Web Application Firewalls

Unverzichtbar für wirklich sicheres E-Business

1. Fehler in Webanwendungen sind dort zu beheben, wo sie auftreten, nämlich in der Webanwendung, und nicht durch Filter in der Web Application Firewall (WAF). 2. Eine WAF kann auf der Anwendungsebene nicht das leisten, was eine konventionelle Firewall auf der Netzwerkebene kann. Zwei Argumente gegen WAFs gleich am Anfang dieses Artikels – und dennoch soll der Artikel für den Einsatz von WAFs werben. Denn trotz der genannten Einschränkungen: Eine WAF ist in der Lage, die Sicherheit von Webanwendungen dramatisch zu erhöhen. Nur muss man sich dazu von der „Klassische-Firewall-Perspektive" ein wenig entfernen und einen anwendungsgerechteren Blickwinkel einnehmen.

Schauen wir zunächst aber noch einmal auf die vorgenannten Einschränkungen: Ein WAF ist zunächst einmal als zusätzliche Sicherungslinie zu verstehen. Bleiben Schwachstellen in der Webanwendung unentdeckt, so kann eine gut auf die zu schützenden Anwendungen abgestimmte WAF das Angriffsfenster (also die Möglichkeit, einen Angriff zu führen) deutlich verkleinern und in vielen Fällen das Schlimmste verhindern. Ein 100%-iges Blocken von schädlichen Zugriffen lässt sich auf der Webanwendungsebene grundsätzlich nur im Spezialfall, nicht aber für den Allgemeinfall lösen. Somit kann eine WAF hier auch keine Wunder wirken.

Nicht nur aus dieser prinzipiellen Schwäche folgt, dass Schwachstellen in Webanwendungen nicht an der WAF behoben werden sollen (Ausnahme sie unten). Die Webanwendung selbst muss so sicher sein, dass sie vollständig ohne WAF auskommt.

Nun aber zu gewichtigen Gründen, warum eine WAF ein Bestandteil einer jeden Webinfrastruktur sein sollte.

URL-Encryption

Das vielleicht sensibelste Datum einer Webanwendung ist die SessionID. Sie wird nach der erfolgreichen Authentisierung ausgegeben und ist fortan das alleinige Merkmal, das den Zugriff auf das Konto des angemeldeten Benutzers regelt. Gelangt sie in die Hände eines Angreifers, so ermöglicht sie diesem den direkten Zugriff auf das Benutzerkonto – die Kenntnis von UserID und Passwort ist nicht mehr erforderlich. Nun sind jedoch die heutigen Programmierframeworks nur sehr rudimentär mit Vorkehrungen zum Schutze der SessionID ausgestattet. Im Gegenteil: Eine Webanwendung, die sich weitgehend auf die State-of-the-Art-Mechanismen abstützt, ist eher anfällig für die vielen Angriffsformen auf die SessionID. Diese lauten da etwa: Session-Hijacking, Session-Fixation, Session-Denial-of-Service, Session Riding, etc. Kommt ein solcher Angriff durch, so sind die Daten des angegriffenen Benutzers gefährdet - deren Vertraulichkeit ebenso wie deren Integrität. Das von einigen WAFs angebotene Feature der URL-Encryption schiebt all diesen Angriffsformen einen sicheren Riegel vor. Und dies – von Sonderfällen abgesehen – ohne dass in die Webanwendung eingegriffen werden müsste. Dies gelingt auf folgende Weise:

Der von der Anwendung in die Seite ausgegebene Link dieser Form

```
http://www.example.com/action.do?cmd=adduser&role=admin
```

wird von der WAF in die Form

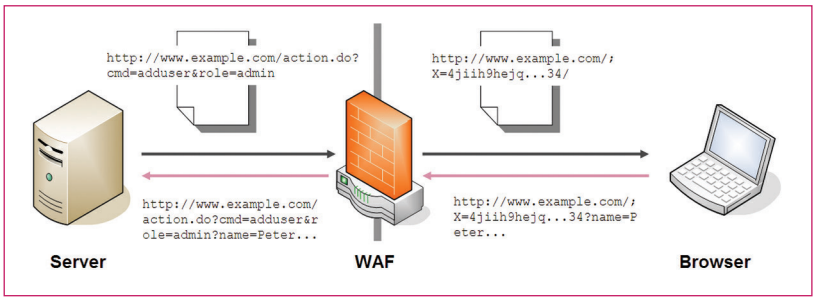
```
http://www.example.com/;X=4jiih9hejqp30d23jtojpwfe40903ndjzh937jmdwoirjdiie34/
```

übersetzt und beim Benutzer-Klick von der WAF wieder in die für die Anwendung verständliche Form zurücktransformiert (siehe Bild).

Die wahre Gestalt des Links gelangt damit niemals nach außen. Der Knackpunkt dabei ist: Die Verschlüsselung ist für jede Session, d.h. jeden Benutzer, eine andere. Das hier hinter X= stehende *Secret* unterscheidet sich von Benutzer zu Benutzer und von Login zu Login. Ein Angreifer, der in den Besitz der SessionID gelangt ist, müsste, damit die genannten Angriffe gelingen, auch das Secret des jeweiligen Benutzers ermitteln. Ein Unterfangen, das ihm, von besonderen Gegebenheiten einmal angesehen, nicht gelingen wird. Neben dieser vollständigen Verhinderung der auf die Session zielenden Angriffe bewirkt die URL-Encryption auch die Verkleinerung des Angriffsfensters für andere Arten von Angriffen. So sind beispielsweise die Manipulationsmöglichkeiten von Parametern im URL deutlich eingeschränkt, was bestimmte Formen der Privilege Escalation – also der Erhöhung der eigenen Zugriffsprivilegien – verhindert. Ähnliches gilt für das Erraten von Zugriffspfaden auf geschützte Objekte durch Brute-Force- oder Enumerationstechniken.

Site Usage Enforcement

Das für eine Webanwendung wohl am schwierigsten in den Griff zu bekommende Problem ist die Abwehr von automatisierten Angriffen. Wenn also ein Angreifer ein Skript oder Tool nutzt und die Anwendung mit Zigtausenden von Requests bom-bardiert: um Passworte zu erraten, versteckte Objekte ausfindig zu machen, Kontaktformulare zum Spammen zu missbrauchen oder um die Site durch Herstellung einer Überlast in die Knie zu zwingen. Der Schutz vor diesen Bedrohungen ist alles andere als trivial – schnell ist ein legitimer Benutzer



ausgesperrt oder die Anwendung auch für Nichtangreifer nicht mehr zugänglich – und nirgendwo besser aufgehoben als in einer eigenen Komponente außerhalb der Webanwendung. Und so haben Application Firewalls sich dieser Aufgabe angenommen mit Features wie diesen:

> **Blocken von Zugriffen** abhängig von der Wiederholungshäufigkeit, dem Zeitverhalten und Merkmalen wie der IP-Adresse, dem User-Agent- und Referer-Header.
> **Referer-Checking**: Durch intelligente Prüfung des Referer-Headers, den der Browser bei jedem Zugriff mitschickt, lassen sich (bis zu einem gewissen Grad) Phishing-Attacken und missbräuchliche Verlinkung der eigenen Seiten erkennen und verhindern. Auch ist es damit elegant möglich, die ungewollte Überlastung zu verhindern, die dadurch zustande kommt, dass Seiten eine plötzliche Berühmtheit erlangen, etwa durch Verlinkung auf Nachrichtensites wie spiegel.de oder in Newslettern.

Lösung des Patch-Dilemmas

„Never touch a running System" - jeder Betriebsverantwortliche wird wohl schon einmal die Erfahrung gemacht haben, dass sein System nach dem Einspielen eines Patches nicht mehr so gut funktioniert hat wie davor. Patches wollen also zunächst ausgiebig getestet werden. Was aber, wenn es sich bei dem Patch um die Behebung einer entdeckten Schwachstelle in einer Komponente der Webpräsenz handelt? Etwa im Content-Management-System und für jeden über das Web ausnutzbar. Dann ist schnelles Handeln erforderlich. Doch soll man sich für die sofortige Behebung der Schwachstelle entscheiden und das Risiko schädlicher Seiteneffekte des Patches inkaufnehmen? Aus diesem Dilemma kann die WAF heraushelfen. Oft findet die Ausnutzung derartiger Schwachstellen auf der Anwendungsebene statt und lässt sich einfach durch Definition eines geeigneten Filters in der WAF blocken – solange bis der Patch getestet und eingespielt ist.

Behebung von Schwachstellen

Folgendes Szenario: Der Penetrationstest der Webanwendungen hat Schwachstellen in alten Webanwendungen aufgezeigt. Eine Behebung wäre unverhältnismäßig teuer und ein Abschalten kommt auch nicht in Frage. Hier darf ausnahmsweise die Behebung mithilfe der WAF erfolgen. Eine klassische Firewall kann in aller Regel nicht geeignet konfiguriert werden. Eine WAF, die mit Begriffen wie Formular, Feld und Session umgehen kann, lässt sich hingegen sehr wohl in dem gewünschten Differenzierungsgrad auf die jeweilige Schwach-stelle justieren. Insbesondere bei so verbreiteten Sicherheitsproblemen wie Cross-Site Scripting und SQL-Injection lassen sich die Filtertechniken von WAFs wirkungsvoll anwenden. Aber Achtung: Das Beheben von Schwachstellen mithilfe von WAF-Filtern darf nicht zur Regel werden. Das käme dem Automobilhersteller gleich, der bei seinen Autos auf die Knautschzone verzichtet, weil sie ja einen Airbag haben. Die Hauptfunktion der WAF ist, was das Filtern betrifft, die der zusätzlichen Sicherungslinie für all die Schwachstellen in der Anwendung, die unentdeckt geblieben sind.

Fazit

Eine WAF bringt zusätzlich zu Ihrer Rolle als Second-Line-of-Defense wirkungsvolle Techniken zur Absicherung von Webanwendungen mit, die weder von der Webanwendung selbst noch von einer klassischen Firewall erbracht werden können. Sie sind damit ein wesentliches Element einer sicheren Webinfrastruktur.

Thomas Schreiber ist Geschäftsführer der SecureNet GmbH in München und Berater für Web Applicationen Security