ASA/PIX 8.x: Block Certain Websites (URLs) Using Regular Expressions With MPF Configuration Example

Contents

Introduction
Prerequisites
Requirements
Components Used
Related Products
Conventions
Background Information
Modular Policy Framework Overview
Regular Expression
Configure
Network Diagram
Configurations
ASA CLI Configuration
ASA Configuration 8.x with ASDM 6.x
Verify
Troubleshoot
Related Information

Introduction

This document describes how to configure the Cisco Security Appliances ASA/PIX 8.x that uses Regular Expressions with Modular Policy Framework (MPF) in order to block the certain websites (URLs).

Note: This configuration does not block all application downloads. For reliable file blocking, a dedicated appliance such as Ironport S Series or a module such as the CSC module for the ASA should be used.

Note: HTTPS filtering is not supported on ASA. ASA cannot do deep packet inspection or inspection based on regular expression for HTTPS traffic, because in HTTPS, content of packet is encrypted (SSL).

Prerequisites

Requirements
This document assumes that Cisco Security Appliance is configured and works properly.

**Components Used**

- Cisco 5500 Series Adaptive Security Appliance (ASA) that runs the software version 8.0(x) and later
- Cisco Adaptive Security Device Manager (ASDM) version 6.x for ASA 8.x

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

**Related Products**

This configuration can also be used with the Cisco 500 Series PIX that runs the software version 8.0(x) and later.

**Conventions**

Refer to the [Cisco Technical Tips Conventions](#) for more information on document conventions.

**Background Information**

**Modular Policy Framework Overview**

MPF provides a consistent and flexible way to configure security appliance features. For example, you can use MPF to create a timeout configuration that is specific to a particular TCP application, as opposed to one that applies to all TCP applications.

MPF supports these features:

- TCP normalization, TCP and UDP connection limits and timeouts, and TCP sequence number randomization
- CSC
- Application inspection
- IPS
- QoS input policing
- QoS output policing
- QoS priority queue

The configuration of the MPF consists of four tasks:

1. Identify the Layer 3 and 4 traffic to which you want to apply actions. Refer to [Identifying Traffic Using a Layer 3/4 Class Map](#) for more information.
2. (Application inspection only) Define special actions for application inspection traffic. Refer to [Configuring Special Actions for Application Inspections](#) for more information.
3. Apply actions to the Layer 3 and 4 traffic. Refer to [Defining Actions Using a Layer 3/4 Policy Map](#) for more information.
**Regular Expression**

A regular expression matches text strings either literally as an exact string, or by the use of metacharacters so you can match multiple variants of a text string. You can use a regular expression to match the content of certain application traffic; for example, you can match a URL string inside an HTTP packet.

**Note:** Use Ctrl+V in order to escape all of the special characters in the CLI, such as question mark (?) or a tab. For example, type `d[Ctrl+V]?g` in order to enter `d?g` in the configuration.

For the creation of a regular expression, use the `regex` command, which can be used for various features that require text matching. For example, you can configure special actions for application inspection with the use of the Modular Policy Framework that uses an inspection policy map. Refer to the `policy map type inspect` command for more information. In the inspection policy map, you can identify the traffic you want to act upon if you create an inspection class map that contains one or more `match` commands or you can use `match` commands directly in the inspection policy map. Some `match` commands let you identify text in a packet using a regular expression; for example, you can match URL strings inside HTTP packets. You can group regular expressions in a regular expression class map. Refer to the `class-map type regex` command for more information.

This table lists the metacharacters that have special meanings.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Dot</td>
<td>Matches any single character. For example, <code>d.g</code> matches dog, dag, dtg, and any word that contains those characters, such as doggonnit.</td>
</tr>
<tr>
<td>(exp)</td>
<td>Subexpression</td>
<td>A subexpression segregates characters from surrounding characters, so that you can use other metacharacters on the subexpression. For example, `d(o</td>
</tr>
<tr>
<td></td>
<td>Altarneation</td>
<td>Matches either expression it separates. For example, `dog</td>
</tr>
<tr>
<td>?</td>
<td>Question mark</td>
<td>A quantifier that indicates that there are 0 or 1 of the previous expression. For example, <code>lo?se</code> matches lse or lose. <strong>Note:</strong> You must enter Ctrl+V and then the question mark or else the help function is invoked.</td>
</tr>
<tr>
<td>*</td>
<td>Asterisk</td>
<td>A quantifier that indicates that there are 0, 1 or any number of the previous expression. For example, <code>lo*se</code> matches</td>
</tr>
<tr>
<td>Pattern</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><code>{x}</code></td>
<td>Repeat quantifier. Repeat exactly x times. For example, <code>ab(xy){3}z</code> matches <code>abxyxyxyz</code>.</td>
<td></td>
</tr>
<tr>
<td><code>{x,}</code></td>
<td>Minimum repeat quantifier. Repeat at least x times. For example, <code>ab(xy){2,}z</code> matches <code>abxyx</code>, <code>abxyxyxy</code>, and so forth.</td>
<td></td>
</tr>
<tr>
<td><code>[abc]</code></td>
<td>Character class. Matches any character in the brackets. For example, <code>[abc]</code> matches a, b, or c.</td>
<td></td>
</tr>
<tr>
<td><code>[^abc]</code></td>
<td>Negated character class. Matches a single character that is not contained within the brackets. For example, <code>[^abc]</code> matches any character other than a, b, or c. <code>[^A-Z]</code> matches any single character that is not an uppercase letter.</td>
<td></td>
</tr>
<tr>
<td><code>[a-c]</code></td>
<td>Character range class. Matches any character in the range. <code>[a-z]</code> matches any lowercase letter. You can mix characters and ranges: <code>[abcq-z]</code> matches a, b, c, q, r, s, t, u, v, w, x, y, z, and so does <code>[a-cq-z]</code>. The dash (-) character is literal only if it is the last or the first character within the brackets: <code>[abc-]</code> or <code>[-abc]</code>.</td>
<td></td>
</tr>
<tr>
<td><code>&quot;&quot;&quot;</code></td>
<td>Quotation marks. Preserves trailing or leading spaces in the string. For example, <code>&quot; test&quot;</code> preserves the leading space when it looks for a match.</td>
<td></td>
</tr>
<tr>
<td><code>^</code></td>
<td>Caret. Specifies the beginning of a line.</td>
<td></td>
</tr>
<tr>
<td><code>\</code></td>
<td>Escape character. When used with a metacharacter, matches a literal character. For example, <code>\[</code> matches the left square bracket.</td>
<td></td>
</tr>
<tr>
<td><code>char</code></td>
<td>Character. When character is not a metacharacter, matches the literal character.</td>
<td></td>
</tr>
<tr>
<td><code>\r</code></td>
<td>Carriage return. Matches a carriage return 0x0d.</td>
<td></td>
</tr>
<tr>
<td><code>\n</code></td>
<td>Newline. Matches a new line 0x0a.</td>
<td></td>
</tr>
<tr>
<td><code>\t</code></td>
<td>Tab. Matches a tab 0x09.</td>
<td></td>
</tr>
<tr>
<td><code>\f</code></td>
<td>Formfeed. Matches a form feed 0x0c.</td>
<td></td>
</tr>
<tr>
<td><code>\xNN</code></td>
<td>Escape hexadecimal number. Matches an ASCII character that uses a hexadecimal that is exactly two digits.</td>
<td></td>
</tr>
<tr>
<td>Escape octal number</td>
<td>Matches an ASCII character as octal that is exactly three digits. For example, the character 040 represents a space.</td>
<td></td>
</tr>
</tbody>
</table>

**Configure**

In this section, you are presented with the information to configure the features described in this document.

**Note:** Use the [Command Lookup Tool](https://example.com) ([registered](https://example.com) customers only) in order to obtain more information on the commands used in this section.

**Network Diagram**

This document uses this network setup:

![Network Diagram](image.png)

**Configurations**

This document uses these configurations:

- ASA CLI Configuration
- ASA Configuration 8.x with ASDM 6.x

**ASA CLI Configuration**

![ASA CLI Configuration](image.png)
ciscoasa#show running-config : Saved : ASA Version 8.0(2) ! hostname ciscoasa domain-name
default.domain.invalid enable password 8Ry2YjIyt7RRXU24 encrypted names ! interface Ethernet0/0 nameif inside security-level 100 ip address 10.1.1.1 255.255.255.0 ! interface Ethernet0/1 nameif outside security-level 0 ip address 192.168.1.1 255.255.255.0 ! interface Ethernet0/2 nameif DMZ security-level 90 ip address 10.77.241.142 255.255.255.192 ! interface Ethernet0/3 shutdown no nameif no security-level no ip address ! interface Management0/0 shutdown no nameif no security-level no ip address ! passwd 2KFQnbNIdI.2KYOU encrypted regex urllist1 ".\.*\(\[Re\][Xx]\[Re\]|\[Cc]\[Oo]\[Mm]\]|\[Bb]\[Aa]\[Tt]\]\)
HTTP/l.\{01\}" !--- Extensions such as .exe, .com, .bat to be captured and !--- provided the http version being used by web browser must be either 1.0 or 1.1 regex urllist2 ".\.*\(\[Pp]\[Ii]\[Ff]\]\|\[Vv]\[Bb]\[Ss]\]\|\[Ww]\[Ss]\[Hh]\]\)
HTTP/l.\{01\}" !--- Extensions such as .pif, .vbs, .wsh to be captured !--- and provided the http version being used by web browser must be either !--- 1.0 or 1.1 regex urllist3 ".\.*\(\(\[Dd]\[Oo]\[Cc]\]\|\[Xx]\[Ll]\[Ss]\]\|\[Pp]\[Pp]\[Tt]\]\)
HTTP/l.\{01\}" !--- Extensions such as .doc(word), .xls(ms-excel), .ppt to be captured and provided !--- the http version being used by web browser must be either 1.0 or 1.1 regex urllist4 ".\.*\(\[Zz]\[Ii]\[Pp]\]\|\[Tt]\[Aa]\[Rr]\]\|\[Tt]\[Gg]\[Zz]\)
HTTP/l.\{01\}" !--- Extensions such as .zip, .tar, .tgz to be captured and provided !--- the http version being used by web browser must be either 1.0 or 1.1 regex domainlist1 "\.[\w]+\.com" regex domainlist2 "\.[\w]+\.myspace\.com" regex domainlist3 "\.[\w]+\.youtube\.com" !--- Captures the URLs with domain name like yahoo.com, !--- youtube.com and myspace.com regex contenttype "Content-Type" regex applicationheader "application/.*" !--- Captures the application header and type of !--- content in order for analysis boot system disk0:/asa802-k8.bin ftp mode passive dns server-group DefaultDNS domain-name default.domain.invalid access-list inside_mpc extended permit tcp any any eq www access-list inside_mpc extended permit tcp any any eq 8080 !--- Filters the http and port 8080 !--- traffic in order to block the specific traffic with regular !--- expressions pager lines 24 mtu inside 1500 mtu outside 1500 mtu DMZ 1500 no failover icmp unreachable rate-limit 1 burst-size 1 asdm image disk0:/asdm-602.bin no asdm history enable arp timeout 14400 route DMZ 0.0.0.0 0.0.0.0 10.77.241.129 1 timeout xlate 3:00:00 timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 icmp 0:00:02 timeout sunrpc 0:10:00 h323 0:05:00 h225 1:00:00 mgcp 0:05:00 mgcp-pat 0:05:00 timeout sip 0:30:00 sip_media 0:02:00 sip-invite 0:03:00 sip-disconnect 0:02:00 timeout uauth 0:05:00 absolute dynamic-access-policy-record DfltAccessPolicy http server enable http 0.0.0.0 0.0.0.0 DMZ no snmp-server location no snmp-server contact snmp-server enable traps snmp authentication linkup linkdown coldstart no crypto isakmp nat-traversal telnet timeout 5 ssh timeout 5 console timeout 0 threat-detection basic-threat threat-detection statistics access-list ! class-map type regex match-any DomainBlockList match regex domainlist1 match regex domainlist2 match regex domainlist3 !--- Class map created in order to match the domain names !--- to be blocked class-map type inspect http match-all
ASA Configuration 8.x with ASDM 6.x

Complete these steps in order to configure the regular expressions and apply them into MPF to block the specific websites as shown.

1. Create Regular Expressions
   Choose Configuration > Firewall > Objects > Regular Expressions and click Add under the tab Regular Expression in order to create regular expressions as shown. Create a regular expression domainlist1 in order to capture the domain name yahoo.com. Click OK.
Create a regular expression **domainlist2** in order to capture the domain name **myspace.com**. Click OK.

Create a regular expression **domainlist3** in order to capture the domain name **youtube.com**. Click OK.

Create a regular expression **urllist1** in order to capture the file extensions such as **exe**, **com** and **bat** provided that the http version being used by web browser must be either 1.0 or 1.1. Click
OK. Create a regular expression `urllist2` in order to capture the file extensions such as `.pif`, `.vbs` and `.wsh` provided that the http version being used by web browser must be either 1.0 or 1.1. Click

OK. Create a regular expression `urllist3` in order to capture the file extensions such as `.doc`, `.xls` and `.ppt` provided that the http version being used by web browser must be either 1.0 or 1.1. Click

OK. Create a regular expression `urllist4` in order to capture the file extensions such as `.zip`, `.tar` and `.tgz` provided that the http version being used by web browser must be either 1.0 or 1.1. Click
OK. Create a regular expression `contenttype` in order to capture the content type. Click

OK. Create a regular expression `applicationheader` in order to capture the various application header. Click

OK. Equivalent CLI Configuration

2. Create Regular Expression Classes Choose Configuration > Firewall > Objects > Regular Expressions and click Add under the tab Regular Expression Classes in order to create the various classes as shown. Create a regular expression class `DomainBlockList` in order to match any of the regular expressions domainlist1, domainlist2 and domainlist3. Click OK.
Create a regular expression class **URLBlockList** in order to match any of the regular expressions urllist1, urllist2, urllist3 and urllist4. Click **OK**.
Equivalent CLI Configuration

3. Inspect the identified traffic with Class maps

Choose Configuration > Firewall > Objects > Class Maps > HTTP > Add in order to create a class map to inspect the http traffic identified by various regular expressions as shown. Create a class map AppHeaderClass in order to match the response header with regular expressions captures.
Click OK Create a class map **BlockDomainsClass** in order to match the request header with regular expressions captures.
Click **OK.** Create a class map **BlockURLsClass** in order to match the request uri with regular expressions.

Click **OK.**

4. **Equivalent CLI Configuration**

Choose **Configuration > Firewall > Objects > Inspect Maps > HTTP** in order to create a **http_inspection_policy** to set the action for the matched traffic as shown. Click **OK.**
Choose **Configuration > Firewall > Objects > Inspect Maps > HTTP > http_inspection_policy (double click)** and click **Details > Add** in order to set the actions for the various Classes created so far.

Set the action as **Drop Connection** and **Enable** the logging for the Criterion as Request Method and Value as
Connect. Click OK. Set the action as **Drop Connection** and **Enable** the logging for the class **AppHeaderClass**.
Click OK. Set the action as **Reset** and **Enable** the logging for the class **BlockDomainsClass**.
Set the action as **Reset** and **Enable** the logging for the class BlockURLsClass. Click **OK**.

Equivalent CLI Configuration

5. **Apply the inspection http policy to the interface** Choose **Configuration > Firewall > Service Policy Rules > Add > Add Service Policy Rule.**

HTTP Traffic Choose the **Interface** radio button with inside interface from the drop down menu and Policy Name as **inside-policy**. Click **Next.**
Create a class map `httptraffic` and check the **Source** and **Destination IP Address** (uses ACL). Click **Next**.
Choose the Source and Destination as any with service as tcp-udp/http. Click Next.
Check the **HTTP** radio button and click
Configure. Check the radio button **Select a HTTP inspect map for the control over inspection** as shown. Click
Port 8080 Traffic Again, choose Add > Add Service Policy Rule.
Choose the radio button **Add rule to existing traffic class** and choose **httptraffic** from the drop down menu. Click **Next**.
Choose the Source and Destination as any with **tcp/8080**. Click **Next**.
Click Finish.
Click **Apply**. **Equivalent CLI Configuration**

**Verify**

Use this section in order to confirm that your configuration works properly.
The **Output Interpreter Tool** (registered customers only) (OIT) supports certain **show** commands. Use the OIT to view an analysis of **show** command output.

- **show running-config regex** — Shows the regular expressions that have been configured.
  ```
  ciscoasa#show running-config regex regex urllist1 ".*\([Ee][_EQe][Ee]\|[Cc][Qc][Cc]\|\[Bb][Aa][Tt]\) HTTP/1.0" regex urllist2 ".*\([Pp][Ii][Ff]\|\[Vv][Bb][Ss]\|\[Ww][Ss]\) HTTP/1.0" regex urllist3 ".*\([Zz][Ii][Pp]\|\[Tt][Aa][Rr]\|\[Tt][Gg][Zz]\) HTTP/1.0" regex urllist4 ".*\([Dd][Oo][Cc]\|\[Xx][Ll][Ss]\|\[Pp][Pp][Tt]\) HTTP/1.0" regex domainlist1 "\.yahoo\..*" regex domainlist2 "\.myspace\..*" regex domainlist3 "\.youtube\..*" regex contenttype "Content-Type" regex applicationheader "application/.*"
  ```

- **show running-config class-map** — Shows the class maps that have been configured.
  ```
  DomainBlockList match regex domainlist1 match regex domainlist2 match regex domainlist3 class-map type inspect http match-all BlockDomainsClass match regex domainlist1 match regex domainlist2 match regex domainlist3 match regex domainlist4 class-map inspection_default match default-inspection-traffic class-map type inspect http match-all AppHeaderClass match response header regex contenttype regex applicationheader class-map httptraffic match access-list inside_mpc class-map type inspect http match-all BlockURLsClass match request uri regex class URLBlockList !
  ```

- **show running-config policy-map type inspect http** — Shows the policy maps that inspects the http traffic that have been configured.
  ```
  http ! policy-map type inspect http http_inspection_policy parameters protocol-violation action drop-connection class AppHeaderClass drop-connection log match request method connect drop-connection log class BlockDomainsClass reset log class BlockURLsClass reset log !
  ```

- **show running-config policy-map** — Displays all the policy-map configurations as well as the default policy-map configuration.
  ```
  policy-map type inspect dns preset_dns_map parameters message-length maximum 512 policy-map type inspect http http_inspection_policy parameters protocol-violation action drop-connection class AppHeaderClass drop-connection log match request method connect drop-connection log class BlockDomainsClass reset log class BlockURLsClass reset log policy-map global_policy class inspection_default inspect dns preset_dns_map inspect ftp inspect h323 h225 inspect h323 ras inspect netbios inspect rsh inspect rtsp inspect skinny inspect smtp inspect sgnet inspect sunrpc inspect tftp inspect sip inspect xdmcp policy-map policy inside-pc policy httptraffic inspect http_inspection_policy !
  ```

- **show running-config service-policy** — Displays all currently running service policy configurations.
  ```
  service-policy service-policy global_policy global service-policy inside-policy interface inside
  ```

- **show running-config access-list** — Displays the access-list configuration that runs on the security appliance.
  ```
  access-list inside_mpc extended permit tcp any eq www access-list inside_mpc extended permit tcp any eq 8080
  ```

**Troubleshoot**

This section provides information you can use to troubleshoot your configuration.

**Note:** Refer to **Important Information on Debug Commands** before you use **debug** commands.

- **debug http** — Shows the debug messages for HTTP traffic

**Related Information**
- Cisco ASA 5500 Series Adaptive Security Appliances Support
- Cisco Adaptive Security Device Manager (ASDM) Support
- Cisco PIX 500 Series Security Appliances Support
- Cisco PIX Firewall Software
- Cisco Secure PIX Firewall Command References
- Security Product Field Notices (including PIX)
- Requests for Comments (RFCs) 📖
- Technical Support & Documentation - Cisco Systems