a Hewlett Packard
Enterprise company

ClearPass Ingress Event Engine

Setup of the ClearPass Ingress Event Engine within ClearPass 6.6

Version	**Date**	**Modified By**	**Comments**
0.1	March 2016	Danny Jump	Early Draft Versions
1.0	April 2016	Danny Jump	Initial Published Version

TABLE OF CONTENTS

TABLE OF FIGURES

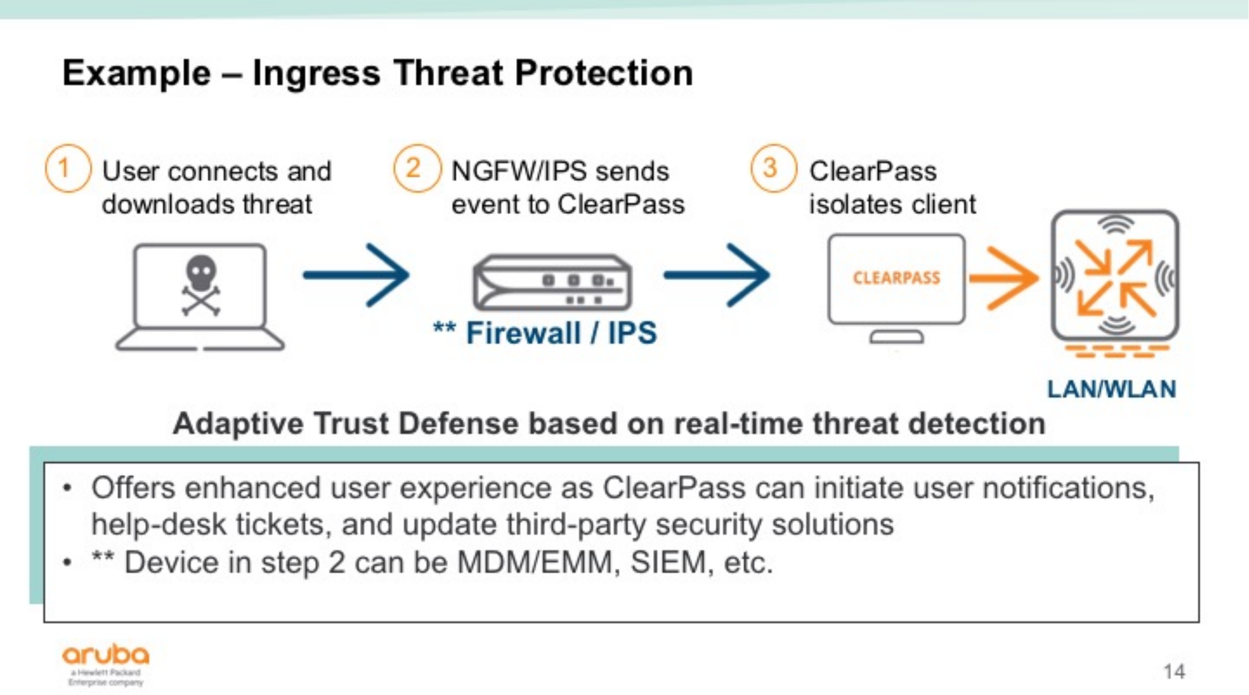
Introduction

This TechNote covers the setup/configuration and monitoring of the Ingress Event Engine [IEE]. This is a new feature added in ClearPass Policy Manager 6.6 released in April 2016.

The IEE delivers a new dimension to the capabilities on how ClearPass can interoperate with Devices and Users. Prior to this feature two events were able to trigger CPPM in taking actions, the first was a typical WEB/802.1x network authentication, the second was the ability for a 3rd party system to utilize our exposed XMLAPI or RESTful API's.

In adding IEE, we have provided a 3rd dimension where an inbound syslog can be the trigger for CPPM to take action on the authenticated networks devices & users.

Below a pictorial view of the end-to-end workflow utilizing CPPM IEE.



Example – Ingress Threat Protection

1 User connects and downloads threat

2 NGFW/IPS sends event to ClearPass

3 ClearPass isolates client

Adaptive Trust Defense based on real-time threat detection

- Offers enhanced user experience as ClearPass can initiate user notifications, help-desk tickets, and update third-party security solutions
- ** Device in step 2 can be MDM/EMM, SIEM, etc.

Figure 1 - IEE Overview

Configuring Ingress Event Processing

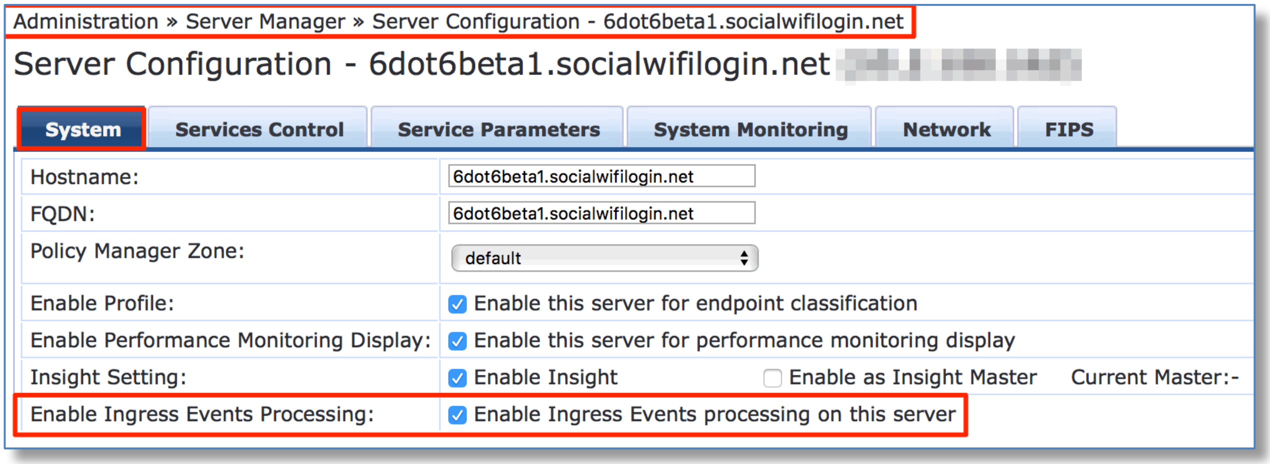
Starting within the ClearPass 6.6 release we added a new feature to the ClearPass Exchange Framework specific to our ability to consume **inbound** messages, parse them and trigger an action in the sense of an enforcement update for the user/endpoint. An example of this, could be we receive an update regarding the posture/health of the endpoint so we want to react by triggering an enforcement update for the device/user's role/vlan/dACL or/and we might want to trigger an update to a data-center or internet access firewall based upon this new posture/health context we have just received.

Providing this new feature allows for event driven real-time enforcement to happen within the enterprise network and enables a more effective framework of protection.

Multiple steps are required to configure the Ingress Event Notification Framework. One of the more complex steps relates to the definition of the Inbound parsing dictionary. We supply several by default, Palo Alto NGFW , Juniper SRX, CheckPoint FW-1 and Infoblox. we will add more as we receive request from the field and crowd sourcing will add others. One important point to mention is that the Ingress parsing Dictionaries can be added at anytime, they do not require a ClearPass s/w release or patch to add them. More on this later.

Checking Inbound Event Notification is enabled

Let's check the basics first..... we need to enable the Ingress Events processing engine. Enable this under **Administration->Server Manager->Server Configuration->[Your CPPM Node]->System** as below.


Administration » Server Manager » Server Configuration - 6dot6beta1.socialwifilogin.net
System
Services Control
Service Parameters
System Monitoring
Network
FIPS

Server Configuration - 6dot6beta1.socialwifilogin.net [redacted]

Hostname: 6dot6beta1.socialwifilogin.net

FQDN: 6dot6beta1.socialwifilogin.net

Policy Manager Zone: default

Enable Profile: ☑ Enable this server for endpoint classification

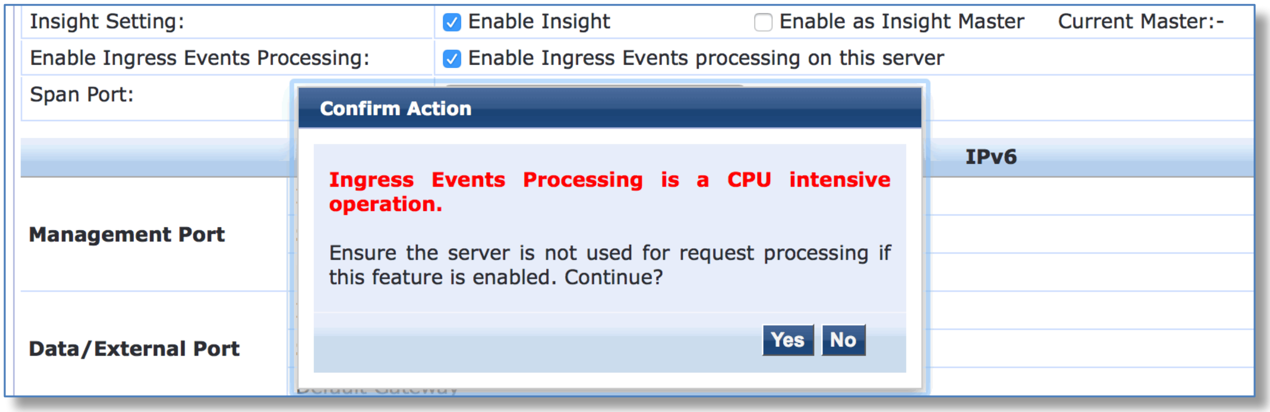
Enable Performance Monitoring Display: ☑ Enable this server for performance monitoring display

Insight Setting: ☑ Enable Insight □ Enable as Insight Master Current Master:-

Enable Ingress Events Processing: ☑ Enable Ingress Events processing on this server

Figure 2 - Enable Ingress Events Processing

Note when enabling this feature, a warning message is displayed as shown below which highlights and warns of the potential consequences. Be aware that is can generate a significant CPU load on the node. Carful consideration needs to be used and we specifically do not want CPPM to be receiving a constant stream of syslog messages that it has to then process. CPPM should only be receiving syslog messages by **exception** that it has to process and take action on, if you send a constant stream of syslog then the overhead will likely cause the node to become CPU bound and the potential failure/timeout of the primary function, the Authentication of Users/Computers.



Insight Setting: ☑ Enable Insight □ Enable as Insight Master Current Master:-

Enable Ingress Events Processing: ☑ Enable Ingress Events processing on this server

Span Port:

Confirm Action

Ingress Events Processing is a CPU intensive operation.

Ensure the server is not used for request processing if this feature is enabled. Continue?

Figure 3 – "Warning" message when enabling Ingress Event Processing

Guidelines for where-not and when-not to enable this feature?

Care must be taken when utilizing this feature, there are no absolute deployments when it can't be enabled but the following guidelines provide some guidance. Ultimately the overriding resource issue here is CPU cycles. It possible that a node can be running at 80% CPU load and its still be OK to enable this feature. Conversely you can have a node running at 50% CPU load and enabling this feature will cause issues. The underlying problem is NOT the current CPU load but the load that the inbound syslog messages will generate. This is why we make the point that syslog messages should only be sent to the CPPM node by **exception** and careful planning needs to be undertaken in configuring the syslog on the source. Nodes with multiple CPU's will fair better.

Hardware

Specific care needs to be taken when using CP-HW-500 appliances. Especially if you plan to enable this on the older CP-HW-500 hardware, again the gating factor is "What CPU load will this generate on the node?".

Virtual Machine

In respect of VM's its always been difficult to control the resource's a VM is allocated, especially the CPU's. With this being a very CPU specific loading workload we should pay extra attention to what CPU's are in use on the VM's and the current CPU usage.

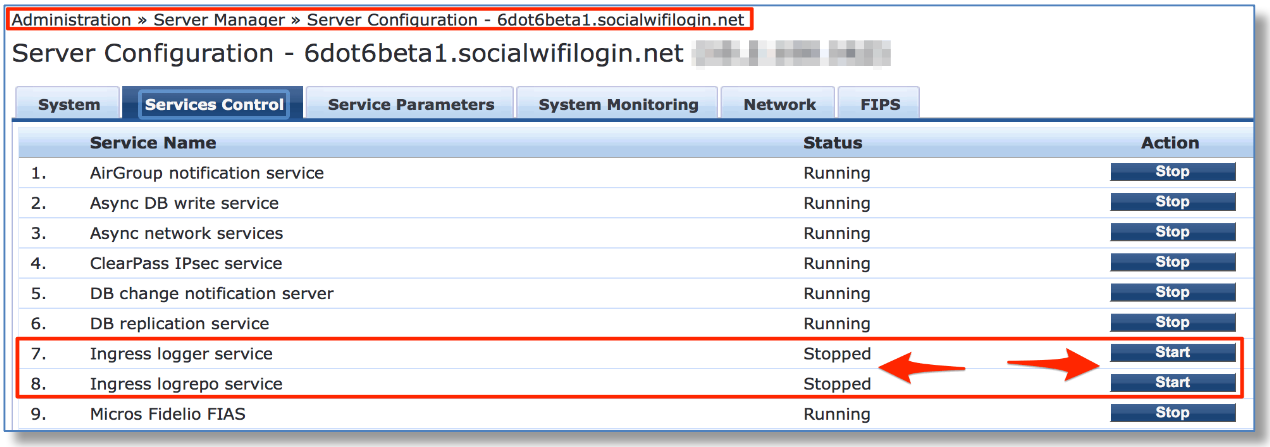
General

In a cluster of multiple nodes or when the syslog messages might be numerous then a design consideration will be to consider dedicating a node to processing the Ingress events. If this feature is expected to be a major part of a deployment, then a process to model the expected load should be undertaken to establish if a dedicated node is required. If the deployment is based around the use of h/w appliances and a dedicated h/w node is not available to help model the load, then consider deploying a temporary VM to establish a baseline and to better understand the demands of the Ingres workload.

We **do not** recommend deploying this feature on a node that is already running TCP Profiling. The combination of these two CPU intensive task should be avoided.

Checking if Ingress Daemons are running?

As a part of the Ingress framework, we have added two new daemons, check they are running, if they are **'Stopped'**, then start them using the controls as shown below.



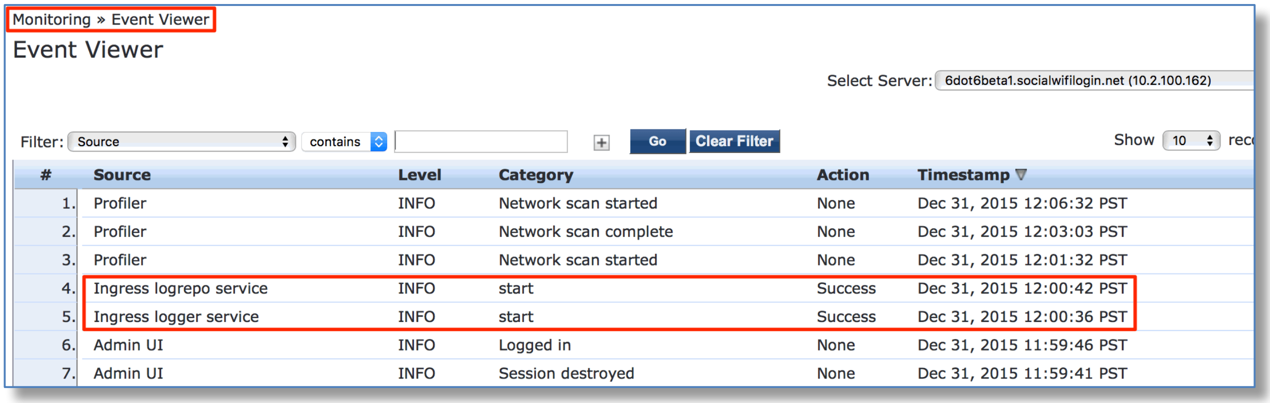
Administration » Server Manager » Server Configuration - 6dot6beta1.socialwifilogin.net

Server Configuration - 6dot6beta1.socialwifilogin.net [redacted]

	Service Name	Status	Action
1.	AirGroup notification service	Running	Stop
2.	Async DB write service	Running	Stop
3.	Async network services	Running	Stop
4.	ClearPass IPsec service	Running	Stop
5.	DB change notification server	Running	Stop
6.	DB replication service	Running	Stop
7.	Ingress logger service	Stopped	Start
8.	Ingress logrepo service	Stopped	Start
9.	Micros Fidelio FIAS	Running	Stop

Figure 4 - Check Ingress Event daemons are running?

If they do not start check under **Monitoring-> Event Viewer** for additional messages.



Monitoring » Event Viewer

Event Viewer

Select Server: 6dot6beta1.socialwifilogin.net (10.2.100.162)

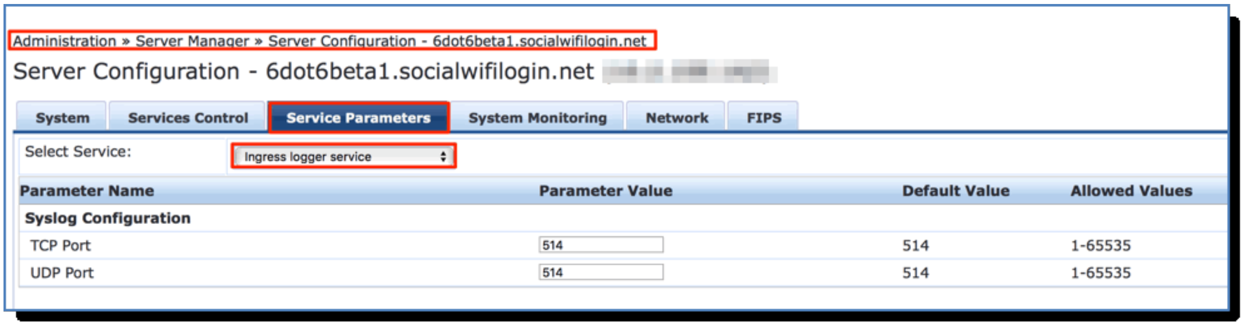
Filter: Source contains

#	Source	Level	Category	Action	Timestamp ▽
1.	Profiler	INFO	Network scan started	None	Dec 31, 2015 12:06:32 PST
2.	Profiler	INFO	Network scan complete	None	Dec 31, 2015 12:03:03 PST
3.	Profiler	INFO	Network scan started	None	Dec 31, 2015 12:01:32 PST
4.	Ingress logrepo service	INFO	start	Success	Dec 31, 2015 12:00:42 PST
5.	Ingress logger service	INFO	start	Success	Dec 31, 2015 12:00:36 PST
6.	Admin UI	INFO	Logged in	None	Dec 31, 2015 11:59:46 PST
7.	Admin UI	INFO	Session destroyed	None	Dec 31, 2015 11:59:41 PST

Figure 5 – Checking Ingress daemons in Event Viewer

Checking on the configured Inbound Ingress Listening Port?

As shown above there are a couple of new daemons, one of these, the logger service is responsible for listening on the TCP/UDP port defined below [default 514] and ingesting the syslog records on that port, configure the TCP/UDP port from the following location **Administration->Server Manager->Server Configuration->[Your CPPM Node]-> Service Parameters [select Ingress logger]… set the TCP/UDP port number required**


Administration » Server Manager » Server Configuration - 6dot6beta1.socialwifilogin.net
System
Services Control
Service Parameters
System Monitoring
Network
FIPS

Server Configuration - 6dot6beta1.socialwifilogin.net [redacted]

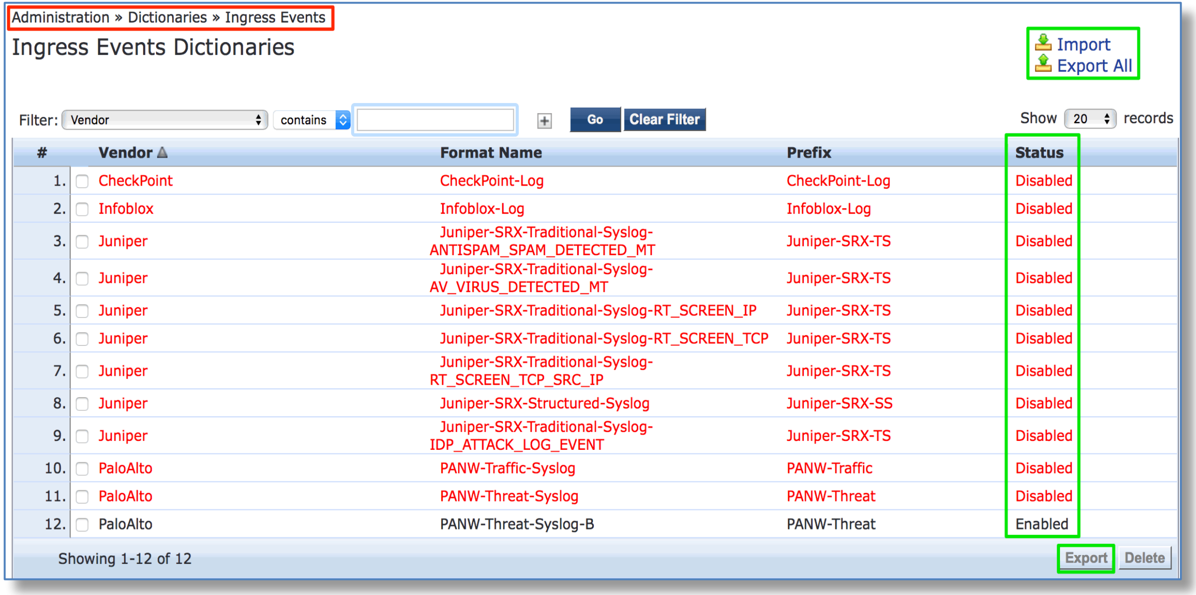
Select Service: Ingress logger service

Parameter Name	Parameter Value	Default Value	Allowed Values
Syslog Configuration			
TCP Port	514	514	1-65535
UDP Port	514	514	1-65535

Figure 6 - Configuring the Ingress listening port

Configuring the Ingress Events Dictionary

This is the actual file which takes the structured/unstructured syslog and turns it into fields/attributes that we can reference within a Namespace. This section of the configuration can be difficult, especially when adding a new Vendor. We supply multiple IEE dictionaries by default and will add more over time. In this initial release we have provided dictionaries for Juniper, Palo-Alto, CheckPoint and InfoBlox. For the SRX there are several dictionaries, as the SRX supports multiple syslog formats, structured & traditional. For Palo-Alto we added support for Traffic and Threat formatted syslogs. As you can see below the supplied dictionaries are disabled. **Only enable the required dictionaries.**



Ingress Events Dictionaries

Filter: Vendor contains

Show 20 records

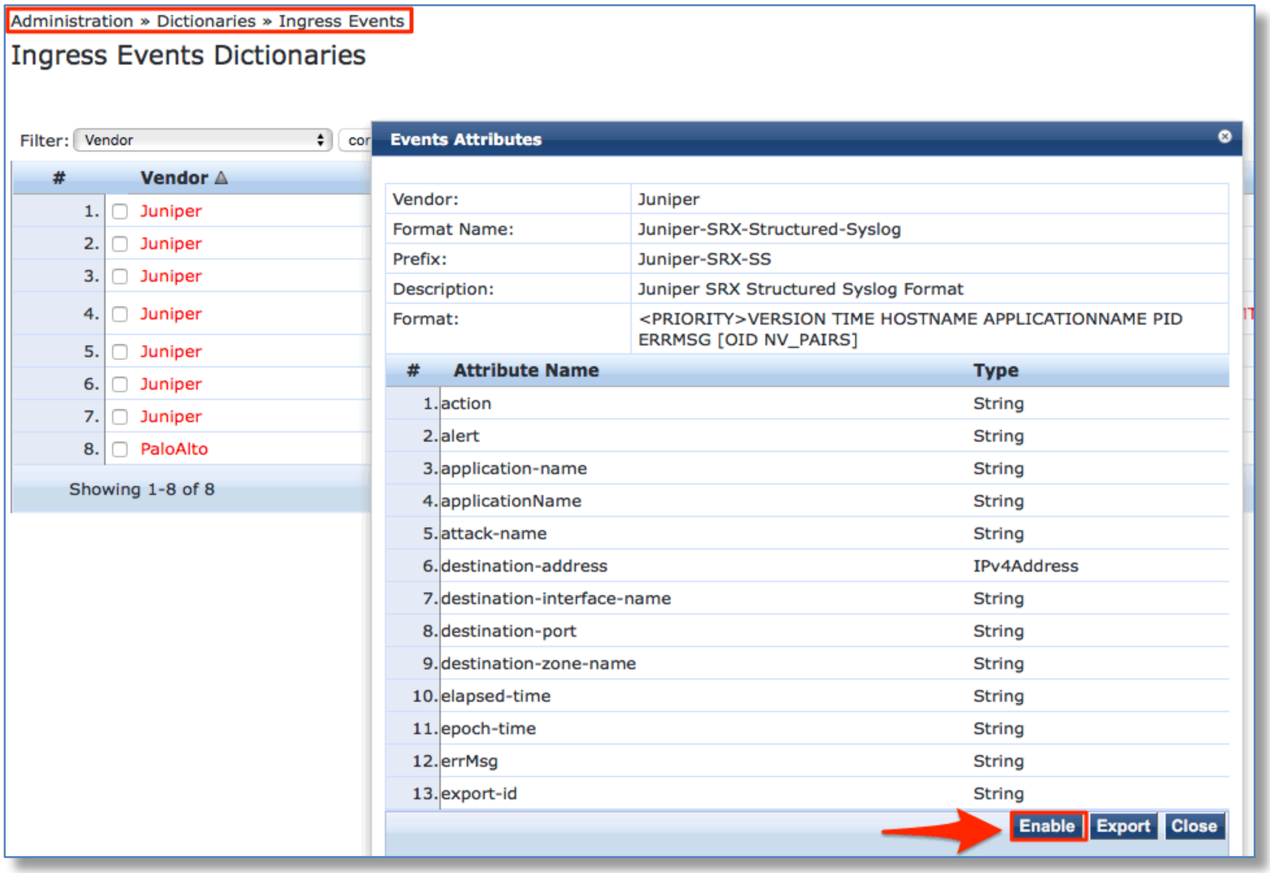
#		Vendor	Format Name	Prefix	Status
1.	□	CheckPoint	CheckPoint-Log	CheckPoint-Log	Disabled
2.	□	Infoblox	Infoblox-Log	Infoblox-Log	Disabled
3.	□	Juniper	Juniper-SRX-Traditional-Syslog-ANTISPAM_SPAM_DETECTED_MT	Juniper-SRX-TS	Disabled
4.	□	Juniper	Juniper-SRX-Traditional-Syslog-AV_VIRUS_DETECTED_MT	Juniper-SRX-TS	Disabled
5.	□	Juniper	Juniper-SRX-Traditional-Syslog-RT_SCREEN_IP	Juniper-SRX-TS	Disabled
6.	□	Juniper	Juniper-SRX-Traditional-Syslog-RT_SCREEN_TCP	Juniper-SRX-TS	Disabled
7.	□	Juniper	Juniper-SRX-Traditional-Syslog-RT_SCREEN_TCP_SRC_IP	Juniper-SRX-TS	Disabled
8.	□	Juniper	Juniper-SRX-Structured-Syslog	Juniper-SRX-SS	Disabled
9.	□	Juniper	Juniper-SRX-Traditional-Syslog-IDP_ATTACK_LOG_EVENT	Juniper-SRX-TS	Disabled
10.	□	PaloAlto	PANW-Traffic-Syslog	PANW-Traffic	Disabled
11.	□	PaloAlto	PANW-Threat-Syslog	PANW-Threat	Disabled
12.	□	PaloAlto	PANW-Threat-Syslog-B	PANW-Threat	Enabled

Showing 1-12 of 12

Figure 7 - Ingress Dictionaries

<u>Note:</u> From here you can also Export a dictionary. One of the Aruba supplied dictionaries could be used as a starting point/template if you want to build you own for a new vendor. Once 'refined' it can then be imported as highlighted in the Top-RHS of the screen.

Enable the required dictionary from **Administration-> Dictionaries -> Ingress Events.** Click on the Dictionary required and **enable** it as shown below for a Juniper Dictionary.



Ingress Events Dictionaries

#	Vendor
1.	☐ Juniper
2.	☐ Juniper
3.	☐ Juniper
4.	☐ Juniper
5.	☐ Juniper
6.	☐ Juniper
7.	☐ Juniper
8.	☐ PaloAlto

Showing 1-8 of 8

Vendor:	Juniper
Format Name:	Juniper-SRX-Structured-Syslog
Prefix:	Juniper-SRX-SS
Description:	Juniper SRX Structured Syslog Format
Format:	<PRIORITY>VERSION TIME HOSTNAME APPLICATIONNAME PID ERRMSG [OID NV_PAIRS]

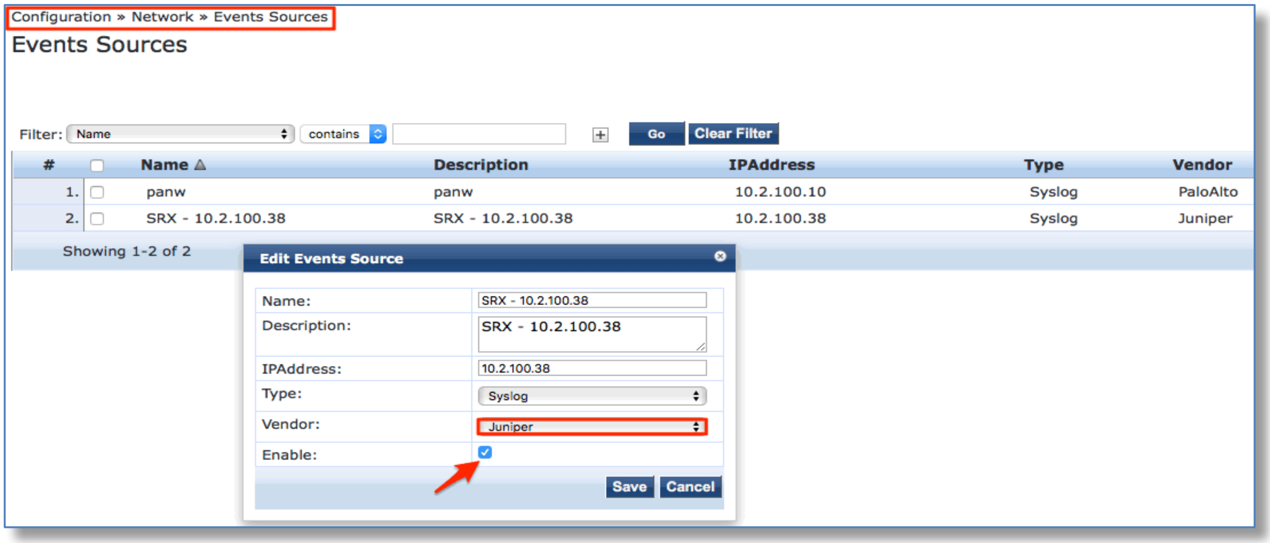
#	Attribute Name	Type
1.	action	String
2.	alert	String
3.	application-name	String
4.	applicationName	String
5.	attack-name	String
6.	destination-address	IPv4Address
7.	destination-interface-name	String
8.	destination-port	String
9.	destination-zone-name	String
10.	elapsed-time	String
11.	epoch-time	String
12.	errMsg	String
13.	export-id	String

Figure 8 - Enabling Ingress Dictionaries

Notice the Attribute Name's above, these are the fields that parse the incoming syslog data, later we will be able to reference these fields in our policy processing.

Add the Event Source

Next we need to define the event source, go to **Configuration-> Network -> Events Sources -> [Add your node].** The below example is adding a Juniper SRX firewall, a Palo-Alto firewall is already added.



Configuration » Network » Events Sources

Events Sources

Filter: Name contains

#	☐	Name	Description	IPAddress	Type	Vendor
1.	☐	panw	panw	10.2.100.10	Syslog	PaloAlto
2.	☐	SRX - 10.2.100.38	SRX - 10.2.100.38	10.2.100.38	Syslog	Juniper

Showing 1-2 of 2

Edit Events Source

Name:	SRX - 10.2.100.38
Description:	SRX - 10.2.100.38
IPAddress:	10.2.100.38
Type:	Syslog
Vendor:	Juniper
Enable:	☑

Figure 9 - Adding the Event Source [Syslog]

At this time the only message **Type** supported is Syslog, we are considering adding additional Ingres data-streams as demand and customer use-cases develop. Choose the correct vendor from the drop-down and ensure you **enable** the node as shown above.

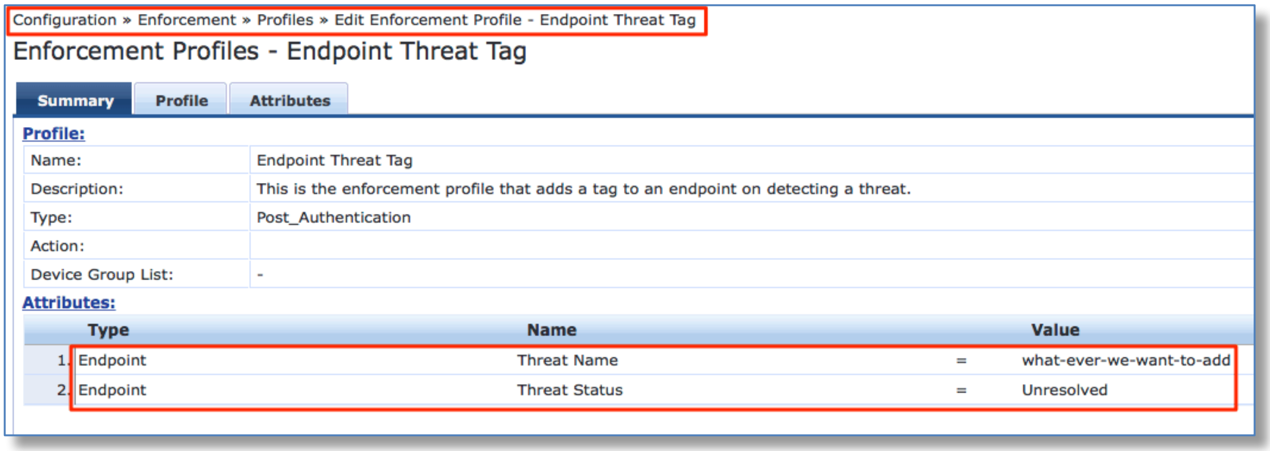
Building the Actions and Events

Next we need to add a number of profiles/services etc. to enable CPPM to be able to ingest the logs, parse the data and extract the content and context of the syslogs, and then take an action based upon the syslog data received. The following sections takes you through these steps. The method we use below might not be the exact method you utilize within your environment to take advantage of the Ingress Events engine but it demonstrates and hopefully educates you in the framework and methodology that it provides and explains the concept and required steps behind the Ingress framework.

Add the Enforcement Policy – Update Endpoint

We need to create a policy that we will use to update an endpoint attribute/tag when an “threat” has been detected. Create an Enforcement Policy under **Configuration -> Enforcement -> Profiles -> [of type ClearPass Entity Update]** You can create your own endpoint attribute or use on of the ones we supply. If you want to use your own you’ll need to create it first in the Attributes Dictionary. An example of one we created is shown below. Again, depending on the logic you will use, you might just update an endpoint tag, or you might as shown below also tag the endpoint as being in an unresolved state.

You'll need to also consider in your logic how your remove/reset these attributes once the threat has been resolved/remediated.



Configuration » Enforcement » Profiles » Edit Enforcement Profile - Endpoint Threat Tag

Enforcement Profiles - Endpoint Threat Tag

Profile:

Name:	Endpoint Threat Tag
Description:	This is the enforcement profile that adds a tag to an endpoint on detecting a threat.
Type:	Post_Authentication
Action:	
Device Group List:	-

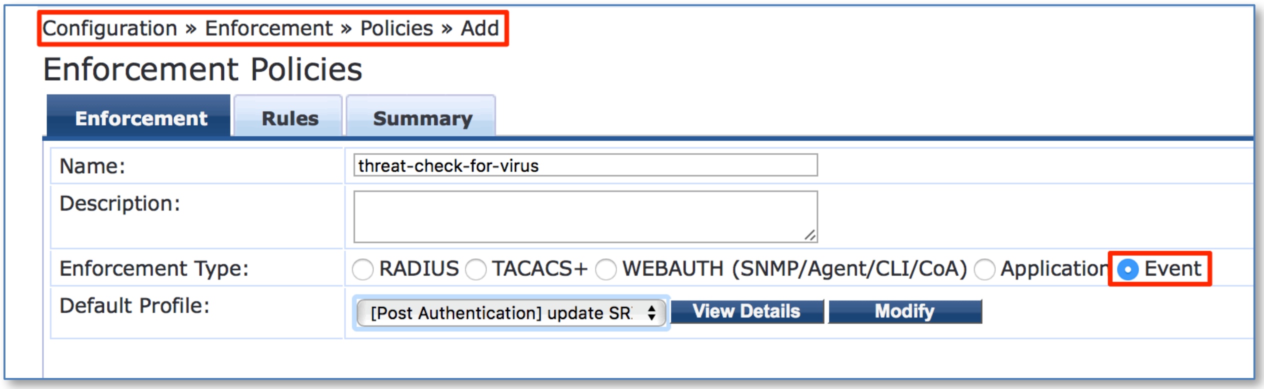
Attributes:

	Type	Name		Value
1.	Endpoint	Threat Name	=	what-ever-we-want-to-add
2.	Endpoint	Threat Status	=	Unresolved

Figure 10 – Adding an Enforcement Profile to update endpoint 'Threat' attributes

Add the Event Enforcement Policy

Next we need to create an **Event** Enforcement Policy. Create this under **Configuration -> Enforcement -> Policies [add]** be sure to select as highlighted below of the the RHS the Enforcement Type as **Event**. This is a new type of Enforcement Policy introduced in CPPM 6.6.



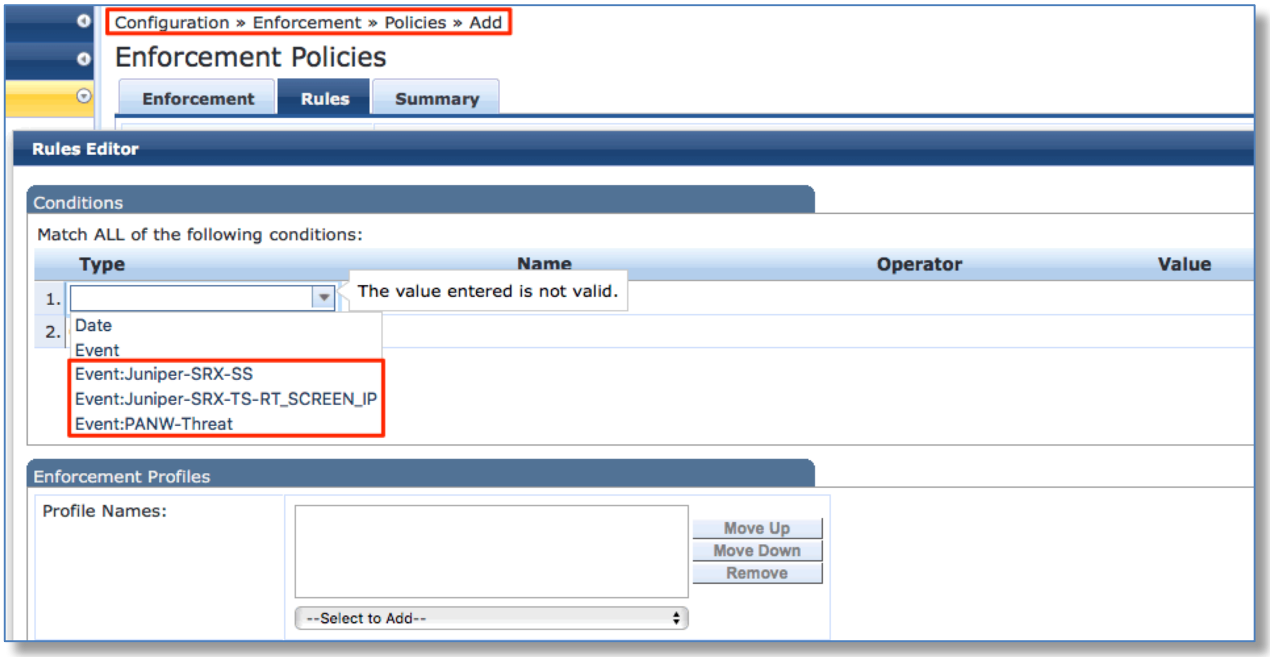
Configuration » Enforcement » Policies » Add

Enforcement Policies

Name:	threat-check-for-virus
Description:	
Enforcement Type:	○ RADIUS ○ TACACS+ ○ WEBAUTH (SNMP/Agent/CLI/CoA) ○ Application ● Event
Default Profile:	[Post Authentication] update SR

Figure 11 – Creating an Event Enforcement Policy

The Rules section here is the heart and powerhouse of the Event Engine. Based upon what Event Dictionaries you enable will determine what is available here in the rules definition. For this example, I enabled two SRX and a PANW dictionary, so I see these available in the rules engine when I go to add an event rule. If you have created and enable your own Event Dictionary, then this should appear here if.



Configuration » Enforcement » Policies » Add

Enforcement Policies

Rules Editor

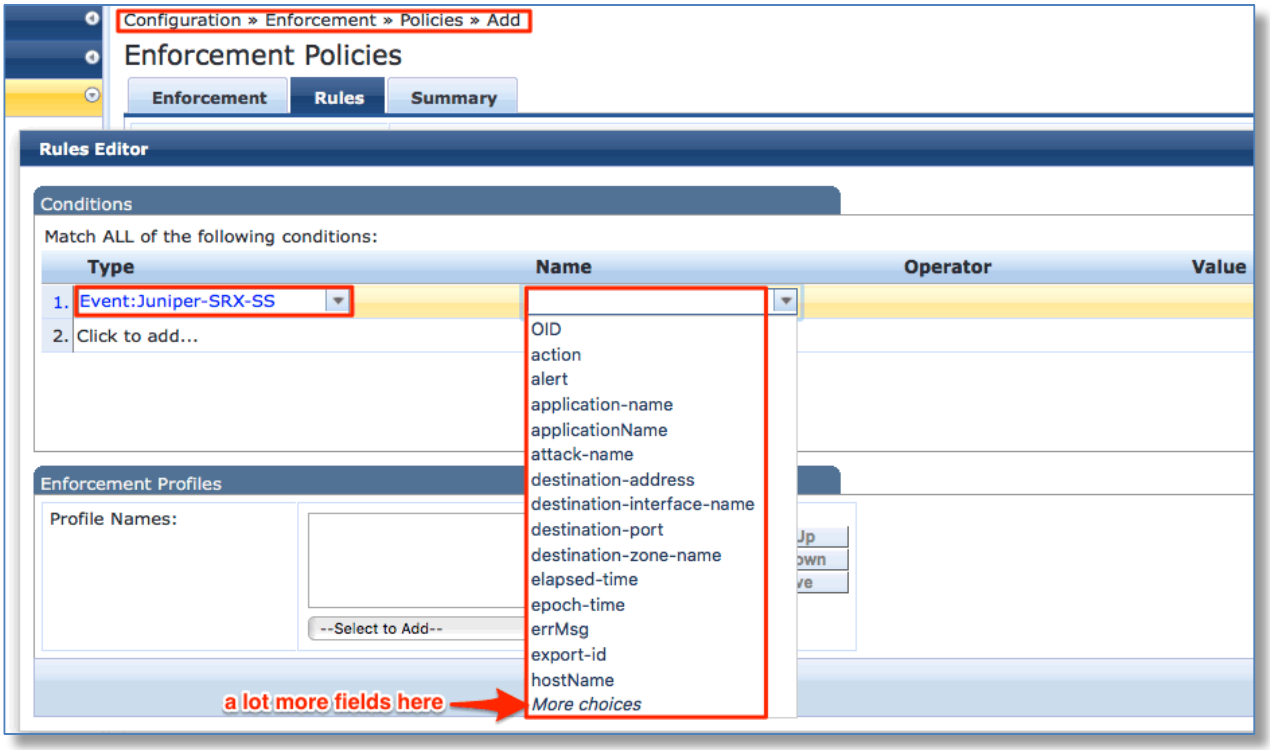
Conditions

Match ALL of the following conditions:

The value entered is not valid.

Figure 12 - Starting to build an event rule

If we choose the **'Juniper-SRX-SS'** and then expanding the namespace **'Name'**, we see all the fields that are configured in this Ingres Event Dictionary.



Configuration » Enforcement » Policies » Add

Enforcement Policies

Rules Editor

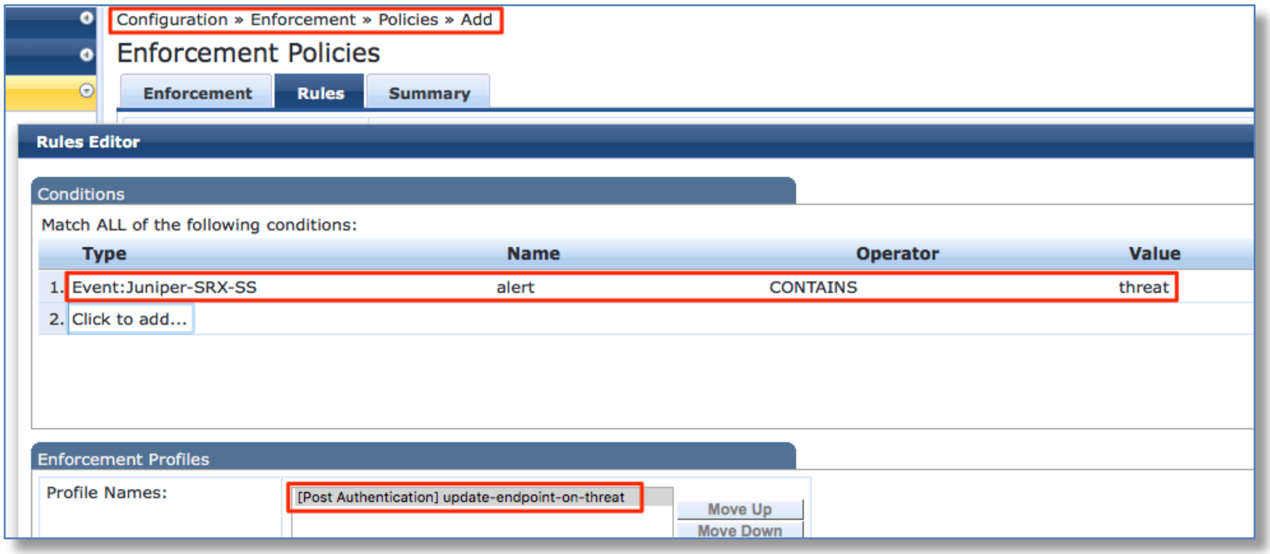
Conditions

Match ALL of the following conditions:

Type	Name	Operator	Value
1. Event:Juniper-SRX-SS			
2. Click to add...			

Figure 13 – Building the Event Rules from the Ingress Dictionary

In our case we will use just a simple check to demonstrate the configuration. We will set the policy to trigger an update to the endpoint if we see the word "threat" in the alert field.


Configuration » Enforcement » Policies » Add

Enforcement Policies

Match ALL of the following conditions:

	Type	Name	Operator	Value
1.	Event:Juniper-SRX-SS	alert	CONTAINS	threat
2.	Click to add...			

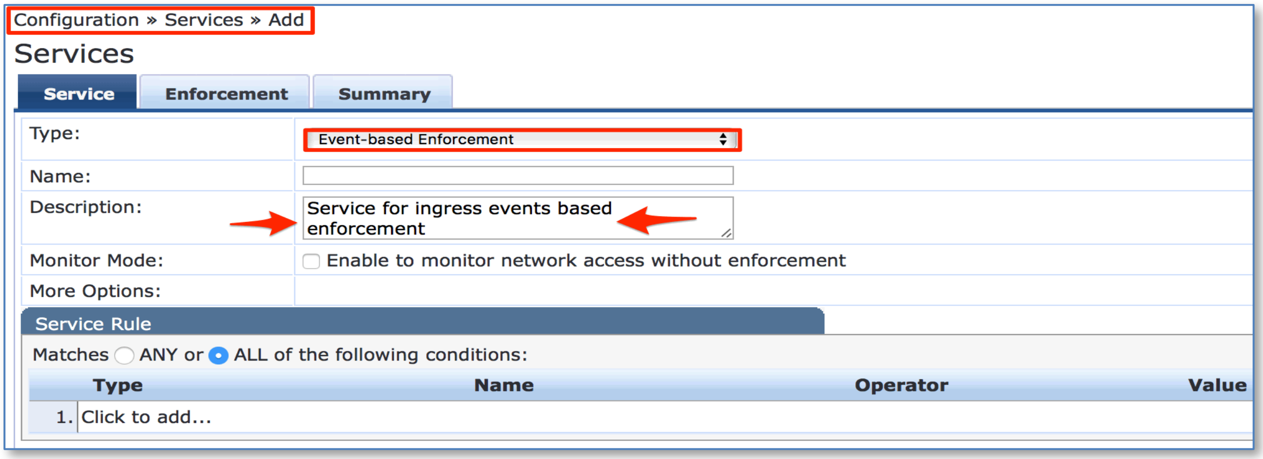
Profile Names: [Post Authentication] update-endpoint-on-threat

Figure 14 – Defining the event rule trigger and the Enforcement-Profile to call

As show above in the previous screen shot, there are numerous fields available to us in the namespace. Any of these fields in the namespace 'Name' [as in alert above] can be utilized to trigger an event with operators such as CONTAINS, EQUALS, EXISTS, BEGINS_WITH, NOT_EQUAL, NOT_CONTAINS, NOT_BELONGS_TO Etc.

Add the Event Service

Now that all of the parsing/enforcement is completed, we need to add the Event Service. Create this under **Configuration -> Services [add 'Event-based Enforcement']**, this is a **new** service type which was added in CPPM 6.6.


Configuration » Services » Add

Services

Type: Event-based Enforcement

Name:

Description: Service for ingress events based enforcement

Monitor Mode: □ Enable to monitor network access without enforcement

More Options:

Matches ○ ANY or ● ALL of the following conditions:

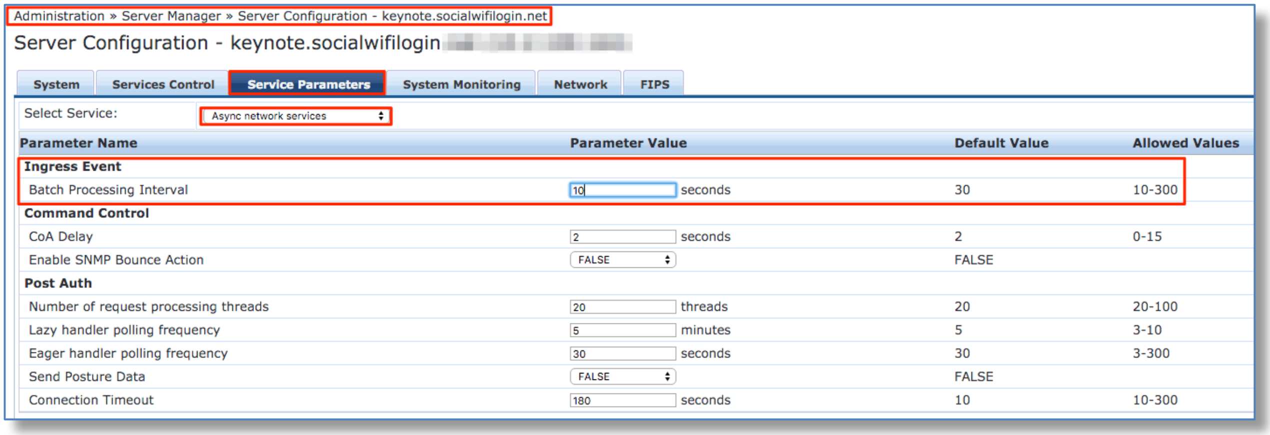
	Type	Name	Operator	Value
1.	Click to add...			

Figure 15 – Adding the Event Service

With all the above pieces in place, we are now setup to receive/parse syslog for a Juniper SRX firewall and [as in our use case] mark an endpoint in our DB that shows as being malicious/out-of-policy, more likely for this type of a malicious threat you'd want to trigger an enforcement-action to quarantine/isolate this device/user. A typical enforcement event here would be to issue a CoA and/or change the role of the user. The setup for any of the supported vendors or user-defined dictionary would be extremely similar, the only key difference is selecting the namespace field to match and the marching criteria to trigger an actionable event.

Tuning the Ingress Event Engine [IEE] Processing

The process of ingesting the syslog, parsing the data and writing this to the EventDb before an Event is triggered can take a period of time. This is very dependent on the physical resources available on the node, the number of Dictionaries enabled and the Batch Processing Interval. The Batch Processing Interval determines the frequency the Event Engine checks the EventDb for new entries. On lower powered h/w and older h/w the default of 30 seconds should suffice but it might need to be increased. For the CP-25K h/w or a node dedicated to IEE processing this parameter might need to be lowered to make the process more "real-time".



Administration » Server Manager » Server Configuration - keynote.socialwifilogin.net

Server Configuration - keynote.socialwifilogin[redacted]

Select Service: Async network services

Parameter Name	Parameter Value	Default Value	Allowed Values
Ingress Event			
Batch Processing Interval	10 seconds	30	10-300
Command Control			
CoA Delay	2 seconds	2	0-15
Enable SNMP Bounce Action	FALSE	FALSE	
Post Auth			
Number of request processing threads	20 threads	20	20-100
Lazy handler polling frequency	5 minutes	5	3-10
Eager handler polling frequency	30 seconds	30	3-300
Send Posture Data	FALSE	FALSE	
Connection Timeout	180 seconds	10	10-300

Figure 16 - Ingress Event Engine Processing Frequency

Example of Syslog configuration on Juniper SRX

The extent of syslog configuration on a Juniper SRX is extremely extensive. Logs can be stored locally or sent to an external syslog server. You can log events from the 'system' and/or you can log events from 'traffic'. We are going to be mainly interested in Data Plane traffic logs, these are generated by process's that control data such as the firewall and IPS process.

Note: For the process of integration with CPPM, we will have to use '**stream'** [default is '**event**'] mode.

To get started let's take a look at a section of JUNOS configuration specific to system syslog, the below represent a basic default setup for the system syslog, as we said were not particularly interested in the system level syslogs, and this is shown to show the basic setup of the system syslog, the below section is NOT configured to send system syslogs to an external syslog server but log internally. It likely for the process of integration with CPPM we will NOT be interested in the log messages coming out of the system syslog.

```
root@SRX650-TME> show configuration system syslog
archive size 100k files 3;
user * {
    any emergency;
}
file messages {
    any critical;
    authorization info;
}
file interactive-commands {
    interactive-commands error;
}
```

Figure 17 – Example of JUNOS system syslog configuration

As mentioned above our primary interest will relate to the logs coming out of the traffic on the data-plane. To configure the SRX to forward syslog messages to a CPPM node we need to configure a number of setting. The below example configuration contains the core requirements to send to two syslog servers. I added two destinations to demonstrate the ability of the SRX to send syslog to multiple syslog targets and differentiate in the type of logs that are sent, by syslog type, formatting, category etc.

In the below we have some configuration which is system/node specific such as **stream mode, source-address** and we have two CPPM nodes defined [**to_cppm** & **to_beta1**] , which have settings that can override those from the system level configuration such as the syslog formatting.

Note on **stream-mode**, this is normally used on high end SRX devices but can be configured on any model, its configured under the security stanza. In **steam-mode**, logs are sent to the remote syslog server [in this case CPPM] straight from the data plane and do not affect the SRX device performance. Because of that they can **not** be saved locally (no control plane processing). However, as we've mentioned previously, care must be taken not so overflow the CPPM node with syslog messages.

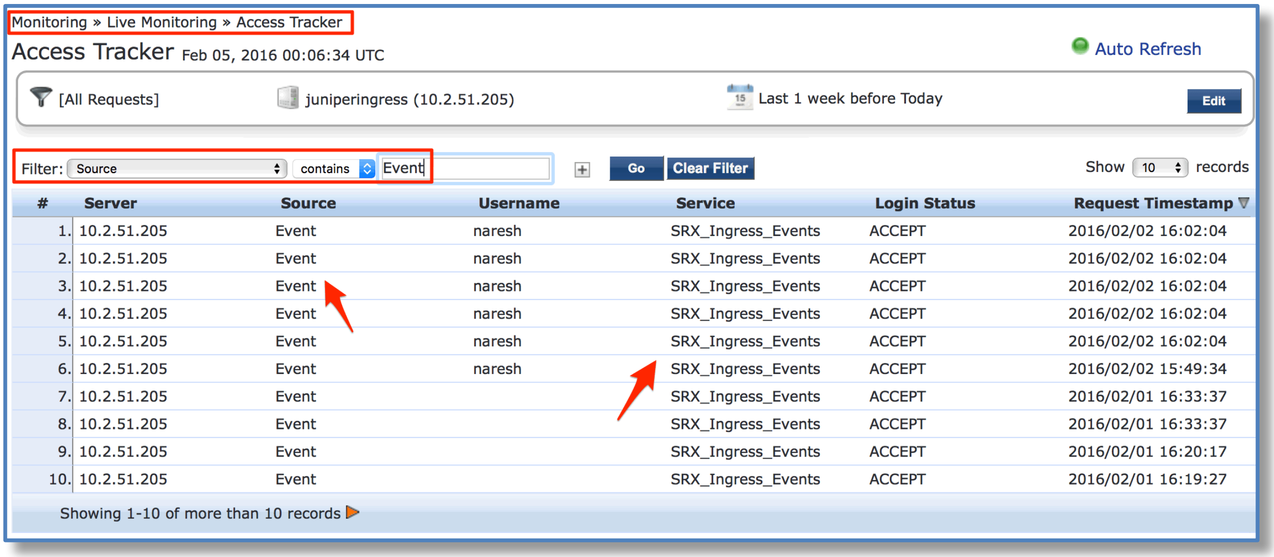
```
root@SRX650-TME> show configuration security log
mode stream;
format syslog;
source-address 10.2.51.132;
stream to_cppm {
    severity debug;
    format sd-syslog;
    filter threat-attack;
    host {
        10.2.51.210;
        port 514;
    }
}
stream to_beta1 {
    severity critical;
    format syslog;
    category idp;
    host {
        10.2.100.162;
        port 514;
    }
}
```

Figure 18 – An example of the traffic log configuration

As a follow up to the above basic configuration we can utilize the ability of the JUNOS configuration to limit the syslog received by CPPM.

CPPM Access Tracker Logs Examples

Below is an example of the logs that get posted in the CPPM Access Tracker when an ‘Event’ occurs. The below is filtering just on ‘Events’ in Access Tracker.



Monitoring » Live Monitoring » Access Tracker

Access Tracker Feb 05, 2016 00:06:34 UTC

[All Requests] juniperingress (10.2.51.205) Last 1 week before Today

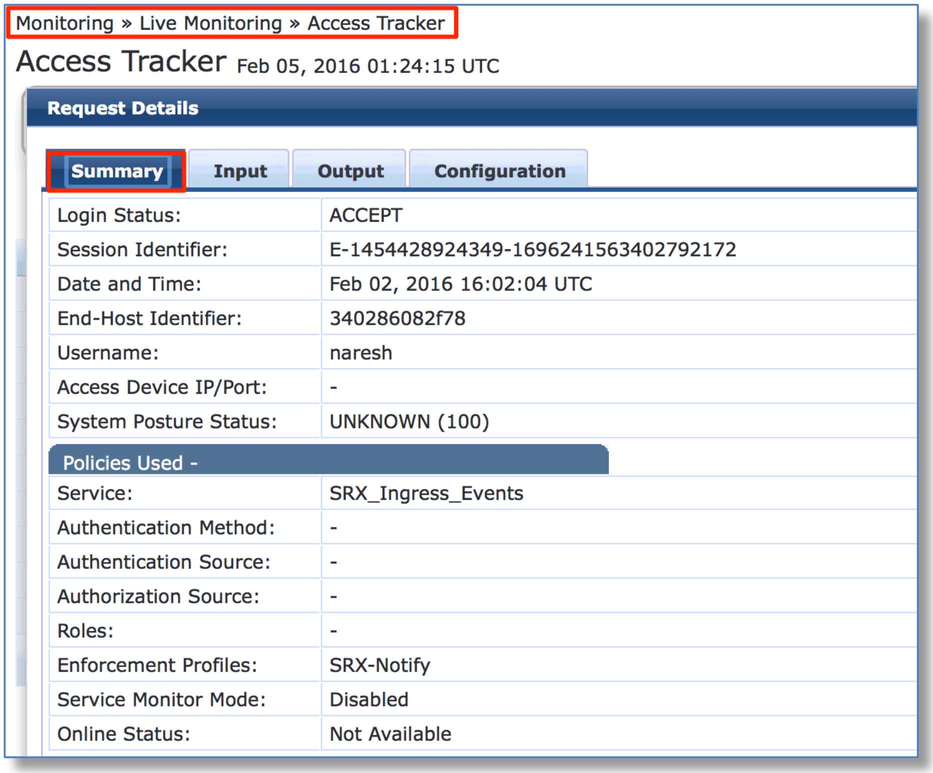
Filter: Source contains Event

Show 10 records

#	Server	Source	Username	Service	Login Status	Request Timestamp ▽
1.	10.2.51.205	Event	naresh	SRX_Ingress_Events	ACCEPT	2016/02/02 16:02:04
2.	10.2.51.205	Event	naresh	SRX_Ingress_Events	ACCEPT	2016/02/02 16:02:04
3.	10.2.51.205	Event	naresh	SRX_Ingress_Events	ACCEPT	2016/02/02 16:02:04
4.	10.2.51.205	Event	naresh	SRX_Ingress_Events	ACCEPT	2016/02/02 16:02:04
5.	10.2.51.205	Event	naresh	SRX_Ingress_Events	ACCEPT	2016/02/02 16:02:04
6.	10.2.51.205	Event	naresh	SRX_Ingress_Events	ACCEPT	2016/02/02 15:49:34
7.	10.2.51.205	Event		SRX_Ingress_Events	ACCEPT	2016/02/01 16:33:37
8.	10.2.51.205	Event		SRX_Ingress_Events	ACCEPT	2016/02/01 16:33:37
9.	10.2.51.205	Event		SRX_Ingress_Events	ACCEPT	2016/02/01 16:20:17
10.	10.2.51.205	Event		SRX_Ingress_Events	ACCEPT	2016/02/01 16:19:27

Showing 1-10 of more than 10 records ▶

Figure 19 - Showing 'events' in Access Tracker



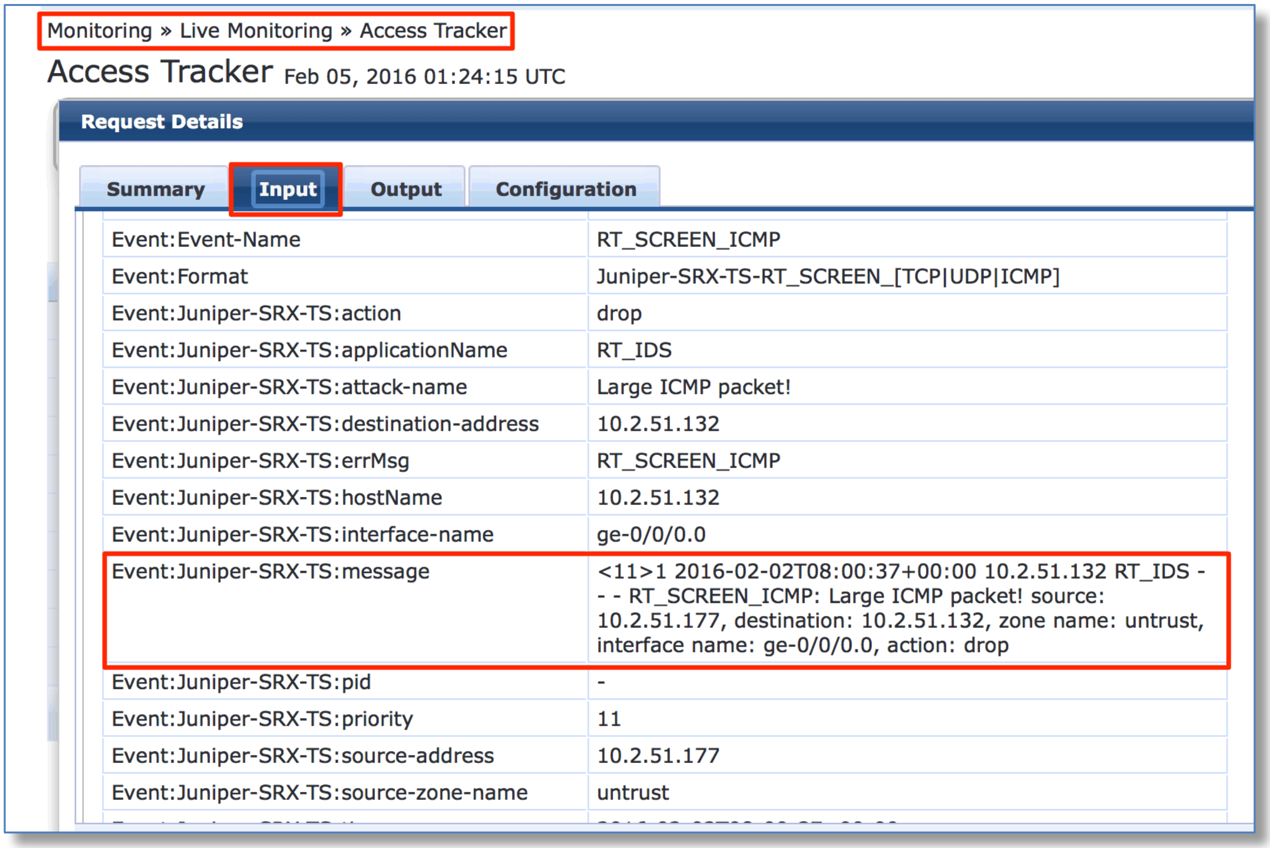
Monitoring » Live Monitoring » Access Tracker

Access Tracker Feb 05, 2016 01:24:15 UTC

Request Details

Login Status:	ACCEPT
Session Identifier:	E-1454428924349-1696241563402792172
Date and Time:	Feb 02, 2016 16:02:04 UTC
End-Host Identifier:	340286082f78
Username:	naresh
Access Device IP/Port:	-
System Posture Status:	UNKNOWN (100)
Policies Used -	
Service:	SRX_Ingress_Events
Authentication Method:	-
Authentication Source:	-
Authorization Source:	-
Roles:	-
Enforcement Profiles:	SRX-Notify
Service Monitor Mode:	Disabled
Online Status:	Not Available

Figure 20 - Low level detail of an 'Event'



Monitoring » Live Monitoring » Access Tracker

Access Tracker Feb 05, 2016 01:24:15 UTC

Request Details

Event:Event-Name	RT_SCREEN_ICMP
Event:Format	Juniper-SRX-TS-RT_SCREEN_[TCP|UDP|ICMP]
Event:Juniper-SRX-TS:action	drop
Event:Juniper-SRX-TS:applicationName	RT_IDS
Event:Juniper-SRX-TS:attack-name	Large ICMP packet!
Event:Juniper-SRX-TS:destination-address	10.2.51.132
Event:Juniper-SRX-TS:errMsg	RT_SCREEN_ICMP
Event:Juniper-SRX-TS:hostName	10.2.51.132
Event:Juniper-SRX-TS:interface-name	ge-0/0/0.0
Event:Juniper-SRX-TS:message	<11>1 2016-02-02T08:00:37+00:00 10.2.51.132 RT_IDS - - - RT_SCREEN_ICMP: Large ICMP packet! source: 10.2.51.177, destination: 10.2.51.132, zone name: untrust, interface name: ge-0/0/0.0, action: drop
Event:Juniper-SRX-TS:pid	-
Event:Juniper-SRX-TS:priority	11
Event:Juniper-SRX-TS:source-address	10.2.51.177
Event:Juniper-SRX-TS:source-zone-name	untrust

Figure 21 - Event Detail - showing syslog data

Appendix A – Notes on Dictionary creation

The supplied syslog dictionaries define how ClearPass interprets the incoming syslog messages regardless of their formatting, be that RAW-BSD, CEF or LEEF. These dictionaries define how when syslog messages are received how ClearPass parses out the Key-Value-Pairs and then populates the EventDB with the received data in a normalized format. This data is then processed by the Event Processing Engine and if rules are matched, triggers actionable events.

The dictionaries are written in XML and are based upon a formatting engine called logstash. An example of a XML Dictionary is below.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TipsContents xmlns="http://www.avendasys.com/tipsapiDefs/1.0">
 <TipsHeader exportTime="Mon Mar 21 20:44:07 PDT 2016" version="6.6"/>
 <IngressEvents>
  <IngressEvent>
   <Vendor>Juniper</Vendor>
   <Description>Juniper SRX Traditional SysLog Format for RT_SCREEN_IP messages</Description>
   <FormatName>Juniper-SRX-Traditional-Syslog-RT_SCREEN_IP</FormatName>
   <Format>&lt;priority code&gt;version timestamp hostname process - - - tag: attack-name source: source-
address, destination: destination-address, protocol-id: protocol-id, zone name: source-zone-name, interface name:
interface-name, action: action</Format>
   <Prefix>Juniper-SRX-TS</Prefix>
   <Enabled>false</Enabled>
   <Sample>&lt;11&gt;1 2015-11-02T06:02:20.970Z xihai RT_IDS - - - RT_SCREEN_IP: Fragmented traffic! source:
21.0.0.2, destination: 31.0.0.99, protocol-id: 17, zone name: trust, interface name: reth0.0, action: drop</Sample>
   <Filter>filter {
          grok {
        match =&gt; { "message" =&gt; "&lt;%{POSINT:priority}&gt;%{POSINT:version}
(%{TIMESTAMP_ISO8601:time} )?(%{SYSLOGHOST:hostName} )?%{DATA:applicationName} %{DATA:pid}
%{DATA:unknown-field-1} %{DATA:unknown-field-2} %{DATA:errMsg}: %{DATA:attack-name} source:
%{DATA:source-address}, destination: %{DATA:destination-address}, protocol-id: %{DATA:protocol-id}, zone name:
%{DATA:source-zone-name}, interface name: %{DATA:interface-name}, action: %{DATA:action}" }
                add_tag =&gt; [ "TS" ]
            }
            if("TS" in [tags]){
            mutate {
                remove_field =&gt; ['@version','host','path','@message']
                add_field =&gt; [ 'Event:Source-IP-Address', '%{source-address}' ]
                add_field =&gt; [ 'Event:Destination-IP-Address', '%{destination-address}' ]
                add_field =&gt; [ 'Event:Threat-Name', '%{attack-name}' ]
                add_field =&gt; [ 'Event:Event-Name', '%{errMsg}' ]
                add_field =&gt; [ 'Event:Pattern-Name', 'Juniper-SRX-TS' ]
                add_field =&gt; [ 'Event:Format', 'Juniper-SRX-TS-RT_SCREEN_IP' ]
                add_field =&gt; [ 'Event:Timestamp', '%{time}' ]
            }
```

```
                ruby {
                        code =&gt; "
                                data = event.clone.to_hash;
                                data.each do |k,v|
                                        if (k != '@timestamp' and !k.start_with?('Event:'))
                                                newFieldName = 'Event:Juniper-SRX-TS:'+ k
                                                event[newFieldName] = v
                                                event.remove(k)
                                        end
                                end
          tstamp = Time.now.to_i
          tstamp_str = Time.at(tstamp).strftime('%Y-%m-%d %H:%M:%S')
          event['Event:Timestamp'] = tstamp_str
                        "
                }}
        }</Filter>
   <FieldMapping>
    <Field AllowedValues="" DataType="Time" Name="time"/>
    <Field AllowedValues="" DataType="String" Name="hostName"/>
    <Field AllowedValues="" DataType="String" Name="applicationName"/>
    <Field AllowedValues="" DataType="String" Name="errMsg"/>
    <Field AllowedValues="" DataType="IPv4Address" Name="source-address"/>
    <Field AllowedValues="" DataType="IPv4Address" Name="destination-address"/>
    <Field AllowedValues="" DataType="String" Name="pid"/>
    <Field AllowedValues="" DataType="String" Name="unknown-field-1"/>
    <Field AllowedValues="" DataType="String" Name="unknown-field-2"/>
    <Field AllowedValues="" DataType="String" Name="source-zone-name"/>
    <Field AllowedValues="" DataType="String" Name="interface-name"/>
    <Field AllowedValues="" DataType="String" Name="protocol-id"/>
    <Field AllowedValues="" DataType="String" Name="action"/>
    <Field AllowedValues="" DataType="String" Name="attack-name"/>
   </FieldMapping>
   <GenericFieldMapping>
    <Field GenericName="Threat-Name" Name="attack-name"/>
    <Field GenericName="Destination-IP-Address" Name="destination-address"/>
    <Field GenericName="Event-Name" Name="errMsg"/>
    <Field GenericName="Source-IP-Address" Name="source-address"/>
    <Field GenericName="Timestamp" Name="time"/>
   </GenericFieldMapping>
  </IngressEvent>
 </IngressEvents>
</TipsContents>
```

Figure 22 - XML of an Event Dictionary