Web Application Firewall Evaluation Criteria

Version 1.0 (January 16, 2006)

Table of Contents

Introduction

Web Application Firewalls (WAF) represent a new breed of information security technology that is designed to protect web sites (web applications) from attack. WAF solutions are capable of preventing attacks that network firewalls and intrusion detection systems can't. They also do not require modification of the application source code. As today's web application attacks expand and their relative level of sophistication increases, it is vitally important to develop a standardised criteria for product evaluation. How else can we accurately compare or measure the performance of a particular solution?

The goal of this project is to develop a set of web application firewall evaluation criteria; a testing methodology that can be used by any reasonably skilled technician to independently assess the quality of a WAF solution. However, our aim is not to document the features that *must* be supported in order for a product to be called a web application firewall. Web application firewalls are simply too complex to be treated like this.

To conclude: the purpose of this document to draw one's attention to the features that are of *potential importance* to a given project. This comprehensive list should be used as basis to form a much shorter list of features that are *required for the project*. The shorter list should then be used to evaluate multiple web application firewall products.

Current categories are as follows:

1. Deployment Architecture
2. HTTP Support
3. Detection Techniques
4. Protection Techniques
5. Logging
6. Reporting
7. Management
8. Performance
9. XML

We expect to cover the following categories in the subsequent releases:

- Compliance, certifications, and interoperability.
- Increase coverage of performance issues (especially on the network level).
- Increase coverage of the XML-related functionality.

Contributors

This document is a result of team effort. The following people have contributed their time and expertise to the project:

- Robert Auger (SPI Dynamics)
- Ryan C. Barnett (EDS)

- Charlie Cano (F5)
- Anton Chuvakin (netForensics)
- Matthieu Estrade (Bee Ware)
- Sagar Golla (Secureprise)
- Jeremiah Grossman (WhiteHat Security)
- Achim Hoffmann (Individual)
- Amit Klein (Individual)
- Mark Kraynak (Imperva)
- Vidyaranya Maddi (Cisco Systems)
- Ofer Maor (Hacktics)
- Cyrill Osterwalder (Seclutions AG)
- Sylvain Maret (e-Xpert Solutions)
- Gunnar Peterson (Arctec Group)
- Pradeep Pillai (Cisco Systems)
- Kurt R. Roemer (NetContinuum)
- Kenneth Salchow (F5)
- Rafael San Miguel (daVinci Consulting)
- Greg Smith (Citrix Systems)
- David Movshovitz (F5)
- Ivan Ristic (Thinking Stone) [*Project Leader*]
- Ory Segal (Watchfire)
- Ofer Shezaf (Breach Security)
- Andrew Stern (F5)
- Bob Walder (NSS Group)

Contact

Participation in the Web Application Firewall Evaluation Criteria project is open to all. If you wish to comment on the evaluation criteria or join the team mailing list please contact Ivan Ristic via email (`<ivanr@webkreator.com>`).

Categories

Section 1 - Deployment Architecture

This section highlights the questions key to determining the feasibility of web application firewall deployment in a given environment.

1.1 Modes of Operation

Can the device be operated in both passive and active (inline) mode?

Describe which of the following *active modes* of operation apply to the WAF:

1. *Bridge*. Can be installed as a transparent bridge. Can it be configured to fail open?
2. *Router*. Network must be reconfigured to direct traffic through the WAF.
3. *Reverse Proxy*. Traffic is re-directed to flow through the WAF by making changes to DNS configuration or by traffic redirection on the network level.
4. *Embedded*. WAF is installed as a web server plug-in. Which web servers are supported? Explain the level of integration with the web server. Some embedded web application firewalls may only tap into the communication channel and do everything themselves. Others may rely on the web server to do as much of the work as possible. (Both approaches have their advantages and disadvantages.)

1.2 SSL

SSL is often used to protect traffic coming from and going to web applications. While this type of protection achieves the goal of data protection, it hides the data from the protection systems (e.g. intrusion detection systems, web application firewalls) at the same time. Since SSL is in widespread use - in fact, secure deployments require it - if a WAF cannot get to the traffic then it will be unable to perform its function.

Describe how the WAF can be deployed to access the protected data:

1. *Terminates SSL*. The network needs to be re-configured to move the SSL operations to the WAF itself. WAF decrypts the encrypted traffic to get access to the HTTP data. The communication between the WAF and the web server can be in plain-text, or SSL-encrypted.
2. *Passively decrypts SSL*. Configured with a copy of the web server's SSL private key WAF decrypts the SSL traffic. The original data stream travels unaffected to the web servers, where it is separately decrypted and processed.
3. *Not Applicable*. Working embedded in a web server, a WAF can be positioned to work just after the SSL data is decrypted into plain-text.

Client certificates criteria:

1. Are client certificates supported in passive mode?
2. Are client certificates supported in active mode?

3. In termination mode, can the content from client certificates be sent to the application using some alternative transport method (e.g. request headers).

Other SSL criteria:

1. In termination mode, can the backend traffic (i.e. the traffic from the WAF to the web server) be encrypted via SSL?
2. Does the WAF support client certificates for backend communication?
3. Are all major cipher suites supported by the SSL implementation. Which ones?
4. Can the WAF retrieve SSL keys from an external key storage facility (e.g. network-based *Hardware Security Module*)?
5. Is the SSL implementation FIPS 140-2 certified? Which FIPS levels are supported (level II and/or III)?
6. Is there support for hardware-based SSL acceleration? If there is, are the SSL certificates securely stored in the hardware?

1.3 Traffic Blocking

If the WAF is capable of blocking offending traffic, describe the nature of the blocking functionality:

1. *Connection Intermediation*. Traffic is intercepted and network protocol connections are terminated on the WAF. Attacks are blocked by not forwarding the blocked requests to the destination.
2. *Connection Interruption*. Traffic is inspected, but not terminated by the WAF. Attacks are blocked by stopping the connection to the destination. This can be either before any packets arrive at the destination (e.g. a single-packet attack), or after a partial connection has been buffered, but not completed, on the destination (e.g. in the case of segmented packets).
3. *Connection Reset*. Traffic is inspected by the WAF either via active, passive or embedded inspection mechanism. Attacks are blocked by resetting the relevant network (TCP) connections. Connection reset is often used in conjunction with other blocking mechanisms.
4. *Blocking via third-party device*. Traffic is inspected by the WAF. Attacks are blocked by notifying other devices (e.g. router or network firewall) to block a connection.

Describe the scope of blocking capabilities:

1. Blocks the HTTP request.
2. Blocks the connection.
3. Blocks the IP address.
4. Blocks the application session.
5. Blocks the application user.

When blocking is taking place on the HTTP level, can the WAF be configured to present the user with a friendly, meaningful, message? Can a unique transaction ID be presented to the user (see Section 5.1)?

For a WAF that supports blocking, is it possible to turn blocking off (completely, or for certain types of requests only - determined dynamically for every request)?

1.4 Method of Delivery

Describe how WAF is delivered:

1. *Appliance*. Software bundled with hardware. Appliances are usually highly optimised and may contain specialised hardware components to increase performance. Are there other optional hardware components that may further increase performance (except for SSL, which was already covered in 1.2)?
2. *Software only*. WAF is delivered as software application that can be installed on a generic computer. Describe the reference hardware configuration. Is the application delivered integrated with an operating system? Or does it require an operating system to be installed on? Which operating systems are supported (including versions)? Are there other optional hardware components that can increase performance further (such as SSL encryption cards)?

1.5 High availability

There are two aspects of the high availability requirement. One is to prevent the web application firewall from becoming a single point of failure. The other, to allow the WAF to scale and remain fully functional for very busy sites. The questions are as follows:

1. When used in active mode, is it possible to configure the WAF to fail open?
2. Does the WAF support multi-node operation? Describe possible architectures.
3. In a two-node high availability system, how does the unit fail over, and in what timeframe?
4. In a two-node high availability system, can the second node share the load?
5. Is the node state shared? (If the state is shared then a node can go down without any impact on the system.) Specifically, does the stateful nature apply to SSL sessions?
6. Can multiple WAF nodes be configured to work in a cluster (of more than two nodes)? What is the maximum number of nodes supported?
7. Can nodes be geographically distributed across data centres? How does this affect failover and state?

1.6 Inline operation

1. Does the WAF support complete virtualisation of the external application representation? Describe the support for virtualisation:
   a. Virtual hosts.
   b. Network ports.
   c. URL mappings.
   d. Authentication realms. (Covered in more detail in Section 2.7.)
   e. Cookies.
   f. Sessions.
2. Does the WAF support HTML (response) rewriting to change URIs after virtualisation takes

place?

3. Is access to the HTML rewriting functionality given to the user? Which parts of the traffic can be rewritten?
   a. Request headers.
   b. Response headers.
   c. Parameters (including those in `QUERY_STRING`, `application/x-www-form-urlencoded POST` requests, and `multipart/form-data POST` requests.
   d. Request body - raw.
   e. Request body - XML.
   f. Request body - other (list supported content types).
   g. Response body - raw.
   h. Response body - HTML.
   i. Response body - other (list supported content types).
4. Is there support for caching?
5. Is there support for transparent response compression?
6. Can the WAF completely rebuild back-end requests?

1.7 Support for non-HTTP traffic

Is there support for protocols other than HTTP (for example: FTP, DNS, LDAP)? Describe the functionality.

For the WAFs that operate inline and on the network level: is the network-level firewall functionality supported? Describe it.

Section 2 - HTTP and HTML Support

2.1 Supported HTTP versions

Provides support for:

1. HTTP/0.9
2. HTTP/1.0
3. HTTP/1.1

2.2 Widely-used encoding types

The following encoding types should be supported:

1. Supports `application/x-www-form-urlencoded` encoding
2. Supports `multipart/form-data` encoding
3. Supports v0 cookies

4. Supports v1 cookies
5. Supports chunked encoding in requests
6. Supports chunked encoding in responses
7. Supports request compression
8. Supports response compression

2.3 Strict protocol validation

1. Can restrict methods used
2. Can restrict protocols and protocol versions used
3. Strict (per-RFC) request validation
4. Validate URL-encoded characters
5. Validation of the non-standard `%uXXYY` encoding
6. Can enforce cookie types used (e.g. only allow v1 cookies)

Restrictions

Which of the following restriction methods are supported:

1. Element content.
2. Element length.
3. Element byte range.
4. Character set validation (e.g. UTF-8). Which character sets are supported?

How flexible is the WAF when it comes to applying different restriction policies to different parts of the request? Can restrictions be set at the application level, the object level, and/or the parameter level?

2.3 Additional protocol limits

The following protocol limits should be supported:

1. Can restrict request method length
2. Can restrict request line length
3. Can restrict request URI length
4. Can restrict query string length
5. Can restrict protocol (name and version) length
6. Can restrict the number of headers
7. Can restrict header name length
8. Can restrict header value length
9. Can restrict request body length
10. Can restrict cookie name length
11. Can restrict cookie value length

12. Can restrict the number of cookies

2.4 HTML Restrictions

1. Can restrict parameter name length
2. Can restrict parameter value length
3. Can restrict the number of parameters
4. Can restrict combined parameter length (names and values together)

2.5 File transfers

1. Support for `POST` file uploads (`multipart/form-data` encoding)
2. Support for `PUT` file uploads
3. Individual file size can be restricted
4. Combined file size can be restricted
5. The number of files in a request can be restricted
6. Is it possible to add custom logic to inspect uploaded files?

2.6 Support for different character encodings

Web Application Firewall must evaluate request data in the character encoding used by the application in order to provide adequate protection.

2.7 Authentication

This section covers support for standard or widely deployed authentication methods. The supports needs to be evaluated from two points of view.

Can WAF support applications that use these authentication methods?

1. Basic Authentication
2. Digest Authentication
3. NTLM Authentication
4. Client SSL certificates

Can WAF support these authentication methods directly (i.e. providing an additional authentication layer)?

1. Basic Authentication
2. Digest Authentication
3. NTLM Authentication
4. Client SSL certificates
5. Two-factor authentication

Note

More details on support for SSL client certificates are available in Section 1.2.

Can authentication be integrated with any kind of backend authentication scheme? Is there built-in support for common authentication backend types (e.g. LDAP, RADIUS, etc).

Can WAF support security assertion and Federated Identity protocols?

1. SAML (1.0, 1.1, 2.0)
2. Liberty-ID-FF (1.0, 1.1, 1.2)
3. WS-Trust

2.8 Response filtering

Is the WAF capable of analysing the outgoing responses (sometimes referred to as "Identity Theft Protection", or "Intellectual Property Firewalling")?

1. Response headers
2. Response body
3. Response status

Section 3 - Detection Techniques

3.1 Normalisation techniques

Web systems are often designed as a combination of many related systems. This allows the attackers to attempt evasion by transforming the attack payload into a form that appears harmless to the web application firewall. The large variety of backend systems used makes the work of WAFs very difficult. In order to make rules and signatures useful WAFs must attempt to detect evasion attempts and transform input data into a normalised form.

Does the WAF support the following normalisation methods:

1. URL-decoding (e.g. `%XX`)
2. Null byte string termination
3. Self-referencing paths (i.e. use of `/./` and encoded equivalents)
4. Path back-references (i.e. use of `/../` and encoded equivalents)
5. Mixed case
6. Excessive use of whitespace
7. Comment removal (e.g. convert `DELETE/**/FROM` to `DELETE FROM`)
8. Conversion of (Windows-supported) backslash characters into forward slash characters.
9. Conversion of IIS-specific Unicode encoding (`%uXXYY`)
10. Decode HTML entities (e.g. `&#99;`, `&quot;`, `&#xAA;`)
11. Escaped characters (e.g. `\t`, `\001`, `\xAA`, `\uAABB`)

3.2 Negative security model

When negative security model is used transactions are allowed by default. Only those transactions that contain attacks are rejected. The success of this model depends on the WAF being able to detect bad requests.

The questions are as follows:

1. *Signature-based*. Signature-based products usually detect attacks by performing a string or a regular expression match against traffic.
2. *Rule-based*. Rules are similar to signatures but allow for a more complex logic to be formed (e.g. logical `AND`, logical `OR`). They also allow for specific parts of each transaction to be targeted in a rule.

3.3 Positive security model

Positive security model denies all transactions by default, but uses rules to allow only those transactions that are known to be valid and safe. This approach can be more efficient (fewer rules to process per transaction) and more secure, but it requires very good understanding of the applications being protected. The positive security model can also be more difficult to maintain if the web application changes frequently.

The questions in this section are as follows:

1. Is the positive security model supported?
2. Can positive security model be configured manually?
3. Can positive security model be configured automatically (training)? What tools are available for automatic configuration? How is the automatic configuration maintained (dynamically, manually, completely re-learn)?
4. Does the tool support model updates (without re-training)? Can this be done during run-time or does the WAF need to be taken offline for policy modifications?

Positive security model can be achieved using strict content-validation policies, or using statistical analysis or neural networks in an approach commonly referred to as anomaly-based detection.

3.4 Signature and rule database

Signature/rule database:

1. Does the product come with a database of signatures (or rules), which are designed to detect known problems and attacks.
2. How many products does the database support?
3. How often is the database updated?
4. Is the database cross-referenced to operating systems, applications, and versions, allowing the administrator to apply only the applicable signatures?

3.5 Extension APIs

Extension APIs allows programmers to build plug-ins. Plug-ins can contain custom logic to evaluate requests.

Section 4 - Protection Techniques

This section lists specific requirements for protection that are not (and cannot be) covered using generic protection mechanisms described in other sections. Describing all possible web application security classes of problem is out of scope of this document. You will find this area was already covered in the Threat Classification project: http://www.webappsec.org/projects/threat/.

4.1 Brute Force Attacks Mitigation

1. Detect brute force attacks against access controls (HTTP status code `401`).
2. Detect brute force attacks (repeated requests for the same resource) against any part of the application.
3. Custom brute force attack detection for applications that do not return `401` (probably with a regular expression applied to a response part).
4. React by either slowing or blocking the attacker down.
5. Detect brute force attacks against session management (too many sessions given out to a single IP address or range).
6. Detect automated clients (session requests are too fast).

4.2 Cookie Protections Measures

1. Sign cookies, to prevent clients from changing them.
2. Encrypt cookies, to hide contents.
3. Hide cookies altogether (cookie mechanism virtualisation: only send unique cookie IDs back to the client).
4. Is it possible to configure protective measures per application or otherwise control where they are applied?

4.3 Session Attacks Mitigation

1. Complete virtualisation of the session handling mechanism used in the application. Which transport methods are supported:
   a. Cookies
   b. Form parameters
   c. URI rewriting
2. Can session IDs be tied in with the SSL session ID values?
3. Can session IDs be tied in to the authentication mechanism?
4. Is it possible to configure protective measures per application or otherwise control where they

are applied?

4.4 Hidden Form Fields Protection

Is there a mechanism to (optionally) protect certain hidden form fields from changing?

4.5 Cryptographic URL and Parameter Protection

Does the WAF support any kind of generic cryptographic URL Encryption to protect application URLs and parameters against manipulation (forceful browsing, session riding, etc.)?

4.6 Strict Request Flow Enforcement

Strict request flow enforcements refers to the technique where a WAF monitors individual user sessions and keep track of the links followed and of the links that can be followed at any given time.

Section 5 - Logging

5.1 Unique transaction IDs

Unique transaction ID is assigned to every HTTP transaction (a transaction being a request and response pair), and included with every log message.

5.2 Access logs

Access logs refer to a record of all transactions that went through the WAF. Access logs are most commonly delivered as files, although other forms are also in use (e.g. databases).

1. Access logs can be exported as files.
2. Which commonly-used log formats are supported?
3. Access log format can be customised.
4. Access logs can be sent to the logging server via Syslog.
5. Access logs can periodically be uploaded to the logging server (e.g. via FTP, SFTP, WebDAV, or SCP).

5.3 Event logs and notification

Event logs refer to a record of all suspicious transactions. Events logs are commonly delivered as files, or are stored in a database.

We are expecting web application firewalls to support some of the following notification methods:

1. Email
2. Syslog
3. SNMP Trap
4. OPSEC Event Logging API (ELA - http://www.opsec.com/intro/sdk_overview.html#ela)
5. Notification via HTTP(S) push.

6. Periodic event log upload via one of the supported upload mechanisms (see 5.2).

It is also important to note the format in which the events are imported. Is the format:

1. Plain-text based?
2. XML-based?
3. Or is there an API to customise the format?

5.4 Full transaction logs

Full transaction logs must include complete HTTP requests and responses. The emphasis is on the request body, with the option to record the response body too. The requirements are as follows:

1. The transaction log format should be well documented.
2. It should be possible to configure the detail level. For example, to allow users to record request bodies but not record response bodies. A really smart tool could be configured to record limited detail on non-suspicious transactions, but to record full detail of suspicious transactions.
3. It should be possible to configure which transactions are logged. For example, a user may want to log only the suspicious transactions, or only the transactions that involve dynamic processing on the server.
4. A WAF that understands sessions could be configured to start logging full session data once a suspicious transaction is detected.

5.5 Log access

The following are requirements for log access:

1. Logs can be exported (as files) via FTP or SCP, periodically or on demand.
2. Logs can be accessed directly in the a database (either real or simulated), using an ODBC/JDBC driver. The database schema is published and documented.
3. Logs can be accessed via an API (pull).
4. API exists that allows a plugin to be developed to manipulate log entries as they arrive (push).
5. Can logs be digitally signed to thwart tampering?

5.6 Syslog support

If the tool supports Syslog then the following requirements need to be evaluated too:

1. Supports UDP-based Syslog logging.
2. Supports connection-oriented Syslog logging.
3. Connection-oriented Syslog logging can be wrapped in SSL to protect the logs in transport.
4. When SSL is used, client and server certificates can be specified for additional security.

5.7 Log retention

Log retention refers to the ability of the device to preserve the important logs in accordance with the organisational policies, and to prevent the system from overflowing.

1. Maximum age of the logs can be specified. Logs older than the specified age are deleted.
2. Maximum size of the log store can be specified. Once the store reaches the maximum size the oldest logs are deleted.
3. There is support for multiple log retention policies (e.g. transactions involving policy violations can be kept for longer).
4. Logs can be automatically backed up before they are deleted.

5.8 Handling of sensitive data

1. Can the WAF remove (sanitise) sensitive data from the logs?
2. Is sanitation configurable (e.g. explicitly specify which parameters to obfuscate)?
3. Can the WAF automatically detect sensitive data? Can this feature be customised?
4. Which obfuscation methods are available? (One simple method is to replace every byte with a character, for example `*`. Another, to use a reversable transformation, which would allow staff to decode data if necessary.)

Section 6 - Reporting

6.1 Event reports

While it is expected the analysts will review the events frequently, reports are needed to track the overall security level:

1. Can generate comprehensive event reports. Which filters are available:
    a. Date or time ranges?
    b. IP address ranges?
    c. Types of incidents
    d. Other (please specify).
2. Reports can be generated on demand.
3. Reports can be generated on schedule (e.g. daily, weekly)

6.2 Report formats

The following report formats are deemed of relevance:

1. Word
2. RTF
3. HTML
4. PDF
5. XML (for further, custom, processing)

6.3 Report presentation

The goal of customisation is for reports to appear as most other documents produced in the organisation:

1. Visual customisation (e.g. colour, logo, etc).
2. Content customisation (target groups, e.g. Executive, Developers, etc)
3. Do reports contain appropriate graphs?
4. Which target groups are supported out-of-the-box?

6.4 Report distribution

How can reports be distributed:

1. Automatically via email (push).
2. Automatically via FTP (push).
3. Through a report portal (pull).

Section 7 - Management

Management is a key part of any network device. This is especially true for application security devices, since this layer is constantly changing to address specific needs, which are directly related with business requirements. For this reason, a WAF's management component should meet a set of requirements that will guarantee adequate policy enforcement, refinement, and verification. Also, management should be designed and implemented carefully, so that it doesn't interfere with service components in terms of throughput or availability. The following are topics that must be covered by WAF's management components. They are classified into several categories in order to better understand what requisite belongs to which component. Also, this approach make easier to evaluate a WAF device from a high abstraction level: components can be scored individually without delving into specific details related to them.

7.1 Policy Management

7.1.1 Simplicity to relax automatically-built policies

Most of WAF devices can automatically learn application logic and build a policy that addresses security and service requirements in that application. Nevertheless, if this policy generates too many false positives, there should be an easy procedure to selectively relax its enforcement rules, instead of having to rebuild it from scratch with lower security level.

7.1.2 Simplicity to manually accept false positives

There must be an easy way to include legitimate requests originally considered as attacks by the current security policy. This task shouldn't be harder than clicking on a log entry and pushing changes to the WAF device.

7.1.3 Ability to define different policies for different applications

This is very self-descriptive. Newer applications probably need a learning-mode phase that shouldn't apply to the rest of applications for which a stable policy has been already built.

Also see Section 2.2.4 for more requirements for the policy granularity.

7.1.4 Ability to create custom attack signatures or events

Positive Security Model should clear the need for signatures. Nevertheless, many commercial products combine both methods of detection for improved accuracy. In this case, management components should include facilities to develop custom signatures that identify specific, unique risks associated with protected applications.

7.1.5 Ability to customise Denial of Service policies

There is no standard definition of what can be considered a denial of service in every environment. For this reason, WAFs must let the administrator define or refine parameters used to identify this type of attacks. Even better, the device may be smart enough to build an statistical model of what is considered legitimate use of an application and identify when a DoS is taking place, in terms of deviation from this model.

7.1.6 Ability to combine detection and prevention

Policies must offer a high level of granularity. Switching between detection and policy enforcement (prevention) modes cannot be associated with the whole application, but must be granularly associated with every component of an application or every type of attack. This lets an administrator just monitor some potential illegal activity in the application while blocking true intrusion attempts at the same time.

7.1.7 Policy roll-back mechanism

If the new version of one policy fails to correctly protect the website or affects the way services are provided, there should exist one mechanism to easily roll-back to the previous applied policy.

7.1.8 Policy Management across multiple systems

Can a policy be shared among multiple systems? If so, how is it synchronised? Can a policy be exported as a flat file and manually imported into other systems? How is versioning controlled in this scenario?

7.2 Profile Learning Process

7.2.1 Ability to recognise trusted hosts

Learning mode is based on watching traffic between clients and applications, considering that no attacks will occur over that period of time. This assumption may not be true in many cases; for this reason, there must be management configuration facilities to specify trusted hosts that will let the WAF device learn only legitimate traffic.

7.2.2 Ability to learn about the application without human intervention

There must exist some built-in mechanism that lets the WAF learn application logic without the need for users to manually browse the website. This can be a spider, scanner or crawler that automatically visits every link on every page and behaves as a human user, making use of all the services in the website. The spider must be accurate enough and fully integrated with the policy editor. Also, it may be useful to schedule these scan tasks in order to quickly identify changes made to web applications and adjust the current policy accordingly.

7.2.3 Ability to adjust policies outside the GUI

Sometimes wizards and WAF's intelligence is not enough to create accurate policies. For this reason, there must exist a way to delve into low-level details of policies and adjust them somehow. This can mean making changes to regular expressions or adding new ones.

7.2.4 Ability to inspect policy

1. Can the application security policy be easily audited and reported on? How?
2. Can changes to the application security policy be logged and reported on? In what detail (for example: change made, date, time, administrative user)?

7.3 Configuration Management

7.3.1 Role-based management with user authentication

There must be multiple profiles in the device user login, to support segregated responsibilities. These might include, but not be limited to: device operators (for box configuration), application owners (for policy configuration), auditors (for log inspection) and so forth.

7.3.2 Support for trusted hosts

Trusted hosts (identified by IP addresses or IP address ranges) sometimes need to be allowed to perform activities that are otherwise prohibited by policy. For example, when a penetration test is executed it is necessary to turn off the protective measures and/or prevent the detected attacks from escalating into alerts.

1. Can the WAF suppress prevention techniques for trusted hosts and only log the error messages as notices?
2. Can the WAF ignore traffic coming from trusted hosts (not even logging the notices)?

7.3.3 Automatic signature update and install

If signatures are included as an additional detection method, there should be an option to automatically download and install updates in every sensor.

7.3.4 Ability to remotely manage sensors

It's recommended that WAF devices communicate with management systems over a dedicated management network. This enables administrators to securely perform maintenance and configuration tasks. Also, communications between the management systems and WAF devices must be simple enough to traverse firewalls and secure enough (that is, encrypted with strong algorithms) to avoid eavesdropping or other types of attacks based on protocol analysis.

7.4 Logs and Monitoring

7.4.1 Ability to identify and notify system faults and loss of performance

The management components must be able to monitor system status and alert the administrator when an error condition occurs or when performance is somehow impacted. Typical notification mechanisms are e-mail, SNMP, syslog and pager.

7.4.2 Ability to suppress logs

Log suppression is a simple mechanism to aggregate logs in a clever way and simplify WAF's activity reviews: similar entries are presented as being just one, with an additional column showing how many times this entry was registered. Criteria used for joining entries are user-defined.

7.4.3 Ability to generate service and system statistics

Statistics can help an administrator check if the WAF device is performing as expected. Also, it provides hints about the protected servers' activity and performance. The following is basic statistical data that must be available in the ligament tool:

- number of requests/connections/sessions per second
- percentage of malicious requests
- percentage of blocked requests (when combining detection and prevention)
- number of errors over time
- types of browsers identified (through User-Agent header)
- most accessed URLs

This data should be available in readily-accessed logs, or even displayed on the device itself (with the recognition that advanced user behaviour reporting is not the primary function of a WAF).

7.5 Miscellaneous

7.5.1 Interface robustness and reliability

Buggy interfaces are painful when dealing with WAF devices. For this reason, special care should be taken to avoid exceptions, management subsystems faults and error messages while using the UI. There must exist a mechanism to assure that if an error occurs while pushing new policies to the sensor, the WAF device will get back to its original state.

On the other side, the UI must be hardened enough to make sure that it is not vulnerable to the same types of attacks it protects of. This includes common techniques like SQL Injection or Parameter Tampering , but also advanced forms of exploitation.

7.5.2 Management User Interface Implementation

Is the management interface implemented as:

1. Native application?
2. Web-based application?

7.5.3 Management Interface

Is there a separate network interface to form a separate channel for device management?

Is there a dedicated management interface? How is it implemented:

1. Separate network interface for SSH/HTTPS access?
2. Serial console access?

Is two-factor authentication supported for management access? Describe what is supported.

7.5.4 Backend Control API

Does the WAF provide any kind of backend control API so that backend applications can influence the behaviour of the WAF (e.g. terminate user session, block IP, delay logon retry, etc.)?

7.5.5 WAF Security

How secure is the WAF itself? Which operating system is it based on? Is the underlying operating system maintained and regularly patched? Can the patching process be automated?

Section 8 - Performance

We acknowledge the issue of performance is a complex one. Measuring the performance of a WAF on network level is especially difficult and out of scope of this document. (For one approach have a look at the methodology used by the NSS Group: http://www.nss.co.uk/WebApp/Ed1/WebApp%AD_Performance_Testing.htm) The two subsections we include here are targeted toward measuring the overhead of a WAF when no protective measures enabled. We are likely to expand this section over time.

8.1 HTTP-level performance

1. Maximum new connections per second.
2. Maximum throughput per second (retrieving a single 32KB HTML page).
3. Maximum requests per second (with Keep-Alives enabled).
4. Maximum number of concurrent connections.
5. Request latency.

The above performance criteria assumes maximum values with no packet loss.

8.2 HTTP-level performance with SSL enabled

Performance measurement with SSL enabled (but disabled at the backend):

1. Maximum new (not resumed) SSL connections per second.

2. Maximum SSL session resumptions per second.
3. Maximum SSL traffic throughput with a given encryption algorithm (retrieving a single 32KB HTML page).
4. Maximum requests per second (with Keep-Alives enabled).
5. Maximum number of concurrent connections.
6. Request latency.

The above performance criteria assumes maximum values with no packet loss.

8.3 Performance under load

How is system manageability affected by production traffic load? Is the box manageable under high attack loads?

Section 9 - XML

This section contains only the basic XML questions at the time. We expect the content to be expanded in the subsequent releases.

1. Does the WAF support protection of XML Web Services?
2. Can WS-I Basic conformance be specified (http://www.ws-i.org)?
3. Can the WAF restrict XML Web Services access to methods defined via Web Services Description Language (WSDL) ?
    a. Can the WAF deny XML Web Services access to an administrator-defined set of WSDL methods?
    b. Can the WAF restrict XML Web Services inputs to an administrator-defined set(s) of data types (parts) and formats?
4. Does the WAF perform validation for both Web Services RPC communications?
5. Does the WAF perform validation for Web Services XML Documents?

Describe additional XML Web Services functionality.

A. Licence