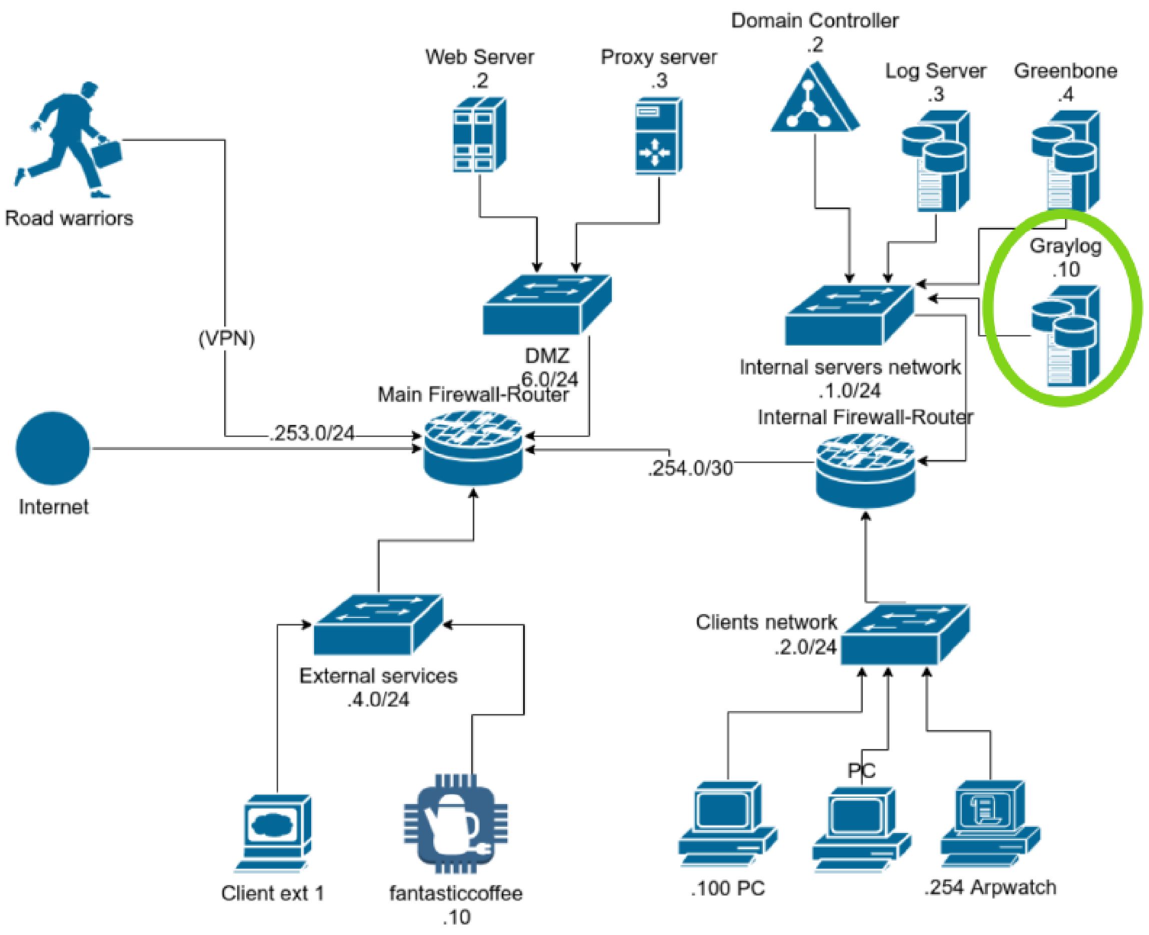
Assignment 3 Group [redacted]
Student names and numbers:

- [redacted]
- [redacted]
- [redacted]
- [redacted]

The network infrastructure in which we will perform the assignments requested is the following:

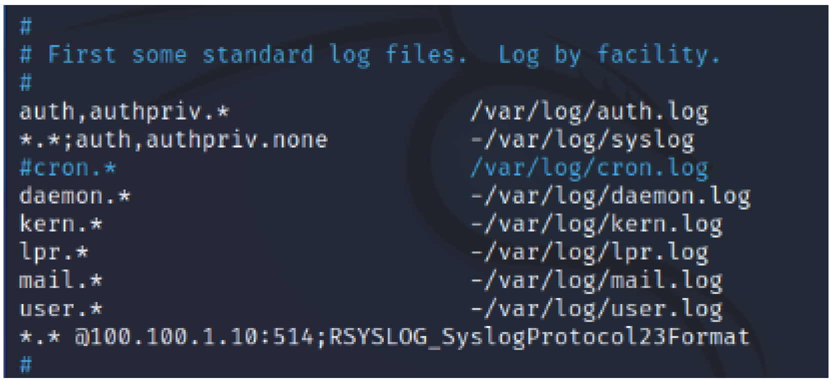


The host graylog is up and running in the internal server network. We can access its web interface using the link http://100.100.1.10:9000, with the credentials admin:PND2021-22.

Log collection from the topology of the network.

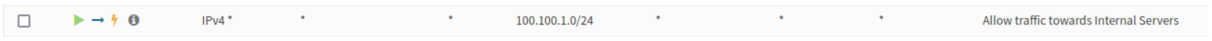
In the second assignment we already set most of the network to send the logs to the Graylog machine. Those logs we managed by the rsyslog service that was just storing them in /var/log. We now want the graylog service itself to manage those logs by listing and analyzing them in his web interface.
To do so we should tell the hosts of the network to send the logs again to the Graylog server on another port (since the 514 is already used by rsyslog to store them locally). Since we want to minimize the amount of traffic in the network and one log is already sent twice (one towards the log server and one towards Graylog server, both on the port 514), we can simply

reuse the ones already sent. Graylog has already stored locally in */var/log* the logs of almost everyone in the network so we can use them.
The first thing we need to do is to edit the CLIENTS hosts configuration so they can send the logs towards Graylog (since this subnet was the one missing on sending logs due to the assignment 2). Modify /etc/rsyslog.conf of both kali.acme and arpwatch like so:



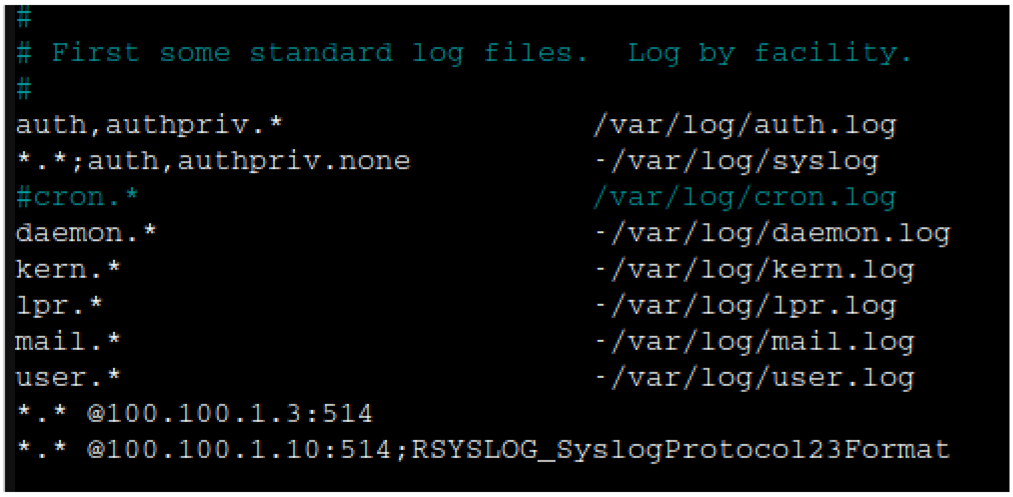
```
#
# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log
*.* @100.100.1.10:514;RSYSLOG_SyslogProtocol23Format
#
```

and restart the service with *sudo systemctl restart rsyslog.service*.
The above text line **.* @100.100.1.10:514:RSYSLOG_SyslogProtool23Format* added simply tells the host to send every log to 100.100.1.10 (Graylog server) on the port 514 using a common graylog UDP format (otherwise Graylog will not be able to elaborate the data received from the hosts). We chose the UDP protocol because it's faster than the TCP (and for real time monitoring, the freshness of the logs can be essential) and we didn't want to congest the network with handshakes for every log sent.
Note that the CLIENTS hosts just send the logs to Graylog and not to the Log server because this was the request of the assignment 2 and we didn't want to alter too much the previous requests of the past assignments.
We don't need to add a new firewall rule on the CLIENTS interface since in the previous assignment we allowed all the traffic from CLIENTS towards the Internal Servers:



☐	▶ → ϟ ⓘ	IPv4 *	*	*	100.100.1.0/24	*	*	*	Allow traffic towards Internal Servers

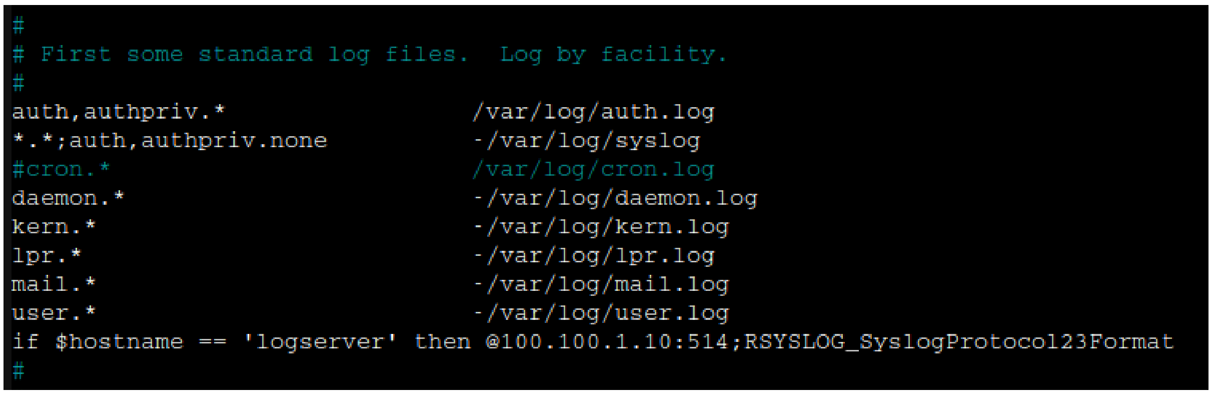
For both the other two subnets (DMZ and EXTERNAL_CLIENTS), they already send the logs to both the Log server and Graylog (due to a previous request in assignment 2). The only thing we need to do is to specify the protocol the hosts have to send logs. This can be done in the following way by editing */etc/rsyslog.conf*:



```
#
# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log
*.* @100.100.1.3:514
*.* @100.100.1.10:514;RSYSLOG_SyslogProtocol23Format
```

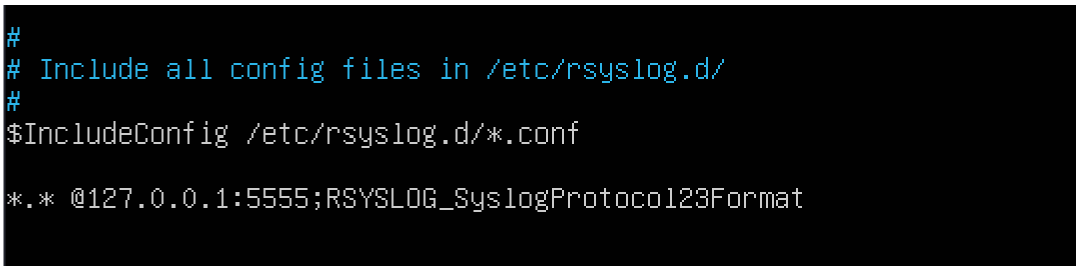
Basically the same as it was in the assignment 2 but the difference is the protocol added (needed for the Graylog service to elaborate the data).

Since we would like to monitor even the logs of the Log server itself on Graylog, then we have to tell the Log server to send its personal logs only, to Graylog: to do so, simply add the following line in its */etc/rsyslog.conf*:



```
#
# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log
if $hostname == 'logserver' then @100.100.1.10:514;RSYSLOG_SyslogProtocol23Format
#
```

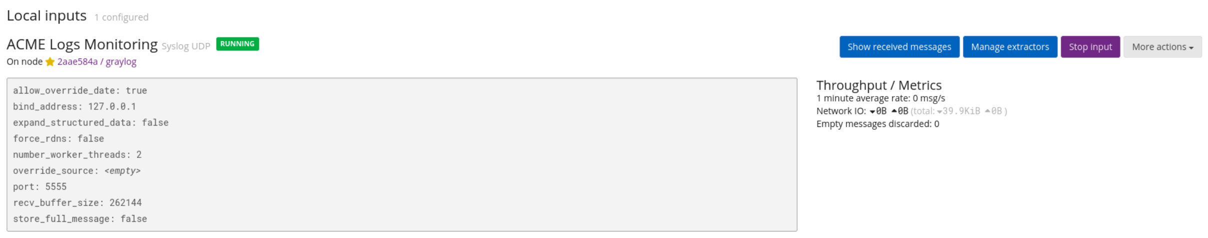
The above line tells the Log server to send every log whose hostname is “logserver” to Graylog (so basically he will send just his logs among all the ones he receives from all the ACME network).
Now that the hosts are ready, we can configure our Graylog server. The Graylog service cannot work on port 514 because rsyslog is already working there by receiving all the logs of the network, so we have to tell the Graylog server to send all the logs stored in /var/log to himself in the port 5555 (where the Graylog service will be listening).
To do so, simply add the line **.* @127.0.0.1:5555;RSYSLOG_SyslogProtocol23Format* in his */etc/rsyslog.conf*:



```
#
# Include all config files in /etc/rsyslog.d/
#
$IncludeConfig /etc/rsyslog.d/*.conf

*.* @127.0.0.1:5555;RSYSLOG_SyslogProtocol23Format
```

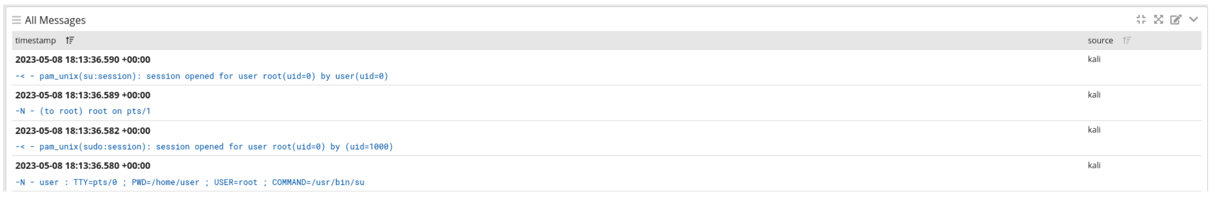
Now visit the Graylog web page on 100.100.1.10 and on System->Inputs add one Syslog UDP input to listen on the port 5555 of 127.0.0.1. This lets Graylog take all the logs arriving at the port 5555 of the server, elaborate them and store them on an elasticsearch database.
When the input is up and running we should see something like this:


Local inputs 1 configured
ACME Logs Monitoring Syslog UDP RUNNING
On node 2aae584a / graylog
Show received messages
Manage extractors
Stop input
More actions

```
allow_override_date: true
bind_address: 127.0.0.1
expand_structured_data: false
force_rdns: false
number_worker_threads: 2
override_source: <empty>
port: 5555
recv_buffer_size: 262144
store_full_message: false
```

Throughput / Metrics
1 minute average rate: 0 msg/s
Network IO: ▾0B ▴0B (total: ▾39.9KiB ▴0B)
Empty messages discarded: 0

Note that we are already receiving logs, in fact in the right part of the image we can see a total of 39.9 KiB received.
To test that all is working as it should, we simply generated a log event on kali.acme by requesting a root session or by writing some random string on the logs with the command *logger “text”*; then in the Graylog web page we switched to the search bar and updated the message section. This is the result:



All Messages

timestamp	source
2023-05-08 18:13:36.590 +00:00	kali

```
-< - pam_unix(su:session): session opened for user root(uid=0) by user(uid=0)
```

timestamp	source
2023-05-08 18:13:36.589 +00:00	kali

```
-N - (to root) root on pts/1
```

timestamp	source
2023-05-08 18:13:36.582 +00:00	kali

```
-< - pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
```

timestamp	source
2023-05-08 18:13:36.580 +00:00	kali

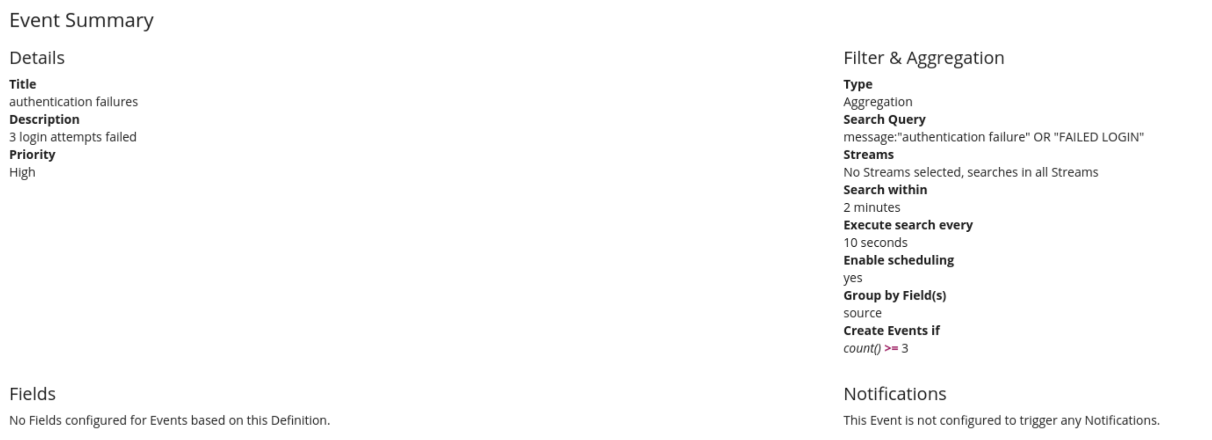
```
-N - user : TTY=pts/0 ; PWD=/home/user ; USER=root ; COMMAND=/usr/bin/su
```

We checked in the same way that the logs of every machine are correctly visualized in the Graylog interface.

Set up the most suitable alerts and dashboards to properly monitor the systems running.

We added some alerts that are useful to monitor the security of the network. The alerts are the following:

Authentication failures:



Event Summary

Details

Title
authentication failures
Description
3 login attempts failed
Priority
High

Filter & Aggregation

Type
Aggregation
Search Query
message:"authentication failure" OR "FAILED LOGIN"
Streams
No Streams selected, searches in all Streams
Search within
2 minutes
Execute search every
10 seconds
Enable scheduling
yes
Group by Field(s)
source
Create Events if
count() >= 3

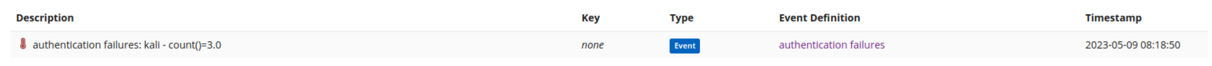
Fields

No Fields configured for Events based on this Definition.

Notifications

This Event is not configured to trigger any Notifications.

This alert simply alerts us when there is a number greater or equal than 3 of failed login attempts in an interval time of 2 minutes for every single user of the network. This is useful because this event may suggest an occurring brute force attack (in fact we gave it a high priority). To test this, we failed an host authentication for 3 times consecutively and, as we can see in the image below, it triggered the alert for the specific host:

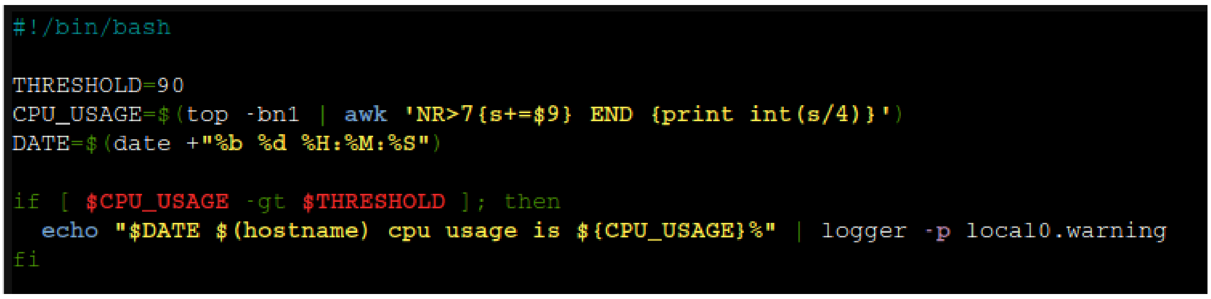


Description	Key	Type	Event Definition	Timestamp
authentication failures: kali - count()=3.0	*none*	Event	authentication failures	2023-05-09 08:18:50

CPU usage:

We wanted to monitor the cpu usage of the machines that in our minds are the most critical ones: the webserver and the logserver. To do so we need an alert that warns us if the CPU of one of the two servers above has reached a threshold set by us. Since there is not a default Unix log that gets written in the syslog file if this happens, we need to create a bash script that periodically checks for the CPU usage and, if there is some problem, it writes a warning log.

So we first coded the following bash script:

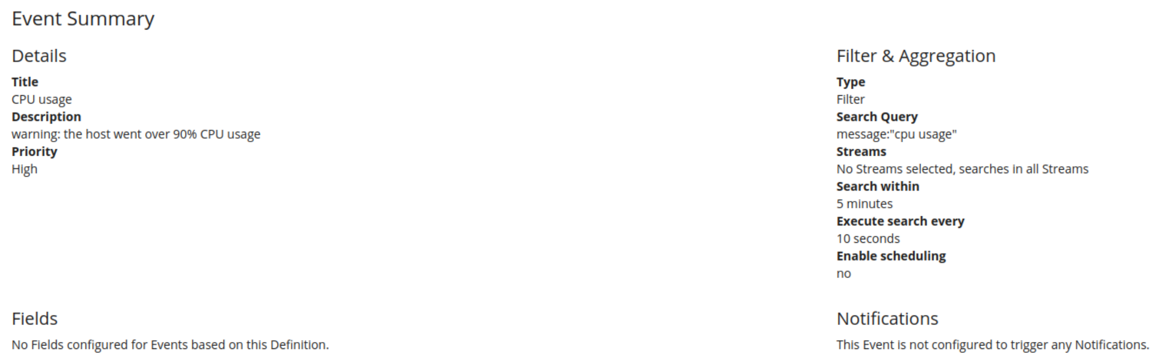


```
#!/bin/bash

THRESHOLD=90
CPU_USAGE=$(top -bn1 | awk 'NR>7{s+=$9} END {print int(s/4)}')
DATE=$(date +"%b %d %H:%M:%S")

if [ $CPU_USAGE -gt $THRESHOLD ]; then
  echo "$DATE $(hostname) cpu usage is ${CPU_USAGE}%" | logger -p local0.warning
fi
```

and inserted it in */usr/local/bin* of the webserver and the log server. This script, when executed, checks for the CPU usage of the machine and if this value is above the threshold of 90%, then it prints a warning log in the syslog file. We added this script to */etc/crontab* so that it is executed every 5 minutes. Now we just need to create the associated alert on Graylog:



Event Summary

Details

Title
CPU usage
Description
warning: the host went over 90% CPU usage
Priority
High

Filter & Aggregation

Type
Filter
Search Query
message:"cpu usage"
Streams
No Streams selected, searches in all Streams
Search within
5 minutes
Execute search every
10 seconds
Enable scheduling
no

Fields

No Fields configured for Events based on this Definition.

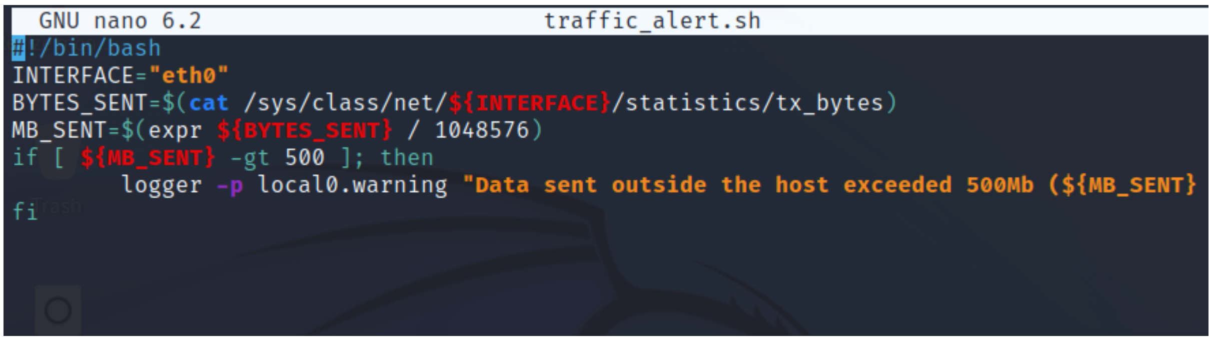
Notifications

This Event is not configured to trigger any Notifications.

The rule above searches for the text string: ”cpu usage” in the logs received in the last 5 minutes. Since that’s exactly what the script writes in the log, as soon as the warning CPU log arrives at Graylog, the alert gets triggered. For instance if the web server is having an high CPU usage which may affect his performances or may indicate some occurring attacks (for example a DOS), we could promptly actuate a response.
The above passages have been executed on the log server too so that his CPU usage can be monitored as the webserver one.
To test that this alert works, we momentarily change the threshold to 0% so that as soon as the script is executed, the log is written in the syslog file, sent to Graylog and the alert pops out.

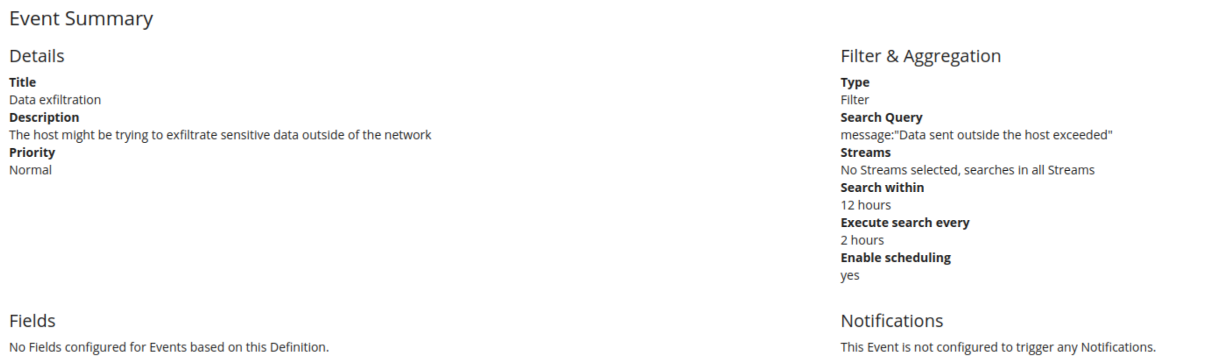
Data exfiltration

We wanted to check if some host in the EXTERNAL_CLIENTS or in the CLIENTS subnet might be executing a data exfiltration to let some sensitive files out of the ACME network. To do so, we set an ideal threshold of 500Mb that every single host could potentially send out without triggering any alert. As soon as this limit is overcome, a daily script running on every host writes a warning log on syslog. The script is the following:



```
  GNU nano 6.2                              traffic_alert.sh
#!/bin/bash
INTERFACE="eth0"
BYTES_SENT=$(cat /sys/class/net/${INTERFACE}/statistics/tx_bytes)
MB_SENT=$(expr ${BYTES_SENT} / 1048576)
if [ ${MB_SENT} -gt 500 ]; then
        logger -p local0.warning "Data sent outside the host exceeded 500Mb (${MB_SENT}
fi
```

and it has been inserted in /usr/local/bin and executed daily by crontab in every host of EXTERNAL_CLIENTS and CLIENTS. The script above checks the *tx_bytes* file in the ETH0 interface of the host on which it's running, converts the value in Mb and compares it with the threshold limit. Once the script is up and running on every host, we simply added the alert on Graylog that searches for the string "Data sent outside the host exceeded" (what actually the script writes on the log as message) every two hours on the log set of the last 12 hours.



Event Summary

Details

Title
Data exfiltration
Description
The host might be trying to exfiltrate sensitive data outside of the network
Priority
Normal

Filter & Aggregation

Type
Filter
Search Query
message:"Data sent outside the host exceeded"
Streams
No Streams selected, searches in all Streams
Search within
12 hours
Execute search every
2 hours
Enable scheduling
yes

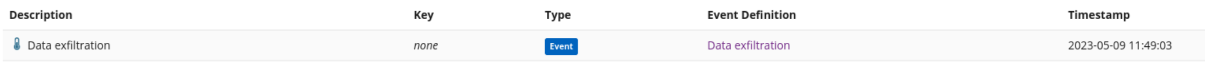
Fields

No Fields configured for Events based on this Definition.

Notifications

This Event is not configured to trigger any Notifications.

To test this alert, we modified momentarily the script on one of the hosts by putting the threshold to 0 (so it gets triggered immediately as soon as the script it's executed). Then we executed the script manually and we verified that the alert was created on Graylog.

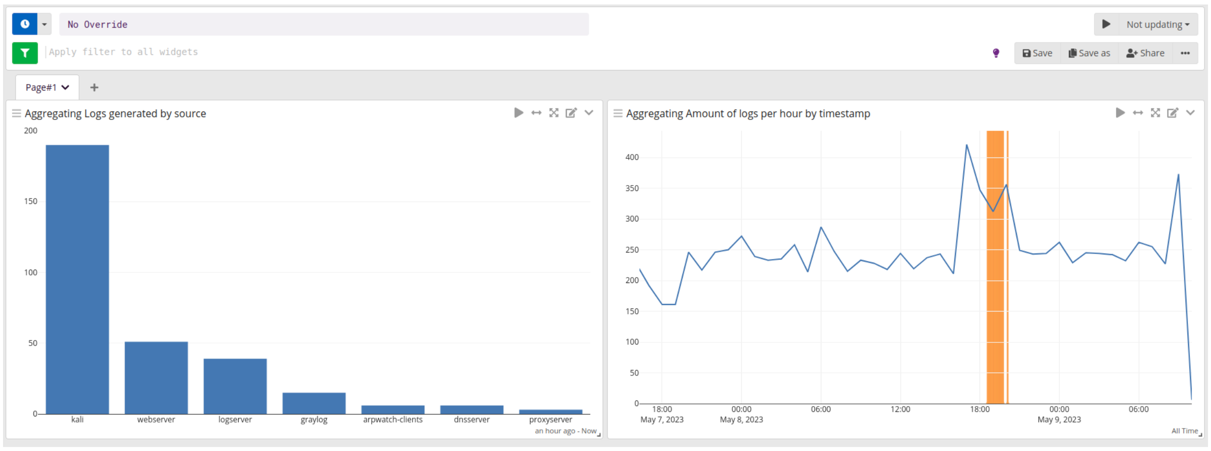


Description	Key	Type	Event Definition	Timestamp
Data exfiltration	*none*	Event	Data exfiltration	2023-05-09 11:49:03

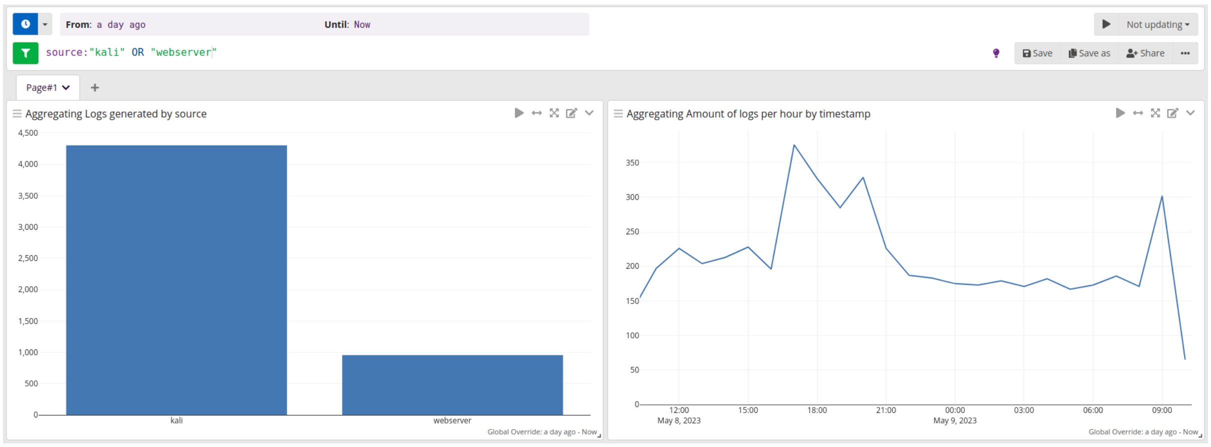
The dashboards created are the following:

Top source and network logs activity

We created a dashboard that helps an eventual IT department to understand who are the most active hosts in the network (in terms of logs sending) and at what hours the logs are mostly sent to both the logs servers. The dashboard is the following:



On the left we can see a bar chart which has the hostnames on the x-axis and the amount of logs generated by that specific host on the y-axis. This bar chart is built on the logs set collected by Graylog in the last hour but this option can be easily changed in the time bar at the top left of the page (which is now set on “No Override”, that means that it is taking the default time set on which we created the graph). On the right there is a line chart with the time on the x-axis and the logs generated in an interval of 1 hour on the y-axis: this lets us see the activity hour by hour of the entire network. Note that in this graph there are orange vertical lines that represent the alerts we created and occurred in the network (so it is useful to visualize at what hour of the day they occurred and verify if there is some pattern that hides a malicious activity behind them). Obviously as for the left chart, the time set of the logs can be modified. We can even apply the graphs on specific subsets of logs by making a query in the search bar on the top left side of the screen: this allows us for example to build these graphs only for the sources kali or webserver in the last day of activity of the network and this is the result:



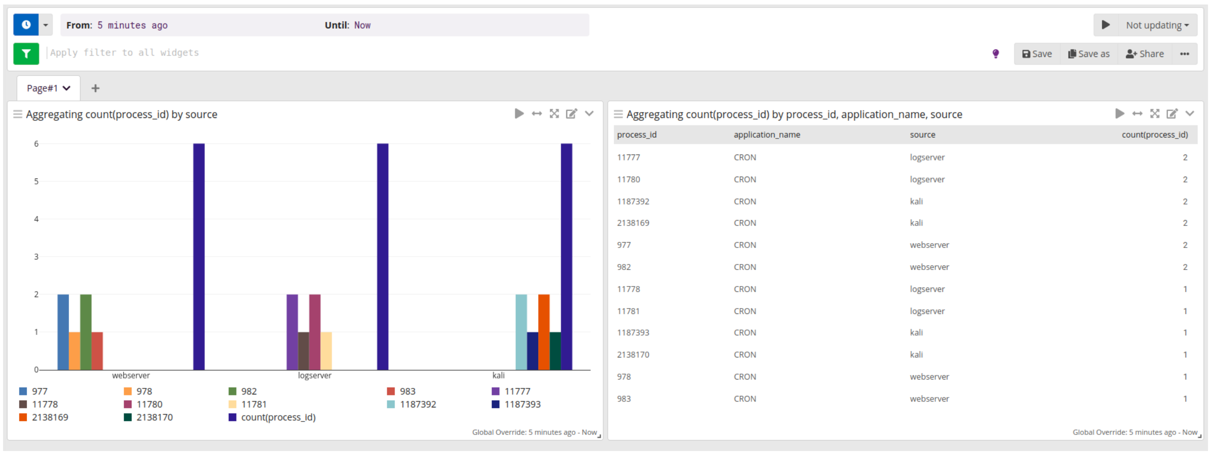
From: a day ago Until: Now

```
source:"kali" OR "webserver"
```

Not updating
Save
Save as
Share
Page#1
Aggregating Logs generated by source
kali
webserver
Global Override: a day ago - Now
Aggregating Amount of logs per hour by timestamp
May 8, 2023
May 9, 2023
Global Override: a day ago - Now

Processes IDs

We created a dashboard that allows an eventual SOC analyst to monitor the processes ids. The dashboard is the following:



Aggregating count(process_id) by process_id, application_name, source

process_id	application_name	source	count(process_id)
11777	CRON	logserver	2
11780	CRON	logserver	2
1187392	CRON	kali	2
2138169	CRON	kali	2
977	CRON	webserver	2
982	CRON	webserver	2
11778	CRON	logserver	1
11781	CRON	logserver	1
1187393	CRON	kali	1
2138170	CRON	kali	1
978	CRON	webserver	1
983	CRON	webserver	1

Global Override: 5 minutes ago - Now

On the left we have a bar chart which counts the number of logs generated by every process id of every host and on the right we have a table which associates every process id with his application name and source. This is useful because in case there is a malware on some host, it might generate a high amount of logs. So we can immediately see it, recognize its process id and perform an action against that specific malicious process.