Content Rewriting

https://campus.barracuda.com/doc/4259910/

This article applies to Barracuda Web Application Firewall Model 460 or higher.

The Barracuda Web Application Firewall allows you to rewrite selected content of requests and responses. This feature can be used to implement website cloaking and translation of URLs and headers in requests and responses. It can translate the internal codes, headers, and cookies so they are concealed from external users. Content rewriting allows you to configure address translation rules for application-specific packets sent through the Barracuda Web Application Firewall.

Configuring URL Translation

When a web server returns a URL, sensitive information about the web server may be revealed, which could be used to launch a variety of web attacks against the server. URL Translation modifies the prefix, domain, and response body of an internal URL to an externally viewable URL, thus preventing potential attacks.

URL Translation can externalize internal applications, which link to internal servers (not defined in the external DNS name space). For example, Company ABC has an internal application registered in the internal DNS as finance.abc. URL Translation can make this application available to external partners behind a common public domain such as *www.companyabc.com* without exposing the internal name space. Through URL Translation, Company ABC can map different internal and external prefixes so the internal application is available on the public Internet as *www.companyabc.com/finance.abc*. To configure URL Translation, go to **WEBSITES > Website Translations > URL Translations**. Click **Help** on that page for detailed configuration instructions.

Rewrite Rule Examples

In a rewrite rule directive, the two main arguments are: a URL pattern (which is a regular expression) and a replacement string (which contains references to the matched sub-strings). Every () in the regex pattern is a sub-string that can be referenced using $1, $2 and so on.

```
Example 1:
Old Value - /example/(.*).html
Rewrite Value - /modified/$1.aspx
With the above rule, an incoming URL such as "/example/dir/subdir/page.html"
invokes "/modified/dir/subdir/page.aspx" to be executed on the server.

Example 2:
Old Value - /Server/(.*)/other/(.*).html
```

```
Rewrite Value - /Myserver/$1/testing/$2.asp
With the above rule, an incoming URL such as "/Server/data/other/page.html"
invokes "/Myserver/data/testing/page.asp" to be executed on the server.

Example 3:
Old Value - /Server/(.*)/other/(.*).html
Rewrite Value - /Myserver/$1/testing/$2.asp
Rewrite Condition - (Client-IP eq 10.11.24.49)
With the above rule, an incoming URL such as "/Server/data/other/page.html"
invokes "/Myserver/data/testing/page.asp" to be executed on the server if the
request is from 10.11.24.49.
```

Configuring HTTP Request Rewrite

HTTP Request Rewrite allows incoming requests to be rewritten or redirected. Headers can be added, removed, or edited on the Barracuda Web Application Firewall before the request is forwarded to the backend server. The URL can be rewritten to map to a different resource. A redirect response can also be issued to the clients to point them to an updated location or resource. For example, Request Rewrite is used by default to relay the client IP address to the backend server (in Proxy mode) by inserting the header X-Forwarded-For with the value of the client IP. The backend server can extract and use this value. Similarly, authentication parameters (such as certificate details or username) can be forwarded by inserting request headers and using macros. See How to Pass Client Certificate Details to a Backend Server for more details. To configure HTTP Request Rewrite, use **WEBSITES > Website Translations > HTTP Request Rewrite**. For detailed configuration instructions, click **Help** on that page. To format a Request Rewrite condition, refer to the **Rewrite Condition Format** section.

If a client sends an "Accept-Encoding" header in the request, the response from the server may be compressed. In such case, the policies/rules mentioned below cannot be applied on the response. Therefore, a Request Rewrite rule to remove the "Accept-Encoding" header will be automatically added under **HTTP Request Rewrite** on the **WEBSITES > Website Translations** page when any of the following policies are enabled:

- CSRF and Hidden Parameter Protection
- Website Translations
- Web Scraping
- Data Theft Protection
- URL Protection
- URL Encryption
- Instant SSL
- Response Body Rewrite
- Adaptive Profiling

- DDoS

Configuring HTTP Response Rewrite

This policy sets rewrite rules for outbound responses. It allows you to add, delete, or rewrite headers. Response Rewrites are used for many purposes. For example, if a response includes a header listing the source IP address, Response Rewrite can delete that header, thereby preventing external users from seeing the actual IP address of the server.
To configure HTTP Response Rewrite, go to **WEBSITES > Website Translations > HTTP Response Rewrite**. For detailed configuration instructions, click **Help** on that page.

Configuring Request Rewrite and Response Rewrite

To configure a Request Rewrite and Response Rewrite rule, perform the following steps:

1. Go to the **WEBSITES > Website Translations** page, and in the **HTTP Request Rewrite** section or **HTTP Response Rewrite** section, specify values for the following fields:
   - **Rule Name** – Enter a name for the Request or Response Rewrite rule.
   - **Sequence Number**– Set the sequence number for the Request or Response Rewrite policy. This number determines the order of execution for multiple configured policies from highest (1) to lowest (1500).
   - **Action**– Set the action to: *Insert Header* - Inserts a header to the request. *Remove Header* - Removes the header from the request. *Rewrite Header* - Rewrites the value of the existing header in the request.
   - **Header Name** – Enter the relevant header name, for example X-Forwarded-For.
   - **Old Value** – Enter the initial request header to be rewritten if the action is *Rewrite Header*. An asterisk (*) rewrites all named headers, or specify the value or expression to be rewritten.
   - **Rewrite Value** – Enter the new value of the header to be rewritten when the **Action** is set to *InsertHeader* or *Rewrite Header*. Use the macros listed below to specify parameters from the client. When rewriting a header you can specify one or more fields using the separators such as colon (:), semicolon (;), space ( ) and comma (,). In Rewrite Value, the fields can be defined, for example: "Name=abc_cookie; Domain=example.com:Path=/". The rewrite value supports substring addressing of matches, i.e., the matching substrings can be referenced using $1,$2,...$n. See Supported Macros below for a list of macros supported for rewrite values**.**
   - **Rewrite Condition** – Set the condition under which a rewrite should occur. An asterisk (*) indicates there are no conditions (applies to all). Details on the format of the Rewrite Condition are explained below in Rewrite Condition Format.
2. Click **Add** to add the above settings.

Note: When multiple policies are configured, the request continues to be processed by other (higher sequence number) policies. If you wish to stop processing after a particular rule is matched, click **Edit** next to the rule and set **Continue Processing** to *No*.

Rewrite Condition Format

The request Rewrite Condition specifies when a rewrite should occur. The Rewrite Condition is made up of expressions combining Request Rewrite Tokens and Operations for Request Rewrite and Response Rewrite Conditionson those tokens for Request Rewrites. These expressions can then be joined with each other using logical or (or, OR, ||) or logical and (and, AND, &&). Examples of Rewrite Conditions: (Header User-Agent co mozilla), (URI rco /abc*html), (Client-IP eq 10.0.0.1), (Method eq POST). An asterisk indicates there are no conditions for rewrite, so the rewrite is done in every case.

Request Rewrite Tokens

These tokens can be used in a Request Rewrite condition:

- **Header**: The HTTP header in the request. The word **Header** precedes the name of the relevant header or * to indicate all headers. Examples: Header Accept co soap, Header Soap-Action ex.
- **Client-IP**: The IP address of the client sending the request. The IP address can be either a host IP address or a subnet specified by a subnet mask. Only operations EQ and NEQ can be combined with this token. Examples: Client-IP eq 192.168.1.0/24 (subnet qualified by a netmask) Client-IP eq 192.168.1.10 (host IP address)
- **URI**: The Uniform Resource Identifier of the resource on which to apply the rule. Example: URI rco /abc*html
- **Method**: The HTTP method in the request. Example: Method eq GET
- **Http-Version**: The HTTP protocol version of the request. Example: HTTP-Version eq HTTP/1.1
- **Parameter**: The query part of the URL that is passed to the servers as a name-value pair. In addition, the word "$NONAME_PARAM" can be used when the parameter name is absent. Examples: Parameter sid eq 1234, Parameter $NONAME_PARAM co abcd
- **Pathinfo**: The portion of URL that contains extra information about the path of the resource on the server. Example: `pathinfo rco abc*`

Response Rewrite Tokens

These tokens can be used in a Response Rewrite condition:

- **Header**: The HTTP header in the request. The word **Header** precedes the name of the relevant

header or * to indicate all headers. Examples: Header Accept co soap, Header Soap-Action ex.
- **Response-Header**: An HTTP header on the response path. The term "Response-Header" should be followed by the name of the header on which the action is to be applied. Example: Response-Header Set-Cookie co sessionid.
- **Status-Code**: The status code of the response returned by the servers. Example: Status-Code eq 200

Operations for Request Rewrite and Response Rewrite Conditions

These operations can be combined with Request Rewrite tokens and Response Rewrite tokens in a Request or Response Rewrite condition:

- **contains, CONTAINS, co, CO** – Token contains the given value.
- **ncontains, nCONTAINS, nco, nCO** – Token does not contain the given value.
- **rcontains, rCONTAINS, rco, rCO** – Token contains the given value that is interpreted as a regular expression.
- **equals, EQUALS, eq, EQ** – Token equals the given value.
- **nequals, nEQUALS, neq, nEQ** – Token does not equal the given value.
- **requals, rEQUALS, req, rEQ** – Token equals the given value interpreted as a regular expression.
- **exists, EXISTS, ex, EX** – Token exists.
- **nexists, nEXISTS, nex, nEX** – Token does not exist.

Configuring Response Body Rewrite

This policy sets the rule for searching and replacing any text string in the response body. Only responses whose content type begins with text/ can be searched, including text/html, text/plain, text/javascript, text/css, text/xml. Neither flash nor applet content can be searched. The search and replace strings should be text rather than regular expressions. Meta-characters cannot be used, such as \r or \n in either search or replace, which means you cannot search and replace any multi-byte charset strings.

To configure Response Body Rewrite, go to **WEBSITES > Website Translations > Response Body Rewrite**. For detailed configuration instructions, click **Help** on that page.

Supported Macros

For Request Rewrites:

The following macros are supported *ONLY* for Request Rewrite Values:

- **$SRC_ADDR** – Inserts the source (client) IP address. You can use it for the new value (Rewrite Value parameter) when inserting or rewriting a header
- **$URI** Should be specified in the new value, if you are rewriting or redirecting the URI. $URI specifies the complete request URI including the query string.
- **$X509_VERSION** – The client certificate's X.509 version string.
- **$X509_SERIAL_NUMBER** – The serial number of the client certificate.
- **$X509_SIGNATURE_ALGORITHM** – The signature algorithm used in the client certificate.
- **$X509_ISSUER** – The issuer string of the client certificate.
- **$X509_NOT_VALID_BEFORE** – Time before which the client certificate is not valid.
- **$X509_NOT_VALID_AFTER** – Time after which the client certificate is not valid.
- **$X509_SUBJECT** – The subject string of the client certificate.
- **$X509_SUBJECT_PUBLIC_KEY_TYPE** – The X.509 Certificate Subject Key Identifier string of the client certificate.
- **$X509_SUBJECT_PUBLIC_KEY** – Public key modulus of the client certificate.
- **$X509_SUBJECT_PUBLIC_KEY_RSA_BITS** – Size of the client certificate's public key, in bits.
- **$X509_EXTENSIONS** – The client certificate's X.509 Extensions string.
- **$X509_HASH** – The X.509 Hash string of the client certificate.
- **$X509_WHOLE** – The X.509 client certificate, represented as a string in PEM format.
- **X509_SAN_EMAIL and X509_IAN_EMAIL** – Macros copy the information of an email from the client certificate and send it back to the backend server in HTTP header.
- **$AUTH_USER** – Adds the username.*
- **$AUTH_PASSWD** – Adds the password.*
- **$AUTH_GROUPS** – Adds the user roles.*
- **$LOG_UID** – The unique ID used to identify a log.
- **$COUNTRY_CODE** - The two-letter country code of the location from where the client is sending the request.

*Notes for specific situations:

  - The URL is not protected; access control or authentication is off. The value substituted for the above three macros will be the special string **NCURLNotProtected**.
  - The client has not logged in. The value substituted for the above three macros will be the special string **NCNoUserSession**.
  - The user does not belong to any groups. The value substituted for **$AUTH_GROUPS** will be the special string **NCNOUserRoles**.

For Response Page

- **%action-id** – The attack ID of the violation that resulted in displaying this response page.

- **%host** – The host that sent this request.
- **%s** – The URL of the request that caused this violation.
- **%client-ip** – The client IP address of the request that caused the violation.
- **%attack-time** – The time at which the violation occurred.
- **%attack-name** – The attack name of the violation that resulted in displaying the response page.

Default Rewrite Rules

Type of Rule	Request Rewrite
Rule Action	Insert Header
Name of Header	X-Forwarded-For
Value	$SRC_ADDR
Rule Description	• This rule is added for every virtual service of type HTTP, HTTPS, and Instant SSL on the Barracuda WAF. • The rule inserts the “X-Forwarded-For” header into the requests sent to the backend servers configured under the virtual service. • The header value is the client’s IP address from the incoming TCP connection.
Reason	By default, the Barracuda WAF terminates the TCP/IP connection for the original request and creates a new request using the Barracuda WAF’s interface IP address as the new source IP address. This rule is useful for extracting the actual client IP address from the client request and inserting it as an HTTP header in the request sent to the backend server.
Type of Rule	Request Rewrite
Rule Action	Remove Header
Name of Header	Accept-Encoding
Rule Description	• This rule is added for every virtual service of type HTTP, HTTPS, and Instant SSL on the Barracuda WAF. • The rule removes the “Accept-Encoding” header and its values from the HTTP request sent to the backend servers configured under the virtual service.
Reason	The Accept-Encoding request HTTP header indicates the content encoding to the client. This is usually the compression algorithm supported by the client. The server uses this information to select the encoding mechanism for compressing response data. When this request header is removed from the request sent, the server sends the response data without compression. This enables the Barracuda WAF to do the following: • HTTP response body checks Data Theft Protection • Response body rewrite tasks used in features such as: ◦ Website Translations > Response Body Rewrites ◦ CSRF attack prevention in HTML forms ◦ Inserting JS challenge for web scraping ◦ Inserting JS for client fingerprinting ◦ Inserting JS challenge for identifying suspicious client in application DDOS