ELK stack Documentation

AUTHOR: Blueliv

DATE: February 15, 2017

Contents

1 Introduction **2**

1.1 The ELK stack 2

1.2 Logstash Plugins 2

1.3 Blueliv on ELK 3

2 Installation **4**

2.1 Requirements 4

2.2 Blueliv Plugin 4

2.3 Configuration 4

2.3.1 Example 6

3 Getting Started **8**

3.1 Logstash Configuration 8

3.2 Elasticsearch Configuration 9

3.3 Kibana Configuration 12

3.4 Indexes 12

3.5 Charts 13

3.6 Dashboards 14

3.6.1 Example Dashboards 14

4 Registration **18**

Introduction

1.1 The ELK stack

The ELK stack contains **E**lasticsearch, **L**ogstash, and **K**ibana. Although they have all been built to work exceptionally well together, each one is a separate open source project that is driven by Elasticsearch. The role of each of these technologies is divided as follows:

Logstash Tool for managing events and logs. You can use it to collect logs, parse them, and store them for later use, such as searching. You can collect several different types of data from different sources (logs, HTTP, TCP, Kafka, etc) and then filter or transform them.

Elasticsearch Search server based on Lucene. It provides a distributed, full-text search engine with a RESTful web interface and schema-free JSON documents. You can use it to index and aggregate data from Logstash.

Kibana Browser based analytics and search dashboard for Elasticsearch. You can set-up dashboards with graphics and tables to visualize your indexed and aggregated data from Elasticsearch.

1.2 Logstash Plugins

Since version 1.5, there is a new plugin management system for Logstash. Plugin repositories are now located at Github. Plugins are now self-contained Ruby gems hosted in RubyGems. Logstash comes with a script called `plugin`, which is used to install the extra plugins. All plugins can be found listed and searchable at Github. There are three kinds of Logstash plugin:

Input To provide input data into Logstash.

Filter To filter data from a given input.

Output To save the filtered data anywhere you want, such as Elasticsearch, for indexing and searching data.

1.3 Blueliv on ELK

Blueliv automatically integrates Blueliv's Cyber Threat Intelligence into ELK stack, providing an *input* plugin for Logstash. The use of this app will add Cyber Threat Intelligence to your existing data, addressing a comprehensive range of cyber threats including compromised URLs, domains, IPs, etc. to turn global threat data into predictive, actionable intelligence specifically for your enterprise and the unique threats it faces.

Our powerful networks of specialized search engines constantly scour the web for up-to-the- minute data and delivers real-time actionable information.

Unsurpassed cyber threat intelligence, now at your disposal.

Installation

2.1 Requirements

This plugin was developed and tested against Logstash 1.5.0, so it supports this and newer versions. Older versions are **not** supported due to changes in Logstash plugin API. As of Elasticsearch and Kibana, you must use Elasticsearch version $\geq$ 2.4.0 and Kibana $\geq$ 4.6.0 in order to work with Elasticsearch 2.4.x.

2.2 Blueliv Plugin

Blueliv provides an *input* plugin, delivering data about active Crime Servers and Bot IPs. To install it, please run the following command in the directory where you installed Logstash:

```
# For logstash versions previous to 2.3.0
bin/plugin install --version 1.0.1 logstash-input-blueliv

# In logstash versions >= 2.3.0
bin/logstash-plugin install logstash-input-blueliv
```

2.3 Configuration

Before using our plugin, you should configure your API Key (get yours here) and the feeds that you want to receive. The plugin has the following configuration parameters:

api_url the URL of the API. By default its value is `https://api.blueliv.com`, but you should change it if you are using our Free API (`https://freeapi.blueliv.com`).

api_key API key to access our feeds. This parameter is **mandatory** in order to get the plugin working. You can get your Free API key here or contact us via email in order to get a commercial account.

http_timeout HTTP timeout for each API call. Default: `500` seconds.

feeds it is a Ruby hash that specifies the parameters to access each one of our feeds. Each feed may be configured with the following properties:

active if the feed is active or not. Default: `false`.

feed_type the type of the feed that you want. The available types are the following:

Attacks This feed has `recent` and `last`. `Recent` is executed every 3 hours and `last` is executed every 30 minutes. See https://apidocs.blueliv.com/#attacking-ips for more information.

Bot IPs This feed has `pos`, `non_pos`, `full` and `test`. `Pos` the ones that are Point-of-Sale, `non_pos` to retrieve the ones are not a Point-of-Sale, `full` to retrieve all and `test` returns a small test data set. See https://apidocs.blueliv.com/#bot-ip for more information.

Crime Servers This feed has `last`, `recent` and `test`. `Last` is executed every 15 minutes, `recent` every hour and `test` returns a small test data set. See https://apidocs.blueliv.com/#crimeservers for more information.

Malwares This feed has `last` and `recent`. `Last` is executed every 10 minutes and `recent` is executed every hour See https://apidocs.blueliv.com/#malware for more information.

interval interval of polling data from our API. Is not recommended use this configuration because default values are optimized for our endpoints.

The default configuration for `feeds` parameter is available at listing 2.1.

```
{
  "attacks" => {
    "active" => false,
    "feed_type" => "recent",
    "interval" => 600
  },
  "botips" => {
    "active" => false,
    "feed_type" => "test",
    "interval" => 600
  },
  "crimeservers" => {
    "active" => true,
    "feed_type" => "test",
    "interval" => 900
  },
  "malwares" => {
    "active" => false,
    "feed_type" => "recent",
    "interval" => 3600
  }
}
```

Listing 2.1: Default `feeds` configuration

2.3.1 Example

An example of an input configuration is shown in listing 2.2.

```
input {
  blueliv {
    api_key => "<YOUR API KEY>"
    feeds => {
      "attacks" => {
        "active" => true
      }
      "botips" => {
        "active" => true
        "feed_type" => "non_pos"
      }
      "crimeservers" => {
        "active" => true
        "feed_type" => "recent"
      }
      "malwares" => {
        "active" => true
      }
    }
  }
}
```

Listing 2.2: Plugin configuration example

Be aware that if you do not specify a given field, the default value will be configured. In this case, we did not touch the `interval` field for the feeds, so the defaults will apply.

Getting Started

In this chapter we are going to explain how to set-up properly a full ELK stack in order to take advantage of the cyber-threat intelligence provided by Blueliv. Please be aware that his is a configuration suggestion and you may change it to fit your needs. You may skip some of the steps if you already have configured your ELK stack.

3.1 Logstash Configuration

Our Logstash configuration is divided in two main components, as shown in listing 3.1.

input where you must set-up your API key and cyber-threat feeds that you want

output where we set-up the way we want to set-up the *output*. Our configurations go as following:

- **index** where we are indexing each feed in a different Elasticsearch *index*, i.e., `blueliv-attacks`, `blueliv-crimeservers`, `blueliv-ips` and `blueliv-malwares`.
- **manage_template** where we tell Logstash that we will not manage Elasticsearch *template* for our *indexes*. We explain more on this in 3.2.
- **document_id** we provide a `document_id` for Elasticsearch *indexes*, so when a object - such as a Crime Server - is updated, it will update its entry.

```
input {
  blueliv {
    api_key => "<YOUR API KEY>"
    feeds => {
      "attacks" => {
        "active" => false,
        "feed_type" => "recent",
      },
      "botips" => {
        "active" => true
        "feed_type" => "non_pos"
      }
      "crimeservers" => {
        "active" => true
        "feed_type" => "recent"
      }
      "malwares" => {
        "active" => false,
        "feed_type" => "recent",
      }
    }
  }
}
output {
  elasticsearch {
    index => "blueliv-%{@collection}"
    manage_template => false
    document_id => "%{document_id}"
  }
}
```

Listing 3.1: Full Logstash configuration example

You can find these configurations at our GitHub. If you are are using our Free API, you should use these configurations.

Please note that the first time you run this plugin in Logstash it may take some time to download and index all the items.

3.2 Elasticsearch Configuration

The only configuration that you need to perform in Elasticsearch is to upload Blueliv's index template, shown in listing 3.2. In order to upload it you should make a REST call to the following Elasticsearch end-point. You may use `curl` to do it. If you save our template (3.2) as `template.json`, you could just run the following command:

```
> curl -XPUT localhost:9200/_template/blueliv -d @template.json
```

```
{
    "template": "blueliv-*",
"settings" : {
    "index.refresh_interval": "5s"
},
"mappings": {
    "_default_": {
        "dynamic_templates": [
            {
                "message_field": {
                  "mapping": {
                    "index": "analyzed",
                    "omit_norms": true,
                    "type": "string"
                  },
                  "match_mapping_type": "string",
                  "match": "message"
                }
              },
              {
                "string_fields": {
                  "mapping": {
                    "index": "analyzed",
                    "omit_norms": true,
                    "type": "string",
                    "fields": {
                      "raw": {
                        "ignore_above": 256,
                        "index": "not_analyzed",
                        "type": "string"
                      }
                    }
                  },
                  "match_mapping_type": "string",
                  "match": "*"
                }
              }
         ],
         "_all": {
             "omit_norms": true,
             "enabled": true
         },
         "properties" : {
           "location": {
                "type": "geo_point"
            },
            "lastSeenAt": {
```

```
          "type": "date",
          "format": "yyyy-MM-dd'T'HH:mm:ssZ"
        },
        "firstSeenAt": {
          "type": "date",
          "format": "yyyy-MM-dd'T'HH:mm:ssZ"
        },
        "updatedAt": {
          "type": "date",
          "format": "yyyy-MM-dd'T'HH:mm:ssZ"
        },
        "@version": {
            "index": "not_analyzed",
            "type": "string"
        }
    }
},
"blueliv-attacks": {
    "properties": {
        "firstEvent": {
            "type": "date",
            "format": "yyyy-MM-dd'T'HH:mm:ssZ"
        },
        "lastEvent": {
            "type": "date",
            "format": "yyyy-MM-dd'T'HH:mm:ssZ"
        },
        "source": {
            "type": "object",
            "properties": {
                "ip": {
                  "type": "ip"
                }
            }
        }
    }
},
"blueliv-malwares": {
    "properties": {
        "analyzedAt": {
            "type": "date",
            "format": "yyyy-MM-dd'T'HH:mm:ssZ"
        }
    }
}
```

```
    }
}
```

Listing 3.2: Full Logstash configuration example

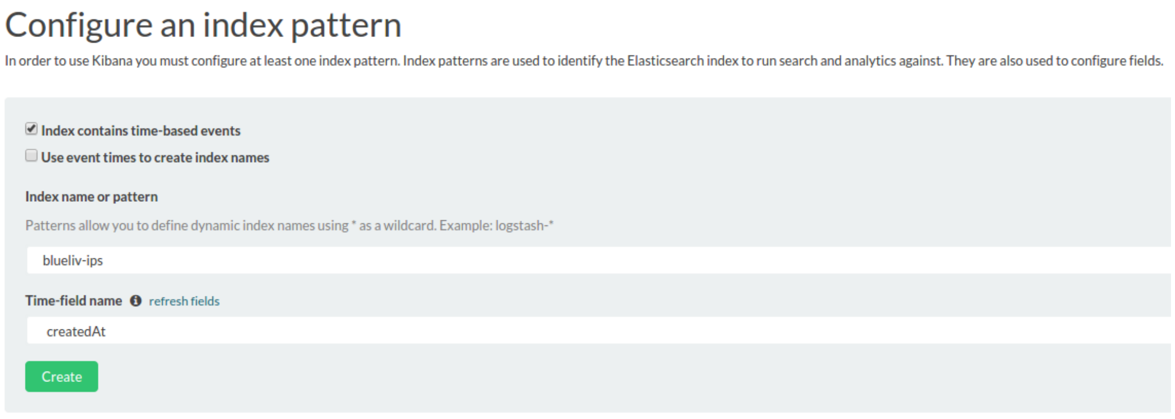
You can find these configurations at our GitHub.

3.3 Kibana Configuration

In this section we will show you how to set-up your Elasticsearch indexes (3.4), create charts and data tables (3.5) and put it all together in a dashboard (3.6). In section 3.6.1 you can find the example dashboards that we have set-up.

3.4 Indexes

To access your Kibana installation you should browse to the corresponding address (by default: `http://localhost:5601`). Then you must set-up the indexes that you want to get data of. As shown in figure 3.1.



Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

- [✓] Index contains time-based events
- [ ] Use event times to create index names

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

blueliv-ips

Time-field name refresh fields

createdAt

Figure 3.1: Configuring an index in Kibana.

As you may noticed, you can configure an index (e.g., `blueliv-ips`) or a pattern that has all the indexes you want (e.g., `blueliv-*`). You must select a Time-field, which will be the default field used to filter time-based events. We recommend use the followings for every case:

blueliv-attacks use field `lastEvent`

blueliv-crimeservers use field `createdAt` or lastSeenAt

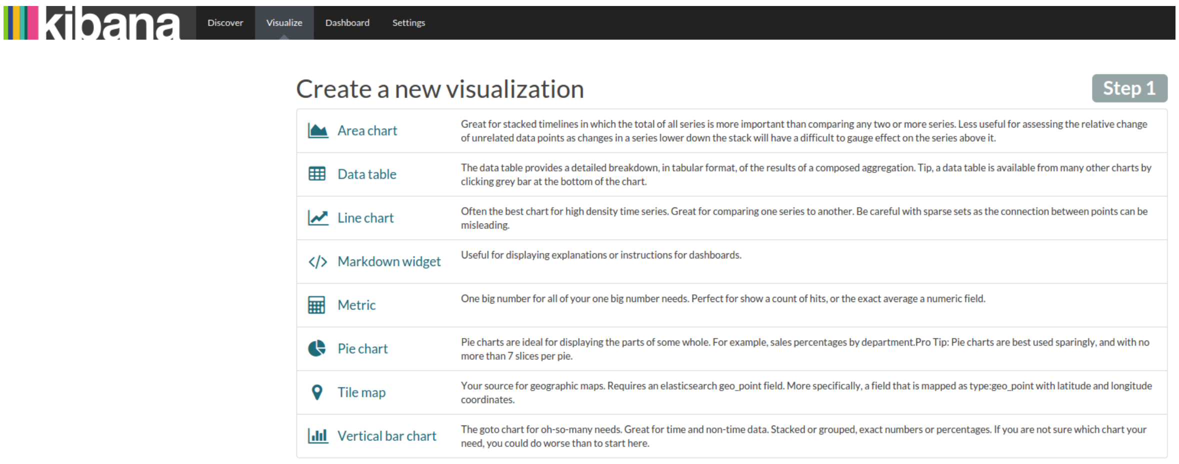
blueliv-ips (Bot IPs feed) use field `createdAt`

blueliv-malwares use field `lastEvent`

In any case, you can see more information about the fields on the API documentation in https://apidocs.blueliv.com

3.5 Charts

After configuring the indexes, you can now play with the indexed data, either through search or creating charts and data tables. If you click in `Visualize`, on top menu (3.2), you can choose one of the available charts or data table.



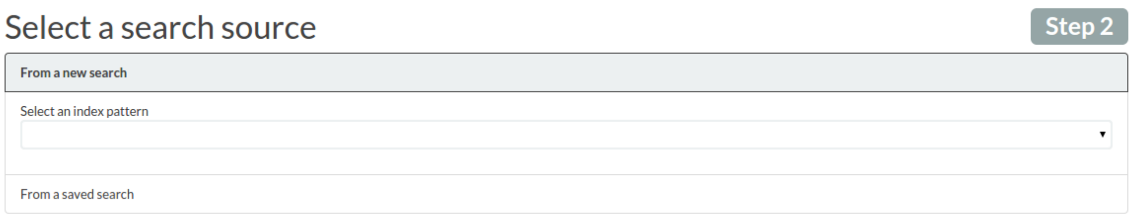
Create a new visualization

Step 1

Area chart	Great for stacked timelines in which the total of all series is more important than comparing any two or more series. Less useful for assessing the relative change of unrelated data points as changes in a series lower down the stack will have a difficult to gauge effect on the series above it.
Data table	The data table provides a detailed breakdown, in tabular format, of the results of a composed aggregation. Tip, a data table is available from many other charts by clicking grey bar at the bottom of the chart.
Line chart	Often the best chart for high density time series. Great for comparing one series to another. Be careful with sparse sets as the connection between points can be misleading.
Markdown widget	Useful for displaying explanations or instructions for dashboards.
Metric	One big number for all of your one big number needs. Perfect for show a count of hits, or the exact average a numeric field.
Pie chart	Pie charts are ideal for displaying the parts of some whole. For example, sales percentages by department.Pro Tip: Pie charts are best used sparingly, and with no more than 7 slices per pie.
Tile map	Your source for geographic maps. Requires an elasticsearch geo_point field. More specifically, a field that is mapped as type:geo_point with latitude and longitude coordinates.
Vertical bar chart	The goto chart for oh-so-many needs. Great for time and non-time data. Stacked or grouped, exact numbers or percentages. If you are not sure which chart your need, you could do worse than to start here.

Figure 3.2: Choosing a chart in Kibana.

After that, you must choose the source for your chart or data table, either a *new search* or a saved one (3.3) on an Elasticsearch index.



Select a search source

Step 2

From a new search

Select an index pattern

From a saved search

Figure 3.3: Choosing a source for a chart in Kibana.

After that you may choose the field that you want to show, as well as the aggregation (geo-location, count, average, etc). You may find some examples in section 3.6.1.

3.6 Dashboards

Kibana dashboards are just a view where you can get some of your graphs in a glance as well as rearrange them freely. If you click in `Dashboard`, on top menu (3.2), you can create a new dashboard or load an existing one. If you want to add a new graph into a dashboard, just click on `Add Visualization` button (last left-to-right in figure 3.4). After that you should choose one of the graphs (3.5) you created before (see 3.5).

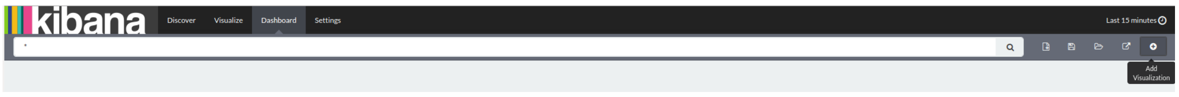


Figure 3.4: Adding a new graph into a Kibana dashboard.

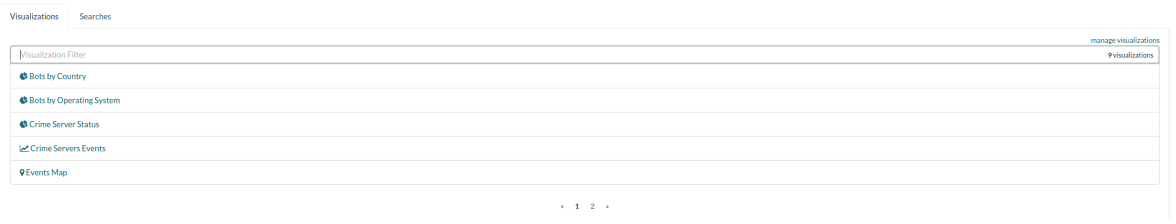
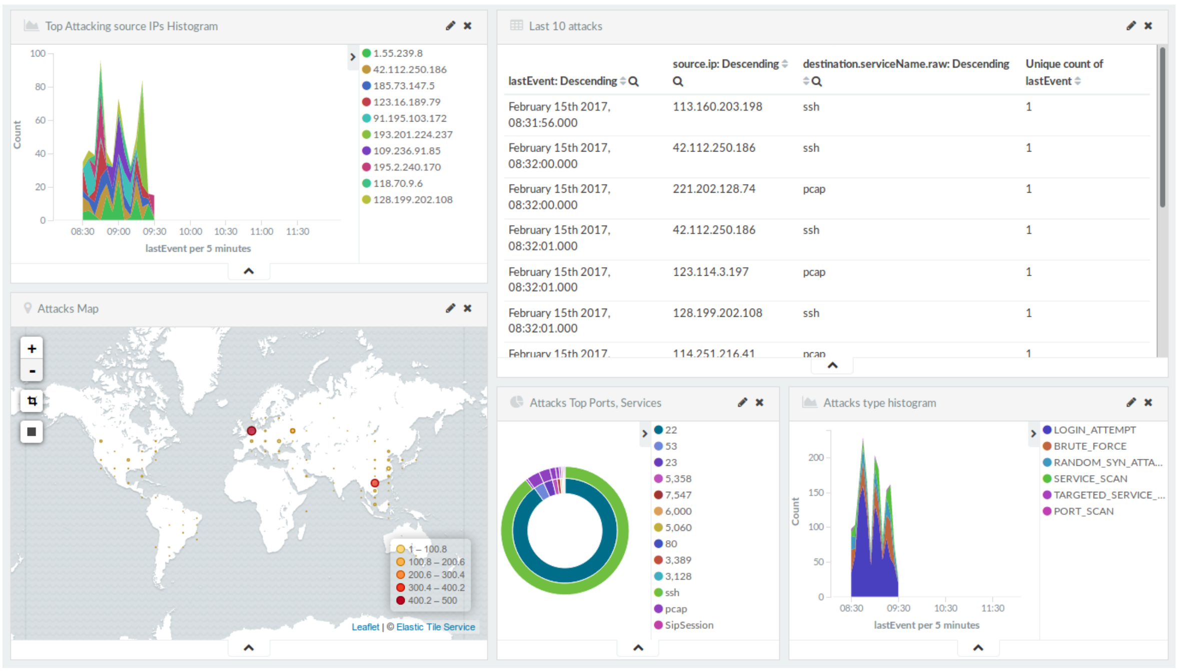


Figure 3.5: Select a saved graph to add into a Kibana dashboard.

You can find these dashboards at our GitHub.

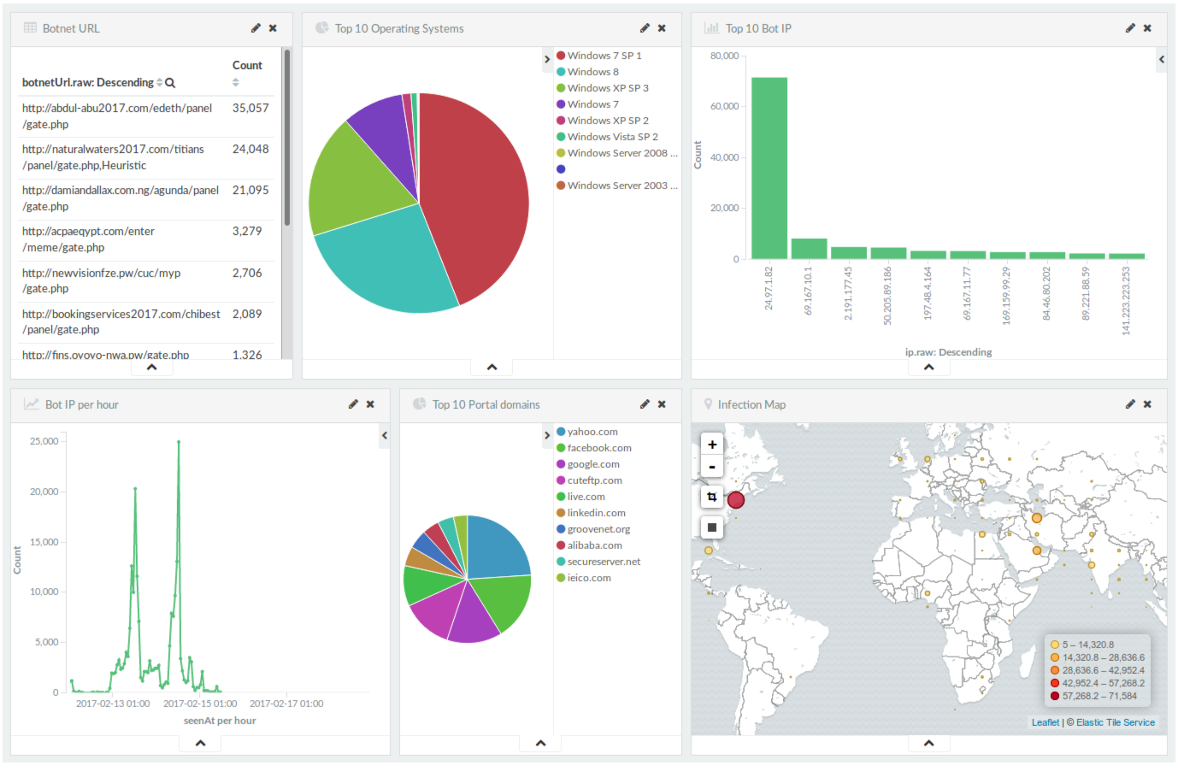
3.6.1 Example Dashboards

Here you can find four example Dashboards (3.6): Attacks, BotIPs, Crime-Servers and Malwares. Please be aware that these dashboards will only work if you set-up the indexes like we explained in listing 3.4. Otherwise you need to create them yourself.



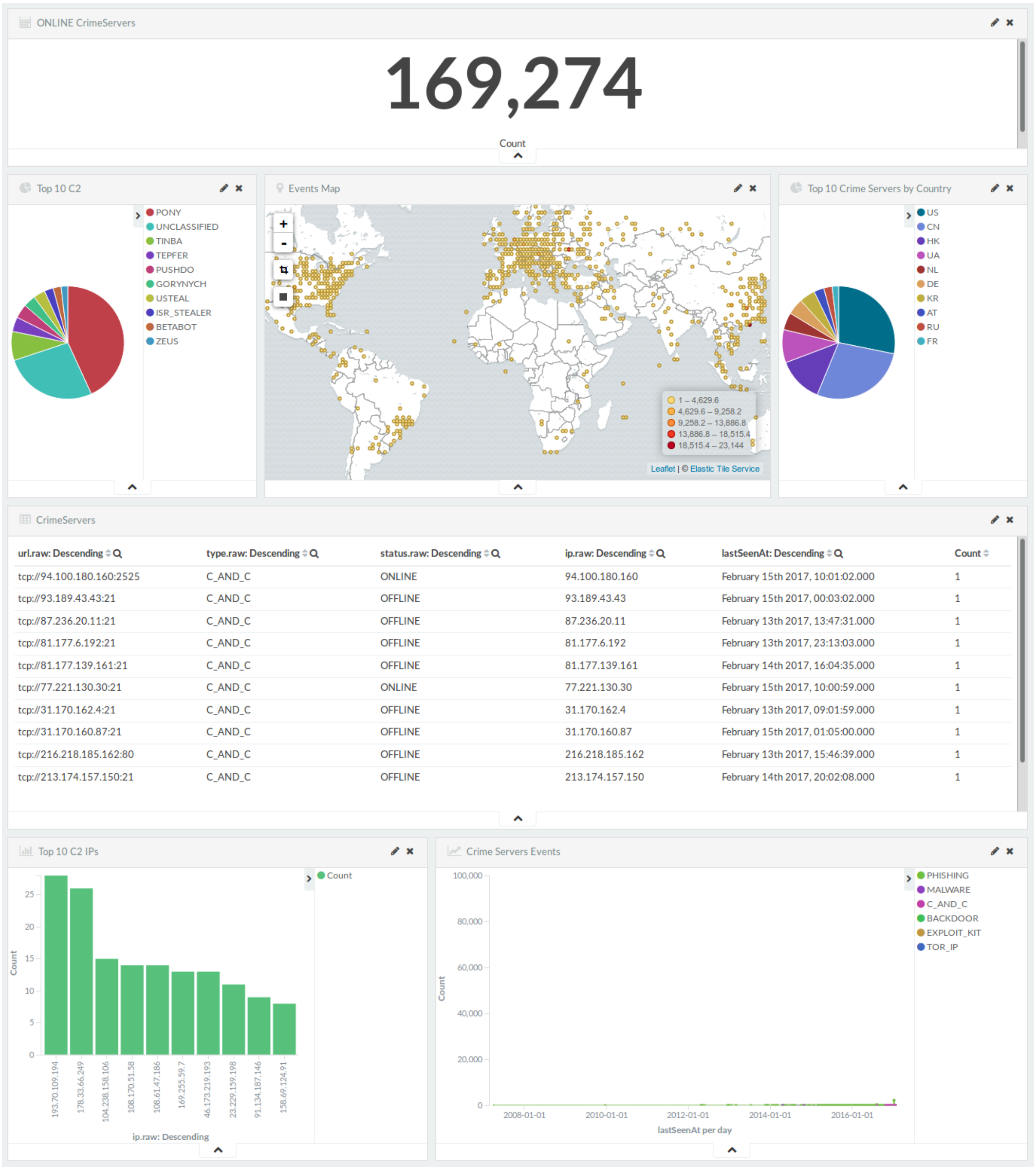
lastEvent: Descending	source.ip: Descending	destination.serviceName.raw: Descending	Unique count of lastEvent
February 15th 2017, 08:31:56.000	113.160.203.198	ssh	1
February 15th 2017, 08:32:00.000	42.112.250.186	ssh	1
February 15th 2017, 08:32:00.000	221.202.128.74	pcap	1
February 15th 2017, 08:32:01.000	42.112.250.186	ssh	1
February 15th 2017, 08:32:01.000	123.114.3.197	pcap	1
February 15th 2017, 08:32:01.000	128.199.202.108	ssh	1
February 15th 2017,	114.251.216.41	pcap	1

Figure 3.6: Attacks Example Dashboard



botnetUrl.raw: Descending	Count
http://abdul-abu2017.com/edeth/panel/gate.php	35,057
http://naturalwaters2017.com/titians/panel/gate.php,Heuristic	24,048
http://damiandallax.com.ng/agunda/panel/gate.php	21,095
http://acpaeqypt.com/enter/meme/gate.php	3,279
http://newvisionfze.pw/cuc/myp/gate.php	2,706
http://bookingservices2017.com/chibest/panel/gate.php	2,089
http://fins.ovovo-nwa.pw/gate.php	1,326

Figure 3.7: Bot IPs Example Dashboard



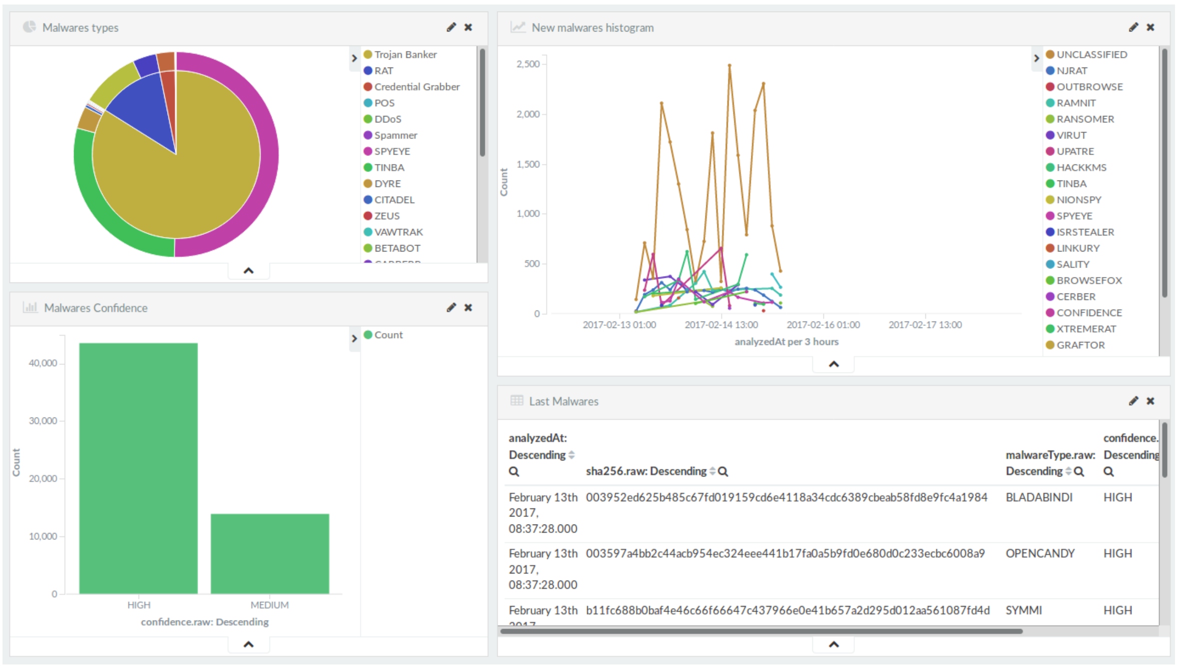
169,274

Count

CrimeServers

url.raw: Descending	type.raw: Descending	status.raw: Descending	ip.raw: Descending	lastSeenAt: Descending	Count
tcp://94.100.180.160:2525	C_AND_C	ONLINE	94.100.180.160	February 15th 2017, 10:01:02.000	1
tcp://93.189.43.43:21	C_AND_C	OFFLINE	93.189.43.43	February 15th 2017, 00:03:02.000	1
tcp://87.236.20.11:21	C_AND_C	OFFLINE	87.236.20.11	February 13th 2017, 13:47:31.000	1
tcp://81.177.6.192:21	C_AND_C	OFFLINE	81.177.6.192	February 13th 2017, 23:13:03.000	1
tcp://81.177.139.161:21	C_AND_C	OFFLINE	81.177.139.161	February 14th 2017, 16:04:35.000	1
tcp://77.221.130.30:21	C_AND_C	ONLINE	77.221.130.30	February 15th 2017, 10:00:59.000	1
tcp://31.170.162.4:21	C_AND_C	OFFLINE	31.170.162.4	February 13th 2017, 09:01:59.000	1
tcp://31.170.160.87:21	C_AND_C	OFFLINE	31.170.160.87	February 15th 2017, 01:05:00.000	1
tcp://216.218.185.162:80	C_AND_C	OFFLINE	216.218.185.162	February 13th 2017, 15:46:39.000	1
tcp://213.174.157.150:21	C_AND_C	OFFLINE	213.174.157.150	February 14th 2017, 20:02:08.000	1

Figure 3.8: Crime Servers Example Dashboard



analyzedAt: Descending	sha256.raw: Descending	malwareType.raw: Descending	confidence. Descending
February 13th 2017, 08:37:28.000	003952ed625b485c67fd019159cd6e4118a34cdc6389cbeab58fd8e9fc4a1984	BLADABINDI	HIGH
February 13th 2017, 08:37:28.000	003597a4bb2c44acb954ec324eee441b17fa0a5b9fd0e680d0c233ecbc6008a9	OPENCANDY	HIGH
February 13th	b11fc688b0baf4e46c66f66647c437966e0e41b657a2d295d012aa561087fd4d	SYMMI	HIGH

Figure 3.9: Malwares Example Dashboard

Registration

If you are interested in getting full access to our Threat Intelligence feed, contact us via email to get your API credentials that will allow you to update your local Data Base with current and real-time Threat Intelligence updates.

There are two access modes, Commercial and Free. If you are using the Free access, you will not have access to all the data we offer.

IMPROVE YOUR CYBER THREAT VISIBILITY

info@blueliv.com **www.blueliv.com**