dan *secondary* dengan 4 kali pengujian yang terdapat pada tabel berikut:

Tabel 3. Pengujian pada *Server Primary* dan *Secondary*

Primary	***Secondary***	**Response Code**	**Status Server**
Online	*Online*	200	*Available*
Online	*Offline*	200	*Available*
Offline	*Online*	200	*Available*
Offline	*Offline*	-	*Not Available*

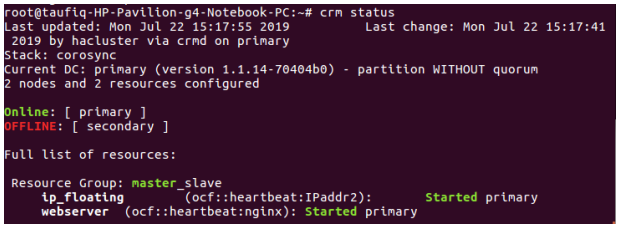
A. Pengujian pertama peneliti akan menguji menyalakan kedua server *primary* dan secondary, yang dimana jika keduanya menyala akan dihandle oleh *primary* server, status bisa diperiksa dengan cara:



Gambar 8. Status *Cluster* ketika *Primary dan Secondary Online*

Pada gambar tersebut terlihat yang menghandle *server* utama adalah *primary server*.

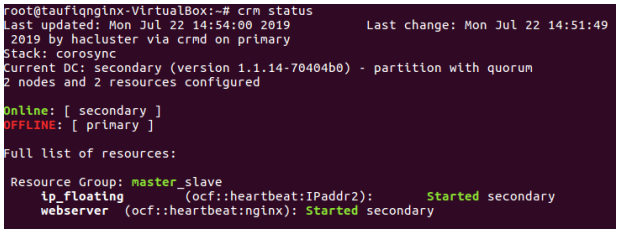
B. Pengujian kedua, peneliti akan menguji menyalakan hanya *server primary* dan *server secondary* dengan keadaan *offline*:



Gambar 9. Status *Cluster* ketika *Primary Online Secondary Offline*

Pada gambar tersebut terlihat yang menghandle *server* utama adalah *primary server*.

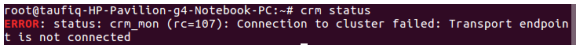
C. Pengujian ketiga, peneliti akan menguji menyalakan hanya *server secondary* dan *server primary* dengan keadaan *offline*.



Gambar 10. Status *Cluster* ketika *primary offline secondary online*

Pada gambar tersebut terlihat yang menghandle *server* utama adalah *primary secondary server*.

D. Pengujian keempat, peneliti akan menguji mematikan kedua *server secondary* dan *server primary* dengan keadaan *offline*.



Gambar 11. Status *Cluster* ketika *Primary* dan *Secondary Offline*

Pada gambar tersebut, status dari cluster server tidak ada response dikarenakan kedua server dalam keadaaan mati (*offline*).

6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

Kesimpulan yang dapat peneliti ambil dalam penelitian terkait pembuatan Rancangan dan Implementasi *High Available Web Application Firewall* ini adalah sebagai berikut:

1. *Web Application Firewall* (WAF) berbasis *Nginx* dan *Modsecurity* berhasil dirancang dan diimplementasikan oleh peneliti. Rancangan yang dibuat oleh peneliti yaitu berdasarkan rancangan fisik dan logic dengan menggunakan *server primary*, *server secondary*, dan *server upstream*. Dengan mengintegrasikan *reverse proxy Nginx* dengan modul *Modsecurity* sehingga *Nginx* dapat melakukan peran sebagai WAF yang dapat melindungi *upstream server* atau aplikasi web yang berada di balik (belakang) *reverse proxy Nginx* dari berbagai serangan yang telah dikenal (diketahui) oleh modul *Modsecurity* terhadap

request yang dikirimkan oleh *client* serta dengan tingkat ketersediaan yang tinggi dengan menerapkan *cluster server* menggunakan *Corosync* dan *Pacemaker*.

2. Berdasarkan hasil perancangan, penerapan, dan pengujian yang sudah dilakukan oleh peneliti, proses rancangan dan implementasi *Web Application Firewall* berbasis *Nginx* dan *Modsecurity*, yang memanfaatkan *Nginx* dengan fitur module *Modsecurity* menunjukan hasil yang efektif. Hal ini, dibuktikan dari hasil pengujian sebanyak 11 pola serangan SQL *injection* , 11 pola serangan *Cross Site Scripting* (XSS), dan 14 kali percobaan serangan *Brute Force* yang mana seluruhnya berhasil dideteksi dan dicegah oleh WAF berbasis *Nginx* dan *Modsecurity*.
3. Berdasarkan hasil perancangan, penerapan, dan pengujian yang sudah dilakukan oleh peneliti, proses sistem high *availability*, yang memanfaatkan fitur *failover clustering* yang ada pada *Corosync* dan *pacemaker* menunjukan hasil yang efektif. Hal ini dibuktikan setelah 15 kali percobaan *server primary* dan *secondary* dijalankan dengan konfigurasi aktif/ pasif. *Server primary* selalu mengambil IP publik, setiap kali *server primary offline*, maka *server secondary* yang akan mengambil IP publik dan menjadi server utama.

6.2 Saran

Berdasarkan penelitian yang telah dilakukan, terdapat saran yang dapat dilakukan untuk penelitian selanjutnya, yaitu:

1. Penelitian berikutnya, dapat melakukan pengujian lebih banyak pola serangan yang terdapat pada OWASP agar pengujian dari kemananan *website* dapat dilakukan dengan lebih optimal.
2. Diberikannya suatu notifikasi seperti menggunakan media *social*, *SMS*, atau *E-Mail* saat terjadinya penyerangan *application web* sehingga meminimalisir memeriksa *log* melalui *server* dikarenakan pada saat memeriksa *log* secara *real-time* dapat memakan banyak *memory*.
3. *Cluster server* dapat dikembangkan dan diimplementasikan pada lingkungan yang sesungguhnya.

DAFTAR PUSTAKA

[1] R. Y. J. dkk, “Implementasi Keamanan Aplikasi Web,” Bandung: Telkom University Bandung, 2015.

[2] M. Hayati, “Implementasi Web Server Nginx pada Sistem Monitoring Rumah Menggunakan Sistem Operasi Raspbian Jessie,” Padang: Politeknik Negeri Padang, 2016.

[3] R. Hidayat, “Cara Praktis Membangun Website Gratis,” Jakarta: Elex Media Komputindo, 2010.

[4] D. . R. H. d. Laitupa, “Implementasi *Modsecurity* sebagai Sistem Monitoring Keamanan Aplikasi Web secara *Real Time*,” Bandung: Telkom University Bandung, 2015.

[5] J. Purnomo, “Implementasi dan Analisis *High availability Server* dengan Teknik *Failover Clustering* Menggunakan *Heartbeat*,” Salatiga: Universitas Kristen Satya Wacana, Salatiga, 2017

[6] F. K. dkk, “Implementasi *Server Cluster High Availability* pada Web Server,” Bekasi: Universitas Islam “45”, Bekasi, 2016.

[7] M. I. Hasan, “Pokok-Pokok Materi Metodologi Penelitian dan Aplikasinya,” Jakarta: Penerbit Ghalia Indonesia, Jakarta, 2002.