FireEye

FireEye and Splunk: Intro to Integration

Table of Contents

Introduction

Are you a Splunk ninja that just purchased a FireEye appliance? If so, this paper should help introduce you to FireEye and Splunk integration options in less than an hour. The majority of this information is designed to walk the reader through building a dashboard while learning how to carve Splunk data. For those readers that want to quickly get to one possible end product, they should start with the "Sample FireEye Dashboards" section.

Current Integration Efforts

If your organization is using the latest version of Splunk (6.x), try out our free FireEye App for Splunk Enterprise v3 (http://apps.splunk.com/app/1845/). This new app provides increased flexibility by supporting multiple FireEye appliances as well as multiple protocols and formats for sending data to Splunk. This app may not be fully backward compatible because it takes advantage of many Splunk 6.x features that were not previously available.

If your organization is still using Splunk version 4.x or 5.x, you can easily download and use the free--but unsupported--Splunk for FireEye v2 app to integrate the two technologies. This Splunk app utilizes and parses FireEye's rich extended XML output. This downloadable app is available here: https://apps.splunk.com/app/409/.

The rest of this article is written for those that want to start from scratch or start from one of the above apps and learn to customize them. This article will outline various protocols and formats available from FireEye and explore the parsing options provided by Splunk.

Architecture Note

The devices linked to Splunk will depend heavily on the environment's architecture—mainly the number and type of appliances you have deployed. This may also have an effect on adding or removing fields from our provided Splunk queries. Fortunately, Splunk is flexible allowing users to choose the fields they want displayed. Let's quickly review what your architecture may look like.

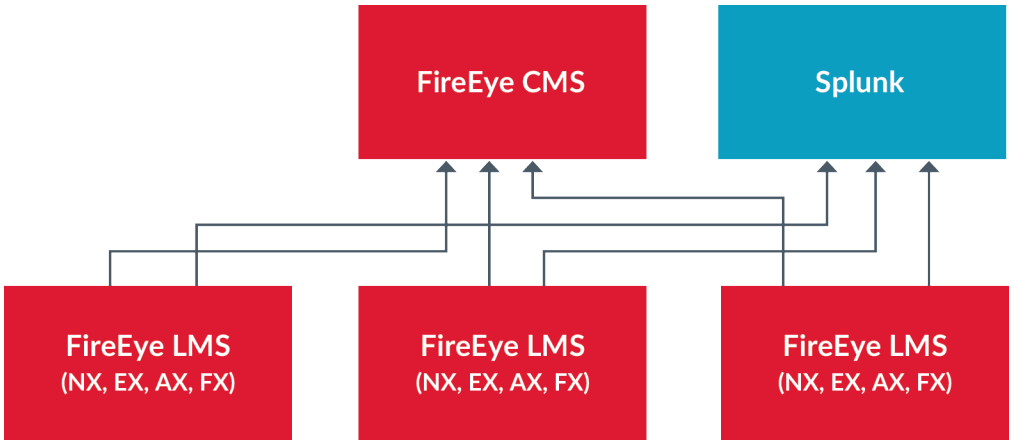
FireEye LMS -> Splunk Architecture:

This is the smallest architecture because there are only one or two FireEye devices with no CMS. In this case the event IDs in the FireEye Local Management System (LMS) Appliances will match the event IDs in Splunk. Additionally, the source appliance field may not be as important with one FireEye sensor because it will be evident which appliance the event originated from. This will all make more sense later.

Having only one appliance may be somewhat of a rare deployment unless it is a fairly small organization or sub organization. Due to the limitations of our test environment, we developed most of the dashboards in this type of a setup, however, we added the field for originating appliance in the enhanced queries for clients that have multiple appliances.

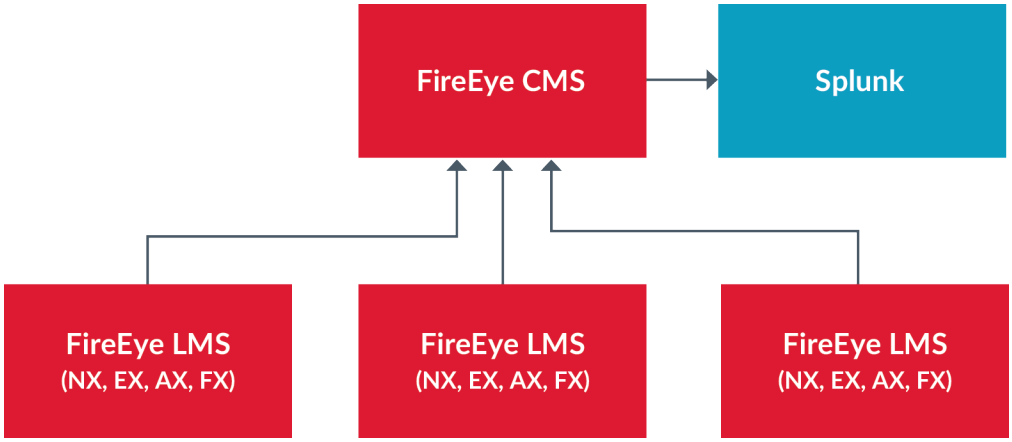
Multiple FireEye LMS -> Splunk Architecture:

This architecture is currently required even if a CMS is present because the CMS itself cannot yet send notifications—it instructs each LMS to do so. With multiple appliances sending events, we want our Splunk search to identify the appliance that witnessed the event. Advantage: Event IDs in LMS and Splunk will match. Event notification does not have to go to CMS first, which may be slightly faster. Disadvantage: Events IDs in CMS and Splunk will not match.



FireEye CMS -> Splunk Architecture:

This architecture has become a possibility in the 7.1 version of the FireEye CMS. It functions similarly to the first architecture where only one LMS is feeding the Splunk receiver. Advantage: Event IDs in CMS and Splunk will match. Disadvantage: Event IDs in LMS and Splunk will not match.



Demo Setup

If you are looking to demo Splunk to see how it fits in your environment, you can download a free trial to try it out before you buy it. For our demonstration purposes, we installed Splunk on a Kali Linux VM that we had sitting around. To download and install your free trial of Splunk, use the following steps:

Download

- Go to: http://www.splunk.com/download
- Register for a free account
- Download the appropriate package
  - Kali 32-bit VM uses: splunk-6.0.1-189883-Linux-i686.tgz
- Drag and drop the tgz into VMware

Extract Splunk

- tar -zxvf splunk-6.0.1-189883-Linux-i686.tgz -C /opt
  - Splunk is now extracted to /opt/splunk

Start Splunk

- /opt/splunk/bin/splunk start
- Accept the EULA and you can now use Splunk

Creating Connectors

Now that we have Splunk ready to go, we have to create the connection between the FireEye and Splunk devices. This involves creating a Splunk listener and configuring the FireEye device to send the data.

Splunk Listener

The Splunk listener needs to be configured so it can receive data from other devices. Perform the following steps to create the listener:

Log into the web UI using a web browser: http://<SplunkBox>:8000
- username: admin
- password: changeme

*Note: It will prompt you to change the password upon first login.

Set up the Splunk listener:
- Click the “Add Data” button
- Select “Syslog”
- Select “Consume syslog over UDP”
- Enter “514” for the port and click the “Save” button
- Click the “Back to home” link

Both FireEye and Splunk allow syslog over TCP as well. There is more overhead, but also more reliable.

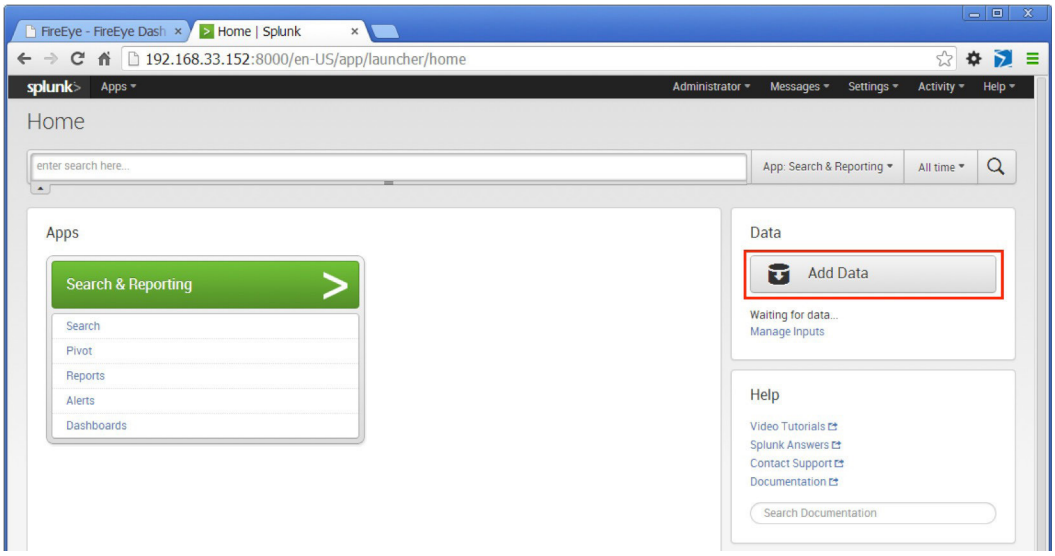


Figure 1: Adding a data connector in Splunk

FireEye Data

Now that Splunk is listening and ready for data, we have to configure FireEye to send syslog data to the connector. The FireEye appliances are very flexible regarding Notification output and support the following formats under syslog:

CEF	Text – Normal	JSON – Normal	XML – Normal
LEEF	Text – Concise	JSON – Concise	XML – Concise
CSV	Text –Extended	JSON –Extended	XML –Extended

For our tutorial, we will use CEF — but it does not mean that it is the best format. It is just one possible option (see the "Parsing Other Formats" section for more details). Complete the following steps to send data to Splunk using CEF:

- Log into the FireEye appliance with an administrator account
- Click Settings
- Click Notifications
- Click rsyslog
- Check the "Event type" check box
- Make sure Rsyslog settings are:
  - Default format: CEF
  - Default delivery: Per event
  - Default send as: Alert

Next to the "Add Rsyslog Server" button, type "Splunk". Then click the "Add Rsyslog Server" button. Enter the IP address of the Splunk server in the "IP Address" field, and click the "Update" button below. Change the protocol dropdown to TCP if you decided to use TCP when setting up the Splunk receiver.

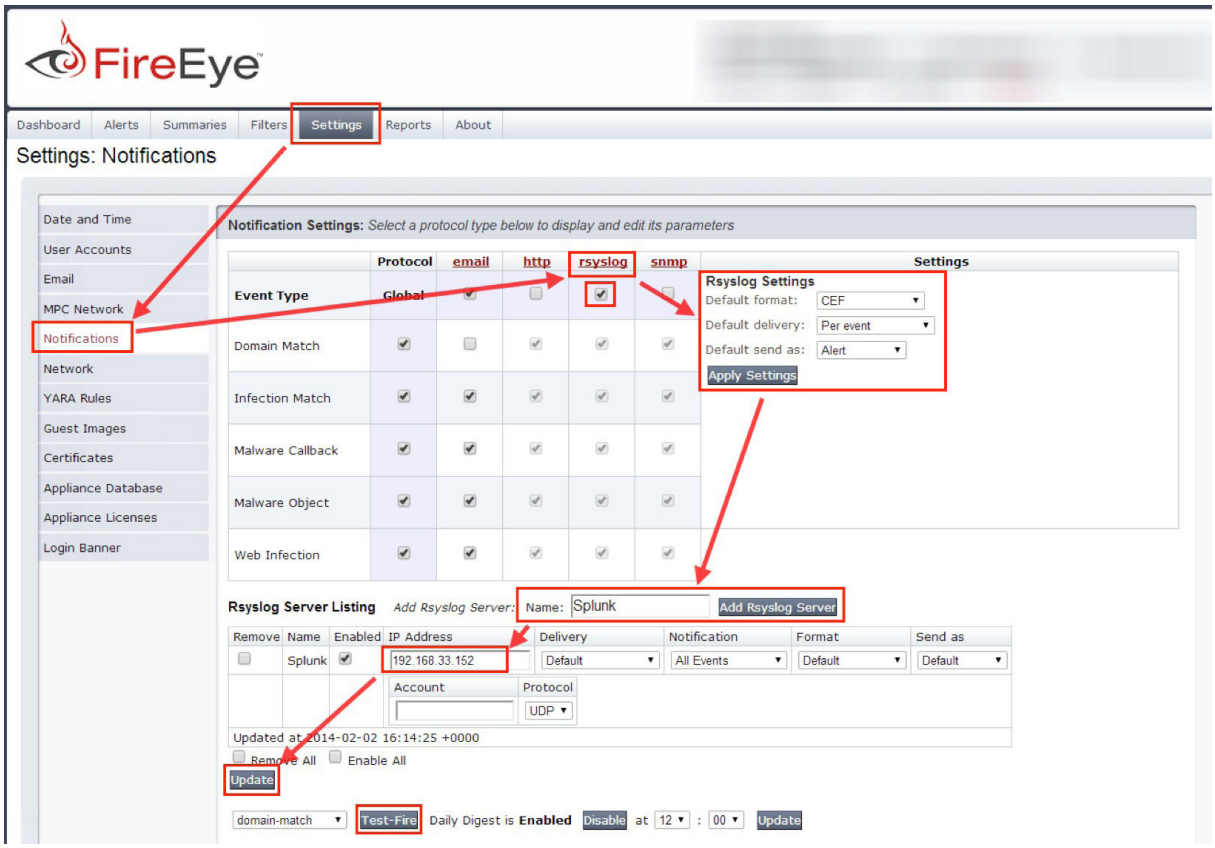


Figure 2: Steps to configure the FireEye appliance to send data to Splunk

Now you can test the sending and receiving of notifications on the same FireEye Notifications page by clicking the “Test-Fire” button at the bottom. Flip back over to the Splunk interface and check out the raw event data.

Examining a Raw Event

Now that the connectors are set up, we can view the raw data.

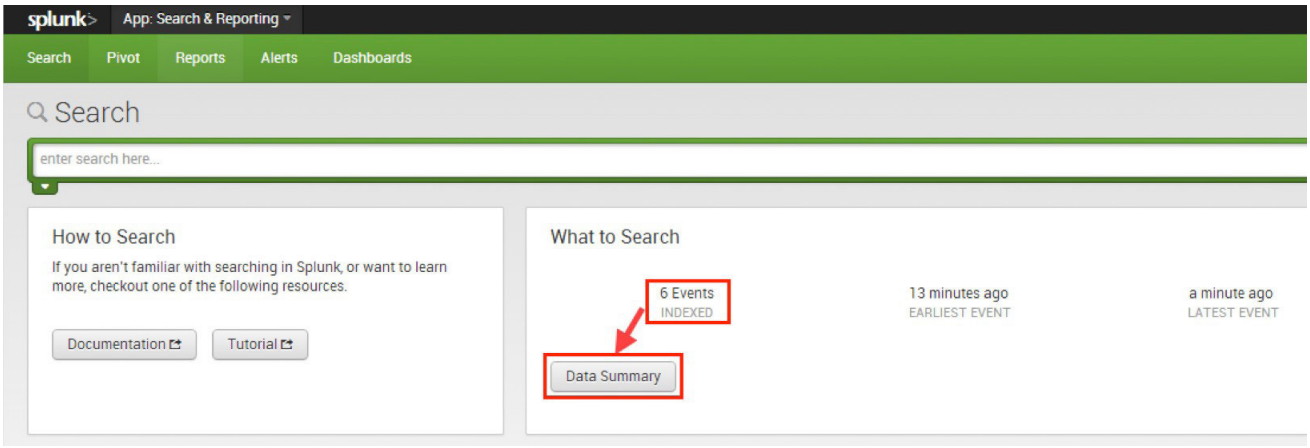


Figure 3: The Splunk dashboard now shows events

After clicking on the Data Summary button, you can see the raw CEF events. They will look something like the following:

```
Feb 2 11:57:59 192.168.33.131 fenotify-2.alert: CEF:0|FireEye
|MPS|6.2.0.74484|WI|web-infection|5|rt=Feb 02 2014 16:57:47 Z
src=169.250.0.1 dpt=20 shost=OC-testing.fe-notify-examples.com
proto=tcp dst=127.0.0.20 dvchost=WebMPS cs3Label=osinfo cs3=FireEye-
TestEvent OS Info filePath=compl_0_2- someurl.x1y2z3.com spt=10
dvc=192.168.33.131 smac=XX:XX:XX:XX:XX:XX cn1Label=vlan cn1=0
externalId=2 cs4Label=link cs4=https:// WebMPS.localdomain/event_
stream/ events_for_bot?inc_id\=2 dproc=IEx123 dmac=XX:XX:XX:XX:XX:XX
cs2Label=anomaly cs2=anomaly-tag datatheft keylogger cs1Label=sname
cs1=FireEye-TestEvent-SIG

dpt = 20 dst = 127.0.0.20 proto = tcp spt = 10 src = 169.250.0.1
```

At first when looking at this data, it looks a bit confusing. Fortunately, the Splunk dashboard highlights and separates the data so it is a little easier to view and understand.

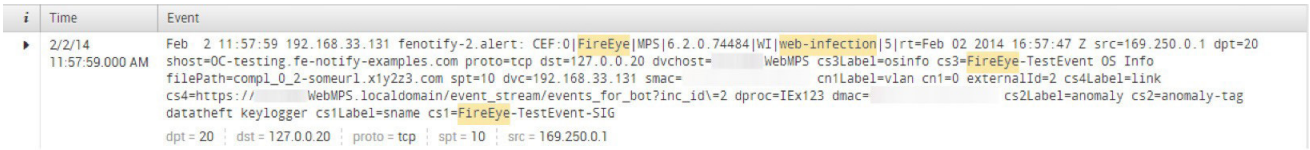


Figure 4: Search term and mouse over highlighting

How to Replicate a FireEye Dashboard

Now that we have data in Splunk, we need to figure out how to carve it up. Our example below will use alert data from a FireEye Web MPS (NX platform).

Simple Searches

Splunk's search capability is quite powerful. Searching can be as simple as you like — just using a keyword or two — or it can be complex, using pipes, regular expressions, and built-in functions.

Try using the search term FireEye in Splunk. It should return FireEye events. This is great, but be careful using such a simple search because you may get unintended results of other logs that contain the word "FireEye".

Instead, try using: **`CEF:0\|FireEye`**

Remember that the pipe is a reserved character to Splunk so we have to escape it using a backslash (\). This will look for "**`CEF:0|FireEye`**" in the packet, which ensures that the search result will at least be a CEF packet from a FireEye device.

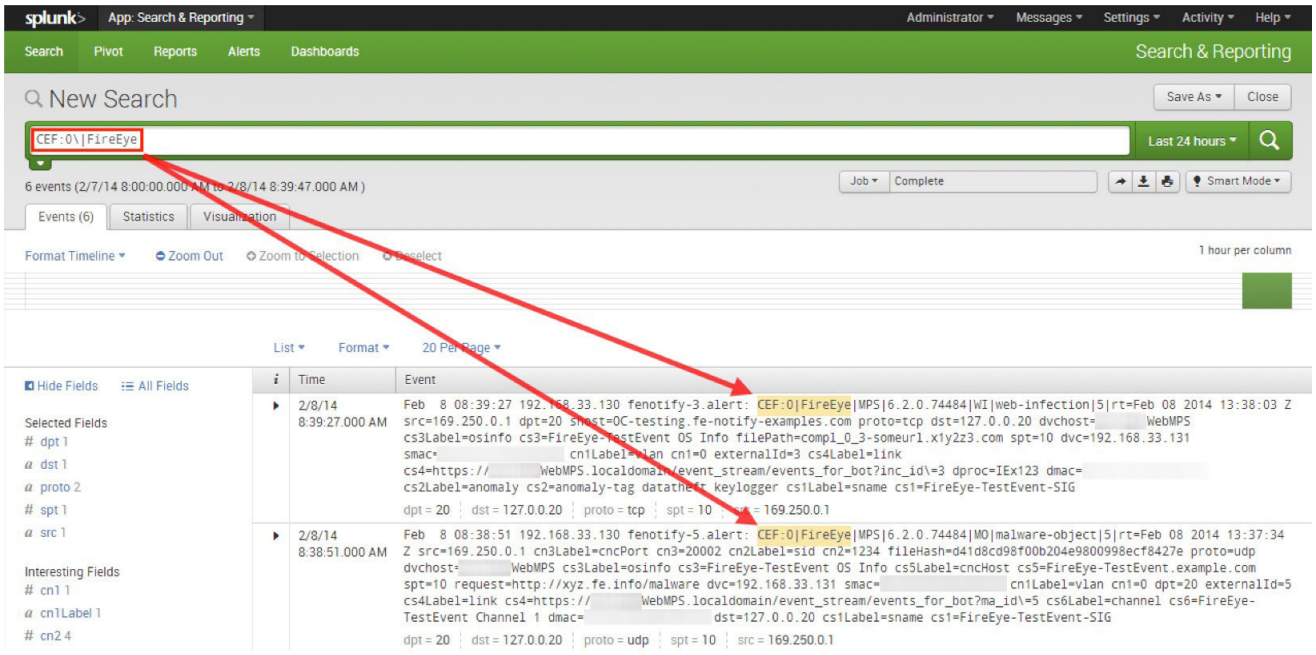


Figure 5: Using the more specific CEF search, we are ensuring that we receive the specific packets of interest.

Piping Search Results

Now that we know how to find the relevant FireEye CEF packets, we only want to select the relevant columns—not all of them. For this, we will use a pipe in the Splunk search bar.

A FireEye wMPS Alert Dashboard contains the following columns:
Type, ID, File Type (FT), Malware (name), Severity, Time (UTC), Source IP, Target IP, URL/MD5, Location

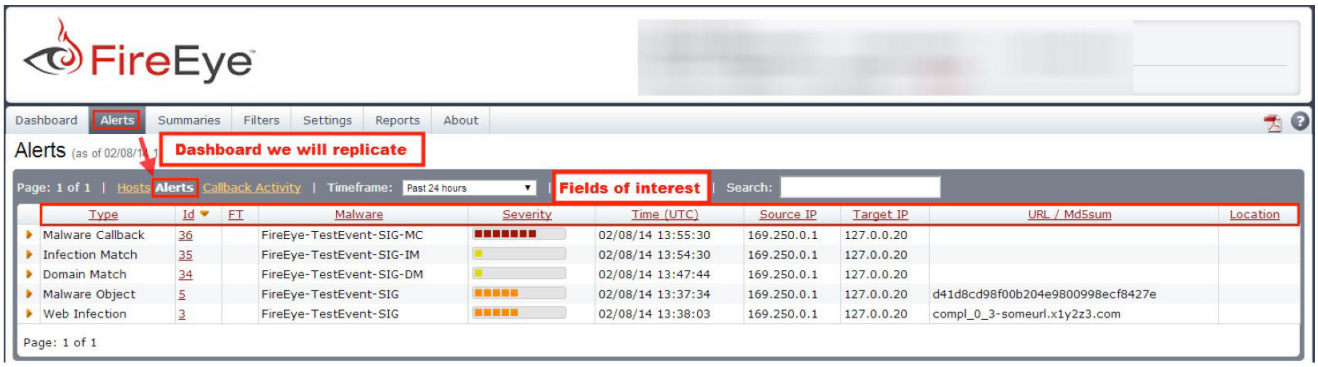


Figure 6: FireEye event fields of interest

Not all of these fields are passed in the CEF packet though. So we should first create a map of Web MPS dashboard fields to Splunk parsed fields. We have done so in the table below:

FireEye field	Splunk field
Type	**Not a parsed field**
ID	externalId
File Type	**Not sent**
Malware	cs1
Severity	**Not parsed without some help**
Time (UTC)	**Not a parsed field**
Source IP	src
Target IP	dst
MD5	fileHash
URL (malware callback, domain match, malware object)	cs5
URL (web infection)	filePath
Location	**Not sent**

So far, we cannot do much about information that is not sent in the CEF packet because the data does not exist in Splunk. The information that is present but not parsed as a Splunk field can be extracted using regular expressions, which we will talk about in the next section. However, all of the remaining information that is parsed by Splunk is easily accessible and displayed by piping the field name to the table command as shown in the example below:

Ex: `CEF:0\|FireEye | table externalId,cs1,src,dst,fileHash,filePath,cs5`

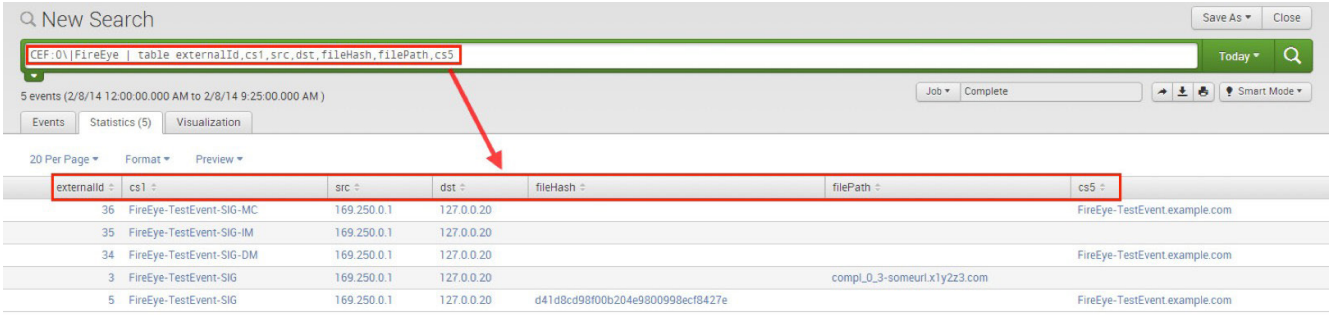


Figure 7: A simple pipe to table will format the fields in a similar fashion to the FireEye dashboard

Not a bad start for accessing fields that are already parsed by Splunk. We are only missing Type, FT, Severity, Time, and Location. Severity requires regex parsing and a lookup table which is beyond the scope of this article. For now, let's look at parsing the Type and Time fields.

Using Regular Expressions

Since some of the data we are interested in is not parsed as a field (event type and event time), we must use regular expressions to extract these fields. If you are a regex ninja, feel free to use your powers for good and get the data you need. If not, no big deal — Splunk includes an interactive field extractor feature that will build the regex needed to extract data of interest. To use this field extractor, perform the following:

- Search for the event you are interested in: CEF:0\|FireEye
- Click the black arrow next to an event to drop down the details
- Event Actions -> Extract fields
- Highlight one of the event types and copy and paste it into the example value box
- Click generate and verify the accuracy of the regex by looking at the highlighted values on the right

We will do this for both the event type and event time.

Get the Event Type

Ex: Domain Match

Interactive field extractor result: `(?i)\|.*?\|(?P<FIELDNAME>[a-z]+\-[a-z]+)(?=\|)`

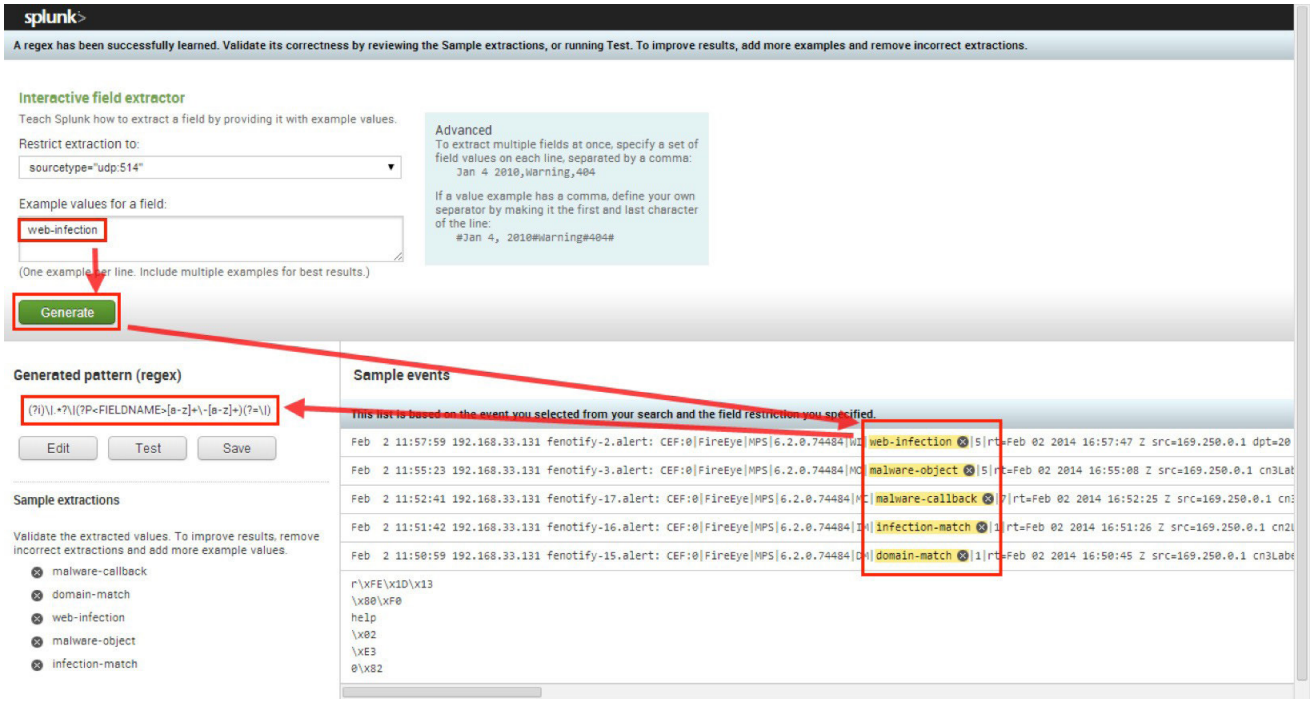


Figure 8: The field extractor created an accurate regex to obtain the event type field

Now we can plug the regex into the rex command and surround it with quotes to make it functional. Ex: `rex "(?i)\\|.*?\\|(?P<FIELDNAME>[a-z]+\\-[a-z]+)(?=\\|)"`

Note that we can change FIELDNAME to whatever we wish, and we can access the field later by using the unique name we provide.

Get the Event Time

There are two times in the CEF packets. In this case we are interested in the FireEye event time, not the time that Splunk received the event (Splunk time is accessible by referencing the _time field). When we use the Interactive field extractor for the FireEye time, it yields the following regex:

```
"(?i)\|rt=(?P<FIELDNAME>.+?)\s+\w+="
```

Now that we have the Event type and the Event time, let's put it all together. Since we feel the 5-tuple data is useful for event correlation, we will add in protocol, source port, and destination port as a bonus to our new dashboard:

```
CEF:0\|FireEye | rex "(?i)\\|.*?\\|(?P<Type>[a-z]+\\-[a-z]+)(?=\\|)" |
rex
"(?i)\|rt=(?P<Time>.+?)\s+\w+=" | table Type,externalId,cs1,Time,proto
,src,spt,dst, dpt,fileHash,filePath,cs5
```

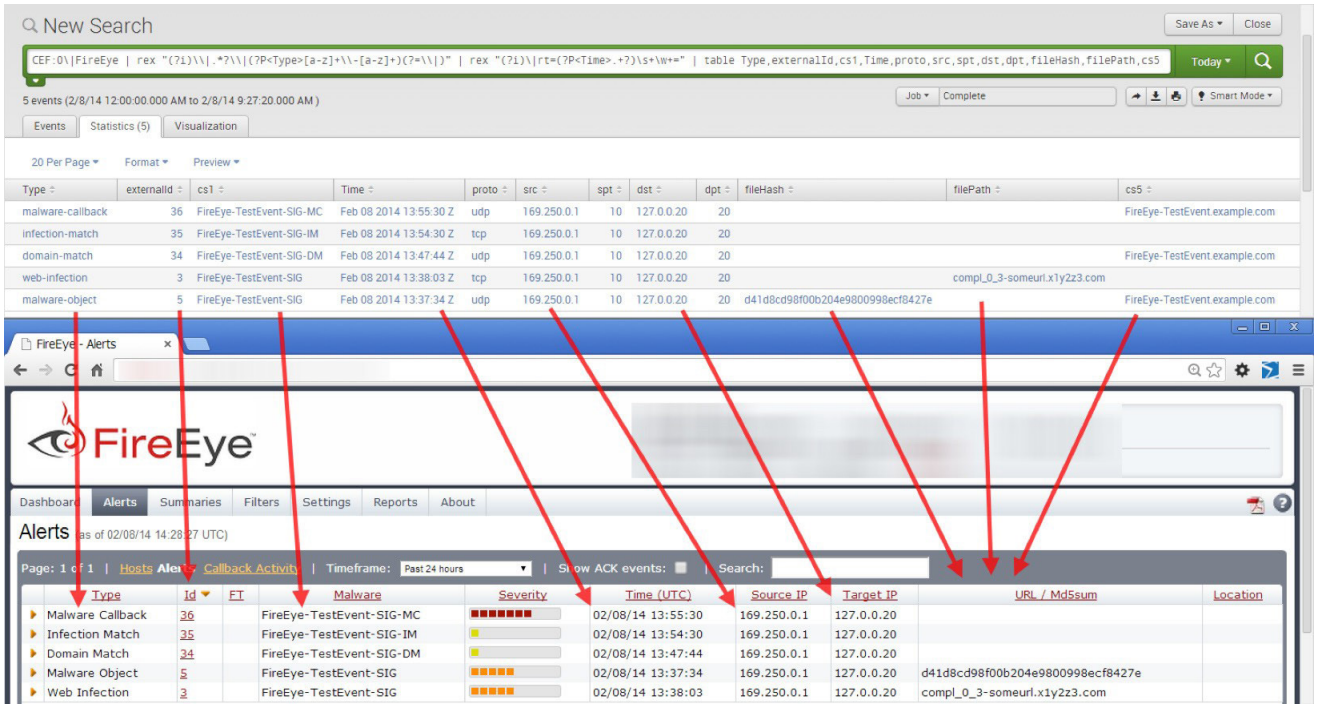


Figure 9: Our dashboard is closer to completion as we add the type and time fields

Using Conditionals

If you did not notice, the FireEye appliance dashboard combines the field that contains the MD5 hash and URL. The data that is populated in that field depends on the event type. If it is a "Malware Object" event type, the "URL/Md5sum" field contains an MD5 hash. If it is another event, it contains a URL. The problem that we have is that not all events use the same field for a URL. A web infection event uses the filePath field. Malware Callback, Domain Match, Malware Object use the cs5 field (which is not displayed on the FireEye dashboard. We can either have 3 separate fields on our dashboard (fileHash, filePath, and cs5) or we can find a way to use conditional statements in Splunk to combine the fields. Let's investigate combining the fields.

If statement

If this were a case in which we had only two fields to pick between, we could use an if statement and achieve our aggregation objective with the following Splunk syntax:

```
eval   varname=if(condition,ActionForTrue,ActionForFalse)
```

Knowing this, our search now utilizes the following conditional:

```
<previous search> | eval UrlHash=if(Type==”malware-
object”,fileHash,filePath)
```

Explanation: If Type is equal to malware-object, then UrlHash will be the fileHash field, otherwise it will be the filePath field.

Combining our previous search and our conditional, our search now looks like the following:

```
CEF:0\|FireEye  | rex  "(?i)\\|.*?\\|(?P<Type>[a-z]+\\-
[a-z]+)(?=\\|)"  | rex "(?i)\|rt=(?P<Time>.+?)\s+\w+=" | eval
UrlHash=if(Type=="malware-
object",fileHash,filePath) | table Type,externalId,cs1,Time,proto,src,
spt,dst,dpt,Url Hash,cs5
```

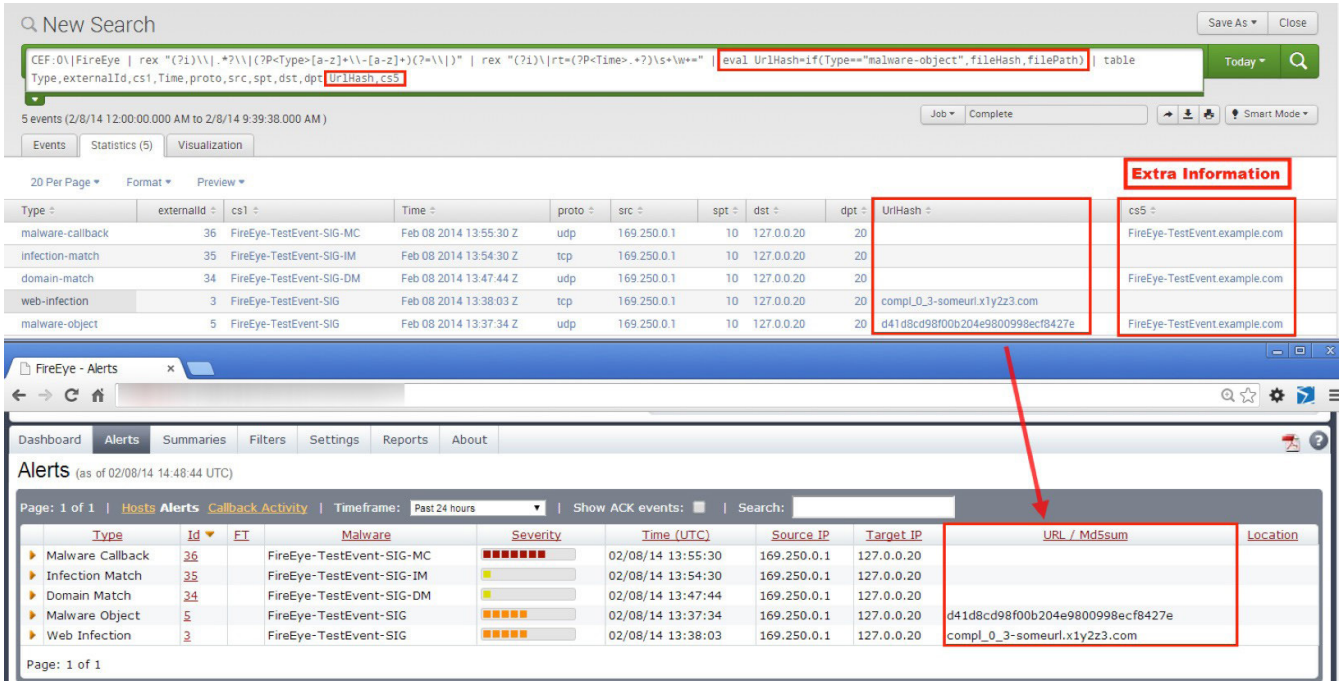


Figure 10: The fileHash and filePath fields were combined using an if statement; however, the cs5 field is not displayed on the original FireEye dashboard

Take note of the following:

- The fileHash and cs5 fields were replaced by UrlHash — which will contain the correct value relative to the event type.
- Even though Malware Object contained both an MD5 hash and a URL, it was overridden by the MD5 hash which is desirable in this case.
- We still have to deal with the cs5 field if we want to include this additional information

Case statement

Since we find the cs5 field useful, in our custom dashboard, we are choosing to combine the cs5 field into the URL/MD5sum field as well. This is possible by using a Splunk case statement:

```
eval varname=case(condition,ActionForTrue, condition,ActionForTrue,
condition,ActionForTrue)
```

Knowing this, our search now utilizes the following conditional:

```
<previous search> | eval UrlHash=case(Type=="malware-object",fileHash,
Type=="web-in fection",filePath,Type=="malware-callback"  OR
Type=="domain-match",cs5)
```

Explanation: If Type is equal to malware-object, then UrlHash will be the fileHash field; If Type is equal to web-infection, then UrlHash will be the filePath field; If Type is equal to malware-callback or domain- match then UrlHash will be the cs5 field.

Combining our previous search and our conditional, our search now looks like the following:

```
CEF:0\|FireEye | rex "(?i)\\|.*?\\|(?P<Type>[a-z]+\\-[a-z]+)
(?=\\|)" | rex "(?i)\|rt=(?P<Time>.+?)\s+\w+=" | eval UrlHash=case
(Type=="malware-object",fileHash, Type=="web-infection", filePath,
Type=="malware-callback" OR Type=="domain- match",cs5) | table Type,ex
ternalId,cs1,Time,proto,src,spt,dst,dpt,UrlHash
```

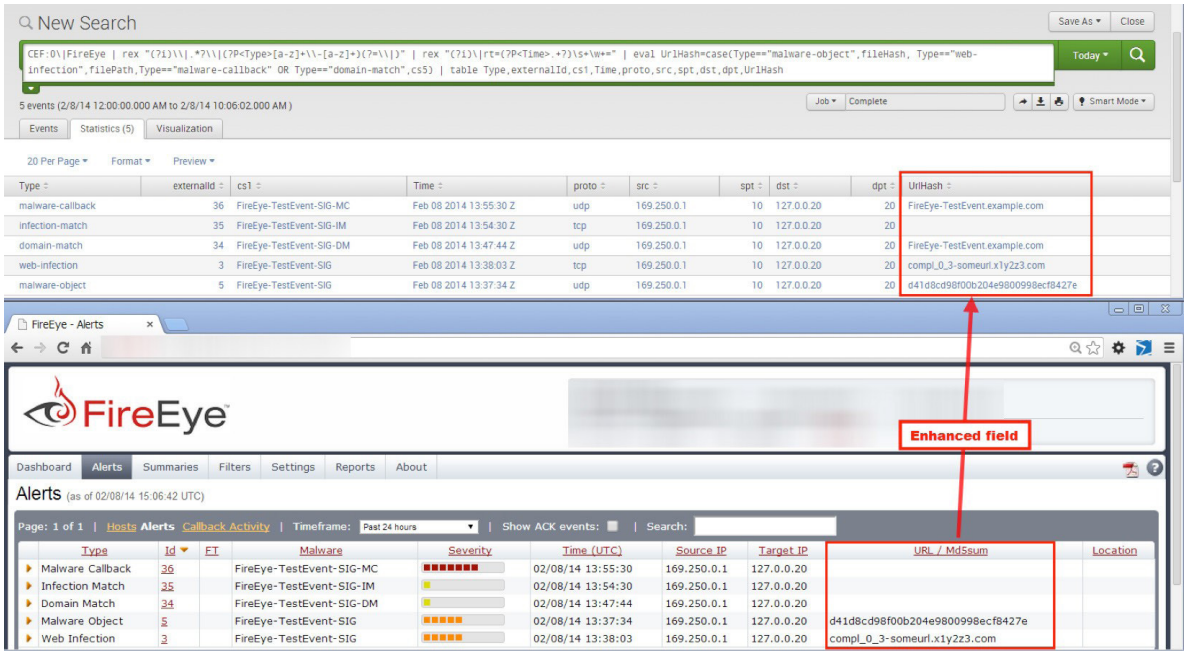


Figure 11: The latest dashboard that has fileHash, filePath, and cs5 combined.

Sorting Searches

This looks great, except it appears that the FireEye dashboard defaults to sorting by the Event ID. No problem, we can add that as well by piping the data to the sort command:

```
CEF:0\|FireEye | rex "(?i)\\|.*?\\|(?P<Type>[a-z]+\\-[a-z]+)(?=\\|)" |
rex
"(?i)\|rt=(?P<Time>.+?)\s+\w+=" | eval UrlHash=case (Type=="malware-
object",fileHash, Type=="web-infection", filePath, Type=="malware-
callback" OR Type=="domain-
match",cs5) | table Type,externalId,cs1,Time,proto,src,spt,dst,
dpt,UrlHash | sort
-externalId
```

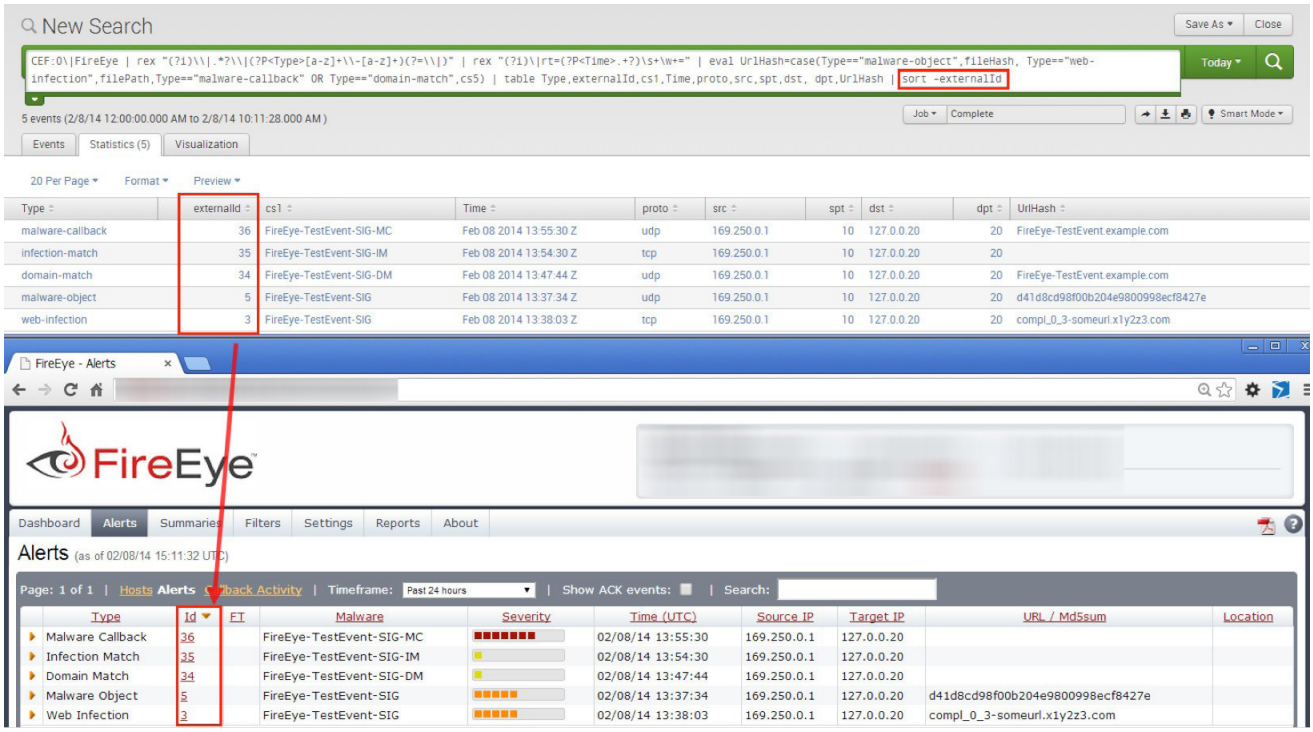


Figure 12: Dashboards default to the same default sort (by Id)

Multiple LMS Note: If you have multiple FireEye LMS appliances, their event IDs may be drastically different depending on when the appliance was installed and how much activity it sees — thus it is not advisable to sort by event ID (as the FireEye dashboard does). Instead, sort by time (as long as it is normalized—ex: All appliances use UTC). Sorting by time removes the event ID order discrepancy.

Multiple LMS sort by Time

```
CEF:0\|FireEye | rex "(?i)\\|.*?\\|(?P<Type>[a-z]+\\-[a-z]+)(?=\\|)" |
rex
"(?i)\|rt=(?P<Time>.+?)\s+\w+=" | eval UrlHash=case (Type=="malware-
object",fileHash, Type=="web-infection", filePath, Type=="malware-
callback" OR Type=="domain-
match",cs5) | table Type,externalId,cs1,Time,proto,src,spt,dst,dpt,Url
Hash | sort -Time
```

Now we are really close to having a great dashboard, but the column headings could be renamed to have a cleaner look.

Renaming the Columns

Renaming column headers is very easy in Splunk. Just pipe all of your data to the rename command and then rename the column headers as you would do in a SQL database using “as”.

Ex: **<search> | rename externalId as “ID”**

Combining our previous work with the rename command yields the following:

```
CEF:0\|FireEye | rex "(?i)\\|.*?\\|(?P<Type>[a-z]+\\-[a-z]+)(?=\\|)" |
rex "(?i)\|rt=(?P<Time>.+?)\s+\w
+=" | eval UrlHash=case (Type=="malware-object",fileHash, Type=="web-
infection", filePath, Type=="malware-callback" OR Type=="domain-
match",cs5) | table Type,externalId,cs1,Time,proto,src,
spt,dst,dpt,UrlHash | sort -externalId | rename externalId as "ID",cs1
as "Malware",proto as "Protocol",src as "Source IP",spt as "Source
Port",dst as "Target IP",dpt as "Target Port",UrlHash as "URL/Md5sum"
```

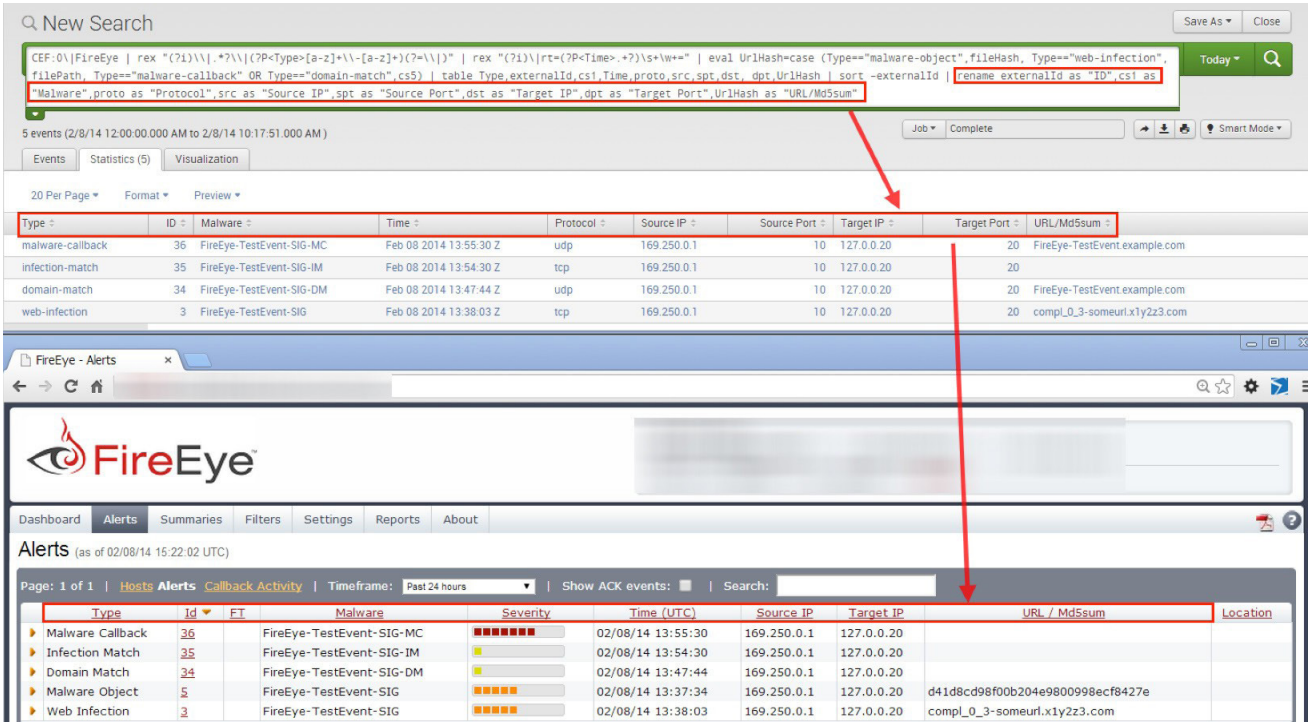


Figure 13: Fields are renamed to match the FireEye dashboard

This is just the dashboard we were looking to create. However, no one wants to continue to type or even copy and paste that monstrosity into the search bar. Fortunately, Splunk allows us to save a search as a Dashboard Panel.

Save As Dashboard Panel

Now that we have the query that we want, we can save this as a Dashboard Panel for quick access to the data. After you make sure the search is the way you want it (our latest search string), click the “Save As” drop-down and select “Dashboard Panel”.

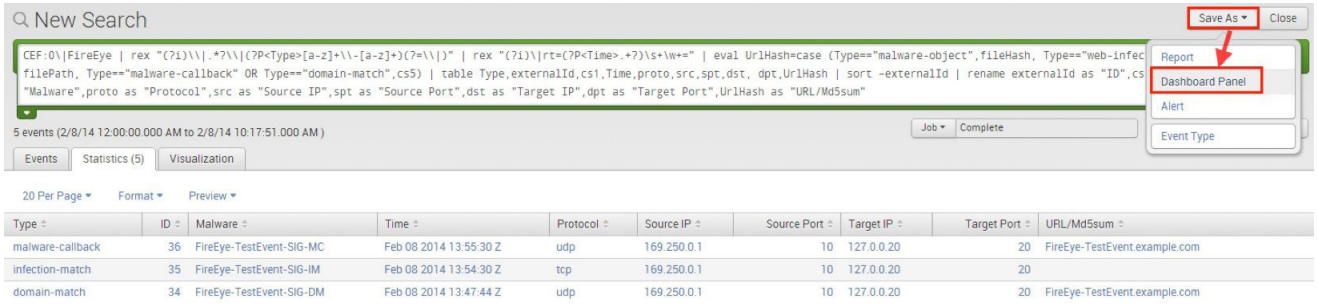
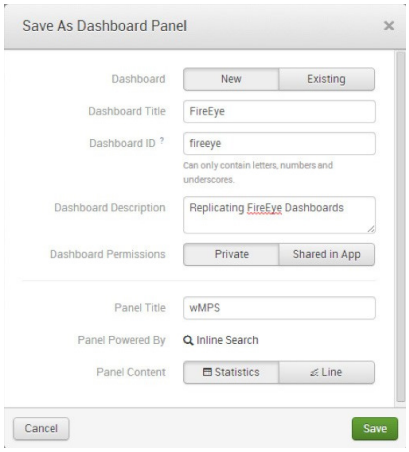


Figure 14: Saving a search to a Dashboard Panel

Since this is a new Dashboard for us, we will use the following settings:

- Dashboard: New
- Dashboard Title: FireEye
- Description: Replicating FireEye Dashboards
- Permission: Shared in App
- Panel Title: wMPS
- Panel Content: Statistics



When content with the data, click the “Save” button and then “View Dashboards”. Now, anytime we want to view the FireEye wMPS search that we created, we just click the “Dashboards” link at the top.

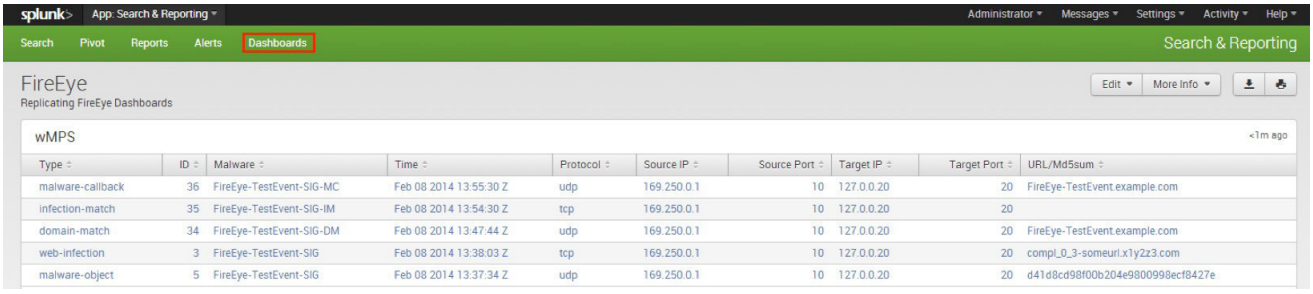


Figure 15: Saved our previous search as a Splunk Dashboard.

Time frames

What is a dashboard without the ability to adjust time frames? If you noticed, in the screenshot above, by default we are lacking a time frame selector on our dashboard. So, let's add one. Click the "Edit" drop down. Click "Edit Panels". Click "Add Time Range Picker". Select Last 24 Hours. Click the "Done" button.

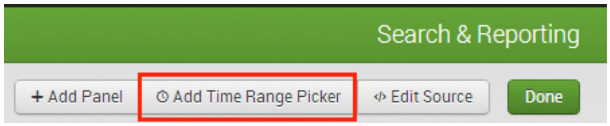


Figure 16: Adding a Time Range Picker to our dashboard

Pro-tip: If for some reason, your time frame button is not working as expected (ours did not by default), you have overridden the time picker in your search. Here is how to fix it:

Go to your dashboard. Click the "Edit" drop down. Click "Edit Panels". Click the magnifying glass drop down, and select "Edit Search String". In the "Time Range Scope" section, select the "Dashboard" button and click save. Now the "Time Range Picker" button will function as expected.

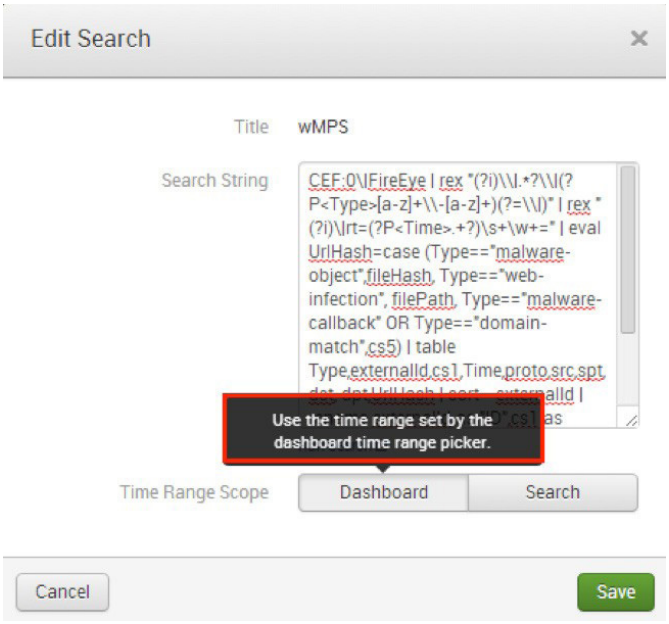


Figure 17: Setting the Time Range Scope to use the dashboard instead of the search string time frame

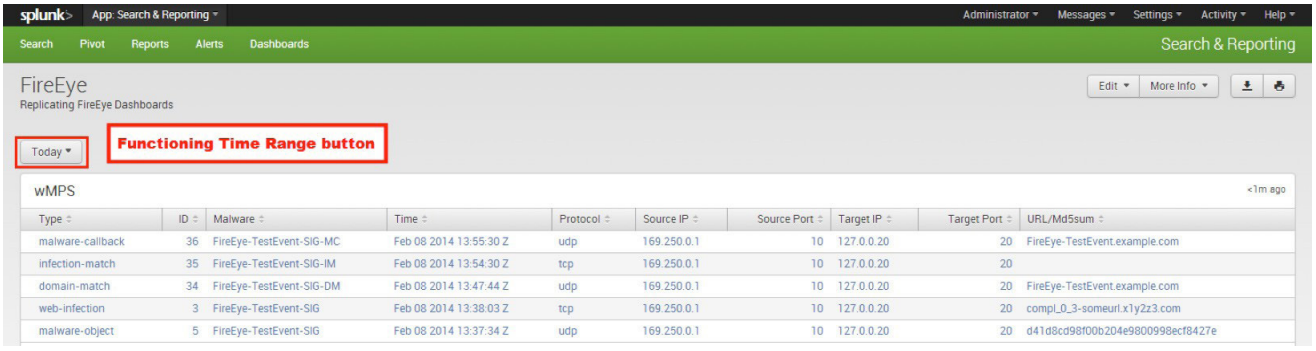


Figure 18: Dashboard with a functioning Time Range Picker button

Parsing Other Formats

We mentioned earlier that FireEye is flexible in its notification output formats because it supports the following:

CEF	Text – Normal	JSON – Normal	XML – Normal
LEEF	Text – Concise	JSON – Concise	XML – Concise
CSV	Text –Extended	JSON –Extended	XML –Extended

After experimenting with the formats, it appears that CEF, LEEF, and CSV are all viable formats to send to Splunk (via syslog) because Splunk sees an event as one packet (not many packets). As a personal choice, it looks like Splunk parses CEF more easily than LEEF. However, CSV exposes more fields than both CEF and LEEF and appears to be easy to parse — thus it may be a better choice in formats (without moving to HTTP POST of XML or JSON data).

Since we were just parsing CEF, we have to change the FireEye syslog format to CSV now. This is a pretty easy change in the FireEye device by going to “Settings” and then “Notifications”. Then click “syslog”. FireEye provides the ability to change the default syslog format for all of the syslog servers or the default can be overridden per server. Here, we will override the default setting for this Splunk server only. Make sure you remember to click the “Update” button when finished. Finally, fire off a test event so you have CSV data to look at in Splunk.

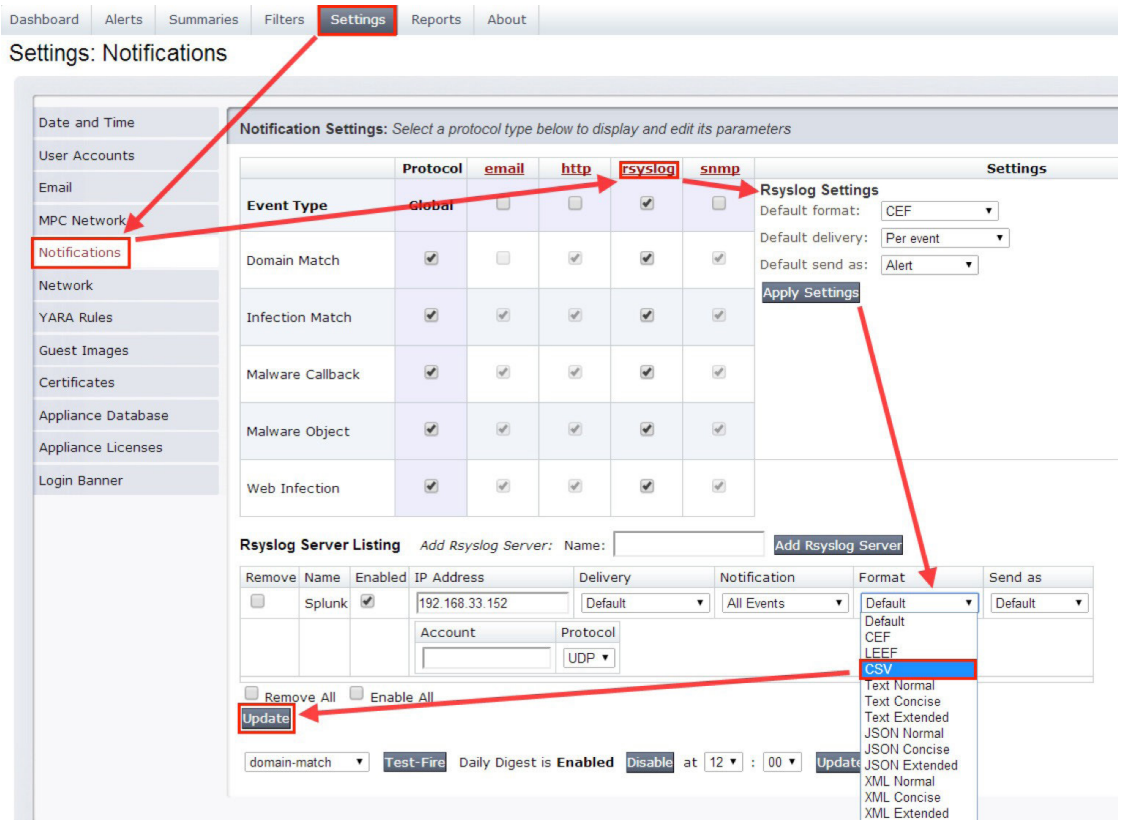


Figure 19: Changing the FireEye appliance to use CSV for Splunk event notification.

We can inspect a CSV field with the following search: CSV:0:FireEye

A typical CSV event looks like the following:

```
Feb 8 12:27:12 192.168.33.131 fenotify-40.alert: CSV:0:FireEye:Web
MPS:6.2.0.74484:DM:domain-match,osinfo=,sev=minr,malware_type=,
ale rtid=40,app=,spt=10,locations=,smac=XX:XX:XX:XX:XX:XX,header=,cnch
ost=FireEye-TestEvent.example.com,alertType=domain-match,shost=DM-
testing.fe-notify- examples.com,dst=127.0.0.20,original_name=,      ap
plication=,sid=5432;30,malware- note=,objurl=,mwurl=,profile=,dmac=
XX:XX:XX:XX:XX:XX,product=Web       MPS,sname=FireEye-TestEvent-SIG-
DM;FireEye-TestEvent- SIG-IM,fileHash=,dvchost=WebMPS, occurred=2014-
02-08T17:26:4 0Z,release=6.2.0.74484, link=https://WebMPS.localdomain/
event_stream/events_for_bot? ev_id=40,cncport=20002,src=169.250.0.1,d
pt=20,anomaly=,dv c=192.168.33.131,channel=FireEye-TestEvent Channel
1,action=notified,os=, stype=blacklist;bot- command,
```

The CSV fields map as follows:

FireEye field	Splunk field
Type	alertType (CEF required regular expression)
ID	alertid
File Type	**Not sent** (Not sent in CEF either)
Malware	sname
Severity	sev (More difficult to parse using CEF)
Time (UTC)	occurred (CEF required regular expression)
Source IP	src
Target IP	dst
MD5	fileHash
URL (malware callback, domain match, malware object)	cnchost
URL (web infection)	objurl
Location	locations?

Since Splunk can parse the Type and Time fields from the FireEye CSV output, we no longer need the regular expressions. However, we will still need to use a case statement to combine the MD5/URL field. Using the same process that we followed above, the following Splunk search should yield a similar result to what we had before:

```
CSV:0:FireEye | eval UrlHash=case (alertType=="malware-
object",fileHash, alertType=="web-infection", objurl,
alertType=="malware-callback" OR alertType=="domain-match",cnchost)|
sort -alertid | table alertType,alertid,sname,sev,occurred,src,spt,
dst,dpt,UrlHash,locations | rename alertType as "Type", alertid as
"ID",sname as "Malware",sev as "Severity",occurred as "Occurred",src
as "Source IP",spt as "Source Port",dst as "Target IP",dpt as
"Target Port",UrlHash as "URL/Md5sum",locations as "Location"
```

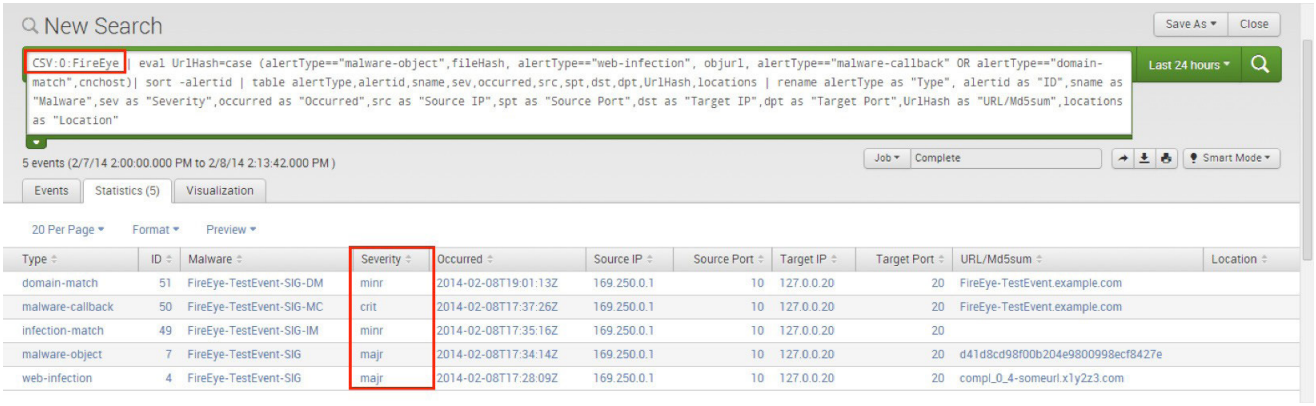


Figure 20: Parsing a CSV event

Note: When parsing CSV, you get the severity field, but not the protocol field.

Figure 21: Severity ratings from the FireEye dashboard

Since we are satisfied with this dashboard, we can take the same steps to save this search as a dashboard panel. However, we will save this panel to our current FireEye dashboard, so we can provide a comparison of the fields and layout.

Quick Differences

There are a few differences between these two formats (CEF and CSV). Most notably:

- Available fields
  - CEF provides the protocol field
  - CSV provides the severity field and location fields
- Time format
  - CEF is more human readable: Feb 08 2014 13:55:30 Z
  - CSV is more machine friendly: 2014-02-08T19:01:13Z
- Speed
  - CEF is slower due to using two regular expressions

Bottom line: Pick the format that makes the most sense for your organization.

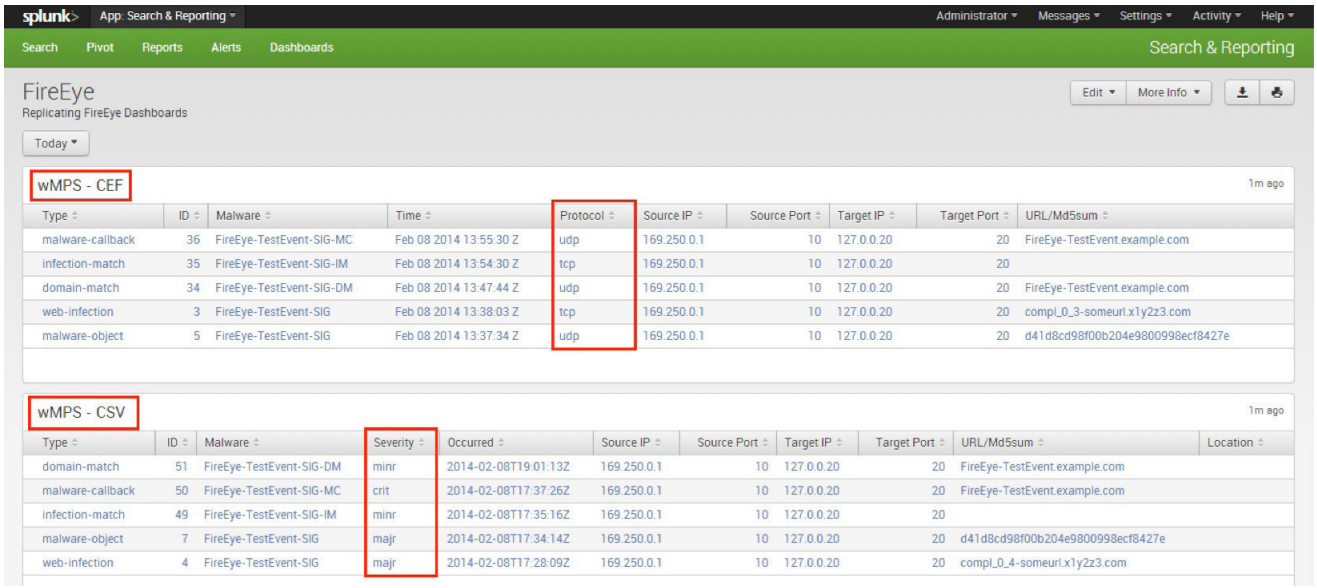


Figure 22: Single FireEye dashboard with two panels showing the differences in exposed data between CEF and CSV events

Sample FireEye Dashboards

Now that we have walked you through the process of creating a FireEye dashboard in Splunk, we will provide some dashboards that we put together.

wMPS (NX)

Our step by step instructions already showed you how to create the Splunk searches for a wMPS dashboard using CEF and CSV, but here they are again for convenience:

Alerts -> Alerts

CEF:

```
CEF:0\|FireEye | rex "(?i)\\|.*?\\|(?P<Type>[a-z]+\\-[a-z]+)(?=\\|)" |
rex "(?i)\|rt=(?P<Time>.+?)\s+\w+=" | eval UrlHash=case
(Type=="malware-object",fileHash, Type=="web-infection", filePath,
Type=="malware-callback" OR Type=="domain-match",cs5) | table Type
,externalId,cs1,Time,proto,src,spt,dst, dpt,UrlHash | sort -Time |
rename externalId as "ID",cs1 as "Malware",proto as "Protocol",src as
"Source IP",spt as "Source Port",dst as "Target IP",dpt as "Target
Port",UrlHash as "URL/Md5sum"
```

CSV:

```
CSV:0:FireEye | eval UrlHash=case (alertType=="malware-
object",fileHash, alertType=="web-infection", objurl,
alertType=="malware-callback" OR alertType=="domain-match",cnchost)|
sort -occurred | table alertType,alertid,sname,sev,occurred,src,spt,
dst,dpt,UrlHash,locations | rename alertType as "Type", alertid as
"ID",sname as "Malware",sev as "Severity",occurred as "Occurred",src
as "Source IP",spt as "Source Port",dst as "Target IP",dpt as "Target
Port",UrlHash as "URL/Md5sum",locations as "Location"
```

Alerts -> Callback Activity

CSV:

```
CSV:0:FireEye alertType="domain-match" | stats count(cnchost) as
"Events", distinct_count(src) as "Hosts", max(occurred) as "Last Seen"
by cnchost | table cnchost, locations, Events, Hosts, "Last Seen" |
rename cnchost as "C&C Server", locations as "Locations" | sort -"Last
Seen"
```

Panels to Enhance Visibility

Now that you have a Splunk dashboard that replicates the FireEye Dashboard, let's go further by adding more panels that will enhance the network defender's visibility into attacks. Splunk has many visualization features that allow users to build charts and graphs for trending and analytics.

wMPS (NX)

We can take some of the fields that we exposed in the FireEye dashboard above and create pie charts to summarize key data points. This can help gauge if an organization is experiencing an increase or lull in attacks, as well as identifying trends in malware, and Command and Control (C2) ports. Each chart below is optional and may depend on the organization or security team's preferences.

Severity Pie Chart:

```
CSV:0:FireEye | chart count by sev Select pie chart
```

After typing in the search above, select the visualization tab, then pie chart. Click the "Save As" button, select "Dashboard Panel", click the "Existing" Button. Select the FireEye Dashboard and provide a title of "Severity".

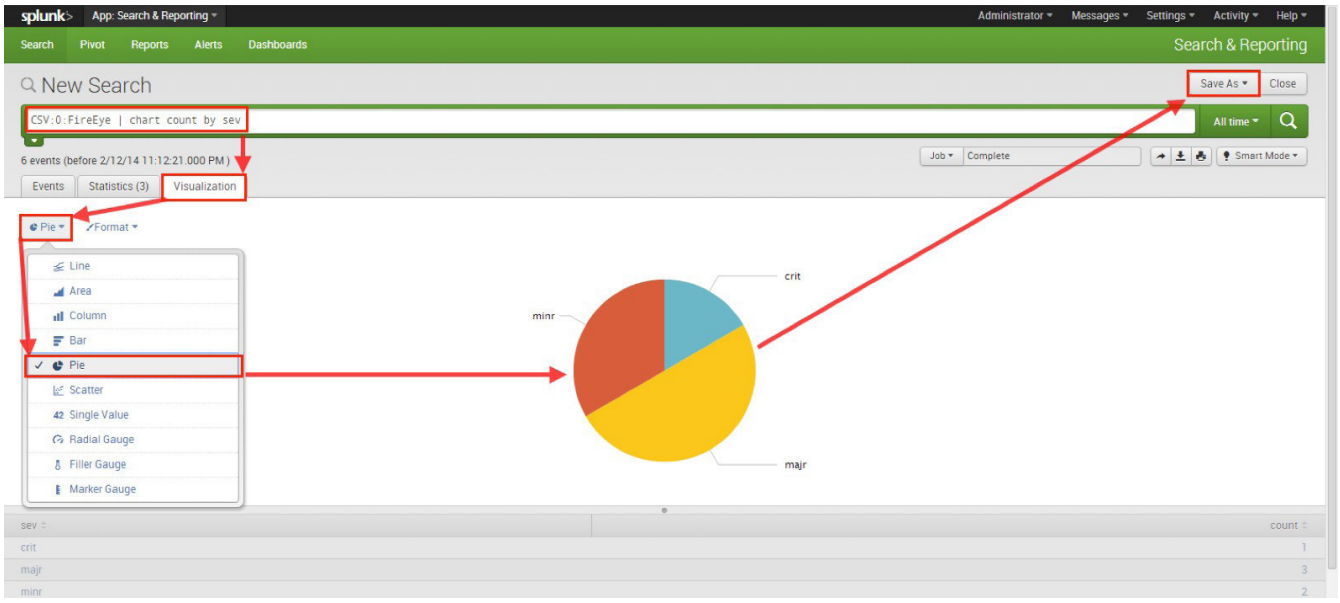


Figure 23: The process of saving a severity pie chart to our FireEye dashboard

Malware pie chart:

```
CSV:0:FireEye | chart count by sname Select pie chart
```

Top 20 most active source IPs:

```
CSV:0:FireEye | top limit=20 src Select pie chart
```

Top 20 most active target IPs:

```
CSV:0:FireEye | top limit=20 dst Select pie chart
```

Top 20 most active destination ports:

```
CSV:0:FireEye | top limit=20 dpt Select pie chart
```

Most active sensor:

```
CSV:0:FireEye | chart count by dvchost Select pie chart
```

After adding the desired panels, feel free to move the panels around and combine them on one line by clicking the "Edit" button, then "Edit Panels". Experiment by clicking and dragging the top bar of the panel to rearrange the view. A screenshot below of our demo setup is just one possible layout.

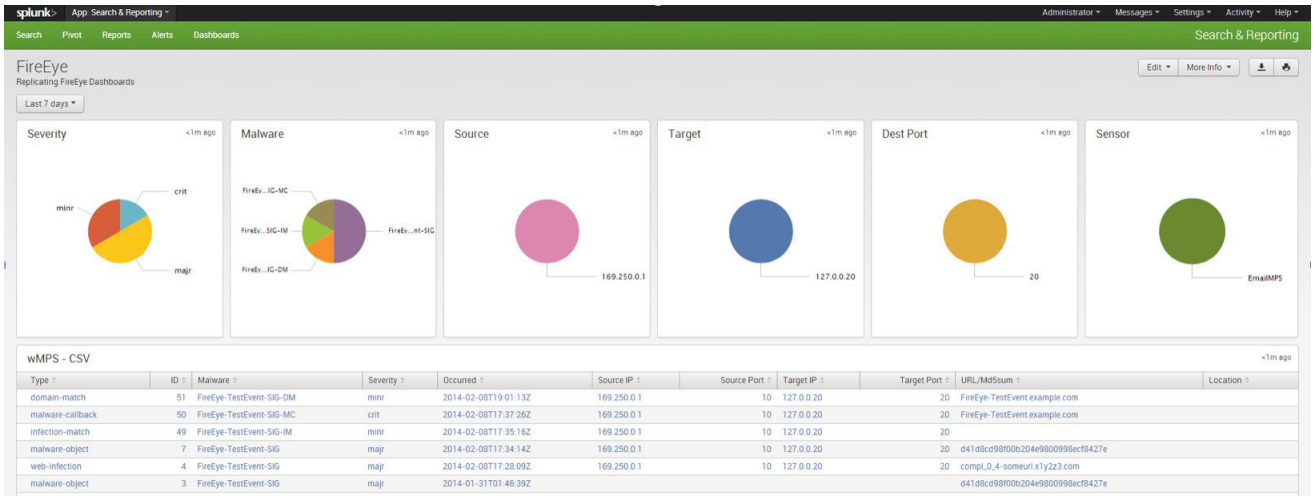
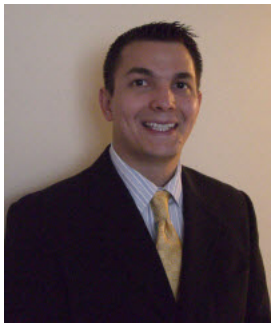


Figure 24: The FireEye Dashboard designed to enhance visibility

Conclusion

Using a FireEye device, a free demo of Splunk, and Google, we were able to investigate the different syslog formats and replicate FireEye Dashboards in Splunk. In writing this guide, we have discovered that there are many ways to tackle this problem because FireEye has robust event notification and Splunk is flexible when ingesting these events. We are sharing this information in hopes that you see even a fraction of what is possible. Who knows, it may save you a little time as well. We would love to hear your feedback, sample FireEye dashboards, and any pro-tips you have for consuming and displaying data in Splunk.



About the Author

Tony Lee has more than ten years of professional experience pursuing his passion in all areas of information security. He is currently a Technical Director at Mandiant, a FireEye Company, advancing many of the network penetration testing service lines. His interests of late are kiosk hacking, post exploitation tactics, and malware research. As an avid educator, Tony has instructed thousands of students at many venues worldwide, including government, universities, corporations, and conferences such as Black Hat. He takes every opportunity to share knowledge as a contributing author to Hacking Exposed 7, frequent blogger, and a lead instructor for a series of classes. He holds a Bachelor of Science degree in computer engineering from Virginia Polytechnic Institute and State University and a Master of Science degree in security informatics from The Johns Hopkins University.

Email: Tony.Lee -at- FireEye.com
Linked-in: http://www.linkedin.com/in/tonyleevt

Special Thanks

Dennis Hanzlik
Dan Dumond
Ian Ahl
Dave Pany
Karen Kukoda
Leianne Lamb
Brian Stoner
Gunpreet Singh
Kate Scott

About FireEye

FireEye has invented a purpose-built, virtual machine-based security platform that provides real-time threat protection to enterprises and governments worldwide against the next generation of cyber attacks. These highly sophisticated cyber attacks easily circumvent traditional signature-based defenses, such as next-generation firewalls, IPS, anti-virus, and gateways. The FireEye Threat Prevention Platform provides real-time, dynamic threat protection without the use of signatures to protect an organization across the primary threat vectors and across the different stages of an attack life cycle. The core of the FireEye platform is a virtual execution engine, complemented by dynamic threat intelligence, to identify and block cyber attacks in real time. FireEye has over 2,500 customers across 65 countries, including over 150 of the Fortune 500.

FireEye, Inc. | 1440 McCarthy Blvd. Milpitas, CA 95035 | 408.321.6300 | 877.FIREEYE (347.3393) | info@fireeye.com | www.fireeye.com

FireEye