Custom Application Signatures

Tech Note

PAN-OS 4.1

Revision A

Contents

Overview

One of the fundamental requirements of a next-generation firewall is to identify and control applications on any port, not just standard ports. Palo Alto Networks firewalls not only identify a large number of applications, they also provide a flexible web-based interface, which enables administrators to develop custom signatures to identify any application, whether it is web-based or a client-server application.

Why Custom App-IDs

- To identify proprietary applications.
- To achieve granularity of visibility and control over traffic particular to your environment. If your traffic is classified as unkown-tcp/udp, HTTP or SSL, you could bring visibility by developing custom App-IDs.
- To identify ephemeral apps with topical interest.
    - Ex: ESPN3-Video for soccer world cup, March Madness, Wikileaks.
- To identify nested applications.
    - Further Identify Facebook-apps – Farmville, chat, marketplace, etc.
- To perform QoS for your specific application.
- URL filtering is incapable of providing control to administrators on websites that replicate on a different host, emulating the same look-n-feel as well as content. Example: wikileaks.com

Palo Alto Networks firewalls give you the ability to develop custom application signatures to address the above mentioned scenarios, as well as many others. When custom application signatures are written and used in a policy, the session will track the application as it changes.

Example: The user may start with simple web browsing, but then may upload data using the same web application. In this case, the traffic will be classified as 'uploading' or may download data using the same application in which case the traffic will be classified as 'downloading'.

Objectives

The intent of this Tech Note is to showcase the application signature development methodology using the web based tool built into the Palo Alto Networks firewall management platform. With the help of a packet capture tool like Wireshark, you can then fine tune your signature.

Signatures for Custom App-IDs

Palo Alto Networks firewalls support this feature as of PAN-OS 3.0. In this Tech Note, we will show you how to research and develop signatures for HTTP-based apps to detect and control an application.

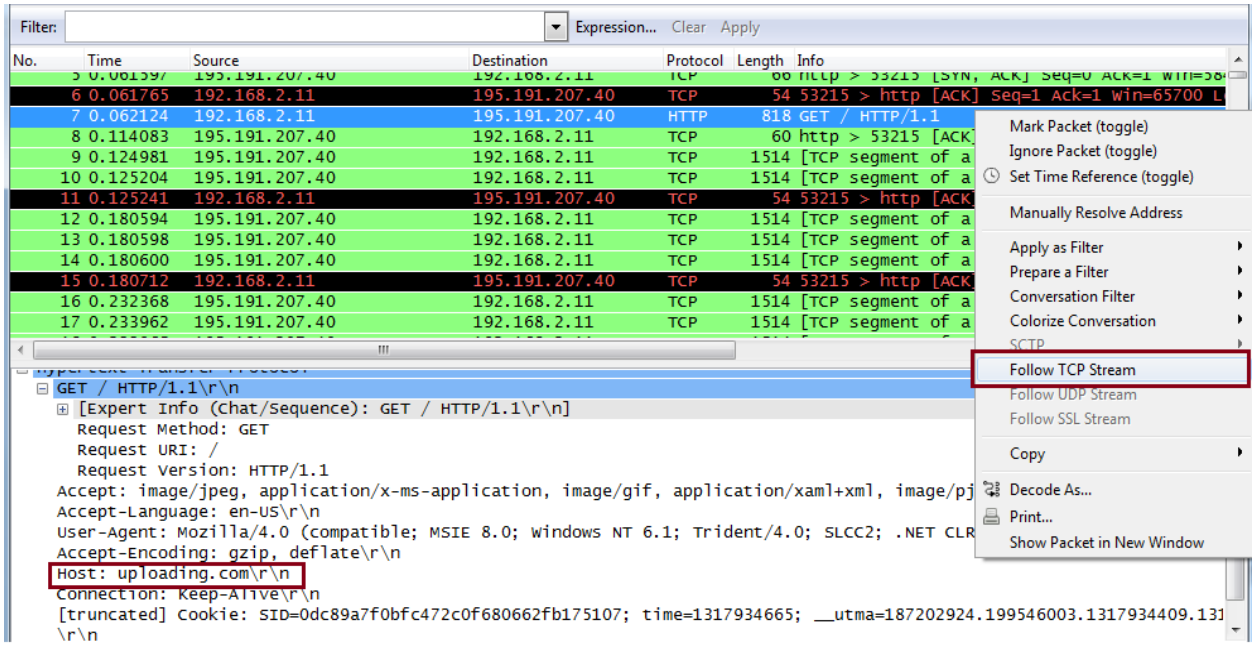
In this example, we use the web-based application at uploading.com. Our objective is not to block the site completely but rather provide ability to download data but restrict upload. The firewall has a built in application signature to identify uploading.com, so in this example, we will show you how to expand on that signature and develop signatures to control the ability to upload while enabling downloads.

Research the Application

- Use packet capture and analyzer tools to research the application. You have an option to perform packet capture on the firewall itself or on a client PC. In this example, we will use Wireshark as our packet capture an analyzer tool on a client PC.
- Start Wireshark protocol analyzer to capture the packets between the client browser and uploading.com server and then locate a packet in the session. Multiple sessions might be created for different application scenarios.

Your PCAP must capture all those sessions created by your application and a signature is required for each type of scenario.

- In this scenario, we have selected HTTP GET request. Right click and select *Follow TCP stream*.



- The Follow TCP Stream window will appear. Examine the output for information that will positively identify this particular application. Here are the first couple packets of that data exchange. The content in red is the HTTP request while the content in blue color is the HTTP response data. The GET and Response together forms a HTTP transaction.

- If we were to write a signature to detect the "uploading.com" application, we would create a signature that searches for the "uploading.com" pattern in HTTP request host header context.

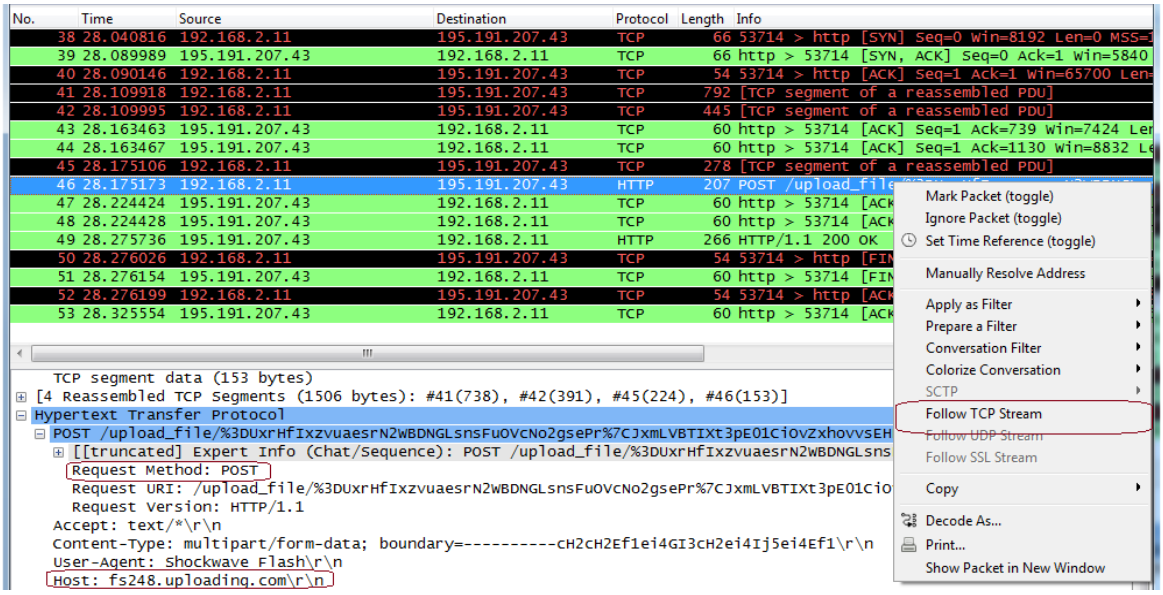
Operator	Context	Pattern
pattern-match	http-req-host-header	uploading\.com

- On the firewall, check if the application shows up as uploading. We will then use uploading.com as our base application and expand on this base application to build signatures to control downloading and uploading.

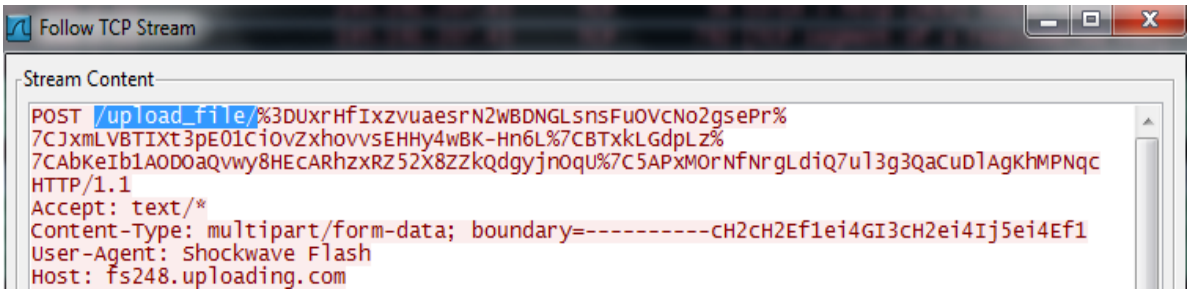
Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule
10/06 14:16:49	end	l3-trust	l3-untrust	192.168.2....	paloaltonetwork\...	195.191.207.40	80	uploading	allow	rule1

Identify Patterns for Upload and Download

1 To identify the signatures for uploading, start Wireshark protocol analyzer to capture the packets between the browser (client) and uploading.com server when you initiate an upload. Look for the HTTP POST request packets in Wireshark, as follows:

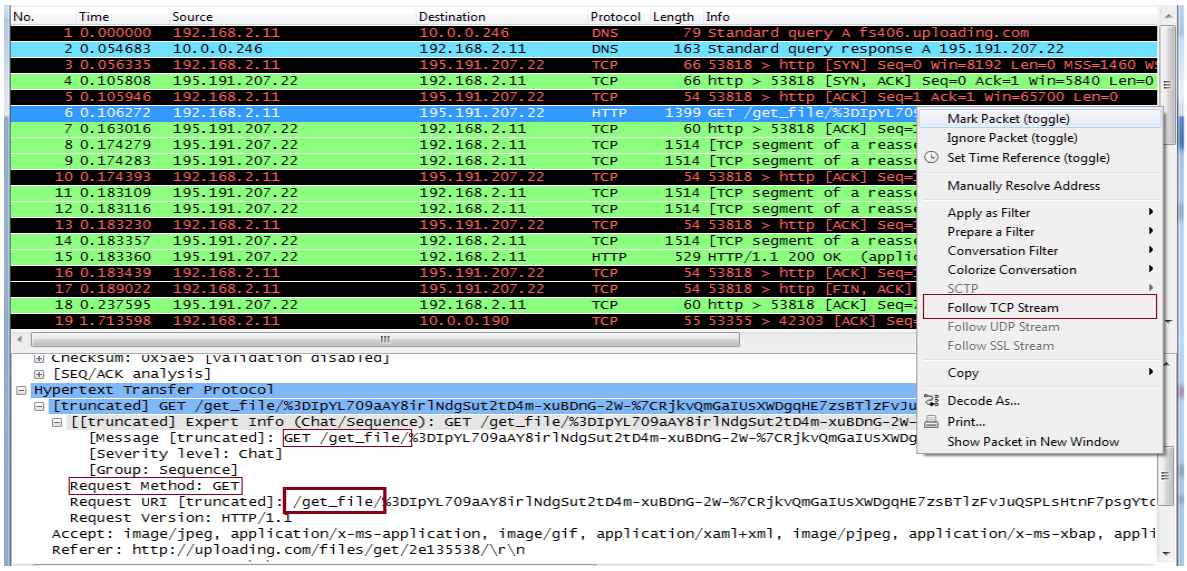


2 Locate the POST packet in the session. In this scenario I have selected HTTP POST request. Right click and select *Follow TCP stream*.

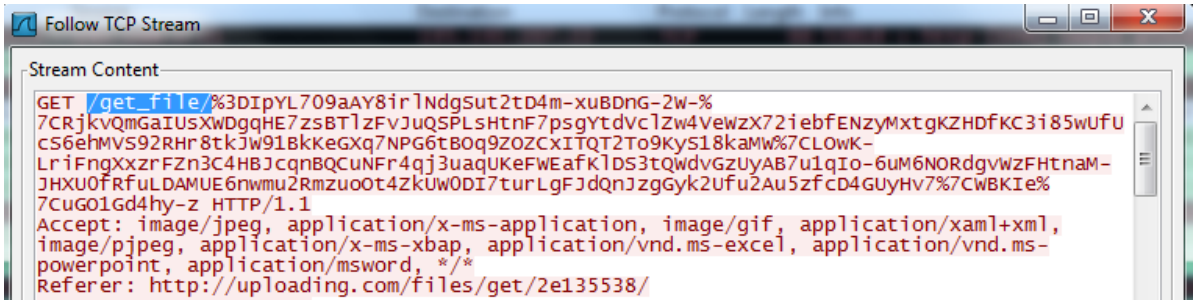


3 In the above packet capture, the most obvious pattern is /upload_file/ in http-method: POST.

4 Identify the signature for downloading by starting Wireshark protocol analyzer to capture the packets between the browser and uploading.com server when you initiate a download. Look for the HTTP GET request packets in Wireshark, as follows:



5 Locate the GET packet in the session. In this scenario we have selected HTTP GET request. Right click and select *Follow TCP stream*.



6 In the above packet capture, the most obvious pattern is /get_file/ in HTTP method GET.

Context Definitions

Please refer to the following document for a list of available contexts, their definitions and examples. Custom Signature Contexts

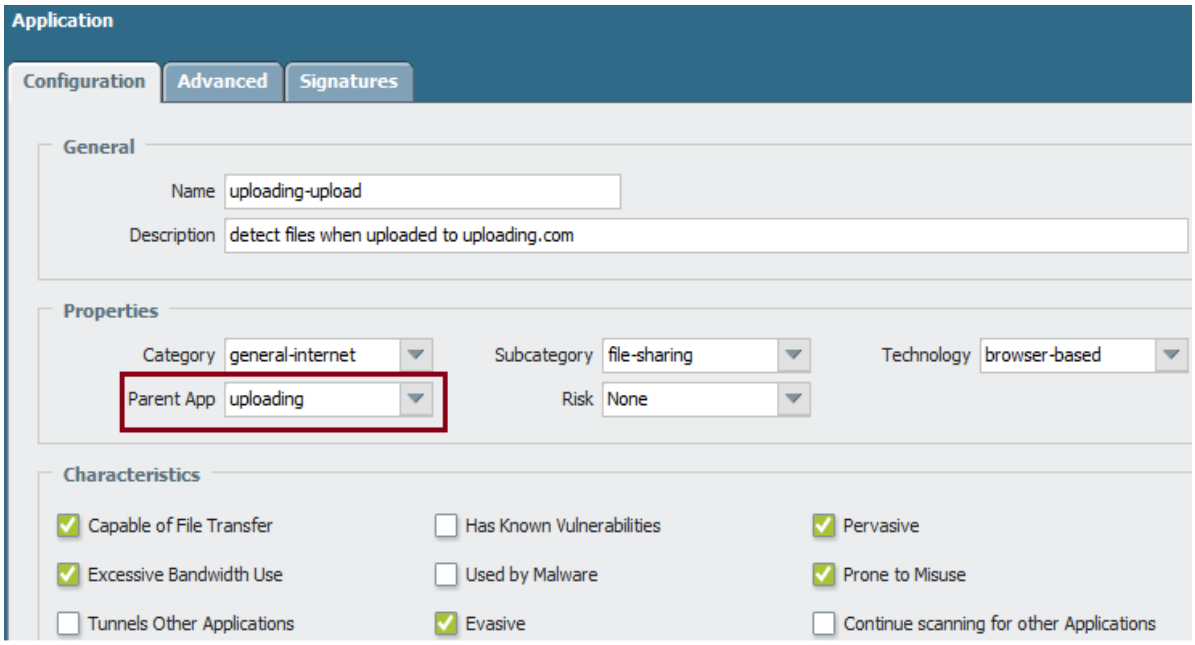
Decoders with Custom Contexts

PAN-OS S/W provides the ability to develop signatures for contexts within the following protocols. New decoders and contexts are added in weekly content releases.

FTP	HTTP	IMAP	SMTP	MS-RPC	SMB	MS-SQL
RTSP	SSH	TELNET	File body	Unknown TCP/UDP		

Creating a Custom App-ID

1 To write a signature, go to the **Objects** tab > **Application** screen and click **New**. Please fill in the fields as appropriate, example follows:

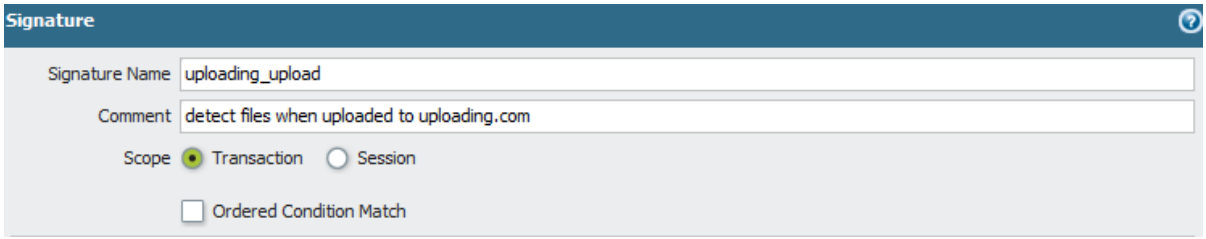


2 Since we are expanding on the predefined app-ID *uploading*, we select *uploading* as our application's Parent app in the properties section. By choosing uploading as the parent app, The App ID engine would classify the traffic to the child app in this case uploading-upload. Uploading is a browser-based file-sharing application and hence we entered the properties appropriately.

3 We have also set the characteristic of this application. In the above example, we have indicated that this application is able to transfer files, attempts to evade firewalls (evasive), is widely used, and also uses too much bandwidth. Since we didn't select 'continue scanning for other applications', the first matching signature is

reported and the firewall stops looking for additional matching application. The properties and characteristics in Application section are pivotal in creating Application Filters, which is beyond the scope of this Tech notes.

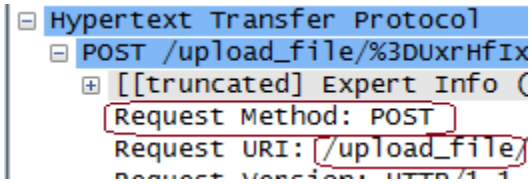
4 In the same window, select the **Signatures** tab, (We will come back to Advanced tab later) and click **New**. In the window that appears, enter a name and a brief description for the signature. Example follows:



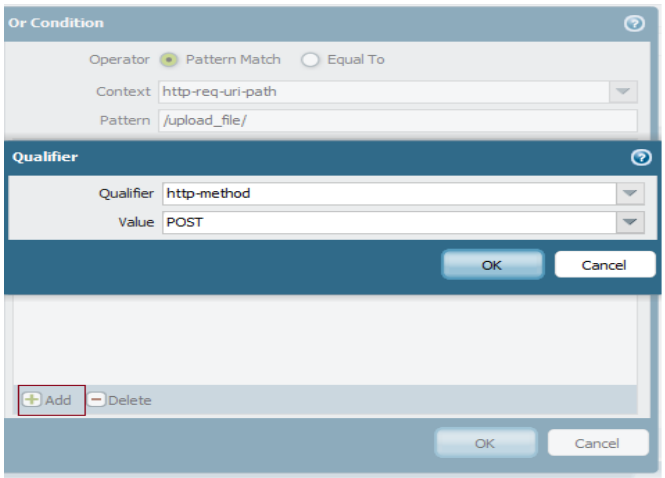
5 In the scope, you select if this signature applies to the current transaction or to the full TCP session. An HTTP request and a response constitute one transaction. A session could have one or more transactions. The key is all the 'conditions' in the signature should match any single transaction when 'Transaction' is selected. On the other hand, when Session is selected – conditions of the signature can match across transactions in the session.
In this particular example we are looking for /upload_file/ in http-method: POST.

6 The key word /upload_file/ in POST method is a part of the same transaction but not across different transactions, which is why we have selected **Transaction** as opposed to **Session.**

7 In the Wireshark output (shown below) we have noticed that the key word /upload_file/ is found in request_uri field of HTTP POST method.

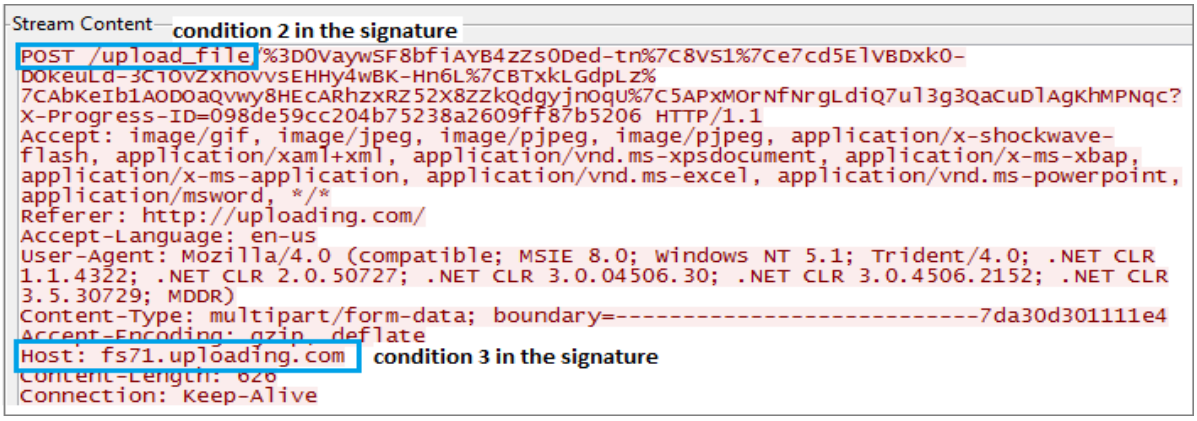


8 Click on **Add or Condition**, in the window that appears, select the context from the drop down menu. Select http-req-uri-path as the context with pattern as /upload_file/ and POST being our qualifying method. Click **Add** and select POST as the value qualifier http-method.

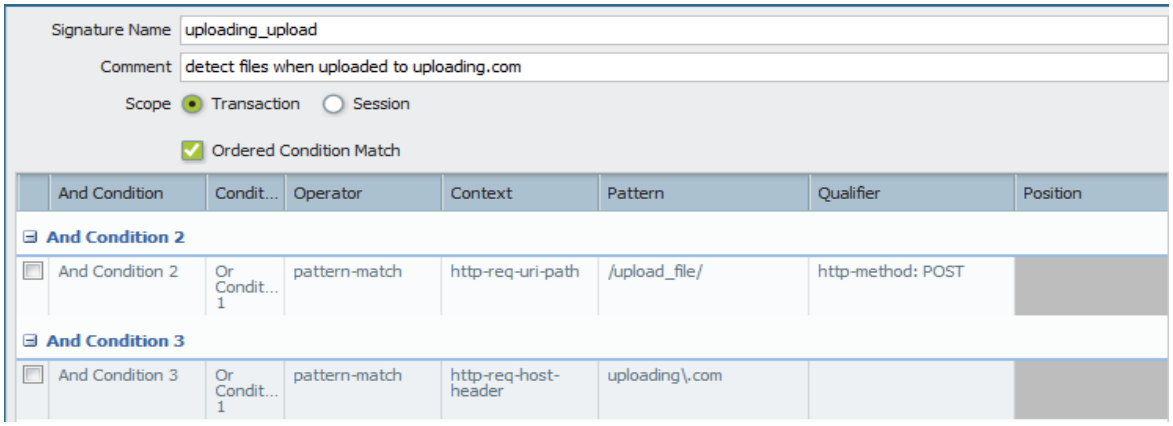


9 In simple terms, we are looking for /upload_file/ in POST method of http-req-uri-path context. We are interested in this pattern only if the firewall detects the application to be uploading.com. We add an extra condition which ensures the signature is relevant to uploading application.

10 In the HTTP POST transaction listed below we observe that the pattern /upload_file/ is seen first and then the host name fs71.uploading.com and the signatures we developed are ordered accordingly. We first place the signature to match /upload_file/ and then we place the signature to match the hostname. We also select the check box for 'ordered Condition Match'.

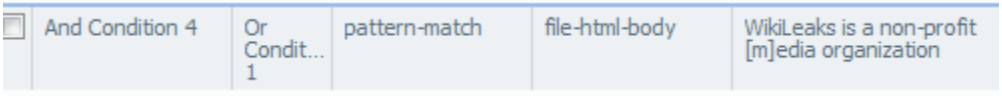


11 Essentially we wrote a signature to detect uploading at uploading.com. The signatures look something like the below.



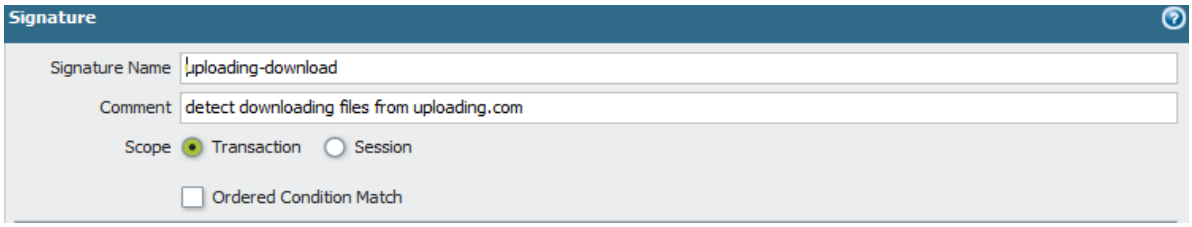
Things to remember:

- The pattern must be a minimum of 7 bytes.
- Any pattern exceeding 32 bytes need to be broken down by placing open and close square brackets on or before the 32 character. In the example below we are matching the content of wiki-leaks site.

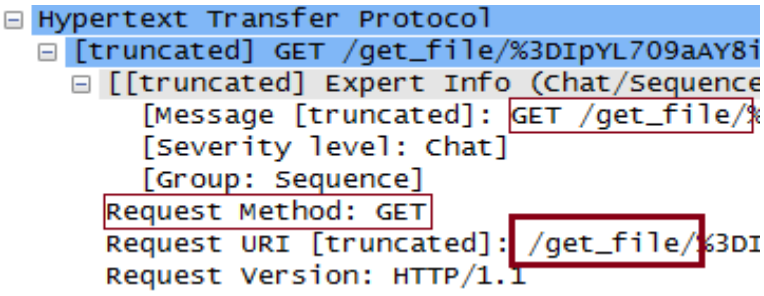


We will now write a signature to detect downloads from the same site.

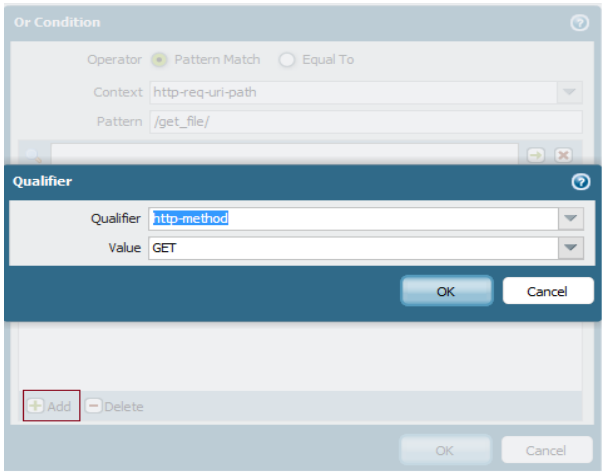
12 Follow steps 1 thru 4 and add an App-ID named uploading-download. Use uploading as its parent app and fill in the fields of the signature tab. Example follows:



In the Wireshark output (shown below) we have noticed that the key word /get_file/ is found in request_uri field of HTTP GET method.

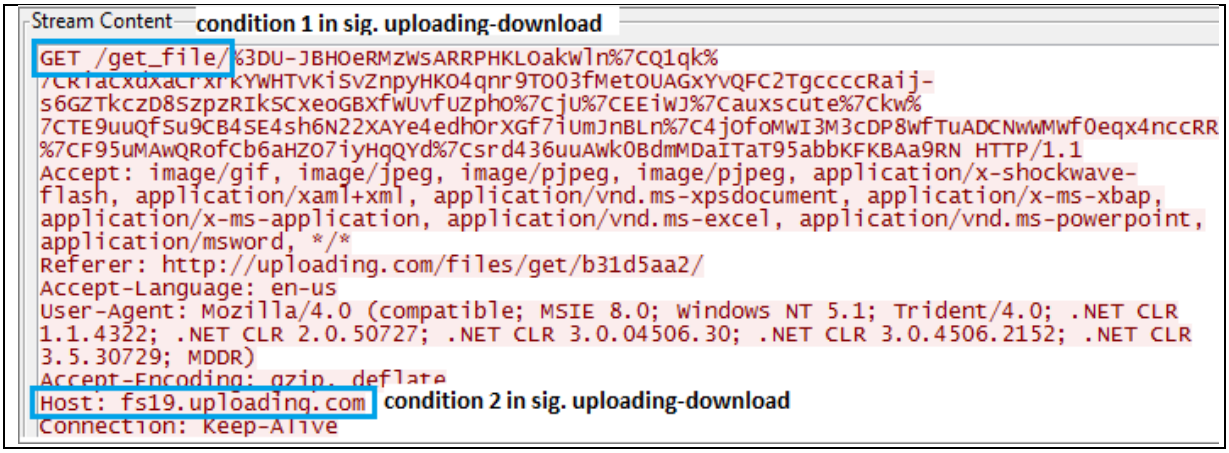


13 Click on **Add or Condition**, in the window that appears and select the context from the drop down menu. Select http-req-uri-path as the context with pattern as /get_file/ and GET being our qualifying method. Click Add and select GET as the value qualifier http-method.



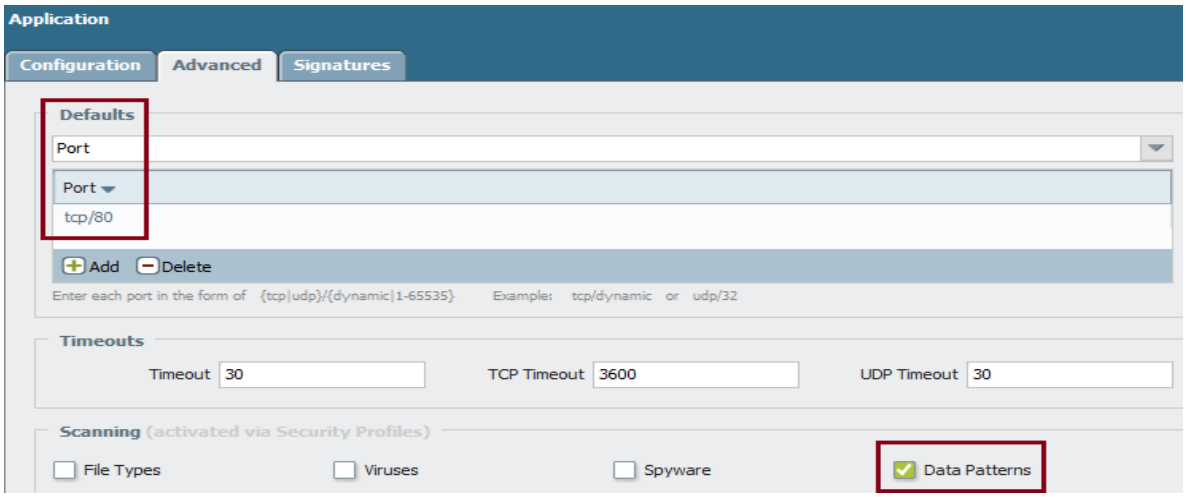
14 In simple terms, we are looking for /get_file/ in GET method of http-req-uri-path context. We are interested in this pattern only if the firewall detects the application to be uploading.com. We add an extra condition which ensures the signature is relevant to uploading application.

15 In the HTTP GET transaction listed below we observe that the pattern /get_file/ is seen first and then the host name fs19.uploading.com and the signatures we developed are ordered accordingly. So, we first place the signature to match /get_file/ and then we place the signature to match the hostname. We also select the check box for 'ordered Condition Match'.



16 The signatures for uploading-download will look something like the following:

17 In the same window select **Advanced** tab.



18 Uploading.com communicates on port 80, so we selected TCP/80 as the default port. If your App is running on TCP port 2000, then you would specify TCP/2000. These selections are useful when 'application-default' is selected as the service in the security policy, in which case the configured port is used as a service to limit the applications.

In the drop down box there is protocol and ICMP type as well. If you are going to inspect ICMP packets, choose type ICMP. If you plan to classify traffic based on IP protocol then choose the IP protocol option. When protocols or ICMP types are used in defaults, traffic that matches the protocol is classified as that App-ID and signature matching will not occur since the application is already classified at protocol level. You could imagine this as an application override concept.

19 Timeout – We have entered 30 i.e. after 30 idle seconds the flow will be terminated. This value is used for non TCP/UDP streams and also for TCP/UDP applications when TCP/UDP timeouts are not specified. For TCP/UDP applications, specify timeouts in TCP Timeout and UDP Timeout fields. Default timeouts will be applied when timeouts are not specified in the fields.

20 Scanning - If you plan to scan for specific file types, viruses, spyware, and data patterns in the data flows of the application for which you are developing this application signature, then select the appropriate options.
In this example, we have selected the **Data Patterns** check box. Scanning the flows for a specified pattern is done by creating a custom data filtering profile – which is beyond the scope of this tech note.

21 Commit your signatures.

False Positives

Signature writing is an iterative process. A good application signature is precise and covers all application scenarios. So it's imperative that you test the signature meticulously. Once you crafted the pattern, commit the configuration, run the traffic and see if all the sessions generated by the application are matched against the signatures you wrote. You check this by looking into the traffic logs of monitor tab. If you notice some traffic is not matching your signature that means your signature is not complete. In Wireshark follow the TCP streams of the sessions that are not matched to identify unique patterns.

Test the Signature

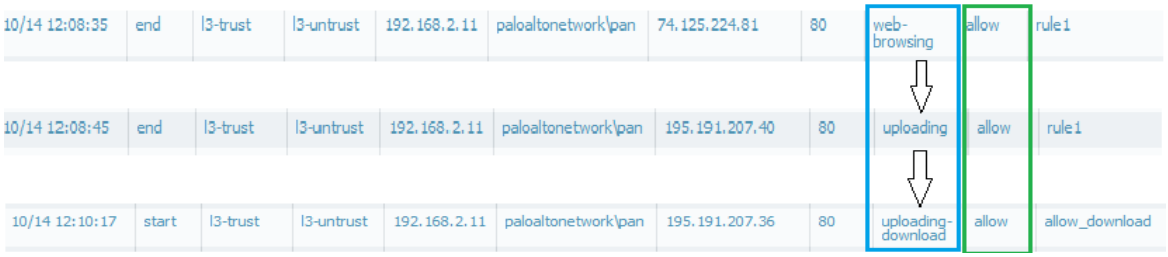
1. On the client machine, open a browser and connect to the uploading.com site. To go beyond more than just the home page— use the upload feature to upload a file. As you notice below, initially the firewall classified this as web-browsing, later it transitioned to uploading (predefined app). The firewall then detected uploading of files into this site and changed the session to "uploading-upload".

   The Application Uploading is allowed, however uploading-upload is matched against a deny policy ("deny_upload") and the device drops the packets.

10/14 11:32:05	start	l3-trust	l3-untrust	192.168.2.11	paloaltonetwork\pan	195.191.207.50	80	web-browsing	allow	rule1
10/14 11:32:05	start	l3-trust	l3-untrust	192.168.2.11	paloaltonetwork\pan	195.191.207.50	80	uploading	allow	rule1
10/14 11:32:08	deny	l3-trust	l3-untrust	192.168.2.11	paloaltonetwork\pan	195.191.207.50	80	uploading-upload	deny	deny_upload

2. Download a file from uploading.com. As you notice below, the firewall initially classified this as web-browsing, later it transitioned to uploading (predefined app). Then the firewall detected downloading of files from this site and changed the session to "uploading-download".

   The Application uploading-upload is denied however uploading-download is allowed. As you notice the app uploading-download is matched against a allow policy ("allow_download") and the user is able to download the file.



10/14 12:08:35	end	l3-trust	l3-untrust	192.168.2.11	paloaltonetwork\pan	74.125.224.81	80	web-browsing	allow	rule1
10/14 12:08:45	end	l3-trust	l3-untrust	192.168.2.11	paloaltonetwork\pan	195.191.207.40	80	uploading	allow	rule1
10/14 12:10:17	start	l3-trust	l3-untrust	192.168.2.11	paloaltonetwork\pan	195.191.207.36	80	uploading-download	allow	allow_download

As you notice the custom App IDs will help classify traffic and thereby you could enable or control the traffic.