Elastic Stack for Security Monitoring in a Nutshell

2019 Pass the SALT Workshop
Overview

Introduction to Elastic Stack
Beats
Logstash
Elasticsearch
Kibana
Elastic Stack Alerting and Security
• This is an introductory workshop
• You probably won’t hear/see a lot of new things if you have:
  • Used Elastic Stack in the past;
  • Took the Elastic training...;
  • Followed SANS SEC455, SEC555, FOR572, etc.;
• If you are stuck, please do not suffer in silence!
• ais_workshop_xubuntu-18.04.2-desktop-amd64
• VMware Workstation, Player, or Fusion
  • You can try VirtualBox too, but you are on your own with that... sorry! 😊
• 8 GB RAM
• 30-50 GB disk space
• Keyboard layout: EN-US !!!
• Workshop VM (Ubuntu) user/pass: user / Workshop1234%
  • Normally, it should not require password for login and sudo
About David

• Managing partner at Alzette Information Security (@AlzetteInfoSec)
• Network penetration testing, security architectures, security monitoring, incident response
• Instructor at SANS Institute: FOR572
• BSides Luxembourg organizer https://bsideslux.lu
• Twitter: @DavidSzili
• E-mail: david.szili@alzetteinfosec.com
• Blog: http://jumpespjump.blogspot.com
About Eva

- Managing partner at Alzette Information Security (@AlzetteInfoSec)
- Web application penetration testing, source code review, security monitoring
- CyberWayFinder
- BSides Luxembourg organizer https://bsideslux.lu
- Twitter: @EvaSzilagyiSec
- E-mail: eva.szilagyi@alzetteinfosec.com
- Blog: http://jumpespjump.blogspot.com
Introduction to Elastic Stack

2019 Pass the SALT Workshop
What is Elastic Stack?
- 4 main components:
  - Elasticsearch
  - Logstash
  - Kibana
  - Beats
- And several other smaller components
  - Elastic Stack Features (X-Pack)
  - APM (Application Performance Monitoring)

Why Elastic Stack?
- (Free) Open Source Software
- Distributed, real-time search and analytics (very scalable)
- Parsing and data enrichment
- Large Community
- InfoSec Projects built around it:
  - Security Onion
  - Moloch (Elasticsearch)
  - SOF-ELK
  - SELKS
  - HELK
  - ROCK NSM
Elastic Stack History

Early 2000s: Shay Banon’s Recipe App

2012: Elasticsearch Inc.

2015: "Release Bonanza“, Beats, Elastic Cloud (AWS)

2016: Elastic Stack 5.0

2017: Elastic Cloud Enterprise (ECE)

2018: Open source X-Pack, New York Stock Exchange

2019: Core security features (TLS, RBAC) are free, SIEM, EndGame

Source: https://www.elastic.co/about/history-of-elasticsearch
Elastic Stack (Very) High-Level Overview

• **Beats**: single-purpose data shippers
• **Logstash**: server-side data processing pipeline
• **Elasticsearch**: distributed search and analytics engine
• **Kibana**: visualization and dashboards

See also: [https://www.elastic.co/assets/blt2614227bb99b9878/architecture-best-practices.pdf](https://www.elastic.co/assets/blt2614227bb99b9878/architecture-best-practices.pdf)
Beats

2019 Pass the SALT Workshop
Beats: Lightweight Data Shippers

• Lightweight log agents
• Written in Go
• Can send to Logstash or directly to Elasticsearch

Beats Family:
  • Filebeat
  • Winlogbeat
  • Auditbeat
  • Packetbeat
  • Heartbeat
  • Metricbeat
  • Functionbeat
  • Etc.

See also: https://www.elastic.co/guide/en/beats/libbeat/current/index.html
Beats Configuration Examples

Winlogbeat

```bash
#================================= Winlogbeat specific options
winlogbeat.event_logs:
- name: ForwardedEvents

#================================= Outputs ========

#================================= Logstash output -----
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.1.1:5044"]

#================================= Logging ========
logging.to_files: true
logging.files:
  path: D:/winlogbeat/Logs
logging.level: info
```

Filebeat

```bash
#================================= Filebeat inputs ==========================

filebeat.inputs:
  # Each - is an input. Most options can be set at the input level, so
  # you can use different inputs for various configurations.
  # Below are the input specific configurations.
  type: log
  # Change to true to enable this input configuration.
  enabled: true
  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /home/user/Elastic Stack Workshop/01_Beats/logs/*.log
  # Exclude lines. A list of regular expressions to match. It drops the lines
  # that are matching any regular expression from the list.
  exclude_lines: ['^#']

#================================= Outputs ========

#================================= Logstash output -----
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```
Logstash Overview

• LOTS AND LOTS of plugins!
  • **Input**: tcp, udp, syslog, beats, jdbc, kafka, rabbitmq, file, exec, cloudwatch, etc.
  • **Filter**: csv, json, xml, kv, grok, date, mutate, split, useragent, ruby, drop, etc.
  • **Output**: elasticsearch, graphite, nagios, kafka, rabbitmq, radis, file, email, irc, etc.

• Easy to learn and use

Input Plugin Examples

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>beats</td>
<td>Events from Elastic Beats</td>
</tr>
<tr>
<td>cloudwatch</td>
<td>Events from AWS CloudWatch</td>
</tr>
<tr>
<td>file</td>
<td>Streams events from files</td>
</tr>
<tr>
<td>jdbc</td>
<td>Events from JDBC data</td>
</tr>
<tr>
<td>kafka</td>
<td>Reads events from Kafka</td>
</tr>
<tr>
<td>rabbitmq</td>
<td>Pulls events from RabbitMQ</td>
</tr>
<tr>
<td>s3</td>
<td>Events from files in S3</td>
</tr>
<tr>
<td>snmp</td>
<td>Polls devices using SNMP</td>
</tr>
<tr>
<td>syslog</td>
<td>Reads syslog messages</td>
</tr>
</tbody>
</table>

See also: [https://www.elastic.co/guide/en/logstash/current/input-plugins.html](https://www.elastic.co/guide/en/logstash/current/input-plugins.html)
### Filter Plugin Examples

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cidr</td>
<td>Check IP against net blocks</td>
</tr>
<tr>
<td>csv</td>
<td>Parses CSV data into fields</td>
</tr>
<tr>
<td>date</td>
<td>Parses dates from fields</td>
</tr>
<tr>
<td>dissect</td>
<td>Extracts unstructured data</td>
</tr>
<tr>
<td>drop</td>
<td>Drops all events</td>
</tr>
<tr>
<td>elasticsearch</td>
<td>Gets data from Elasticsearch</td>
</tr>
<tr>
<td>geopip</td>
<td>Geo info about an IP</td>
</tr>
<tr>
<td>grok</td>
<td>Parses unstructured data</td>
</tr>
<tr>
<td>json</td>
<td>Parses JSON data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kv</td>
<td>Parses key-value pairs</td>
</tr>
<tr>
<td>mutate</td>
<td>Performs mutations on fields</td>
</tr>
<tr>
<td>ruby</td>
<td>Executes Ruby code</td>
</tr>
<tr>
<td>split</td>
<td>Splits multi-line messages</td>
</tr>
<tr>
<td>translate</td>
<td>Replaces field contents</td>
</tr>
<tr>
<td>truncate</td>
<td>Truncates fields</td>
</tr>
<tr>
<td>urldecode</td>
<td>Decodes URL-encoded fields</td>
</tr>
<tr>
<td>useragent</td>
<td>Parses user agent strings</td>
</tr>
<tr>
<td>xml</td>
<td>Parses XML data</td>
</tr>
</tbody>
</table>

Filters - The Easy Stuff

**JSON**

```json
filter {
  ...
  json {
    source => "message"
  }
  ...
  mutate {
    remove_field => [ "message" ]
  }
}
```

**CSV**

```json
filter {
  ...
  csv {
    separator => " "
  }
  ...
  mutate {
    remove_field => [ "message" ]
  }
}
```
Filters - RegExp vs. Grok, Dissect (1)

**RegExp**

- (?<!\[0-9\])(?:(?::25[0-5]|2[0-4]\[0-9\]|\[0-1\]??[0-9]{1,2})[.](?:25[0-5]|2[0-4]\[0-9\]|\[0-1\]??[0-9]{1,2})[.](?:25[0-5]|2[0-4]\[0-9\]|\[0-1\]??[0-9]{1,2})[.](?:25[0-5]|2[0-4]\[0-9\]|\[0-1\]??[0-9]{1,2}))(?!\[0-9\])

**Dissect**

- String-based split operation
- Very fast

**Grok**

- %{IPV4:source_ip}
- Pre-cooked RegExp patterns
- Custom Patterns:
  - (?<queue_id>[0-9A-F]{10,11})

**Grok Debuggers:**

- Heroku App: [http://grokdebug.herokuapp.com](http://grokdebug.herokuapp.com)
- Source: [https://github.com/nickethier/grokdebug](https://github.com/nickethier/grokdebug)
- Docker: [https://hub.docker.com/r/fdrouet/grokdebug](https://hub.docker.com/r/fdrouet/grokdebug)
- Kibana / Dev Tools / Grok Debugger
Filters - RegExp vs. Grok, Dissect (2)

dissect

```json
filter {
    ...,
    dissect {
        mapping => {
            "message" => "%{ts} %{+ts} %{+ts} %{src} %{prog}[ %{pid}]: %{msg}"  
        }
    }
    ...,
}
```

grok

```json
filter {
    ...,
    grok {
        match => {
            "message" => "%{SYSLOGTIMESTAMP: syslog_timestamp} %{SYSLOGHOST: syslog_hostname} %{DATA: syslog_program}(?:\[%{POSINT: syslog_pid}\])?: %{GREEDYDATA: syslog_message}"
            
            "message" => "%{SYSLOGBASE2} %{GREEDYDATA: message}"
        }
    }
    ...,
}
```
Filters - Enrichment Examples

```ruby
filter {
  if [program] == "bro_dns" {
    ruby {
      code => "event.set('query_length', event.get('query').length)"
    }
  }
  ...
}
```

```geoip
filter {
  if [resp_h_routable] == "true" {
    geoip {
      source => "id.resp_h"
      target => "geoip"
      default_database_type => "City"
    }
    geoip {
      source => "id.resp_h"
      target => "geoip"
      default_database_type => "ASN"
    }
  }
  ...
}
```
## Output Plugin Examples

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csv</td>
<td>Writes events to disk in CSV</td>
</tr>
<tr>
<td>elasticsearch</td>
<td>Stores logs in Elasticsearch</td>
</tr>
<tr>
<td>email</td>
<td>Sends email to an address</td>
</tr>
<tr>
<td>exec</td>
<td>Runs a command</td>
</tr>
<tr>
<td>file</td>
<td>Writes events to files</td>
</tr>
<tr>
<td>graphite</td>
<td>Writes metrics to Graphite</td>
</tr>
<tr>
<td>kafka</td>
<td>Writes events to Kafka</td>
</tr>
<tr>
<td>rabbitmq</td>
<td>Pushes events to RabbitMQ</td>
</tr>
<tr>
<td>redis</td>
<td>Sends events to Redis</td>
</tr>
</tbody>
</table>

```ruby
class output {
  elasticsearch {
    hosts => 
      ["localhost:9200"]
  }
}
```

```ruby
class output {
  stdout {
    codec => rubydebug
  }
}
```

Elastic Common Schema (ECS)

- Specification that provides a consistent and customizable way to structure your data in Elasticsearch
  - Searches can be created more narrowly
  - Field names are easier to remember
- ECS GitHub: [https://github.com/elastic/ecs](https://github.com/elastic/ecs)

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECS Core Fields</td>
<td>Fully defined set of field names that exists under a defined set of ECS top-level objects</td>
</tr>
<tr>
<td>ECS Extended Fields</td>
<td>Partially defined set of field names that exists under the same set of ECS top-level objects</td>
</tr>
<tr>
<td>Custom Fields</td>
<td>Undefined and unnamed set of fields that exists under a user-supplied set of non-ECS top-level objects that must not conflict with ECS fields or objects</td>
</tr>
</tbody>
</table>
Elasticsearch Overview

- Storage and Search
- Built on Apache Lucene
  - “wrapper” written in Java
- REST API
- JSON over HTTP
- Distributed
- Real-time

More info: John Hubbard - The Elastic Stack as a SIEM: https://www.youtube.com/watch?v=v69kyU5XMFI
Elasticsearch Terms

- **Cluster**: All nodes
- **Node**: Elasticsearch instance
- **Index**: Set of documents (group of shards)
- **Shard**: 
  - Subset of documents in an index
  - Apache Lucene instance
  - Primary (like RAID 0) and Replica (like RAID 1)
- **Document**: JSON object in Elasticsearch
Elasticsearch vs. Relational Database

- **Mapping:**
  - Defines field names and datatypes in documents
  - Can add new fields, but **existing fields cannot be changed!**

- **Field:**
  - Key-value pair in a document
  - Metadata like: _index, _id, etc.

- **WORM (Write Once Read Many) vs. ACID (Atomicity, Consistency, Isolation, Durability)**

<table>
<thead>
<tr>
<th>Elasticsearch</th>
<th>Relational Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>Database</td>
</tr>
<tr>
<td>Mapping</td>
<td>Schema</td>
</tr>
<tr>
<td>Document</td>
<td>Row</td>
</tr>
<tr>
<td>Field</td>
<td>Column</td>
</tr>
</tbody>
</table>

```json
{
  "PWD" => "/home/user",
  "syslog_timestamp" => "Mar 17 15:29:49",
  "USER" => "root",
  "syslog_program" => "sudo",
  "@timestamp" => "2019-03-17T14:29:49.000Z",
  "COMMAND" => "/usr/bin/docker pull broplatform/bro:2.6",
  "TTY" => "pts/0",
  "@version" => "1",
  "syslog_pid" => "1931",
  "host" => "ws-vm",
  "syslog_hostname" => "ws-vm"
}```
### Data Types (Few Examples)

**Core**
- *text*
- *keyword*
- long, *integer*, short, byte
- double, float, half_float, scaled_float
- *boolean*
- binary

**Geo**
- *geo_point*
- geo_shape

**Specialized**
- *date*
- *ip*

**Complex**
- *array*
- *object*
- *nested*

**Multi-fields**
- Indexed as more one type

Etc.

Text vs. Keyword

**Text type**
- “Full-text value”
- Payload, message, etc.
- Analyzed and **tokenized**
- Cannot be used for
  - Sorting
  - Aggregations

**Keyword type**
- “Exact value”
- IP, port, protocol, user, etc.
- Exact match / not match
- Can be used for
  - Sorting
  - Aggregations
Kibana

2019 Pass the SALT Workshop
Kibana Overview

- Web-based analytic interface
- Searches
  - Apache Lucene syntax
- Filters
- Visualizations, Dashboards
  - Stored in JSON
- Plugins
  - Reporting, Alerting, etc.
Kibana Features

- **Discover**: Search
- **Visualize**: Graphs, charts
  - Vega, Vega-Lite
- **Dashboard**: Visualizations and saved searches
- **Timelion**: Time series visualizations
- **Canvas**: Presentation
- **Machine Learning** (Paid)
- **Graph** (Paid)
- **Infrastructure**: Metricbeats monitoring
- **Logs**: Filebeat monitoring
- **APM**: Application Performance Monitoring
- **Uptime**: Monitor the status of network endpoints
- **SIEM**: Interactive workspace for security investigations
- **Dev Tools**: API access
- **Monitoring**: Cluster health
- **Management**: Cluster management
- etc.
Index Patterns

- Must choose an index pattern
  - Discovery (Searches)
  - Visualization
- Limits the indices searched
- Relates to index naming scheme
- Can use the * wildcard
  - "logstash-*"

Steps:
1. Create Elasticsearch index
2. “Create index pattern”
3. Select index/indices
4. Define @timestamp field
## Search - Apache Lucene Query Syntax (1)

<table>
<thead>
<tr>
<th>Search Type</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Term</td>
<td><code>&lt;term&gt;</code></td>
<td>hello</td>
</tr>
<tr>
<td>Phrase</td>
<td>“&lt;term&gt;”</td>
<td>“hello world”</td>
</tr>
<tr>
<td>Fields</td>
<td><code>&lt;field&gt;:&lt;term&gt;</code></td>
<td>title:hello</td>
</tr>
<tr>
<td>AND</td>
<td><code>&lt;term-a&gt; AND &lt;term-b&gt;</code></td>
<td>hello AND world // hello world</td>
</tr>
<tr>
<td>OR</td>
<td><code>&lt;term-a&gt; OR &lt;term-b&gt;</code></td>
<td>hello OR world</td>
</tr>
<tr>
<td>NOT</td>
<td>NOT <code>&lt;term-a&gt;</code></td>
<td>NOT “hello world”</td>
</tr>
<tr>
<td></td>
<td>!&lt;term-a&gt;</td>
<td>!“hello world”</td>
</tr>
<tr>
<td>Must match</td>
<td>+&lt;term&gt;</td>
<td>+hello</td>
</tr>
<tr>
<td>Must not match</td>
<td>-&lt;term&gt;</td>
<td>-hello</td>
</tr>
</tbody>
</table>
## Search - Apache Lucene Query Syntax (2)

<table>
<thead>
<tr>
<th>Search Type</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field exists</td>
<td><code>_exists_:&lt;field&gt;</code></td>
<td><code>_exists_:title</code></td>
</tr>
<tr>
<td>Field does not exists</td>
<td>NOT <code>_exists_:&lt;field&gt;</code></td>
<td>NOT <code>_exists_:title</code></td>
</tr>
<tr>
<td></td>
<td>! <code>_exists_:&lt;field&gt;</code></td>
<td>! <code>_exists_:title</code></td>
</tr>
<tr>
<td></td>
<td><code>-exists_:&lt;field&gt;</code></td>
<td><code>-exists_:title</code></td>
</tr>
<tr>
<td>Wildcard search</td>
<td>?, *</td>
<td>h?llo, hell*</td>
</tr>
<tr>
<td>Fuzzy search</td>
<td><code>&lt;term&gt;~[&lt;number&gt;]</code></td>
<td>hello-2</td>
</tr>
<tr>
<td>Proximity search</td>
<td>“&lt;term&gt;”~[&lt;number&gt;]</td>
<td>“hello world”~5</td>
</tr>
<tr>
<td>Range</td>
<td><code>&lt;field&gt;:[&lt;value-a&gt; TO &lt;value-b&gt;]</code></td>
<td>port:[1 TO 1024]</td>
</tr>
<tr>
<td></td>
<td><code>&lt;field&gt;:{&lt;value-a&gt; TO &lt;value-b}&gt;</code></td>
<td>title:{hello TO world}</td>
</tr>
</tbody>
</table>
Search vs. Filters And Time Range

- **Search**: Using the Query bar and the Apache Lucene Query Syntax
- **Filter**: Using the Filters Box and the Elasticsearch Query DSL (Domain Specific Language)
## Visualizations

<table>
<thead>
<tr>
<th>Visualization</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>Basic Charts</td>
</tr>
<tr>
<td>Heat Map</td>
<td>Basic Charts</td>
</tr>
<tr>
<td>Horizontal Bar</td>
<td>Basic Charts</td>
</tr>
<tr>
<td>Line</td>
<td>Basic Charts</td>
</tr>
<tr>
<td>Pie</td>
<td>Basic Charts</td>
</tr>
<tr>
<td>Vertical Bar</td>
<td>Basic Charts</td>
</tr>
<tr>
<td>Data Table</td>
<td>Data</td>
</tr>
<tr>
<td>Gauge</td>
<td>Data</td>
</tr>
<tr>
<td>Goal</td>
<td>Data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visualization</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>Data</td>
</tr>
<tr>
<td>Coordinate Map</td>
<td>Maps</td>
</tr>
<tr>
<td>Region Map</td>
<td>Maps</td>
</tr>
<tr>
<td>Timelion</td>
<td>Time Series</td>
</tr>
<tr>
<td>Visual Builder (E)</td>
<td>Time Series</td>
</tr>
<tr>
<td>Controls (E)</td>
<td>Other</td>
</tr>
<tr>
<td>Markdown</td>
<td>Other</td>
</tr>
<tr>
<td>Tag Cloud</td>
<td>Other</td>
</tr>
<tr>
<td>Vega (E)</td>
<td>Other</td>
</tr>
</tbody>
</table>
Visualizations use Elasticsearch Aggregations

**Metrics: value to calculate**
- Count
- Average
- Sum
- Min
- Max
- Unique Count

**Bucket: aggregation (grouping)**
- Standard Deviation
- Top Hit
- Percentiles
- etc.
- Date Histogram (by time)
- Date Range
- Filter
- Geo Distance
- IP Range
- Range
- Sampler
- Significant Text
- **Terms** (by field)
- etc.
Vega and VegaLite

- Vega Graphs
  - Visualization grammar
  - Declarative language
  - JSON format
- Supported from Elastic 6
- Vega vs VegaLite
  - VegaLite: simplified Vega
  - [https://vega.github.io/vega/](https://vega.github.io/vega/)
  - [https://vega.github.io/vega-lite/](https://vega.github.io/vega-lite/)

Based on: [https://www.elastic.co/blog/sankey-visualization-with-vega-in-kibana](https://www.elastic.co/blog/sankey-visualization-with-vega-in-kibana)
Digital Marketing

Marketing Funnel
- 120,462 Website Visitors
- 20,643 Emails Registered
- 546 Campaign Responses
- 123 Conversions

Conversion Ratio: 1.03%
Bounce Rate: 49.4%
Email Campaigns: 24
Newsletter Open Rate: 53.8%

Campaign Responses
- Weekly Visitors by Day

Subscribers by Platform
- YouTube
- Twitter
- Instagram
- Facebook

Previous year: [Data]
This year: [Data]
Elastic SIEM

Source: https://www.elastic.co/blog/introducing-elastic-siem
Kibana Hands-On Scenario

syslog-ng

logstash ➔ Elasticsearch ➔ Kibana

ALZETTE
INFORMATION SECURITY
Elastic Stack Alerting and Security

2019 Pass the SALT Workshop
Watcher vs. ElastAlert

**Watcher**
- Part of X-Pack
- Elasticsearch API
- JSON format
- **Watches**: Triggers, Inputs, Conditions, Transforms, Actions

**ElastAlert**
- Developed by Yelp
- [https://github.com/Yelp/elastalert](https://github.com/Yelp/elastalert)
- Simple framework for alerting
- YAML format
- **Components**: Rules and Alerts
ElastAlert Overview

1. Elasticsearch is periodically queried
2. Data is passed to the rules
3. When a match occurs, one or more alerts are triggered
4. Alerts take action based on the match

• **Rule types**: Any, Blacklist, Whitelist, Change, Frequency, Spike, Flatline, New Term, Cardinality, Metric Aggregation, Percentage Match

• **Alert types**: Command, Email, JIRA, ServiceNow, Slack, PagerDuty, GoogleChat, Mattermost, Telegram, etc.

• [https://elastalert.readthedocs.io](https://elastalert.readthedocs.io)
ElastAlert Examples

```python
es_host: localhost
es_port: 9200

ame: Example frequency rule
type: frequency
index: logstash-*
num_events: 3
timeframe: hours: 1

filter:
- term:
  program: "bro_http"
- term:
  user_agent: "jupdate"

alert: "email"
email: "workshop@example.com"
```

```python
es_host: localhost
es_port: 9200

name: Example new term rule
type: new_term
index: logstash-*
fields: "user_agent"
terms_window_size:
  days: 90

filter:
- term:
  program: "bro_http"
- term:
  id_orig_h: "192.168.1.105"

alert: "email"
email: "workshop@example.com"
```
ElastAlert Hands-On

2019 Pass the SALT Workshop
Security

- **Elastic Stack Security**: [https://www.elastic.co/products/stack/security](https://www.elastic.co/products/stack/security)
  - Part of Elastic Stack Features (formerly X-Pack)
  - “Starting in version 6.8 and 7.1, core security features like TLS, file and native realm authentication, and role-based access control are now free.”

- **ReadonlyREST**: [https://readonlyrest.com](https://readonlyrest.com)
  - 3rd party
  - Free community version

- **Search Guard**: [https://search-guard.com](https://search-guard.com)
  - 3rd party
  - Free community version

- **NGINX reverse proxy + Basic Auth**: [https://www.nginx.com](https://www.nginx.com)
  - No RBAC at all
References

• Elastic Website
  • [https://www.elastic.co](https://www.elastic.co)

• Elastic Documentation
  • [https://www.elastic.co/guide/index.html](https://www.elastic.co/guide/index.html)

• John Hubbard - The Elastic Stack as a SIEM
  • [https://www.youtube.com/watch?v=v69kyU5XMFI](https://www.youtube.com/watch?v=v69kyU5XMFI)

• ElastAlert
  • [https://github.com/Yelp/elastalert](https://github.com/Yelp/elastalert)