

Create ElastAlert index pattern

1 Create index pattern

★ *packetbeat*

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations. ☐ Include system indices

Step 1 of 2: Define index pattern

Index pattern

elastalert

You can use a * as a wildcard in your index pattern. You can't use spaces or the characters \, /, ?, *, <, >, |.

✓ Success! Your index pattern matches 2 indices.

elastalert_status

elastalert_status_status

Include system indices

Step 2 of 2: Configure settings

You've defined *elastalert* as your index pattern. Now you can specify some settings before we create it.

Time Filter field name Refresh

@timestamp

The Time Filter will use this field to filter your data by time. You can choose not to have a time field, but you will not be able to narrow down your data by a time range.

> Show advanced options

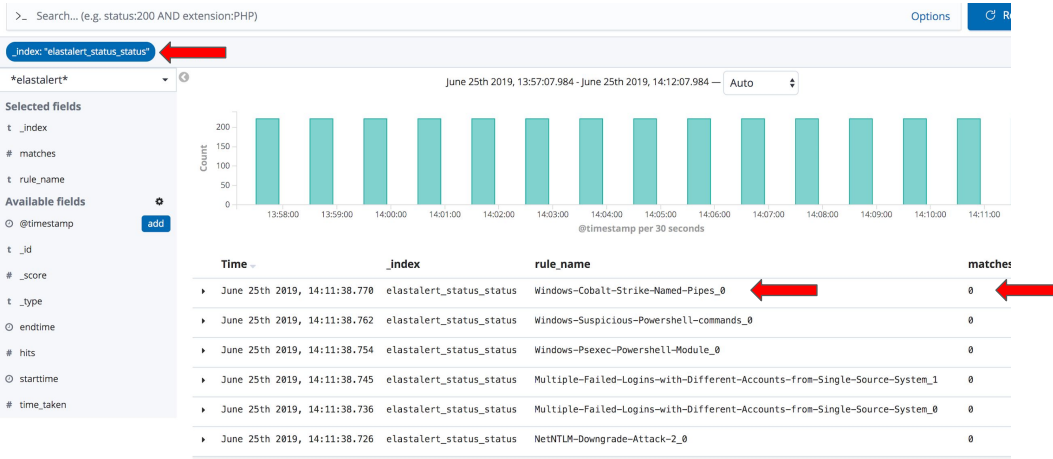
4 Create index pattern

We are going to create an ElastAlert rule to trigger on long command lines from Sysmon's process create event. To generate alerts we are going to import logs from Cyb3rWard0g's Mordor project (<https://github.com/Cyb3rWard0g/mordor>).

First we need to create an index pattern to be able to view the ElastAlert logs in Kibana. Open Kibana either from the shortcut on the CentOS machine or by visiting <http://<CentOS IP>:5601/> from your host machine.

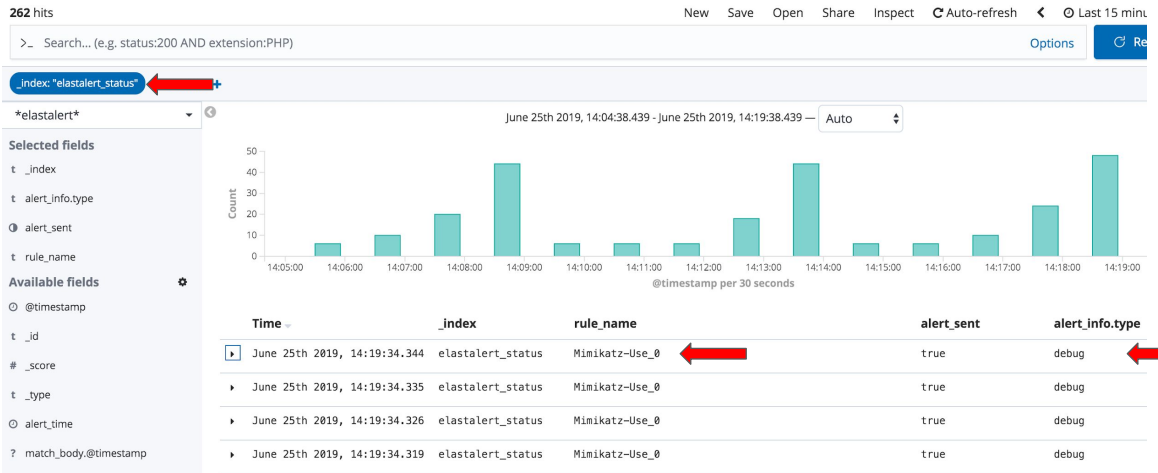
- 1) Click on Create index pattern
- 2) Type *elastalert* then click Next Step
- 3) Select @timestamp from the dropdown
- 4) Finally click, Create index pattern

ElastAlert Status index



Once the index pattern is created we can use to the discover tab to view the Elastalert logs. The ElastAlert Status writeback index contains information on each rule that runs and the number of matches and hits that were returned. In the screenshot above we see the rule_name of the Sigma rules and the number of matches that were returned when the rule was run.

ElastAlert index



The Elastalert writeback index is a log of information about every alert triggered. It contains the full log(s) that tripped the alert and details about the alert that was generated.

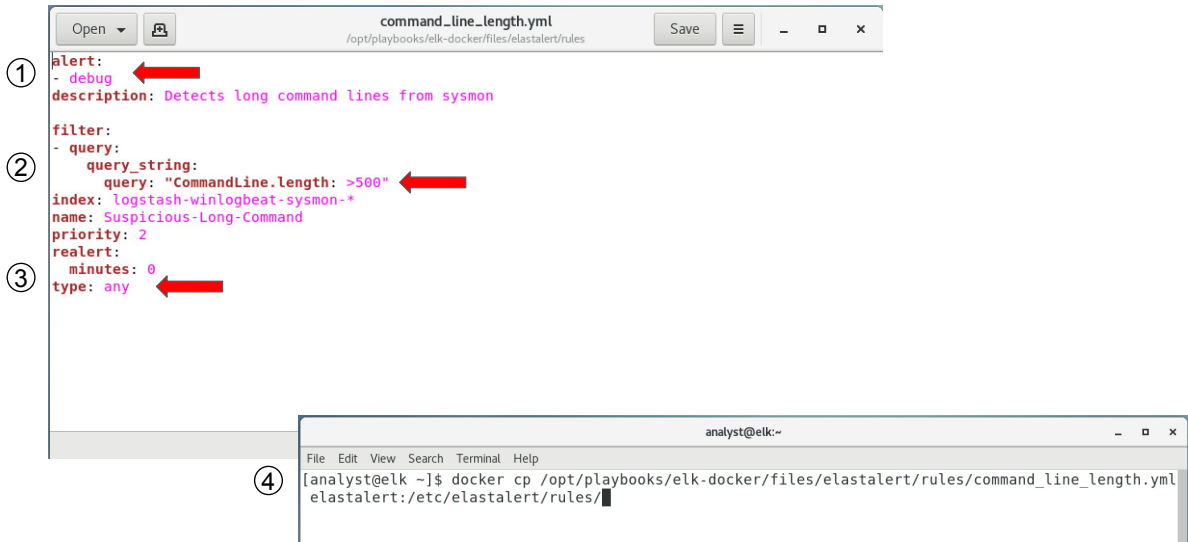
ElastAlert rules directory



```
analyst@elk ~]$ elastalert
analyst@elk ~]$ cd rules
$ pwd
/etc/elastalert/rules
$ ls
helk_all_psexec_psh.yml
helk_all_susp_powershell_commands.yml
helk_sysmon_cobalt_strike_msagent.yml
sigma_av_exploiting.yml
sigma_av_password_dumper.yml
sigma_av_relevant_files.yml
sigma_av_webshell.yml
sigma_win_netsh_fw_add.yml
sigma_win_netsh_port_fwd_3389.yml
sigma_win_netsh_port_fwd.yml
sigma_win_office_shell.yml
sigma_win_office_spawn_exe_frc
sigma_win_overpass_the_hash.yml
sigma_win_pass_the_hash.yml
```

- 1) In the terminal, run the elastalert command alias to open a shell in the Elastalert container
- 2) Change directories into the rules folder
- 3) Run ls to view all the rules Elastalert is configured for. All of the rules were generated from Sigma (<https://github.com/Neo23x0/sigma>) and added to the Elastalert container for you.

Copy ElastAlert command line rule into container




In the terminal, open the rule we are going to add by running:

`sudo vi /opt/playbooks/elk-docker/files/elastalert/rules/command_line_length.yml`

`sudo gedit /opt/playbooks/elk-docker/files/elastalert/rules/command_line_length.yml`

- 1) For this lab we are going to set the alert field to debug, but ElastAlert has many alerting integrations including: e-mail, Jira, theHive, Opsgenie, and many more
- 2) The alert is using our command line length field we created in the earlier lab. In this case, we are looking for commands that are greater than 500 characters
- 3) We'll use any for the alert type which will generate an alert for every hit the query returns. There are several other rule types Elastalert supports as well: <https://elastalert.readthedocs.io/en/latest/ruletypes.html#rule-types>
- 4) Run the following command to copy the rule into the Elastalert container:
`docker cp /opt/playbooks/elk-docker/files/elastalert/rules/command_line_length.yml elastalert:/etc/elastalert/rules/`

Add command line length to Mordor Logstash pipeline



The screenshot shows a terminal window at the top with the prompt `analyst@elk:~` and a menu bar with `File Edit View Search Terminal Help`. Below the terminal is a gedit editor window titled `30-filter.conf` with the path `/opt/elk-siem/logstash/pipeline/mordor-logs`. The editor contains the following Ruby code:

```
filter {  
  if [event_id] == 1 and [source_name] == "Microsoft-Windows-Sysmon"  
  {  
    ② ruby {  
      code => "event.set('[CommandLine][length]', event.get('[event_data][CommandLine]').length)"  
    }  
  }  
}
```

The code is annotated with circled numbers: ① points to the terminal command `sudo gedit /opt/elk-siem/logstash/pipeline/mordor-logs/30-filter.conf`, and ② points to the `ruby` block in the configuration file.

- 1) Create a new filter conf file in the Mordor pipeline to add the command line length field:
`sudo vi /opt/elk-siem/logstash/pipeline/mordor-logs/30-filter.conf` or
`sudo gedit /opt/elk-siem/logstash/pipeline/mordor-logs/30-filter.conf`
- 2) Paste the following contents into the file to add the new field with Ruby, double check the text pasted correctly into the conf file:

```
filter {  
  if [event_id] == 1 and [source_name] == "Microsoft-Windows-Sysmon"  
  {  
    ruby {  
      code => "event.set('[CommandLine][length]',  
event.get('[event_data][CommandLine]').length)"  
    }  
  }  
}
```

Generate logs and enable the Mordor pipeline

analyst@elk:~

File Edit View Search Terminal Help

① [analyst@elk ~]\$ sudo /opt/elk-siem/logstash/mordor-logs/current-date.sh

*pipelines.yml
/opt/elk-siem/logstash/config

Save

```
# This file is where you define your pipelines. You can define multiple.
# For more information on multiple pipelines, see the documentation:
# https://www.elastic.co/guide/en/logstash/current/multiple-pipelines.ht

- pipeline.id: winlogbeat
  path.config: "/usr/share/logstash/pipeline/winlogbeat"
- pipeline.id: filebeat
  path.config: "/usr/share/logstash/pipeline/filebeat"
- pipeline.id: packetbeat
  path.config: "/usr/share/logstash/pipeline/packetbeat"
- pipeline.id: elk-logs
  path.config: "/usr/share/logstash/pipeline/elk-logs"
- pipeline.id: splunk
  path.config: "/usr/share/logstash/pipeline/splunk"
- pipeline.id: nxlog
  path.config: "/usr/share/logstash/pipeline/nxlog"
- pipeline.id: mordor-logs
  path.config: "/usr/share/logstash/pipeline/mordor-logs"
- pipeline.id: ad-enrichment
  path.config: "/usr/share/logstash/pipeline/ad-enrichment"
- pipeline.id: snmp
  path.config: "/usr/share/logstash/pipeline/snmp"
```

②

We will be using a few datasets from Cyb3rWard0g's Mordor project (<https://github.com/Cyb3rWard0g/mordor>). The Mordor data can be used to simulate adversarial techniques using JSON documents.

- 1) Run `sudo /opt/elk-siem/logstash/mordor-logs/current-date.sh` to import mordor logs. This current replaces the date in the Mordor data set with the current date.
- 2) Remove the comment (#) from the Mordor pipeline section in the pipelines.yml to enable it: `sudo vi /opt/elk-siem/logstash/config/pipelines.yml` or `sudo gedit /opt/elk-siem/logstash/config/pipelines.yml`

Verify the mordor-logs pipeline is running

The screenshot displays the Kibana Logstash monitoring dashboard. On the left, a sidebar contains navigation links: APM, Uptime, Dev Tools, Monitoring (highlighted with a circled 1), and Management. Below this is a secondary sidebar with links: Canvas, Maps, Machine Learning, Infrastructure, Logs, APM, Uptime, Dev Tools, Monitoring (highlighted), and Management. The main content area is titled 'Logstash' and features three summary cards at the top: 'Overview' (Events Received: 21.5k, Events Emittted: 21.5k), 'Nodes: 1' (Uptime: 20 minutes, JVM Heap: 72.56%, 359.1 MB / 494.9 MB), and 'Pipelines: 9' (With Memory Queues: 9, With Persistent Queues: 0, highlighted with a circled 2). Below these cards is a table of pipelines, sorted by 'Events Emittted' in descending order. The pipelines listed are: ad-enrichment, elk-logs, filebeat, mordor-logs (highlighted with a circled 3), nxlog, packetbeat, snmp, splunk (highlighted with a circled 4), and winlogbeat. The 'mordor-logs' pipeline is expanded, showing a filter rule: 'if [event_id] == 1 and [source_name] == "Microsoft-Windows-Sysmon"', with a 'ruby' filter type, 0% completion, N/A status, and 0 e/s received.

- 1) Click on Monitoring in Kibana
- 2) Under logstash click Pipelines,
- 3) Then click on mordor logs
- 4) Verify the Ruby filter we added shows up in the mordor logs pipeline

View the Elastalert index for alerts

The screenshot shows the Kibana interface with the Discover tab selected. The search bar contains the query `rule_name: "Suspicious-Long-Command" _index: "elastalert_status"`. The left sidebar shows the navigation menu with 'Discover' highlighted. The main view displays a histogram of event counts over time, with a peak at 22:45:00. Below the histogram, a table of events is shown, with the first row highlighted. The table columns are Time, _index, alert_sent, match_body.CommandLine.length, and match_body.host.name.

Time	_index	alert_sent	match_body.CommandLine.length	match_body.host.name
June 25th 2019, 22:45:57.933	elastalert_status	true	5432	WECServer
June 25th 2019, 22:45:57.925	elastalert_status	true	5483	WECServer
June 25th 2019, 22:45:57.915	elastalert_status	true	5450	WECServer

- 1) Click on Discover in Kibana
- 2) Add a filter for rule_name matching Suspicious-Long-Command
- 3) Add another filter for _index matching elastalert_status, which contains all the tripped Elastalert rules
- 4) Click the arrow next to the event to view the full alert details