Apache Metron

A Case Study of a Modern Streaming Architecture on Hadoop

Casey Stella
@casey_stella

HORTONWORKS®

2017

Introduction

Hi, I'm Casey Stella!

Apache Metron: A Cybersecurity Analytics Platform

- Metron provides a scalable, advanced security analytics framework to offer a centralized tool for security monitoring and analysis.

Apache Metron: A Cybersecurity Analytics Platform

- Metron provides a scalable, advanced security analytics framework to offer a centralized tool for security monitoring and analysis.
- Metron was initiated at Cisco in 2014 as OpenSOC.

Apache Metron: A Cybersecurity Analytics Platform

- Metron provides a scalable, advanced security analytics framework to offer a centralized tool for security monitoring and analysis.
- Metron was initiated at Cisco in 2014 as OpenSOC.
- Metron was submitted to the Apache Incubator in December 2015

Apache Metron: A Cybersecurity Analytics Platform

- Metron provides a scalable, advanced security analytics framework to offer a centralized tool for security monitoring and analysis.
- Metron was initiated at Cisco in 2014 as OpenSOC.
- Metron was submitted to the Apache Incubator in December 2015
- Metron graduated to a top level project in April 2017

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
    - Kafka provides a unified data bus

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
  - Kafka provides a unified data bus
  - Storm providing a distributed streaming framework

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
    - Kafka provides a unified data bus
    - Storm providing a distributed streaming framework
    - HBase provides a low latency key/value lookup store for enrichments and profiles

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
  - Kafka provides a unified data bus
  - Storm providing a distributed streaming framework
  - HBase provides a low latency key/value lookup store for enrichments and profiles
  - Zookeeper provides a distributed configuration store

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
  - Kafka provides a unified data bus
  - Storm providing a distributed streaming framework
  - HBase provides a low latency key/value lookup store for enrichments and profiles
  - Zookeeper provides a distributed configuration store
- Ingested network telemetry can be enriched pluggably

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
  - Kafka provides a unified data bus
  - Storm providing a distributed streaming framework
  - HBase provides a low latency key/value lookup store for enrichments and profiles
  - Zookeeper provides a distributed configuration store
- Ingested network telemetry can be enriched pluggably
  - New enrichments can be done live on running topologies without restart

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
    - Kafka provides a unified data bus
    - Storm providing a distributed streaming framework
    - HBase provides a low latency key/value lookup store for enrichments and profiles
    - Zookeeper provides a distributed configuration store
- Ingested network telemetry can be enriched pluggably
    - New enrichments can be done live on running topologies without restart
    - New enrichment capabilities can be added via user defined functions

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
  - Kafka provides a unified data bus
  - Storm providing a distributed streaming framework
  - HBase provides a low latency key/value lookup store for enrichments and profiles
  - Zookeeper provides a distributed configuration store
- Ingested network telemetry can be enriched pluggably
  - New enrichments can be done live on running topologies without restart
  - New enrichment capabilities can be added via user defined functions
  - Enrichments can be composed through a domain specific language called **Stellar**

Characteristics of Metron

- Metron is built atop the Apache Hadoop ecosystem handle capturing, ingesting, enriching and storing streaming data at scale
  - Kafka provides a unified data bus
  - Storm providing a distributed streaming framework
  - HBase provides a low latency key/value lookup store for enrichments and profiles
  - Zookeeper provides a distributed configuration store
- Ingested network telemetry can be enriched pluggably
  - New enrichments can be done live on running topologies without restart
  - New enrichment capabilities can be added via user defined functions
  - Enrichments can be composed through a domain specific language called **Stellar**
- Data stored in HBase can be the source of enrichments

Characteristics of Metron

- Enriched telemetry can be indexed into a Security data lake
  - Indexes supported are pluggable and include HDFS, Solr and Elasticsearch
- Advanced analytics can be done on streaming data

Characteristics of Metron

- Enriched telemetry can be indexed into a Security data lake
    - Indexes supported are pluggable and include HDFS, Solr and Elasticsearch
- Advanced analytics can be done on streaming data
    - Probabalistic data structures (e.g. sketches) can sketch streaming data across time and enable approximate distribution, set existence and distinct count queries

Characteristics of Metron

- Enriched telemetry can be indexed into a Security data lake
    - Indexes supported are pluggable and include HDFS, Solr and Elasticsearch
- Advanced analytics can be done on streaming data
    - Probabalistic data structures (e.g. sketches) can sketch streaming data across time and enable approximate distribution, set existence and distinct count queries
    - Models can be deployed using Yarn, autodiscovered via Zookeeper and interrogated via Stellar functions

Stellar

Metron needed the ability to allow users to pluggably and consistently enrich and transform streaming data. Out of this need, we created **Stellar**:

Stellar

Metron needed the ability to allow users to pluggably and consistently enrich and transform streaming data. Out of this need, we created **Stellar**:

- Interact with the various enabling Hadoop components in a unified manner

Stellar

Metron needed the ability to allow users to pluggably and consistently enrich and transform streaming data. Out of this need, we created **Stellar**:

- Interact with the various enabling Hadoop components in a unified manner
- Compose a rich set of built-in functions with user defined functions

Stellar

Metron needed the ability to allow users to pluggably and consistently enrich and transform streaming data. Out of this need, we created **Stellar**:

- Interact with the various enabling Hadoop components in a unified manner
- Compose a rich set of built-in functions with user defined functions
- Provide simple primitives around the functions: boolean operations, conditionals, numerical computation.

Stellar

Metron needed the ability to allow users to pluggably and consistently enrich and transform streaming data. Out of this need, we created **Stellar**:

- Interact with the various enabling Hadoop components in a unified manner
- Compose a rich set of built-in functions with user defined functions
- Provide simple primitives around the functions: boolean operations, conditionals, numerical computation.

Think of Stellar as Excel functions that we can run on streaming data.

Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.

Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.
- Each telemetry source is ingested into kafka

Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.
- Each telemetry source is ingested into kafka
- A Storm **parser** topology is used to convert the raw telemetry format to a normalized JSON Map

Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.
- Each telemetry source is ingested into kafka
- A Storm **parser** topology is used to convert the raw telemetry format to a normalized JSON Map
    - Common network-related raw telemetry formats

Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.
- Each telemetry source is ingested into kafka
- A Storm **parser** topology is used to convert the raw telemetry format to a normalized JSON Map
  - Common network-related raw telemetry formats
  - Generic formats such as CSV and JSON

Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.
- Each telemetry source is ingested into kafka
- A Storm **parser** topology is used to convert the raw telemetry format to a normalized JSON Map
    - Common network-related raw telemetry formats
    - Generic formats such as CSV and JSON
    - Specifying the parser via Grok

Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.
- Each telemetry source is ingested into kafka
- A Storm **parser** topology is used to convert the raw telemetry format to a normalized JSON Map
  - Common network-related raw telemetry formats
  - Generic formats such as CSV and JSON
  - Specifying the parser via Grok
  - Creating your own parser in a JVM-based language

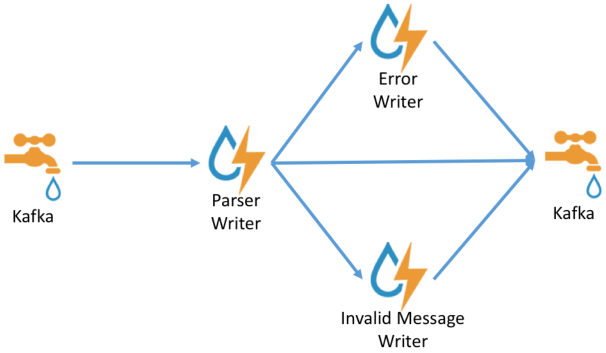
Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.
- Each telemetry source is ingested into kafka
- A Storm **parser** topology is used to convert the raw telemetry format to a normalized JSON Map
  - Common network-related raw telemetry formats
  - Generic formats such as CSV and JSON
  - Specifying the parser via Grok
  - Creating your own parser in a JVM-based language
- Transformations and normalizations can be done post-parse via Stellar statements

Data Ingest: Parsers

- Telemetry data comes in a variety of formats and velocities.
- Each telemetry source is ingested into kafka
- A Storm **parser** topology is used to convert the raw telemetry format to a normalized JSON Map
    - Common network-related raw telemetry formats
    - Generic formats such as CSV and JSON
    - Specifying the parser via Grok
    - Creating your own parser in a JVM-based language
- Transformations and normalizations can be done post-parse via Stellar statements
- The normalized data across all telemetries is written to an **enrichment** kafka topic

Data Ingest: Parsers



Enrichment

- The enrichment topology takes the various normalized telemetry sources and allows users to enrich the messages with broader context

Enrichment

- The enrichment topology takes the various normalized telemetry sources and allows users to enrich the messages with broader context
    - Enriching with reference data ingested into HBase

Enrichment

- The enrichment topology takes the various normalized telemetry sources and allows users to enrich the messages with broader context
  - Enriching with reference data ingested into HBase
  - Enriching via arbitrary Stellar expressions

Enrichment

- The enrichment topology takes the various normalized telemetry sources and allows users to enrich the messages with broader context
  - Enriching with reference data ingested into HBase
  - Enriching via arbitrary Stellar expressions
  - Enriching with Geolocation data

Enrichment

- The enrichment topology takes the various normalized telemetry sources and allows users to enrich the messages with broader context
  - Enriching with reference data ingested into HBase
  - Enriching via arbitrary Stellar expressions
  - Enriching with Geolocation data
- Enrichment is split into two phases

Enrichment

- The enrichment topology takes the various normalized telemetry sources and allows users to enrich the messages with broader context
  - Enriching with reference data ingested into HBase
  - Enriching via arbitrary Stellar expressions
  - Enriching with Geolocation data
- Enrichment is split into two phases
  - Preparatory Enrichment

Enrichment

- The enrichment topology takes the various normalized telemetry sources and allows users to enrich the messages with broader context
  - Enriching with reference data ingested into HBase
  - Enriching via arbitrary Stellar expressions
  - Enriching with Geolocation data
- Enrichment is split into two phases
  - Preparatory Enrichment
  - Threat Intelligence Enrichment with risk triage

Enrichment

- The enrichment topology takes the various normalized telemetry sources and allows users to enrich the messages with broader context
    - Enriching with reference data ingested into HBase
    - Enriching via arbitrary Stellar expressions
    - Enriching with Geolocation data
- Enrichment is split into two phases
    - Preparatory Enrichment
    - Threat Intelligence Enrichment with risk triage
- If messages are marked with an **is_alert** field, they can have a triage score computed via Stellar which defines their priority as threats

Enrichment: Stellar

Stellar is the primary method for enrichment in Metron.

- User defined enrichment functions can be enabled through adding a jar implementing the function to HDFS.

Enrichment: Stellar

Stellar is the primary method for enrichment in Metron.

- User defined enrichment functions can be enabled through adding a jar implementing the function to HDFS.
- Stellar enrichments can be executed asynchronously across storm workers and their results joined together

Enrichment: Stellar

Stellar is the primary method for enrichment in Metron.

- User defined enrichment functions can be enabled through adding a jar implementing the function to HDFS.
- Stellar enrichments can be executed asynchronously across storm workers and their results joined together

Stellar functions are provided for, among others,

- Interrogating ML models

Enrichment: Stellar

Stellar is the primary method for enrichment in Metron.

- User defined enrichment functions can be enabled through adding a jar implementing the function to HDFS.
- Stellar enrichments can be executed asynchronously across storm workers and their results joined together

Stellar functions are provided for, among others,

- Interrogating ML models
- Reading reference data from HBase

Enrichment: Stellar

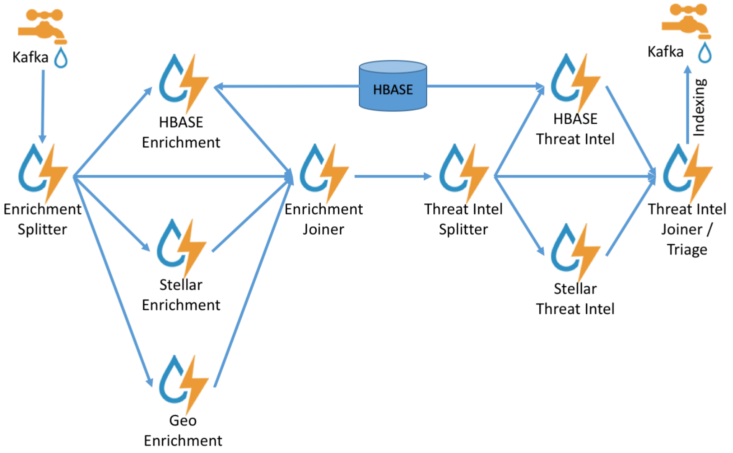
Stellar is the primary method for enrichment in Metron.

- User defined enrichment functions can be enabled through adding a jar implementing the function to HDFS.
- Stellar enrichments can be executed asynchronously across storm workers and their results joined together

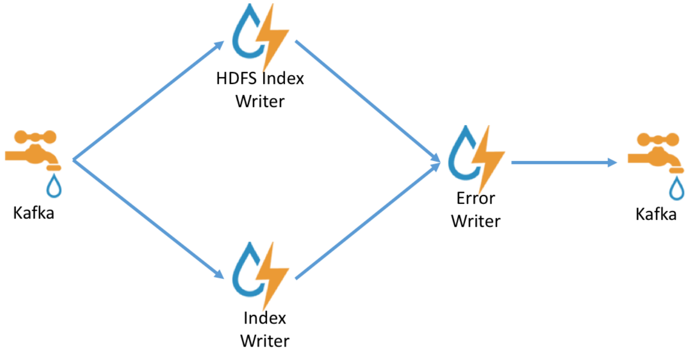
Stellar functions are provided for, among others,

- Interrogating ML models
- Reading reference data from HBase
- Reading historical profiles

Enrichment


Kafka
HBASE Enrichment
HBASE
HBASE Threat Intel
Kafka
Indexing
Enrichment Splitter
Enrichment Joiner
Threat Intel Splitter
Threat Intel Joiner / Triage
Stellar Enrichment
Stellar Threat Intel
Geo Enrichment

Indexing



Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.

Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.
- This is insufficient for a number of scenarios

Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.
- This is insufficient for a number of scenarios
  - Correlating between different sources

Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.
- This is insufficient for a number of scenarios
  - Correlating between different sources
  - Making judgments based on past events

Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.
- This is insufficient for a number of scenarios
  - Correlating between different sources
  - Making judgments based on past events
  - Both at the same time

Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.
- This is insufficient for a number of scenarios
  - Correlating between different sources
  - Making judgments based on past events
  - Both at the same time
- Operating across multiple sources has scalability implications

Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.
- This is insufficient for a number of scenarios
  - Correlating between different sources
  - Making judgments based on past events
  - Both at the same time
- Operating across multiple sources has scalability implications
- Waiting on the data you want from the other stream isn't plausible

Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.
- This is insufficient for a number of scenarios
    - Correlating between different sources
    - Making judgments based on past events
    - Both at the same time
- Operating across multiple sources has scalability implications
- Waiting on the data you want from the other stream isn't plausible
- Sometimes you want to change your mind about how much history you want to refer to

Profiler: Motivation

- Enrichments and parsers operate within the context of a single message.
- This is insufficient for a number of scenarios
    - Correlating between different sources
    - Making judgments based on past events
    - Both at the same time
- Operating across multiple sources has scalability implications
- Waiting on the data you want from the other stream isn't plausible
- Sometimes you want to change your mind about how much history you want to refer to
- In cyber security, advanced actors wait for months, so you need data months back!

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.
- The Profiler is a storm topology that takes the enriched data
    - Capture aggregations of data in a window to HBase

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.
- The Profiler is a storm topology that takes the enriched data
    - Capture aggregations of data in a window to HBase
    - Uses Stellar to define how to aggregate data

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.
- The Profiler is a storm topology that takes the enriched data
    - Capture aggregations of data in a window to HBase
    - Uses Stellar to define how to aggregate data
    - Uses Stellar to define a filter on which messages to consider

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.
- The Profiler is a storm topology that takes the enriched data
    - Capture aggregations of data in a window to HBase
    - Uses Stellar to define how to aggregate data
    - Uses Stellar to define a filter on which messages to consider
- These aggregations can be read anywhere Stellar is used

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.
- The Profiler is a storm topology that takes the enriched data
  - Capture aggregations of data in a window to HBase
  - Uses Stellar to define how to aggregate data
  - Uses Stellar to define a filter on which messages to consider
- These aggregations can be read anywhere Stellar is used
- This enables things like

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.
- The Profiler is a storm topology that takes the enriched data
    - Capture aggregations of data in a window to HBase
    - Uses Stellar to define how to aggregate data
    - Uses Stellar to define a filter on which messages to consider
- These aggregations can be read anywhere Stellar is used
- This enables things like
    - Temporal outlier detection

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.
- The Profiler is a storm topology that takes the enriched data
    - Capture aggregations of data in a window to HBase
    - Uses Stellar to define how to aggregate data
    - Uses Stellar to define a filter on which messages to consider
- These aggregations can be read anywhere Stellar is used
- This enables things like
    - Temporal outlier detection
    - Historical context from other sources when building threat triage rules

Profiler: Solution

- Compromise: Operate on windows in time rather than individual records
- Windows should be able to be specified very flexibly to avoid seasonal aberrations.
- The Profiler is a storm topology that takes the enriched data
  - Capture aggregations of data in a window to HBase
  - Uses Stellar to define how to aggregate data
  - Uses Stellar to define a filter on which messages to consider
- These aggregations can be read anywhere Stellar is used
- This enables things like
  - Temporal outlier detection
  - Historical context from other sources when building threat triage rules

Profiler: Aggregations

- Example aggregations:

Profiler: Aggregations

- Example aggregations:
    - Distributional statistics: median, mean, percentile

Profiler: Aggregations

- Example aggregations:
  - Distributional statistics: median, mean, percentile
  - Set operations: Contains, Cardinality

Profiler: Aggregations

- Example aggregations:
  - Distributional statistics: median, mean, percentile
  - Set operations: Contains, Cardinality
  - Simple counts

Profiler: Aggregations

- Example aggregations:
  - Distributional statistics: median, mean, percentile
  - Set operations: Contains, Cardinality
  - Simple counts
- Aggregations challenges

Profiler: Aggregations

- Example aggregations:
  - Distributional statistics: median, mean, percentile
  - Set operations: Contains, Cardinality
  - Simple counts
- Aggregations challenges
  - May be big objects if naively done (e.g. set operations)

Profiler: Aggregations

- Example aggregations:
  - Distributional statistics: median, mean, percentile
  - Set operations: Contains, Cardinality
  - Simple counts
- Aggregations challenges
  - May be big objects if naively done (e.g. set operations)
  - May not be able to be merged across time (e.g. distributions)

Profiler: Aggregations

- Example aggregations:
  - Distributional statistics: median, mean, percentile
  - Set operations: Contains, Cardinality
  - Simple counts
- Aggregations challenges
  - May be big objects if naively done (e.g. set operations)
  - May not be able to be merged across time (e.g. distributions)
  - Profile reading should be decoupled from writing

Profiler: Aggregations

- Example aggregations:
  - Distributional statistics: median, mean, percentile
  - Set operations: Contains, Cardinality
  - Simple counts
- Aggregations challenges
  - May be big objects if naively done (e.g. set operations)
  - May not be able to be merged across time (e.g. distributions)
  - Profile reading should be decoupled from writing
- Approach: Use approximate data structures
  - Set operations: Bloom Filters, HyperLogLog approximations

Profiler: Aggregations

- Example aggregations:
  - Distributional statistics: median, mean, percentile
  - Set operations: Contains, Cardinality
  - Simple counts
- Aggregations challenges
  - May be big objects if naively done (e.g. set operations)
  - May not be able to be merged across time (e.g. distributions)
  - Profile reading should be decoupled from writing
- Approach: Use approximate data structures
  - Set operations: Bloom Filters, HyperLogLog approximations
  - Distributional Statistics: t-digests

Demo

- Los Alamos National Lab released an open data set representing 58 consecutive days of de-identified event data collected from five sources within Los Alamos National Laboratory's corporate, internal computer network.
- Among other telemetry sources, authentication logs and a set of well-defined red teaming events that present bad behavior within the 58 days are provided.
- We will look at the authentication logs around a breach and show how we can use Metron to pick out offending user's activity leading up to the event
    - Look for users who are attempting to authenticate to many distinct hosts more than 5 standard deviations from the median across all users.

```
"profile": "attempts_by_user",
"foreach": "user",
"onlyif": "source.type == 'auth'",
"init" : {
  "total" : "HLLP_INIT(5,6)"
         },
"update": {
  "total" : "HLLP_ADD(total, ip_dst_addr)"
          },
"result" : {
   "profile" : "total",
   "triage" : {
       "total_count" : "HLLP_CARDINALITY(total)"
              }
}
```

```
"profile": "auth_distribution",
"foreach": "'global'",
"onlyif": "source.type == 'profiler' && profile == 'attempts_by_user'",
"init" : {
  "s" : "STATS_INIT()"
         },
"update": {
  "s" : "STATS_ADD(s, total_count)"
          },
"result": "s"
```

```
window := PROFILE_WINDOW('...')
profile := PROFILE_GET('attempts_by_user', user, window)
distinct_auth_attempts := HLLP_CARDINALITY(GET_LAST(profile))
distribution_profile := PROFILE_GET('auth_distribution', 'global', window)
stats := STATS_MERGE(distribution_profile)
distinct_auth_attempts_median := STATS_PERCENTILE(stats, 0.5)
distinct_auth_attempts_stddev := STATS_SD(stats)
```

Questions

Thanks for your attention! Don't forget to come to the cybersecurity Bird of a Feather session Thursday.

- Find me at http://caseystella.com
- Twitter handle: @casey_stella
- Email address: cstella@hortonworks.com