Elasticsearch

Use cases and security best practices
OWASP Geneva meeting - 16.11.15

Julien Bachmann / julien /dot/ bachmann /at/ nagra /dot/ com

@milkmix_

map | you are here

- **Introduction**
  - me
  - disclaimer
- *Data is the new bacon*
- Platform presentation
- Security best practices for Elasticsearch

intro | who am I

- 7 years performing pentests and incident response
- Since 1.5 year playing on the defensive side within Kudelski Security
  - Security Architect in the Technology Team
  - *“Full stack security architect : from assembly to Gartner reports and beyond”*

intro | disclaimer

- Use cases presented here are far from complete
- Many types of attacks
  - many ways to detect them manually or automatically through logs
- IDS are another way, have a look at
  - https://speakerdeck.com/milkmix/clusis-campus-2015-introduction-to-suricata-ids
- Windows logs
  - https://speakerdeck.com/milkmix/import-module-incidentresponse

map | you are here

- Introduction
- ***Data is the new bacon***
  - use-cases for incidents detection
  - suspicious connections
  - sql injections
  - webshell
- Platform presentation
- Security best practices for Elasticsearch

part 1 | *data is the new bacon*

- Admin got a new idea
  - “why not leverage logs and detect attacks with them?”
  - need to define use-cases before jumping straight into the technology

use-cases | suspicious connections

- Bob (our admin) is administering his servers using SSH
  - default port is changed in order to remove brute-force attempt by kiddies
  - SSH generates events in
    - `/var/log/auth.log`

use-cases | suspicious connections

```
Nov  9 16:27:58 ucloud sshd[30957]: Accepted publickey for milkmix from 178.255.153.166 port 52077 ssh2: RSA 3f:34
```

date host ps user ingress IP

use-cases | suspicious connections

- Facts
  - Bob is always administering his servers from Switzerland
  - IP could be matched against GeoIP database to retrieve country of origin
- Example of use-case
  - detect fraudulent connections coming from a different country
  - match source IP against known malicious hosts

use-cases | suspicious connections

- Note
  - administering servers over the Internet is not common
  - even on AWS you might have a VPN as an enterprise or a single IP to connect from
  - generate your own GeoIP.dat for internal addressing:
    - https://github.com/mteodoro/mmutils

use-cases | sql injection

- Bob servers are running PHP scripts
  - some querying MySQL database (not even NoSQL, lame… ;))
- Apache logs are located in
  - `/var/apache2/access.log`

use-cases | sql injection

```
192.168.22.1 - - [09/Nov/2015:17:11:30 +0100] "GET /main.php?name=plop HTTP/1.1" 200 366 "-" "Mozilla/5.0 (Macintosh;
```

use-cases | sql injection

- Facts
  - Apache *access.log* contains number of bytes sent
  - exploiting SQL injection should generate a bigger request/response
  - exploiting a blind SQL injection requires more requests/responses
- Example of use-case
  - detect SQL injection exploitation by detecting higher *bytes-sent* value
  - detect blind SQL injection exploitation using queries frequency

use-cases | webshell

- Still on the PHP scripts
  - some pages allow to upload documents
  - Bob fears the following two vulnerabilities:
    1. *unrestricted upload of file with dangerous type*
    2. *improper control of filename for include/require statement in PHP*
- Apache logs are located in
  - `/var/apache2/access.log`

use-cases | webshell

```
192.168.22.1 - - [09/Nov/2015:17:11:30 +0100] "GET /main.php?name=plop HTTP/1.1" 200 366 "-" "Mozilla/5.0 (Macintosh;
```

source IP date URI return code user-agent

use-cases | webshell

- Facts
  - Apache *access.log* contains names of PHP scripts
  - if an attacker exploits the two vulnerabilities to upload a remote shell his accesses will be there
- Example of use-case
  - using the URI, detect PHP script which was not requested in the last 30 days (or shorter if you are in agile mode)

map | you are here

- Introduction
- *Data is the new bacon*
- **Platform presentation**
  - logstash
  - elasticsearch
  - kibana
  - elastalert
- Security best practices for Elasticsearch

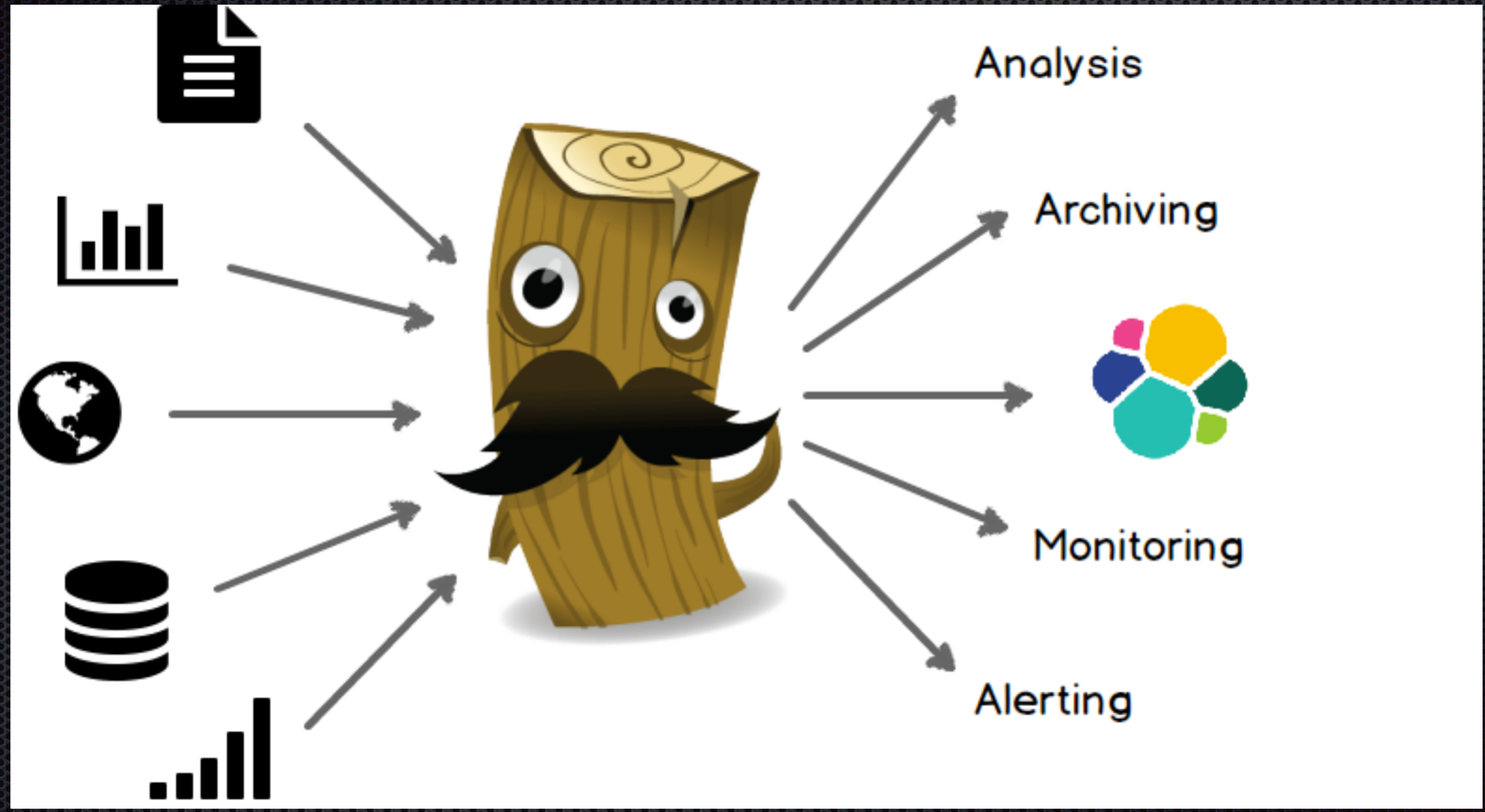
part 2 | platform presentation

- Although having a strong *grep-fu* he is willing to try the 2015 way
  - “I should have a look at those search database everyone has been talking about…”
  - Elasticsearch for example !
- *Wait… how do I ship logs to this Elasticsearch thing?*

elk | the stack

- Stands for **Elasticsearch**, **Logstash** and **Kibana**
  - not really used in that particular order
  - it includes:
    - a log collector with enrichment capabilities : Logstash
    - a search database based on Lucene : Elasticsearch
    - an interface to keep management happy : Kibana

elk | logstash

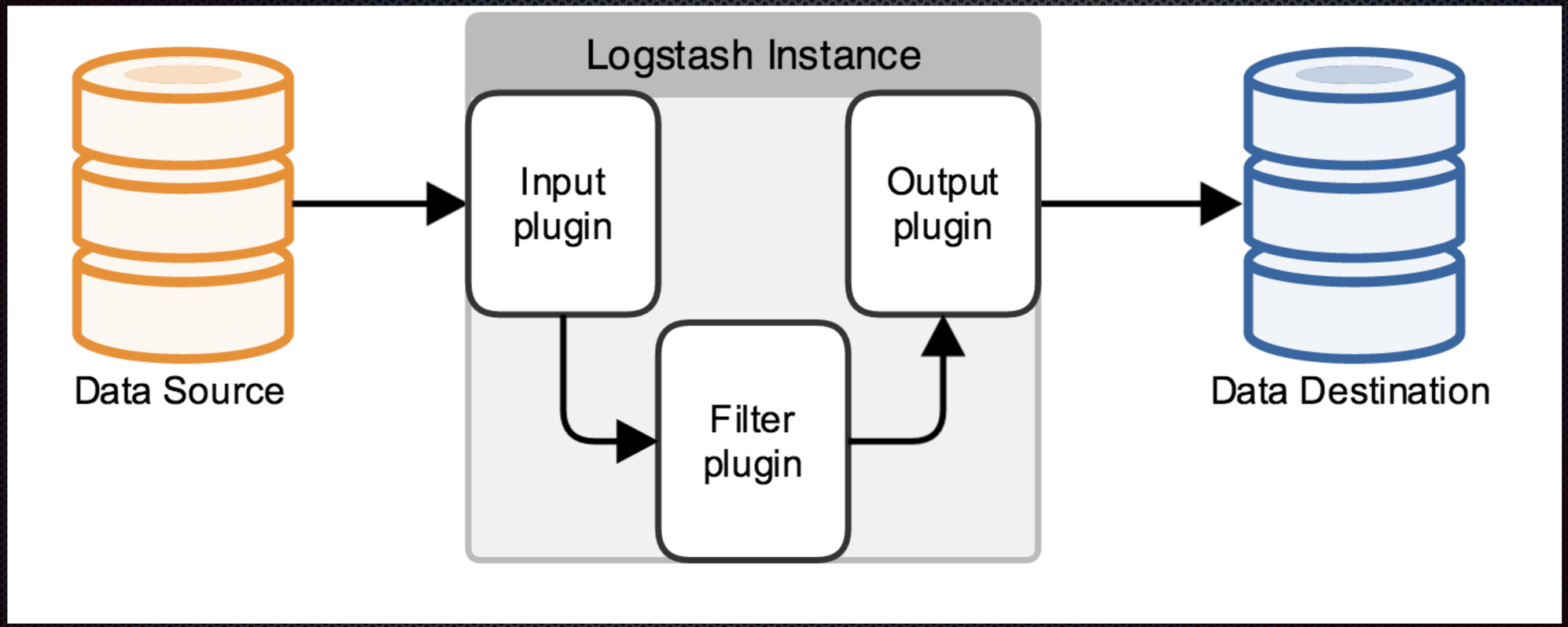


Credits: elastic.co

elk | logstash

- Logs collector, enrichment and shipper
  - *unifies data from disparate sources and normalise the data into destinations of your choice*
  - filters input using *Grok* language
  - enriches data using plugins
  - …

elk | logstash



Credits: elastic.co

elk | logstash

- Standard configuration file

```
1  input {
2      tcp {
3          port => 1337
4          codec => json
5      }
6  }
7
8  filter {
9  }
10
11 output {
12     stdout {
13         codec => rubydebug
14     }
15 }
```

elk | logstash

- Inputs
  - file, tcp/udp, syslog, twitter, sqlite, irc, kafka, …
  - codecs to automatically parse known file types
- Filters
  - grok, mutate, ruby, geoip, …
- Output
  - debug, elasticsearch, file, …

elk | logstash

- Plugins
  - easily develop plugin in Ruby (uh…)
  - ex: enrich logs with data from an external database to map user's identity

elk | logstash

- Example with `auth.log`
  - input: file
    - tips : don’t forget about the `.sincedb` file
  - filters: need to extract relevant info and enrich ingress IP with GeoIP data
  - output : debug for the moment

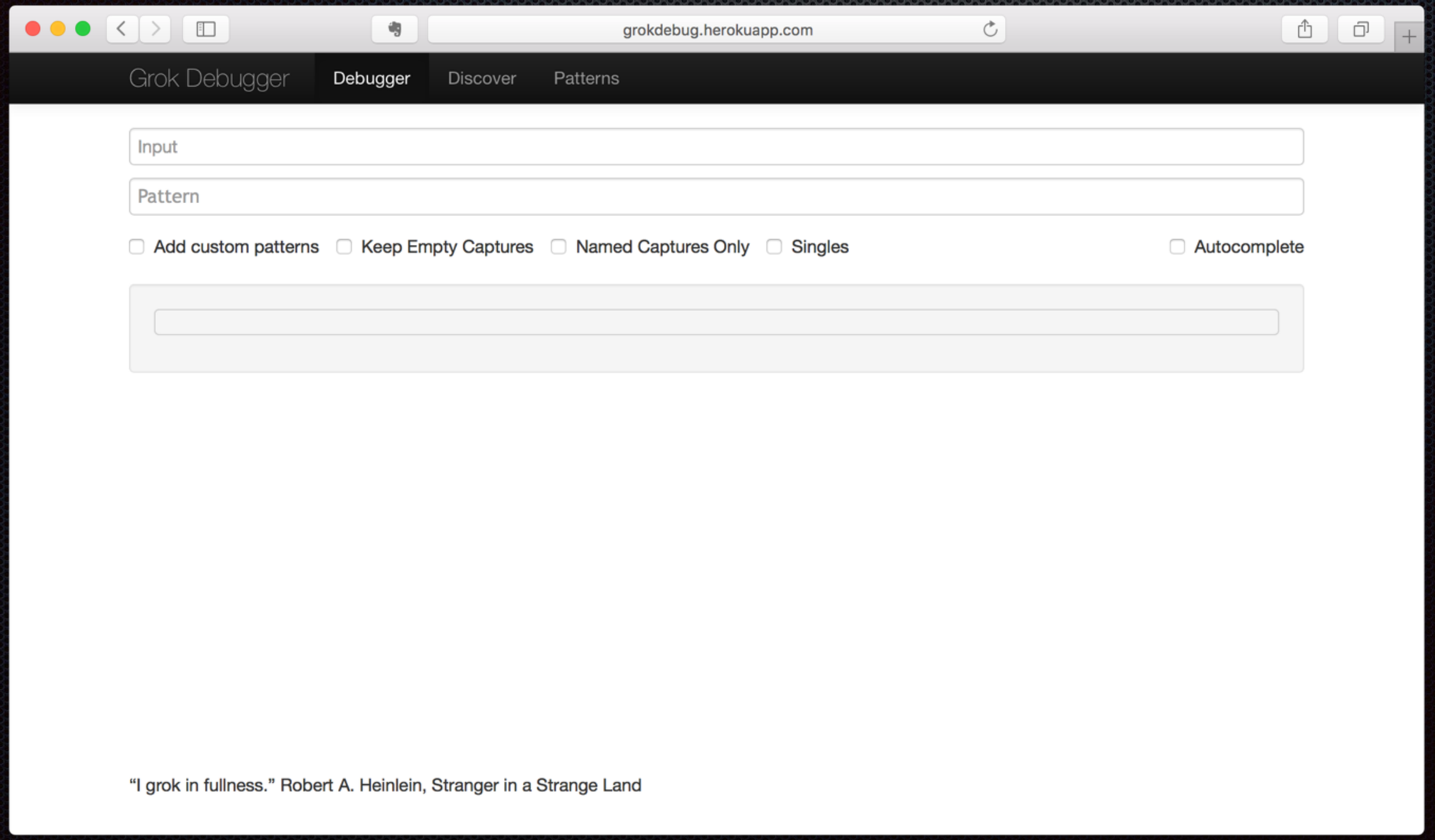
elk | logstash

```
input {
  file {
    type => "linux-syslog"
    path => [ "/home/milkmix/ucloud.log" ]
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  grok {
    match => [
    "message", "%{SYSLOGTIMESTAMP:timestamp} %{HOSTNAME:host_target} sshd\[%{BASE10NUM}\]: Accepted password for %{USERNAME:username} from %{IP:src_ip} port %{BASE10NUM:port} ssh2",
    "message", "%{SYSLOGTIMESTAMP:timestamp} %{HOSTNAME:host_target} sshd\[%{BASE10NUM}\]: Accepted publickey for %{USERNAME:username} from %{IP:src_ip} port %{BASE10NUM:port} %{GREEDYDATA}"
    ]
  }
  date {
    match => [ "timestamp", "MMM dd HH:mm:ss" ]
  }
  mutate {
    remove_field => [ "type", "port", "timestamp" ]
  }
  if "_grokparsefailure" in [tags] {
    drop { }
  }
  geoip {
    database => "./GeoIP.dat"
    source => "src_ip"
  }
}

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

elk | logstash


```
{
        "message" => "Nov 10 15:34:02 ucloud sshd[27975]: Accepted publickey for milkmix from 178.255.153.167 port 57898 ssh2: RSA 3f:34
       "@version" => "1",
     "@timestamp" => "2015-11-10T14:34:02.000Z",
           "host" => "ubuntu",
           "path" => "/home/milkmix/ucloud.log",
    "host_target" => "ucloud",
       "username" => "milkmix",
         "src_ip" => "178.255.153.167",
          "geoip" => {
                         "ip" => "178.255.153.167",
               "country_code" => 43,
              "country_code2" => "CH",
              "country_code3" => "CHE",
               "country_name" => "Switzerland",
             "continent_code" => "EU"
    }
}
{
        "message" => "Nov 11 15:41:40 ucloud sshd[27439]: Accepted publickey for milkmix from 198.11.246.180 port 62191 ssh2: RSA 3f:34:
       "@version" => "1",
     "@timestamp" => "2015-11-11T14:41:40.000Z",
           "host" => "ubuntu",
           "path" => "/home/milkmix/ucloud.log",
    "host_target" => "ucloud",
       "username" => "milkmix",
         "src_ip" => "198.11.246.180",
          "geoip" => {
                         "ip" => "198.11.246.180",
               "country_code" => 225,
              "country_code2" => "US",
              "country_code3" => "USA",
               "country_name" => "United States",
             "continent_code" => "NA"
    }
}
```

elk | logstash



elk | logstash

- Example with `access.log`
  - input: file
  - filters
    - use Grok patterns to speed-up the configuration
    - separate script name from his arguments
  - output : debug for the moment

elk | logstash

- Grok patterns

```
COMMONAPACHELOG %{IPORHOST:clientip} %{HTTPDUSER:ident} %
{USER:auth} \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %
{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})?|%
{DATA:rawrequest})" %{NUMBER:response} (?:%{NUMBER:bytes}|-)

COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %
{QS:agent}
```

elk | logstash

```
input {
  file {
    path => [ "/var/log/apache2/access.log" ]
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  grok {
    match => [
      "message", "%{COMBINEDAPACHELOG}"
    ]
  }
  grok {
    match => [
      "request", "%{NOTSPACE:script}.php(\?%{NOTSPACE:args})?"
    ]
  }
  date {
   match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
  mutate {
    remove_field => [ "@version", "type", "port", "timestamp" ]
  }
  if "_grokparsefailure" in [tags] {
    drop { }
  }
}

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

elk | logstash

```
{
       "message" => "192.168.22.1 - - [09/Nov/2015:17:11:30 +0100] \"GET /main.php?name=plop HTTP/1.1\" 200 366 \"-\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 1
me/46.0.2490.80 Safari/537.36\"",
    "@timestamp" => "2015-11-09T16:11:30.000Z",
          "host" => "ubuntu",
          "path" => "/var/log/apache2/access.log",
      "clientip" => "192.168.22.1",
         "ident" => "-",
          "auth" => "-",
          "verb" => "GET",
       "request" => "/main.php?name=plop",
   "httpversion" => "1.1",
      "response" => "200",
         "bytes" => "366",
      "referrer" => "\"-\"",
         "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36\"",
        "script" => "/main",
          "args" => "name=plop"
}
{
       "message" => "192.168.22.1 - - [11/Nov/2015:15:51:05 +0100] \"GET /error.php HTTP/1.1\" 404 502 \"-\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) A
490.80 Safari/537.36\"",
    "@timestamp" => "2015-11-11T14:51:05.000Z",
          "host" => "ubuntu",
          "path" => "/var/log/apache2/access.log",
      "clientip" => "192.168.22.1",
         "ident" => "-",
          "auth" => "-",
          "verb" => "GET",
       "request" => "/error.php",
   "httpversion" => "1.1",
      "response" => "404",
         "bytes" => "502",
      "referrer" => "\"-\"",
         "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36\"",
        "script" => "/error"
}
```

elk | logstash

- In real life, some additional actions are required
  - verify that all servers are time synchronised and/or timezone correctly set
  - which fields should be kept?
  - what information should be added to the events?
  - …
- Events parsing is one of the pain-point when doing logs management/SIEM
  - same applies for **AlienVault**, **Splunk**, …

elk | logstash

- *Does it scale?*
  - if you really need it, it can yes
  - use **Apache Kafka** nodes to collect logs
  - forward from them to **logstash** to enrich/forward
- *And for Windows events?*
  - use **NXLog** to ship events from Windows hosts

elk | elasticsearch

- Search database
  - “schema-free”
  - full text search thanks to **Lucene** backend
  - distributed and scalable
  - replication of your data across nodes
  - easy to use REST-API

elk | elasticsearch

- Configuration
  - `config/elasticsearch.yaml`
  - quite easy to create a cluster
    - set `cluster.name` to desired value
    - allow nodes to communicate together on unicast
  - load balancer nodes
    - `node.data: false`
    - `node.master: false`

elk | elasticsearch

- Configuration
  - not all of the configuration is easy
  - `ES_HEAP_SIZE`
  - `number_of_{shards, replicas}` for indexes
  - manage logs rotation using **curator**
  - …

elk | elasticsearch

- Structure
  - documents have an **_id**
    - automatically generated but can be forced if needed by use-case
  - documents are regrouped in **_type**
  - an **index** regroups several types
    - {index}/{type}/{id}

elk | elasticsearch

- *Schema-free*
  - technically yes since you can throw in a *json* file and have it indexed
  - in the background ES is creating the schema for you !
  - in order to have correct and faster results in search mode, a correct **mapping** is required
  - default one might not be optimal or functional for you
  - ex: hosts name with **.** which is also a separator for default indexer

elk | elasticsearch

- Mapping
  - defines *type*, indexer and other properties of document's fields
  - *type* can be **string**, **integer**, **IP**, **date**, **boolean**, **binary**, **array**, **geopoint**, …
  - *format* is for date fields
  - *index* is defined to **analysed** by default, other value is **not_analyzed**

elk | elasticsearch

- Important point on mappings !
  - once defined a mapping cannot be changed for an index
  - need to re-index all of it
    - yep, this could be quite bad if you just discovered it after indexing 1TB
  - you can use **aliases** on indexes to create new mapping faster
  - think about your use-cases and perform tests gradually

elk | elasticsearch

- Put mapping
  - `curl -XPUT 'http://localhost:9200/sshd/' -d@auth.log.mapping`
- Retrieving index mapping
  - `curl -XGET ‘http://localhost:9200/sshd/_mapping?pretty'`

elk | elasticsearch

```
[milkmix@ubuntu :: logstash-2.0.0 )) curl -XGET 'http://localhost:9200/sshd/_mapping?pretty'
{
  "sshd" : {
    "mappings" : {
      "logs" : {
        "properties" : {
          "@timestamp" : {
            "type" : "date",
            "format" : "strict_date_optional_time||epoch_millis"
          },
          "@version" : {
            "type" : "string"
          },
          "geoip" : {
            "properties" : {
              "continent_code" : {
                "type" : "string"
              },
              "country_code" : {
                "type" : "long"
              },
              "country_code2" : {
                "type" : "string",
                "index" : "not_analyzed"
              },
              "country_code3" : {
                "type" : "string"
              },
              "country_name" : {
                "type" : "string",
                "index" : "not_analyzed"
              },
              "ip" : {
                "type" : "string"
              }
            }
          },
          "host" : {
            "type" : "string",
            "index" : "not_analyzed"
          },
          "host_target" : {
            "type" : "string"
          },
          "message" : {
            "type" : "string"
          },
          "path" : {
            "type" : "string"
          },
```

elk | elasticsearch

- *Wait, go back one slide! How did you send the sshd logs into ES ?*
  - using the **elasticsearch** output in logstash :)

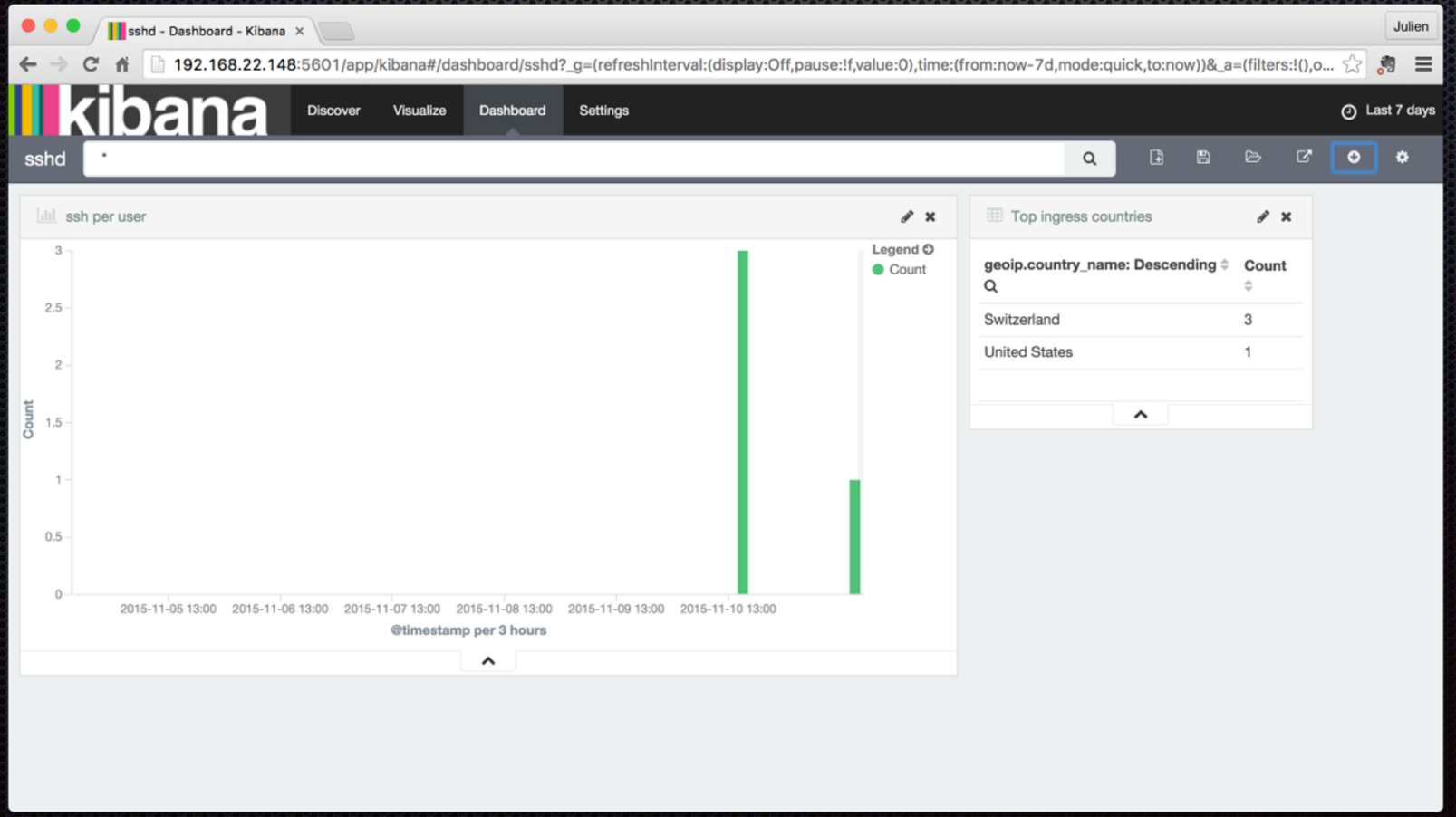
```
29 output {
30   stdout {
31     codec => rubydebug
32   }
33   elasticsearch {
34     hosts => [ "localhost" ]
35     index => "sshd"
36   }
37 }
```

elk | kibana

- Graphical interface to Elasticsearch
  - really easy to set-up
  - might be limited for specific use-cases : increase your *es-query-fu*

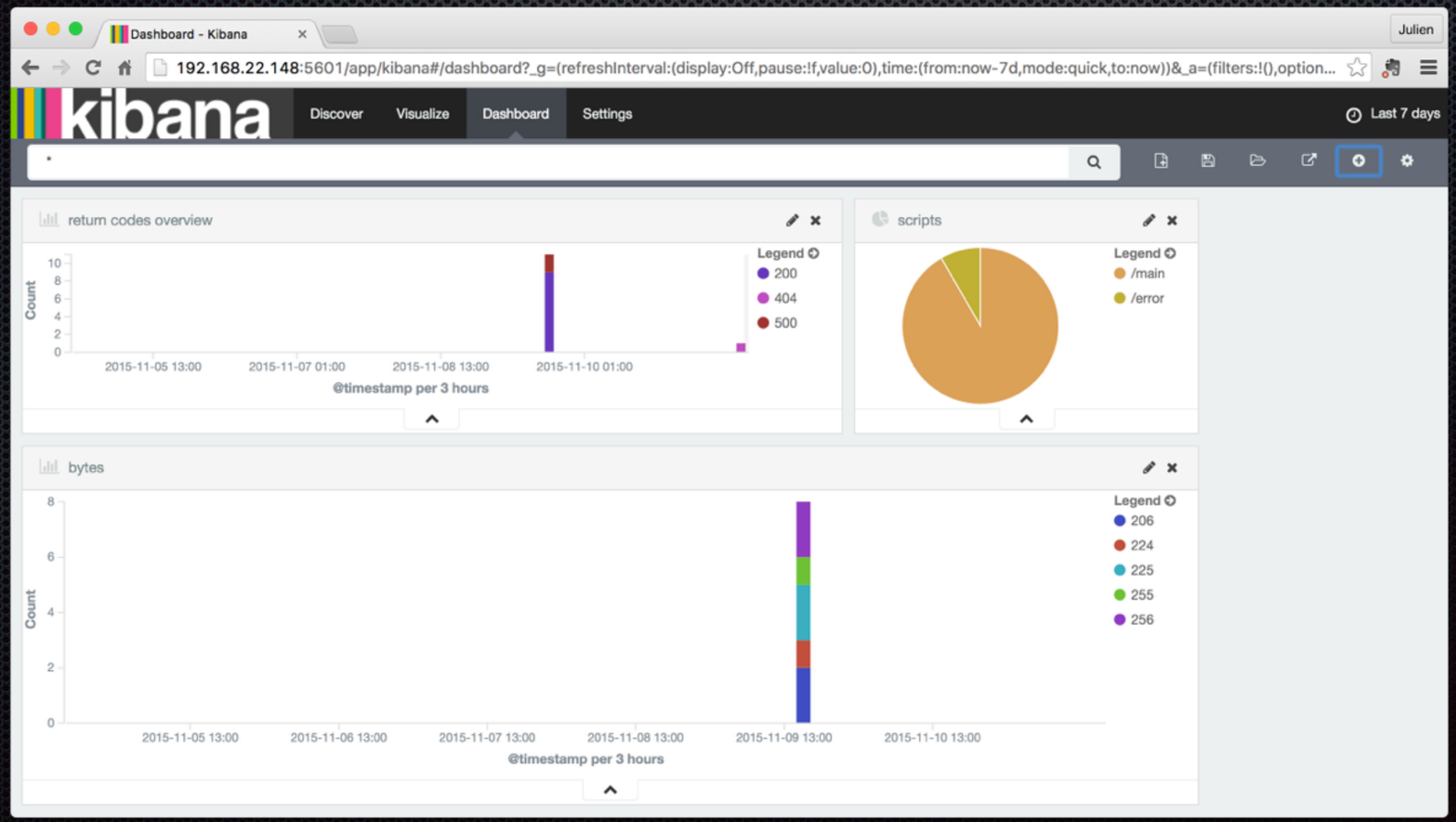
elk | kibana

- Sample dashboard for sshd logs

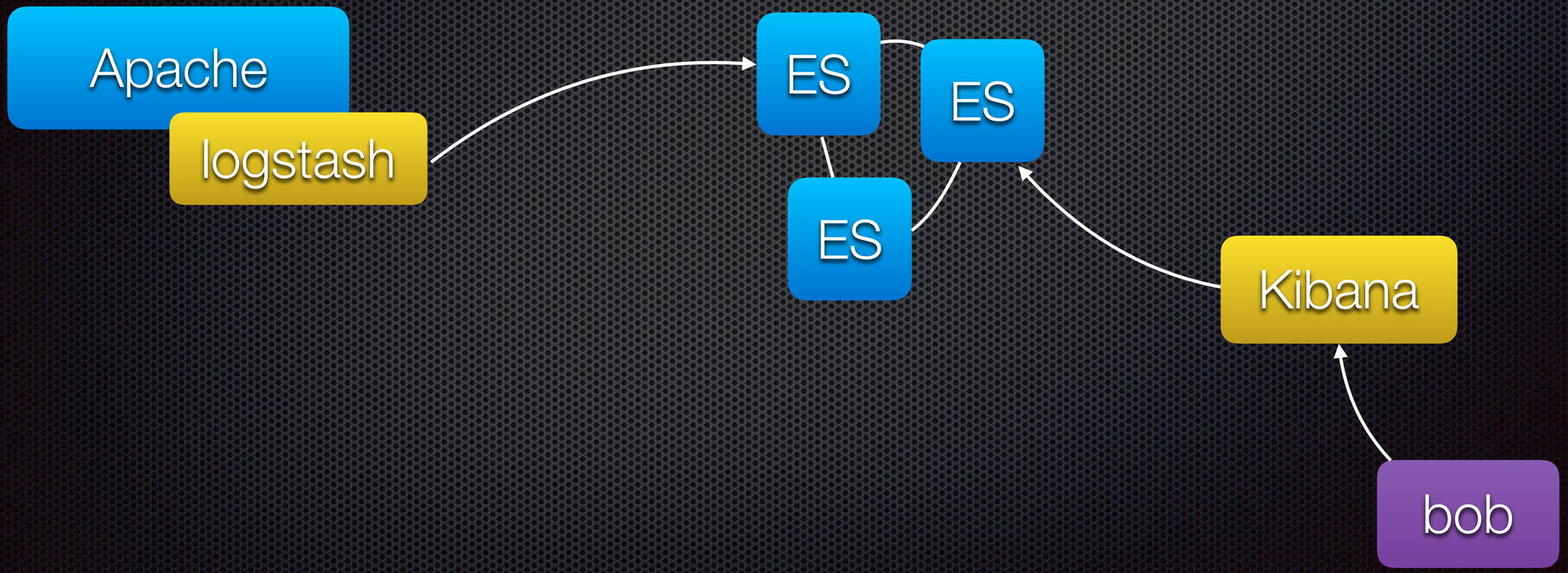


elk | kibana

- Sample dashboard for apache logs



elk | summary



alerting | elastalert

- Open source project by Yelp
  - made to answer to the : *how do I watch over thousand of servers?*
  - https://github.com/Yelp/elastalert

alerting | elastalert

- Concept
  - use events stored in Elasticsearch
  - simple rules written in yaml files
  - generate alerts to several providers
    - conventionals : email, Jira
    - or for the more hipsters of you : Slack, HipChat, PagerDuty

alerting | elastalert

- Types of alerts
  - blacklist / whitelist
  - value change
  - new term
  - cardinality
  - frequency
  - spike
  - flatline

alerting | elastalert

- Back to our use-cases
  - ingress ssh connection from a different country: **new term** or **change**
  - high number of queries : **frequency**
  - webshell deployed by attacker : **new term**

alerting | elastalert

- Ingress ssh countries

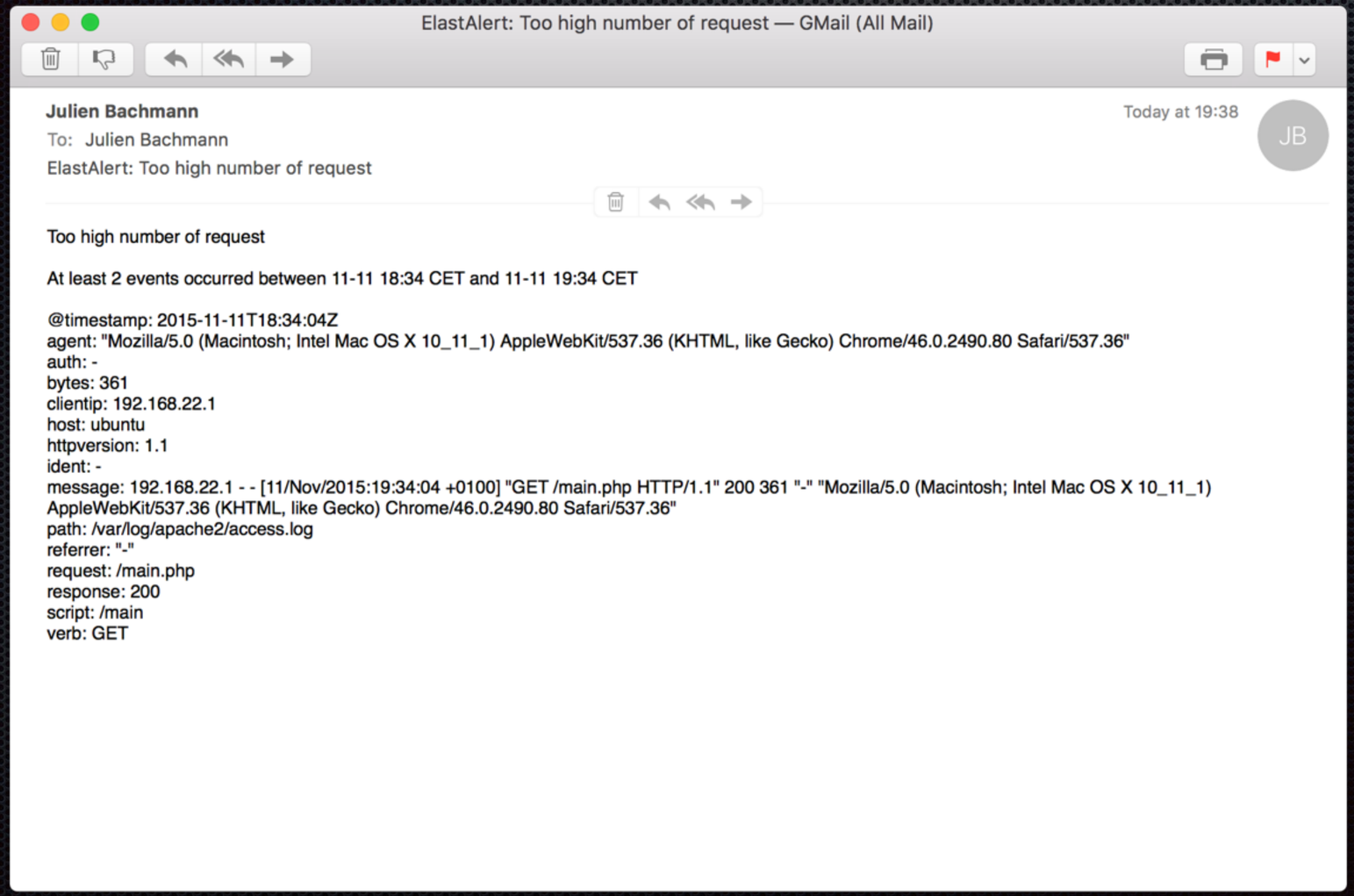
```
es_host: localhost
es_port: 9200
index: sshd

name: New ingress country for sshd
type: new_term

fields:
 - "geoip.country_code2"
terms_window_size:
  days: 30

filter:
- term:
    _type: "logs"
- term:
    username: milkmix

alert:
- "email"
alert_text: |
        W00t w00t! Cyber surveillance just detected a suspicious connection coming from {0} ({1}).
alert_text_args:
- geoip.country_name
- src_ip
email:
- "admin@domain.com"
```

alerting | elastalert

- High number of HTTP queries

```yaml
es_host: localhost
es_port: 9200

name: Too high number of request

type: frequency
index: apache

num_events: 100
timeframe:
  hours: 1

filter:
- term:
    response: 200

alert:
- "email"
email:
- "admin@domain.com"
```

alerting | elastalert



alerting | elastalert

- Limitations
  - not possible to correlate between multiple indexes
  - no rules on term values
    - could be circumvented using filters but not all features will work
- But
  - elastalert is designed to be extensible
  - new rule types can be developed

map | you are here

- Introduction
- *Data is the new bacon*
- Platform presentation
- **Security best practices for Elasticsearch**
  - default behaviour
  - network / transport
  - authentication / authorisation
  - hardening
  - *shield*

part 3 | *wait, where is my data?!?*

- Admin got back to work but ES cluster looks down
  - service not running anymore
  - after rebooting it, it appears that all data has been deleted

concept | elasticsearch

Responsibility

> *Elasticsearch has no concept of a user. Essentially, anyone that can send arbitrary requests to your cluster is a “super user”.*

concept | elasticsearch

- REST API
  - get
  - index
  - delete
  - update

concept | elasticsearch

- Based on two parts
  - HTTP verbs
    - GET, PUT, DELETE
  - URL
    - action : `_search`, `_mapping`, `_update`, `_shutdown`, `_snapshot/_restore`, ...
    - path : index or alias (transparent)

concept | elasticsearch

- On the network side
  - cleartext protocol
  - cluster nodes discovery using unicast

concept | elasticsearch

- At the application level
  - possibility to perform dynamic scripting
  - plugins mechanism
  - secure development
    - CVE-2015-5531 : directory traversal allowing to read arbitrary files
    - CVE-2015-4093 : XSS
    - CVE-2015-1427 : sandbox bypass, execute arbitrary shell commands
    - …

protection | plan

- Several factors on which to operate
  - network segmentation
  - transport security
  - authentication / authorisation
  - hardening

protection | network segmentation

- Separate Elasticsearch cluster from the rest of the network
  - dedicated VLAN + firewall
  - setup a load-balancing node and make it the only network-reachable endpoint
  - also applicable to Hadoop and the like, …

protection | transport security

- Could be difficult to set proper SSL tunnels between nodes
  - need a PKI (but who doesn't in 2015? ;))
  - wrap Elasticsearch in `stunnel` or similar solution
- Easier
  - network segmentation so inter-nodes communications are not accessible
  - Kibana/querying host behind a jump host
  - access through SSH tunnelling

protection | transport security

- *Ok, but when I have X writers and not only consumers for ES ?*
  - set-up a reverse proxy with SSL connections only
  - `Nginx` for example

```
ssl on;

ssl_certificate /etc/ssl/cacert.pem;

ssl_certificate_key /etc/ssl/privkey.pem;
```

protection | authentication

- Set-up a reverse proxy
  - `nginx` again
  - `auth_basic` / `auth_basic_user_file` options in the configuration file
  - do not forget to also add transport security for the credentials security
  - **Kibana** and **ElastAlert** are compatible

protection | authorization

- Set-up a reverse proxy
  - `nginx` again
  - filter by location and HTTP verb

```
limit_except GET {

…

}
```

protection | hardening

- Beware if you are using packaged solutions
  - didn’t specifically look at them
  - could be bundled with unnecessary (vulnerable) services
- Disable dynamic scripting
  - now the default setting

protection | monitoring

- Do not forget to monitor your cluster status
  - elastic.co Marvel
  - elastichq

protection | not that easy

- *This seems cool, but not really simple to set-up*
  - many points to cover
  - probably why elastic.co released a product to circumvent this
  - **Shield**
    - please note the references to Marvel comics :)

protection | shield

- Functionalities
  - authentication (local, LDAP, AD, PKI)
  - role based access control
  - granular level of security at the document and field level
  - inter-nodes transport security
  - auditing

protection | shield

- This is unfortunately not a freeware
  - require to have a subscription based license
  - this is highly recommended as soon as you step out of the POC garden
  - expertise on ES could save you quite some time

protection | shield

- Demo version for 60 days

```
[milkmix@ubuntu :: elasticsearch-2.0.0 ›› ./bin/plugin install elasticsearch/license/latest
 -> Installing elasticsearch/license/latest...
 Trying https://download.elastic.co/elasticsearch/license/license-latest.zip ...
 Downloading .......DONE
 Verifying https://download.elastic.co/elasticsearch/license/license-latest.zip checksums if available ...
 Downloading .DONE
 Installed license into /home/milkmix/local/logs/elasticsearch-2.0.0/plugins/license
[milkmix@ubuntu :: elasticsearch-2.0.0 ›› ./bin/plugin install elasticsearch/shield/latest
 -> Installing elasticsearch/shield/latest...
 Trying https://download.elastic.co/elasticsearch/shield/shield-latest.zip ...
 Downloading ..................................................................................................DONE
 Verifying https://download.elastic.co/elasticsearch/shield/shield-latest.zip checksums if available ...
 Downloading .DONE
 Installed shield into /home/milkmix/local/logs/elasticsearch-2.0.0/plugins/shield
```

protection | shield

```
[milkmix@ubuntu :: ~ ›› curl -XGET 'http://localhost:9200/sshd/_mapping?pretty'
{
  "error" : {
    "root_cause" : [ {
      "type" : "security_exception",
      "reason" : "missing authentication token for REST request [/sshd/_mapping?pretty]",
      "header" : {
        "WWW-Authenticate" : "Basic realm=\"shield\""
      }
    } ],
    "type" : "security_exception",
    "reason" : "missing authentication token for REST request [/sshd/_mapping?pretty]",
    "header" : {
      "WWW-Authenticate" : "Basic realm=\"shield\""
    }
  },
  "status" : 401
}
```

```
[milkmix@ubuntu :: elasticsearch-2.0.0 ›› curl -u es_admin -XGET 'http://localhost:9200/sshd/_mapping?pretty'
[Enter host password for user 'es_admin':
{
  "sshd" : {
    "mappings" : {
      "logs" : {
        "properties" : {
          "@timestamp" : {
            "type" : "date",
            "format" : "strict_date_optional_time||epoch_millis"
          },
          "@version" : {
            "type" : "string"
          },
          "geoip" : {
            "properties" : {
```

protection | shield

- Local configuration
  - not centralised: configuration files to be pushed to each member/node
  - highly recommend to use **Ansible** or other automation solution
  - simple yaml file
    - `roles.yaml`

protection | shield

- Roles
  - Apache servers : write in `apache` index
  - Linux servers accessed through ssh : write in `sshd` index
  - Kibana : read both indexes (and the one for itself)
  - ElastAlert : read both indexes, write in `elastalert_status`

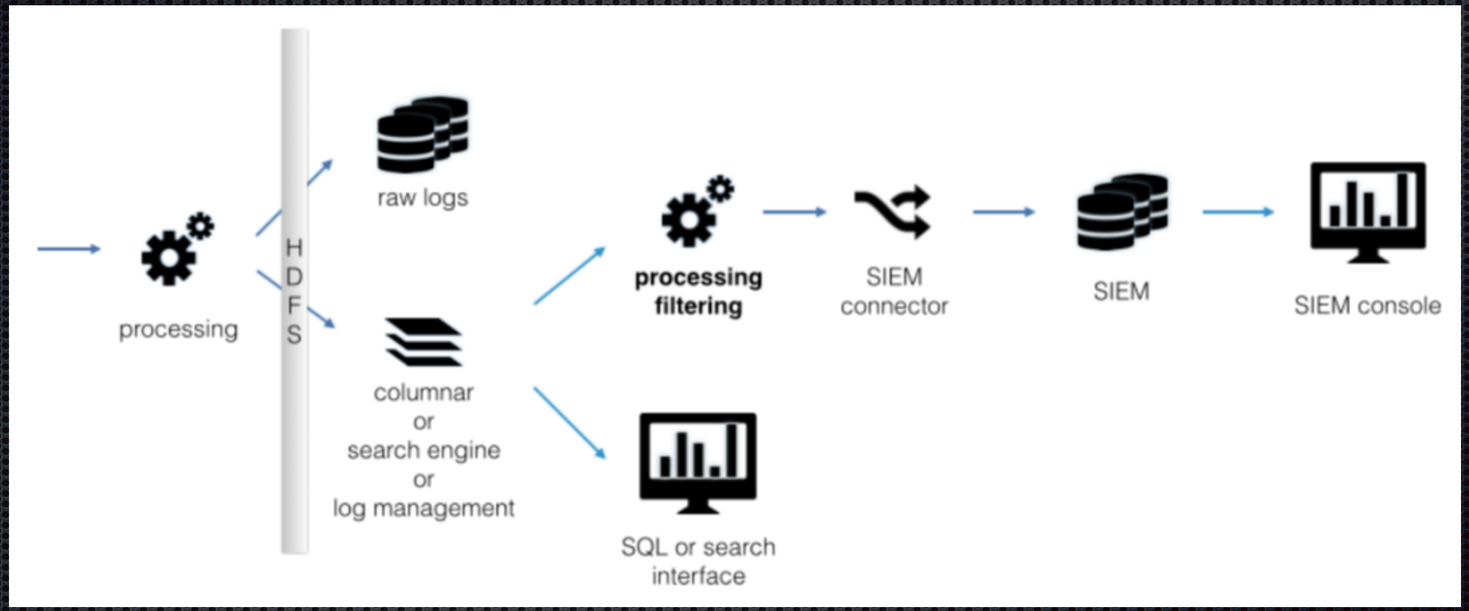
conclusion | wrap-up

- Elasticsearch is not a SIEM by itself
  - log management : OK
  - events correlation : not automated
- Need some external development and administration compared to COTS solutions
- Or choose the “buy way” instead of the “make-way”

conclusion | wrap-up

- Full open source solution might rather look like the following
  - logs, context, pcap, … storage : HDFS
  - some use-cases : Elasticsearch
  - some others: Cassandra
  - and others: Neo4J
- Add some machine learning and shake hard… ;)

conclusion | wrap-up



Credits: raffy.ch

conclusion | wrap-up

- Important points before going into a SIEM/SOC project
  - state your current security maturity level
  - list your assets, associated risks, threat models, …
  - think about your use-cases
    - ex: work with results from pentests
  - list external sources that should be accessible from the SIEM
    - ex: threat intelligence feeds

conclusion | readings

- Raffy blog
  - SIEM use-cases
    - http://raffy.ch/blog/2015/05/07/security-monitoring-siem-use-cases/
  - Big data lake
    - http://pixlcloud.com/security-big-data-lake/

conclusion | readings

- Florent blog
  - serie on log management
    - http://www.ikangae.net/category/log-management/

conclusion | questions ?

KUDELSKI SECURITY

@milkmix_