Best Practices:

Use of Web Application Firewalls

Version 1.0.4, March 2008, English translation 25. May 2008

Author: OWASP German Chapter with collaboration from:

Maximilian Dermann

Mirko Dziadzka

Boris Hemkemeier

Achim Hoffmann

Alexander Meisel

Matthias Rohr

Thomas Schreiber

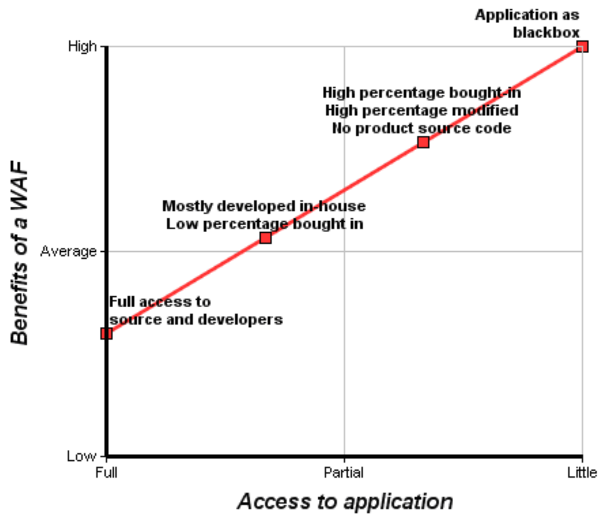
Abstract

Web applications of all kinds, whether online shops or partner portals, have in recent years increasingly become the target of hacker attacks. The attackers are using methods which are specifically aimed at exploiting potential weak spots in the web application software itself – and this is why they are not detected, or are not detected with sufficient accuracy, by traditional IT security systems such as network firewalls or IDS/IPS systems. OWASP develops tools and best practices to support developers, project managers and security testers in the development and operation of secure web applications. Additional protection against attacks, in particular for already productive web applications, is offered by what is still a emerging category of IT security systems, known as *Web Application Firewalls* (hereinafter referred to simply as *WAF*), often also called *Web Application Shields* or *Web Application Security Filters*.

One of the criteria for meeting the security standard of the credit card industry currently in force (PCI DSS - Payment Card Industry Data Security Standard v.1.1) for example, is either a regular source code review or the use of a WAF.

The document is aimed primarily at technical decision-makers, especially those responsible for operations and security as well as application owners (specialist department, technical application managers) evaluating the use of a WAF. Special attention has been paid – wherever possible – to the display of work estimates – including in comparison to possible alternatives such as modifications to the source code.

In addition to the importance of the web application regarding turnover or image – the term *access* to a web application used in this document can be a good criterion in the decision-making process relating to the use of WAFs. Specifically, the *access* to a web application, measures the extent to which the required changes to the application source code are actually carried out in-house, on time,or can be carried out by third parties. As ilustrated by the graph below, a web application to which there is no access, can only be protected sensibly by a WAF (additional benefit of the WAF),.Even with an application in full access, a WAF can be used as a central service point for various services such as secure session management, which can be implemented for all applications equally, and as a suitable means for proactive safety measures such as URL encryption



Further key topics dicussed in this paper include best practices for processes concerning the installation and operation of a WAF as well as –in particular for larger companies – a description of the role of the *WAF application manager*.

About …

This document has been developed by the *OWASP German Chapter*. The authors are employees of companys, who are consulting on the use and operation of WAFs, are producing WAFs and/or are setting up WAFs.

Authors

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	laexander.meisel@artofdefence.com	art of defence GmbH
Matthias Rohr	matthias.rohr@securenet.de	SecureNet GmbH
Thomas Schreiber	thomas.schreiber@securenet.de	SecureNet GmbH

Terminology

The specialist terms used in this document are not explained in detail and knowledge of their meaning is assumed. No glossary has been included in order to keep the volume manageable and to keep to the actual subject of this paper as closely as possible – *the use of Web Application Firewalls.*

Detailed definitions and more in-depth descriptions concerning WAS – Web Application Security – can be found at:

- 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

Licence

Contents

A1 Introduction and aim of this document

A1.1 Introduction

Whether the online branch of a bank, an online-shop, a customer-, partner- or employee-portal – all of these web applications are available to their customers – as well as their attackers – around the clock due to the *always on* nature of the internet. Attacks such as SQL injection, cross-site scripting or session hijacking are aimed at vulnerabilities in the web applications itself – and not at those on the network level. For this reason, traditional IT security systems such as firewalls or IDS/IPS are either totally unable to guard against these attacks or are incapable of offering comprehensive protection.

From a technical point of view the fundamental issue is, that the web, especially the HTTP protocol, was not designed for such complex applications which are currently state of the art. Many vulnerabilities have their origin here: for example, HTTP is not stateful, i.e. sessions or stateful applications must be defined separately and implemented securely. These vulnerabilities are increased even further by the high degree of complexity of the web scripts, frameworks and web technologies frequently used.

In addition to the recent introduction of industrial standards, e.g. the data security standard of the credit card industry (PCI DSS v1.1), security breaches in Germany which have only recently been revealed, such as the loss of approx. 70,000 items of customer data incl. credit card information for online ticker dealer *kartenhaus.de*, have ensured an increased level of interest in possible security measures against application level attacks.

This document covers a category of security systems, the *Web Application Firewalls* (WAF), which are especially well suited for securing web applications which are already in production.

A1.2 Definition of the term WAF – Web Application Firewall

In this document, a WAF is defined as a security solution on the web application level which – from a technical point of view – does not depend on the application itself. This document focuses on the exposition and evaluation of the security methods and functions provided by a WAF. Aspects of the deployment within the existing IT infrastructure – whether as a hardware appliance, a software plug-in for a web server or as an add-on for existing infrastructure components, such as load balancers or network firewalls – are only covered in brief. Unlike the definition in WAFEC – it is not assumed that a WAF has to be available as a separate hardware appliance in front of the web servers; this certainly does not represent the best implementation option, especially in large, fast-growing infrastructures.

A1.3 Target readership and objective

The document is aimed primarily at technical decision-makers, especially those responsible for operations and security as well as application owners (specialist department, technical application managers) evaluating the use of a WAF. Special attention has been paid – wherever possible – to the display of work estimates. Further key topics discussed in this paper include best practices for processes concerning the installation and operation of a WAF as well as – in particular for larger companies – a description of the role of the *WAF application manager*.

A2 Characteristics of web applications with regard to Web Application Security

A2.1 Higher level aspects within the organization

Especially within larger organizations, many aspects need to be taken into account regarding the importance of the security of the web applications in operation.

One of the most important aspects is the number of productive web applications in the company. Large companies often operate – in-house or externally – web applications numbering in the hundreds. Even if a prioritisation of each individual web application in order of its relevance for the success of the organization is reasonable , it is nevertheless necessary to assume that all web applications operated in-house – depending on the architecture – could permit an attack on internal systems given the right attack methods. Even web applications which seem to be "unimportant" at first glance should at minimum be secured against known attacks.

The following aspects should be considered when prioritizing web applications in regard to their importance for the organization:

- Access to personal data of customers, partners and/or employees
- Access to confidential information
- Essential requirement for the completion of critical business processes
- Relevance for the aittainment of critical (security-)certifications.

Possible effects of the non-availability or data loss in the web applications include:

- Interruption of business processes (including those of customers or partners)
- Loss of reputation
- Damage compensation claims
- Revocation of licenses
- Loss of confidential information.

For other aspects such as risks and costs, see A4.3 and A6.4.

A2.2 Technical aspects of each of the company's individual web application

The decision regarding suitable security measures for a web application essentially depends on the relevant phase in the application development process. This means that in the design phase suitable tools for the implementation as well as test- and quality-assurance-tools can be selected; where appropriate the developers can also be trained in web application security and the relevant time frame until the deployment into productive operation can be extended.

For already completed or productive applications, very different aspects are relevant with regard to subsequent possible security measures, such as:

- Complete documentation of the architecture and the source code or availability of the developers of the web application
- Maintenance contracts for all components of the application architecture
- Short error rectification times by the manufacturer of third party products used

Only if these aspects have been met, the application can be secured within the existing application infrastructure, not regarding the amount of work involved.

A3 Overview of Web Application Firewall (WAF) features

A3.1 Where WAFs fit into the Web Application Security field as a whole

The basic principle is that every web application should be developed as secure as possible. This is because the later a vulnerability is detected in the life cycle of a web application, the greater the risk of a successful attack, and often also the amount of work involved in correcting the issue.

In addition to appropriate training measures, e.g. on the basis of the OWASP guidelines the application development can be supported effectively by the use various tools. Tools such as *Stinger* are normally based on a framework – J2EE in this example; they are part of the application (even if they can be added to completed applications conforming to J2EE) and, from an organisational point of view, are thus generally subject to the normal application release cycle. At their core, they effective help developers in making their application more secure. Unlike WAFs, they will always be part of the application, however. These tools are mentioned in this document at various points, in particular in relation to the comparative amount of work for various security measures, but they themselves are not the focus of this document.

In the development phase, methods such as static source code analysis help to promptly detect and rectify vulnerabilities in the code. This additonally includes penetration tests, ideally carried out by experts, which cover the vulnerabilities in the external behaviour of the web application in productive operation as well.

In this context, it is the primary function of a WAF to secure web applications against detected vulnerabilities , with as little effort as possible, so that they cannot be exploited by attackers. This is already a very challenging task due to the high degree of complexity of the typical web-application infrastructure: web servers, application servers, frameworks, as well as the typical components of a web application; session handling with cookies, input validation, etc.

The main aim in using a WAF is therefore securing the existing, often productive web applications, where the required changes within the application can no longer be implemented or can only be implemented with a disproportionately large amount of work. This applies to vulnerabilities in particular which have been revealed via a penetration test or even via analysis of the source code, , and – especially in the short term – cannot be fixed within the application. Besides the basic protection via blacklisting – in other words the description of known attack patterns – the basic feauture of the WAF is the option of whitelisting which can be configured appropriately. With active whitelisting, the rule set of the WAF describes the exact behaviour of the application; the configuration of suitable whitelists is often supported via a learning mode.

In addition, several WAFs also offer functionalities which extend beyond a purely protective nature and which can therefore also be used in the design process in order to avoid unnecessary work. The WAF therefore becomes a central service point for completing tasks which should otherwise be on the application side, but which can and should be adressed in the same way for all applications. Examples of this include secure session management for all applications based on cookie stores, central authentication and authorisation, the collection of all relevant error messages and log files or the option for proactive security mechanisms such as URL encryption.

The table below uses what are currently the most well-known vulnerabilities or methods of attack on web applications to indicate the protection offered by WAFs. The usual functionality of a WAF is assumed, although not all WAFs available on the market necessarily offer all the functionality described here.

A3.2 Typical security mechanisms of WAFs using specific vulnerabilities as example

The table below gives possible security measures (Countermeasure column) for typical threats, vulnerabilities and attacks (Problem column), and in the WAF column, evaluates how well a WAF can protect the application. The symbols indicate:

- **+** very well covered by a WAF
- **-** cannot be covered (or only to a small degree) by a WAF
- **!** dependent on the WAF/application/requirements
- **=** can partially be covered by a WAF

Problem	WAF	Countermeasure
Cookie protection	+ + ! !	Cookies can be signed Cookies can be encrypted Cookies can be completely hidden or replaced (Cookie Store) Cookies can be linked to the client IP
Information leakage	+	Cloaking filter, outgoing pages can be “cleaned” (error messages, comments, undesirable information)
Session riding (CSRF)	+	URL encryption / token
Session timeout	!	Timeout for active and inactive (idle) sessions can be specified (if the WAF can manage the sessions itself). Even if the sessions are managed by the application, the WAF can detect these and terminate them with the appropriate configuration.
Session fixation	=	Can be prevented if the WAF manages the sessions itself
Session hijacking	-	Difficult to prevent, although the WAF can issue an alarm in the event of irregularities (e.g. changing IP) or terminate a session with changing IP
File upload	+	Virus check (generally via external systems) via ICAP linked to the WAF
Parameter tampering	+ +	In addition to/instead of data validation (see below), parameter manipulation can be prevented via URL encryption (GET) and parameter encryption (GET and POST) Site usage enforcement, meaning the possible sequence of URLs can be fixed or can be detected
Forced browsing	+ +	Can be prevented via URL encryption Site usage enforcement
Path traversal (URL) link validation	+ +	Can be prevented via URL encryption Site usage enforcement
Path traversal (parameter), path manipulation	+	See parameter tampering and data validation
Logging	+	All or only specific/permitted parts of the data of a request and of the connected tests can be logged
Priv. escalation	-	Privilege escalation cannot be checked, or can only be checked to a limited degree, for example via cookie/parameter encryption
Logical level	-	Application logic going beyond the validity of URLs and form fields, cannot normally be checked by a WAF
Anti-automation	=	Automatic attacks can be partially detected and blocked (e.g. number of requests/time interval, identical requests, etc.)

Application DoS (moderate)	= =	Transactions, IPs, and/or users can be blocked connections, and /or sessions can be ended
SSL	+ + +	WAF can force SSL with pre-defined encryption strength (depending on the infrastructure scenario) SSL termination on the WAF, forwarding of the SSL data (e.g. client certificate) to application SSL connection possible from WAF to application
Data validation (relating to field/content/context/appl)	+ + ! -	Can be tested to very detailed degree (length, constant value/range of values, e.g. for SELECT, character area); validation possible with whitelist and/or blacklist (signature) Rules can in part be generated automatically High dependency on application, specific fields (hidden form) or pre-defined parameters in the URL; can be automatically verified by the WAF however Risk due to *false positives*, problematic with business critical applications in particular
Data validation (general/global)	+	HTTP(w3c) conformity, a WAF conducts a canonalisation of the data so that it is available to the application in a standardised form
Buffer overflow	+	See data validation [1]
Format string attack	=	Can be detected using data validation if the corresponding characters or strings are filtered (difficult in practice, as precise knowledge of the application is required to do this) For the majority of the hidden input fields, this can be carried out without knowledge of the application
Cross-site scripting	=	Using data validation, only *reflected XSS* can be detected and prevented, *persistent XSS* cannot be detected, *DOM-based XSS* only to be limited degree if part of the attack is sent in parameters of the request
Cross-site tracing	+	Restriction of the HTTP method to, for example GETor POST,
WebDAV	+	Restriction to only reading WebDAV methods possible
Code injection (PHP, perl,	+	See data validation [1]
Command injection	+	See data validation [1]
SQL injection	+	See data validation [1]
LDAP injection	+	See data validation [1]
XML/Xpath injection	+	See data validation [1]
Just-in-time patching (hotfix patching)	+	Using data validation (see above), the WAF can protect against newly detected vulnerabilities and/or attacks (*Zero Day Exploit*)
HTTP response splitting (HTTP splitting)	!	Can only be detected using data validation in URL and/or parameters if %0d%0a is filtered – however this can be carried out on virtually any input field without impairing the functionality of the
HTTP request smuggling	+	Is prevented via strict testing of the conformity to standardsof each request

[1] Basic protection with blacklisting generally sufficient, other options be combining blacklisting and whitelisting

A4 Overview of benefits and risks of Web Application Firewalls

The specific potential benefits of a WAF described here are explained in detail in the in-depth overview in the next chapter. This chapter is used primarily as a summary for decision-makers who only want to work through the next chapter as an overview.

A4.1 Main benefits of WAFs

The main benefit of a WAF is the subsequent protection of completed, productive web applications on the application level with a reasonable amount of effort and without having to change the application itself.

On the one hand, the WAF offers a *basic protection* against known attacks or vulnerabilities based on blacklists: The data security standard of the credit card industry (PCI DSS v.1.1) for example, in its current version prescribes the use of a WAF – as an alternative to regular code reviews by a specialist – as an adequate measure to protect web applications. The WAF is therefore a suitable tool for attaining industrial standards as well as fulfilling legal requirements.

The use of a WAF becomes especially relevant in the case of concrete vulnerabilities, for example uncovered via penetration tests or source code reviews. Even if it were possible to fix the vulnerability in the application promptly and with a reasonable amount of effort, the modified version can generally only be deployed at the next maintenance interval, often 2-4 weeks later (patch dilemma). For a WAF with whitelisting, the vulnerability can be fixed promptly (hotfix), so that it cannot be exploited before the next scheduled maintenance. WAFs are especially fast in this aspect, meaning they can collaborate with source code analysis tools, so that detected external vulnerabilities can automatically result in a recommended rule set for the WAF.

A WAF is particularly important in securing productive web applications which themselves in turn consist of multiple components and which cannot be quickly changed by the operator; e.g. in the case of poorly documented applications or regarding third-party products without sufficient maintenance cycles. A WAF is the only option for promptly closing external vulnerabilities.

A4.2 Additional benefits of WAFs – depending on the actual functionality of the product

There are other considerable potential benefits which are due to the central role of the WAF. The error location process is simplified considerably if the WAF supports cerntral error messages in contrast to individually generated error messages by several applications. Errror messages can then be centrally evaluated at the WAF. The same applies to all aspects of monitoring and reporting. As a central service point, the WAF can implement tasks which can be solved in the same way for every application. A good example of this is secure session management for all applications based on cookie stores.

Many WAFs also provide proactive security mechanisms such as URL encryption or site usage enforcement, in order to minimise the area of attack with as little effort as possible. In addition, the use of a WAF increases the robustness of a web applications to external attacks.

WAFs offer other additional benefits depending on the type of implementation. A hardware appliance in front of the web servers can often terminate SSL connections and also sometimes has load balancer capabilities. This can be desirable, but can also be provided by suitable web application security add-ons for products already in use. In high-security environments, however, the existing security guidelines frequently prohibit the termination of SSL connections in front of the web server. In this case, WAFs which are implemented as a plug-in for the web server are especially well-suited.

The WAF can also provide a SSL termination if the application to be protected or its web server or application server does not have this capability.

A4.3 Risks in the use of WAFs

Note that changes in the existing IT, web and any application infrastructure are required when using a WAF. Depending on the WAF's implementation – e.g. hardware appliance vs. embedded WAF – there are also additional tasks and risks:

- Yet-another-proxy argument (increased complexity of the IT infrastructure)
- Organisational tasks (see A8.2 Role model when operating WAFs)
- Training the WAF
  - On each new release of the web application
  - Testing
- False positives (which may have a significant business impact)
- More complex troubleshooting
  - WAFs also have/generate errors
  - Responsibility for system-wide error situations
- Any potential effect on the web application if the WAF terminates the application session, for example
- Cost-effectiveness

A5 Security versus OWASP TOP10 – a comparison of WAFs and other methods

This chapter covers the various security options for what is known as the *OWASP Top10 vulnerabilities*. Three different classes of web applications are used as examples:

- T1: a web application in the design phase, new application
- T2: an already productive application (with MVC architecture), which can be easily adapted
- T3: a productive application which cannot or only with difficulty be modified.

Security measures within the application or the application architecture itself are described in detail and are evaluated, based on these three classes, either with the use of a WAF or, alternatively by definition of an appropriate security policy The security measures are also assessed in regard to the amount of work required for their implementation . In some instances, there are notes on special functionalities of WAFs or assumptions on the application infrastructure used, as these do not apply globally.

As the table below clearly shows, especially in the case of applications which are in production, the use of WAFs very often requires the least amount of work.. In the case of applications which cannot be modified or which are difficult to modify, in some instances the use of WAFs is actually the only feasible security measure.

In the table below, the *Work volume* column lists the estimated amount of work required for the application types (T1, T2, T3), a WAF or a security policy (P) in regard to the threat (*Top 10* column) Comments and notes for each type regarding the implementation of security measures can be found in the *Comment* column. The categories for the work volume are:

- **1** little work required
- **2** moderate amount of work required
- **3** considerable amount of work required
- • not normally implemented

	Top10	Type	Comment	Work volume
A1	Cross-site scripting (XSS)	T1	E.g. by the consistent use of taglibs (Java), or controls (ASP.NET), or additional frameworks (PHPIDS).	1
		T2	Input encoding is difficult to integrate (e.g. using OWASP Stinger), using an upstream WAF is a better solution here. For .NET applications XSS filters can be activated.	3 (.NET: 2)
		T3	For .NET applications, activate XSS filters.	- (.NET: 2)
		WAF	WAF does not permit output validation in this case, as it does not recognise the context of the data. The validation must be carried out during the input phase, and may be correlated with the output	2
		P		-
A2	Injection flaws	T1	Can be avoided by using an OR mapper (e.g. Hibernate) or consistent parameterisation of all inputs (e.g. stored procedures or ideally: prepared statements). Other injection flaws (e.g. with XML) can only be avoided with dedicated output coding, where necessary.	1
		T2	Complicated, as program modifications are required.	3

		T3	-	-
		WAF	WAF with blacklisting: In principle can only search for specific characters or character strings and prevent processing. Essentially there are problems with this approach in the degree of coverage as well as with possible filter evasion attacks (e.g. with multiple coding) if no input normalisation is carried out. This works very well with known attacks (e.g. SQL injection), but certainly less well with protocols not known to the WAF or with proprietary protocols. In addition, injection attacks on a some types of input data can be effectively prevented using URL encryption and hidden form parameter protection. An example of this is the item number in an online shop, which traditionally would often be used for SQL injection attacks, but it should never actually be possible for users to manipulate these directly. WAF with whitelisting: For all other input fields, there is a whitelist approach. Here the WAF can make suggestions for the individual fields following a learning phase. This means that not all, but the majority of the input fields can be protected against all types of injection attacks.	2
		P	In the case of SQL injection: Specifications for database access permissions, otherwise little or no options.	-
A3	Malicious File Execution	T1	Integrating upload scanners or whitelisting of the permitted remote inclusions.	2
		T2		3
		T3	-	-
		WAF	Whitelisting of the parameters for the permitted inclusion of URLs external to the system - inclusion of upload scanners via ICAP protocol - response analysis to prevent the display of critical data (partially also error messages).	1-2
		P	Specifications for deployment platform, specifications for access permissons	2
A4	Insecure Direct Object Reference	T1	Implementation of an object virtualisation is very time-consuming, as database objects are frequently mapped to parameters by the frameworks in use (OR mapper). Protection requires intensive testing.	3
		T2	Prevention of ID manipulation generally necessitates code modifications. Protection requires intensive testing.	3
		T3	-	-
		WAF	Protection against ID manipulation using ID virtualisation or hidden parameter protection.	1
		P	Use of impersonification and delegation.	3
A5	Cross-site Request Forgery (CSRF)	T1	Can be solved using specific application architecture.	1
		T2	Significant amount of work. Program changes generally required.	3
		T3	-	-
		WAF	Can be prevented using page token or URL encryption.	1
		P		-
A6.1	Information Leakage	T1	Tool-supported testing with high test coverage and relevant focus.	2
		T2	Tool-supported testing with high test coverage and relevant	2

			focus.	
		T3	-	-
		WAF	Automatic filtering of comments possible. Site usage enforcement can prevent access to existing but unpublished (unlinked) documents. Traditional examples are backup files on the web server which contain database passwords in plain text and whose URL can be guessed by the attacker.	1-2
		P	Requirement for programmers and authors not to enter any comments. Specifications for the design of error messages.	2
A6.2	Improper Error Handling	T1	Can be configured declaratively depending on the platform.	1
		T2	Can be configured declaratively depending on the platform.	1
		T3	Can be configured declaratively depending on the platform.	1 / -
		WAF	Difficult to detect.	2
		P	-	-
A7.1	Broken Authentication	T1	Link-up to a central access management system with appropriate security standards	1
		T2	Link-up to a central access management system with appropriate security standards. Program modifications may be required.	2
		T3	-	-
		WAF	Depends on the abilities of the WAF. A WAF can carry out authentication independent of the application and thus permit a link-up to a central authentication infrastructure without changing the application.	2
		P	Specifications with regard to password complexity.	2
A7.2	Session Management	T1	On the design level, e.g. using session manager design pattern, otherwise numerous options. Amount of implementation work partially dependent on application server, see also A7.1, if the session management is carried out by the access management system.	2
		T2	Can be integrated centrally to a large extent (using filters, listeners or hardened server configuration); nevertheless, a large amount of work in some places; see also A7.1, if the session management is carried out by the access management system.	2-3
		T3	Depends on application server, partially configurable	-
		WAF	Hardening of insecure session management possible via various techniques (e.g. page tokens).	1
		P	-	-
A8	Insecure Cryptographic Storage	T1	Use of crypto APIs.	1
		T2	Use of crypto APIs. Subsequent implementation requires numerous program modifications.	3
		T3	-	-
		WAF	-	-
		P	Specifications for saving sensitive data.	-
A9	Insecure Communiations	T1	Can be configured declaratively in the application or web server.	1
		T2	Can be configured declaratively in the application or web server. Very high amount of work if URL schema (HTTP)	1 / -

			has been hard-coded.	
		T3	Can be configured declaratively in the application or web server (if there is access). Not possible if URL schema (HTTP) has been hard-coded.	1 / -
		WAF	Can secure HTTP applications using HTTPS.	1
		P	-	-
A10	Failure to Restrict URL Access	T1	Use of a front controller with gateway. Code must still check user assignment via the program at various points (e.g. in the service). Gaps possible.	1-2
		T2	Differs depending on the application. URL access permissions can be configured declaratively with J2EE and .NET. Prevention of ID manipulation generally necessitates code modifications.	2-3
		T3	Differs depending on the application. URL access permissions can be configured declaratively with J2EE and .NET.	3
		WAF	Page tokens or URL encryption can be used to restrict users to pages received from the application as links. The application must not display protected links, however (limited access pattern). With site usage enforcement, the user can only access linked content. Specific URLs/sub-trees can also be excluded via whitelist/blacklist approaches (e.g. only allow access for *.html, *.php, *.gif, *.jpg – but not for *.bak or other extensions).	1

A6 Criteria for deciding whether or not to use a WAF

A6.1 Organization-wide criteria

Core criteria in this area are:

- Importance of the web application(s) for the success of the oganization (proportional turnover, reputation)
- Importance of the loss of data of the web application (customer data, confidential information, reputation)
- Number of web applications
- Basic legal conditions or industrial standards
- Complexity
- Operating costs
- Performance
- Scalability

A6.2 Criteria with regard to a web application

The term of *access* to the web application is introduced and explained below. The checklist in appendix A8.1 is used to determine the degree of access individually for each web application, using a points system.

The *access to a web application* can be used as a measure of the extent to which the organization in posession of the application can promptly carry out or initiate and implement the necessary changes to the web application, in other words has access to the source code of the application.

A web application in the design phase (see T1 in A5) can be considered as a special case of a *web application with optimum access*.

The other extreme, a *web application without access* is an application consisting of many undocumented components, for example, whose developer cannot be contacted, and which uses third-party software products, which are no longer maintained by the manufacturer, or – in case of open source projects - by the community (see T3 in A5).

Important criteria for determining the degree of access to a web application, are:

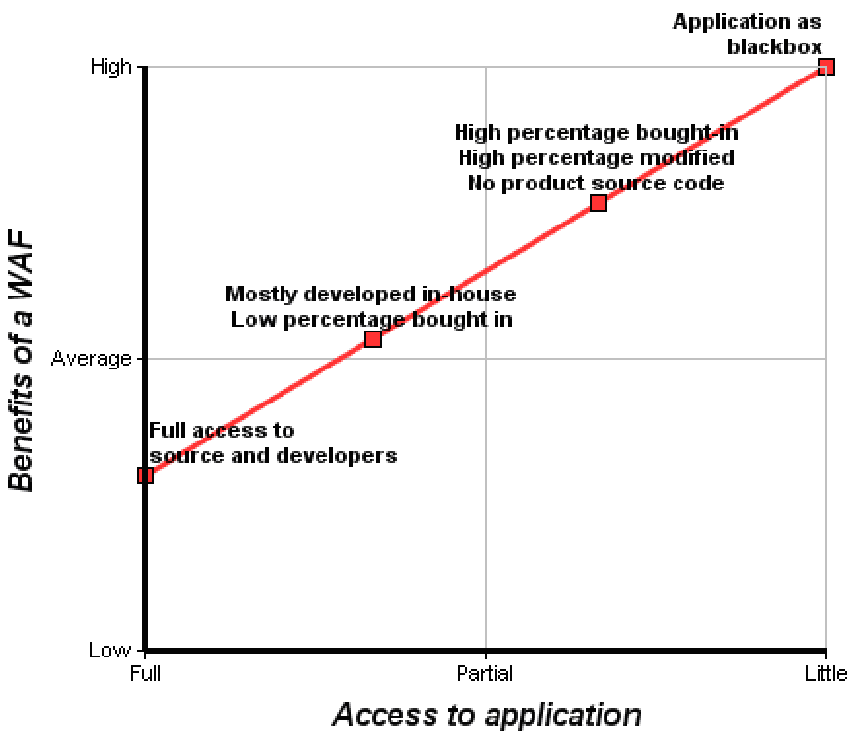
- Complete documentation of the architecture and the source code or availability of the developers of the web application
- Maintenance contracts for all components of the application architecture
- Short error rectification times by the manufacturer for all third-party products used (portals, frameworks, SAP, etc.).

Other important criteria for each web application are given in the checklist which can be found in the appendix.

A6.3 Evaluation and summary

The degree of access can be determined for every web application using the checklist in appendix A8.1. This also allows to determine a mean value of access for all the web applications of an organization; it is important to note that applications which are critical to the success or the image of the organization need to be rated accordingly.

The illustration given below may be useful as a guide in the decision-making process regarding the benefits of using a WAF:



If an organization has full access to their web applications, the use of a WAF primarily provides a reduction of the cost of operation – especially due to the additional benefits of a WAF given in A3 as a central service point, as well as some comparatively easy-to-implement security mechanisms, see A4.

If there is virtually no access to the web applications, the use of a WAF is definitely appropriate as this is the only way that the relevant security measures can be implemented.

With decreasing access to the web application – and depending on its importance and complexity – the benefits stemming from the use of a WAF grow rapidly: from a *second line of defence* to true *full protection* of the web application from outside influence, attained by the use of whitelisting. Using a WAF often results in the least additional work for the required security level.

A6.4 A consideration of the financial aspects

The cost-effectiveness of the procurement and the operation of a WAF can be considered from multiple points of view:

- Avoidance of financial damage resulting from successful attacks on the web application
- Lower costs for reaching the nescessary protection level for the web application in comparison to other options
- Savings via the use of central services which are made available by a WAF for multiple web applications, and therefore no longer have to be implemented or configured in every application.

When protecting applications with insufficient access (see A6.2), but which still need to be protected, the costs of a WAF can either be viewed as a strategic investment, or where realistic, set against the costs of replacing the application in question.

The costs of using a WAF normally consist of the following components:

- Licence costs
- Licence updates / software support
- Project costs for evaluating and introducing a WAF
- (Partial) costs for operating the necessary platform
- Personnel costs for the WAF application manager(s)
- time required in projects for coordination with the WAF application manager.

A7 Best practices for introducing and operating a WAF

A7.1 Aspects of the existing web infrastructure

7.1.1 Central or decentral infrastructure – predictable changes

It is essential to note that its the WAF that needs to be integrated into the existing Web infrastructure – and its planned or foreseeable changes – and not the infrastructure which needs to be fundamentally changed due to the implementation of a WAF.

Accordingly, a WAF can be installed in a central infrastructure which is not predicted to change, as a central infrastructure component, e.g. as a hardware appliance; whereas with an infrastructure which is still decentral, but which may be growing quickly – for example a large online shop – a distributed WAF approach, e.g. as a plug-in into the existing web servers, is more appropriate. With regard to the infrastructure aspects, those WAF products are particularly flexible, which combine an essentially distributed implementation approach with a central administration point and therefore offer the benefits of both scenarios.

hat is worth mentioning – and becoming increasingly important with regard to probable future developments – is the option of hardened infrastructures using virtualisation. When selecting the WAF, it is particularly important that the WAF can also be integrated seamlessly into a virtualised approach.

7.1.2 Performance criteria

With regard to technical performance, it is necessary to ensure that the required WAF infrastructure supports the main key performance indicators of the existing web infrastructure. Statements which purely refer to the GB throughput of hardware should not be taken at face value, as the given numbers are often not achievable in practice. What is more important are the typical key performance indicators of a web application such as the number of simultaneous users of the application and on that basis, the number of HTTP requests per time unit on average and at peak load times. It should be noted that many applications have high-load phases which occur only rarely, e.g. during the Christmas season for an online shop.

A7.2 Organisational aspects

7.2.1 Conforming to existing security policies

As far as possible, existing security policies should not have to be changed due to the implementation of a WAF.

A typical example is SSL termination “in front of the web servers”. This is often denied, in particular in high-security infrastructures, by the existing security guidelines This policy can be maintained by the use of a suitable WAF, as a plug-in on the web server with the SSL termination still subsequently being carried out in the web server.

7.2.2 New role model: WAF application manager

After the one-off task of commissioning, the subsequent successful use of a WAF essentially depends on the seamless interaction of the WAF with all other components of the application infrastructure . These include both obvious issues such as understanding of and appropriate response to error and alarm messages originating from the WAF, as well as aspects such as the modification of the WAF rule set in conjunction with changes to the applications being protected. To fully exploit the opportunity presented by a WAF as a central service point for instance for secure session management, positive collaboration with application development is required.

In other words: In order to fully exploit the potential of a WAF, it is not sufficient to view the WAF solely as an infrastructure component.

For this reason, we propose the new role of a *WAF application manager* – in addition to the role of a *WAF platform manager*, who in a similar way to a *network firewall platform manager* is responsible for the infrastructure-related aspects of the WAF - for each application which – metaphorically speaking – represents the bridge between the WAF and the specialist application. This person must have excellent knowledge of the WAF in order to be able to configure and monitor it for each individual application. He or she must know the application well to be able to classify and interpret messages coming from the WAF. A *WAF application manager* will normally maintain the WAF configuration for multiple applications. An example would be managing the WAF for all web-based SAP systems, whilst the shop system is managed by another WAF application manager.

A detailed description of the proposed role model can be found in appendix A8.3.

A7.3 Iterative procedure for implementation – from basic security to full protection

An iterative procedure has been tried and trusted as best practice in the implementation and operation of WAFs.

7.3.1 Step 1: Specification of role distribution / inclusion of application development

First the responsibilities need to be defined, ideally on the basis of the role concept presented above. If the web application development is being carried out in-house, this needs to be integrated into the process as early on as possible. This means that all applications not yet in production use the central functions of the WAF as soon as possible, which increases security and saves time and money. In addition, possible obstacles on the personal level can also be overcome at an early stage.

7.3.2 Step 2: Basic protection for all web applications

Regardless of the characteristics of the web application in question, *basic protection*, normally implemented as blacklisting, is activated first. Initial evaluations normally show the first successful protection measures, or show false positives – i.e. rules are set too strictly At the same time this phase serves as training for the organisational processes.

7.3.3 Step 3: Creating a priority list of all existing web applications

The priciple for this list of priorities can be the measure of the access to the web application according to the checklist in appendix A8.1, in addition to the higher level criteria such as a loss of reputation, etc..

7.3.4 Further steps: Full protection of the web applications according to priority

Web applications are *fully protected* from outiside attack with whitelist rule sets in a step by step process according to the priority list. This is normally supported by a learning mode in the WAF or a source code review/penetration test. The *WAF application manager*, in collaboration with the specialist application manager, ensures the full availability of the application at all times, including during a conversion of the rule set.

A8 Appendices

A8.1 Checklist: Access to a web application from a security-standpoint

The following checklist can be used to evaluate the *access* that a company has to the web application. Access to a web application gets better, as more more points are accumulated.

Criterion	Points	Comment
Documentation complete The documentation for the application is complete in such detail, that potential vulnerabilities relating to security can be detected and rectified. This especially pertains to the documentation of the architecture and the the source code	2	Especially important is a detailed documentation of the architecture, as well as a description of the interfaces between the individual components and a description of the validations taking place on these interfaces. Documentation on this level of detail is normally not available.
Developers available The developers who originally designed and implemented the application are still available for modifications.	3	
Maintenance contracts for all components There are contracts covering the rectification of errors or with open source components, there is an active community continuiing the development for all components of the application (web server, application server, database, etc.) and the application itself.	5	No maintenance contract, no possibility for bug fixes.
Error rectification times by the manufacturer are short. The response times from the manufacturer from the reporting of an error to delivery of a patch are less than a week for critical errors. Theses can either be error rectification times based on contracts or empirical error rectification times, e.g. for open source products.	3	Important, but only helps to a limited extent.
Automated tests exist There are automated tests for quality assurance of the application representing a high degree of test coverage and they are used with new releases.	1	Tests tend to check whether the required functionality is available. Security in this context does mean that the undesirable functionality is not present -> this does not normally accomplish much.
Source code analysis has been completed in past development and ongoing development of the application, an automated source code analysis (whitebox test) is carried out with the focus on application security.	3	The analysis must be carried out by a specialist, regardless of whether it is automated or carried out by external experts.
Low complexity ,Fewer than 1000 hours have been spent purely on implemeting the	1	Based on experience, complexity is best measured using the time spent on implementing the application. *Lines of code* or *function foints* provide

application (not including project management) in the development phase.		very different results, depending on who is doing the counting. Ideally, it would be better to consider the complexity of the architecture, not the time spent on implementation,.
Central controller present The architecture of the application includes a central controller, which processes all the inputs and outputs of the application (MVC).	3	
Security framework is used The application uses a security framework that, among other things, provides validators/filters for input and output..	4	This means mainly that the developers have considered security aspects as important. Certainly a very positive and important issue, see last point.
Security audit has been carried out A security audit/penetration test has been carried out against the application and all vulnerabilities detected in the audit have been rectified.	2	
Developers have been trained in secure programming and are experienced.	5	Always the most important thing are trained developers!

A8.2 Role model when operating a WAF

The role model described here should be implemented primarily when the WAF carries out tasks in the context of whitelisting described in this document, in order to protect the web applications, in addition to functioning as a *second line of defence* and basic security. It should therefore be configured as closely as possible to the functionality of the web application.

The introduction of a WAF is normally carried out as part of a project. The decisive factor for a longterm, successful operation of a WAF, however, is a role model in which the responsibilities of all parties involved are defined in the overall software development cycle. A WAF has both characteristics of an infrastructure component, and its behaviour is also highly specific to the application. Its configuration and behaviour can even vary considerably between different releases of the same application. The configuration of a WAF is much more complex than that of a traditional firewall. To put it simply, it no longer suffices to configure a single IP for an application, instead each input field of that application has to be configured.

In larger IT organisations, operation of the network, to which the firewall belongs, and of the applications, is carried out by different organizational units, sometimes even by different companies. Most operating concepts follow this organizational separation with a role concept which makes a clear distinction between tasks on the infrastructure level (network and operating system) and on the application level.

As with a firewall, the role of a *WAF platform manager* is required, who is responsible for the operational aspects of the WAF. We are proposing the new role of a *WAF application manager* whos responsibilities lie between the WAF and the individual application. An application manager is still required. This manager is not required to have a deeper understanding of the WAF, however

The *WAF application manager* is the bridge between the WAF and the specialist application. This person must have excellent knowledge of the WAF to be able to configure it and monitor it for the individual application. He or she must know the application well to be able to classify and interpret messages coming from the WAF. A *WAF application manager* will normally maintain the WAF configuration for multiple applications. An example would be maintaining the WAF for all web-based SAP systems, whilst the shop system is maintained by another *WAF application manager*.

This means that, on the one hand the specific requirements for the secure and efficient operation of a WAF are taken into account, and on the other hand, the traditional roles of infrastructure or *platform manager* and *application manager* remain unchanged within highly structured organisations.

A8.3 The individual roles

8.3.1 WAF platform manager

Tasks:

- Planning of the operational architecture of the WAF
- Responsiblity for operation and support of the WAF, including capacity planning
- Allocation of URLs to individual applications
- Patch and version management of the WAF
- Management and administration of the application manager WAF

Knowledge:

- Knowledge of the WAF, its operation, administration and the authorisation concept

8.3.2 WAF application manager (per application)

Tasks:

- Implementation and maintainance of the WAF configuration specific to the application
- Monitoring and analysis of the log files (at least on the second level)
- Contact for error messages, in particular false positives analysis in collaboration with the application manager
- Close cooperation with the WAF application managers and platform managers
- Test of WAF functionalities for the application, especially when deploying new versions of the application

Knowledge:

- In-depth knowledge of the WAF configuration in relation to application-specific security mechanisms
- Very good knowledge of the behaviour of the application, in particular input, output, uploads, downloads, character sets, etc.

8.3.3 Application manager

- Operation or development of the application to be protected
- Knowledge of the application architecture and the input fields, provides these to the WAF application manager.