WAF: Uma análise de desempenho e eficácia

Everton dos Santos Silva, Lucas Azevedo da Silva, Matheus J. da Paixão Frez, Felipe M. H. de Oliveira Malara, Nilson Mori Lazarin

[1]Bacharelado em Sistemas de Informação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ) – Nova Friburgo, RJ – Brazil

nilson.lazarin@cefet-rj.br

Abstract. *This article discusses the performance, use of resources and the use of some open source Web Application Firewall (WAF). The performance will be analyzed for each of the three proposed tools. The purpose of this article is for the reader to be able to understand what a WAF is, what its purpose, and with our results, be capable of choosing which WAF is suitable to his web server. In the end, we made a conclusion on the overall efficiency of the proposed WAFs.*

Resumo. *Este artigo discute o desempenho, consumo de recursos e a utilização de alguns Web Application Firewall (WAF) open source. Será analisado o desempenho para cada uma das três ferramentas propostas. O objetivo deste artigo é que o leitor seja capaz de compreender o que é um WAF, para que serve o mesmo, e que seja capaz de escolher com base em nossos resultados qual WAF se adequa melhor ao seu servidor web. No final tiramos algumas conclusões sobre a eficácia geral dos Wafs propostos.*

1. Introdução

Atualmente cerca de 25% das aplicações web tem pelo menos uma falha de segurança de alta severidade, sendo a XSS a mais comum e cerca de 28% dos aplicativos da web não tem nenhuma proteção de força bruta em suas páginas de login [Acunetix 2020]. Visando proteger as aplicações web, os *Web Application Firewall* (WAF) monitoram, filtram e bloqueiam pacotes de dados conforme eles trafegam em site ou aplicativo da web, contribuindo para uma segurança das aplicações e servidores.

Muitos invasores executam scanners para identificar vulnerabilidades de segurança em um site ou aplicativo para posteriormente lançarem os ataques apropriados. Nos experimentos foram utilizados três ferramentas: Nikto[1] um scanner de vulnerabilidades catalogadas no OSVDB (*Open Sourced Vulnerability Database*); OWASP Zed Attack Proxy[2] um scanner de código aberto desenvolvido pela OWASP, do tipo *proxy man-in-the-middle*, atuando entre o navegador do testador e o aplicativo da web alvo, capaz de interceptar, alterar e encaminhar pacotes para o destino; e WPScan[3]um scanner de caixa preta para vulnerabilidades específicas da plataforma WordPress [Broad and Bindner 2014].

Uma possível barreira de proteção contra invasores é a utilização de um WAF que aplica um conjunto de regras a uma comunicação HTTP e pode ser implantado para proteger um aplicativo da web específico ou conjunto de aplicativos da web, sendo considerado

[1]https://cirt.net/Nikto2
[2]https://www.zaproxy.org/
[3]https://wpscan.com/

um proxy reverso [Memon et al. 2018] [OWASP 2020]. Este trabalho apresenta um comparativo entre três WAF. Foram considerados diversos aspectos, dentre eles vale destacar: uso de CPU, uso de memória RAM e escrita de disco durante um ataque. Além disso, comparamos as vulnerabilidades encontradas pelos scanners e a severidade das mesmas contra cada WAF.

2. Metodologia e resultados

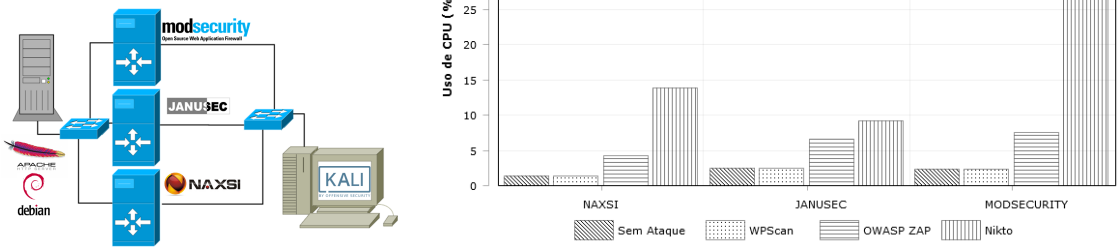
Neste trabalho foram realizadas duas análises: Uma análise de desempenho onde foram considerados os indicadores de uso de CPU, consumo de memória RAM e escrita em disco, mensurando o consumo em estado normal e sob ataque; E uma análise de eficácia que considerou as descobertas dos scanners de vulnerabilidade apresentados anteriormente.

A escolha dos WAFs a serem analisados foi realizada através das seguintes etapas: 1ª) Foram listados todos os repositórios públicos identificados com *#waf* no GitHub[4], 233 repositórios foram encontrados; 2ª) Selecionamos apenas as ferramentas tipo WAF, ignorando *plugins* e outros acessórios, 46 repositórios restaram; 3ª) Escolhemos apenas WAF que são tipo *Standalone*, ou seja, que são autossuficientes e rodam como serviço, 27 repositórios foram selecionados; 4ª) Selecionamos apenas repositórios com mais de 500 estrelas (marcado como favorito por outro membro da comunidade) e que possuíam atualização recente de pelo menos um ano, restando os repositórios listados na Tabela 1.

Repositório	Licenciamento	Estrelas	Contribuidores	Versões	Última versão
https://github.com/SpiderLabs/ModSecurity	Apache 2.0	4028	59	90	13/01/2020
https://github.com/nbs-system/naxsi	GPL-3.0	3645	39	44	17/11/2020
https://github.com/Janusec/janusec	GNU AGPLv3	744	2	3	21/12/2020

Tabela 1. *Web Application Firewall* analisados

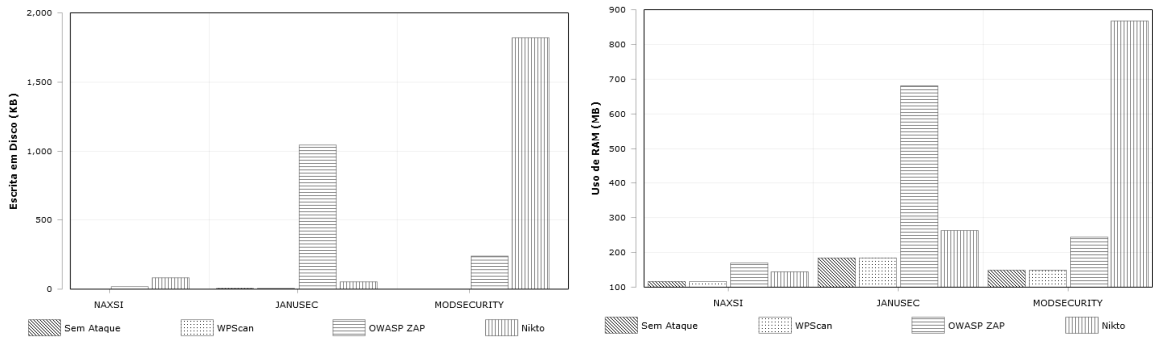
Os ataques foram realizados utilizando o Kali Linux, contra três WAF Standalone que protegem o servidor de hospedagem, um Debian 10 com HTTP Apache Server. Todos os membros do ambiente de testes, apresentado na Figura 1a, são máquinas virtuais com a configuração: CPU de 2 núcleos, 1 GB de RAM, 32 GB de HD.



(a) Ambiente **(b)** Uso de CPU (%)

Figura 1. Ambiente de Testes e Consumo de CPU.

[4]https://github.com/topics/waf



(a) uso Disco (KB) **(b)** uso RAM (MB)

Figura 2. Consumo de I/O e RAM.

O resultado dos testes de desempenho entre os WAF analisados, em situação normal e sob ataque dos scanners são apresentados em: Figura 1b, percentual de uso de CPU; Figura 2b, consumo em MB da memória RAM; e Figura 2a, escrita no disco em KB.

Durante a análise de eficácia sob ataque do WPScan nenhum plugin foi identificado, foram reportados que o XML-RPC e WP-Cron estão habilitados, e o diretório de uploads está com a listagem habilitada, além dos seguintes resultados: Contra o NAXSI foi possível identificar o readme do Wordpress e a versão utilizada (5.4.4). Foi obtido também a versão e o servidor utilizado no WAF (NGINX); Contra o ModSecurity foi possível identificar a versão e o servidor do WAF (NGINX), o readme e a versão Wordpress, como também o tema da aplicação (twenty-twenty); Contra o Janusec foi obtido o servidor da aplicação (Apache), o readme e a versão do Wordpress.

Os resultados da análise de eficácia sob ataque do OWASP ZAP são apresentados na Tabela 2. Contra o NAXSI foram informados 14 tipos de alerta. Contra o Janusec foram informados 13 tipos de alerta. Contra o ModSecurity foram informados 14 tipos de alerta.

ID	Alertas (OWASP ZAP)	Risco	JANUSEC	ModSecurity	NAXSI
OR01	Application Error Disclosure	Médio	15x	6x	3x
OR02	Directory Browsing	Médio	44x	35x	50x
OR03	X-Frame-Options Header Not Set	Médio	80x	76x	60x
OR04	Absence of Anti-CSRF Tokens	Baixo	3x	18x	3x
OR05	Application Error Disclosure	Baixo	X	X	X
OR06	Content-Type Header Missing	Baixo	-	X	X
OR07	Cookie No HttpOnly Flag	Baixo	X	2x	X
OR08	Cookie Without SameSite Attribute	Baixo	X	2x	X
OR09	Cross-Domain javascript source file inclusion	Baixo	5x	3x	5x
OR10	Private IP Disclosure	Baixo	2x	X	2x
OR11	Web Browser XSS Protection Not Enabled	Baixo	88x	87x	82x
OR12	X-Content-Type-Options Header Missing	Baixo	266x	583x	303x
OR13	Information Disclosure – Suspicious Comments	Informativo	28x	36x	28x
OR14	Timestamp Disclosure – Unix	Informativo	36861x	110823x	45745x

Tabela 2. Vulnerabilidades e Alertas OWASP ZAP

Os resultados da análise de eficácia sob ataque do Nikto, são apresentados na Tabela 3. Contra o NAXSI foram reportados 17 alertas. Contra o Janusec foram reportados

8 alertas. Contra o ModSecurity foram reportados 3 alertas.

ID	Vulnerabilidades (Nikto)	Risco	JANUSEC	ModSecurity	NAXSI
NR01	The anti-clickjacking X-Frame-Options header is not present	Baixo	X	X	X
NR02	The X-XSS-Protection header is not defined	Baixo	X	X	X
NR03	The X-Content-Type-Options headers is not set	Baixo	X	X	X
NR04	No CGI directories found	Informativo	X		X
NR05	OSVDB 119 - Web Publisher should be disabled	Médio	X		X
NR06	OSVDB 576 - Weblogic allows source code or directory listing	Baixo			X
NR07	OSVDB 3233 - Apache default file found	Baixo	X		X
NR08	OSVDB 3268 - Directory indexing found	Baixo	X		X
NR09	OSVDB 3288 - Abyss reveals directory listing when are requested	Médio			X

Tabela 3. Vulnerabilidades e Alertas Nikto

3. Conclusão

Tendo em vista o exposto, podemos perceber que, embora cada WAF tenha sua lógica para lidar com os ataques, em geral, nenhum apresentou vulnerabilidades ou alertas de alta severidade.

O ModSecurity foi o WAF com maior destaque pois, mesmo tendo consumido mais recursos sob ataque, foi o mais eficaz no resguardo das informações, deixando os scanners Nikto e ZAP encontrarem 12 alertas de baixo risco, 3 de médio e 2 informativos, entretanto, foi o único a deixar o WPScan descobrir o tema do WordPress. Em seguida temos o Janusec que teve um uso moderado de recursos durantes os ataques a mesma medida que foi capaz de proteger as informações com 13 alertas de baixo risco, 4 de médio e 3 informativos relatados pelo Nikto e ZAP, além de deixar o WPScan identificar o servidor HTTP. Por último temos o NAXSI que embora tenha sido o WAF que menos utilizou recursos, foi o que mais teve dificuldades na proteção dos dados e teve a maior quantidade de alertas reportados no geral e 15 alertas de baixo risco, 5 de médio e 3 informativos, além de deixar o WPScan identificar a versão do WordPress.

Contudo, para melhor aproveitamento das ferramentas apresentadas é necessário um conhecimento prévio sobre segurança de redes, já que para aplicar as configurações completas são envolvidos alguns conhecimentos avançados de redes. Tendo em vista essa limitação, em trabalhos futuros poderia ser analisado o desempenho dos WAFs de acordo com suas configurações mais avançadas, analisando as melhores configurações para cada tipo de aplicação na web.

Referências

Acunetix (2020). Acunetix Web Application Vulnerability Report 2020. Disponível em: https://www.acunetix.com/white-papers/acunetix-web-application-vulnerability-report-2020/ Acessado em: 15/01/2021.

Broad, J. and Bindner, A. (2014). *Hacking com Kali Linux: Técnicas práticas para testes de invasão*. Novatec Editora Ltda, São Paulo.

Memon, F., Garrett, O., and Pleshakov, M. (2018). *MODSECURITY 3.0 & NGINX: Quick Start Guide*. NGINX, Inc.

OWASP (2020). Web Application Firewall | OWASP. Disponível em: https://owasp.org/www-community/Web_Application_Firewall. Acessado em: 15/01/2021.