



RSA CHARGE 2017

Dallas, TX | October 17-19, 2017

Event Stream Analysis (ESA) Rules
Angela Stranahan – Software Principle Engineer, RSA

AGENDA

- Introduction to ESA
- Configuring ESA
- Basics of Event Processing Language (EPL)
 - Event Streams and Filtering
 - Handling Multi-Valued Meta
 - Using the EsperTech Online tool
- Troubleshooting ESA Rules
 - Deploying Rules with Custom Meta
 - Event Timing
- ESA Best Practices
- Advanced Use Cases
 - Adding Enrichments
 - Working with Contexts
 - Using Named Windows
- Resources

INTRODUCTION TO ESA RULES

WHAT IS EVENT STREAM ANALYSIS (ESA)

Provides near real-time correlation and complex event processing of log and packet meta

- Complex event processing is a concept that provides processing of multiple events with the goal of identifying the meaningful events within the event stream

A Complex Event is a series of events that when occurring in a particular order create a particular result

- For example, log in to a computer, ssh to a unix server, scp badbios.sh to a file server, execute badbios.sh, log out

Profiling the Asset Environment

Background Information

- ▶ How the asset is used
- ▶ Who can do what
- ▶ Type of environment / location (internet, intranet, etc.)

Information Assets

- ▶ Communication channel / regulatory landscape
- ▶ Business functions / usage scenarios / risk profile
- ▶ Operational and support procedures / SLA

Asset Ownership

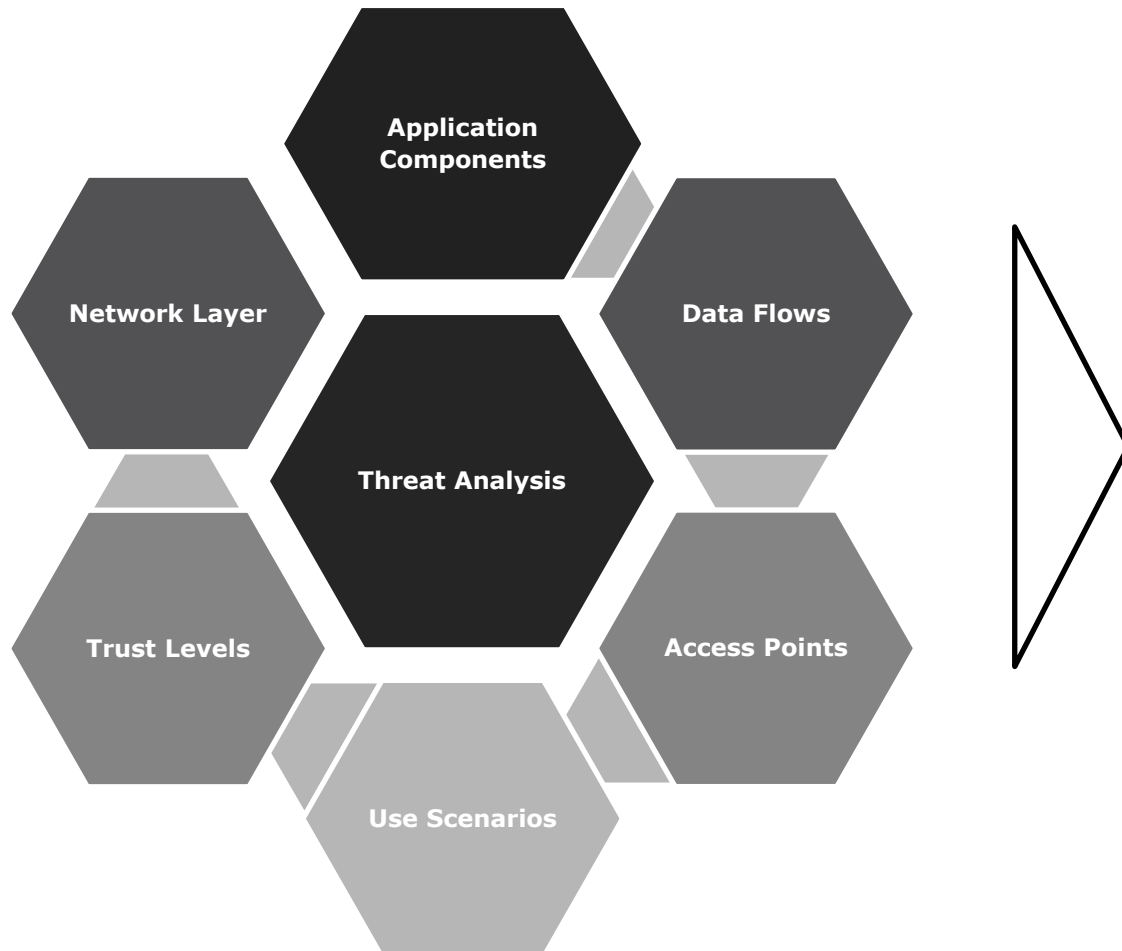
- ▶ Who is the business owners
- ▶ Who is the technical owner
- ▶ Security PoC, IT Lead, System Administrator

Data Asset

- ▶ What data does the asset contain
- ▶ Data classification
- ▶ Asset Value

Objective: Understand the platform and the essential specs of the system

Identify the Threats, Threat Agent...



Correlate all the information to:

1. Depict abuse scenarios for each single component (traversing logical/functional, application and physical layer)
2. Understand what the threat agent might want and what are the goals
3. List and revise all the security controls and countermeasures
4. If new vulnerabilities are identified
 1. Rank the threat
 2. Based on what the assets are: define, prioritize and implement mitigation strategies
5. Extend the research of this threat to other assets in the corporation
6. Identification of residual risk
7. Monitoring strategy to manage the risk

Identify the threats and risks the asset could potentially face!

Anatomy of a Threat Indicator

NetWitness providing the required «building blocks»

Port and protocol
agnostic service
identification

Advanced File Type
Detection

Workstation and server
logs

Geo localization

Threat Intelligence

Endpoint analysis

Event Stream Analysis linking the “dots”

Who

What

When

Where

How

Implementing a Threat Indicator

Plan

- Are all the meta keys available and the values correctly populated?
- Are all the required data enrichment sources configured?

Implement

- What are the different statements?
- What are the conditions to link those statements together?
- What is the most appropriate timeframe?
- Is the ESA Rule Builder enough or Expert mode is required?
- Does the rule trigger in the EPL tryout website*?

Test

- Does the rule validate correctly in ESA?
- Can the rule be deployed without any error?
- Is the rule triggering?
- How does the alert and constituent events look like?

Maintain

- What is the false positive/negative rate?
- How many system resources is the rule using?

ESA Views

Rules

Create alerts based on data in the NWDB using conditions, notifications and enrichments

Services

View statistics for ESA services and rules
Enable and disable rule stats

Settings

View available ESA meta keys, create sources for enrichments and data connections

Summary

View results of alerts by timeline and severity
Drill into alerts

ESA Components

Alert

Output from a rule that matches data in the environment

Template

Convert the rule syntax into code (Esper) that ESA understands - Advanced and Basic

Constituent Events

All of the events involved in an alert, including the trigger event

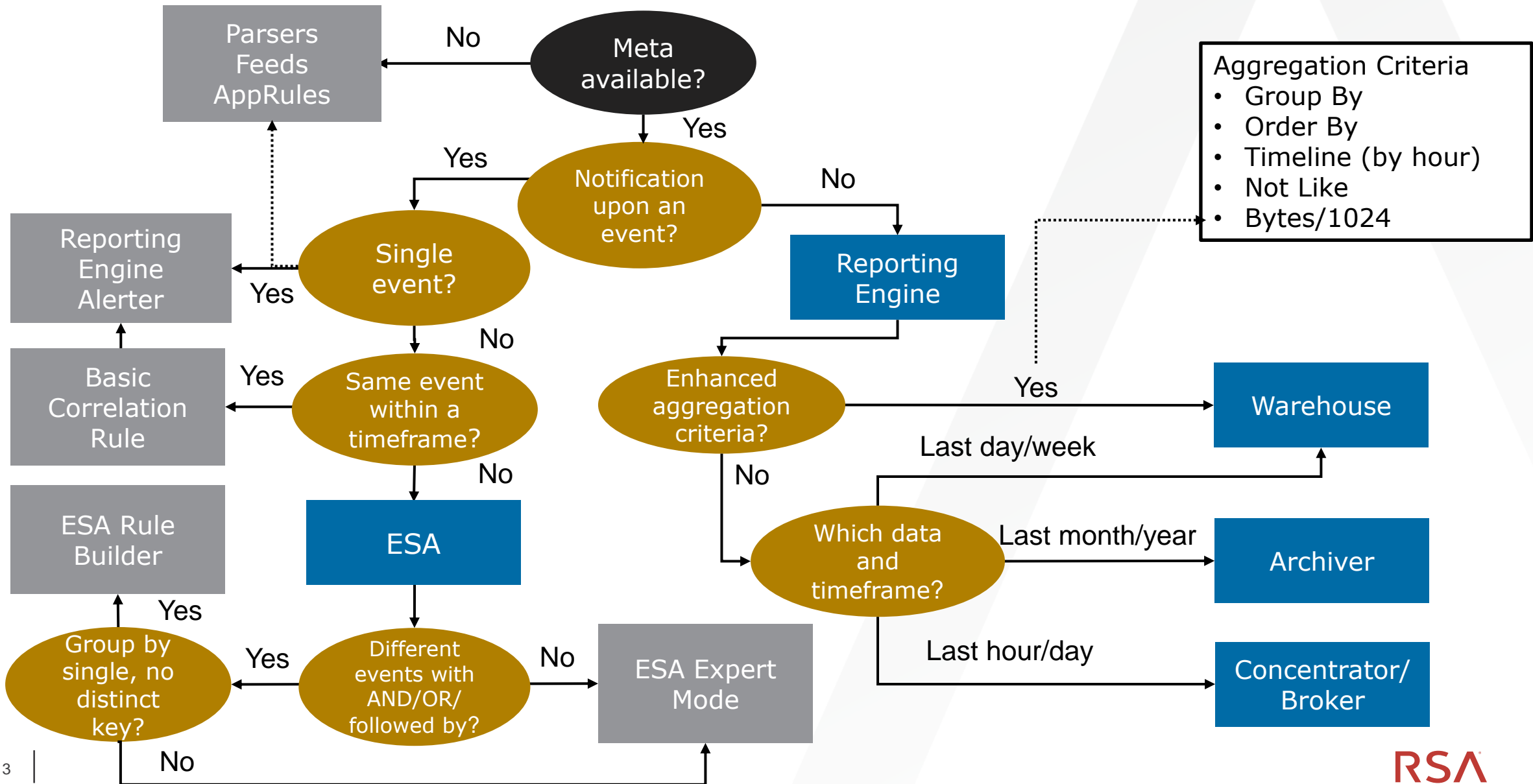
Rule Library

A list of all the ESA Rules that have been created

Deployments

A list of the ESA Rules that have been deployed to an ESA device

Reporting, Correlation and ESA Alerts Decision



CONFIGURING ESA

Adding Data Sources to ESA

Select a data source, Concentrator recommended, from which to gather data for ESA alerts. You can select one or more sources.

The screenshot shows the RSA Security Center interface. The top navigation bar includes 'Administration', 'Hosts', 'Services', 'Event Sources', 'Health & Wellness', and 'System'. The current page is 'ESA - Event Stream Analysis' with a 'Config' dropdown. The 'Data Sources' section is active, showing a list of available services. The 'Concentrator - Concentrator' service is selected, indicated by a checked checkbox.

<input type="checkbox"/>		Name ^	Address	Type
<input checked="" type="checkbox"/>		Concentrator - Concentrator	10.101.240.41	Concentrator
<input type="checkbox"/>		LogDecoder - Log Collector	10.101.240.40	Log Collector
<input type="checkbox"/>		LogDecoder - Log Decoder	10.101.240.40	Log Decoder
<input type="checkbox"/>		PacketDecoder - Decoder	10.101.240.39	Decoder
<input type="checkbox"/>		RemoteLC - Log Collector	10.101.240.43	Log Collector

Configuring Advanced Settings

Preserve events for rules that choose multiple events



Improve performance



Alert Engine

Max Constituent Events

Forward Alerts On Message Bus

Debug Rules?

[Apply](#)

Event Stream Engine

Max Pattern Subexpressions

[Apply](#)

ADD AN ESA DEPLOYMENT

Under the Alerts > Configure > Rules click Add from the dropdown and name the ESA deployment

The screenshot displays the RSA configuration interface. At the top, there are navigation tabs: Alerts, Summary, and Configure. Below these are sub-tabs: Rules, Services, and Settings. The main content area is titled 'Deployment - ESA Demo' and includes a description: 'Deployments map rules from your rule library to the appr'. There is a section for 'ESA Services' with the instruction 'Choose from available ESA Services. Remove services'. A table below this section shows a list of services, with one entry 'nwesalog - Event Stream Analysis' visible. A dropdown menu is open over the table, showing options: Add, Delete, Edit, and Refresh.

Status	Name ^
Deployed	nwesalog - Event Stream Analysis

DEMO

- Configuring ESA
- Components of ESA rules
- Editing a Live Rule
- Alerting within ESA

BASICS OF EVENT PROCESSING LANGUAGE (EPL)

Esper Processing Language

Esper is what allows for advanced correlation of metadata

- Metadata is consumed by the ESA appliance from one or more Concentrators

The data is fed through a stream, which is a sequence of events, called an Event stream

- An example of an EPL rule to alert on everything is:

```
SELECT * FROM Event;
```

- In this rule, you are selecting everything (*) from the stream, which is called Event

- You can add a filter for specific meta from the stream:

```
SELECT * FROM Event(user_dst='Lee')
```

- Or multiple pieces of meta:

```
SELECT * FROM Event(user_dst = 'Lee' AND event_cat_name =  
'User.Activity.Failed Logins');
```

Data Windows

There are several data window types that you can use to see specific views of the data returned by a rule

The `win:length(size)` window extends the specific number of elements into the past

```
SELECT * FROM Event(user_dst IS 'JohnDoe').win:length(5) GROUP
BY user_dst HAVING COUNT(*) = 5;
```

The `win:length_batch(size)` window batches events and releases them when a given minimum number of events has been collected.

```
SELECT * FROM Event(user_dst IS 'JohnDoe').win:length_batch(5)
GROUP BY user_dst HAVING COUNT(*) = 5;
```

The `win:time(time period)` window extends the specific time interval into the past

```
SELECT * FROM Event(user_dst IS 'JohnDoe').win:time(10 sec)
GROUP BY user_dst HAVING COUNT(*) = 5;
```

A Sample Library of EPL Templates – Simple Select

- *Same event repeated multiple times*

EXAMPLE: The same IP Source connecting to the same IP destination within 1 minutes on more than 255 different ports

EPL TEMPLATE:

```
SELECT * FROM Event
(device_class = 'Firewall').win:time_batch(1 min)
GROUP BY ip_src,ip_dst
HAVING COUNT(DISTINCT ip_dstport) > 254;
```

A Sample Library of EPL Templates – Match Recognize

- *Two events without another in the middle*
- *One event and then no more for a timeframe*

EXAMPLE: Same IP blocked by the firewall then allowed but on a different port

EPL RULE:

```
SELECT * FROM Event(device_class = 'Firewall').win:time(10 minutes)
MATCH_RECOGNIZE (
  PARTITION BY ip_src
  MEASURES D as d, P as p
  PATTERN (D P)
  DEFINE
  D as D.category = 'Deny',
  P as P.category = 'Permit'
  AND D.ip_dstport != P.ip_dstport
);
```

A Sample Library of EPL Templates – Pattern

- *Sequence of different events*
- *Same event with different values*

EXAMPLE: User login without a logout within 12 hours

EPL RULE:

```
SELECT * FROM PATTERN
[a = Event(dec_activity = 'Login') ->
(timer:interval(12 hours)
AND NOT Event(user_dst =a.user_dst AND ec_activity='Logout'))];
```


A Sample Library of EPL Templates – Watchlist

- *One event with something in common with another rule*
- *An event not preceded by another event*

EXAMPLE: A device infected by a virus during the last 20 minutes is connecting to a malicious website:

EPL RULE:

```
CREATE WINDOW WatchList.win:time(20 min) (ip_src string);
```

```
INSERT INTO WatchList SELECT ip_src from Event(virusname IS NOT NULL);
```

```
SELECT * FROM Event(threat_source IS NOT NULL)  
WHERE ip_src IN (SELECT ip_src FROM WatchList);
```

A Sample Library of EPL Templates – Baseline

- *A significant change based on a statistical parameter*
- *A comparison between two different timeframes*

EXAMPLE: 500% events raise from a specific system compared to the previous hour

EPL RULE:

```
CREATE WINDOW Baseline.std:groupwin(ip_src).win:length(2) (ip_src
string,num long);
```

```
INSERT INTO Baseline SELECT ip_src, count(*) AS num FROM
Event.win:time_batch(1 hour) GROUP BY ip_src;
```

```
SELECT ip_src,num,sum(num)-num AS PreviousHour FROM Baseline GROUP BY
ip_src HAVING num > 5*(sum(num)-num) and sum(num)-num != 0;
```

EPL Example: Filter Statement with Views

RDP traffic from Same source to Multiple different destinations:

- RDP traffic identified as **service 3389**
- Checks for **IS NOT NULL** on metadata being used in views
- A view is created for **each ip.src** using std:groupwin
- Each **ip.dst must be different** as designated by the std:unique view
- The data window will be open for **180 seconds** or for **3 matching events**
- Group By and Having clause used to keep batches less than 3 events from outputting alert

```
SELECT * FROM Event (  
    ip_src IS NOT NULL  
    AND  
    ip_dst IS NOT NULL  
    AND  
    service = 3389  
).std:groupwin(ip_src)  
.win:time_length_batch(180  
seconds, 3)  
.std:unique(ip_dst)  
GROUP BY ip_src HAVING COUNT(*)  
= 3;
```

EPL Example: Suspicious Activity Correlation

Rule pseudocode:

Three failed login attempts against the VPN...

followed by a **successful login** with the same username on a Unix system...

followed by a **successful connection** through the firewall to a suspicious country from the same IP where the successful login took place...

within 2 minutes

EPL Example: Suspicious Activity Correlation, continued

```
SELECT a.time as time, a.user_dst as user_dst, a.device_ip  
as device_ip, a.device_type as device_type, c.ip_src as  
ip_src, d.ip_dst as ip_dst, d.country_dst as country_dst
```

```
from pattern[every
```

Pattern Matching Stream

```
(a=Event(device_class='VPN' and action='failed') ->  
([2]Event(device_class='VPN' and action='failed' and  
user_dst=a.user_dst)  
  
->(c=Event(device_class='Unix' and ec_activity='Logon' and  
ec_outcome='Success' and user_dst=a.user_dst)  
  
-> d=Event(device_class='Firewall' and  
event_cat_name='Network.Connections.Successful' and  
country_dst in ('Belarus') and ip_src=c.ip_src)))  
where timer:within(2 min)];
```

EPL Example: Suspicious Activity Correlation, continued

```
SELECT a.time as time, a.user_dst as user_dst, a.device_ip  
as device_ip, a.device_type as device_type, c.ip_src as  
ip_src, d.ip_dst as ip_dst, d.country_dst as country_dst
```

```
from pattern[every
```

Three failed logins

```
(a=Event(device_class='VPN' and action='failed') ->  
([2]Event(device_class='VPN' and action='failed' and  
user_dst=a.user_dst)
```

```
->(c=Event(device_class='Unix' and ec_activity='Logon' and  
ec_outcome='Success' and user_dst=a.user_dst)
```

```
-> d=Event(device_class='Firewall' and  
event_cat_name='Network.Connections.Successful' and  
country_dst in ('Belarus') and ip_src=c.ip_src)))
```

```
where timer:within(2 min)];
```

EPL Example: Suspicious Activity Correlation, continued

```
SELECT a.time as time, a.user_dst as user_dst, a.device_ip  
as device_ip, a.device_type as device_type, c.ip_src as  
ip_src, d.ip_dst as ip_dst, d.country_dst as country_dst
```

```
from pattern[every
```

```
(a=Event(device_class='VPN' and action='failed') ->  
([2]Event(device_class='VPN' and action='failed' and  
user_dst=a.user_dst)
```

Successful logins

```
->(c=Event(device_class='Unix' and ec_activity='Logon' and  
ec_outcome='Success' and user_dst=a.user_dst)
```

```
->d=Event(device_class='Firewall' and  
event_cat_name='Network.Connections.Successful' and  
country_dst in ('Belarus') and ip_src=c.ip_src)))
```

```
where timer:within(2 min)];
```


EPL Example: Suspicious Activity Correlation, continued

```
SELECT a.time as time, a.user_dst as user_dst, a.device_ip
as device_ip, a.device_type as device_type, c.ip_src as
ip_src, d.ip_dst as ip_dst, d.country_dst as country_dst

from pattern[every

(a=Event(device_class='VPN' and action='failed') ->
([2]Event(device_class='VPN' and action='failed' and
user_dst=a.user_dst)

->(c=Event(device_class='Unix' and ec_activity='Logon' and
ec_outcome='Success' and user_dst=a.user_dst)

-> d=Event(device_class='Firewall' and
event_cat_name='Network.Connections.Successful' and
country_dst in ('Belarus') and ip_src=c.ip_src)))

where timer:within(2 min)];
```

Successful outbound

EPL Example: Suspicious Activity Correlation, continued

```
SELECT a.time as time, a.user_dst as user_dst, a.device_ip  
as device_ip, a.device_type as device_type, c.ip_src as  
ip_src, d.ip_dst as ip_dst, d.country_dst as country_dst
```

```
from pattern[every
```

```
(a=Event(device_class='VPN' and action='failed') ->  
([2]Event(device_class='VPN' and action='failed' and  
user_dst=a.user_dst)
```

```
->(c=Event(device_class='Unix' and ec_activity='Logon' and  
ec_outcome='Success' and user_dst=a.user_dst)
```

```
-> d=Event(device_class='Firewall' and  
event_cat_name='Network.Connections.Successful' and  
country_dst in ('Belarus') and ip_src=c.ip_src)))
```

```
where timer:within(2 min)];
```

Within two minutes

Comments

Comments can appear anywhere in the EPL or pattern statement text where whitespace is allowed.

Comments can be written in two ways: slash-slash (// ...) comments and slash-star (/ * ... * /) comments.

Examples:

Slash-slash comments extend to the end of the line

```
// This comment extends to the end of the line.
```

```
Select * from Event // this is a slash-slash comment
```

Slash-star comments can span multiple lines:

```
/* This comment is a "slash-star" comment
```

```
that spans
```

```
multiple lines.
```

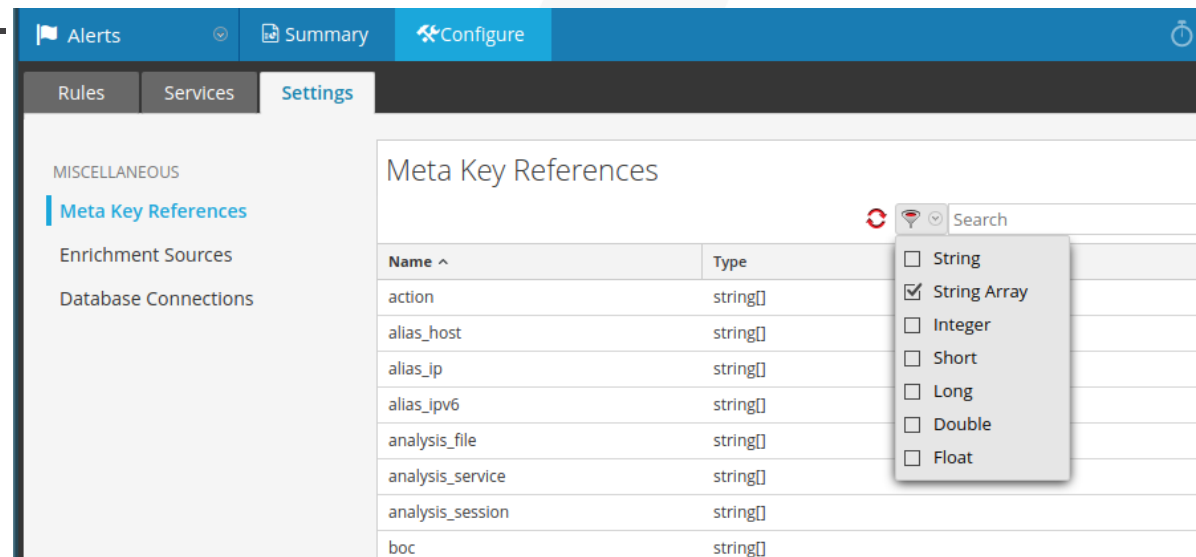
```
*/
```



HANDLING MULTI- VALUED META

MULTI-VALUED META OVERVIEW

- Multi-valued meta keys are typed differently between the default Esper implementation and the one developed for ESA
- Esper within ESA uses vectors while the default implementation assumes the array type
- Vectors are needed to support grouping and aggregation for multi-valued meta
- Custom functions have been developed to support equivalent functions to the default implementation
- To see if a particular key is typed as multi-valued, go to Alerts > Configure > Settings > Meta Key References. Filter by String Array.



The screenshot shows the RSA configuration interface. The top navigation bar includes 'Alerts', 'Summary', and 'Configure'. The left sidebar has 'Rules', 'Services', and 'Settings'. Under 'Settings', there is a 'MISCELLANEOUS' section with 'Meta Key References' selected. The main content area displays a table titled 'Meta Key References' with columns 'Name ^' and 'Type'. A dropdown menu is open over the 'Type' column, showing options: String, String Array (checked), Integer, Short, Long, Double, and Float. The table lists several keys, all with 'string[]' as their type.

Name ^	Type
action	string[]
alias_host	string[]
alias_ip	string[]
alias_ipv6	string[]
analysis_file	string[]
analysis_service	string[]
analysis_session	string[]
boc	string[]

EXACT MATCH MULTI-VALUED META

- Use ANY(meta) if you want at least one of the values in the meta key to match the value

```
@RSAAlert SELECT * FROM Event (  
    'Bo' = ANY(username) OR 'Joe' = ANY(username)  
)
```

- Use ALL(meta) if you want every value within the meta key to exactly match

```
@RSAAlert SELECT * FROM Event (  
    'Bo' = ALL(username) OR 'Joe' = ALL(username)  
)
```

CASE-INSENSITIVE MATCH MULTI-VALUED META

- **ESA version 10.6+ custom functions**

- **isOneOfIgnoreCase** triggers when at least of one the lower-case list of values are contained case insensitively

```
@RSAAlert SELECT * from Event(  
    isOneOfIgnoreCase(username,{'alpha','beta','gamma'})  
)
```

- **isNotOneOfIgnoreCase** triggers if none of the lower-case list of values are contained case insensitively

```
@RSAAlert SELECT * from Event(  
    isNotOneOfIgnoreCase(username,{'alpha','beta','gamma'})  
)
```


FUZZY COMPARE MULTI-VALUED META

■ Prior to version 10.6.4

- Cast the meta to a string and make the comparison

```
@RSAAlert SELECT * FROM Event (  
    cast(alias_host, string) LIKE '%www.xn-%'  
)
```

■ Version 10.6.4+ custom functions

- Use the user-defined Esper functions
 - makeLike to perform LIKE comparison against multi-valued meta keys
 - matchRegex to perform regular expression comparison against multi-valued meta keys

```
@RSAAlert SELECT * FROM Event (  
    matchLike(alias_host, '%www.xn-%')  
)
```

```
@RSAAlert SELECT * FROM Event (  
    matchRegex(alias_host, '.*www.xn-.*')  
)
```

INTERSECTION MULTI-VALUED META

- Intersection is an expression comparing two multi-valued meta keys and returning true if there is a least one element common between them
- Custom function supported in 10.6.2+
- May also use other Java-supported syntax such as `getIntersection(alias_host, e1.alias_host).contains("abc")`

@RSAAlert

```
SELECT * FROM PATTERN [
```

```
  /* Statement: FireEye WebMPS Exploit.Kit */
```

```
    e1=Event(policy_name .toLowerCase() LIKE 'exploit.kit%' A
```

```
    AND user_agent .toLowerCase() NOT IN ( 'sam proxy check' ) AND url IS NOT NULL )
```

```
  ->
```

```
  /* Statement: followed by Proxy Log NOT 403 */
```

```
    e2=Event(device_type .toLowerCase() IN ( 'mcafeewg' )
```

```
    AND result_code .toLowerCase() NOT IN ( '403' ) AND url IS NOT NULL AND
```

```
    ip_src=e1.ip_src AND getIntersection(url, e1.url).size() > 0)
```

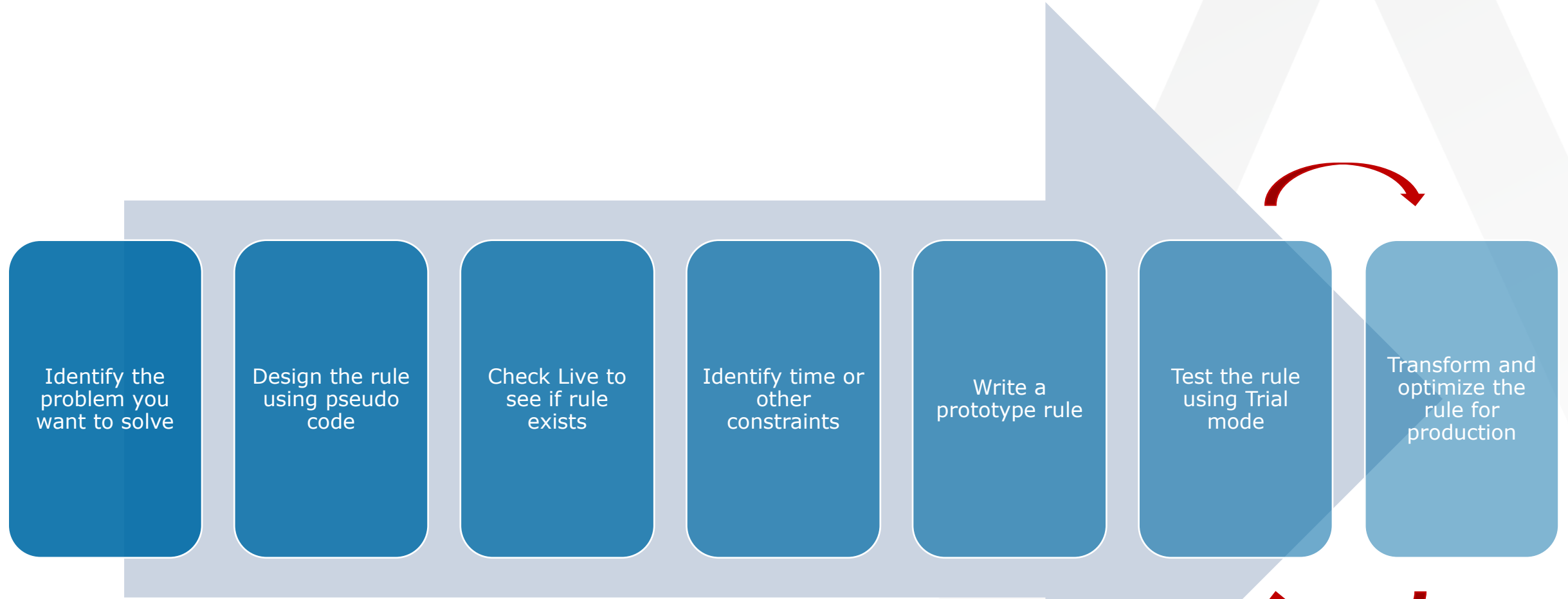
where timer:within(60 Minutes)

DEMO

- Esper Online tool
 - Setting up the scenario
 - Simple event stream filtering (with varying views)
 - Results review

CREATING ESA RULES

Recommended Process for Creating ESA Rules



Live Rules

There are several ESA rules available from Live that you can use as a starting point

The screenshot displays the RSA Live interface. On the left is a navigation menu with options: Live, Dashboard, Investigation, Incidents, Alerts, Reports, Administration, Live (selected), Profile, and Sign Out. The main content area is titled 'Live' and includes a search bar, 'Configure', and 'Feeds' buttons. The 'Search Criteria' section has a 'Keywords' field and a 'Resource Types' dropdown menu, which is highlighted with a red box and shows 'RSA Event Stream Analysis Rule' selected. Below this are 'Tags' and 'Required Meta Keys' fields. The 'Matching Resources' section features a table with columns for 'Subscribed' and 'Name'. The table lists various system events, each with a checkbox in the 'Subscribed' column.

Subscribed	Name
<input type="checkbox"/>	Account Added to Administrators Group and Removed
<input type="checkbox"/>	Account Removals From Protected Groups on Domain Contr...
<input type="checkbox"/>	Active Directory Policy Modified
<input type="checkbox"/>	Adapter Entered Promiscuous Mode
<input type="checkbox"/>	Adapter in Promiscuous mode after Multiple login attempts
<input type="checkbox"/>	Adapter in Promiscuous mode after User Creation and Login
<input type="checkbox"/>	Advanced Rule Template
<input type="checkbox"/>	Aggressive Internal Database Scan
<input type="checkbox"/>	Aggressive Internal NetBIOS scan

Edit Live Rule

You can use the Live rule as is or you can use it as a starting point

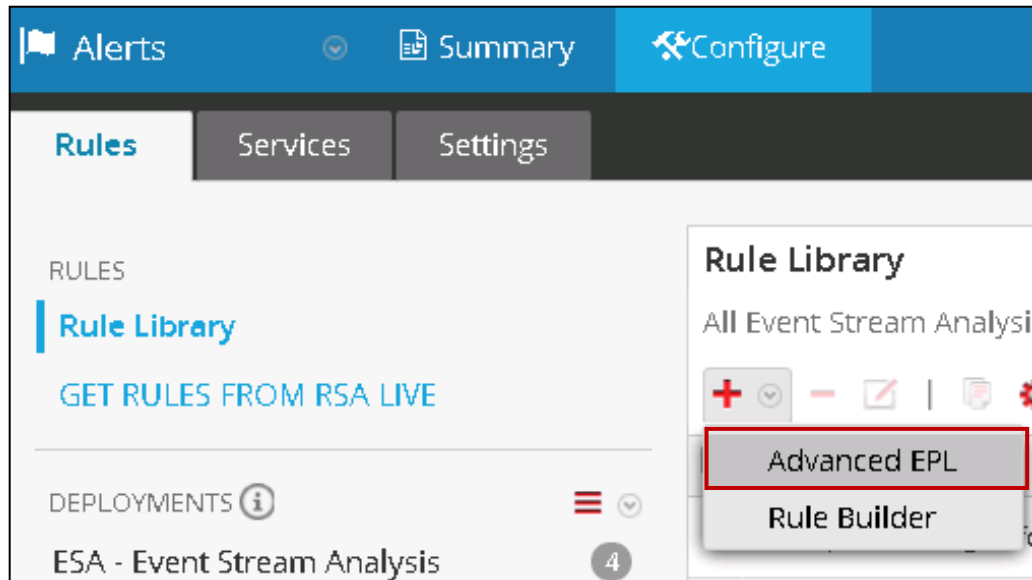
You can edit the syntax by:

1. Editing the rule
2. Selecting **Show Syntax**
3. Copying the syntax
4. Making edits
5. Creating a new rule using the new syntax

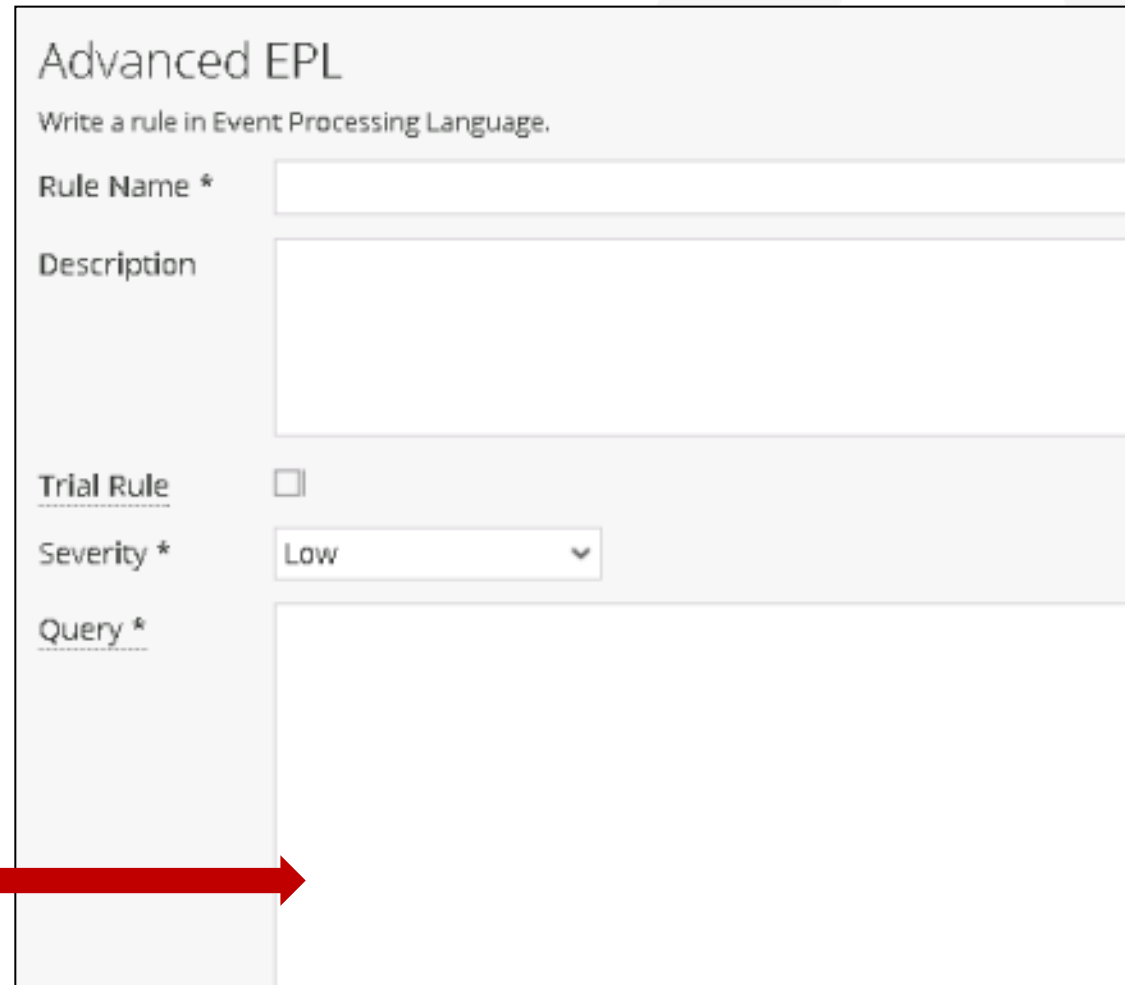
You can also change the rule parameters

Creating Advanced Rules

To create an EPL rule, select Advanced EPL in the Rule Library



Enter the EPL syntax into the query field

A screenshot of the 'Advanced EPL' configuration form. The form is titled 'Advanced EPL' and includes the instruction 'Write a rule in Event Processing Language.' The fields are: 'Rule Name *' (text input), 'Description' (text area), 'Trial Rule' (checkbox), 'Severity *' (dropdown menu set to 'Low'), and 'Query *' (text area). A red arrow points from the text 'Enter the EPL syntax into the query field' to the 'Query *' field.

Writing and Testing Rules Guidelines

Can the rule be implemented with the Wizard?

Do you need to use expert mode?

Check the requirements:

Are all meta keys available?

Do we have all the values to trigger?

Check whether the rule is correct:

Does the rule work in the EPL tryout website?

Does the rule validate correctly in ESA?

Can the rule be enabled without any errors?

Check whether the rule works:

Is the rule triggering?

How does the alert generated look?

Are the constituent events what we were expecting?

EPL tryout website: <http://esper-epl-tryout.appspot.com/epltryout/mainform.html>

EPL Reference Guide: <http://esper.codehaus.org/esper/documentation/documentation.html>

DEMO

- Basic Rule Builder
- Advanced Rules

LAB

- Edit a Live Rule
- Create a Basic Rule
- Create an Advanced Rule

TROUBLESHOOTING ESA RULES

DEPLOYING RULES WITH CUSTOM META

- You may need custom meta that is not currently collected to enrich an ESA rule
- Deploying an ESA rule with meta keys not within its schema will cause an error

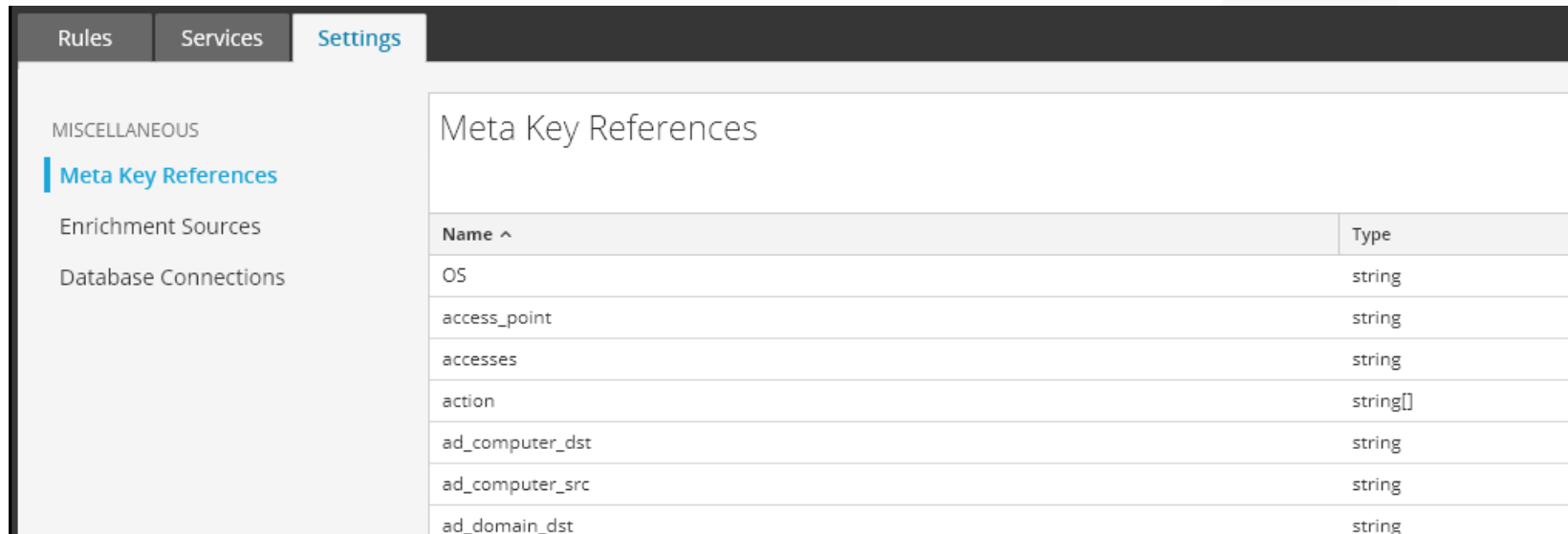
The image shows two overlapping windows. The top window is a dark grey error dialog with a red 'X' icon and the text: "ESA was unable to deploy one or more rules, and these rules were disabled. Common issues include: missing metadata, invalid rule syntax, and unavailable external connections at the time of deployment." Below the text is a blue "OK" button. The bottom window is titled "Log Message" and displays the following details:

Timestamp	2017-08-02T11:15:35.884
Logger Name	com.rsa.netwitness.core.epl.EplModuleManager
Thread	Carlos@69e309fa-149(run(SetEplModule))(admin)
Level	WARN
Message	Esper deployment of module "Exploit Kit" (id=5981b394e4b05a5f69148eb1(default)) failed. Reason: Deployment failed in module 'Module_1922925317_Alert' in module url '5981b394e4b05a5f69148eb1' in expression '@RSAAlert select * from Event(policy = 'exploit kit')': Failed to validate filter expression 'policy="exploit kit": Property named 'policy' is not valid in any stream [@RSAAlert select * from Event(policy = 'exploit kit')]

The error message in the log is circled in red.

DEPLOYING RULES WITH CUSTOM META

- In order for ESA to recognize custom meta, you will need to:
 - (Logs only) Ensure the in meta is marked with the flag of None in the **tablemap-custom.xml**
 - Add the meta in the **index-concentrator-custom.xml** file
 - Send in events with the custom meta. ESA will sync with the updates to the Concentrator's index file once it receives the events
 - To confirm a key has been recognized by ESA, go to **Alerts > Configure > Settings > Meta Key References**



The screenshot shows the RSA Security Center interface. The top navigation bar includes 'Rules', 'Services', and 'Settings'. The left sidebar menu is expanded to 'Settings', with 'Meta Key References' selected. The main content area displays the 'Meta Key References' configuration page, which contains a table with the following data:

Name ^	Type
OS	string
access_point	string
accesses	string
action	string[]
ad_computer_dst	string
ad_computer_src	string
ad_domain_dst	string

ESA Time

It is critical that ESA's server time be aligned with the rest of Security Analytics

- If you can use SA as the NTP server, that is the best option

Esper has no real sense of time; rather we tell Esper when to advance time

For ESA we need to determine how to tell Esper that time has advanced

Presently, we advance time based on when an event is received from the Concentrator:

1. ESA receives the Event
2. ESA "stamps" the Event with the current ESA Server Time
3. ESA processes the events

Time Concerns

Some log sources send logs in batches rather than immediately

If a Concentrator is aggregating from multiple Decoders, this can result in time-order issues

Packet Decoders sort packets out of time order

Aggregation can occur out of time order

Latency: Someone may generate a failed logon at 10:03:05, but due to basic network latency, ESA may not see it until 10:03:08. If that's the case, according to ESA, what time did the failed logon occur?

Time Scenario 1

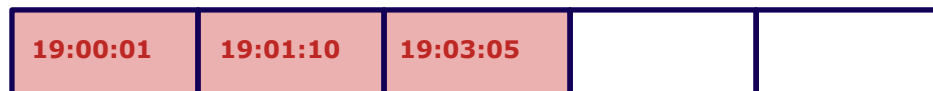
Event Received: 19:00:01 (Window generated & Event added)



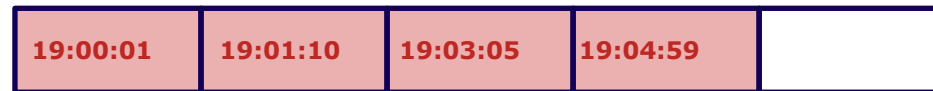
Event Received: 19:01:10



Event Received: 19:03:05



Event Received: 19:04:59



If the 5th Event is received at 19:05:00, what happens?

- **Alert!**

If the 5th Event is received at 19:05:02, what happens?

- **No Alert, but 19:00:01 disappears and the other times slide one slot to the left; you now have 4 Events with the first Event at 19:01:10**

Time Scenario 2

Event Received: 19:00:01 (Window generated & Event added)



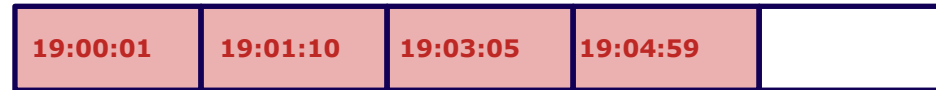
Event Received: 19:01:10



Event Received: 19:03:05



Event Received: 19:04:59



If the 5th Event is received at 19:05:00, what happens?

- **Alert!**

If the 5th Event is received at 19:05:02, what happens?

- **The whole window “tumbles” and starts over with a new “window” containing one Event at 19:05:02**

Time and Windows

Let's highlight a few potential issues that could arise from the previous scenarios

The Window could be open for an extended period of time

Think of a tumbling window – if it receives 4 events in 60 seconds, but doesn't receive the 5th event for 10 minutes

- Remember: We tell Esper that time has advanced – it doesn't know that X minutes have gone by

Events may not be received from ESA in time for the Alert to trigger

ESA BEST PRACTICES

CONFIDENTIAL

RSA

Best Practices Summary by Task

Writing Rules

Create alerts for actionable events

Configure new rules as trial rules

Configure Alert notifications after testing your rule

Make rules specific to limit resource usage

Deploying Live Rules

Deploy RSA Live rules in small batches

Read the rule descriptions provided with RSA Live rules

Set your parameters

Deploying Rules

Deploy rules in small batches so you can observe how they react in your environment

Test rules before you configure alert notifications

Monitor system health as a part of your deployment process

System Health

Configure the alerts database to maintain a healthy level of alerts

Set up new rules as trial rules and tune as needed

Set up thresholds in Health & Wellness to alert when memory usage is high

Trial Rules

To avoid creating rules that use excessive memory, you can create a trial rule by setting the global threshold for the percentage of memory the rule may use

If the threshold is exceeded, the rule is disabled

A trial rule allows you to:

- Deploy the rule with an added safeguard
- Optionally, view a snapshot of memory utilization to determine whether the rule creates memory issues
- Know if you must modify the rule criteria to improve performance

As a best practice, when you add a new rule or edit an existing rule, select the Trial Rule option

Writing for Accuracy

Use `select *` to retrieve all event properties. This is required for working EPL.

- Be aware that a maximum of 100 events per alert are returned (configurable at device)

Be sure to include a check for IS NOT NULL with metadata used within statement views

Use annotation to alert a statement **@RSAAlert**

- Without the @RSAAlert annotation no alerts are generated
 - Add @**RSAAlert** before the statement that is supposed to generate the alert
 - It can be also used for troubleshooting complex rules

Writing for Accuracy, continued

EPL is case-sensitive

- Check to see how the meta value is written in **Investigation > Events > View All Meta** or use the **.toLowerCase()** function

Some meta keys are also reserved EPL keywords

- They have to be escaped: ``group` = 'admin account'`

Some meta keys are defined as arrays

- A text comparison would generate an error
- A different syntax is required: `'GET' = any (action)`

Writing for Performance

Include boundaries for statements such as time and counts

- An alert without boundaries may continuously trigger

Use Match Recognize whenever possible for pattern-based event use cases due to lower memory use

Avoid creating a large number of groups with a small or zero number of data points

- A highly selective metadata field like event.desc could create a separate view for a majority of events

Instead of where clauses, directly filter the event streams

ESA Best Practices

- **When creating Rules, ensure you are focused on notifications for Events that require an immediate action**
 - If you're writing a Rule for an Event that does not require immediate action or simply so you have an awareness, then the Reporting Engine is a better solution
 - Reduces overhead on ESA
- **Configure Alert notifications after you have completed the Rule testing and tuning**
 - This prevents an email or notification issue

ESA Best Practices, continued

- **Rules need to be as specific as possible**
 - Ensure that you are only reviewing/capturing the meta required to trigger the Rule
 - Keep your time window as small as you can
- **Remember to tune Alerts so they are only capturing content you are interested in**
 - Helps with overhead on ESA
 - Limits the number of Alerts the SOC Analyst has to follow-up on
- **RSA recommends deploying Live Rules in small batches....**
 - ...or deploy them one at a time
 - If something goes wrong, you are not left wondering which Rule is the problem
 - If you get flooded with Alerts, you can act more quickly

ESA Best Practices, continued

- **Always monitor system health!**
- **Set up Alerts in Health and Wellness for ESA performance measurement**
- **Don't let the Database grow too large; otherwise you'll start taking some performance hits**
 - This will affect every piece of ESA – writing to the DB, Summary, Alerting, etc.

@HINT WITH GROUPING WINDOWS

- Recommend not using the view 'std.groupwin' if possible to avoid potential high memory use.
- If needed, then use the `@Hint("reclaim_group_aged=age_in_seconds")`
- The `@Hint("reclaim_group_aged=age_in_seconds")` hint instructs the engine to discard aggregation state that has not been updated for the specified time period
 - The `@Hint` should be equal to the time of the data window (unless using a batch window)
 - Batching windows retain the previous and existing batch in memory. For batch windows, double the amount of time

```
@Hint('reclaim_group_aged=120')  
SELECT * FROM Event(  
  medium = 32  
  AND (msg_id.toLowerCase() = 'post')  
  AND user_dst IS NOT NULL  
  AND extension IN ('pdf','doc','xls','docx','xlsx','ppt','pptx') AND rbytes > 10000  
  AND device_type='cacheflowelff'  
  AND alert IN ('VIP', 'CritAsset')  
) .std:groupwin(user_dst)  
  .win:time_length_batch(60 seconds,20)  
GROUP BY user_dst HAVING COUNT(*) = 20;
```

@SUPPRESOVERLAPPINGMATCHES

- A new thread will be created for every 'a' event in the PATTERN statement below
 - This means that multiple 'a' events will match with the same 'b' event.
 - This could result in unexpected and undesirable number of alerts for the same user during the time window.
 - Use @SuppressOverlappingMatches with the PATTERN syntax using every

```
SELECT * FROM PATTERN @SuppressOverlappingMatches [  
    every a = Event(device_class='Web Logs' AND host_dst = 'icanhazip.com')  
  
    -> b = Event(category LIKE '%Botnet%' AND device_class='Web Logs' AND user_dst=a.user_dst)  
    where timer:within(300 seconds)  
]
```

STRICT MATCH RECOGNIZE

- Greedy regular expression matching is known to have a negative performance impact due to the need of the engine to backtrack and hold state
- As a general rule, use strict pattern matching within a match recognize statement

Greedy Regular Expression

```
SELECT * FROM Event(  
ec_activity='Create' OR ec_activity='Delete'  
)  
.win:time(7200 seconds)  
match_recognize (  
partition by user_src measures C as c, D as d  
pattern (C+ D+)  
define C as C.ec_activity='Create' ,  
D as D.ec_activity='Delete');
```

Strict Regular Expression

```
SELECT * FROM Event(  
ec_activity='Create' OR ec_activity='Delete'  
)  
.win:time(7200 seconds)  
match_recognize (  
partition by user_src measures C as c, D as d  
pattern (C D)  
define C as C.ec_activity='Create' ,  
D as D.ec_activity='Delete');
```

OUTPUT RATE LIMITING

- If a rule is triggering frequently, you may only want to store the first occurrence within a time period per unique meta value
 - Alerts are not stored and taking up space in the database
 - Separate from notification suppression, which may be done within the UI and does not influence alert storage
- Alerts below are suppressed per user_dst using 'output first every' syntax.
 - Only the first alert within the 60 minute time window will be stored in the database and alerted
 - Allow constituent events to be retained within the alert by using window aggregation 'window(*)'. Result without window aggregation would be only the first of the 20 events per user_dst.

```
SELECT window(*) FROM Event( medium = 32
AND alert_id LIKE 'known_bad%'
AND alert IN ('VIP' , 'CritAsset')
).std:groupwin(ip_src)
.win:time(1 Minutes)
GROUP BY ip_src HAVING COUNT(*) >= 20 ouptut first every 60 minutes;
```

ADVANCED USE CASES

Enrichments

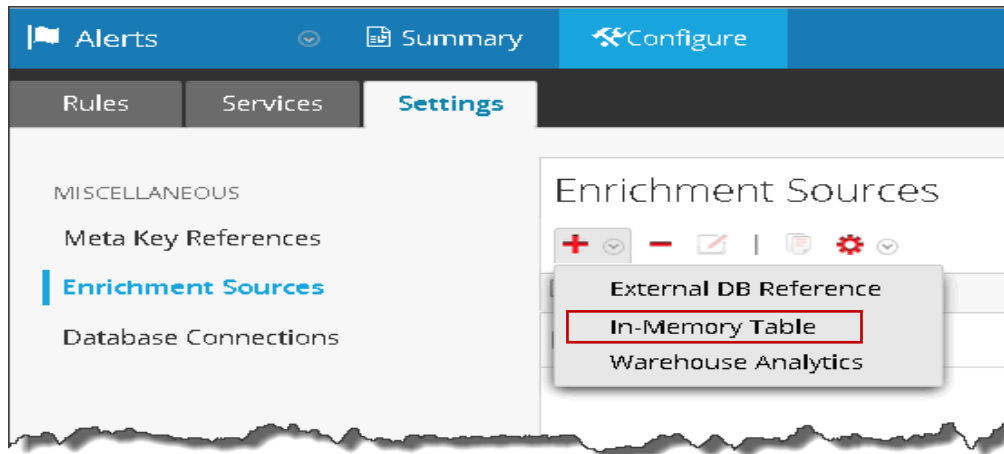
- Allow inclusion of contextual information into correlation logic and alert output using lookups in the following sources:
 - External DB Reference
 - In-Memory Table (Esper Named Windows)
 - RSA Data Science
- Make the alert more meaningful

Alerts generated from a rule can incorporate additional information, such as:

- Risk score of the destination domain
- Supervisor of the employee to which the alert pertains
- Name of the user

Configuring Enrichments

You configure the enrichment source from which you want to pull data by selecting the source, for example, In-Memory Table, and adding the table parameters to the configuration



The 'In-Memory Table' configuration dialog box is shown. It includes the following fields and options:

- Upload Type:** Adhoc Recurring
- Enable:**
- User-Defined Table Name*:** UserRole
- Description:** This table provides user context, such as department, ip address, and workstation name.
- Import Data:** PeopleFeed.csv
- Expert Mode:**
- Persist:**
- Table Columns:**

	Name	Type
<input type="checkbox"/>	Name	
<input type="checkbox"/>	/ip/	string
<input type="checkbox"/>	user	string
<input type="checkbox"/>	workstation	string
- Max Rows:**

[For information on how to define and use an In-Memory Table, see the documentation](#)

Configuring Enrichments, continued

Add the enrichment to the alert

Enrichments + -

Type	Enrichment Source	ESA Event Stream Meta	Enrichment Source Column Name
<input type="checkbox"/> In-Memory Table	UserRole	user_dst	user

Save Close Show Syntax

The enrichment data appears in the Event meta of the alert

User Account Password Change

A modification to an administrative account password

2014-11-21T18:41:06

Severity High

Of Events 1

Event Meta **Events**

Date	Source	Destination	Username	Alias Host
2014-11-21T18:40:44			admin	

Additional Meta

UserRole { address : 10.4.3.46 , role : ADMIN , user : admin , workstation : ADMINTJASPERD1C }

alert.id account:password-change

category User Account Management

device.class Windows Hosts

device.ip 10.101.240.40

device.type winevent_nic

did logdecoder

ec.activity Modify

ec.outcome Success

ec.subject Password

ec.theme Password

esa.time 1416595266023

event.cat.name User.Management.Password.Modification

event.computer dc.mstomlin.local

NAMED WINDOWS

- **A named window is a**
 - Global data window
 - Can take part in many statement queries
 - Can be inserted-into and deleted-from by multiple statement
- **Events leave a named window either because**
 - The expiry policy of the data window removes events
 - Statements that use the on delete clause to explicitly delete from a named window.
- General flow is to create the window, insert into the window and optionally delete from it

Create Window `HttpsJoinedWindow.win:time(15 minutes)(device_class string, ip_dstport integer, service integer , tcp_dstport integer, device_type string, ip_dst string);`

INSERT INTO `HttpsJoinedWindow`

@RSAAlert INSERT INTO `HttpsIntrusionTrigger` `SELECT * FROM`

ON `HttpsIntrusionTrigger` **DELETE** `from` `HttpsJoinedWindow`

JOINING EVENTS

- Different types of joins similar to SQL are possible in Esper EPL
- Use to correlate events that may arrive in any order
- Typically would combine with a named window and on delete statements to alert on unique events combinations

```
@RSAAlert
```

```
INSERT INTO HttpsIntrusionTrigger
```

```
SELECT * FROM
```

```
  HttpsJoinedWindow(ip_dst IS NOT NULL and device_class IN ('IPS', 'IDS', 'Firewall') AND ip_dstport=443) as s1,
```

```
  HttpsJoinedWindow(ip_dst IS NOT NULL and service!=443 and tcp_dstport=443) as s2,
```

```
  HttpsJoinedWindow(ip_dst IS NOT NULL and device_type='rsaecat') as s3
```

```
  where s1.ip_dst = s2.ip_dst and s1.ip_dst = s3.ip_dst;
```

WORKING WITH CONTEXTS

- Aggregates events over time periods (overlapping or non-overlapping) without retaining any events in memory
- Can apply to multiple statements, but cannot force load of context similar to an enrichment
- Example context for ‘Non Working Hours’
 - Set the working hours as '09:00' – '18:00'
 - Any 'event.cat.name LIKE system.config%' outside the working hours will trigger

```
create context NotWorkingHours start (0, 18, *, *, *) end (0, 9, *, *, *);
```

```
context NotWorkingHours select * from Event(event_cat_name LIKE 'system.config%');
```

DYNAMIC NAMING

- The @Name annotation is used to dynamically generate statement names in ESA alerts and notifications
- This annotation has meta keys enclosed in curly brackets
- Meta key name truncated after 64 characters with “...”
- Statement name truncated after 128 characters with “...”
- Supported on version 10.6.4+

@Name (“Login Event to {ip_src} by {user_dst}”)

LAB

- Add An Enrichment
- Create a Context
- Use a Data Window

RESOURCES

- Details on the Esper engine behavior and language:
 - http://espertech.com/esper/release-5.3.0/esper-reference/html_single/
- Query solutions for event use cases:
 - http://esper.codehaus.org/tutorials/solution_patterns/solution_patterns.html
- Contact EsperTech customer support to create a free account and ask questions about the language:
 - <http://espertech.helpserve.com>
- Esper mailing list archive and discussion board:
 - <http://esper.codehaus.org/about/esper/maillinglist.html>
- Esper EPL Online, to test advanced EPL statements with schema-based scenarios. This is a good resource for when you start the development process outside of RSA NetWitness:
 - <http://esper-epl-tryout.appspot.com/epltryout/mainform.html>
- RSA Link
 - RSA Security Analytics Alerting Using ESA Guide
 - Content Quickstart Guide