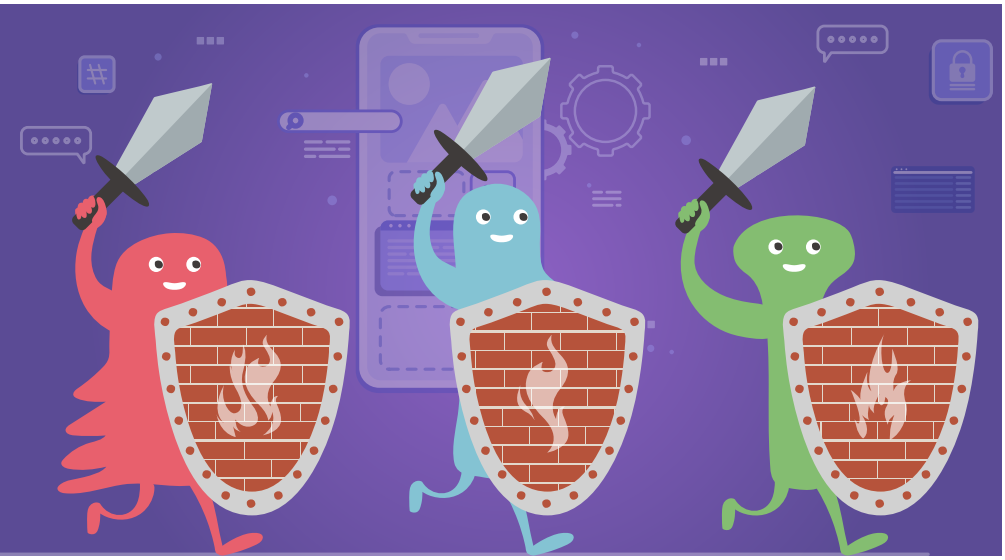


Drei freie Web-Application-Firewalls im Vergleich

Beschützer im Web

Markus Stubbig

Logging, Anmeldung und Sicherheit selbst programmieren? Unsinn! Einfach eine Web-Application-Firewall vor die eigene Webanwendung klemmen und umfangreichen Schutz genießen. Dieser Artikel stellt drei kostenfreie Webbeschützer vor.

Würden sich alle Besucher Ihrer virtuellen Präsenz an die Spielregeln halten, bräuchten Computernetze weder Firewalls noch Logdateien. Leider sind in der Realität unter dem Publikum auch schwarze Schafe vertreten. Die Palette reicht vom Skript-Kiddie bis zum Auftragshacker – und die sind Ihrer Webanwendung nicht wohlgesinnt. Wer Webdienste im Internet anbietet, muss sich auch mit bösartigen Zugriffen beschäftigen.

Der erste Schutz der eigenen Webanwendung ist ein aktueller Webserver auf einem gehärteten Betriebssystem. Wer mehr Verteidigung benötigt oder ältere Anwendungen einsetzen muss, kann den Schutz mittels Web-Application-Firewall (WAF) erweitern, einer zusätzlichen Software, die die Webanfragen untersucht, bevor sie die eigentliche Webapplikation erreichen. Erkennt die WAF eine verdächtige Aktivität, kann sie die Verbindung blockieren. Das ist längst kein Geheimtipp mehr, findet sich die WAF doch schon seit 2006 im BSI-Maßnahmenkatalog Sicherheit von Webanwendungen (siehe ix.de/z6kh).

iX-TRACT

- Web-Application-Firewalls (WAF) analysieren HTTP-Zugriffe und blockieren verdächtige Anfragen.
- Durch geschickte Platzierung im Datenstrom kann eine WAF auch verschlüsselten HTTPS-Traffic mitlesen.
- WAFs sind kein Allheilmittel für schlecht programmierte Webanwendungen.

Eine klassische Firewall kann den HTTPS-Zugriff lediglich erlauben oder verbieten. Die spezialisierte Webfirewall untersucht die HTTP-Daten auf Verdächtiges. Beispielsweise wird sie stutzig, wenn sich ein Benutzer mit dem Namen `'DROP DATABASE forum'` anmelden möchte, denn hier probiert der Angreifer, einen SQL-Befehl einzuschleusen.

Artenvielfalt

Grundsätzlich kann eine WAF ein Teil des Webservers sein oder als vorgeschaltete Anwendung arbeiten. Die gängigen Webserver lassen sich über Module erweitern und erhalten darüber den gewünschten Webschutz: Für Apache HTTP gibt es ModSecurity, für Nginx existiert NAXSI und wer den IIS von Microsoft einsetzt, wird mit WebKnight den gewünschten Schutz erreichen.

Aber nicht jeder Webserver kommt mit einem bequemen Security-Modul daher. Wer beispielsweise eine ältere, webbasierte Industriesteuerung betreiben muss, kann den Webbeschützer als dedizierte Appliance implementieren. Hierfür schnüren die allermeisten Netzausrüster fertige Pakete, in denen nicht selten ein Open-Source-Produkt steckt. Beispielsweise nutzt die Sophos XG Firewall das Gespann aus Apache und ModSecurity, während F5 Networks auf Nginx setzt und die gleichnamige Firma kurzerhand gekauft hat.

Im Zeitalter der Cloud gibt es passende As-a-Service-Angebote, die die Webseiten ihrer Kunden per WAF absichern. Das Prinzip ist dasselbe wie bei der On-Premises-Appliance, nur dass der Cloud-Anbieter die Software bereitstellt und hostet. Zu den großen Playern gehören Akamai, Cloudflare und Fastly.

Freie Kandidaten

Dieser Artikel stellt drei Open-Source-WAF-Produkte vor, die sowohl im eigenen Datacenter als auch in der Cloud funktionieren. Dabei setzen die Anbieter auf bekannte Open-Source-Helferlein oder fertige Bibliotheken. Der Mehrwert dieser Ansätze ist eine schicke Weboberfläche, die die Zauberei in etlichen Konfigurationsdateien übernimmt und als Webmenüs organisiert.

Die Auflistung und die Übersicht in der Tabelle „Freie Web-Application-Firewalls“ kürt keinen Gewinner. Sie soll zeigen, welches Paket welche Vorteile bietet und am besten zur eigenen Umgebung passt. Die angebotenen Tools stehen un-

ter einer freien Lizenz und sind durchgängig IPv6-ready.

Das grundlegende Prinzip ist bei allen drei Kandidaten gleich: Die Appliance arbeitet wie in Abbildung 1 gezeigt als Reverse-Proxy und nimmt die Webanfragen der Clients an. Im Hintergrund untersucht sie deren Inhalt und stellt dieselbe Anfrage an den Zielserver, der für den Client nicht direkt erreichbar ist.

Eine Webanwendung benötigt für ihren neuen Beschützer keine Änderung. Für die Clients ist eine Anpassung am DNS notwendig, da der aufgelöste Servername nicht mehr die IP-Adresse des Webservers zurückliefern muss, sondern die Adresse des Reverse-Proxys.

Vulture Project

Das Vulture Project bietet seit rund zehn Jahren umfassenden Schutz für Webdienste. Die Appliance kommt als fertige virtuelle Maschine (VM), als virtuelle Festplatte oder als Installationsskript. Das richtet sich an Nutzer von Cloud-Plattformen, die keine eigenen VMs importieren können. Für alle anderen ist die schlüsselfertige VM der einfachste Weg, da nach dem Importvorgang die Installation bereits abgeschlossen ist.

Nach dem Start der VM legt der Admin auf der Kommandozeile das Passwort fest und welche Rolle die Appliance innerhalb des Clusterverbunds spielt. In diesem Sinne gilt ein einzelnes Gerät als Ein-Knoten-Cluster. Anschließend gibt die Appliance den Webserver frei und die weitere Konfiguration erfolgt per Browser und Maus.

Die Einrichtung von Vulture verteilt sich über viele Bausteine, die später als Workflow zusammenkommen. In der Taxonomie des Projekts ist die Webfirewall ein Listener und der zu beschützende Webdienst eine Application. Der Listener benötigt ein TLS-Profil, das ein x.509-Zertifikat enthält. Der Webschutz heißt hier „WAF Policy“ und dieser bedient sich aus den „WAF Rulesets“. Wenn gewünscht, kommt noch die „Access Control“ hinzu, die klassisch IP-Adressen blockiert oder nach Kriterien im HTTP-Header filtert. Nicht fehlen sollte ein Logger, der Zugriffe und Verstöße protokolliert.

Mit diesen Puzzleteilen entscheidet der Admin im Workflow-Menü, welche Webanfragen welchen Webserver erreichen sollen und welche Policy dafür gilt. Den konfigurierten Ablauf stellt Vulture als Netzdiagramm dar. Der beispielhafte Workflow in Abbildung 2 präsentiert dem Webclient eine HTTPS-Seite, während im Backend nur unverschlüsseltes HTTP gesprochen wird. Um die Reihenfolge der Abarbeitung kümmert sich Vulture selbst, was die Fehlerquellen minimiert.

Unter der schicken Oberfläche legt Vulture die notwendigen Filterregeln an, konfiguriert den Reverse-Proxy und richtet den Log-Forwarder ein. Ob der eigene Aufbau anschließend funktioniert, lässt sich praktisch mit einem regulären Webbrowser prüfen, der den Listener von Vulture mit dem Namen des Webservers anspricht. Die Namensauflösung erledigt ein DNS-Server – in Laborumgebungen kann die lokale Hosts-Datei aushelfen oder ein Browser-Plug-in wie LiveHosts (siehe ix.de/z6kh).

In größeren Umgebungen kann der Workflow mehrere Applikationsserver ansprechen und damit die Last unter den Servern aufteilen. Ist hohe Verfügbarkeit gefordert, darf Vulture sein Clusterkonzept endlich sinnvoll einsetzen. Beim initialen Einrichten eines weiteren Vulture-Knotens benötigt dieser die IP-Adresse des Masterknotens und den API-Schlüssel, den ebenfalls der Master im Web-GUI unter „Cluster Config“ bereithält. Sobald die beiden Webfirewalls somit einen Cluster darstellen, überträgt der Master seine Konfiguration an den neuen Knoten, der damit einsatzbereit ist.

Freie Web-Application-Firewalls

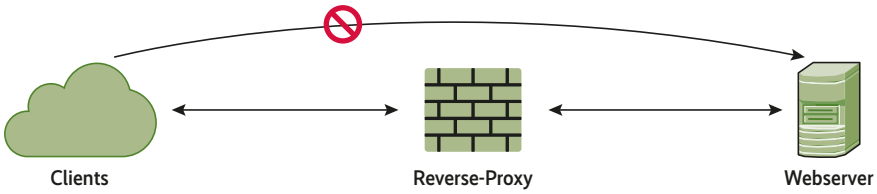
	Vulture Project	Roxy-WI	Janusec
untersuchte Version	1.2.0	5.4.0	1.2.8
Lizenz	(L)GPLv3	Apache 2.0	AGPLv3
verfügbar seit	September 2011	Januar 2018	Juli 2018
Herkunft	Frankreich	Russland	China
bevorzugte Distribution	HardenedBSD	CentOS, Debian, Ubuntu	CentOS, Debian
als Docker-Container	–	✓	✓
als VM	✓	–	–
als Cloud-Dienst	–	✓	–
Datenbank-Backend	MongoDB	SQLite/MySQL	PostgreSQL
Hochverfügbarkeit	CARP	Keepalived	DNS/GSLB
Reverse-Proxy-Software	HAProxy	HAProxy	eigene Methoden
WAF-Software	mod_defender	ModSecurity	eigene Methoden
WAF-Regeln	NAXSI	ModSecurity	eigene Methoden
Backup/Restore	–/–	✓/–	–/–
Web-API	✓	✓	–
Client-/User-Authentifizierung	✓	✓	✓
Server Health Monitoring	✓	✓	✓
kommerzieller Support	–	✓	–
Speicher-/CPU-Bedarf	hoch	niedrig	niedrig
Stabilität	*	*	***

weitere Informationen zu den Projekten siehe ix.de/z6kh

HA über das Betriebssystem

Für die Hochverfügbarkeit auf IP-Ebene greift Vulture auf Methoden des Betriebssystems zurück. Mit dem eingesetzten Common Address Redundancy Protocol (CARP) präsentiert sich der Cluster den anfragenden Clients unter einer zusätzlichen IP-Adresse, die nur der aktive Knoten beantwortet. Damit entsteht ein Aktiv-passiv-Szenario, das den Ausfall des aktiven Vulture-Knotens nach etwa vier Sekunden erkennt. Sogleich springt der nächste Knoten ein und übernimmt die Geschäfte.

Das Herzstück der Webfirewall ist ModDefender, ein WAF-Modul für den Apache. Da Vulture aber keinen Apache benutzt, läuft ModDefender als eigenständiger Daemon und analysiert im Auftrag des Reverse-Proxys, den die Entwickler bevorzugen: HAProxy. Unter der Haube



Die Web-Application-Firewall arbeitet als Reverse-Proxy und lässt damit keinen direkten Webzugriff auf die Server zu (Abb. 1).

kommen noch weitere Open-Source-Kollegen dazu: Das Betriebssystem ist HardenedBSD, Logdateien liegen wahlweise in einer MongoDB oder in Redis und das Regelwerk stammt von NAXSI (Nginx Anti-XSS & SQL Injection).

Wer mehr braucht als reinen Webschutz, wird Vulture zu schätzen wissen. In der Disziplin Authentifizierung kann Vulture aus einer großen Zahl von Identity-Providern wählen. Dazu gehören neben klassischen Protokollen wie LDAP und RADIUS große Namen wie Google, Azure, GitHub und weitere Mitglieder, die OpenID-Connect beherrschen. Optional lässt sich die Anmeldung um ein Einmalpasswort zu einer Mehr-Faktor-Authentifizierung (MFA) erweitern.

Weiter geht es zum Logging: Vulture protokolliert Zugriffe auf die angebotenen Webdienste und kann die Meldungen per künstlicher Intelligenz auf Anomalien untersuchen. Die KI-Engine ist eine sehr junge Erweiterung der Webfirewall und bisher alarmiert sie nur, ohne sich in die Webzugriffe einzumischen.

Sind die Komponenten wie Listener, Application und Policy erst einmal angelegt, läuft die Konfiguration als Wizard. Das Ergebnis ist eine funktionierende Webfirewall mit buntem Netzdiagramm für die eigene Dokumentation. Die Möglichkeiten zum Filtern sind bemerkenswert; der HTTP-Header lässt sich auf vielerlei Arten verbiegen. Hochverfügbarkeit der Webfirewalls und Lastverteilung der Applikationsserver sind von Haus aus dabei.

Demgegenüber scheinen einige Features noch nicht ausgereift (beispielsweise Logging) und manchmal klemmt der Workflow ohne erkennbaren Grund. Auch benötigt Vulture für die vielen Dienste mehr Rechenpower und Arbeitsspeicher als die anderen Kandidaten. Bei der Fehlersuche auf der Kommandozeile kommt erschwerend hinzu, dass alle Dienste in einem BSD-Jail laufen und sich nicht „mal eben so“ im Debugmodus starten lassen.

Roxy-WI

Roxy-WI gibt es nicht als fertige Webfirewall, sondern als Softwaredistribution. Per Weboberfläche entscheidet der Admin, welcher Linux-Host zur Webfirewall ausgerüstet werden soll, und Roxy-WI macht sich sogleich per SSH, wget und Compiler ans Werk. Das Ergebnis ist eine startbereite Webfirewall, die im nächsten Schritt ihre Konfiguration erhält.

Die folgende Einrichtung ist deutlich geradliniger als bei Vulture Project: Im Bereich „Haproxy/Add proxy“ legt man fest, welcher Server mit welcher IP-Adresse zum Reverse-Proxy mutieren soll (Abbildung 3). Dasselbe Webmenü enthält noch die Angaben zu den nutzbaren Schutzmechanismen und welche(n) Server die Webanfragen im Backend treffen sollen. Im Hintergrund macht sich Roxy-WI erneut per SSH auf den Weg zum ausgewählten Server und hinterlässt die angepasste Konfigurationsdatei.

Danach passiert zunächst noch nichts, denn den Neustart des entsprechenden Dienstes muss der Admin per Web-GUI (oder Kommandozeile) selbst triggern. Ab jetzt kann die neue Webfirewall mit HTTP-Anfragen beschossen werden. Alles, was sich böse anfühlt, schreibt Roxy-WI in eine lokale Logdatei.

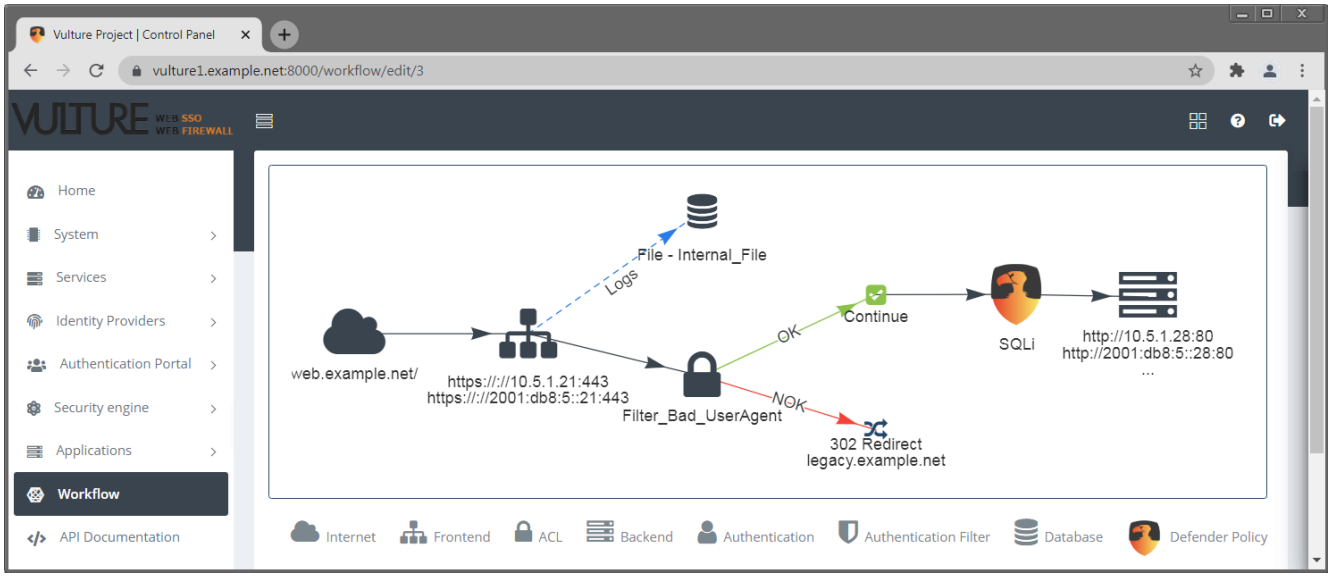
Die Stellschrauben im Web-GUI sind begrenzt. Für fehlende Optionen oder mangelhafte Filter bietet die Webseite stets die Möglichkeit, selbst passende Konfigurationsanweisungen zu stricken. Diese müssen für den Proxydienst HAProxy verständlich formuliert sein, sonst verweigert die Webfirewall beim nächsten Neustart ihre Arbeit.

Für den Wunsch nach Ausfallschutz kann Roxy-WI auf einem weiteren Server die benötigte Software kompilieren, konfigurieren und starten. Aus den beiden fertigen Webfirewalls entsteht im Menü Keepalived/HA der Cluster. Genau wie bei Vulture ist nur eine von beiden aktiv und die andere wartet auf die Havarie. Wer welche Rolle hat, zeigt die Weboberfläche als MASTER oder BACKUP an.

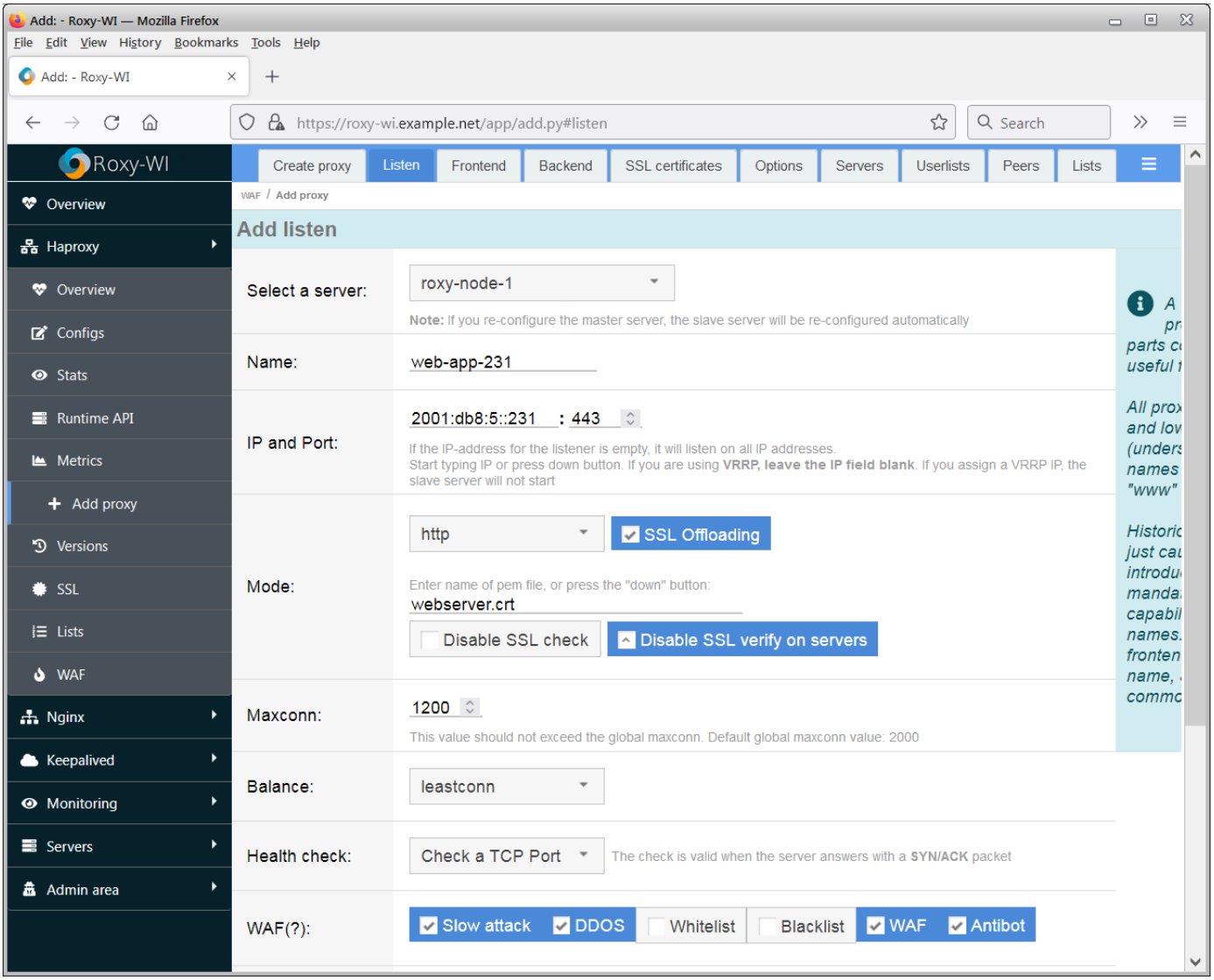
Oberhalb der reinen Webfirewall überwacht Roxy-WI die Dienste auf den verwalteten Servern und holt sich die Metriken regelmäßig ab. Daraus entsteht eine riesige Tabelle über alle Dienste, Server und Webseiten. Für ein schnelles Feedback färbt Roxy-WI problematische Zustände rot ein, sodass der Admin sofort sieht, wo es klemmt.

Roxy-WI verwandelt die zentral festgelegte Firewallrichtlinie in die Syntax des jeweiligen Dienstes und betankt damit die Webfirewalls. Und wenn diese noch keine Software haben, kümmert sich die Software auch um die Installation. Auf diese Weise lassen sich viele Webfirewalls von zentraler Stelle bequem erstellen und konfigurieren. Wenn alles läuft, sammelt Roxy-WI Statistiken und gibt somit Feedback über die erstellte Webfarm.

Auf der Kehrseite der Medaille ist ein geschulter Blick in die Konfigurationsda-



In Vulture komponiert der Admin den Weg vom Clientzugriff durch die Webfirewall bis zum Server mit grafischen Tools (Abb. 2).



Roxy-WI erwartet alle Angaben zur angebotenen Web-App auf seiner Seite (Abb. 3).

teien nötig, wenn das Set-up nicht so funktioniert wie geplant. Auch wenn alles richtig erscheint, passt die generierte Syntax von Roxy-WI eventuell nicht zur eingesetzten Version von HAProxy oder Keepalived. Immerhin akzeptiert der Entwickler dieses Problem und erlaubt ein direktes Editieren der Dateien per GUI.

Ein weiteres Manko ist die versteckte Alarmierung bei von Roxy-WI vereitelten Angriffen. Die Meldungen von ModSecurity fehlen im Web-GUI und schlummern stattdessen in irgendeiner Logdatei. Hier könnte das Dashboard den Admin informieren, wie erfolgreich Roxy-WI die Webserver beschützt.

Janusec

Im Unterschied zur Konkurrenz betritt Janusec ohne großen Schnickschnack die Bühne. Das Ergebnis ist eine übersichtliche Weboberfläche und eine simple Einrichtung: Erstens Zertifikat einspielen. Zweitens IP-Adresse der Webanwendung eintippen. Wäre Janusec kostenpflichtig, wäre dies ein überzeugendes Werbeversprechen.

Janusec kommt als Tarball von GitHub und benötigt lediglich noch eine PostgreSQL-Datenbank, um die Konfiguration abzulegen und mit den Clusterknoten zu teilen. Nach der Installation steht eine Webseite für die Administration bereit.

Im Menübereich Application erfragt die Weboberfläche nur, welches Zertifikat der Browser erhalten soll und an welches Backend die Webanfragen gehen. Die Konfiguration in Abbildung 4 ist beispielhaft, enthält aber alle Optionen, die Janusec anbietet. Schlägt der Webschutz Alarm, erfährt man dies im Bereich WAF Logs. Das Dashboard präsentiert eine Zugriffsstatistik – Zusammenfassungen und Analysen finden sich hier aber nicht.

Ein kleines Goodie gibt es doch: Schutz vor Challenge Collapsar (CC). Bei dieser Technik versucht ein Angreifer, mit vielen Webzugriffen den Zielserver so stark auszulasten, dass er keine weiteren Anfragen bearbeiten kann. Anders als beim Distributed Denial of Service (DDoS) sind die Webzugriffe beim CC so formuliert, dass der Webserver sie aufwendig abarbeiten muss, da sie beispielsweise einen komplexen Datenbankzugriff erfordern. Janusec möchte den CC-Angriff zumindest abschwächen und blockiert nach mehreren unerwünschten Zugriffen die angreifende IP-Adresse. Die Idee ähnelt den Blockade-Tools DenyHosts und Fail2ban.

Da Janusec aus dem asiatischen Raum stammt, sind bei den Authentifizierungsanbietern exotische Namen wie WeChat oder Feishu aufgeführt, aber auch das universelle LDAP ist dabei. Die Software greift für den Betrieb nicht auf andere Open-Source-Programme zurück, sondern

implementiert den Webschutz selbst mithilfe der verfügbaren Bibliotheken der Programmiersprache Go. Das vereinfacht die Installation, denn außer der Datenbank für die Konfiguration gibt es keine Abhängigkeiten. Auch für den Betrieb als HA-Cluster reicht die vorhandene Software aus.

Dürftig ist die fehlende Sicherheit bei der Ersteinrichtung: Der initiale Zugriff auf Janusec erfolgt stets über HTTP, also ohne Verschlüsselung. Über die Weboberfläche lässt sich sogleich eine HTTPS-Konfiguration bauen und für die weitere Einrichtung nutzen. Wer Janusec in der Cloud betreibt und nicht im heimischen Rechenzentrum, sollte für den gesicherten Erstkontakt einen VPN-Tunnel vorschalten.

Beim Thema Hochverfügbarkeit ist Janusec unkompliziert: Die erste Webfirewall wird zum primären Knoten und alle weiteren WAFs sind Replika-Nodes. Die Entscheidung fällt bei der einzigen Frage des Installationsskripts: Master oder Replika. Um den Austausch des Authentifizierungsschlüssels muss sich der Admin selbst kümmern und diesen in der Konfigurationsdatei der Repliken hinterlegen sowie die IP-Adresse des Masters angeben. Nach einem Neustart des Dienstes pusht der Clustermaster alle Einstellungen zu den Repliken, sodass jede Webfirewall ein vollwertiger Reverse-Proxy mit WAF-Funktion ist.

Für die Verteilung der Webanfragen auf die Clusterknoten bringt Janusec keine eigene Methode mit, sodass der Admin auf ein Zusatzpaket wie Keepalived zurückgreifen muss. Das Ergebnis ist derselbe Aktiv-passiv-Aufbau wie bei Roxy-WI, allerdings per Konfiguration auf der Kommandozeile.

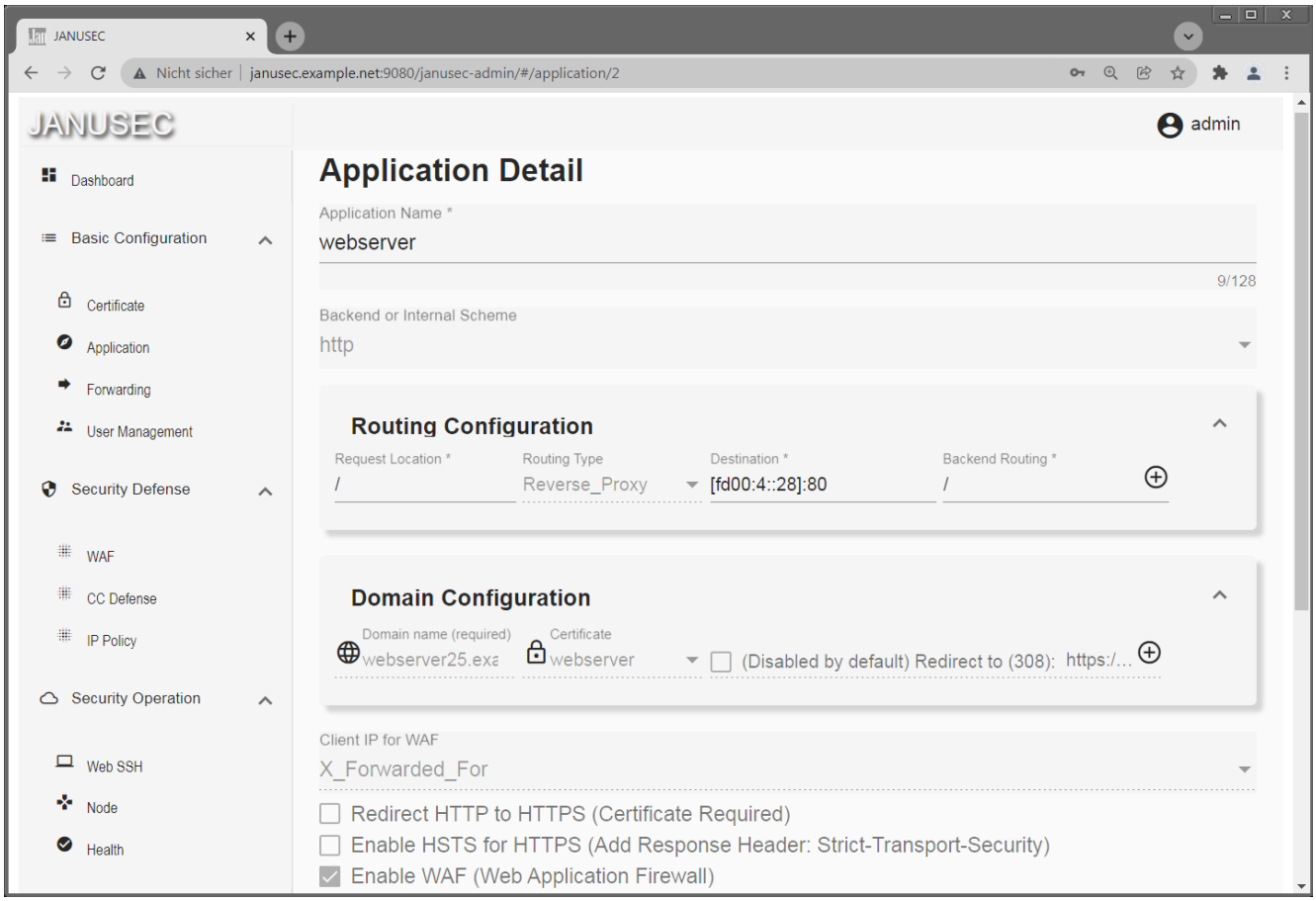
Die Webfirewall in Janusec ist simpel gestrickt und vergleicht HTTP-Anfragen mit vorkonfigurierten Mustern. Eine Anomalieerkennung oder ein bisschen künstliche Intelligenz wäre wünschenswert. Die WAF-Regeln sind als reguläre Ausdrücke formuliert, was die Erweiterung um eigene Einträge ermöglicht. Das webbasierte Dashboard liefert ein rudimentäres Feedback zur Auslastung. Der Ausbau zum Verfügbarkeitscluster mit zentraler Konfiguration ist mit wenigen Handgriffen realisiert. Wer mehr Funktionen benötigt, schaut in die Röhre, denn Plug-ins oder Schnittstellen zu anderen Tools gibt es hier nicht. Nachteilig ist auch die Ausrichtung auf Authentifizierungsanbieter aus dem asiatischen Raum, die in Europa weitgehend unbekannt sind.

Fazit

Da eine WAF eine Webanwendung nur sicherer und nicht besser machen kann, sind auch die hier vorgestellten Kandidaten kein Ersatz für sicher programmierte Anwendungen. Sie beherrschen jedoch alle das kleine Einmaleins des Webschutzes.

Trotz des übersichtlichen Funktionsumfangs ist Janusec mit Abstand die fehlerärmste und stabilste Webfirewall in diesem Artikel. Die Regeln haben die Form von regulären Ausdrücken und sind im selben Format um eigene Einträge erweiterbar. Per Dashboard liefert Janusec eine Kurzübersicht zur Auslastung und die Erweiterung zum HA-Cluster mit zentraler Konfiguration ist flott erledigt. Als Projekt aus dem asiatischen Raum kennt Janusec als Authentifizierungsanbieter neben LDAP nur exotische Namen wie WeChat oder Feishu. Wer auf Google, Facebook und Co. nicht verzichten kann, sollte eher Vulture evaluieren.

Vulture beeindruckt mit seinem großen Funktionsumfang und der grafischen Darstellung. Sind die Komponenten erst einmal angelegt, läuft die Konfiguration als Wizard ab. Das Ergebnis ist eine funktio-



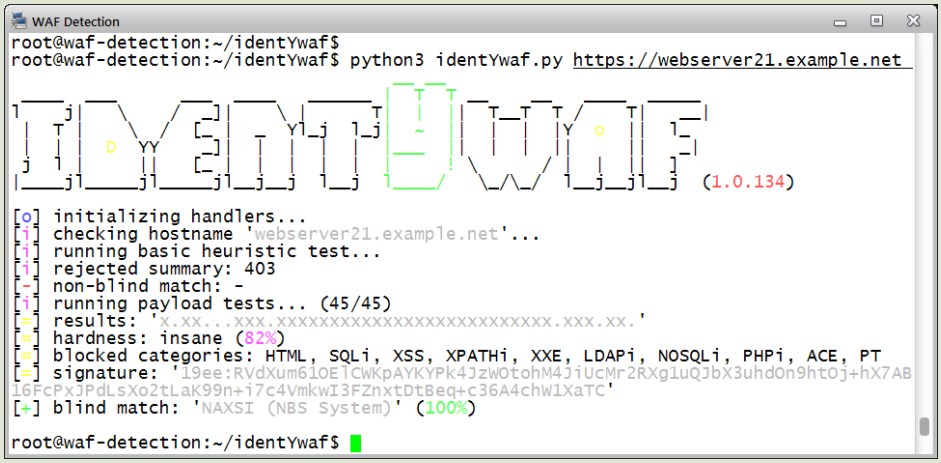
Die Konfiguration einer Web-Application ist bei Janusec übersichtlich (Abb. 4).

Praxistipps zu Web-Application-Firewalls

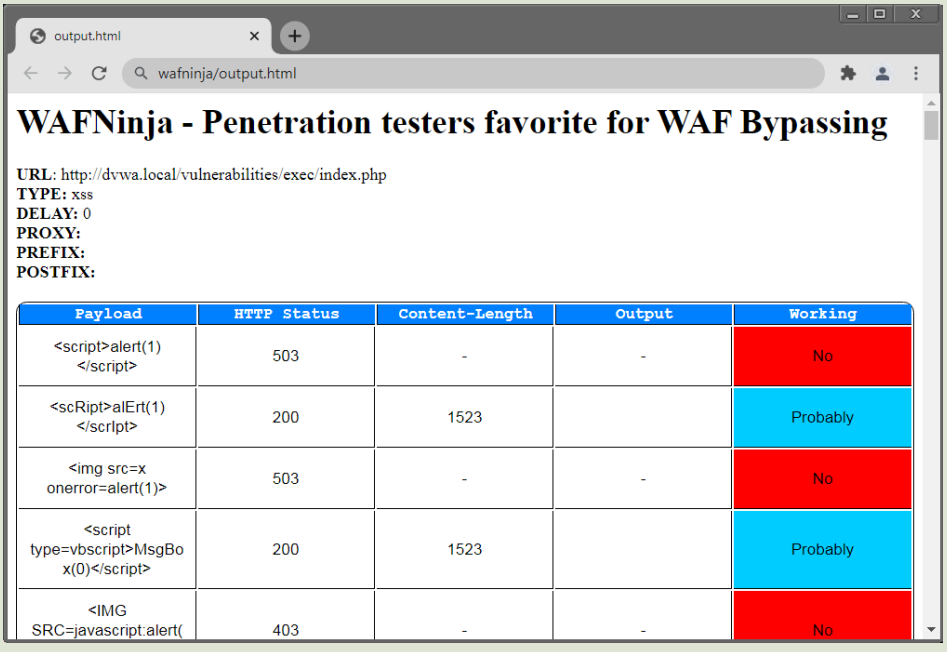
Die vorgestellten Webfirewalls bringen Ausfallschutz als Aktiv-passiv-Cluster mit. Damit ist die beschützte Webanwendung im Fehlerfall einer WAF noch erreichbar, aber die Hälfte der verfügbaren Firewall-Power liegt brach.

Ein kleiner Trick führt dieses Konstrukt zu einem höheren Wirkungsgrad: Die Firewalls erhalten eine zusätzliche HA-Gruppe mit einer weiteren virtuellen IP-Adresse. In dieser HA-Gruppe ist die bislang passive Firewall der Masterknoten und umgekehrt. Im DNS-Eintrag der eigenen Webapplikation stehen beide virtuellen IP-Adressen des WAF-Clusters, sodass sich der Webbrowser für einen Clusterknoten entscheiden muss. Beide Firewalls geben sich gegenseitig Rückendeckung und übernehmen die Last des anderen, sollte der Partner aussteigen.

Das Konzept ist nicht auf zwei Worker limitiert. Jede weitere Webfirewall benötigt eine neue HA-Gruppe mit IP-Adresse, für die sie der Master ist und alle anderen die Vertretung. Leider lässt sich in diesem Szenario die Lastverteilung nicht kontrollieren. Im ungünstigsten Fall erhält die erste Webfirewall 90 Prozent der Anfragen und die zweite Firewall langweilt sich mit den restlichen 10 Prozent.



Das Erkennungstool identYwaf hat die Web-Application-Firewall zuverlässig erkannt (Abb. 5).



WAFNinja hat die Eingabeprüfung der Webfirewall ausgetrickst und zeigt stolz die gefundenen Schwachstellen (Abb. 6).

WAF aufdecken

Im normalen Betrieb bemerken die legitimen Besucher nichts von der Webfirewall. Allerdings ist die WAF nicht komplett unsichtbar und lässt sich im Datenpfad erkennen. Je nach Hersteller präsentiert sich die WAF stolz im HTTP-Header oder bleibt tief unter dem Radar.

Wer eine Webfirewall aufdecken will, muss nicht den HTTP-Datenstrom auf verdächtige Einträge durchstöbern. Der bequeme Weg sind fertige Tools, die die (eigene) Webseite analysieren und sogar den Namen der WAF ermitteln. Zu den bekannten Whistleblowern gehören WhatWaf, wafw00f und identYwaf, die zwischen 60 und 80 unterschiedliche Webfirewalls zuverlässig erkennen. Sogar der Portscanner nmap hat ein Skript dabei, das immerhin eine Handvoll WAF-Anbieter auseinanderhalten kann.

Die Aufklärer senden provokante Webanfragen an den Zielserver, auf die die WAF reagieren muss. Anhand der Antworten kann die Software eine Vermutung äußern oder zumindest einige WAF-Produkte ausschließen. Im Beispiel von Abbildung 5 hat identYwaf die Janusec-Firewall detektiert.

Webschutz umgehen

Es nützt wenig, eine solide Webfirewall aufzubauen, wenn die Hacker an ihr vorbeischleichen können. So einfach ist der Spaziergang allerdings nur, wenn der direkte Zugriff auf den Webserver immer noch möglich ist. Dabei ist die Kunst nicht etwa, die WAF physisch zu umgehen, sondern die bösartigen Webzugriffe so zu tarnen, dass die WAF sie nicht mehr erkennt.

Jetzt ist der Admin erneut in der Rolle des Angreifers, der mithilfe von Tools seinen eigenen Webbeschützer überlisten möchte. Hierbei unterstützt WAFNinja, eine Software zum Austricksen der Eingabeprüfung von Webservern. Wenn diese erfolgreich ist, benötigt das Regelwerk der Webfirewall eine Anpassung. In Abbildung 6 hat WAFNinja in einem absichtlich unsicheren Webdienst gewütet und präsentiert seine Trophäen.

nierende Webfirewall mit buntem Netzdiagramm, bemerkenswerten Filtermöglichkeiten und Hochverfügbarkeit. Einige Funktionen scheinen noch nicht ausgereift und Vulture benötigt mehr Ressourcen als die anderen Kandidaten.

Roxy-WI ist auf Masse aus und verteilt Software und Konfiguration auf beliebig vielen Webfirewalls. Haben diese noch keine Software, kümmert es sich auch um die Installation. So lassen sich viele Webfirewalls von zentraler Stelle aus bequem erstellen und konfigurieren. Wenn alles läuft, sammelt Roxy-WI Statistiken und gibt somit Feedback über die erstellte Webfarm. Allerdings erfordert die Fehlersuche einen durchaus geschulten Blick, ob die von Roxy-WI generierte Syntax zu den eingesetzten Versionen von HAProxy oder Keepalived passt. (avr@ix.de)

Quellen

Links zu den Projektseiten und zum BSI-Maßnahmenkatalog „Sicherheit von Webanwendungen“ siehe ix.de/z6k

Markus Stubbig

ist Systementwickler mit Schwerpunkt Netzwerk und Linux im Automobilumfeld.