Combining logs and threat data feeds

Guest lecture – Aarhus Business Academy
David Thejl-Clayton - 18/03/2019

https://www.ecrimelabs.com

Lab along!

Download the latest MISP virtual machine:

https://www.circl.lu/misp-images/latest/

It is not required to follow the lesson, don't worry. But if you want to leave with something you can use for testing after and follow along with the first part of the lesson then go and grab it!

It can be used in both VMWARE and VirtualBox.

eCrimeLabs

Lesson goal

How to leverage open source intelligence sources to enrich data in your log management platform…

By the end of the lesson you will have a deeper understanding of the following:-

- MISP as a threat feed source – API
- Logstash – Parsing and normalization
- Memcache
- Visualizing the enrichment in ElasticSearch/Kibana

eCrimeLabs

Quick note

- I am no good at formal presentations or teaching from a strict lesson plan…
- A lot of todays lesson will be fluid and I will use the whiteboard, live demo (gods permitting) and general sparring to show you how to accomplish the goal…
- At eCrimes we are in love with Open Source, so this talk will focus primarily on Open Source tools and concepts…

eCrimeLabs

Who am I?

- British but feel like a Dane…
- Living in Denmark for 4 years, with my wife and son.
- Daughter on the way in 6 weeks!
- Of those 4 years, 3.5 have been spent working at JN Data…
- 2 years working with security!

eCrimeLabs

What do I do?

- SOC and Incident Response, Service Architect and Technical Lead at JN Data
  - Incident response lead
  - Threat intelligence analyst
  - Security architect
- Research and development at eCrime Labs
- Blogger at Security Distractions

The question…

- Evil Corp currently log 40.000 endpoints worth of data, via multiple log sources:
  - Network security devices (firewall, proxy)
  - Infrastructure logs (DNS, DHCP)
  - EDR (host logs)
- Can we enrich these logs with open source threat data feeds and our own threat data?

eCrimeLabs

Requirements

The solution must:

- Handle large amounts of events per second
- Provide near realtime enrichment without delaying log ingestion
- Handle the dynamic nature of the intelligence source – Fast in, fast out IOC's
- Handle multiple threat feeds
- Handle multiple IOC types – Domain, IP, hash values etc
- Handle large lists of IOC's
- Be scalable!
- PHEW!

eCrimeLabs

Breaking up the requirements…

A high level plan to solve this question is going to involve research into:-

- Threat data feeds
- A place to store our own threat information
- Logging enrichment
- Combining all of the above!

eCrimeLabs

Threat data feed’s

Before we dive in too deep down the rabbit hole, lets cover some basics….

- What is a threat feed?
  - An ongoing stream of data, relating to potential, current or historical threats.
  - Generally made up of a list of IOC’s of specific data types.
  - There are both open source and private threat data feeds out there.

Threat data feed's

- Examples:-
  - https://zeustracker.abuse.ch/blocklist.php?download=compromised
  - https://ransomwaretracker.abuse.ch/feeds/csv/
  - https://zeustracker.abuse.ch/blocklist.php?download=ipblocklist
  - https://urlhaus.abuse.ch/downloads/csv/

eCrimeLabs

Threat data feed’s

How does data land on a threat data feed?

- Automated based on alerts from a security device…
  - Symantec’s GIN example using MAA’s (whiteboard)
- Malware reverse engineering…
  - IOC’s found within the malware, call back domains etc…
- Added by a threat intelligence analyst from an incident response report.

Threat data feed’s

How does an organisation decide which threat data feeds to use?

A lot of organisations simply want to block all IOC’s on Ransomware Tracker…

Downsides with threat data feeds…

- No standard for listing data… Some contain plain IOC’s, others provide timestamps with the IOC’s…
  - Requires custom parsing per feed if you want to leverage it
- Example:
- https://urlhaus.abuse.ch/downloads/csv/
- https://zeustracker.abuse.ch/blocklist.php?download=compromised

Downsides with threat data feeds…

- There is no real context with a threat data feed…
  - Yeah great we have found an IOC in our infrastructure from the Abuse.ch feed, but exactly what was that IOC used for… Was it a C&C domain, was it used to host the payload?
- But what about Ransomware tracker, the context here is that it is ransomware right?
  - We still need more context, which campaign, is this targeted at us or global?

eCrimeLabs

The current state of play…

- Whiteboard…

eCrimeLabs

What options do we have for threat data feeds?

- We could retrieve a feed and then use that…
  - But this does not meet our requirement for multiple threat data feeds!
- We could retrieve the feeds individually, parse them and combine them as one big feed.
  - But how to separate these feeds? By datatype?
    - Bad domain feed? Bad IP address feed?

Threat data feed - Example

For the rest of the lesson, we will be using the Ransomware Tracker threat data feed…

At the end of the lesson we will come back to the requirement around multiple threat data feeds…

Threat data feed aggregation

Lets not reinvent the wheel here… There are a number of Open Source tools that perform threat feed aggregation…

- MISP – Developed by CIRCL in Luxembourg
- Hippocampe – Developed by the team behind, The Hive and Cortex
- Crits – Developed by Mitre corporation (Att&ck framework)

MISP - Brief

You have probably been hearing quite a lot about MISP recently… So we won’t focus too much on MISP separately as today is about logging!… But we are interested in the following MISP capabilities:-

- Storing threat feed data
- Storing our own intelligence data
- API

eCrimeLabs

MISP – Threat Feeds

- MISP offers the ability to download threat data feeds and store them as events...
- Out of the box it has many of the well known OSINT feeds ready for download, Ransomware tracker, Abuse.ch etc...
- You can also add other not so well known feeds...
  - Custom feed creation...

MISP – Threat Feeds

- Demo

eCrimeLabs

MISP – Storing your own intelligence

You can create your own events within MISP, made up of various attributes.

Attributes are primarily IOC’s and come in a range of datatypes…

MISP – Storing your own intelligence

- Demo

eCrimeLabs

MISP - Tags

You can create custom tags within MISP and assign these tags to attributes or to whole events.

This then allows you to filter events and attributes based on their tag.

Tags are also very useful when looking to the API, as they enable you to extract all IOC’s with a specific tag.

MISP – Tags

To prepare for our enrichment, we will create two new tags.

- Feed-eCrimes – For our own intelligence
- Feed-RansomwareTracker – For the Ransomware Tracker feed

MISP - Tags

- Demo

eCrimeLabs

MISP - API

Alongside all of MISP’s great features within the WebUI, there is a very strong push for automation.

Most tasks that can be done within the GUI can also be done via the API.

We will not dig too deeply into all of the features of the API, but what we are really interested in, is the ability to request a list of IOC’s via the rest API.

MISP - API

You can work with the rest API via the WebUI, this gives you some flexibility in practicing with the type of queries you can make.

It also helps in preparing the right query to move over to your automation script later.

```
{"returnFormat":"text","type":"domain","tags":"Feed-eCrimes","to_ids":"yes"}
```

Ok great so we have threat feed data to enrich with….

- We have a threat feed source, which will handle our aggregation.
- Now we need to look at what we want to enrich…. LOGS!

eCrimeLabs

Logs logs logs logs...

- Logs are one of the most important cogs in an organisations security machine...
- Prevention is cool, but detection is sexy.
- Logging is the main part of detection, you cannot create an alarm without a source to look at!
- When you understand how to log properly, you will understand security!

Log examples...

A firewall should always log the source IP, destination IP, source port and destination port when it accepts or denies traffic.

A proxy server should always log the source IP, destination IP, destination domain name, destination port, http response code etc.

A DNS server should always log the source IP of the requestor and the answer returned.

Logging problems…

Although we have an expectation that each device type will log the same information, as we mentioned in the previous slide about the firewall logs.

We cannot expect that vendors in these areas will use the same field names for each item…

Cisco might log the destination IP field as “destip”

Checkpoint might call it “dst_ip”

Logging problems…

- Why is this a problem?

Example….

An organisation has a two vendor firewall strategy, where they use Checkpoint on the perimeter and Palo Alto for inside segmentation…

Lets use the whiteboard…

eCrimeLabs

Logging problems…

If each device vendor logs using different fields, correlation becomes a serious problem.

The ability to quickly search across multiple indexes for the same field is lost, pivoting off of the data becomes difficult!

It also makes it incredibly difficult to setup alerting, you will need to create multiple alerts to monitor the different field names. Even though you are looking for the same data…

Logging problems…

This problem gets even bigger if we want to correlate across different device types…

Proxy server vs Firewall vs DNS server

Normalization

To handle these issues it is necessary to normalize your data fields in your logs.

If you know you have common fields which show the same data, src.ip, source.ip, src_ip.

Remap it to a common one:

Nef.src

Normalization – Creating your own

You can create your own schema for this normalization, just make sure you are consistent across all logs…

Problem:-

New technicians working with your logs will need to be trained in your schema, if you want to open source any of your parsing or configuration later you will need to explain the schema used.

ElasticSearch – Common Schema

ElasticSearch recently launched their Elastic Common Schema.

This is a standard for normalization that will hopefully be adopted by the community.

Examples:

- source.ip
- Destination.ip

Logstash

We have focused a lot on the theory behind logging and getting consistent data types…

Now we should talk a little about a great tool to do this with…

To the whiteboard!

eCrimeLabs

Logstash

Logstash is part of the ELK stack, it is used to transform the data, enrich it and then transport it further into the chain for eventual indexing into ElasticSearch.

It has three main sections:

- Input – Modules for recieving the data, filebeat, syslog etc
- Filter – Modules for transforming, parsing, enriching
- Output – Modules for transporting the logs further

Logstash – Grok Parsing

```
if [type] == "squidlog" {
        grok {
                match => { "message" =>
"%{POSINT:timestamp}.%{WORD:timestamp_ms}\s+%{NUMBER:response_time}
%{IPORHOST:src.ip} %{NUMBER:src.port} %{WORD:squid_request_status}
%{NUMBER:http_status_code} %{NUMBER:reply_size_include_header}
%{WORD:http_method} %{NOTSPACE:requesturl} %{NOTSPACE:url_path}
%{NUMBER:http_version} %{WORD:squid_response} (?:-|%{IP:destination.ip})
%{NOTSPACE:content_type} %{NOTSPACE:destination.domain}"}
        }
```

Logstash – Grok Parsing

```
1552848816.636 281 192.168.0.3 46113 TCP_REFRESH_MODIFIED 200 1866 GET http://evilcorp.dk/ / 1.1 HIER_DIRECT 188.166.47.188 text/html evilcorp.dk
```

https://grokdebug.herokuapp.com/

OK great, now we have normalized logs and threat data feeds ready…

How do we combine these together and provide enrichment to our data?

Research!

Our requirements stated:-

Our solution must provide near realtime enrichment without delaying log ingestion, but also be able to handle large amounts of events per second.

In order to satisfy this requirement we need to understand the scale of what we are dealing with…

Evil Corp - Example

Evil Corp's firewalls and proxy server log approximately 5.000 events per second...

Each event contains fields we would like to lookup against our threat feed data and enrich before ingestion...

Logstash

Since this enrichment will be performed after normalization and before transport…

We should focus our research into Logstash enrichment options…

Research

ElasticSearch have created an interesting blog post, which touches briefly on how to do this type of enrichment…

https://www.elastic.co/blog/elasticsearch-data-enrichment-with-logstash-a-few-security-examples

eCrimeLabs

Logstash ElasticSearch filter

Logstash has a handy ElasticSearch filter…

The filter is used to query an elasticsearch index, if a match is found, then logstash can use its other capabilities to populate a new field for enrichment.

To use this for our solution, we would download the threat data feeds and then store them in separate indexes inside elasticsearch.

Each IOC would be assigned to a field, for example "ip" : "192.168.0.1"

Logstash ElasticSearch filter

This could be an interesting way to solve the problem… BUT

One of our requirements was to handle the dynamic nature of threat data feeds… If an IOC is removed from a feed and needs to be flushed from our ElasticSearch index we will need to perform this flushing ourselves.

We would need to monitor the feed for changes and then perform deletions on the index where necessary. This would add complexity…

Logstash Memcache Filter Plugin

Towards the bottom of the blog post, there is a section titled "Enrichment at scale"… Now we are talking…

Logstash has a plugin that allows it to lookup keys inside memcache and then work with the value returned. The solution can scale up to 200.000 lookups per second…

Since we know we need to work with around 5.000 events per second, this number suits us quite nicely.

Logstash Memcache Filter Plugin

From previous experience we also knew that memcache has the capability to age out keys/values stored there. By assigning a TTL to each key/value.

This would mean we wouldn’t need to add complexity to handle removing IOC’s, we can simply let them age themselves out.

Logstash filter - Decisions

We have two options, focus on the ElasticSearch filter or Memcache filter.

Since both options are going to require us to handle writing the threat data feed somewhere. But the ElasticSearch filter also requires us to handle removing IOC’s from the stored feeds.

Let’s look at memcache instead.

eCrimeLabs

Memcached

Ring any bells?

https://www.zdnet.com/article/memcached-ddos-the-biggest-baddest-denial-of-service-attacker-yet/

Memcached

Memcached is a memory caching system, it is similar to Redis.

- It is a simple open source key/value store working in memory.
- You put a key into the store and assign it a value.
- It is built for speed and efficiency.
- It can be run on both Windows and Linux.

Memcached

Memcached is very simple and allows you to telnet into the application. The default port for Memcached telnet is 11211.

After telnetting in, you can interact with the application. For our purposes we are interested in only the get and set capabilities.

Get – Lookup a key and return its value

Set – Create a key and assign a value

Memcached

Since you can assign a TTL to each key inside Memcached. This makes it perfect to meet our requirement about dealing with the dymanic nature of threat data feeds.

If an IOC is removed from the feed, we do not want a complex solution to monitor the feed and then remove that IOC from memcached too.

If we set a TTL on the key in Memcached, we can have it aged out instead.

Memcached

Now we have a place to store the threat feed data, how should we structure it inside memcached.

An IOC stored in MISP has a datatype, lets limit ourselves to a couple of these for this design…

- Hostname
- Domain

Memcached - Structure

Example:-

Bbc.co.uk could be listed in MISP as a hostname or a domain name. It could even feature on both datatype lists.

When we query the MISP API, we are specific about which datatype we want.

We then push this to memcache…. If an IOC is on both datatype lists, we will end up only having one in memcache. Usually the most recently pushed!

eCrimeLabs

Memcached - TTL

Since we need to handle the dynamic nature of threat data feeds, and the need to remove IOC’s as quickly as possible.

We need to decide on a TTL for each key/value in memcached.

This will need to be aligned with how often we run our script to populate memcached.

If the script runs every 60 seconds, then our TTL should be set to 110 seconds.

This would allow the script to fail once and not empty the memcache store.

Do you agree?

Python - Memcached

Now we have decided on how the data should be structured inside the memcache store, we need to get the data in there.

There are a number of Python libraries that can be used for interacting with Memcache.

They all provide very similar functionality, we chose pymemcache.

So lets focus on this…

Python - Memcached

```
def misppull():
    headers = {
        'Authorization': 'JXa5yIRYU5qPSxiAJSr1Ju5M6bDQkJ1HTMUTX3ow',
        'Accept': 'application/json',
        'Content-type': 'application/json',
         }

    data = '{"returnFormat":"text","type":"domain","tags":"Feed-eCrimes","to_ids":"yes"}'

    response = requests.post('http://192.168.0.39/attributes/restSearch', headers=headers, data=data)
    return response
```

Python - Memcached

```
if __name__ == '__main__':
    response = misppull()
    domains = (response.text).splitlines()
    for domain in domains:
            client.set("domain-" + domain, "Feed-eCrimes", 120)
```

Enrich enrich enrich…

Now we have our threat data feeds stored into memcached…

We need to return to our logstash memcached filter…

The module is titled ”memcached”…

Logstash – Memcached lookup

```
memcached{
        hosts => ["127.0.0.1:11211"]
        get => {
            "domain-%{destination.domain}" => "[misp_src]"
            }
    }
```

eCrimeLabs

Logstash – Memcached lookup

Remember the normalization fields from earlier?

We are peforming the memcache lookup against destination.domain field.

This can be run against multiple log sources, providing they are mapped against ECS.

If the lookup provides a match, logstash will write the value to the misp_src field.

Logstash – Memcached lookup

If the lookup against memcache does not return a result, then the misp_src field will not be created.

This will result in a problem later when we want to visualize the data in Kibana.

Kibana's index templates will only allow us to work with fields that exist in our index over time. Lets do this on the whiteboard…

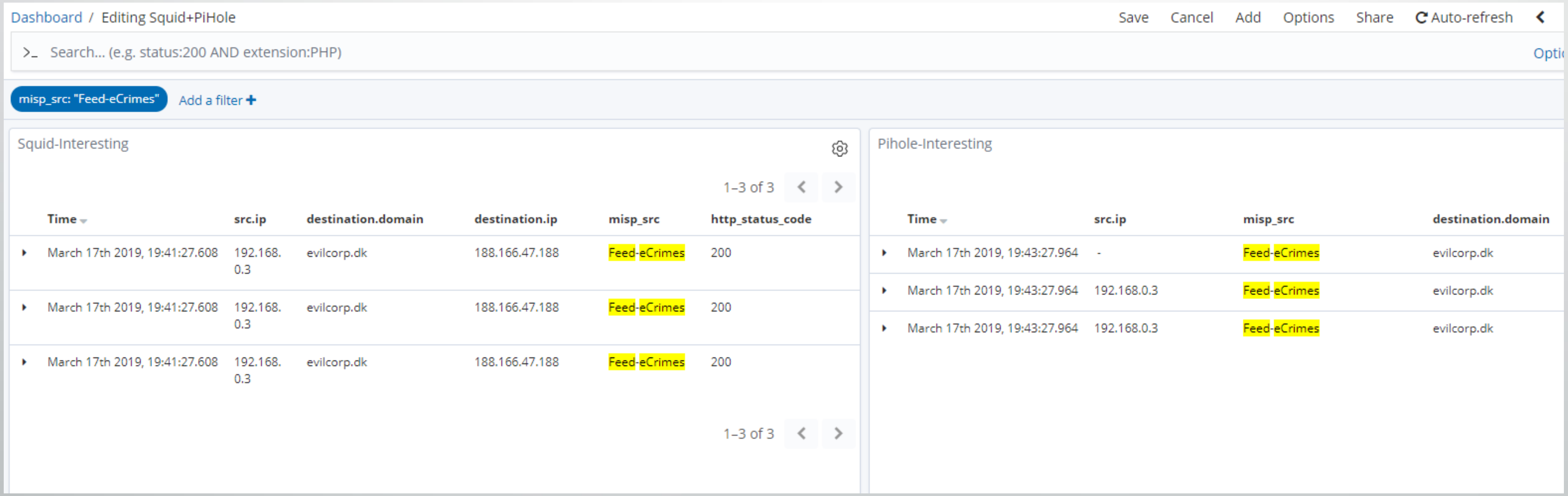
We need to create the misp_src field and assign the value "none" if we do not get a match against memcache.

eCrimeLabs

Logstash – Memcached lookup

```
if ![misp_src]
    {
        mutate
        {
            add_field=> {"[misp_src]" => "none"}
        }
    }
```

Putting it all together…



Dashboard / Editing Squid+PiHole

Squid-Interesting

1–3 of 3

Time	src.ip	destination.domain	destination.ip	misp_src	http_status_code
March 17th 2019, 19:41:27.608	192.168.0.3	evilcorp.dk	188.166.47.188	Feed-eCrimes	200
March 17th 2019, 19:41:27.608	192.168.0.3	evilcorp.dk	188.166.47.188	Feed-eCrimes	200
March 17th 2019, 19:41:27.608	192.168.0.3	evilcorp.dk	188.166.47.188	Feed-eCrimes	200

1–3 of 3

Pihole-Interesting

Time	src.ip	misp_src	destination.domain
March 17th 2019, 19:43:27.964	-	Feed-eCrimes	evilcorp.dk
March 17th 2019, 19:43:27.964	192.168.0.3	Feed-eCrimes	evilcorp.dk
March 17th 2019, 19:43:27.964	192.168.0.3	Feed-eCrimes	evilcorp.dk

Demo demo demo

Squid + PiHole data…