Best Practices:

Einsatz von Web Application Firewalls

Version 1.0.2, März 2008

Autor: OWASP German Chapter
unter Mitwirkung von:
Maximilian Dermann
Mirko Dziadzka
Boris Hemkemeier
Achim Hoffmann
Alexander Meisel
Matthias Rohr
Thomas Schreiber

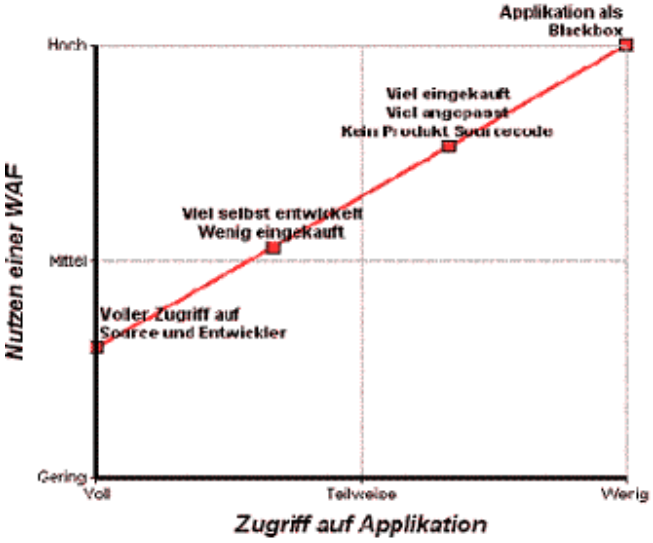
Abstrakt

Webanwendungen aller Art, ob Webshop oder Partnerportal, werden in den letzten Jahren nachweislich immer stärker zur Zielscheibe von Hackerangriffen. Dabei verwenden die Angreifer Methoden, die gezielt potenzielle Schwachstellen der Webanwendungssoftware selbst ausnutzen – und deshalb von klassischen IT-Sicherheitssystemen wie Netzwerk-Firewalls oder IDS-/IPS-Systemen nicht oder nur unzureichend erkannt werden. OWASP entwickelt Tools und Best Practices um Entwickler, Projektleiter und Securitytester bei der Entwicklung und beim Betrieb sicherer Webanwendungen zu unterstützen. Einen weiteren Schutz gegen Angriffe, insbesondere für bereits produktive Webanwendungen, bietet eine noch relativ neue Klasse von IT-Sicherheitssystemen, sogenannte *Web Application Firewalls* (im folgenden einfach *WAF*), oft auch *Web Application Shields* oder *Web Application Security Filter* genannt.

Um z. B. den derzeit gültigen Sicherheitsstandard der Kreditkartenindustrie (PCI DSS - Payment Card Industry Data Security Standard v.1.1) zu erfüllen, muss u. a. entweder ein regelmäßiger Sorcecode-Review oder der Einsatz einer WAF nachgewiesen werden.

Das Dokument wendet sich vornehmlich an technische Entscheider - speziell Betriebsverantwortliche, Sicherheitsverantwortliche, Applikationseigner (Fachabteilung, technisch Anwendungsverantwortlicher), die den Einsatz einer WAF im Unternehmen evaluieren. Spezielles Augenmerk wurde – wo immer möglich – auf die Darstellung von Aufwandsabschätzungen gelegt – auch im Vergleich zu möglichen Alternativen wie z. B. Änderungen im Sourcecode.

Bei der Entscheidungsfindung bzgl. des Einsatzes von WAFs kann – neben der Wichtigkeit der Webapplikation für das Unternehmen z. B. Umsatz oder Image – der im Dokument erläuterte Begriff des *Zugriff*" auf eine Webapplikation ein gutes Kriterium sein. Anschaulich gesprochen misst der *Zugriff* auf eine spezielle Webapplikation durch ein Unternehmen, inwieweit das Unternehmen tatsächlich zeitnah notwendige Änderungen des Source Codes der Applikation selbst durchführen bzw. sicher durch Dritte herbeiführen lassen kann. Wie die unten stehende Grafik illustriert, kann eine Webapplikation, auf die das Unternehmen keinen Zugriff hat, sinnvollerweise nur von einer WAF geschützt werden (hoher Nutzen der WAF), während selbst bei einer Applikation in vollem Zugriff eine WAF als zentrale Servicestelle für diverse Dienste wie sicheres Session-Management, die für alle Applikationen gleich gelöst werden können, und als geeignete Mittel für proaktive Schutzmaßnahmen wie z. B. URL-Encryption immer noch erheblichen Zusatznutzen bietet.



Weitere Schwerpunkte sind Best Practices für die organisatorischen Abläufe bei Installation und Betrieb einer WAF sowie - gerade für größere Unternehmen – eine Erweiterung der Prozess-Organisation um die Rolle *Anwendungsverantwortlicher WAF*.

Über …

Dieses Dokument wurde vom *OWASP German Chapter* erarbeitet. Die Autoren sind Mitarbeiter von Unternehmen, die über den Einsatz und Betrieb von WAFs beraten, WAFs einsetzen und/oder WAFs herstellen.

Autoren

Maximilian Dermann	maximilian.dermann@lht.dlh.de	Lufthansa Technik AG
Mirko Dziadzka	mirko.dziadzka@artofdefence.com	art of defence GmbH
Boris Hemkemeier	boris@owasp.org	OWASP German Chapter
Achim Hoffmann	achim.hoffmann@securenet.de	SecureNet GmbH
Alexander Meisel	alexander.meisel@artofdefence.com	art of defence GmbH
Matthias Rohr	matthias.rohr@securenet.de	SecureNet GmbH
Thomas Schreiber	thomas.schreiber@securenet.de	SecureNet GmbH

Terminologie

Die in diesem Dokument verwendeten Fachbegriffe sind nicht weiter erklärt, sondern werden als bekannt vorausgesetzt. Auf ein Glossar wurde bewusst verzichtet damit der Umfang überschaubar bleibt und sich der Inhalt auf das Eigentliche – *Einsatz von Web Application Firewalls* – konzentriert.

Ausführliche Begriffserklärungen und weiterführende Beschreibungen im Zusammenhang mit WAS – Web Application Security – finden sich in:

- http://www.owasp.org/index.php/Category:Attack OWASP Category:Attack
- http://www.owasp.org/index.php/Category:Threat OWASP Category:Threat
- http://www.owasp.org/index.php/Category:Vulnerability OWASP Category:Vulnerability
- http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.de.doc WASTC
  Web Application Security Consortium: Web Security Threat Classification
- http://www.webappsec.org/projects/wafec/ WAFEC
  Web Application Security Consortium: Web Application Firewall Evaluation Criteria
- http://www.owasp.org/images/9/9c/OWASPAppSecEU2006_WAFs_WhenAreTheyUseful.ppt
  OWASP WAFs When Are They Useful

Lizenz

Inhaltsverzeichnis

A1 Einführung und Zielsetzung dieses Dokuments

A1.1 Einführung

Egal ob Internet-Filiale einer Bank, Online-Shop, Kunden-, Partner- oder Mitarbeiterportal – jede dieser Webanwendungen ist aufgrund der *always on* Natur des Internets für ihre Kunden – aber auch für Angreifer – rund um die Uhr erreichbar. Angriffe wie SQL-Injection, Cross Site Scripting oder Session Hijacking zielen dabei auf Schwachstellen in den Webanwendungen selbst – und nicht auf solche auf der Netzwerk-Ebene. Deshalb können sie auch durch klassische IT-Sicherheitssysteme wie Firewall oder IDS/IPS-Systeme nicht bzw. allenfalls unzureichend abgewehrt werden.

Grundlegend aus technischer Sicht ist dabei, dass das Web, speziell das Protokoll HTTP, nicht für die heute üblichen komplexen Anwendungen konzipiert wurde. Viele Schwachstellen haben hier ihren Ursprung: beispielsweise ist HTTP zustandslos, d. h. Sessions oder Zustände der Applikation müssen separat definiert und sicher implementiert werden. Diese Schwachstellen werden noch verstärkt durch die hohe Komplexität der häufig eingesetzten Web-Scripts, Frameworks und Web-Technologien.

Neben der neuen Einführung von Industriestandards z. B. Datensicherheitsstandard der Kreditkartenindustrie (PCI DSS v1.1) sorgen auch erste bekannt gewordene Einbrüche in Deutschland wie z. B. der Verlust von ca. 70.000 Kundendaten inkl. Kreditkarteninformationen beim Online-Tickethändler *kartenhaus.de* für ein erhöhtes Interesse an möglichen Schutzmaßnahmen gegen Angriffsmethoden auf Anwendungsebene.

Dieses Dokument befasst sich mit einer Klasse von Schutzsystemen, den *Web Application Firewalls* (WAF), die gerade zur Absicherung bereits produktiver Webanwendungen besonders geeignet sind.

A1.2 Begriffsdefinition WAF – Web Application Firewall

In diesem Dokument verstehen wir unter einer WAF eine Schutzlösung auf Webanwendungsebene, die technisch nicht von der Anwendung selbst abhängig ist. Der Fokus in diesem Dokument liegt auf der Darstellung und Bewertung der Schutzmethoden und -funktionen, die eine WAF bereitstellt. Aspekte der technischen Implementierung in die vorhandene IT-Infrastruktur – ob als Hardware-Appliance, als Software-Plug-In in die Webserver oder auch als Add-On für bereits bestehende Infrastruktur-Komponenten wie z. B. Loadbalancer oder Netzwerk-Firewalls – werden nur gestreift. Insbesondere wird hier – im Unterschied zur Definition bei WAFEC – nicht vorausgesetzt, dass eine WAF als separate Hardware-Appliance vor den Webservern stehen muss – was gerade in großen, schnell wachsenden Infrastrukturen sicher nicht die beste Implementierungsvariante darstellt.

A1.3 Zielgruppe und Zielsetzung

Das Dokument wendet sich vornehmlich an technische Entscheider - speziell Betriebsverantwortliche, Sicherheitsverantwortliche, Applikationseigner (Fachabteilung, technisch Anwendungsverantwortliche), die den Einsatz einer WAF im Unternehmen evaluieren. Spezielles Augenmerk wurde – wo immer möglich – auf die Darstellung von Aufwandsabschätzungen gelegt. Weitere Schwerpunkte sind Best Practices für die organisatorischen Abläufe bei Installation und Betrieb einer WAF sowie – gerade für größere Unternehmen – eine Erweiterung der Prozess-Organisation um die Rolle des *Anwendungsverantwortlicher WAF*.

A2 Charakteristika von Webapplikationen hinsichtlich Web Application Security

A2.1 Übergeordnete Aspekte im Unternehmen

Bezüglich der Bedeutung der Sicherheit der im Unternehmen eingesetzten Webapplikationen müssen – gerade bei größeren Unternehmen – viele Aspekte berücksichtigt werden.

Einer der wichtigsten ist allein die Anzahl der produktiven Webanwendungen im Unternehmen. Groß-Unternehmen betreiben – inhouse oder extern – oft deutlich mehr als 100 Webapplikationen. Auch wenn eine Priorisierung jeder einzelnen Webanwendung hinsichtlich der Bedeutung für den Unternehmenserfolg unerlässlich und sehr sinnvoll ist, so ist doch prinzipiell festzuhalten, dass alle inhouse betriebenen Webanwendungen – je nach Architektur – mittels geeigneter Angriffsmethoden einen Zugriff auf die unternehmensinternen Systeme ermöglichen können. Deshalb sollten auch auf den ersten Blick “unwichtige” Webanwendungen zumindest gegen bekannte Angriffe abgesichert werden.

Bei der Priorisierung hinsichtlich der Bedeutung der Webanwendung für den Unternehmenserfolg spielen insbesondere folgende Punkte eine Hauptrolle:

- Zugriff auf personenbezogene Daten von Kunden, Partnern und/oder Mitarbeitern
- Zugriff auf Betriebsgeheimnisse
- unabdingbare Voraussetzung für die Durchführung betriebskritischer Prozesse
- Relevanz für die Erreichung betriebswichtiger Zertifizierungen.

Mögliche Auswirkungen der Nicht-Verfügbarkeit oder des Datenverlusts der Webanwendungen beinhalten z. B.:

- Unterbrechung betrieblicher Prozesse (auch bei Kunden oder Partnern)
- Imageverlust
- Schadenersatzforderungen
- Entzug von Genehmigungen
- Verlust von Betriebsgeheimnissen.

Andere Aspekte wie Risiken und Kosten siehe A4.3 und A6.4.

A2.2 Technische Aspekte jeder einzelnen Webapplikation des Unternehmens

Die Entscheidung über geeignete Sicherheitsmaßnahmen für eine Webapplikation ist maßgeblich von der jeweiligen Phase der Anwendung im Entwicklungsprozess abhängig. So können in der Designphase noch geeignete Werkzeuge für die Implementierung sowie Test- und Quality-Assurance-Tools ausgewählt werden, ggfs. können auch noch die Entwickler hinsichtlich Web Application Security geschult und die relevanten Zeitschienen bis zur Produktivsetzung verlängert werden.

Für bereits fertig gestellte oder produktive Anwendungen sind hinsichtlich nachträglich möglicher Sicherheitsmaßnahmen ganz andere Punkte von Bedeutung, wie z. B.:

- vollständige Dokumentation der Architektur und des Sourcecodes bzw. Verfügbarkeit der Entwickler der Webapplikation
- Wartungsverträge für alle Komponenten der Anwendungsarchitektur
- kurze Fehlerbehebungszeiten durch den Hersteller von eingesetzten Dritt-Produkten

Nur falls diese Punkte erfüllt sind, kann überhaupt – ohne Betrachtung des entstehenden Aufwands – eine Absicherung der Applikation innerhalb der bestehenden Anwendungsinfrastruktur erfolgen.

A3 Fähigkeiten von Web Application Firewalls (WAFs) im Überblick

A3.1 Einordnung von WAFs im Gesamtbereich Web Application Security

Grundsätzlich gilt, dass jede Webapplikation möglichst sicher entwickelt werden sollte. Denn je später eine Schwachstelle im Lebenszyklus einer Webapplikation erkannt wird, desto höher ist das damit verbundene Risiko eines Einbruchs – und oft auch der Aufwand für die Behebung.

Neben entsprechenden Trainingsmaßnahmen z. B. anhand der OWASP-Richtlinien wird die Applikations-Entwicklung hier durch verschiedene Werkzeuge wirkungsvoll unterstützt. Tools wie z. B. *Stinger* sind in der Regel auf ein Framework – im Beispiel J2EE – bezogen; sie sind Teil der Applikation (auch wenn man sie auch zu fertigen J2EE-konformen Applikationen hinzufügen kann) und damit organisatorisch in der Regel dem normalen Applikation-Releasezyklus unterworfen. Im Kern stellen sie eine wirksame Hilfe für den Entwickler dar, seine – spezielle - Applikation sicherer zu machen. Im Unterschied zu WAFs sind sie aber immer eher Teil der Applikation. Diese Tools werden in diesem Dokument an verschiedenen Stellen, insbesondere beim Vergleich des Aufwands von verschiedenen Schutzmaßnahmen, erwähnt, sie sind aber selbst nicht im Fokus des Dokuments.

In der Entwicklungsphase helfen Methoden z. B. der statischen Sourcecode-Analyse, Schwachstellen des Codes rechtzeitig zu erkennen und zu beseitigen. Hinzu kommen Penetrationstests – am besten durch Experten, die auch im Produktivbetrieb Schwachstellen des externen Verhaltens der Webapplikation aufdecken.

In diesem Zusammenhang ist es die Hauptfunktion einer WAF, Schwachstellen, die Webapplikationen in ihrem externen Verhalten aufweisen, mit möglichst geringem Aufwand so abzusichern, dass sie von Angreifern nicht ausgenutzt werden können. Aufgrund der ohnehin hohen Komplexität einer typischen Anwendungsinfrastruktur – Webserver, Applikationsserver, eingesetzte Frameworks – sowie der typischen Komponenten einer Webapplikation – Session-Handling mit Cookies, Input-Validation etc. - ist dies bereits eine sehr komplexe Aufgabe.

Haupteinsatzziel einer WAF ist deshalb die nachträgliche Absicherung bereits bestehender, oft produktiver Webapplikationen, bei denen notwendige Änderungen innerhalb der Applikation grundsätzlich nicht mehr oder nur noch mit unverhältnismäßig hohem Aufwand erfolgen kann. Dies gilt besonders für solche Schwachstellen, die z. B. durch einen Penetrationstest oder auch durch eine Analyse des Source Codes aufgedeckt wurden und – insbesondere kurzfristig – nicht innerhalb der Applikation gefixt werden können. Grundvoraussetzung der WAF ist hierbei, dass sie – über einen Grundschutz durch ein Blacklisting, also der Beschreibung bekannter Angriffsmuster hinaus – die Möglichkeit eines – vernünftig zu bedienenden – Whitelistings anbietet. Beim Whitelisting beschreibt das Regelwerk der WAF genau von der Anwendung gewünschte Verhalten; die Konfiguration geeigneter Whitelists wird oft durch Lernmodi unterstützt.

Darüber hinaus bieten einige WAFs auch Funktionalitäten an, die über den reinen Absicherungs-Charakter hinausgehen und deshalb auch schon im Designprozess berücksichtigt werden können, um unnötigen Aufwand zu vermeiden. Die WAF wird so zur zentralen Servicestelle für die Erledigung von Aufgaben werden, die naturgemäß auf Seiten der Anwendung liegen, aber für alle Anwendungen auf gleiche Weise zu lösen sind. Beispiele hierfür sind das sichere Session-Management für alle Anwendungen auf der Basis von Cookie-Stores, zentrale Authentisierung und Autorisierung, die Zusammenführung aller relevanten Fehlermeldungen und Logfiles oder die Möglichkeit proaktiver Schutzmechanismen wie z.B. URL-Encryption.

Die folgende Tabelle zeigt anhand der wichtigsten derzeit bekannten Schwachstellen bzw. Angriffsmethoden von Webapplikationen, welchen Schutz WAFs hier bieten. Dabei werden in der Regel die genutzten Funktionen der WAF genannt, wobei nicht unbedingt alle am Markt angebotenen WAFs über alle genannten Funktionen verfügen.

A3.2 Typische Schutzmechanismen von WAFs am Beispiel konkreter Schwachstellen

In der folgenden Tabelle sind für typische Bedrohungen, Schwachstellen und Angriffe (Spalte Problem) die möglichen Schutzmechanismen (Spalte Maßnahme) aufgeführt, und in der Spalte WAF bewertet wie gut eine WAF die Anwendung schützen kann. Die Bewertungen sind:

- **+** kann eine WAF sehr gut
- **-** kann die WAF schlecht oder gar nicht
- **!** abhängig von WAF/Anwendung/Anforderungen
- **=** kann teilweise von einer WAF übernommen

Problem	WAF	Maßnahme
Cookieschutz	+	Cookies können signiert werden
	+	Cookies können verschlüsselt werden
	!	Cookies können vollständig versteckt bzw. ausgetauscht werden (Cookie-Store)
	!	Cookies können an die anfragende IP gebunden werden
Information-Leakage	+	Cloaking-Filter, ausgehende Seiten können "bereinigt" werden (Fehlermeldungen, Kommentare, unerwünschte Informationen)
Session-Riding (CSRF)	+	URL-Encryption / Token
Session-Timeout	!	Timeout für aktive und inaktive (idle) Sessions kann festgelegt werden (wenn die WAF die Sessions selbst verwaltet) Auch wenn die Sessions von der Applikation bereitgestellt werden, kann die WAF diese bei entsprechender Konfiguration erkennen und terminieren.
Session-Fixation	=	kann verhindert werden, wenn die WAF die Sessions selbst verwaltet
Session-Hijacking	-	Schutz nur schwer möglich, WAF kann aber bei Unregelmäßigkeiten (z. B. wechselnde IP) alarmieren oder bei wechselnder IP die Session terminieren
File-Upload	+	Virenprüfung (meist durch externe Systeme) via ICAP an die WAF angebunden
Parameter-Tampering	+	zusätzlich/anstatt Data-Validation (siehe unten) kann Parameter-Manipulation durch URL-Encryption (GET) und Parameter-Encryption (GET- und POST) verhindert werden
	+	Site-Usage-Enforcement, d. h. Reihenfolge der URLs kann festgelegt oder erkannt werden
Forced-Browsing	+	kann durch URL-Encryption verhindert werden
	+	Site-Usage-Enforcement
Path-Traversal (URL) Link-Validation	+	kann durch URL-Encryption verhindert werden
	+	Site-Usage-Enforcement
Path-Traversal (Parameter), Path.Manipulation	+	siehe Parameter-Tampering und Data-Validation
Logging	+	alle oder nur bestimmte/erlaubte Teile der Daten eines Requests und der damit verbundenen Prüfungen können protokolliert werden
Priv. Escalation	-	Privilege-Escalation kann nicht, bzw. nur bedingt durch z. B. Cookie-/Parameter-Encryption geprüft werden

Logische Ebene	-	Anwendungslogik, die über die Gültigkeit von URLs und Formfeldern hinausgeht, kann durch eine WAF i. d. R. nicht geprüft werden
Anti-Automation	=	automatische Angriffe können teilweise erkannt und geblockt werden (z. B. Anzahl Requests/Zeitintervall, identische Requests, usw.)
Applikations-DoS (moderat)	= =	Transaktionen, IP, User blocken Verbindungen, Sessions können beendet werden
SSL	+ + +	WAF kann SSL mit vorgegebener Verschlüsselungsstärke erzwingen (je nach Infrastruktur-Szenario) SSL-Terminierung an der WAF, Weitergabe der SSL-Daten (z. B. Client-Zertifikat) an Anwendung SSL-Verbindung von WAF zur Anwendung möglich
Data-Validation (bezogen auf Feld/Inhalt/Kontext/Appl)	+ + ! - 	kann sehr feingranular (Länge, konstanter Wert/Wertebereich z. B. bei SELECT, Zeichenbereich) geprüft werden; Validierung mit Whitelist und/oder Blacklist (Signatur) möglich Regeln können teilweise automatisch erstellt werden teilweise hohe Abhängigkeit zur Anwendung, bestimmte Felder (hidden Form) bzw. vorgegebene Parameter in der URL können aber von der WAF automatisch verifiziert werden Gefahr durch *false positives*, insbesondere bei geschäftskritische Anwendungen problematisch
Data-Validation (generell/global)	+	HTTP(w3c)-Konformität, eine WAF führt eine Kanonalisierung der Daten durch, so dass sie der Anwendung in normalisierter Form vorliegen
Buffer-Overflow	+	siehe Data-Validation [1]
Format-String-Attack	=	kann mittels Data-Validation erkannt werden, wenn entsprechende Zeichen oder Strings gefiltert werden (in der Praxis schwierig, da dazu genaue Kenntnis der Funktionsweise der Anwendung erforderlich ist) Für einen Großteil der Hidden Input Felder kann dies auch ohne Kenntnisse der Anwendung geschehen
Cross-Site-Scripting	=	mittels Data-Validation kann nur *reflected XSS* erkannt und verhindert werden, *persistent XSS* kann nicht erkannt werden, *DOM-based XSS* nur bedingt, wenn ein Teil des Angriffs in Parametern des Requests transportiert wird
Cross-Site-Tracing	+	Beschränkung der HTTP-Methode auf z. B. GET, POST
WebDAV	+	Beschränkung auf z. B. nur lesende WebDAV-Methoden
Code-Injection (PHP, perl, java)	+	siehe Data-Validation [1]
Command-Injection	+	siehe Data-Validation [1]
SQL-Injection	+	siehe Data-Validation [1]
LDAP-Injection	+	siehe Data-Validation [1]
XML-/XPath-Injection	+	siehe Data-Validation [1]
just-in-time-Patching (Hotfix-Patching)	+	mittels Data-Validation (siehe oben) können neu erkannte Schwachstellen und/oder Angriffe (*Zero Day Exploit*) in der WAF abgewehrt werden
HTTP-Response-Splitting (HTTP-Splitting)	!	kann mittels Data-Validation in URL und/oder Parametern nur dann erkannt werden, wenn *%0d%0a* gefiltert wird – dies

		kann aber auf fast allen Eingabefeldern gemacht werden, ohne die Funktionalität der Applikation zu beeinträchtigen
HTTP-Request-Smuggling[1]	+	wird durch strenge Prüfung der Standardkonformität jedes Requests verhindert

[1] Grundschutz mit Blacklisting meist ausreichend, weitere Möglichkeiten durch Kombination von Blacklisting und Whitelisting

A4 Nutzen und Risiken von Web Application Firewalls im Überblick

Die hier dargestellten Nutzenpotenziale von WAFs werden insbesondere anhand des detaillierten Überblicks im nächsten Kapitel ausführlich begründet. Dieses Kapitel dient vornehmlich der Zusammenfassung für Entscheider, die das folgende Kapitel nur im Überblick durcharbeiten wollen.

A4.1 Hauptnutzen von WAFs

Der Hauptnutzen von WAFs liegt in der nachträglichen Absicherung des externen Verhaltens von bereits fertig gestellten, produktiven Webapplikationen mit vertretbarem Aufwand und ohne Änderung der Applikation selbst.

Dies gilt einerseits mit einer Art *Grundschutz* gegen bekannte Angriffsmethoden bzw. Schwachstellen auf der Basis von Blacklists: So sieht z. B. der Datensicherheitsstandard der Kreditkartenindustrie (PCI DSS v.1.1) in seiner derzeit gültigen Version den Einsatz einer WAF – alternativ zu regelmäßigen Code-Reviews durch Spezialisten – als ausreichende Maßnahme zum Schutz der verwendeten Webapplikationen vor. Die WAF ist also hier ein geeignetes Werkzeug zur Einhaltung von Industrie-Standards bzw. auch von rechtlichen Vorgaben.

Besonders deutlich wird der Nutzen von WAFs im Falle von z. B. durch Penetrationstests oder Sourcecode-Reviews entdeckter konkreter Schwachstellen. Selbst wenn die Schwachstelle in der Applikation zeitnah und mit vertretbarem Aufwand gefixt werde könnte, so kann die geänderte Version meist erst im nächsten Wartungsintervall, oft 2-4 Wochen später, eingespielt werden (Patch-Dilemma). Bei einer WAF mit Whitelisting kann hier zeitnah die Schwachstelle nach außen gefixt werden (Hotfix), so dass sie bis zur nächsten planmäßigen Wartung nicht ausgenutzt werden kann. Besonders schnell sind hier WAFs, die z. B. mit Source Code Analyse-Tools so zusammenarbeiten können, dass erkannte Schwachstellen automatisch zu Regelvorschlägen für die WAF führen.

Besondere Bedeutung kommt einer WAF im Falle von produktiven Webapplikationen zu, die selbst wiederum aus vielen Komponenten bestehen, die durch den Betreiber nicht rasch geändert werden können – z. B. im Falle von schlecht dokumentierten Anwendungen oder eingesetzten Dritt-Produkten ohne hinreichende Wartungszyklen. Hier ist eine WAF die einzige Möglichkeit, zeitnah Schwachstellen „nach außen“ abfangen zu können.

A4.2 Zusatznutzen von WAFs – abhängig von deren Funktionalitäten

Weitere erhebliche Nutzenpotentiale sind in der zentralen Rolle der WAF begründet. Die Fehlersuche wird erheblich erleichtert, wenn bei vielen Anwendungen nicht jede einzelne separat Fehlermeldungen generiert, sondern diese zentral an der WAF zur Verfügung stehen und dort ausgewertet werden können. Gleiches gilt z. B. für alle Aspekte des Monitoring und Reporting. Als zentrale Servicestelle kann die WAF Aufgaben übernehmen, die für jede Anwendung auf gleiche Weise zu lösen sind. Ein gutes Beispiel hierfür ist das sichere Session-Management für alle Anwendungen auf der Basis von Cookie Stores.

Viele WAFs bieten auch proaktive Schutzmechanismen wie z. B. URL-Encryption oder Site-Usage-Enforcement, um die Angriffsfläche mit geringem Aufwand möglichst zu minimieren. Zudem erhöht der Einsatz einer WAF die Robustheit der Webanwendungen nach außen.

Je nach Implementierungsart bieten WAFs auch weiteren Zusatznutzen. Eine Hardware-Appliance vor den Webservern kann oft SSL-Verbindungen terminieren und verfügt auch über Loadbalancer-Fähigkeiten. Dies ist manchmal im Unternehmen wünschenswert, kann aber auch durch entsprechende Web-Application-Security-Add-Ons bereits eingesetzter Produkte geleistet werden. In Hochsicherheitsumgebungen verbieten jedoch häufig die vorhandenen

Sicherheitsrichtlinien die Terminierung von SSL-Verbindungen vor den Webservern. Hier sind dann WAFs, die als Plug-In in den Webserver implementiert werden, besonders geeignet.

Die WAF kann auch eine SSL-Terminierung bereitstellen, wenn die zu schützende Anwendung bzw. deren Web- oder Applikationsserver diese Fähigkeit nicht hat.

A4.3 Risiken beim Einsatz von WAFs

Beim Einsatz einer WAF ist zu beachten, dass ein Eingriff in die vorhandene IT-, Web- und evtl. Applikations-Infrastruktur erforderlich ist. Je nach Ausführung der WAF – z. B. Hardware-Appliance vs. Embedded WAF – zeichnen sich auch Mehraufwände und Risiken ab:

- yet-another-proxy-Argument (erhöhte Komplexität der IT-Infrastruktur)
- Organisatorische Aufwände (siehe A8.2 Rollenmodell beim Betrieb von WAFs)
- Trainingsaufwand
    - Bei jedem Releasewechsel der Webanwendung
    - Testaufwände
- False Positives (mit u. U. hohem Business-Impact)
- Komplexere Fehlersuche
    - Auch WAFs haben/machen Fehler
    - Zuständigkeit, Verantwortlichkeit bei systemübergreifenden Fehlersituationen
- Evtl. Einfluss auf die Webanwendung wenn die WAF z. B. die Anwendungssession terminiert
- Wirtschaftlichkeit

A5 Schutz gegen OWASP-TOP10 – WAFs und andere Methoden im Vergleich

Dieses Kapitel befasst sich mit den verschiedenen Schutzmöglichkeiten für die sogenannten *OWASP-Top10-Schwachstellen*. Dabei werden exemplarisch drei verschiedene Webanwendungsklassen zugrunde gelegt:

- T1: eine Webanwendung in der Design-Phase, neue Anwendung
- T2: eine bereits produktive Anwendung (z. B. mit MVC-Architektur), die einfach anpassbar ist
- T3: eine produktive Anwendung, die nicht oder nur sehr schwer anpassbar ist

Am Beispiel dieser drei Klassen werden Schutzmaßnahmen innerhalb der Applikation bzw. der Applikations-Architektur selbst, unter Einsatz von WAFs sowie durch Vorgaben einer entsprechenden Policy detailliert dargestellt sowie hinsichtlich des hierfür nötigen Aufwands bewertet. An einigen Stellen finden sich Hinweise auf spezielle Funktionalitäten von WAFs bzw. Annahmen über die verwendete Applikations-Infrastruktur, da diese nicht allgemein zutreffen.

Wie die unten aufgeführte Tabelle deutlich zeigt, ist der Einsatz von WAFs gerade im Falle bereits produktiver Anwendungen sehr häufig mit dem geringsten Aufwand verbunden. Im Falle von bereits produktiven Anwendungen, die nicht oder nur schwer anpassbar sind, ist der Einsatz von WAFs in einigen Punkten sogar die einzig mögliche Schutzmaßnahme.

In der folgenden Tabelle sind für die Bedrohungen (Spalte Top 10) Kommentare und Hinweise (Spalte Kommentar) für die jeweiligen Anwendungsklassen T1, T2, T3, den Einsatz einer WAF oder durch Umsetzung einer Policy (Spalte Typ) aufgeführt, und bewertet (Spalte Aufwand) wie groß der für eine sichere Implementierung notwendige Aufwand jeweils ist. Die Bewertungen sind:

- **1** mit geringem Aufwand
- **2** mit mittlerem Aufwand
- **3** mit hohem Aufwand)
- **-** gewöhnlich nicht umsetzbar

Top10		Typ	Kommentar	Aufwand
A1	Cross-Site Scripting (XSS)	T1	z. B. durch konsequenten Taglib-Einsatz (Java), oder Controls (ASP.NET), zusätzlicher Frameworks (PHPIDS).	1
		T2	Eingabe-Enkodierung lässt sich nur schwer (z. B. mittels OWASP Stinger) einbauen, besser geht es hier mit einer vorgelagerten WAF. Bei .NET-Anwendungen lässt sich XSS-Filter aktivieren.	3 (.NET: 2)
		T3	Bei .NET-Anwendungen XSS-Filter aktivieren.	- (.NET: 2)
		WAF	WAF ermöglicht in diesem Fall keine Ausgabevalidierung, da sie den Kontext der Daten nicht kennt. Die Validierung muss schon bei der Eingabe erfolgen und kann eventuell mit der Ausgabe korreliert werden	2
		P		-
A2	Injection Flaws	T1	Kann durch Verwendung eines OR-Mappers (z. B. Hibernate) oder konsequente Parametrisierung aller Eingaben (z. B. Stored Procedures oder besser: Prepared Statements) vermieden werden. Andere Injection Flaws (z. B. bei XML) lassen sich ggf. nur	1

			mit dedizierter Ausgabenkodierung vermeiden.	
		T2	Aufwendig, da programmatische Anpassungen notwendig.	3
		T3	-	-
		WAF	WAF mit Blacklisting: kann hier prinzipiell nur nach bestimmten Zeichen oder Zeichenketten suchen und die Verarbeitung vermeiden. Probleme bestehen bei diesem Ansatz grundsätzlich im Abdeckungsgrad sowie bei möglichen Filter-Evasion-Angriffen (z. B. bei Mehrfachkodierung) falls keine Eingabenormalisierung stattfindet. Das funktioniert sehr gut bei bekannten Angriffen (z. B. SQL-Injection), aber sicherlich weniger gut bei der WAF unbekannten bzw. proprietären Protokollen. Zusätzlich können durch URL Encryption und Hidden-Form-Parameter-Protection Injection-Angriffe auf einen Teil der Eingabedaten wirksam verhindert werden. Ein Beispiel hierfür sind Artikelnummern in einem Webshop, die klassischerweise gerne für SQL-Injection Angriffe benutzt werden, jedoch eigentlich nie für den Benutzer direkt manipulierbar sein müssen. WAF mit Whitelisting: Für alle anderen Eingabefelder bietet sich ein Whitelist-Ansatz an. Hier kann die WAF nach einer Lernphase Vorschläge für die einzelnen Felder vorgeben. Damit lassen sich zwar nicht alle, aber ein Großteil der Eingabefelder gegen alle Arten von Injection-Angriffen absichern.	2
		P	Im Falle von SQL-Injection: Vorgaben für Datenbankzugriffsrechte, sonst wenig bis keine Möglichkeit.	-
A3	Malicious File Execution	T1	Einbinden von Uploadscannern bzw. Whitelisting der erlaubten remote inclusions,	2
		T2		3
		T3	-	-
		WAF	Whitelisting der Parameter für das erlaubte Einbinden von systemfremden URL - Einbinden von Upload Scannern via ICAP Protokoll - Responseanalyse, um das Anzeigen kritischer Daten (teilweise auch Fehlermeldungen) zu verhindern.	1-2
		P	Vorgaben für Deploymentplattform, Vorgaben für Zugriffsrechte	2
A4	Insecure Direct Object Reference	T1	Implementierung einer Objekt-Virtualisierung ist sehr aufwendig, da Datenbank-Objekte häufig von verwendeten Frameworks auf Parameter abgebildet werden (OR-Mapper). Schutz erfordert zahlreiche programmatische Prüfungen.	3
		T2	Verhinderung von ID-Manipulation macht meist Code-Anpassungen notwendig. Schutz erfordert zahlreiche programmatische Prüfungen.	3

		T3	-	-
		WAF	Schutz vor ID-Manipulation mittels ID-Virtualisierung bzw. hidden Parameter-Protection.	1
		P	Verwenden von Impersonifikation und Delegation.	3
A5	Cross-Site Request Forgery (CSRF)	T1	Mit spezifischer Anwendungsarchitektur lösbar.	1
		T2	Hoher Aufwand. Meistens programmatische Änderungen erforderlich.	3
		T3	-	-
		WAF	Kann mittels Page-Token oder URL-Encryption unterbunden werden.	1
		P		-
A6.1	Information Leakage	T1	Toolgestütztes Testen mit hoher Testabdeckung und entsprechendem Fokus.	2
		T2	Toolgestütztes Testen mit hoher Testabdeckung und entsprechendem Fokus.	2
		T3	-	-
		WAF	Automatisches Ausfiltern von Kommentaren möglich. Site Usage Enforcement kann den Zugriff auf vorhandene aber nichtveröffentlichte (nicht verlinkte) Dokumente verhindern. Klassisches Beispiel sind Backup Files auf dem Webserver, die Datenbankpasswörter im Klartext enthalten und deren URL vom Angreifer geraten wird.	1-2
		P	Vorgaben für Programmierer und Autoren, keine Kommentare einzutragen. Vorgaben für die Gestaltung von Fehlermeldungen.	2
A6.2	Improper Error Handling	T1	Je nach Plattform deklarativ konfigurierbar.	1
		T2	Je nach Plattform deklarativ konfigurierbar.	1
		T3	Je nach Plattform deklarativ konfigurierbar.	1 / -
		WAF	Schwer zu erkennen.	2
		P	-	-
A7.1	Broken Authentication	T1	Anbindung an ein zentrales Access Management System mit angemessenen Sicherheitsstandards	1
		T2	Anbindung an ein zentrales Access Management System mit angemessenen Sicherheitsstandards. Evtl. programmatische Anpassungen erforderlich.	2
		T3	-	-
		WAF	Abhängig von Fähigkeiten der WAF. WAF kann Authentisierung unabhängig von der Applikation vornehmen und damit ohne Änderung der Applikation eine Anbindung an eine zentrale Authentisierungsinfrastruktur ermöglichen.	2
		P	Vorgaben hinsichtlich Passwortkomplexität.	2
A7.2	Session Management	T1	Auf Designebene z. B. mittels Session-Manager-Design-Pattern, sonst zahlreiche Möglichkeiten. Implementierungsaufwand teilweise abhängig von Applikationsserver, siehe auch A7.1, wenn das Session-Management vom Access-Management-System übernommen wird.	2

		T2	größtenteils zentral einbaubar (mittels Filter, Listener oder gehärteter Serverkonfiguration); dennoch teilweise großer Implementierungs- und Testaufwand; siehe auch A7.1, wenn das Session Management vom Access Management System übernommen wird.	2-3
		T3	Abhängig vom Applikationsserver, teilweise konfigurierbar.	-
		WAF	Härtung eines unsicheren Session-Managements über verschiedene Techniken möglich (z. B. Page-Tokens).	1
		P	-	-
A8	Insecure Cryptographic Storage	T1	Verwendung von Crypto-APIs.	1
		T2	Verwendung von Crypto-APIs. Nachträgliche Implementierung erfordert zahlreiche programmatische Anpassungen.	3
		T3	-	-
		WAF	-	-
		P	Vorgaben zur Speicherung von sensiblen Daten.	-
A9	Insecure Communiations	T1	Deklarativ im Applikations- bzw. Webserver konfigurierbar.	1
		T2	Deklarativ im Applikations- bzw. Webserver konfigurierbar. Sehr hoher Aufwand wenn URL-Schema (HTTP) hardcoded wurde.	1 / -
		T3	Deklarativ im Applikations- bzw. Webserver konfigurierbar (wenn Zugang besteht). Nicht möglich, wenn URL-Schema (HTTP) hardcoded wurde.	1 / -
		WAF	Kann HTTP-Anwendungen mit HTTPS absichern.	1
		P	-	-
A10	Failure to Restrict URL Access	T1	Verwendung eines Front Controllers mit Gateway. Code muss dennoch an verschiedenen Stellen (z.B. im Service) Benutzerzugehörigkeit programmatisch prüfen. Lücken möglich.	1-2
		T2	Je nach Anwendung unterschiedlich. URL-Zugriffsrechte bei J2EE und .NET deklarativ konfigurierbar. Verhinderung von ID-Manipulation macht meist Code-Anpassungen notwendig.	2-3
		T3	Je nach Anwendung unterschiedlich. URL-Zugriffsrechte bei J2EE und .NET deklerativ konfigurierbar.	3
		WAF	Benutzer kann mittels Page-Token oder URL-Encryption auf Seiten eingeschränkt werden, die er von der Anwendung als Link erhält. Die Anwendung darf hierzu allerdings geschützte Links nicht anzeigen (Limited-Access-Pattern). Mit Site-Usage-Enforcement kann der Benutzer nur auf verlinkten Content gelangen. Auch können bestimmte URLs/Teilbäume durch Whitelist/Blacklist Ansätze ausgeschlossen werden (z. B. allow access nur für *.html, *.php, *.gif, *.jpg – aber nicht für *.bak oder andere Extensions).	1

		P	-	-

A6 Kriterien zur Einsatz-Entscheidung von WAFs

A6.1 Unternehmensweite Kriterien

Kernkriterien in diesem Bereich sind:

- Bedeutung der Webapplikation(en) für den Unternehmenserfolg (Umsatzanteil, Image)
- Bedeutung des Verlusts der Daten der Webapplikation (Kundendaten, Unternehmens-geheimnisse, Image)
- Anzahl der Webapplikationen
- rechtliche Rahmenbedingungen bzw. Industriestandards
- Komplexität
- Betriebsaufwände
- Performance
- Skalierbarkeit

A6.2 Kriterien hinsichtlich einer Webapplikation

Im Folgenden wird der Begriff des *Zugriffs* des Unternehmens auf die Webapplikation eingeführt und erläutert. Die Checkliste im Anhang A8.1 dient dazu, für jede Webapplikation den Grad des Zugriffs anhand eines Punktesystems individuell zu bestimmen.

Der *Zugriff des Unternehmens auf eine Webapplikation* soll ein Maß dafür sein, inwieweit das Unternehmen notwendige Änderungen der Webapplikation – also letztlich des Sourcecodes der Applikation – zeitnah selbst durchführen bzw. veranlassen und durchsetzen kann.

Eine Webapplikation in der Design-Phase (siehe T1 in A5) kann hier als Spezialfall einer *Webapplikation mit optimalem Zugriff* betrachtet werden.

Das andere Extrem einer *Webapplikation ohne Zugriff* ist beispielsweise eine Applikation, die aus vielen nicht dokumentierten Komponenten besteht, deren Entwickler ebenfalls nicht mehr kontaktiert werden können und die Dritt-Softwareprodukte verwendet, die seitens des Herstellers – bei Open-Source-Projekten seitens der Community – nicht mehr gepflegt werden (siehe T3 in A5).

Wichtige Kriterien, um das Maß des Zugriffs auf eine Webapplikation zu bestimmen, sind:

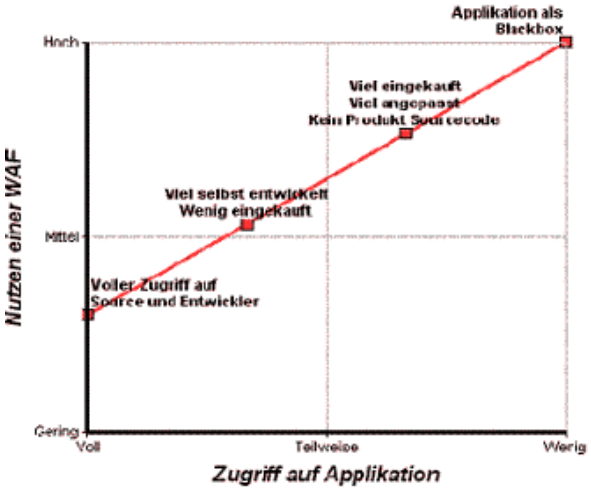
- vollständige Dokumentation der Architektur und des Sourcecodes bzw. Verfügbarkeit auf Entwickler der Webapplikation
- Wartungsverträge für alle Komponenten der Anwendungsarchitektur
- kurze Fehlerbehebungszeiten durch den Hersteller von eingesetzten Dritt-Produkten (Portale, Frameworks, SAP, usw.).

Weitere wesentliche Kriterien pro Webapplikation sind in der genannten Checkliste im Anhang aufgeführt.

A6.3 Bewertung und Zusammenfassung

Mithilfe der Checkliste im Anhang A8.1 kann der Grad des Zugriffs für jede Webapplikation ermittelt werden. Daraus lässt sich auch ein Mittelwert des Zugriffs für alle Webapplikationen eines Unternehmens ermitteln, wobei allerdings zu beachten ist, dass für den Unternehmenserfolg oder das Image besonders wichtige Applikationen auch entsprechend hoch gewichtet werden müssen.

Als Leitfaden bei der Entscheidungsfindung für den Nutzen des Einsatzes einer WAF im Unternehmen kann dabei folgende Darstellung hilfreich sein:



Hat ein Unternehmen alle Webapplikationen im vollen Zugriff, so bringt der Einsatz einer WAF vor allem Aufwandsminimierungen im Betrieb – insbesondere aufgrund der in A3 genannten Zusatznutzen einer WAF als zentralen Servicestelle sowie einiger vergleichsweise einfach zu implementierender Schutzmechanismen, siehe A4.

Ist auf die Webapplikationen praktisch kein Zugriff vorhanden, so ist der Einsatz einer WAF unbedingt sinnvoll, da nur so geeignete Schutzmaßnahmen implementiert werden können.

Mit abnehmendem Zugriff auf die Webapplikation – und abhängig von ihrer Bedeutung und Komplexität – wächst der Nutzen aus dem Einsatz einer WAF sehr rasch – von einer *Second Line of Defense* zu einer echten *Vollversiegelung* der Webapplikation nach außen mittels Whitelisting – da eine WAF im Vergleich oft den geringsten Zusatzaufwand für die notwendige Absicherung verursacht.

A6.4 Wirtschaftlichkeitsbetrachtung

Die Wirtschaftlichkeit der Anschaffung und des Betriebs einer WAF kann aus mehreren Blickwinkeln betrachtet werden:

- Vermeidung von wirtschaftlichen Schäden durch erfolgreiche Angriffe auf Webapplikation
- Geringere Kosten für die als notwendig erkannte Absicherung von Webapplikation gegenüber anderen Optionen
- Einsparungen durch die Nutzungen von zentralen Diensten die von einer WAF für mehrere Webanwendungen zur Verfügung gestellt werden und somit nicht mehr in jeder Anwendung selbst implementiert bzw. konfiguriert werden müssen.

Bei der Absicherung von Anwendungen, auf die kein ausreichender Zugriff besteht (siehe A6.2), deren Schutz aber trotzdem als notwendig erachtet wird können die Kosten für den Einsatz einer WAF entweder als strategische Investition betrachtet, oder, sofern realitätsnah, gegen die Kosten einer Ablösung der fraglichen Anwendung gerechnet werden.

Die Kosten für den Einsatz einer WAF setzen sich in der Regel aus den folgenden Komponenten zusammen:

- Lizenzkosten
- Lizenzwartung
- Projektkosten für die Evaluierung und Einführung einer WAF
- (anteilige) Kosten für den Betrieb der notwenigen Plattform

- Personalkosten für den oder die Anwendungsverantwortlichen WAF
- Aufwände in Projekten für die Abstimmung mit dem Anwendungsverantwortlichen WAF.

A7 Best Practices bei Einführung und Betrieb von WAFs

A7.1 Aspekte der vorhandenen Web-Infrastruktur

7.1.1 Zentrale oder dezentrale Infrastruktur – absehbare Veränderungen

Grundsätzlich ist festzuhalten, dass sich WAFs in die bestehende Web-Infrastruktur – und deren geplante oder absehbare Veränderungen – einfügen müssen – und nicht umgekehrt die Infrastruktur wegen der Implementierung einer WAF grundlegend geändert werden sollte.

Entsprechend kann eine WAF in einer zentralen Infrastruktur, die sich absehbar nicht wesentlich ändert, auch als zentrale Infrastruktur-Komponente z. B. als Hardware-Appliance installiert werden, während bei einer ohnehin bereits dezentralen, möglicherweise schnell wachsenden Infrastruktur – z. B. bei einem großen Online-Shop – ein verteilter WAF-Ansatz, z. B. als Plug-In in die vorhandenen Webserver – besser passend ist. Hinsichtlich der Infrastruktur-Aspekte besonders flexibel sind WAF-Produkte, die einen grundsätzlich verteilten Implementationsansatz mit einer zentralen Administration verbinden und so die Vorteile aus beiden Szenarien verbinden.

Erwähnenswert – und mit Blick auf zukünftige Entwicklungen voraussichtlich immer wichtiger – ist auch die Möglichkeit der dank Virtualisierung gesharten Infrastrukturen. Gerade hier ist bei der Auswahl der WAF zu berücksichtigen, dass auch die WAF sich nahtlos in diesen virtualisierten Ansatz mit einschließen lässt.

7.1.2 Performance-Kriterien

Hinsichtlich der technischen Performance ist darauf zu achten, dass die angestrebte WAF-Infrastruktur die wesentlichen Performance-Kennzahlen der vorhandenen Web-Infrastruktur unterstützt. Reine Aussagen bezüglich des GB-Durchsatzes von Hardware sind hier nicht unbedingt zielführend, da in der Praxis oftmals nicht erreichbar. Wichtiger sind typische Kennzahlen einer Webapplikation wie Anzahl der gleichzeitigen Benutzer der Applikation und daraus abgeleitet die Anzahl der HTTP-Requests pro Zeiteinheit im Mittel und in Spitzenlastzeiten. Hierbei ist zu beachten, dass viele Applikationen Hochlastphasen haben, die nur selten auftreten, z. B. während des Weihnachtsgeschäfts im Falle eines Webshops.

A7.2 Organisatorische Aspekte

7.2.1 Einhaltung bestehender Security Policies

Soweit möglich sollten bestehende Security Policies durch die Implementierung einer WAF nicht geändert werden müssen.

Als typisches Beispiel ist hier die SSL-Terminierung “vor den Webservern” zu nennen. Diese ist, insbesondere in Hochsicherheitsinfrastrukturen, oft durch die bestehenden Sicherheitsrichtlinien ausgeschlossen und kann auch durch den Einsatz einer geeigneten WAF – als Plug-In in den Webserver hinter der im Webserver ohnehin vorgenommenen SSL-Terminierung – aufrecht erhalten werden.

7.2.2 Neues Rollenmodell: Anwendungsverantwortlicher WAF

Der nachhaltig erfolgreiche Einsatz von WAFs hängt entscheidend davon ab, dass auch nach dem Einmalaufwand der Inbetriebnahme das Zusammenspiel der WAF mit allen weiteren Komponenten der Anwendungsinfrastruktur reibungslos funktioniert. Dazu zählen einerseits offensichtliche Punkte wie Verständnis und passende Reaktionsmöglichkeiten auf Fehler- und Alarmmeldungen der WAF, aber auch Aspekte wie die Anpassung des Regelwerks der WAF bei gewünschten Änderungen der zu schützenden Applikationen. Um die Chancen, die eine WAF als

zentrale Servicestelle für z. B. Sicheres Session-Management etc. bietet, voll nutzen zu können, ist zudem eine gute Zusammenarbeit mit der Anwendungsentwicklung notwendig.

Mit anderen Worten: Um das volle Potential einer WAF zu nutzen, genügt es nicht, die WAF als reine Infrastrukturkomponente zu betrachten.

Deshalb schlagen wir hier – neben der Rolle eines *Plattformverantwortlichen WAF*, die ähnlich eines *Plattformverantwortlichen Netzwerk-Firewall* für die infrastrukturellen Aspekte der WAF verantwortlich ist – die neue Rolle eines *Anwendungsverantwortlichen WAF* für jede Anwendung vor, der, bildlich gesprochen, die Brücke zwischen der WAF und der fachlichen Anwendung darstellt. Er muss exzellente Kenntnisse der WAF besitzen, um diese konfigurieren und für die spezielle Anwendung überwachen zu können. Er muss die Anwendung gut kennen, um Meldungen der WAF einordnen und bewerten zu können. In der Regel wird ein *Anwendungsverantwortlicher WAF* die WAF-Konfiguration für mehrere Anwendungen betreuen. Ein Beispiel wäre die Betreuung der WAF für alle webbasierten SAP-Systeme, während das Shopsystem von einem anderen Anwendungsverantwortlichen WAF betreut wird.

Eine ausführliche Beschreibung des vorgeschlagenen Rollenmodells findet sich in Anhang A8.3.

A7.3 Iteratives Vorgehen bei der Implementierung – vom Grundschutz zur Vollversiegelung

Als Best Practice bei der Implementierung und beim Betrieb von WAFs hat sich ein iteratives Vorgehen bewährt.

7.3.1 Schritt 1: Festlegung der Rollenverteilung / Einbindung der Anwendungsentwicklung

Zunächst müssen die Verantwortlichkeiten – idealerweise auf Basis des oben vorgestellten Rollenkonzepts – definiert werden. Erfolgt die Webapplikationsentwicklung inhouse, so ist diese möglichst frühzeitig in den Prozess mit einzubinden. Damit wird erreicht, dass möglichst bald alle noch nicht produktiv gesetzten Applikationen die zentralen Funktionen der WAF nutzen, was die Sicherheit erhöht und Zeit und Geld spart. Außerdem werden mögliche Hürden auf persönlicher Ebene frühzeitig abgebaut.

7.3.2 Schritt 2: Grundschutz für alle Webapplikationen

Unabhängig von den Charakteristika der jeweiligen Webapplikation wird zunächst ein *Grundschutz*, in der Regel als Blacklisting umgesetzt, aktiviert. Erste Auswertungen zeigen in der Regel erste erfolgreiche Abwehrmaßnahmen, oder zeigen False Positives – d. h. zu streng eingestellte Regeln –, gleichzeitig werden die organisatorischen Abläufe eingeübt.

7.3.3 Schritt 3: Erstellung einer Prioritätenliste aller vorhandenen Webapplikationen

Grundlage dieser Prioritätenliste kann – neben den übergeordneten Kriterien wie Image-Verlust usw. - das Maß des Zugriffes auf die Webapplikation gemäß Checkliste im Anhang A8.1 sein.

7.3.4 weitere Schritte: Vollversiegelung der Webapplikationen gemäß Prioritätenliste

In der Regel unterstützt durch Lernmodi der WAF oder Sourcecode-Review/Penetrationsstest, werden die Webapplikationen Schritt für Schritt gemäß der Prioritätenliste mit Whitelist-Regelwerken nach außen *vollversiegelt*. Der *Anwendungsverantwortliche WAF* sorgt hier in Zusammenarbeit mit dem fachlichen Anwendungsverantwortlichen für die jederzeit reibungslose Verfügbarkeit der Anwendung nach außen – auch während der Regelwerksumstellung.

A8 Anhänge

A8.1 Checkliste: Zugriff auf eine Webapplikation unter Security-Gesichtspunkten

Zur Bewertung des *Zugriffs* den ein Unternehmen auf die Webapplikation hat, kann folgende Checkliste dienen. Je mehr Punkte aufaddiert werden, umso besser ist der Zugriff auf eine Webapplikation.

Kriterium	Punkte	Kommentar
Dokumentation vollständig Die Dokumentation der Applikation ist insoweit vollständig, als dass potenzielle Schwachstellen in Bezug auf die Sicherheit erkannt und beseitigt werden können. Dies betrifft insbesondere die Architekturdokumentation und die Dokumentation des Sourcecodes	2	Wichtig ist vor allem eine detaillierte Architekturdokumention, sowie eine Beschreibung der Schnittstellen zwischen den einzelnen Komponenten und eine Beschreibung der Validierungen, die an diesen Schnittstellen stattfinden. Üblicherweise gibt es Dokumentation in diesem Detailgrad nicht.
Entwickler verfügbar Die Entwickler, die die Applikation ursprünglich entworfen und implementiert haben stehen für Anpassungen noch zur Verfügung.	3	
Wartungsverträge für alle Komponenten vorhanden Für die Applikation und alle in der Applikation eingesetzten Komponenten (Webserver, Applikationsserver, Datenbank, usw.) existieren Verträge, die eine Behebung von Fehlern beinhalten oder bei Open-Source-Komponenten existiert eine aktive Community, die diese Komponenten weiterentwickelt.	5	Kein Wartungsvertrag, keine Möglichkeit von Bugfixes.
Fehlerbehebungszeiten durch den Hersteller sind kurz. Die Reaktionszeiten des Herstellers vom Eingang einer Fehlermeldung bis zur Behebung liegen bei kritischen Fehlern unter einer Woche. Dies können entweder vertraglich geregelte Fehlerbehebungszeiten oder empirische Fehlerbehebungszeiten, z. B. bei Open-Source-Produkten sein.	3	Ist wichtig, hilft aber nur begrenzt.
Automatisierte Tests vorhanden Zur Qualitätssicherung der Applikation existieren automatisierte Tests, die einen hohen Testabdeckungsgrad abbilden und bei neuen Releases genutzt werden.	1	Tests tendieren dazu, zu überprüfen, ob die gewünschte Funktionalität vorhanden ist. Security in diesem Umfeld bedeutet aber, dass die nicht gewünschte Funktionalität nicht vorhanden ist -> bringt normalerweise nicht viel.
Quellcode-Analyse wurde durchgeführt Während der Entwicklung und	3	Die Analyse muss von einem Spezialisten durchgeführt werden, unabhängig ob

Weiterentwicklung der Applikation wurde eine automatisierte Quellcode-Analyse (Whiteboxtest) mit dem Fokus auf Applikationssicherheit durchgeführt.		automatisiert oder von externen Experten.
Geringe Komplexität In die Erstellung der Applikation sind inklusive aller Weiterentwicklungen weniger als 1.000 Stunden reiner Implementierungsaufwand (ohne Projektleitung) geflossen.	1	Erfahrungsgemäß ist die Komplexität am besten anhand des Implementierungsaufwands messbar. *Lines of Code* oder *Function Points* liefern doch sehr unterschiedliche Ergebnisse, je nachdem wer gerade zählt. Es sollte hier eher die Komplexität der Architektur, nicht der Implementierungsaufwand betrachtet werden.
Zentraler Controller vorhanden Die Architektur der Applikation beinhaltet einen zentralen Controller, über den sämtliche Eingaben in die Applikation und aus der Applikation heraus laufen (MVC).	3	
Sicherheits-Framework wird genutzt Die Applikation nutzt ein Sicherheitsframework das u. a. Validatoren/Filter für Ein- und Ausgabe zur Verfügung stellt.	4	Das bedeutet erstmal, dass der Entwickler mitgedacht hat. Schon mal eine sehr gute und wichtige Sache, siehe letzten Punkt.
Sicherheitsaudit wurde durchgeführt Über die Applikation wurde ein Sicherheitsaudit/Penetrationstest durchgeführt und alle im Audit erkannten Schwachstellen wurden behoben.	2	
Entwickler wurden in Security Fragen geschult und sind erfahren.	5	Das wichtigste sind immer geschulte Entwickler!

A8.2 Rollenmodell beim Betrieb von WAFs

Das hier beschriebene Rollenmodell sollte vor allem dann umgesetzt werden, wenn die WAF – über die Funktion als *Second Line of Defense* und eine Grundabsicherung hinaus – zum Schutz der Webapplikationen Aufgaben im Sinne des im Dokument beschriebenen Whitelistings übernimmt und deshalb möglichst eng an das externe Verhalten der Webapplikation angepasst wird.

Die Einführung einer WAF wird üblicherweise im Rahmen eines Projektes durchgeführt. Entscheidend für den nachhaltig erfolgreichen Betrieb einer WAF ist jedoch ein Rollenmodell, in welchem die Verantwortlichkeiten aller Beteiligten im gesamten Softwareentwicklungszyklus definiert sind. Eine WAF besitzt einerseits Charakteristika einer Infrastrukturkomponente, andererseits ist ihr Verhalten äußerst anwendungsspezifisch. Ihre Konfiguration und Verhalten kann sich sogar zwischen verschiedenen Releases derselben Anwendung deutlich unterscheiden. Die Konfiguration einer WAF ist weit umfangreicher als die einer klassischen Firewall. Sehr grob vereinfacht wird nicht mehr eine einzelne IP einer Anwendung, sondern jedes Eingabefeld einer Anwendung konfiguriert.

In größeren IT-Organisationen werden der Betrieb der Netze, zu denen die Firewall gehört, und der Anwendungen von verschiedenen Einheiten, manchmal sogar von verschiedenen Firmen, durchgeführt. Die meisten Betriebskonzepte folgen dieser organisatorischen Trennung mit einer Rollenverteilung, die klar zwischen Aufgaben auf Infrastrukturebene (Netzwerk und Betriebssystem) und auf Anwendungsebene unterscheiden.

Es wird wie bei einer Firewall die Rolle eines *Plattformverantwortlichen WAF* benötigt, der für die Betriebsaspekte der WAF zuständig ist. Wir schlagen hier die neue Rolle eines *Anwendungs-*

verantwortlicher WAF vor, die zwischen der Plattform WAF und der individuellen Anwendung angesiedelt ist. Es wird weiterhin der Anwendungsverantwortliche benötigt. Dieser braucht sich aber keine tieferen Kenntnisse der WAF anzueignen.

Der *Anwendungsverantwortliche WAF* ist die Brücke zwischen der WAF und der fachlichen Anwendung. Er muss exzellente Kenntnisse der WAF besitzen, um diese konfigurieren und für die spezielle Anwendung überwachen zu können. Er muss die Anwendung gut kennen, um Meldungen der WAF einordnen und bewerten zu können. In der Regel wird ein *Anwendungsverantwortlicher WAF* die WAF-Konfiguration für mehrere Anwendungen betreuen. Ein Beispiel wäre die Betreuung der WAF für alle webbasierten SAP-Systeme, während das Shopsystem von einem anderen *Anwendungsverantwortlichen WAF* betreut wird.

Damit werden einerseits die spezifischen Anforderungen an den sicheren und effizienten Betrieb einer WAF berücksichtigt und andererseits bestehen weiterhin die klassischen Rollen Infrastruktur- oder *Plattformverantwortlicher* und *Anwendungsverantwortlicher* aus stark strukturierten Organisationen unverändert fort.

A8.3 Die Rollen im Einzelnen

8.3.1 Plattformverantwortlicher WAF

Aufgaben:

- Planung der Betriebsarchitektur der WAF
- Verantwortlich für Betrieb und Support der WAF inklusive Kapazitätsplanung
- Einrichtung und Koordinierung Zuordnung von URLs zu individuellen Anwendungen
- Patch- und Versionsmanagement der WAF
- Verwaltung und Administration der Anwendungsverantwortlichen WAF

Kenntnisse:

- Kenntnisse der WAF, ihres Betriebs, der Administration und des Autorisierungskonzepts

8.3.2 Anwendungsverantwortlicher WAF (je Anwendung)

Aufgaben:

- Implementation und Betreuung der anwendungsspezifischen Konfiguration der WAF
- Monitoring und Analyse der Logfiles (zumindest im Second Level)
- Ansprechpartner für Fehlermeldungen, insbesondere False-Positives-Analyse in Zusammenarbeit mit dem Anwendungsverantwortlichen
- Enge Zusammenarbeit mit Anwendungsverantwortlichen und Plattformverantwortlichen WAF
- Test der WAF-Funktionalitäten für die Anwendung, insbesondere bei Versionsänderungen der Anwendung

Kenntnisse

- Tiefe Kenntnisse der Konfiguration der WAF in Bezug auf anwendungsspezifische Schutzmechanismen
- Sehr gute Kenntnisse des äußeren Verhaltens der Anwendung, insbesondere Eingabe, Ausgabe, Uploads, Downloads, Zeichensätze usw.

8.3.3 Anwendungsverantwortlicher

- Betrieb oder Entwicklung der fachlichen, zu schützenden Anwendung
- Kenntnis der Anwendungsarchitektur, der fachlichen Eingabefelder, liefert diese an den Anwendungsverantwortlichen WAF.