Web Application Firewall Configuration Standard (Barracuda)

*Revision: r1.0*

*Effective Date: ddmmyyy*

*Classification: Internal*

**Internal INFORMATION**

This is a proprietary document and is the property of XXXX.; it contains information that is proprietary, or otherwise restricted from disclosure. If you are not an authorised recipient, please return this document to the above-named owner(s). Dissemination, distribution, copying or use of this document in whole or in part by anyone other than the intended recipient is strictly prohibited without prior written permission of XXXX.

Table of Contents

1. Introduction 3

1.1 Document Definition 3

1.2 Objective 3

1.3 Scope 3

1.3.1 Applicability to employees 3

1.3.1 Applicability to External Parties 3

1.3.2 Applicability to Assets 3

1.4 Industry Configuration Standards 3

1.5 Related Documents / References 3

2. Standard Statements 4

2.1 Configure Request Limits 4

2.2 Secure HTTP Cookies 5

2.3 Configuring URL Protection 6

2.4 Configuring Parameter Protection 8

2.5 Limiting Allowed Methods in HTTP Headers and Content 10

2.6 Configuring Cloaking 10

2.7 Configuring Data Theft Protection 11

2.8 Configuring URL Normalization 13

2.9 Configuring Global ACLs 14

2.10 Configuring Action Policy 15

3. Standard Compliance & Enforcement 17

3.1 Compliance Measures 17

3.2 Enforcement 17

4. Exception Process / Glossary 18

4.1 Exception Process 18

4.2 Glossary / Acronyms 18

5. Document Management 19

5.1 Document Revision Log 19

5.2 Document Ownership 19

5.3 Document Coordinator 19

5.4 Document Approvers 19

5.5 Distribution 19

# Introduction

## Document Definition

This document is a Standard.

For a full description of document types, see XXXX-POL-ALL-001 - Information Security Policy Framework.

## Objective

The objective of this standard is to ensure the XXXXs business and technical requirements for the web application firewall are translated into a baseline that will allow administrators to easily configure the firewall to adequately protect the XXXX’s information assets against external and internal threats.

## Scope

### Applicability to employees

XXXX refers to XXXX as well as its majority-owned subsidiaries and joint ventures (if applicable). This Standard applies to all employees, officers, members of Board of Directors, and all consultants, and contractors.

### Applicability to External Parties

Relevant Standard statements will apply to any external party and be included in contractual obligations on a case-by-case basis.

### Applicability to Assets

This Standard applies to all information assets globally owned by XXXX, or where XXXX has custodial responsibilities.

## Industry Configuration Standards

* Centre for Internet Security (CIS): http://www.cisecurity.org/
* Security Technical Implementation Guides (STIGs): http://iase.disa.mil/stigs

## Related Documents / References

* *XXXX-POL-ALL-001 - Information Security Policy Framework*

# Standard Statements

## Configure Request Limits

Request limits define the validation criterion for incoming requests by enforcing size limits on HTTP request header fields. The requests that have fields larger than the specified maximums are dropped. Properly configured limits mitigate buffer overflow exploits, preventing Denial of Service (DoS) attacks.

**Steps to Configure Request Limits**

1.Go to the SECURITY POLICIES > Request Limits page.

2.Select the policy from the Policy Name drop-down list for which you want to modify request limits settings.

3.In the Request Limits section, specify values for the following fields:

a. Enable Request Limits - When set to Yes, size limit checks are enforced on request headers.

◾ Values: Yes, No

◾ Recommended: Yes

b. Max Request Length - Enter the maximum allowable request length. This includes the Request-Line and all HTTP request headers (for example, User Agent, Cookies, Referer etc.) The Request Length limit does not include the request body, which is typically present for POST requests. Any request, whose length exceeds this limit, will be denied.

◾ Range: 1 byte to 65536 bytes.

◾ Recommended: 32768 bytes

c. Max Request Line Length – Enter the maximum allowable length for the request line. The request line consists of the method, the URL (including any query strings) and the HTTP version. Example:

GET /index.cgi?page=home HTTP/1.1

In the above request line, GET is the method, /index.cgi?page=home is the URL and HTTP/1.1 is the version. The length of the entire line is considered when checking for request line length.

◾ Range: 1 byte to 65536 bytes.

◾ Recommended: 4096 bytes

d. Max URL Length – Enter the maximum allowable URL length including the query string portion of the URL.

◾ Range: 1 byte to 128 kilobytes. No value (empty) implies unlimited.

◾ Recommended: 4096 bytes

e. Max Query Length – Enter the maximum allowable length for the query string portion of the URL.

◾ Range: 1 byte to 60000 bytes. No value (empty) implies unlimited.

◾ Recommended: 4096 bytes

f. Max Number of Cookies – Enter the maximum number of cookies to be allowed.

◾ Range: 1 to 1024. If no value is provided or if the field is left empty, it indicates unlimited value.

◾ Recommended: 40

g. Max Cookie Name Length – Enter the maximum allowable length for cookie name.

◾ Range: 1 byte to 1024 bytes. No value (empty) implies unlimited.

◾ Recommended: 64 bytes

h. Max Cookie Value Length – Enter the maximum allowable length for a cookie value. Requests with Cookie values that are larger than the defined setting are denied.

◾ Range: 1 byte to 32768 bytes. No value (empty) implies unlimited.

◾ Recommended: 4096 bytes

i. Max Number of Headers – Enter the maximum number of headers in a request. If there are more headers than this limit in the request, the request is denied.

◾ Range: 1 to 40. No value (empty) implies unlimited.

◾ Recommended: 20

j. Max Header Name Length – Enter the maximum allowable length for header name.

◾ Range: 1 byte to 1024 bytes. No value (empty) implies unlimited.

◾ Recommended: 32 bytes

k. Max Header Value Length – Enter the maximum allowable length for any request header. A request header could either be an HTTP protocol header such as "Host," "User-Agent" and so on, or a custom header such as "IIS Translate". A request may contain any number of these headers.

◾ Range: 1 byte to 64 kilobytes. No value (empty) implies unlimited.

◾Recommended: 1024 bytes

4.Click Save

## Secure HTTP Cookies

Cookies provide a mechanism to store session state information on client navigation platforms such as browsers and other user agents. Cookies can store user preferences or shopping cart items and can include sensitive information like registration or login credentials. If a cookie can be modified, the system can become vulnerable to attacks and sensitive information can be stolen.

To encrypt or sign cookies and reject tampered cookies, you need to enable cookie security using the following steps:

1.Go to the SECURITY POLICIES > Cookie Security page.

2.Select a policy from the Policy Name list.

3.In the Cookie Security section, select the desired Tamper Proof Mode, either Encrypted or Signed. Recommended: Signed. Note: Encrypting cookie data makes cookie values inaccessible to the client. If required, change values of other parameter(s):

◦ Cookie Max Age – Enter the maximum age in minutes for session cookies after which cookies are automatically expired

◾ Range: 0 to 500000

◾ Recommended: 1440

◾ Units: Minutes

◦ Cookie Replay Protection Type – Select the type of protection to be used to prevent cookie replay attacks from the drop-down list.

◾ Values: Custom Headers, IP, IP and Custom Headers, None

◾ Recommended: IP

◦ Custom Headers – Enter the custom headers to be used in the cookie if the parameter Cookie Replay Protection Type is set to Custom Headers or IP and Custom Headers and click Add.

◦ Secure Cookie – Set to Yes to allow cookies to be returned to the web server if the client makes secure HTTPS connection only.

◾ Values: Yes, No

◾ Recommended: No

◦ HTTP Only – Set to Yes to allow cookies to be returned to the web server only if the client makes an HTTP connection. When a client-side script attempts to read a cookie with HTTP Only enabled, the browser returns an empty string as the result. Enabling HTTP Only prevents the cookie from being accessed through client-side scripts.

◾ Values: Yes, No

◾ Recommended: No

◦ Allow Unrecognized Cookies – Select whether unrecognized cookies should be allowed. Use Custom to temporarily allow unrecognized cookies after deploying. Use Days Allowed to indicate for how long unrecognized cookies are allowed.

◾ Values - Always, Never, Custom

◾ Default - Custom

◦ Days Allowed – Enter the number of days unrecognized cookies will not be rejected. This field is used only when Allow Unrecognized Cookies is Custom.

◦ Cookies Exempted – Add the names of cookies to exempt from the cookie security policy.

4.Click Save.

## Configuring URL Protection

URL requests and embedded parameters in them can contain malicious script. Attacks embedded in URL requests or their parameters are executed with the permissions of the executing component. Injection of operating system or database commands into the parameters of a URL request, cross site scripting, remote file inclusion attacks, and buffer overflow attacks can all be perpetrated through unchecked URL requests or their parameters.

**Steps To Configure URL Protection**

1.Go to the SECURITY POLICIES > URL Protection page.

2.Select the policy from the Policy Name drop-down list for which you want to modify URL protection settings.

3.In the URL Protection section, specify values for the following fields:

a. Enable URL Protection – Select Enable to enforce URL Protection when URL Profiles are not used for validating the incoming requests.

◾ Values: Enable, Disable

◾ Recommended: Enable

b. Allowed Methods – Specify the list of methods to be allowed in the request. The Barracuda Web Application Firewall uses this list to determine whether to allow or deny methods in the requests. See Limiting Allowed Methods in HTTP Headers and Content.

c. Allowed Content Types – Specify the list of content types to be allowed in the POST body for a URL. “application/x-www-form-urlencoded” and “multipart/form-data” are typical content types that are used in form submissions. These content types are recognized by the Barracuda Web Application Firewall and cause it to parse the content into parameters and values. Other content types are used by custom built applications in various special ways. For example, text/xml may be used by Web Services enabled applications. These content types, when encountered in the request, are not given any special consideration, and are passed through to the server if they are allowed by this setting.

d. Max Content Length – Enter the maximum content length to be allowed for POST request body. Note: Only requests with the Content-Length: headers are validated. Requests encoded using "Chunked Encoding" DO NOT have a Content-Length: header, and therefore are not subject to the Content Length check.

◾ Range: 0 to 8388608 bytes. No value (empty) implies unlimited.

◾ Recommended: 32768 bytes

e. Max Parameters – Enter the maximum number of parameters to be allowed in a request. Parameters can be supplied as part of the query string or as part of the request body or both. This limit applies to each method of supplying the parameter individually. For example, if the Max Parameters is 4, a maximum of 4 query string parameters are allowed and a maximum of 4 request body parameters are allowed. Thus, when both methods are combined, a maximum of 8 parameters will be allowed (4 each).

◾ Range: 0 to 1024. No value (empty) implies unlimited.

◾ Recommended: 40

f. Max Upload Files – Enter the maximum number of files that can be uploaded in one request. If the value is set to two (2), then the third (3) file upload is denied. The Passive mode logs every uploaded file that exceeds the max count.

◾ Range: 0 to 1024. No value (empty) implies unlimited.

◾ Recommended: 5

g. Max Parameter Name Length – Enter the maximum length of the parameter name in the request.

◾ Range: 0 to 1024. No value (empty) implies unlimited.

◾ Recommended: 64

◾ Units: Bytes

h. Blocked Attack Types – Select the attack types that needs to be matched in the requests/responses. Attack Types are specifications of malicious patterns. If the value of a parameter matches one of the specified Attack Types, an intrusion is detected and logged on the BASIC > Web Firewall Logs page.

Attack Types are defined with groups of Regular expression patterns. Attack Types for SQL Injection, Cross Site scripting and System Command Injection attacks are provided by default, and one or more of these can be enabled for matching against request parameters.

i. Custom Blocked Attack Types – Select the custom attack types that needs to be matched in the requests/responses.

j. Exception Patterns – Enter the patterns to be allowed as exceptions in spite of them being part of a malicious pattern group. The configuration should be the exact "Pattern Name" as found on the ADVANCED > View Internal Patterns page, or as defined during the creation of a "New Group" through the ADVANCED > Libraries page. The pattern name can also be found in a Web firewall log when a false positive occurs due to such a potentially "exception" pattern. For example, if the parameter value matched "sql-comments" regex pattern under "sql-injection medium" attacks on the ADVANCED > View Internal Patterns page, then adding "sql-comments" to this list will allow "sql-comments" in future.

4.Click Save.

## Configuring Parameter Protection

To protect a service from attacks which employ the parameters of a URL query string or parameters of the form POST parameters, use SECURITY POLICIES > Parameter Protection. Parameter Protection defends web applications from Parameter based attacks when parameter profiles aren't used.

Parameters that contain special characters may have SQL or html tagging expressions embedded in them. Embedded SQL keywords like "OR," "SELECT," or "UNION" in a parameter, or system commands such as "xp\_cmdshell" can exploit web application vulnerabilities. These attack patterns can be configured in Parameter Protection and compared to requests. If a parameter matches, the corresponding request is not processed.

**Steps To Configure Parameter Protection**

1.Go to the SECURITY POLICIES > Parameter Protection page.

2.Select the policy whose parameter protections settings you want to configure from the Policy Name drop-down list.

3.In the Parameter Protection section, configure the following fields:

a. Enable Parameter Protection – Select Yes to enforce parameter protection when Parameter Profiles are not used for validating the incoming requests.

◾ Values: Yes, No

◾ Recommended: Yes

a. Denied Metacharacters – Specify disallowed meta-characters in parameters. Non-printable characters such as "backspace" and UI reserved characters like "?" should be URL encoded. Denied meta-characters help prevent SQL Injection and cross-site scripting attacks. Some specified meta-characters may be valid for some parameters, resulting in valid requests being blocked. The meta-character list should be appropriately tuned for specific parameters to avoid this problem. To add meta-characters, click the Edit icon enter disallowed values.

a. Maximum Parameter Value Length – Specify the maximum allowed length of any parameter value, including no-name parameters.

◾ Range: 0 to 1073741824. Leave blank for "unlimited"

◾ Recommended: 1000

◾ Units: Bytes

a. Maximum Instances – Enter the maximum number of times a parameter is allowed in a request. By default, the value is set to 1. Restricting this value to one (1) avoids a large class of HTTP Parameter pollution attacks and is recommended.

b. Base64 Decode Parameter Value– Set to Yes to apply base64 decoding to the parameter values. If the parameter value adheres to the Data URI Scheme, the base64 decoding is applied on the parameter value irrespective of Base64 Decode Parameter Value is set to Yes or No. If not, the base64 decoding is applied to the parameter value only when Base64 Decode Parameter Value is set to Yes. Once the decoding is successful, other parameter checks are enforced as per the policy settings.

c. Allowed File Upload Type– Select Extensions to allow the files uploaded with extensions specified in File Upload Extensions.

Select Mime Types to identify the content in the files before allowing to be uploaded with the mime types specified in File Upload Mime Types.

d. File Upload Extensions – Specify the extensions of files which may be uploaded. '.' is a special extension allowing files with no extension, and '\*' allows any extension.

e. File Upload Mime Types– Specify the Mime types that are to be allowed as uploaded files. Use a "." to indicate a file with unknown mime type and use a \* to indicate any kind of mime type.

f. Max Upload File Size – Specify the maximum allowed size of individual files being uploaded.

◾ Range: 0 to 51200. Leave blank for "unlimited"

◾ Recommended: 1024

◾ Units: Kilobytes

g. Blocked Attack Types – Select the attack types that needs to be matched in the requests. Attack Types specify malicious patterns. Parameter values which match one of the specified Attack Types indicate an intrusion and are logged on the BASIC > Web Firewall Logs page.

Attack Types are defined by groups of Regular expression patterns. Attack Types for SQL Injection, cross-site scripting and System Command Injection attacks are provided by default, one or more of which can be enabled for comparison to request parameters.

h. Custom Blocked Attack Types – Select the custom attack types that needs to be matched in the requests. For information on how to create custom blocked attack types.

i. Exception Patterns – Enter patterns which should be allowed despite matching a malicious pattern group. Configure the exact "Pattern Name" displayed on the ADVANCED > View Internal Patterns page or configured creating a "New Group"on the ADVANCED > Libraries page. The pattern name is also displayed in the Web Firewall Log when it is wrongly denied (a false positive). For example, if the parameter value matched "sql-comments" regex pattern under "sql-injection medium"on the ADVANCED > View Internal Patterns page, then add "sql-comments" to the list to allow "sql-comments" in future.

j. Ignore Parameters – Specify parameters exempt from all validations. Use this to skip validations for especially large parameters that are automatically generated by servers, such as \_\_VIEWSTATE. Since these parameters are auto-generated, they are less likely to be attacks, and therefore can safely be exempted from validation checks. Note: Ignore Parameter is an exact match; wildcard is not supported. So a value with "\*" does not work like a wildcard. Examples: \_\_VIEWSTATE, POSTBODY

4.Click Save.

## Limiting Allowed Methods in HTTP Headers and Content

While GET and POST are the predominant methods used by web servers for information access,

HTTP allows several less known methods\*:

•HEAD

•GET

•POST

•PUT

•DELETE

•TRACE

•OPTIONS

•CONNECT

The OPTIONS command allows clients to determine which methods the web server supports. Some methods allow modification of stored files, stealing of user credentials, or bypassing environment level access control checks. URL protection allows an explicit way to specify allowed or disallowed methods in URL calls. Disallowing PUT, DELETE, and TRACE is recommended. The allowed request content-types also need to be carefully restricted to prevent similar security threats.

## Configuring Cloaking

Cloaking prevents hackers from obtaining information that could be used to launch a successful subsequent attack. HTTP headers and return codes are masked before sending a response to a client. The response headers are filtered based on the headers defined in the Headers to Filter field

**Steps To Configure Cloaking**

1.Go to the SECURITY POLICIES > Cloaking page.

2.Select the policy from the Policy Name drop-down list for which you want to modify cloaking settings.

3.In the Cloaking section, specify values for the following fields:

a. Suppress Return Code – When set to Yes, the Barracuda Web Application Firewall blocks an HTTP Status code in the response header and inserts a default of custom response page in case of any error responses from the server. Two types of response error codes are suppressed:

i.4xx (client): These are 400-series error codes. These codes are intended for instances when a client seems to have erred when attempting to access a Web page.

ii. 5xx (server): These are 500-series error codes. These codes are intended to indicate that a server is aware that it has a problem or that it is incapable of performing a request. Example: 500: Internal Error.

◾ Values: Yes, No

◾ Recommended: Yes

b. Filter Response Header – Set to Yes to remove HTTP headers in the response before relaying to the client. The HTTP headers are filtered based on the headers defined in the Headers to Filter field below.

◾ Values: Yes, No

◾ Recommended: Yes

a. Headers to Filter – Define the HTTP headers to be removed from the response before serving it to the client.

4.Click Save.

## Configuring Data Theft Protection

Data theft protection prevents unauthorized disclosure of confidential information. Configuring data theft protection requires two steps:

•Specify any at risk data elements handled by the web application using Security Policy.

•Enable protection of these elements where needed, using URL Policy.

**Steps to Configure Data Theft Protection:**

1.From the SECURITY POLICIES > Data Theft Protection page, select the policy for which you want to enable data theft protection.

2.In the Configure Data Theft Protection section, specify values for the following fields:

a. Data Theft Element Name – Enter a name for the data theft element.

b. Enabled – Select Yes to use this data element to be matched in the server response pages. This data element is used for matching server response pages only when Enable Data Theft Protection is also set to Yes on the WEBSITES > Advanced Security page.

◾ Recommended: Yes

d. Identity Theft Type – Select the data type from the drop-down list that the element mentioned in Data Theft Element Name belongs to. The default identity theft patterns (Credit Card, SSN and Directory Indexing) are associated to data types defined under ADVANCED > View Internal Patterns > Identity Theft Patterns. If you want to associate a custom identity theft pattern created on the ADVANCED > Libraries page, select <CUSTOM> from the drop-down list and then select customized identity theft type from the Custom Identity Theft Type field below.

◾ Default: CUSTOM

e. Custom Identity Theft Type – Select the customized identity theft type to be used from the drop-down list.

f. Action – If set to Block, the response sent by the server containing this data type is blocked. The Block mode should be used if the server should never expose this information. In the Cloak mode, a part of the data is cloaked, that is, overwritten with X’s based on Initial Characters to Keep and Trailing Characters to Keep.

◾ Values: Block, Cloak

◾ Recommended: Block

h. Initial Characters to Keep – Enter the number of initial characters to be displayed to the user when the data of this data type is identified in a server page. For example, an online shopping service displays a user’s credit card number 1234 0000 0000 5678. If Initial Characters to Keep is set to 4, the credit card number is displayed as 1234 XXXX XXXX XXXX.

◾ Values: 0 to 100

◾ Recommended: 0

i. Trailing Characters to Keep – Enter the number of trailing characters to be displayed to the user when the data of this data type is identified in a server page. For example, an online shopping service displays a user’s credit card number as 1234 0000 0000 5678. If Trailing Characters to Keep is set to 4, the credit card number is displayed as XXXX XXXX XXXX 5678.

◾ Values: 0 to 100

◾ Recommended: 4

3.Click Add to add the above configuration settings.

**Custom Identity Theft Patterns**

The default data theft types are displayed under Protected Data Types in the SECURITY POLICIES > Data Theft Protection page. You can also create custom identity theft data types on the ADVANCED > Libraries page to use.

Creating a Custom Identity Theft Pattern

1.Go to the ADVANCED > Libraries page, Identity Theft section, enter a name in the New Group field and click Add.

2.Click Add Pattern next to the created identify theft pattern group. The Identity Theft Patterns window appears. Specify values for the following fields:

a. Pattern Name – Enter a name to identify the pattern.

b. Status – Set to On if you wish to use this pattern for pattern matching in the responses.

c. Pattern Regex – Define the regular expression of the pattern or click the Edit icon to select and insert the pattern.

d. Pattern Algorithm – Select the algorithm to associate with the pattern from the drop-down list.

e. Case Sensitive – Select Yes if you wish the pattern defined to be treated as case sensitive.

f. Pattern Description – (Optional). Enter the description for the pattern defined. Example, Visa credit card pattern. This indicates the pattern used here is the visa credit card pattern.

3.Click Add.

**Using a Custom Identity Theft Pattern**

1.Go to the SECURITY POLICIES > Data Theft Protection page.

2.Select a policy from the Policy Name drop-down list.

3.In the Configure Data Theft Protection section, enter a name in the Data Theft Element Name text field.

4.Set Enabled to Yes to use this data element to be matched in the server response pages. This data element is used for matching server response pages only when Enable Data Theft Protection is also set to Yes on the WEBSITES > Advanced Security page.

5.Select CUSTOM from the Identity Theft Type drop-down list.

6.Select the Identity theft pattern you created from the Custom Identity Theft Type drop-down list.

7.Set the Action to Block or Cloak. If set to Block, the response sent by the server containing this data type is blocked. The Block mode should be used if the server is never expected to expose such information. In the Cloak mode, a part of the data is cloaked, that is, overwritten with X’s based on Initial Characters to Keep and Trailing Characters to Keep.

8.If required, change the values of Initial Characters to Keep and Trailing Characters to Keep and click Add.

9.Now, you should bind this policy to a Service, so that any request coming to that service is matched with the pattern and then processed.

**Turning on Data Theft Protection using URL Policy**

To use Data Theft Protection for a requested URL, from the WEBSITES > Advanced Security page you must set Enable Data Theft Protection to Yes for the appropriate URL Policy, either a URL policy matching the requested URL, or if the URL has no matching policy, for the default URL Policy. When Enable Data Theft Protection is set to Yes for a requested URL, the Data Theft Protection settings from the Service's Security Policy will be enforced for this request.

## Configuring URL Normalization

The Barracuda Web Application Firewall normalizes all traffic before applying any security policy string matches. For HTTP data, this requires decoding Unicode, UTF, or Hex to base text, to prevent disguised attacks using encoding formats for which string matches are not effective.

**Steps To Configure URL Normalization**

1.Go to the SECURITY POLICIES > URL Normalization page.

2.Select the policy from the Policy Name drop-down list.

3.In the URL Normalization section, specify values for the following fields:

a. Default Character Set– Select the character set decoding type to be used for incoming requests. By default, it is set to UTF-8. The character set decoding type are:

i. English only: ASCII (7-bit), ISO-8859-1 (8-bit)

ii. Unicode: UTF-8

iii. Chinese: GBK, GB2312, HZ, BIG-FIVE, EUC-TW, ISO-2022-CN

iv. Japanese: Shift-JIS, EUC-JP, ISO-2022-JP

v. Korean: EUC-KR, JOHAR, ISO-2022-KR

vi. Values: ASCII, BIG5, EUC-JP, EUC-KR, EUC-TW, GB2312, GBK HZ, ISO-2022-CN, ISO-2022-JP, ISO-2022-KR, ISO-8859-1, JOHAB, Shift-JIS, UTF-8

vii. Recommended: UTF-8

b. Detect Response Charset – Set to Yes to detect the character set decoding in the response page through the META tags Content-Type headers. When set to No, it will not detect the character set decoding of the response. Instead it will use the static settings of "Default Character set".

◾Recommended: No

c. Parameter Separators – Select the url-decoded parameter separator to be used from the drop-down list.

◾Values: Ampersand and Semicolon, Ampersand only, Semicolon only

◾Recommended: Ampersand only

d. Apply Double Decoding – Set to Yes to detect decoding of the character set after the completion of regular URL normalization. If decoding fails, the request is blocked in active mode and log gets generated in the BASIC > Web Firewall Logs page. In passive mode the request is allowed and also the logs get generated.

◾Recommended: No

4.Click Save.

## Configuring Global ACLs

Global ACLs (URL ACLs) are strict allow/deny rules shareable among multiple services configured on the Barracuda Web Application Firewall. They are associated with configured Security Policies.

**Steps To Configure Global ACLs**

1.Go to the SECURITY POLICIES > Global ACLs page.

2.Select the policy from the Policy Name drop-down list.

3.In the Create Global ACL section, specify values for the following:

a. URL ACL Name – Enter a name for the URL ACL.

b. URL Match – Enter a URL to be matched against the URL in the request. The URL should start with a "/" and can have at most one " \* " anywhere in the URL. Examples: /XXXX/Forms/\*, /images/\*.

c. Extended Match – Define an expression that consists of a combination of HTTP headers and/or query string parameters. This expression is used to match against special attributes in the HTTP headers or query string parameters in the requests. Use '\*' to denote "any request", that is, do not apply the Extended Match condition. For information on how to write extended match expression, see Extended Match Syntax Help.

d. Extended Match Sequence – Enter a number to indicate the order in which the extended match rule must be evaluated in the requests.

◾ Range:1 to 1000

◾ Default: 1

e. Action – Select the action from the drop-down list to be taken on the request matching this URL.

i. Process – Processes any request matching this ACL.

ii. Allow – Allows the request by disabling all security checks on an incoming request that matches the ACL. It also disables Data Theft on such responses.

iii. Deny and Log – Denies any request matching this ACL and also logs the event. The request is not subjected to any security policies. This is an unconditional Deny. When a request is denied, the Barracuda Web Application Firewall sends a cryptic error response.

iv. Deny with no Log – Same as Deny, but the event is not logged.

v. Temporary Redirect – Redirects the denied request with the 302 status code to the URL specified in the Redirect URL field.

vi. Permanent Redirect – Redirects the denied request with the 301 status code to the URL specified in the Redirect URL field.

f. Redirect URL – Specify a URL to which a user should be redirected if Action is set to Redirect.

g. Follow Up Action - Select the required follow up action to be taken whenever the request is denied.

h. Follow Up Action Time - Specify the time (sec) to block the client IP if Follow Up Action is set to Block Client-IP. The time can range between 1 to 600000 seconds.

4.Click Add.

## Configuring Action Policy

Action policy is a collection of settings that decide what action to be taken when a violation occurs. It consists of a set of attack groups and associated attack actions with it. The following attack groups are available:

•advanced-policy-violations

•application-profile-violations

•param-profile-violations

•protocol-violations

•request-policy-violations

•response-violations

•url-profile-violations

**Steps to Edit an Attack Action Policy**

1.Go to the SECURITY POLICIES > Global ACLs page.

2.Select the policy from the Policy Name drop-down list.

3.In the Action Policy section, identify the attack action and click Edit next to it. The Edit Attack Action window appears. Specify values for the following: a. Action – Select the action to be enforced when this attack is encountered.

i. Protect and Log – Blocks any request with the specified attack with a log message.

ii. Protect and no Log – Blocks any request with the specified attack with no log message.

iii. Allow and Log – Logs the request error.

iv. None – Allows the request by ignoring the violation.

b. Deny Response – Select the response to be sent to the client if the request is denied. A deny response is used when Action is set to Protect and Log or Protect and no Log.

i. Close Connection – Closes the connection to the client sending the invalid request.

ii. Send Response – Sends the specified response page for the denied request.

iii. Temporary Redirect – Redirects the request with the 302 status code to the URL specified in the Redirect URL field below.

iv. Permanent Redirect – Redirects the request with the 301 status code to the URL specified in the Redirect URL field below.

c. Redirect URL – Enter the URL to be used to redirect the request if the deny response is set to Redirect. The Redirect URL should be specified when the status-code in HTTP Status is one of 3xx redirect response codes.

d. Response Page – Select the response page to be sent to the client, if the parameter Deny Response is set to Send Response.

e. Follow Up Action – Select the follow up action to be taken if the request is denied.

◾ None – Allows the request by ignoring the violation.

◾ Block Client-IP – Determines whether you need to block any subsequent request from the same client for the time specified in Follow Up Action Time. Subsequent requests will be blocked for a specific service or for all services based on the configuration made in the Advanced Settings page.

◾ Challenge with CAPTCHA – Denies the response and any subsequent requests from the same client IP address will be tracked for the next 900 seconds, and will be challenged with a CAPTCHA image. The client will not be allowed to access any further resource until the CAPTCHA is answered. This is to thwart any reconnaissance efforts from the automated clients which are found to be suspicious due to such attack activity. The number of attempts for solving such a CAPTCHA challenge is five (5), and the number of re-fetches of the CAPTCHA image allowed is 128. Such tracked client IP addresses will have to answer the CAPTCHA if they are idle for more than 300 seconds. Note that the Follow Up Action Time has no relevance to this option.

a. Follow Up Action Time – Specify the time in seconds to block the client IP, if Follow Up Action is set to Block Client IP.

◾ Range: 1 to 600000

◾ Units: Seconds

4.Click Save.

# Standard Compliance & Enforcement

## Compliance Measures

If applicable, compliance with the above Standard can be measured by the following criteria. Example evidence will vary depending on any supporting guidelines implemented to support this Standard. The following list is not exhaustive, and all example evidence types may not be required to validate compliance.

Evidence of compliance can be presented in hard copy or electronic format.

|  |  |
| --- | --- |
| **Criteria** | **Example Evidence** |
| Review the web application firewall configuration settings | * Confirm from the standard if the configuration settings on the firewall are the same |

## Enforcement

All staff of XXXX must comply with all Information Security Standards. Failure to comply with these standards may result in disciplinary action in accordance with the current XXXX Human Resources policy. Disciplinary actions may include, but are not limited to:

* verbal and/or written warnings;
* instant dismissal; and
* actions by judicial and regulatory authorities.

# Exception Process / Glossary

## Exception Process

Non-compliance with the Standard statements described in this document must be reviewed and approved in accordance with the Exception Process defined in *XXXX-POL-ALL-001 - Information Security Policy Framework*.

## Glossary / Acronyms

|  |  |
| --- | --- |
| URL | Universal Resource Locator |
| VPN | Virtual Private Network |

# Document Management

## Document Revision Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Editor** | **Revision #** | **Description of Change** |
|  |  |  |  |

## Document Ownership

This Standard is owned by the YYYY

## Document Coordinator

This Standard is coordinated by the YYYY

## Document Approvers

|  |  |  |
| --- | --- | --- |
| **Approver Name** | **Signature** | **Date** |
|  |  |  |

## Distribution

* *Information Security*
* *IT Department*