CAE Tech Talk

National Centers of Academic Excellence

Apache Metron and Apache Spot – big data tools for cybersecurity

Presenter: Dr. Alex Rudniy from Fairleigh Dickinson University

About the Project

- Cybersecurity Workforce Education - CNAP Initiatives
- NSA Grant No. H98230-17-1-0321 conducted at FDU

Full title:

Developing Hands-on Exercises for Secure Embedded System Design & Security Data Analytics for Computing and Engineering Students

– PI Kalyan Mondal, Ph.D.

1. **Secure Embedded Systems**

   Co-PI Ravi Rao, Ph.D.

2. **Advanced Systems Programming**

   Co-PI William Phillips, Ph.D.

3. **Big Data Analytics & Cybersecurity**

   Co-PI Alex Rudniy, Ph.D.

   - Graduate assistant Pooja Surapaneni, Fall 2017
   - Graduate assistant Suhag Raval, Spring 2018

Project Goals and Status

Task	Status
1. Integrate Apache Metron	☑
2. Integrate Apache Spot	Pending
3. Add more nodes to the cluster	☑
4. Load sybersecurity datasets into a big data warehouse	☑
5.1 Design lab assignments	☑
5.2 Use lab assignments in the Cybersecurity course	☑
6.1. Setup a cluster in a public cloud	☑
6.2. Design and evaluate labs	In progress
6.3. Prepare documentation	In progress
7. Dissemination	In progress

Note: Multiple additional unplanned tasks completed.

Apache Hadoop

- Open-source big data ecosystem allowing:
  - Distributed processing of large datasets
  - Cluster scalability from one node to thousands
  - Supports data redundancy
  - Works on premises
    - Physical vs. virtual environments, e.g. VMware
  - Works in the cloud, e.g. AWS or MS Azure
- Hadoop distributions are available from several vendors, e.g. Hortonworks, Cloudera, etc.
  - Comes with a variety of applications: YARN, Ambari, Hive, Spark, Storm, Hbase, Kafka, Oozie…

  + Metron + Spot

Hadoop at FDU

- A five-node hardware cluster is up since Fall 2015
  - Performed maintenance, tune ups, user management, etc.
  - Acquired skills necessary for the current project
  - Taught students new technologies, students got jobs
- A five-node VMware cluster added in Fall 2017
- Used in a big data class for hands-on assignments
- Taught MapReduce paradigm, HDFS, Ambari, Hive, Pig, Hbase, Spark, etc.

Possibilities for academic institutions:

- Cloudera Academic Partnership (free)
- Hortonworks Academic Program (free)

Part 2 Apache Metron

APACHE
METRON

Apache Metron Evolution

- **Metron** evolved from **OpenSOC**

  = Open Security Operations Center

  = big data security analytics framework for consumption and monitoring **network traffic** and **machine exhaust data** (log files) of a data center.

  - Works on the Hadoop platform
  - Uses Kafka, Storm, and Elasticsearch
  - Supported features:
    - Unstructured data and streaming data ingestion
    - Interactive query, real-time search, scalable compute
    - Real-time alerts, anomaly detection, data correlation
    - Rules and reports, predictive modeling via UI and applications

Apache Metron Evolution

September 2013	• OpenSOC First Prototype
December 2013	• Hortonworks joins the OpenSOC project
2014- April 2015	• OpenSOC platform development finished, first beta test conducted at a customer site
June 2015	• OpenSOC became a community edition
July 2015	• Cisco stops support for OpenSOC
October 2015	• James Sirota, Cisco Chief Data Scientist and Lead of OpenSOC joins Hortonworks
December 2015	• Metron accepted into Apache Incubator
April 2016	• First release of Metron 0.1
April 2017	• Metron graduated Apache Incubator
2018	• Metron's latest version: 0.4.3

Apache Metron

- is a cyber security application framework
  - that allows to ingest, process and store diverse security data feeds at scale
  - to detect cyber anomalies and enable a rapid response.
- Has four key features:

1. Security Data Lake / Data Vault
   - Cost effectively stores enriched telemetry data
2. Pluggable Framework
   - Supports pcap, netflow, bro, snort, fireye, sourcefire, …
3. Security Application
   - Has standard **s**ecurity **i**nformation and **e**vent **m**anagement (SIEM) capabilities
4. Threat Intelligence Platform
   - Contains anomaly detection and machine learning algorithms for real-time data

Metron Functional Themes

Platform

- Hardened platform for performance, scale, extensibility and maintainability, provisioning, managing and monitoring Metron

Data Collection

- Metron can stream, ingest and parse into the platform (e.g. using Kafka, etc.)

Data Processing

- Storm topologies allow real-time processing, such as normalization of telemetry data, enrichment, cross reference with threat intel feeds, alerting, indexing, and storing data

User Interface

- Portal, dashboard and user interfaces for different personas

Another look at Metron

- Metron is a centralized tool for security monitoring and analysis.
- Metron integrates several open source big data technologies
  - Kafka, Storm, Kibana, Elasticsearch, and others.
- Metron is capable of:
  - log aggregation, full packet capture indexing, storage, advanced behavioral analytics and data enrichment
- Metron applies threat intelligence information to security telemetry

Metron has … (1)

- **A mechanism to capture, store, and normalize any type of security telemetry at extremely high rates.**
- Security telemetry is constantly being generated
- It should be ingested at high speeds and pushed to appropriate processing units for advanced computation and analytics

Metron has … (2)

- **Real time processing and application of enrichments**
- For example, adding threat intelligence, geolocation, and DNS information to telemetry being collected.
- Near real-time application of this information to incoming telemetry provides the **context** and **situational awareness**,
  - as well as the **who** and **where** information critical for investigation

Metron has … (3)

- **Efficient information storage**
  - Logs and telemetry are stored such that they can be efficiently mined and analyzed
  - Due to the ability to extract and reconstruct full packets, an analyst can answer questions such as **who the true attacker was**, **what data leaked**, and **where**
  - Long-term storage also enables advanced analytics
    - Apply machine learning techniques to build models
    - Incoming data can then be scored against stored models for advanced anomaly detection.

Metron has … (4)

- **An interface for centralized view of data and alerts passed through the system.**
- Metron's interface contains **alert summaries** with **threat intelligence** and **enrichment** data for that alert **on a single page**.
- Advanced search and full packet extraction are available in the same interface.

Metron Architecture (1)

- Parsers : Parsing data from Kafka
- Enrichment : Enriching data after parsing, capability to tag a message as an alert, and assign a risk triage level via a custom rule language.
- Indexing : with Elasticsearch or Solr into HDFS

Metron Architecture (2)

- Stellar : A custom data transformation language used for simple field transformation, expressing triage rules, etc.
- Model as a Service : YARN application which can deploy machine learning / statistical models into a Hadoop cluster
- Data management: saves data in HBase for further use.
- Profiler : A feature extraction that can generate a profile describing the behavior of an entity (a server, user, subnet or application).

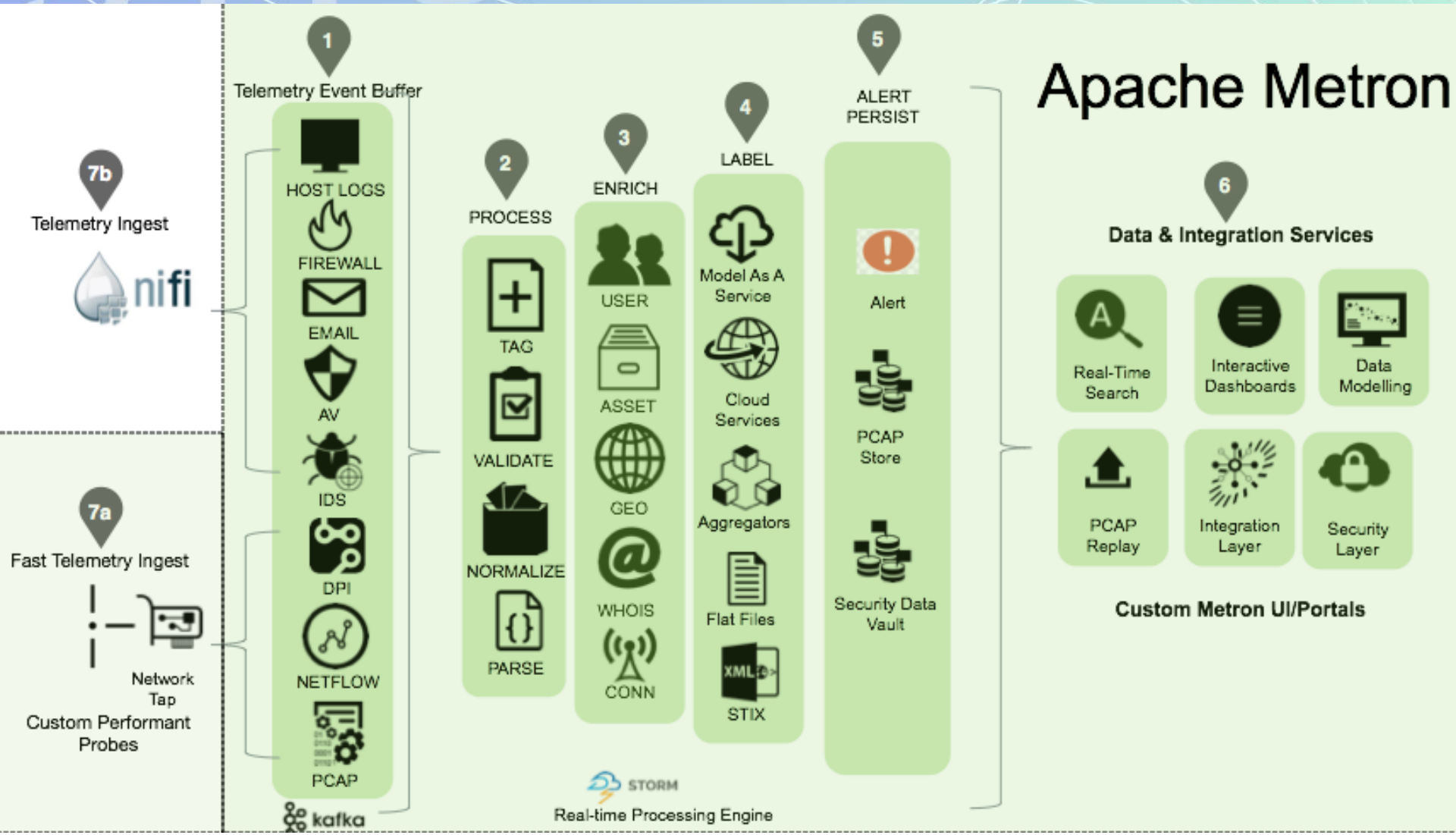
Apache Metron Deployment

- Several deployment scenarios
  - Vagrant-based install
  - Amazon Web Services using EC2 instances
  - Manual install on CentOS 6
  - Ambari Management Pack
  - Ansible-Docker container
  - RPM-Docker
  - RPM packages
  - DEB packages
  - Packer and Virtualbox
  - Single virtual machine

Metron Deployment (more)

- Metron with Kibana and Elasticsearch are included into Hortonworks Cybersecurity Platform
  - Which is and add-on to Hortonworks Data Platform
  - Current versions HDP 2.4.6 & HCP 1.4.1
  - Metron is tested by developers to work with HDP 2.4.5
- HDP & HCP is the best solution for students
  - Due to Ambari graphical user interface
- HCP 1.4.1 does not include the latest Metron
- We built Ambari management pack with the latest Metron
  - Ambari Mpack with Metron, Kibana & Elasticsearch is an analogy to HCP
  - We followed instructions posted in Metron documentation

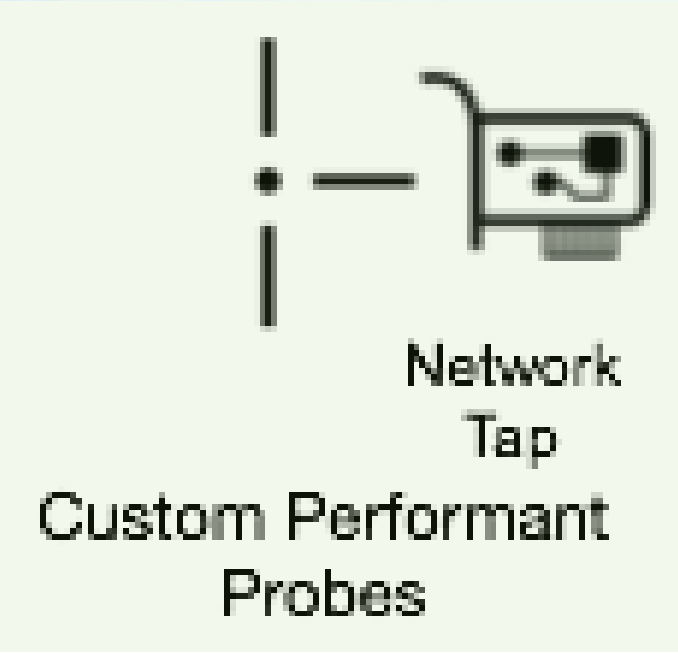
Metron Logical Architecture



Steps 1 to 5: ingest, parse, normalize, enrich, label, index and store all security telemetry data from diverse data sources in an enterprise security data vault.

Metron Step A: Fast Telemetry Ingest

- Data input for high volume network telemetry
  - Packet capture PCAP
  - Netflow / YAF
  - Bro/DPI
  - Custom Metron probes ingesting from network tap



Metron Step A: Fast Telemetry Ingest

Example of raw Bro event captured by Bro probe


```
{
  "http": {
    "id.orig_p": 49206,
    "status_code": 200,
    "method": "GET",
    "request_body_len": 0,
    "id.resp_p": 80,
    "uri": "\/img\/style.css",
    "tags": [],
    "uid": "CqNi7P3HekrXW1OZh8",
    "referrer": "http:\/\/7oqnsnzwwnm6zb7y.gigapaysun.com\/11iQmfg",
    "resp_mime_types": [
      "text\/plain"
    ],
    "trans_depth": 1,
    "host": "7oqnsnzwwnm6zb7y.gigapaysun.com",
    "status_msg": "OK",
    "id.orig_h": "192.168.138.158",
    "response_body_len": 4492,
    "user_agent": "Mozilla\/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; ...",
    "ts": 1.4595338520985450E9,
    "id.resp_h": "95.163.121.204",
    "resp_fuids": [
      "FyAcd62K4Ui32inIc9"
    ]
  }
}
```

Metron Step B: Telemetry Ingest

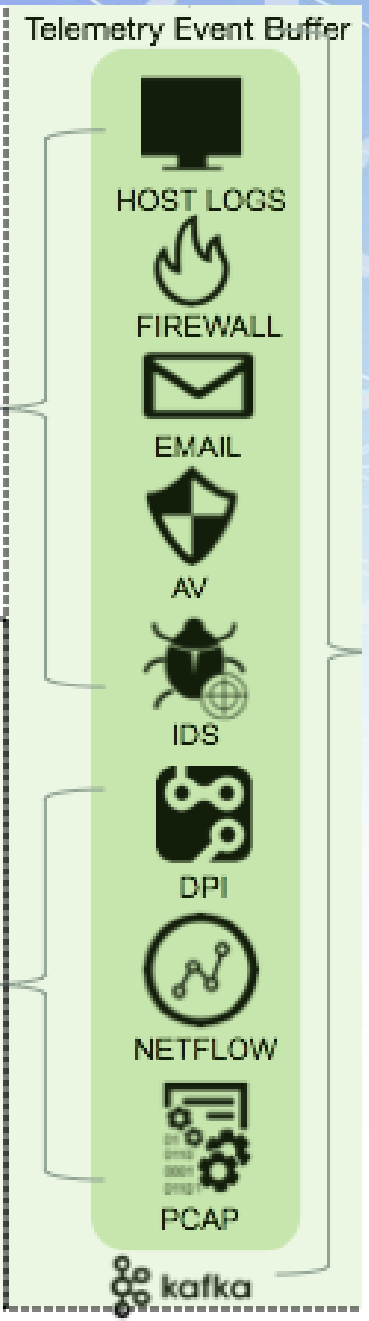
- Metron uses Apache NiFi to ingest data from most telemetry data sources:
  - File
  - Syslog
  - REST
  - HTTP
  - Custom API, etc.
- An example would be capturing data from a FireEye appliance with Nifi's SysLog Processor. The raw Fireye event captured would look something like the following:
-

Metron Step B: Telemetry Ingest

- Example: capturing data from a FireEye appliance with NiFi SysLog Processor
- Raw captured FireEye event:

```
<164>Mar 19 05: 24: 39 10.220.15.15
fenotify-851983.alert: CEF:0|FireEye|CMS|7.2.1.244420|DM|domain-match|1|rt=Feb 09 2015 12: 28:
dvc=10.201.78.57
cn3Label=cncPort
cn3=53
cn2Label=sid
cn2=80494706
shost=dev001srv02.example.com
proto=udp
cs5Label=cncHost
cs5=mfdclk001.org
dvchost=DEVFEYE1
spt=54527
dvc=10.100.25.16
smac=00: 00: 0c: 07:ac: 00
cn1Label=vlan
cn1=0
externalId=851983
cs4Label=link
cs4=https://DEVCMS01.example.com/event_stream/events_for_bot?ev_id\\=851983 dmac=00:1d:a2:af:3
cs1Label=sname
cs1=Trojan.Generic.DNS
```

Metron Step 1: Telemetry Event Buffer



- Raw events from telemetry security data sources
- Will be captured by Apache Nifi or custom Metron probe
- Then pushed into each own Kafka topic
- The arrival into the ingest buffer becomes the beginning of Metron processing

Metron Step 2: Process

STORM
Real-time Processing Engine

- Parse, Normalize, Validate and Tag
- Each raw event will be parsed and normalized into a standardized flat JSON format.
- Every event will be standardized into at least a 7-tuple JSON structure.
- This enables the **topology correlation engine** to work with messages from different topologies using fields such as:
  - **ip_src_addr**: layer 3 source IP
  - **ip_dst_addr**: layer 3 dest IP
  - **ip_src_port**: layer 4 source port
  - **ip_dst_port**: layer 4 dest port
  - **protocol**: layer 4 protocol
  - **timestamp** (epoch)
  - **original_string**: A human friendly string representation of the message
- This step allows validation of a raw event and tagging it with additional metadata, which will be used by downstream processing.

Metron Step 2: Process **Example**


```
{
  "timestamp": 1459533852098,
  "protocol": "http",
  "ip_src_addr": "192.168.138.158",
  "ip_src_port": 49206,
  "ip_dst_addr": "95.163.121.204",
  "ip_dst_port": 80,
  "original_string": "HTTP | id.orig_p:49206 status_code:200 method:GET request_bo
  "bro_timestamp": "1.459533852098545E9",
  "status_code": 200,
  "method": "GET",
  "request_body_len": 0,
  "uri": "\/img\/style.css",
  "tags": [],
  "uid": "CqNi7P3HekrXW1OZh8",
  "referrer": "http:\/\/7oqnsnzwwnm6zb7y.gigapaysun.com\/11iQmfg",
  "resp_mime_types": [
    "text\/plain"
  ],
  "trans_depth": 1,
  "protocol": "http",
  "host": "7oqnsnzwwnm6zb7y.gigapaysun.com",
  "status_msg": "OK",
  "response_body_len": 4492,
  "user_agent": "Mozilla\/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; ...",
  "resp_fuids": [
    "FyAcd62K4Ui32inIc9"
  ]
}
```


Stadard 7 tuple that every element will have

Metron Step 3: Enrich

STORM
Real-time Processing Engine

Example

- An external IP address is enriched with GeoIP information
  - lat/long coordinates & City/State/Country
- or HOST enrichment where an IP gets enriched with Host details
  - IP corresponds to Host X which is part of a web server farm for an e-commerce application

Metron Step 3: Enrich

STORM
Real-time Processing Engine

Geo enrichment for destination and source IPs


```
{
  "timestamp": 1459533852098,
  "protocol": "http",
  "ip_src_addr": "192.168.138.158",
  "ip_src_port": 49206,
  "ip_dst_addr": "95.163.121.204",
  "ip_dst_port": 80,
  "original_string": "HTTP | id.orig_p:49206 status_c  200 metho
  "enrichments.geo.dip.location_point": "41.789029, -88.1333654",
  "enrichments.geo.dip.latitude": "41.789029",
  "enrichments.geo.dip.longitude": "-88.1333654",
  "enrichments.geo.dip.country": "US",
  "enrichments.geo.dip.city": "Naperville",
  "enrichments.geo.dip.postalCode": "60563",
  "enrichments.geo.sip.location_point": "38.635952, -90.223868",
  "enrichments.geo.sip.latitude": "38.635952",
  "enrichments.geo.sip.longitude": "-90.223868",
  "enrichments.geo.sip.country": "US",
  "enrichments.geo.sip.city": "St. Louis",
  "enrichments.geo.sip.postalCode": "63103",
  "bro_timestamp": "1.4595338520985 5E9",
  "status_code": 200,
  "method": "GET",
```

Metron Step 4: Label

- Labeling includes **threat intel cross reference checks**
  - elements of a telemetry are looked up against threat intel feed data sources like Soltra Edge, produced by Stix/Taxii feeds or other threat intel aggregators
  - These threat intel services will then label the telemetry event with threat intel metadata when a hit occurs.
- Also possible to apply analytical models for scoring to telemetry events that are flowing in

Example of a bro event producing a threat intel hit

"threatintels.hbaseThreatIntel.ip_src_addr.malicious_ip" : "alert",

"enrichments.hbaseEnrichment.ip_src_addr.malicious_ip.source-type" : "STIX",

"en richments.hbaseEn richment.ip_s rc_add r.malicious_ip.indicator-type" : "add ress:IPV_4_ADDR",
"enrichments.hbaseEnrichment.ip_src_addr.malicious_ip.source" : "some xml snipeet from STIX file"

Metron Step 5: Alert Persist

ALERT PERSIST

- Certain telemetry events can initiate alerts
  - Then indexed in an alert index store
- Triggering factors:
  - Event type: e.g. any event generated by Snort is an alert
  - Threat intel hit
- All enriched and labeled telemetry events
  - **Indexed** by Elasticsearch or Solr
  - **Preserved** in Hadoop HDFS
  - This forms an enterprise **data vault**

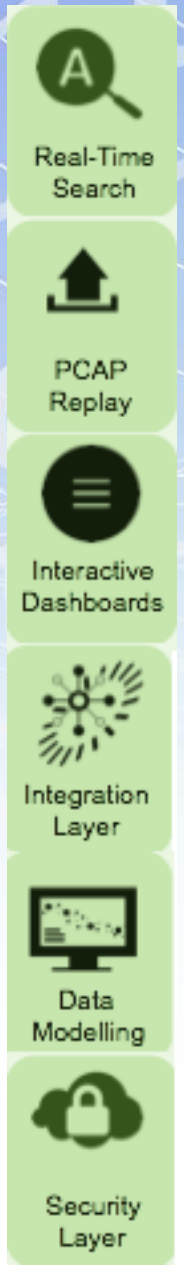
Alert

PCAP Store

Security Data Vault

Metron Step 5: Event stored in HDFS

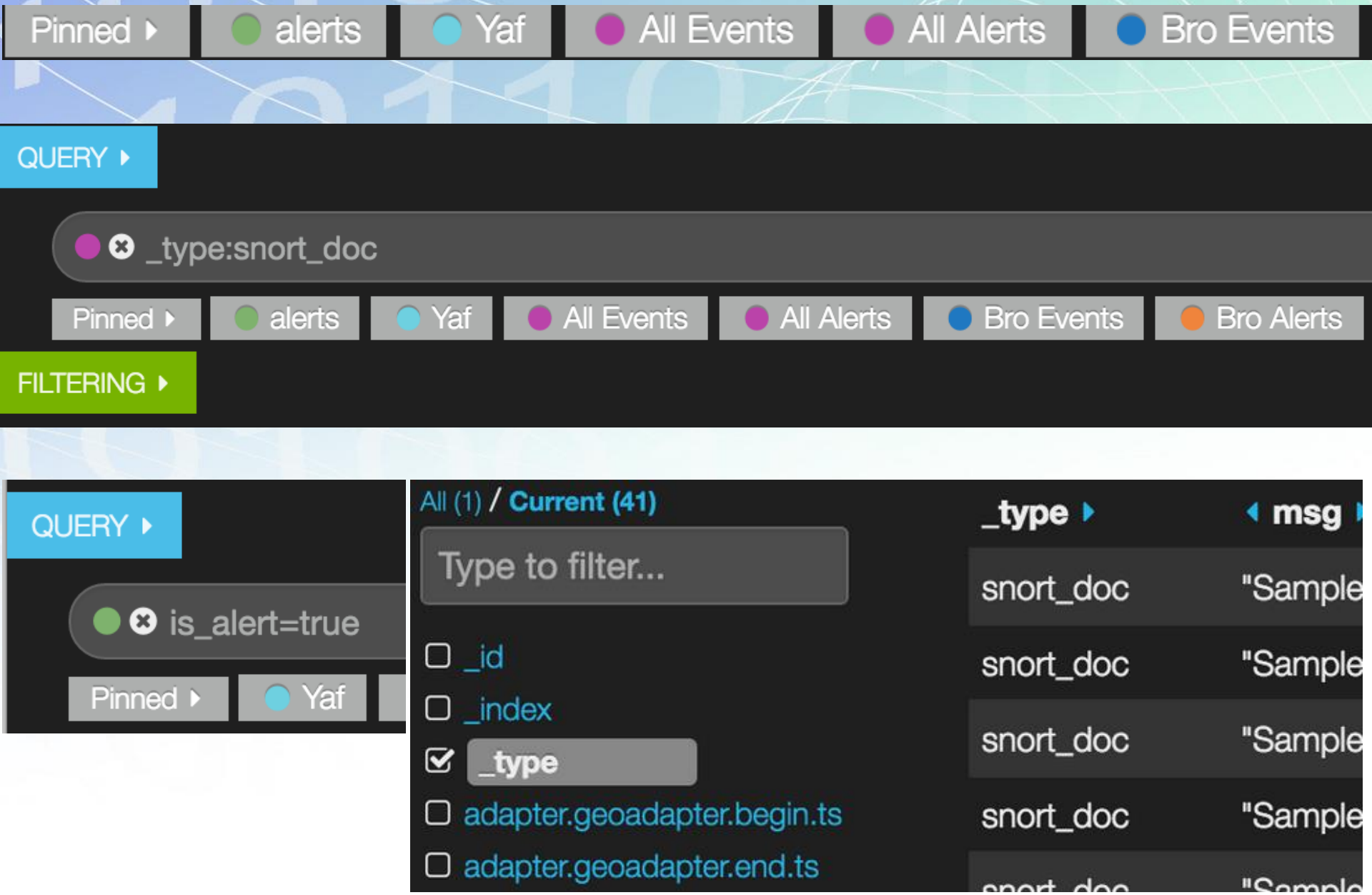
Metron Step 6: UI Portal & Data Integration



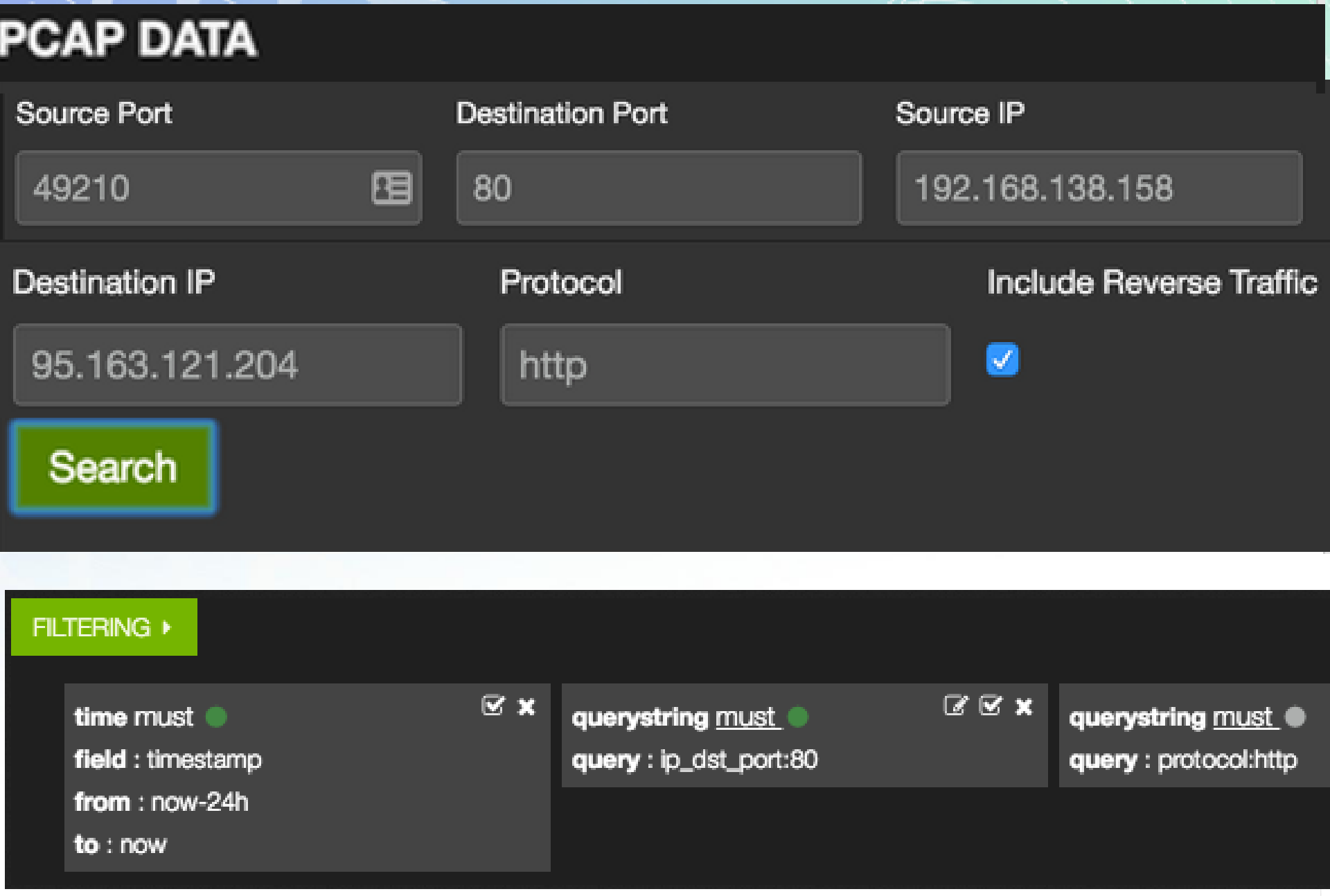
Metron platform provides with a set of services:

- Real-time Search and Interactive Dashboards / Portals
  - single interface for security operation analysts to view alerts and correlate to telemetry events that caused them.
- Data Modeling / Feature Engineering Services by tools such as Jupyter, IPython and Zeppelin.
- Integration and Extensibility Layers - customization for own needs/requirements such as:
  - Ingesting new data sources
  - Adding new parsers
  - Adding new enrichment services
  - Adding new Threat Intel feeds
  - Building, deploying and executing new analytical models
  - Integration with enterprise workflow engines
  - Customizing the Security Dashboards and portals

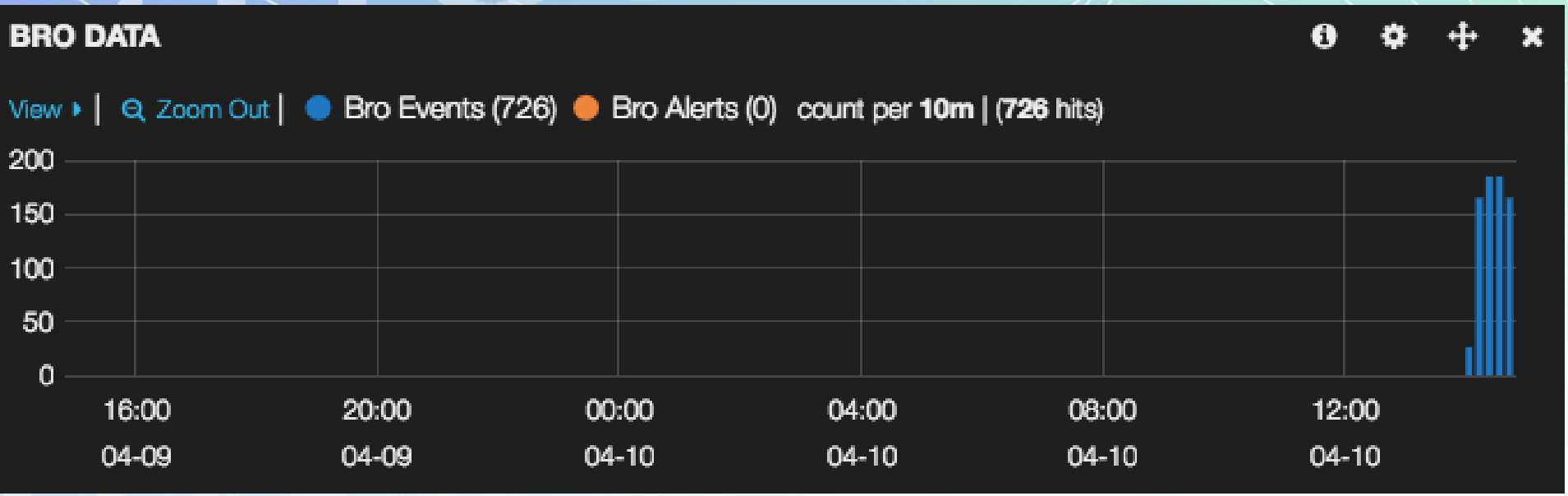
Metron UI Portal Example 1

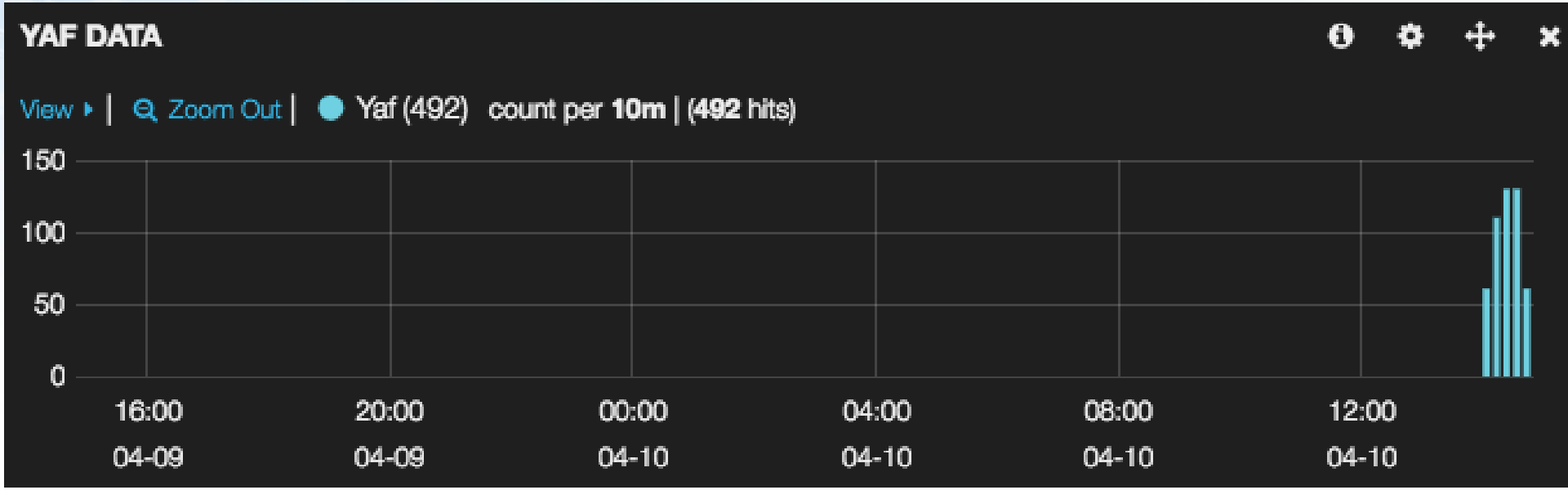


Metron UI Portal Example 2



Metron UI Portal Example 3





Metron UI Portal Example 4

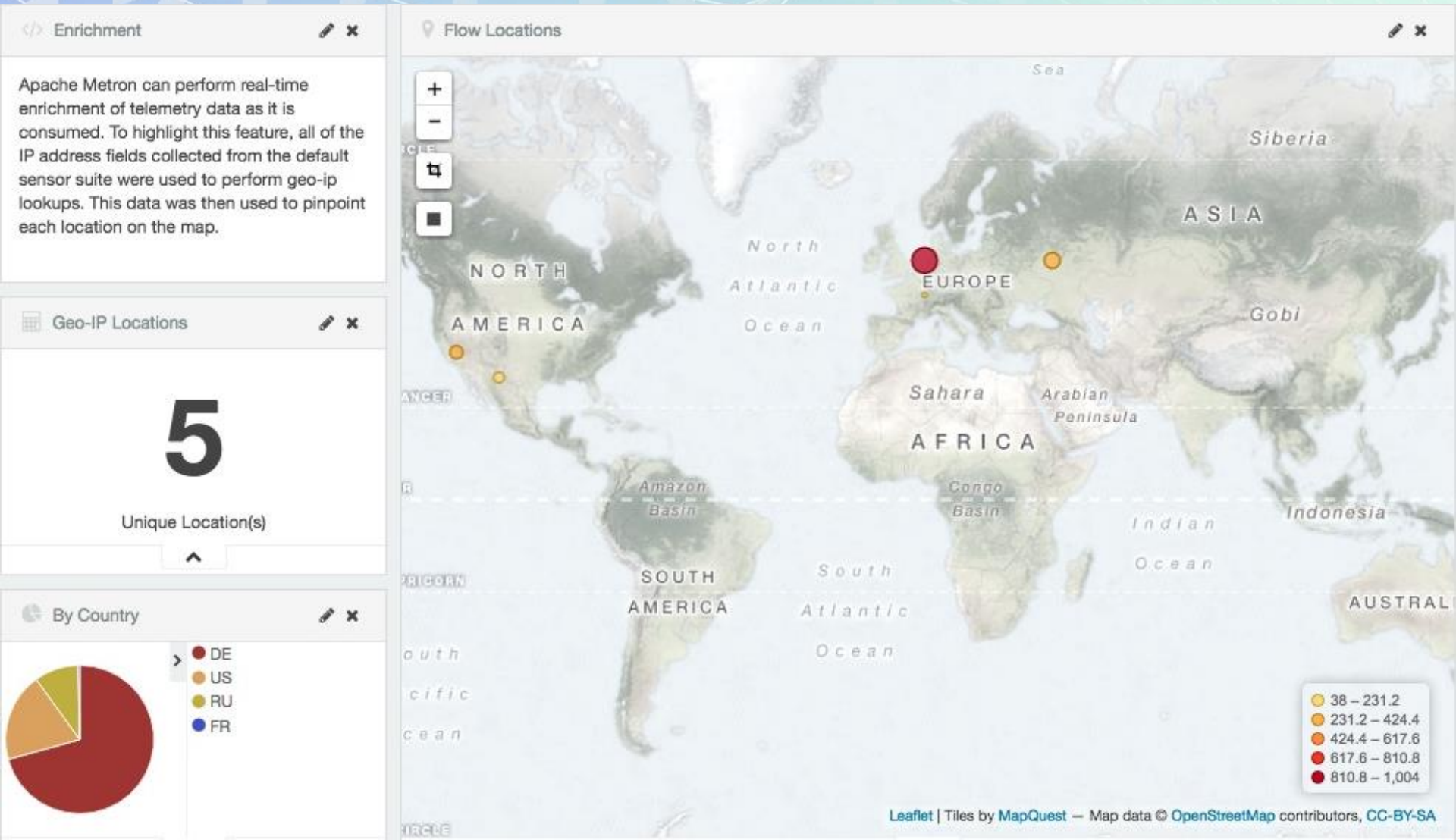
ALERTS

0 to **10** of 1000 available for paging

_type	msg	ip_src_addr	ip_src_port	ip_dst_addr	ip_dst_
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49189	62.75.195.236	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49186	62.75.195.236	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49189	62.75.195.236	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49206	95.163.121.204	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49201	204.152.254.221	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49201	204.152.254.221	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49201	204.152.254.221	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49201	204.152.254.221	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49186	62.75.195.236	80
snort_doc	"Sample Metron Message from Snort"	192.168.138.158	49189	62.75.195.236	80

0 to **10** of 1000 available for paging

Metron Dashboard Panel 1



Metron Dashboard Panel 2

YAF

YAF can be used to generate Netflow-like flow records. These flow records provide significant visibility of the actors communicating over the target network.

YAF Flows

362

Count

Flow Duration

YAF

1 2 3 4 5 ...8 »

Time	ip_src_addr	ip_src_port	ip_dst_addr	ip_dst_port	protocol
June 21st 2016, 11:11:35.878	192.168.138.158	50,509	192.168.138.2	53	UDP
June 21st 2016, 11:11:30.092	192.168.138.158	50,683	192.168.138.2	53	UDP
June 21st 2016, 11:11:29.303	192.168.138.158	60,078	192.168.138.2	53	UDP
June 21st 2016, 11:11:30.092	192.168.138.158	65,315	192.168.138.2	53	UDP
June 21st 2016, 11:11:35.225	192.168.138.158	53,571	192.168.138.2	53	UDP
June 21st 2016, 11:11:35.641	192.168.138.158	61,720	192.168.138.2	53	UDP
June 21st 2016, 11:11:36.064	192.168.138.158	56,753	192.168.138.2	53	UDP
June 21st 2016, 11:12:13.923	192.168.138.158	50,329	192.168.138.2	53	UDP
June 21st 2016, 10:54:49.185	192.168.138.158	49,186	62.75.195.236	80	TCP
June 21st 2016, 11:11:30.250	192.168.138.158	49,186	62.75.195.236	80	TCP
June 21st 2016, 11:05:14.851	192.168.138.158	49,186	62.75.195.236	80	TCP
June 21st 2016, 11:07:19.985	192.168.138.158	49,186	62.75.195.236	80	TCP
June 21st 2016, 10:58:59.455	192.168.138.158	49,186	62.75.195.236	80	TCP
June 21st 2016, 11:19:50.779	192.168.138.158	49,186	62.75.195.236	80	TCP
June 21st 2016, 11:15:40.515	192.168.138.158	49,186	62.75.195.236	80	TCP
June 21st 2016, 10:56:54.323	192.168.138.158	49,186	62.75.195.236	80	TCP

Metron Dashboard Panel 3

Snort

Snort is a Network Intrusion Detection System (NIDS) that is being used to generate alerts identifying known bad events. Snort relies on a fixed set of rules that act as signatures for identifying abnormal events.

Snort Alert Types

1

Alert Type(s)

Top Alerts By Host

Source	Destination	Count
62.75.195.236	192.168.138.158	2,201
192.168.138.158	62.75.195.236	1,253
192.168.138.158	95.163.121.204	321
192.168.138.158	72.34.49.86	284

Snort Alerts

1 2 3 4 5 ...10 »

Time	msg	sig_id	ip_src_addr	ip_src_port	ip_dst_addr	ip_ds
June 21st 2016, 11:21:44.769	"'snort test alert'"	999,158	95.163.121.204	80	192.168.138.158	49,20
June 21st 2016, 11:21:44.640	"'snort test alert'"	999,158	192.168.138.158	49,209	95.163.121.204	80
June 21st 2016, 11:21:44.552	"'snort test alert'"	999,158	192.168.138.158	49,189	62.75.195.236	80
June 21st 2016, 11:21:44.529	"'snort test alert'"	999,158	192.168.138.158	49,206	95.163.121.204	80
June 21st 2016, 11:21:44.390	"'snort test alert'"	999,158	62.75.195.236	80	192.168.138.158	49,18
June 21st 2016, 11:21:42.398	"'snort test alert'"	999,158	192.168.138.158	49,209	95.163.121.204	80
June 21st 2016, 11:21:42.277	"'snort test alert'"	999,158	95.163.121.204	80	192.168.138.158	49,20
June 21st 2016, 11:21:41.086	"'snort test alert'"	999,158	72.34.49.86	80	192.168.138.158	49,20
June 21st 2016, 11:21:41.061	"'snort test alert'"	999,158	192.168.138.158	49,202	72.34.49.86	80
June 21st 2016, 11:21:40.880	"'snort test alert'"	999,158	72.34.49.86	80	192.168.138.158	49,20

Metron Dashboard Panel 4

Web Request Header

The Bro Network Security Monitor is extracting application-level information from raw network packets. In this example, Bro is extracting HTTP(S) requests being made over the network.

Web Requests

445

Count

Web Request Type

Web Requests

1 2 3 4 5 ...9 »

Time	method	host	uri	referrer
June 21st 2016, 11:20:53.147	GET	7oqnsnzwwnm6zb7y.gigapaysun.com	/img/button_pay.png	http://7oqnsnzwwnm6zb7y.gigapaysun.com/11iQmfg
June 21st 2016, 11:20:53.146	GET	7oqnsnzwwnm6zb7y.gigapaysun.com	/img/bitcoin.png	http://7oqnsnzwwnm6zb7y.gigapaysun.com/11iQmfg
June 21st 2016, 11:20:51.040	GET	7oqnsnzwwnm6zb7y.gigapaysun.com	/img/style.css	http://7oqnsnzwwnm6zb7y.gigapaysun.com/11iQmfg
June 21st 2016, 11:20:48.304	POST	7oqnsnzwwnm6zb7y.gigapaysun.com	/11iQmfg	http://7oqnsnzwwnm6zb7y.gigapaysun.com/11iQmfg

Metron Dashboard Panel 5

DNS Requests

Bro is extracting DNS requests and responses being made over the network. Understanding who is making those requests, the frequency, and types can provide a deep understanding of the actors present on the network.

DNS Requests

96

Count

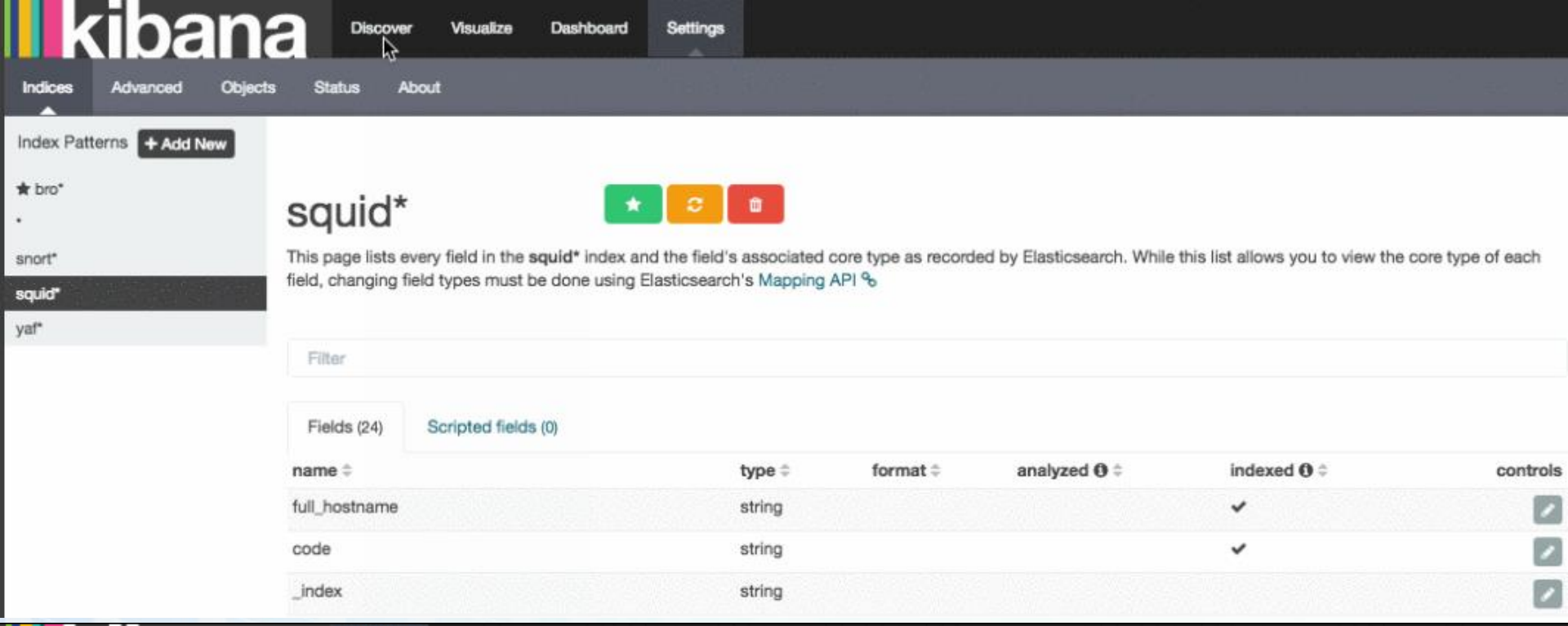
DNS Requests

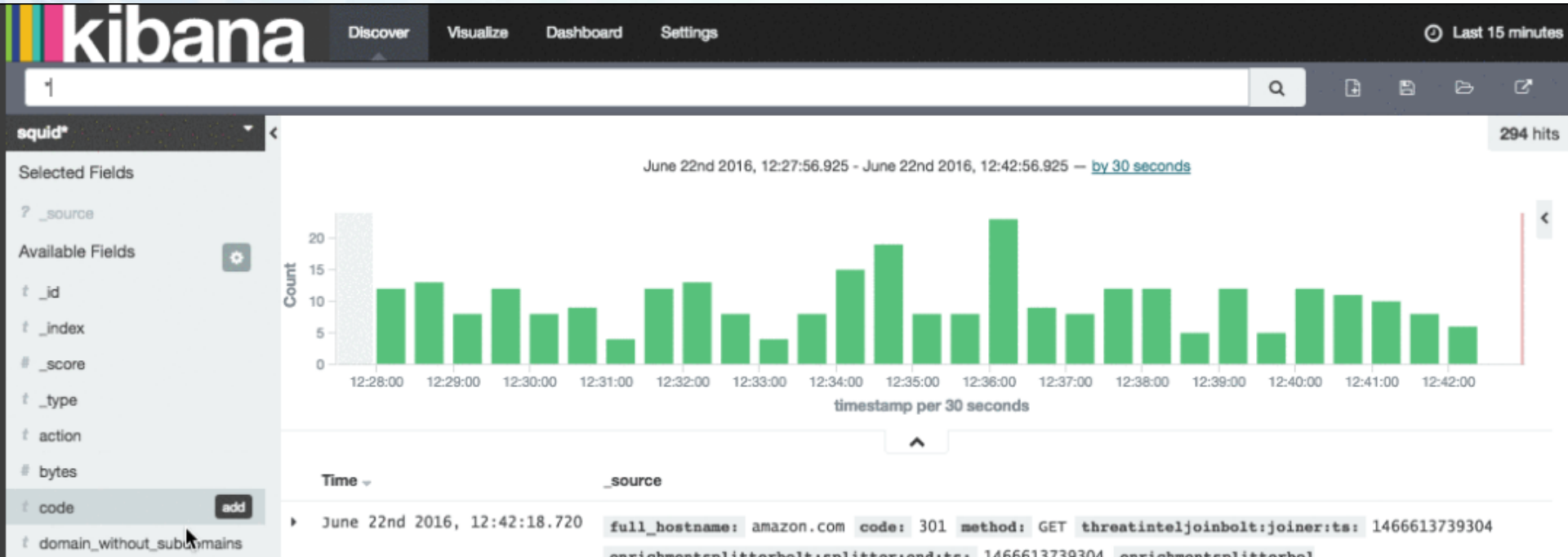
1 2 »

Time	query	qtype_name	answers	ip_src_addr	ip_dst_add
June 21st 2016, 11:20:34.452	7oqnsnzwwnm6zb7y.gigapaysun.com	A	95.163.121.204	192.168.138.158	192.168.13
June 21st 2016, 11:19:56.592	comarksecurity.com	A	72.34.49.86	192.168.138.158	192.168.13
June 21st 2016, 11:19:56.407	kritischerkonsum.uni-koeln.de	A		192.168.138.158	192.168.13
June 21st 2016, 11:19:56.169	runlove.us	A	204.152.254.221	192.168.138.158	192.168.13
June 21st 2016, 11:19:55.753	ip-addr.es	A	188.165.164.184	192.168.138.158	192.168.13
June 21st 2016, 11:19:50.622	r03afd2.c3008e.xc07r.b0f.a39.h7f0fa5eu.vb8fbl.e8mfzdgrf7g0.groupprograms.in	A	62.75.195.236	192.168.138.158	192.168.13
June 21st 2016, 11:19:50.621	ubb67.3c147o.u806a4.w07d919.o5f.f1.b80w.r0faf9.e8mfzdgrf7g0.groupprograms.in	A	62.75.195.236	192.168.138.158	192.168.13
June 21st 2016, 11:19:49.832	va872g.g90e1h.b8.642b63u.j985a2.v33e.37.pa269cc.e8mfzdgrf7g0.groupprograms.in	A	62.75.195.236	192.168.138.158	192.168.13
June 21st 2016, 11:18:29.320	7oqnsnzwwnm6zb7y.gigapaysun.com	A	95.163.121.204	192.168.138.158	192.168.13

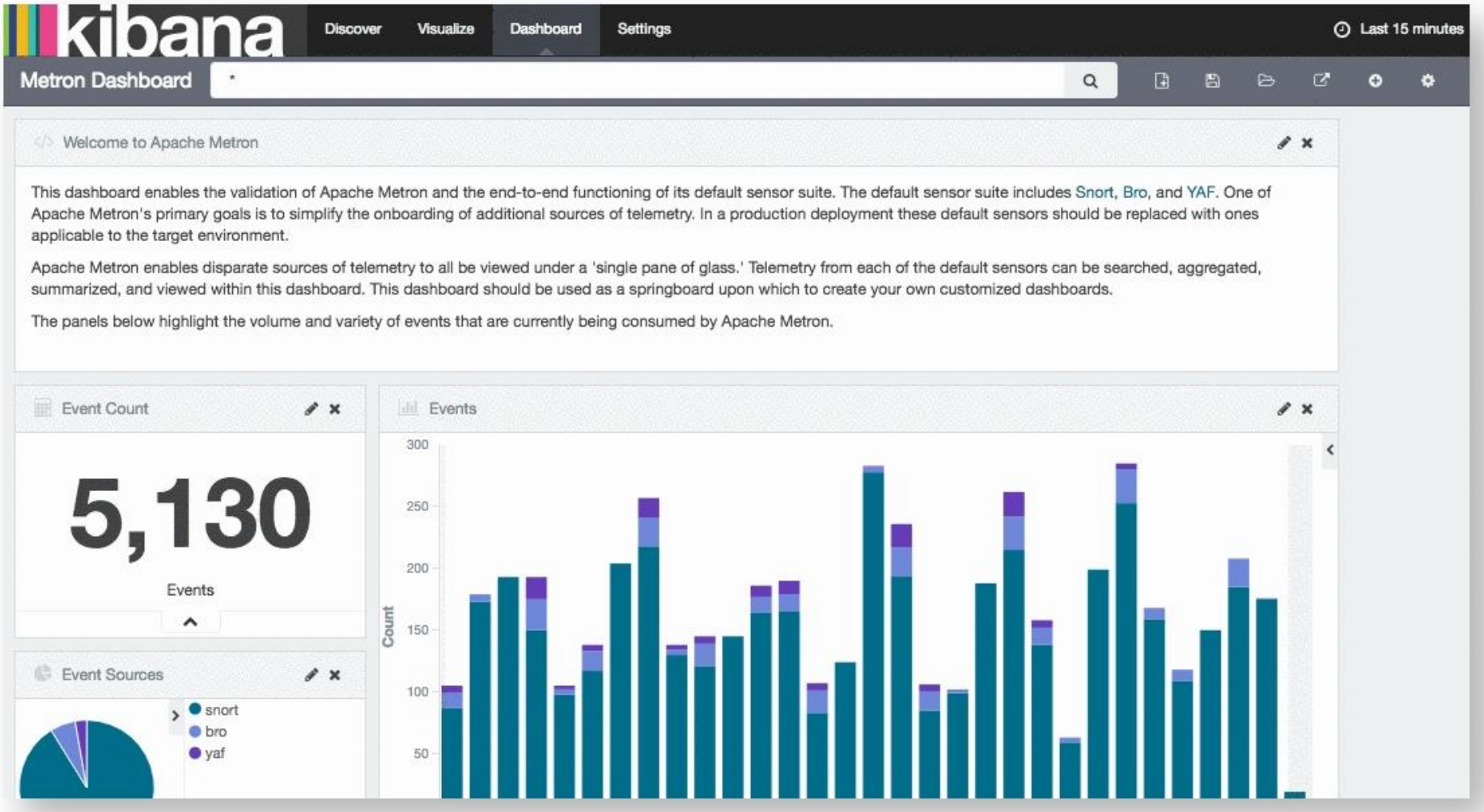
Metron Users & Application

- **SOC Analyst**: Don't spend days looking at alerts created by rules when only a few alerts matter
- **SOC Investigator**: Metron enables massive amounts of data to identify and triage anomalies
- **SOC Manager**: Automatically create incidents/cases with integrated workflow systems
- **Forensic Investigator**: “Just-in-time evidence collection response” transforms data in real-time
- **Security Platform Engineer**: Single platform to manage and operate the ingestion, processing of cyber data
- **Security Data Scientist**: Perform data science activities: train, evaluate and score analytical models


kibana
Discover
Visualize
Dashboard
Settings
Indices
Advanced
Objects
Status
About
Index Patterns
+ Add New
★ bro*
snort*
squid*
yaf*
squid*
This page lists every field in the squid* index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API
Filter
Fields (24)
Scripted fields (0)
name
type
format
analyzed
indexed
controls
full_hostname
string
code
string
_index
string


kibana
Discover
Visualize
Dashboard
Settings
Last 15 minutes
*
squid*
294 hits
Selected Fields
_source
Available Fields
_id
_index
_score
_type
action
bytes
code
add
domain_without_subdomains
June 22nd 2016, 12:27:56.925 - June 22nd 2016, 12:42:56.925 — by 30 seconds
Count
20
15
10
5
0
12:28:00
12:29:00
12:30:00
12:31:00
12:32:00
12:33:00
12:34:00
12:35:00
12:36:00
12:37:00
12:38:00
12:39:00
12:40:00
12:41:00
12:42:00
timestamp per 30 seconds
Time
_source
June 22nd 2016, 12:42:18.720
full_hostname: amazon.com code: 301 method: GET threatinteljoinbolt:joiner:ts: 1466613739304

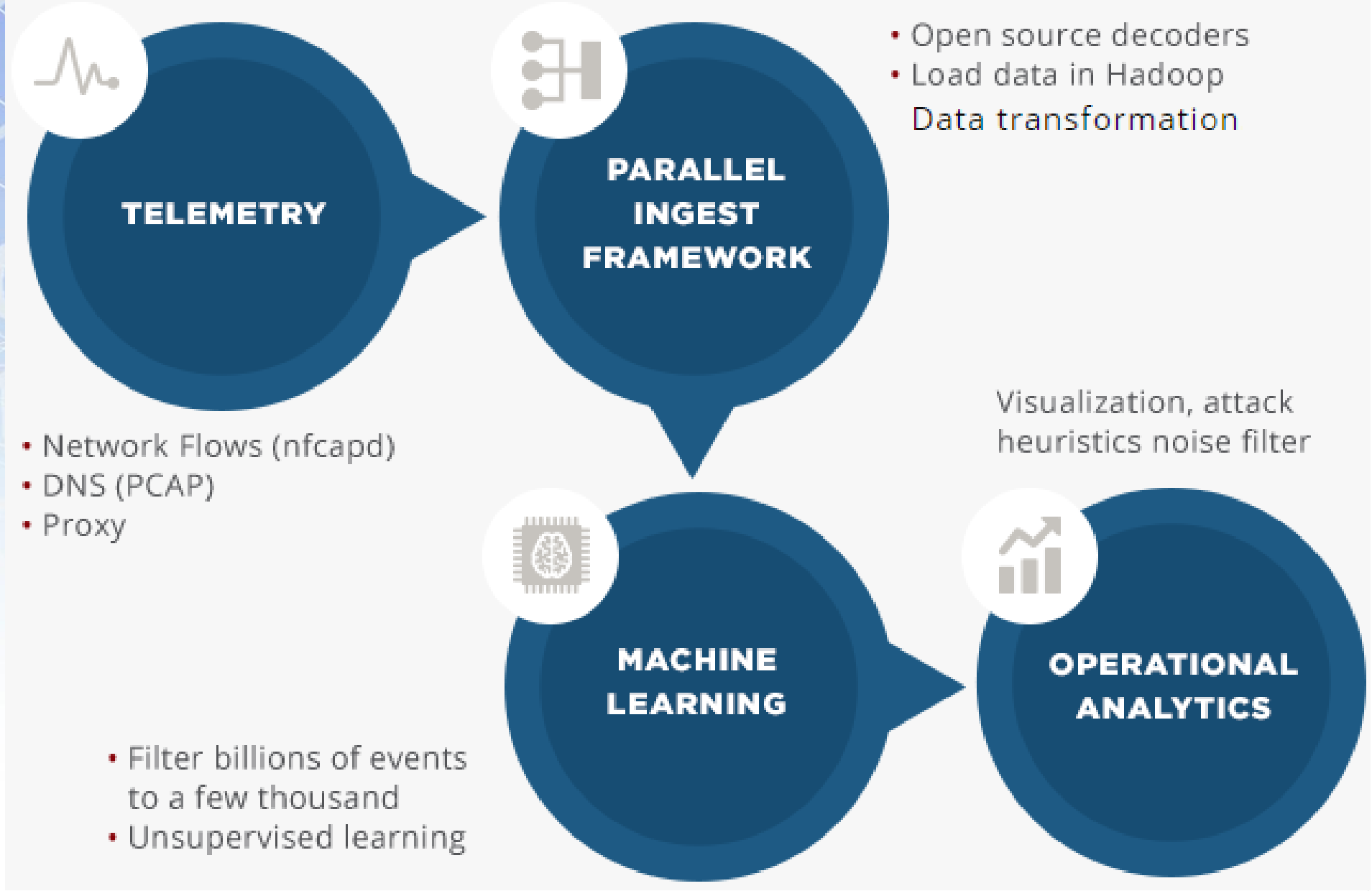
Metron Dashboard in Kibana



Part 3 Apache Spot

http://spot.incubator.apache.org

Apache Spot Architecture



Apache Spot

- Open-source software for analysis of telemetry data flow and packet analysis
- Provides with insight on networks
- Identifies potential security threats or happening attacks
- Accelerates exposing suspicious connections and previously unseen attacks
  - using flow and packet analysis technologies
- Status: development in Apache Incubator
  - Version 1.0 is the latest release (on August 7, 2017)
  - Newest under-development code is on GitHub

Apache Spot UI with sample data

- Running Demo on Docker
  - Install Docker for your platform
  - Run the container:

  docker run -it -p 8889:8889 apachespot/spot-demo
- visit URL in your browser to get started:

http://localhost:8889/files/ui/flow/suspicious.html#date=2016-07-08

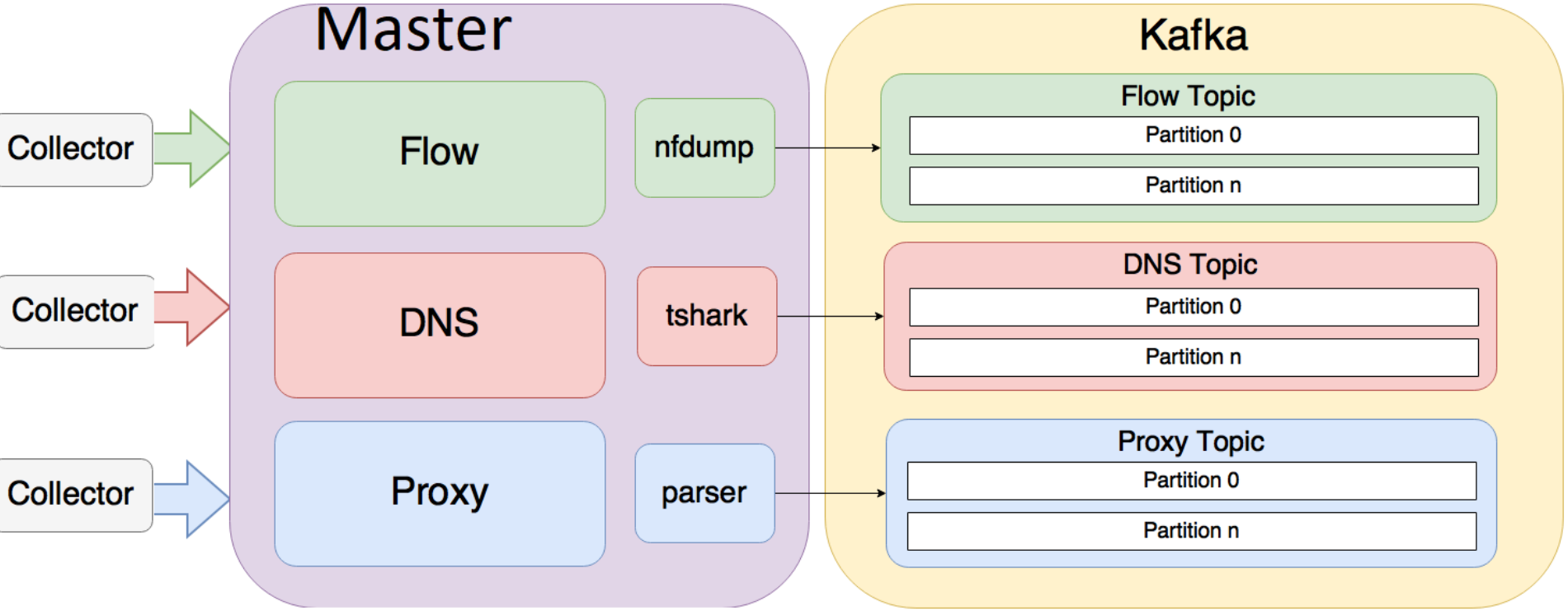
- For the full instructions visit the spot on Docker hub

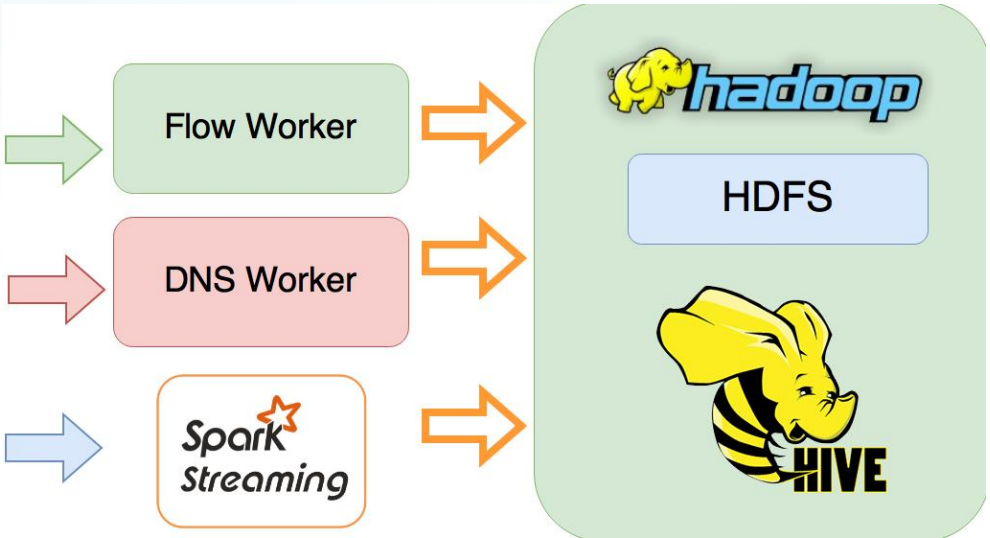
https://hub.docker.com/r/apachespot/spot-demo/

Apache Spot Modules

- spot-ingest
  - Ingest data is captured or transferred into the Hadoop cluster, where they are transformed and loaded into solution data stores.
- spot-ml
  - contains routines for performing *suspicious connections* analyses on netflow, DNS or proxy data gathered from a network.
- spot-oa
  - Operational Analytics (OA) is a collection of modules, which includes both the data processing and transformation as well as the GUI module for data visualization.
- spot-setup
  - Technical aspects of the setup installation process of the Apache Spot solution.

Apache Spot **Ingest**



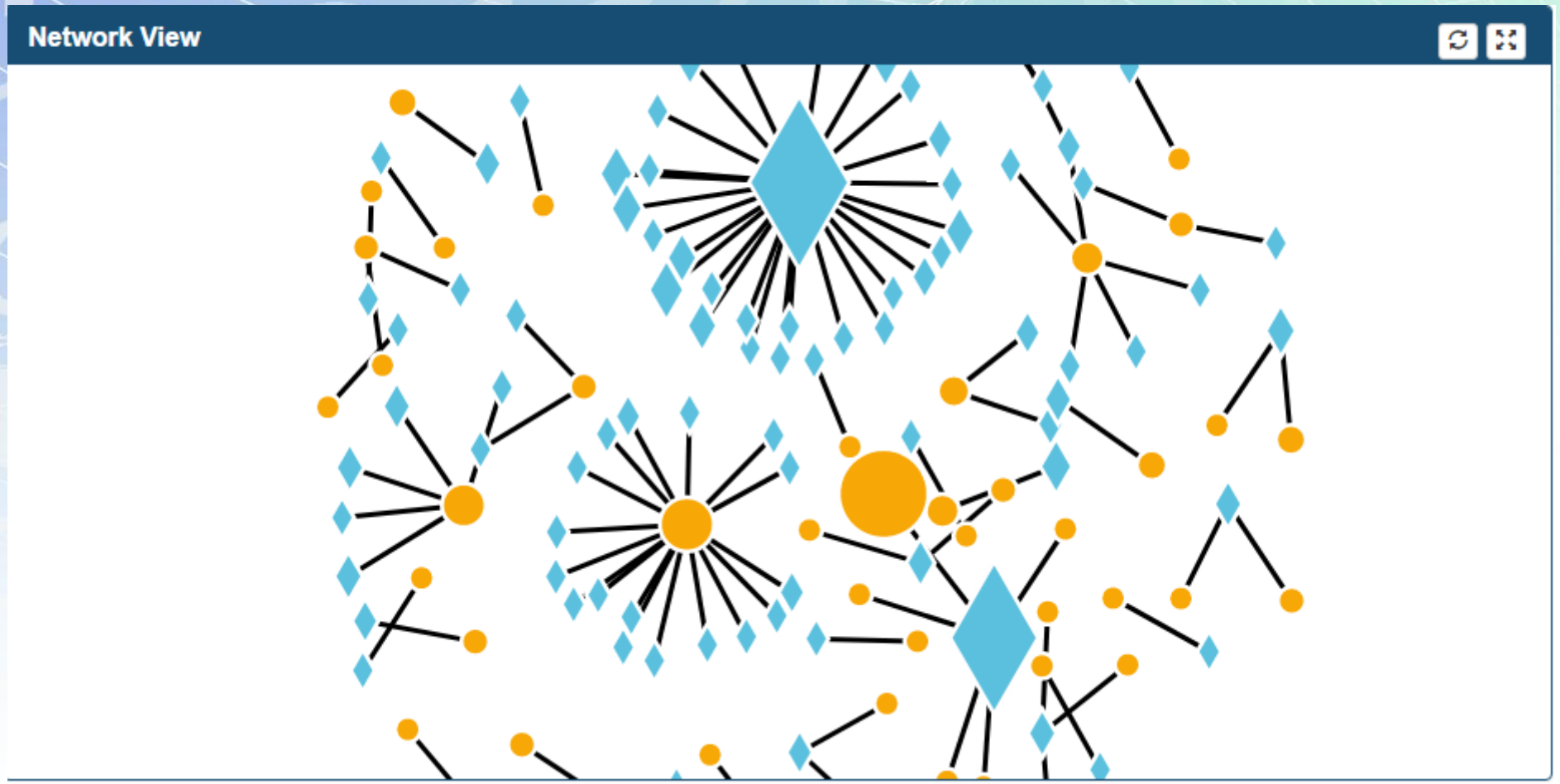


Apache Spot Interface

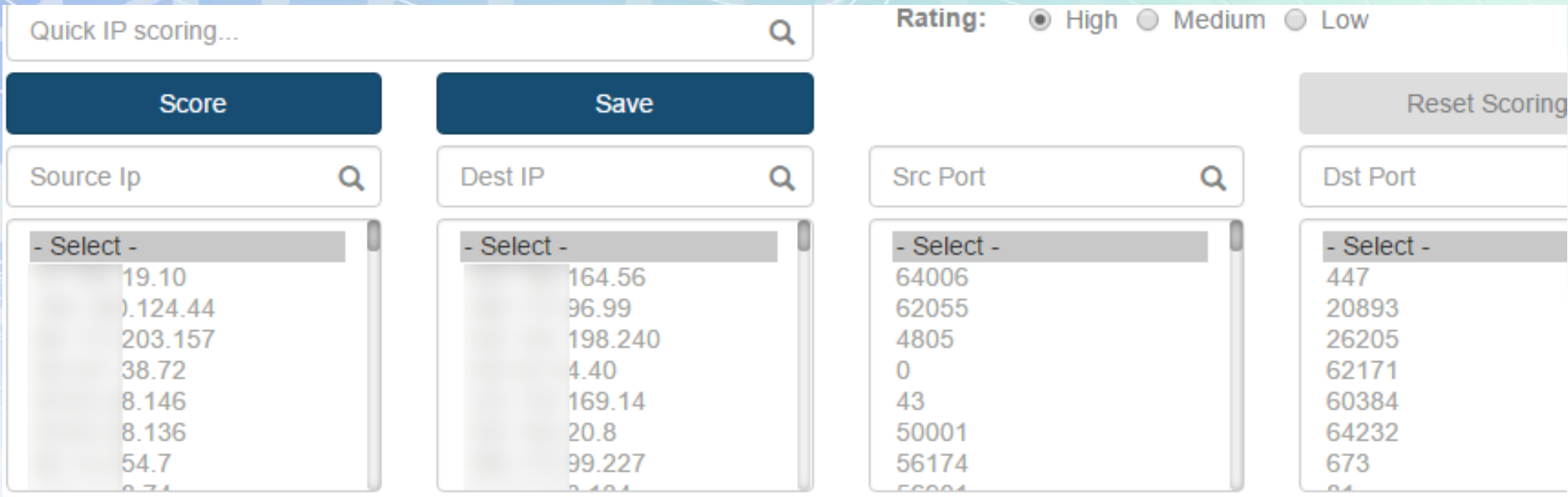
Suspicious

Rank	Time	Source IP	Destination IP	Source Port	Destination Port	Protocol	Input Packets	Input Bytes	Output Packets	Output Bytes
0	2016-10-05 15:33:33	.19.10	.164.56	64006	447	TCP	3	940	0	0
1	2016-10-05 03:07:17	.124.44	.96.99	62055	447	TCP	2	597	0	0
2	2016-10-05 05:38:21	.203.157	.164.56	4805	447	TCP	4	2136	0	0
3	2016-10-05 02:22:11	.38.72	.198.240	0	20893	ICMP	1	139	0	0
4	2016-10-05 14:26:52	59.63.28.146	192.102.198.240	0	26205	ICMP	1	156	0	0
5	2016-10-	28.136	.198.240	0	62171	ICMP	1	133	0	0

Apache Spot Interface



Apache Spot Interface



Apache Spot Interface

Suspicious

5	2016-10-05 15:45:30	.28.136		.198.240		0	62171	ICMP	1	133
6	2016-10-05 02:31:47	.254.7		.198.240		0	60384	ICMP	1	150
7	2016-10-05 18:05:06	.58.74		.54.40		43	64232	TCP	2	80
8	2016-10-05 06:13:24	.230.29		.169.14		50001	673			132
9	2016-10-05 16:48:58	.151.44		.20.8		56174	81	TCP	11	452
10	2016-10-05 16:48:58	.151.44		.20.8		56174	81	TCP	11	452

Row Selected

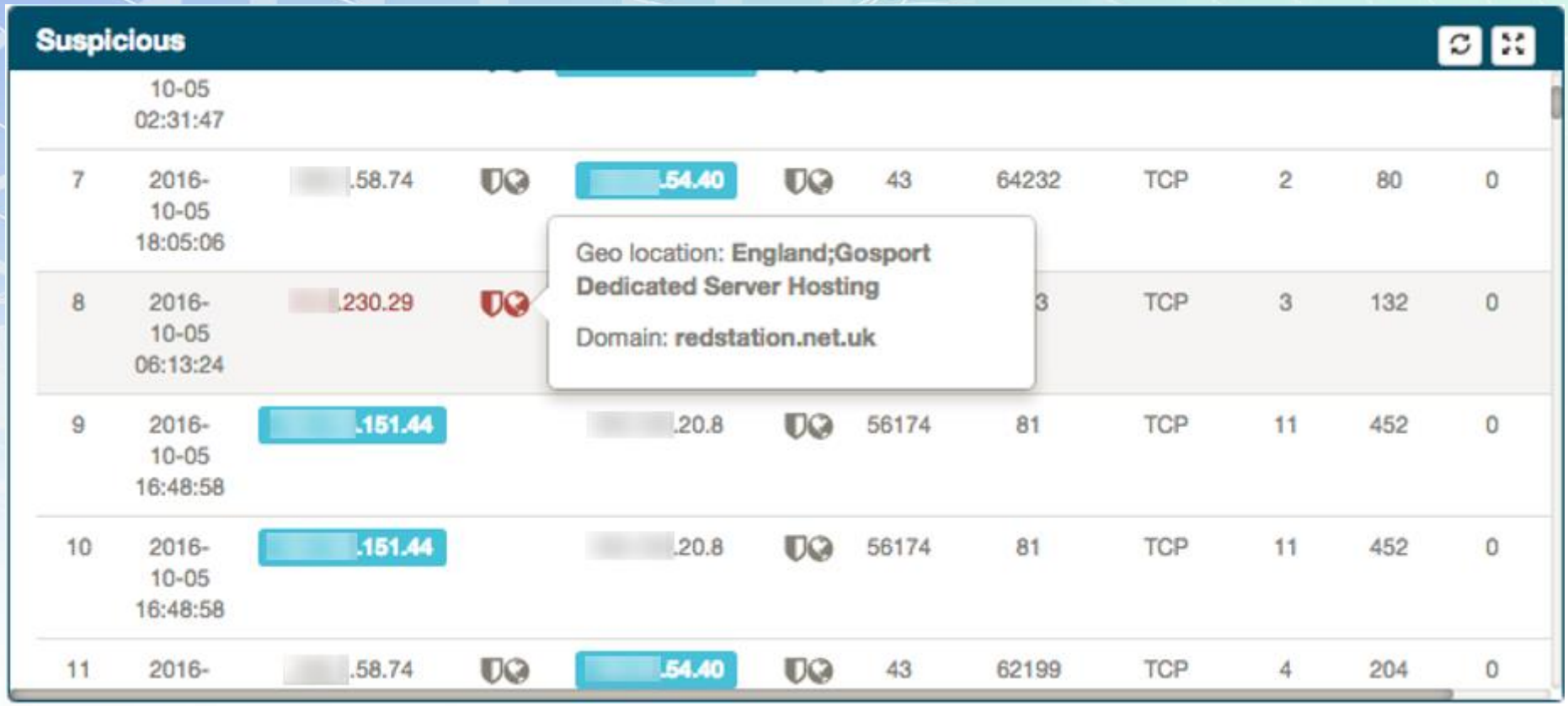
Apache Spot Interface

Suspicious

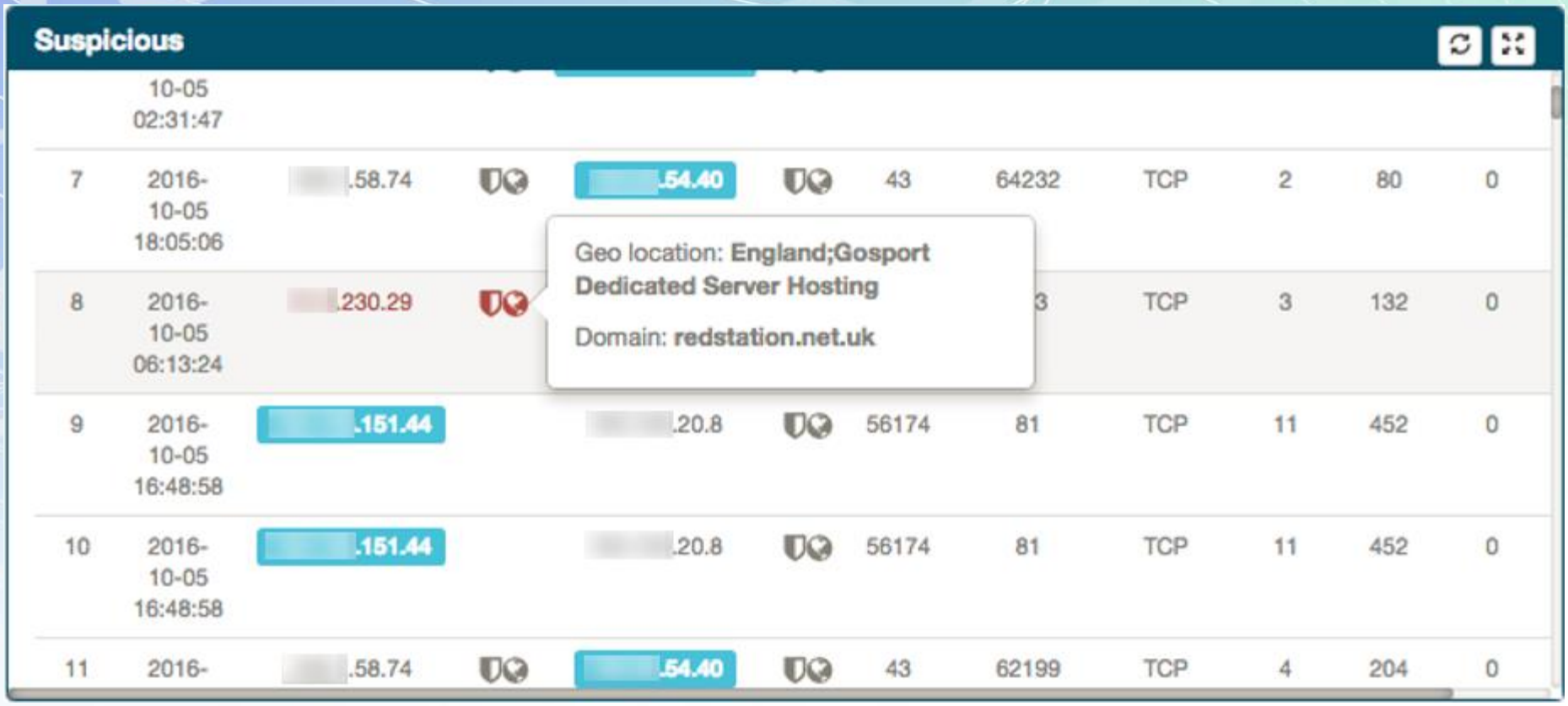
5	2016-10-05 15:45:30	.28.136		.198.240		0	62171	ICMP	1	133
6	2016-10-05 02:31:47	.254.7		.198.240		0	60384	ICMP	1	150
7	2016-10-05 18:05:06	.58.74		.54.40		43	64232	TCP	2	80
8	2016-10-05 06:13:24	.230.29		.169.14		50001	673			132
9	2016-10-05 16:48:58	.151.44		.20.8		56174	81	TCP	11	452
10	2016-10-05 16:48:58	.151.44		.20.8		56174	81	TCP	11	452

Row Selected

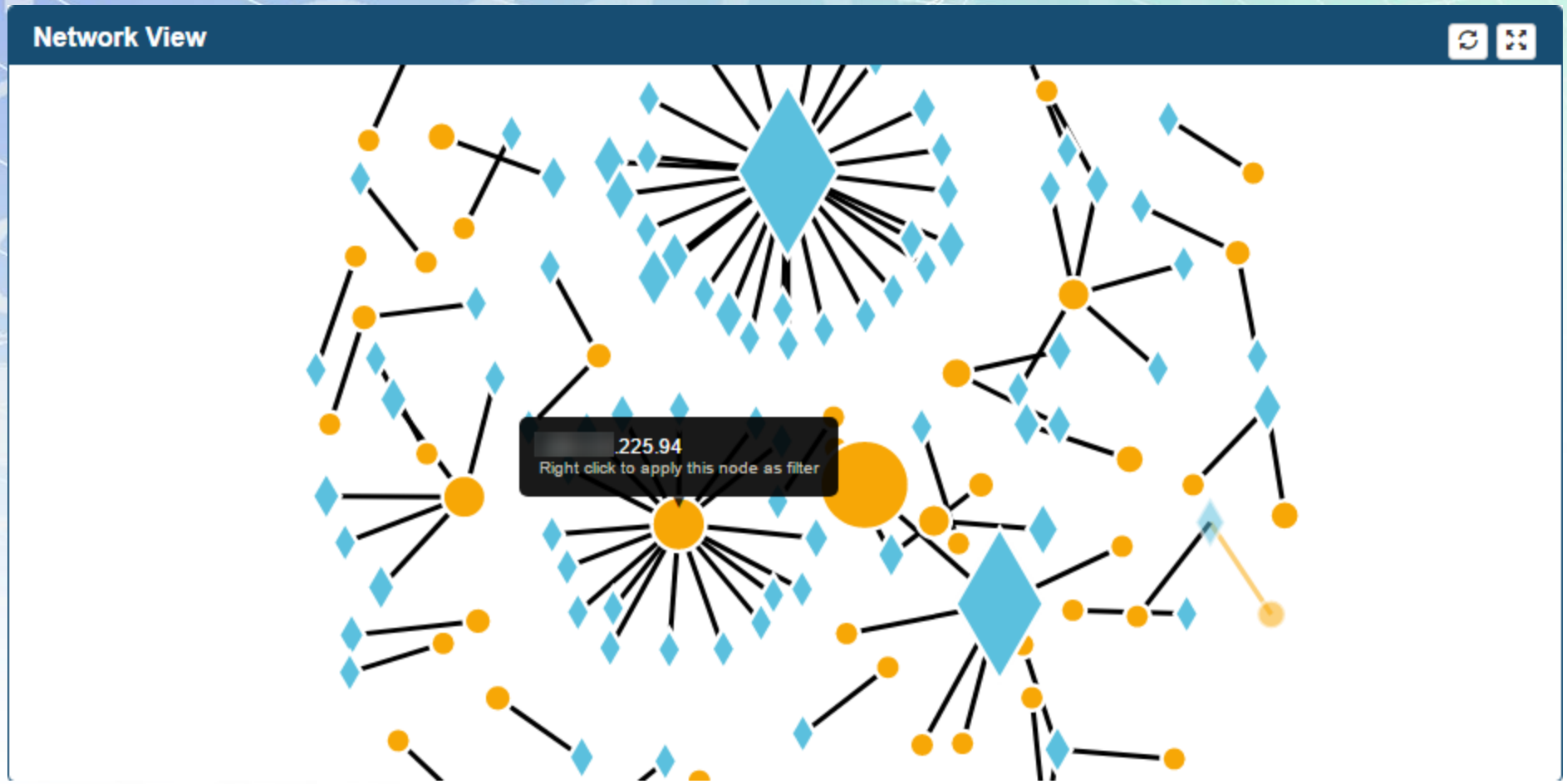
Apache Spot Interface



Apache Spot Interface



Apache Spot Interface



References

- http://spot.incubator.apache.org
- http://opensoc.github.io
- https://community.hortonworks.com/articles/26812/metron-ui-finding-a-needle-in-a-haystack.html
- https://metron.apache.org/
- https://github.com/apache/metron
- https://github.com/apache/incubator-spot