Assignment 3 Report
ACME-6

Francesco Aquino (1851954),
Michele Kryston (1844733),
Ralph Angelo Tancio Almoneda (1837040),
Simone Di Valerio (1835412)

Practical Network Defense, Cybersecurity Master Degree
Sapienza University of Rome
Second Semester 2022/2023

Contents

1 Initial brainstorming

The assignment required us to configure a brand new SIEM running on the ACME, to do so we first did a few rounds of research on the matter. To carry out the task, as stated by the Professor, we used Graylog which analyses and sorts out all the logs that we collect from the hosts of the network. We took into account the suggestion which told us to use a port with a number greater than 1023 for the Graylog log listener to avoid root privilege troubles, and we decided to use port 5140 as opposed to the standard one, which is 514.

2 Implementation & Tests

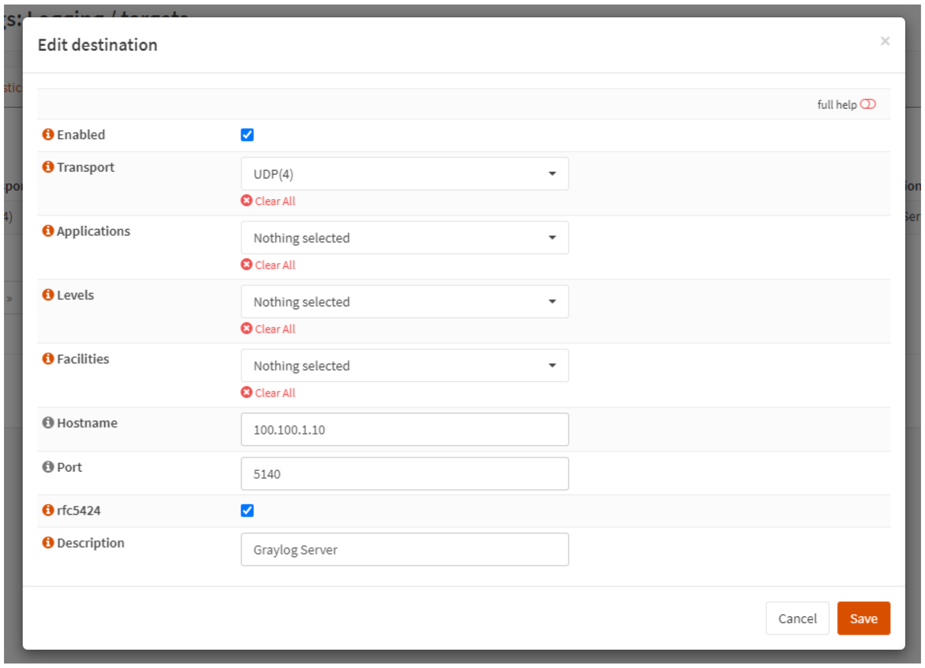
2.1 Host configurations

The first thing we did was to define a new destination where the hosts have to send their logs, and we did this by adding a configuration file in rsyslog for Graylog situated in "/etc/rsyslog.d/graylog.conf", we set the address to 100.100.1.10 (Graylog's IP) and port 5140.

```
*.* @100.100.1.10:5140:RSYSLOG_SyslogProtocol23Format
```

After adding this file to every ACME host (except for those in the Client network because of firewall rules), we restarted the service in them using the command "service rsyslog restart" to reload their rsyslog configurations.

To send logs from the two firewalls (Main and Internal), we modified a setting in OPNsense (System → Logging/Targets → Destinations). We chose as transport UDP, we set applications, levels, and facilities to "Nothing selected" (that means all selected) and as hostname the IP 100.100.1.10 and port the usual 5140.

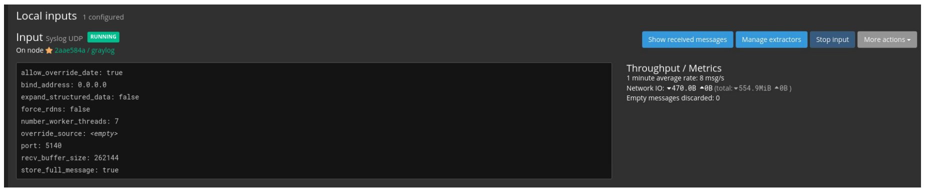


Enabled	✓
Transport	UDP(4)
Applications	Nothing selected
Levels	Nothing selected
Facilities	Nothing selected
Hostname	100.100.1.10
Port	5140
rfc5424	✓
Description	Graylog Server

Figure 1: Implementation to send logs from the firewalls

2.2 Graylog's input

Futher we have logged to Graylog (thanks to the credential we have received by the Prof) using the web interface with address 100.100.1.10:9000 from Client ext 1 (the only one that has a graphical interface and can access Graylog, and has as username kali visible from the logs received by Graylog). The first thing we have done was configuring how Graylog had to listen and receive logs from other hosts. So we navigated to System → Inputs and we configured an input that listens and receives all the logs arriving at port 5140. We decided to use only one port and one listener since the number of hosts in the network is small, even though we know that Graylog allows more inputs to be implemented.

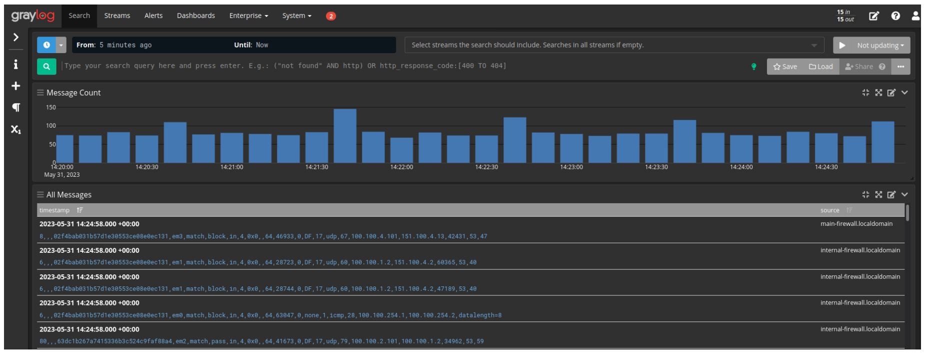


```
allow_override_date: true
bind_address: 0.0.0.0
expand_structured_data: false
force_rdns: false
number_worker_threads: 7
override_source: <empty>
port: 5140
recv_buffer_size: 262144
store_full_message: true
```

```
Throughput / Metrics
1 minute average rate: 8 msg/s
Network IO: ▾470.0B ▴0B (total: ▾554.9MiB ▴0B )
Empty messages discarded: 0
```

Figure 2: Rules that we have chose to implement Graylog's input

After this initial configuration, it was possible to observe all the logs captured by Graylog in the Search section. From here, we started to think about what types of alerts and dashboards we want to implement.

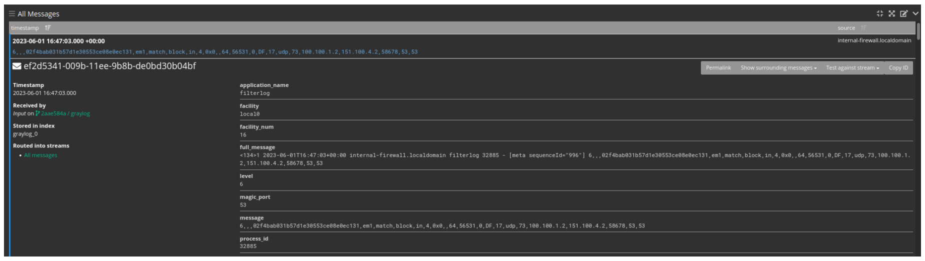


```
2023-05-31 14:24:58.000 +00:00                                                        main-firewall.localdomain
8,,,02f4bab031b57d1e30553ce08e0ec131,em3,match,block,in,4,0x0,,64,46933,0,DF,17,udp,67,100.100.4.101,151.100.4.13,42431,53,47
2023-05-31 14:24:58.000 +00:00                                                        internal-firewall.localdomain
6,,,02f4bab031b57d1e30553ce08e0ec131,em1,match,block,in,4,0x0,,64,28723,0,DF,17,udp,60,100.100.1.2,151.100.4.2,60365,53,40
2023-05-31 14:24:58.000 +00:00                                                        internal-firewall.localdomain
6,,,02f4bab031b57d1e30553ce08e0ec131,em1,match,block,in,4,0x0,,64,28744,0,DF,17,udp,60,100.100.1.2,151.100.4.2,47189,53,40
2023-05-31 14:24:58.000 +00:00                                                        internal-firewall.localdomain
6,,,02f4bab031b57d1e30553ce08e0ec131,em0,match,block,in,4,0x0,,64,63047,0,none,1,icmp,28,100.100.254.1,100.100.254.2,datalength=8
2023-05-31 14:24:58.000 +00:00                                                        internal-firewall.localdomain
80,,,63dc1b267a7415336b3c524c9faf88a4,em2,match,pass,in,4,0x0,,64,41673,0,DF,17,udp,79,100.100.2.101,100.100.1.2,34962,53,59
```

Figure 3: Visualization of Graylog's input

2.3 Extractors

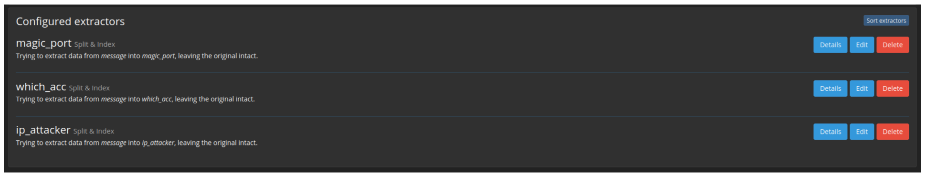
To extract the necessary information to be used for alerts and dashboards, we created three extractors. These extractors allow us to analyse certain packets received as input and create new fields for these packets from other attributes of the same packet.



```
2023-06-01 16:47:03.000 +00:00                                                        internal-firewall.localdomain
6,,,02f4bab031b57d1e30553ce08e0ec131,em1,match,block,in,4,0x0,,64,56531,0,DF,17,udp,73,100.100.1.2,151.100.4.2,58678,53,53
ef2d5341-009b-11ee-9b8b-de0bd30b04bf

Timestamp
2023-06-01 16:47:03.000
Received by
Input on 2aae584a / graylog
Stored in index
graylog_0
Routed into streams
  • All messages

application_name
filterlog
facility
local0
facility_num
16
full_message
<134>1 2023-06-01T16:47:03+00:00 internal-firewall.localdomain filterlog 32885 - [meta sequenceId="996"] 6,,,02f4bab031b57d1e30553ce08e0ec131,em1,match,block,in,4,0x0,,64,56531,0,DF,17,udp,73,100.100.1.2,151.100.4.2,58678,53,53
level
6
magic_port
53
message
6,,,02f4bab031b57d1e30553ce08e0ec131,em1,match,block,in,4,0x0,,64,56531,0,DF,17,udp,73,100.100.1.2,151.100.4.2,58678,53,53
process_id
32885
```

Figure 4: Example of the extractor magic_port generated by the message of a packet that is visible as an attribute

It is possible to view them in the section System → Inputs → Local inputs → Manage extractors.



Configured extractors

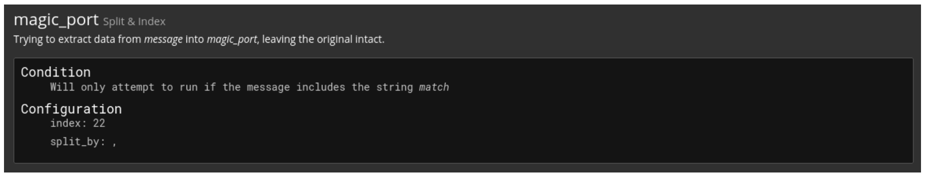
magic_port Split & Index
Trying to extract data from *message* into *magic_port*, leaving the original intact.

which_acc Split & Index
Trying to extract data from *message* into *which_acc*, leaving the original intact.

ip_attacker Split & Index
Trying to extract data from *message* into *ip_attacker*, leaving the original intact.

Figure 5: Extractors created

As we can see from the Image 5 we created three of them and we used for all three the extraction by index because the information in the message are always separated by a value, and we noticed that it is the easiest way to get the data. The first one (magic_port) is used to extract the port from the logs generated by the two firewalls (Main and Internal), we take the logs that allow or block the passage of packets, so that have a "match" word in them (picture 6). After there is which_account that is used to extract information about who connected to OpenVPN on the Main firewall (in our case Alice, Bob or Charles) and the last one ip_attacker used to extract the IP when a user mistypes the password for the SSH connection to the Web Server.



magic_port Split & Index

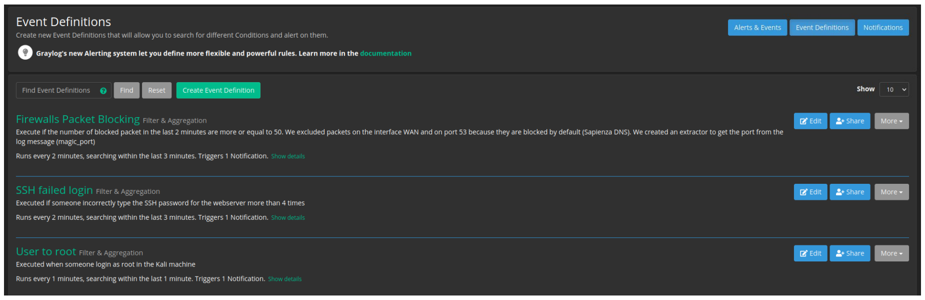
Trying to extract data from *message* into *magic_port*, leaving the original intact.

```
Condition
    Will only attempt to run if the message includes the string match
Configuration
    index: 22
    split_by: ,
```

Figure 6: Implementation of magic_port extractor

2.4 Alerts

The alerts make use of the extractors created to keep track of the accumulated data and notify whenever the number set for each alert is reached. They can be found in the Alert section and their implementation in Alert → Event definitions. As notification, we decided to create an email one (Alert → Notifications) that is used for all three of them.



Event Definitions

Create new Event Definitions that will allow you to search for different Conditions and alert on them.

Graylog's new Alerting system let you define more flexible and powerful rules. Learn more in the documentation

Firewalls Packet Blocking Filter & Aggregation

Execute if the number of blocked packet in the last 2 minutes are more or equal to 50. We excluded packets on the interface WAN and on port 53 because they are blocked by default (Sapienza DNS). We created an extractor to get the port from the log message (magic_port)

Runs every 2 minutes, searching within the last 3 minutes. Triggers 1 Notification.

SSH failed login Filter & Aggregation

Executed if someone incorrectly type the SSH password for the webserver more than 4 times

Runs every 2 minutes, searching within the last 3 minutes. Triggers 1 Notification.

User to root Filter & Aggregation

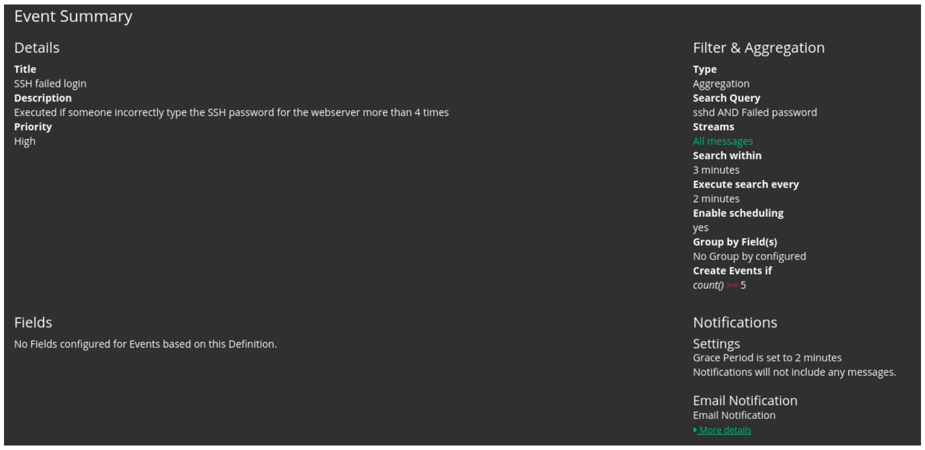
Executed when someone login as root in the Kali machine

Runs every 1 minutes, searching within the last 1 minute. Triggers 1 Notification.

Figure 7: The alerts we have implemented

2.4.1 SSH failed login

We created this alert with the purpose of giving us a notification whenever someone gets the password wrong for the SSH authentication to the Web Server more than 4 times in a short period of time. This was set as High priority with condition type as Filter & Aggregation, and it checks the last 3 minutes of intercepted logs with query "sshd AND Failed password" every two minutes.



Event Summary

Details

Title
SSH failed login
Description
Executed if someone incorrectly type the SSH password for the webserver more than 4 times
Priority
High

Filter & Aggregation

Type
Aggregation
Search Query
sshd AND Failed password
Streams
All messages
Search within
3 minutes
Execute search every
2 minutes
Enable scheduling
yes
Group by Field(s)
No Group by configured
Create Events if
count() >= 5

Fields

No Fields configured for Events based on this Definition.

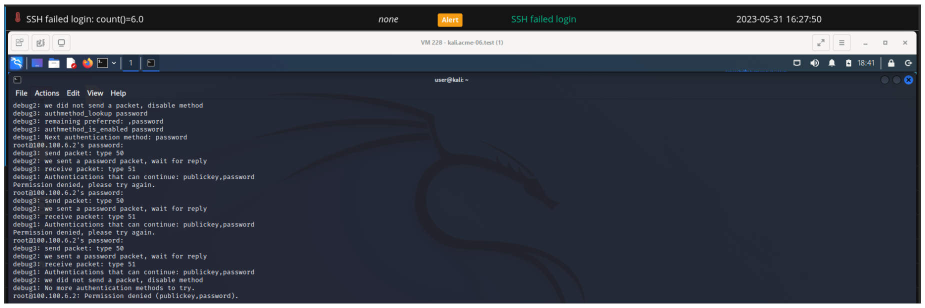
Notifications

Settings
Grace Period is set to 2 minutes
Notifications will not include any messages.

Email Notification
Email Notification
More details

Figure 8: SSH failed login event

To test if this event is captured correctly, we tried an SSH connection several times from the kali machine (only its subnet can use SSH) to the Web Server and as seen in the figure 9 it correctly generates the alert (the number of attemps was 6).



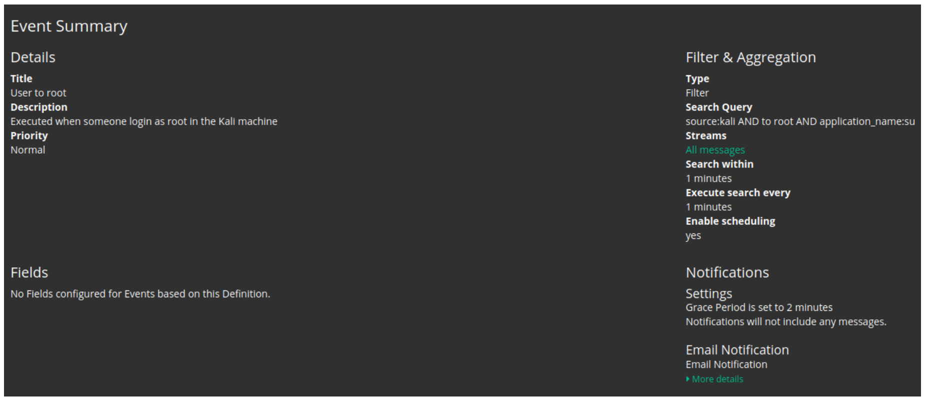
SSH failed login: count()=6.0 none Alert SSH failed login 2023-05-31 16:27:50

```
debug2: we did not send a packet, disable method
debug3: authmethod_lookup password
debug3: remaining preferred: ,password
debug3: authmethod_is_enabled password
debug1: Next authentication method: password
root@100.100.6.2's password:
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
Permission denied, please try again.
root@100.100.6.2's password:
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
Permission denied, please try again.
root@100.100.6.2's password:
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
debug2: we did not send a packet, disable method
debug1: No more authentication methods to try.
root@100.100.6.2: Permission denied (publickey,password).
```

Figure 9: SSH failed login alert instance

2.4.2 User to root

This alert is generated by Graylog every time a user uses the command "sudo su" or "sudo su -" to gain root access in the kali machine (Client ext 1). Every minute, it checks the last minute of said logs. The event is created for definition if the filter with the query "source:kali AND to root AND application_name:su" has results and the priority was set to Normal.


Event Summary

Details

Title
User to root
Description
Executed when someone login as root in the Kali machine
Priority
Normal

Filter & Aggregation

Type
Filter
Search Query
source:kali AND to root AND application_name:su
Streams
All messages
Search within
1 minutes
Execute search every
1 minutes
Enable scheduling
yes

Fields

No Fields configured for Events based on this Definition.

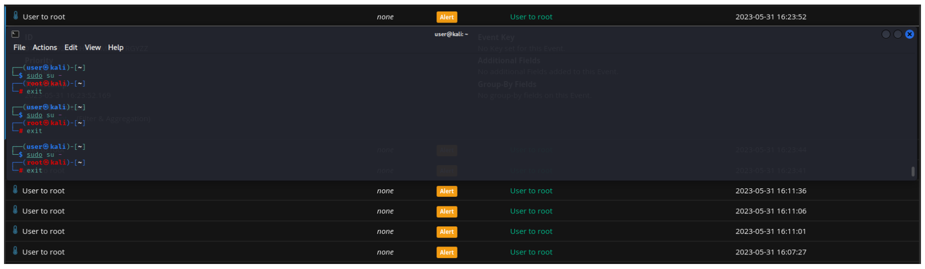
Notifications

Settings
Grace Period is set to 2 minutes
Notifications will not include any messages.

Email Notification
Email Notification

Figure 10: User to root event

To test the correct capture of this event, we tried to gain root access through the command "sudo su -" in the kali machine as it is shown in image 11 and we can observe that it creates the alert.


```
User to root    none    Alert    User to root    2023-05-31 16:23:52
```

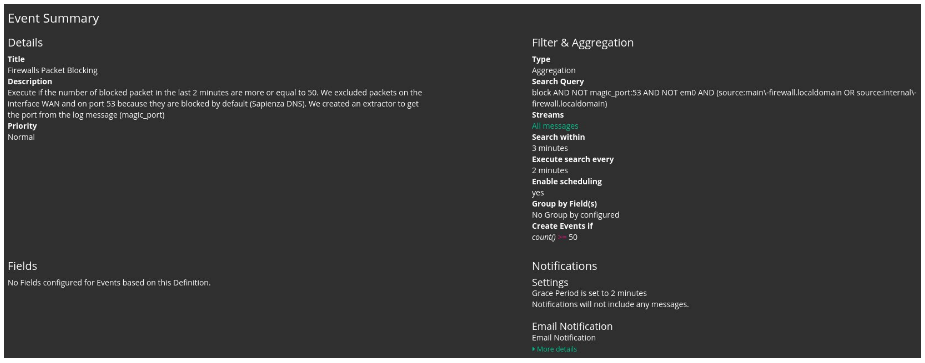
```
┌──(user㉿kali)-[~]
└─$ sudo su -
┌──(root㉿kali)-[~]
└─# exit

┌──(user㉿kali)-[~]
└─$ sudo su -
┌──(root㉿kali)-[~]
└─# exit

┌──(user㉿kali)-[~]
└─$ sudo su -
┌──(root㉿kali)-[~]
└─# exit
```

```
User to root    none    Alert    User to root    2023-05-31 16:11:36
User to root    none    Alert    User to root    2023-05-31 16:11:06
User to root    none    Alert    User to root    2023-05-31 16:11:01
User to root    none    Alert    User to root    2023-05-31 16:07:27
```

Figure 11: User to root alert instance

2.4.3 Firewalls Packet Blocking

With this last alert, we check if the number of blocked packets by the two firewalls in the last 2 minutes have reached or surpassed 50 while excluding the packets on port 53 (Sapienza DNS) and on the WAN interface. This alert uses the extractor made to check from which port the packet is coming from (magic_port). The priority for this alert is set to Normal and just like the first alert it checks the last 3 minutes of logs every 2 minutes.


Event Summary

Details

Title
Firewalls Packet Blocking
Description
Execute if the number of blocked packet in the last 2 minutes are more or equal to 50. We excluded packets on the interface WAN and on port 53 because they are blocked by default (Sapienza DNS). We created an extractor to get the port from the log message (magic_port)
Priority
Normal

Fields

No Fields configured for Events based on this Definition.

Filter & Aggregation

Type
Aggregation
Search Query
block AND NOT magic_port:53 AND NOT em0 AND (source:main\-firewall.localdomain OR source:internal\-firewall.localdomain)
Streams
All messages
Search within
3 minutes
Execute search every
2 minutes
Enable scheduling
yes
Group by Field(s)
No Group by configured
Create Events if
count() >= 50

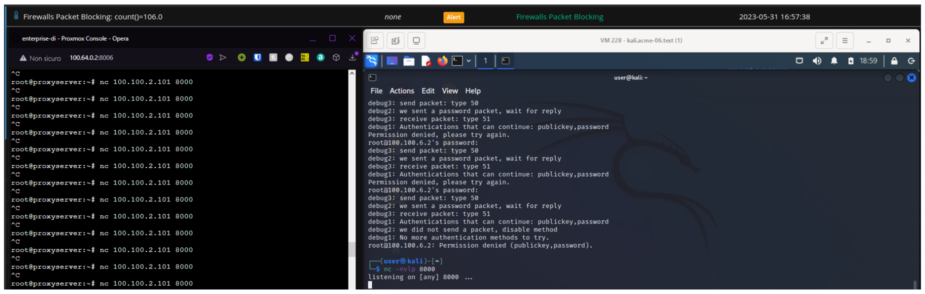
Notifications

Settings
Grace Period is set to 2 minutes
Notifications will not include any messages.

Email Notification
Email Notification

Figure 12: Firewalls Packet Blocking event

To test this last alert, we made multiple netcat calls from the Proxy Server to the Kali machine which are always going to be blocked and as shown in the picture 13 there is the generation of the alert, the exact number of netcat connections were 106.


Firewalls Packet Blocking: count()=106.0 *none* Alert Firewalls Packet Blocking 2023-05-31 16:57:38

enterprise-di - Proxmox Console - Opera
Non sicuro
100.64.0.2:8006
VM 228 - kali.acme-06.test (1)
18:59
user@kali: ~
File Actions Edit View Help

```
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
^C
root@proxyserver:~# nc 100.100.2.101 8000
```

```
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
Permission denied, please try again.
root@100.100.6.2's password:
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
Permission denied, please try again.
root@100.100.6.2's password:
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
debug2: we did not send a packet, disable method
debug1: No more authentication methods to try.
root@100.100.6.2: Permission denied (publickey,password).

┌──(user㉿kali)-[~]
└─$ nc -nvlp 8000
listening on [any] 8000 ...
```

Figure 13: Firewalls Packet Blocking alert instance

2.5 Dashboards

Dashboards are used to better visualise certain network parameters. They can be created in the Dashboard menu, we designed a new section called Main Dashboards.

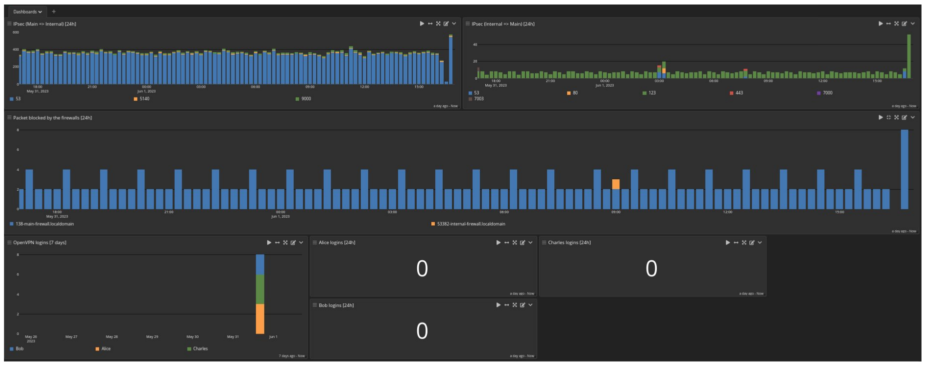


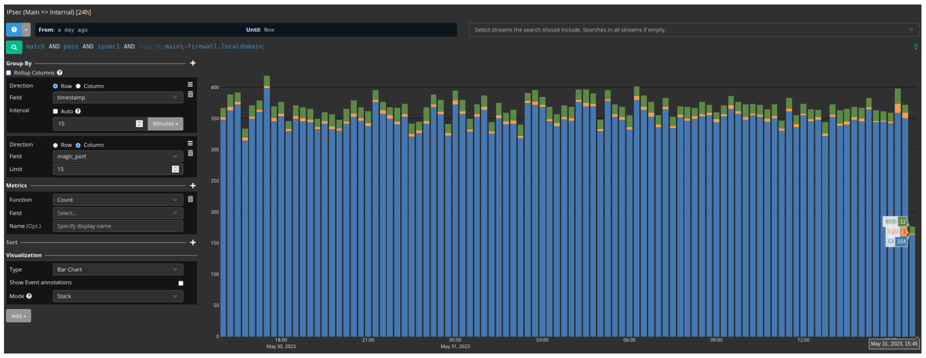
Figure 14: Small preview of our implementations

2.5.1 IPsec (Main ⇒ Internal) [24h]

With the first dashboard, we analysed all the packets which come from the Main Firewall and passed to the Internal Firewall using IPsec. The query used to filter the interested logs is:

```
match AND pass AND ipsec1 AND source:\main\-firewall.localdomain
```

To create the graph, we used two Group By: one for the Row that has as Field the timestamp (15 minutes candles) and one for Column to subdivide the candles into the different ports (using the extractor magic_port). As Metrics, we have the Function Count and as Visualization Bar Chart in Stack mode.



IPsec (Main => Internal) [24h]

From: a day ago Until: Now

```
match AND pass AND ipsec1 AND source:main\-firewall.localdomain
```

Group By

Rollup Columns

Direction: [✓ Row] Column
Field: timestamp
Interval: Auto 15 Minutes

Direction: Row [✓ Column]
Field: magic_port
Limit: 15

Metrics

Function: Count
Field: Select...
Name (Opt.): Specify display name

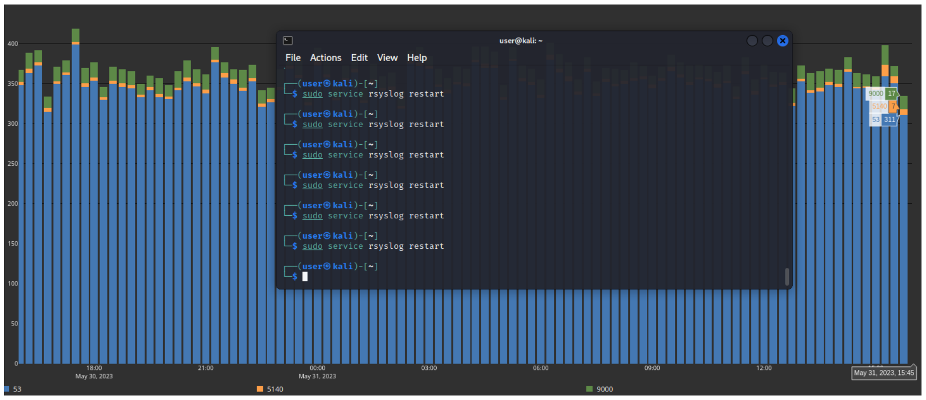
Sort

Visualization

Type: Bar Chart
Show Event annotations
Mode: Stack

Select streams the search should include. Searches in all streams if empty.
Add
May 30, 2023
May 31, 2023
May 31, 2023, 15:45

Figure 15: Dashboard 1 before test



```
┌──(user㉿kali)-[~]
└─$ sudo service rsyslog restart

┌──(user㉿kali)-[~]
└─$ sudo service rsyslog restart

┌──(user㉿kali)-[~]
└─$ sudo service rsyslog restart

┌──(user㉿kali)-[~]
└─$ sudo service rsyslog restart

┌──(user㉿kali)-[~]
└─$ sudo service rsyslog restart

┌──(user㉿kali)-[~]
└─$ sudo service rsyslog restart

┌──(user㉿kali)-[~]
└─$
```

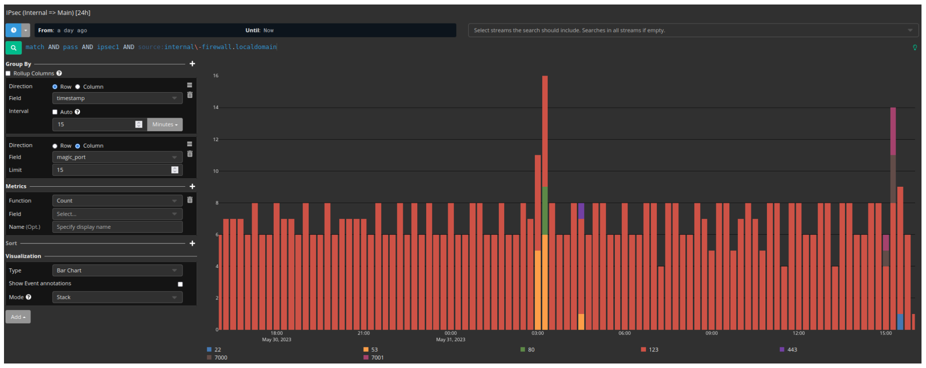
Figure 16: Dashboard 1 after test

To test the proper functionality of this first dashboard, we simply sent IPsec packets using the "sudo service rsyslog restart" command from Client ext 1, as it is shown in the image 16, to increase the logs captured on port 5140. As we can see the graph updated itself from 1 (image 15) to 7 logs (image 16) because every time we execute this command, Client ext 1 generates new logs and sends them to Graylog.

2.5.2 IPsec (Internal ⇒ Main) [24h]

The second aggregation was similarly made for the packets received by the Main Firewall, which came from the Internal Firewall. The query used is the same as the one used in the first dashboard, with only the source changed:

```
match AND pass AND ipsec1 AND source:\internal\-firewall.localdomain
```



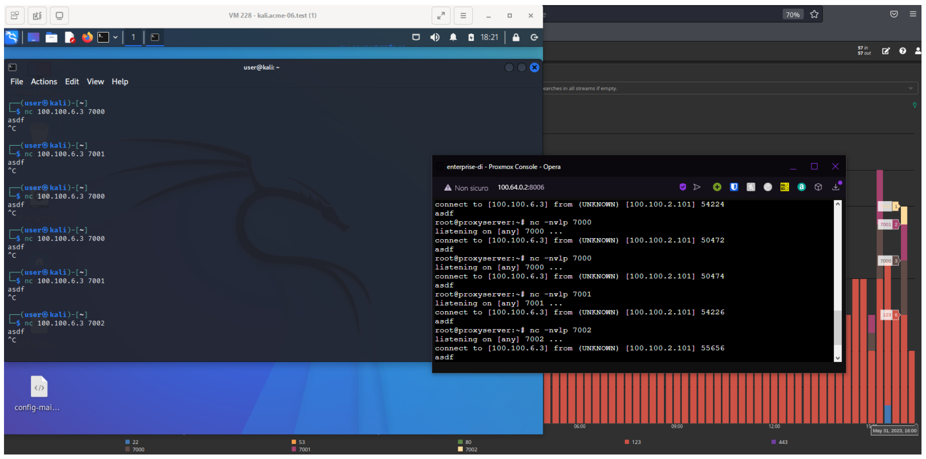
IPsec (Internal => Main) [24h]

From: a day ago Until: Now

```
match AND pass AND ipsec1 AND source:internal\-firewall.localdomain
```

Group By
Rollup Columns
Direction: [✓ Row] Column
Field: timestamp
Interval: Auto
15 Minutes
Direction: Row [✓ Column]
Field: magic_port
Limit: 15
Metrics
Function: Count
Field: Select...
Name (Opt.): Specify display name
Sort
Visualization
Type: Bar Chart
Show Event annotations
Mode: Stack

Figure 17: Dashboard 2 before test


```
┌──(user㉿kali)-[~]
└─$ nc 100.100.6.3 7000
asdf
^C

┌──(user㉿kali)-[~]
└─$ nc 100.100.6.3 7001
asdf
^C

┌──(user㉿kali)-[~]
└─$ nc 100.100.6.3 7000
asdf
^C

┌──(user㉿kali)-[~]
└─$ nc 100.100.6.3 7000
asdf
^C

┌──(user㉿kali)-[~]
└─$ nc 100.100.6.3 7001
asdf
^C

┌──(user㉿kali)-[~]
└─$ nc 100.100.6.3 7002
asdf
^C
```

```
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 54224
asdf
root@proxyserver:~# nc -nvlp 7000
listening on [any] 7000 ...
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 50472
asdf
root@proxyserver:~# nc -nvlp 7000
listening on [any] 7000 ...
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 50474
asdf
root@proxyserver:~# nc -nvlp 7001
listening on [any] 7001 ...
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 54226
asdf
root@proxyserver:~# nc -nvlp 7002
listening on [any] 7002 ...
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 55656
asdf
```
VM 228 - kali.acme-06.test (1)
File Actions Edit View Help
user@kali: ~
config-mai...
enterprise-di - Proxmox Console - Opera
Non sicuro 100.64.0.2:8006
06:00
09:00
12:00
22
7000
53
7001
80
7002
123
443

Figure 18: Dashboard 2 after test

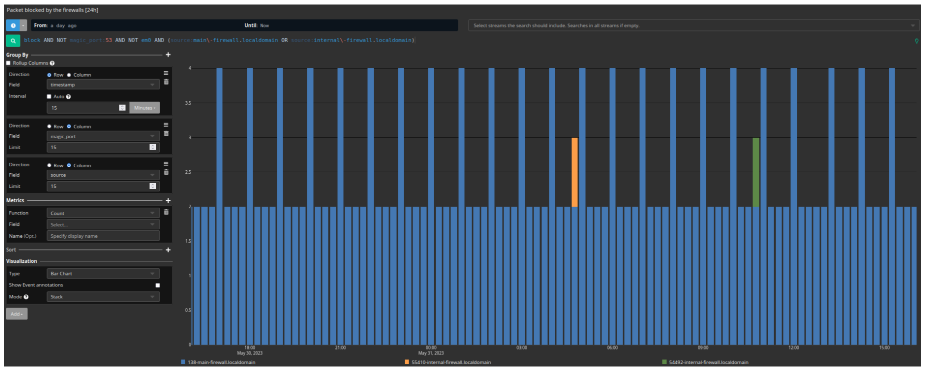
In the figure 18 we can see the test executed to check if the graph correctly works. We opened different netcat connections from kali (in Clients network) to the Proxy server that was in listening, and we can see that the packets are captured and analysed by Graylog.

2.5.3 Packet blocked by the firewalls [24h]

The third dashboard shows which packets are blocked by the firewalls. The query used is:

```
block AND NOT magic_port:53 AND NOT em0 AND
(source:\main\-firewall.localdomain OR source:\internal\-firewall.localdomain)
```

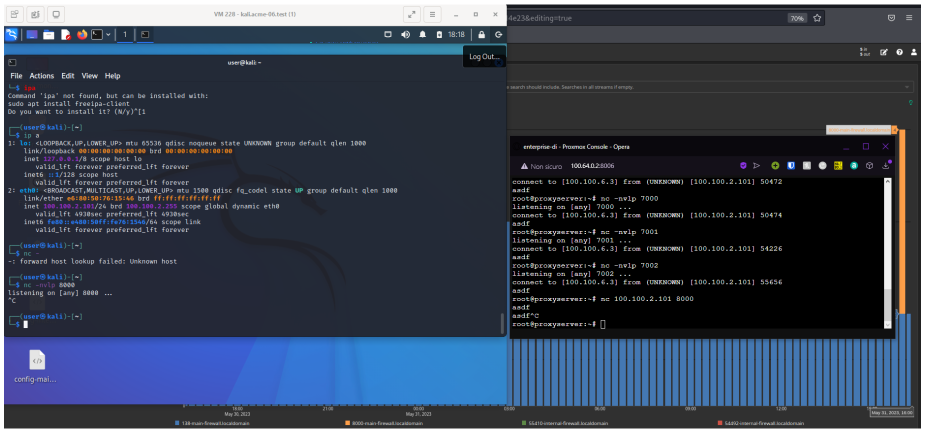
We exclude the logs with port destination 53 as they are continuously sent and blocked by the firewalls, making the aggregation of the other blocked logs obsolete. The same for the "em0" interface, which are not the ones we are looking for.


Packet blocked by the firewalls [24h]

```
block AND NOT magic_port:53 AND NOT em0 AND (source:main\-firewall.localdomain OR source:internal\-firewall.localdomain)
```

Figure 19: Dashboard 3 before test

We used three Group By: the Row has the timestamp (15 minutes for candle), the first Column the port (magic_port) and the other Column the source (Main or Internal firewall). As Metrics, we used Count and as visualization a Bar Chart in Stack mode.


```
└─$ ipa
Command 'ipa' not found, but can be installed with:
sudo apt install freeipa-client
Do you want to install it? (N/y)^[1

┌──(user㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether e6:80:50:76:15:46 brd ff:ff:ff:ff:ff:ff
    inet 100.100.2.101/24 brd 100.100.2.255 scope global dynamic eth0
       valid_lft 4930sec preferred_lft 4930sec
    inet6 fe80::e480:50ff:fe76:1546/64 scope link
       valid_lft forever preferred_lft forever

┌──(user㉿kali)-[~]
└─$ nc -
-: forward host lookup failed: Unknown host

┌──(user㉿kali)-[~]
└─$ nc -nvlp 8000
listening on [any] 8000 ...
^C

┌──(user㉿kali)-[~]
└─$ 
```

```
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 50472
asdf
root@proxyserver:~# nc -nvlp 7000
listening on [any] 7000 ...
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 50474
asdf
root@proxyserver:~# nc -nvlp 7001
listening on [any] 7001 ...
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 54226
asdf
root@proxyserver:~# nc -nvlp 7002
listening on [any] 7002 ...
connect to [100.100.6.3] from (UNKNOWN) [100.100.2.101] 55656
asdf
root@proxyserver:~# nc 100.100.2.101 8000
asdf
asdf^C
root@proxyserver:~# 
```

VM 228 - kali.acme-06.test (1)
user@kali: ~
File Actions Edit View Help
Log Out...
config-mai...
enterprise-di - Proxmox Console - Opera
Non sicuro 100.64.0.2:8006
8000-main-firewall.localdomain 4
May 31, 2023, 16:00
138-main-firewall.localdomain
8000-main-firewall.localdomain
55410-internal-firewall.localdomain
54492-internal-firewall.localdomain

Figure 20: Dashboard 3 after test

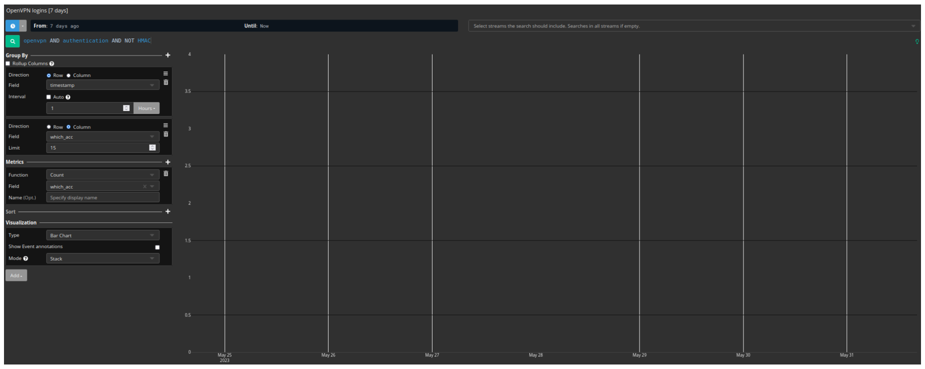
To test this dashboard we tried to connect the Proxy server to kali through netcat, as we know this is not possible by the firewall rules, the dashboard rightly shows us the blocked packet.

2.5.4 OpenVPN logins [7 days]

The fourth one tracks how many times there is a login through OpenVPN. The query used to search the interested logs is:

```
openvpn AND authentication AND NOT HMAC
```

We excluded HMAC logs as the dashboard was picking up some packets which had nothing to do with the authentications.



OpenVPN logins [7 days]

From: 7 days ago   Until: Now

```
openvpn AND authentication AND NOT HMAC
```

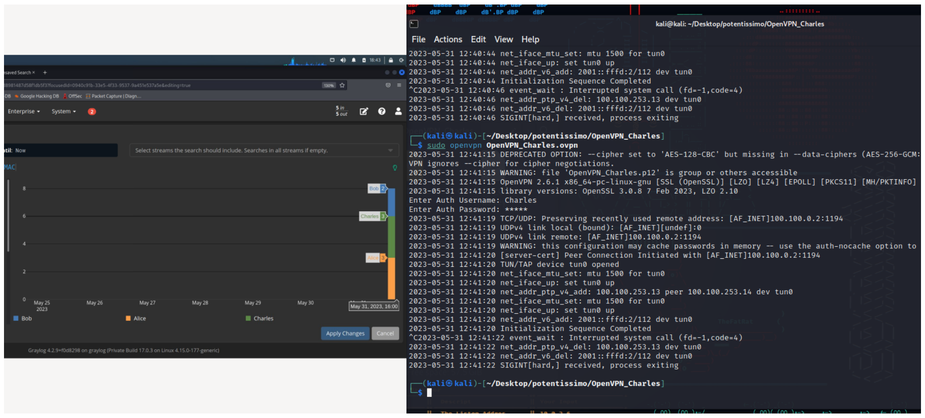
Group By
- Rollup Columns: unchecked
- Direction: [✓ Row] Column — Field: timestamp — Interval: Auto unchecked, 1 Hours
- Direction: Row [✓ Column] — Field: which_acc — Limit: 15

Metrics
- Function: Count — Field: which_acc — Name (Opt.): Specify display name

Visualization
- Type: Bar Chart — Show Event annotations: unchecked — Mode: Stack

Figure 21: Dashboard 4 before test

In the graph we have As Group By: Row with timestamp (1 hour candles) and Column with which_account (extractor to get username of the connection: Alice, Bob or Charles). The Metrics is set to Count and Visualization as always Bar Chart in Stack mode.



```
2023-05-31 12:40:44 net_iface_mtu_set: mtu 1500 for tun0
2023-05-31 12:40:44 net_iface_up: set tun0 up
2023-05-31 12:40:44 net_addr_v6_add: 2001::fffd:2/112 dev tun0
2023-05-31 12:40:44 Initialization Sequence Completed
^C2023-05-31 12:40:46 event_wait : Interrupted system call (fd=-1,code=4)
2023-05-31 12:40:46 net_addr_ptp_v4_del: 100.100.253.13 dev tun0
2023-05-31 12:40:46 net_addr_v6_del: 2001::fffd:2/112 dev tun0
2023-05-31 12:40:46 SIGINT[hard,] received, process exiting

┌──(kali㉿kali)-[~/Desktop/potentissimo/OpenVPN_Charles]
└─$ sudo openvpn OpenVPN_Charles.ovpn
2023-05-31 12:41:15 DEPRECATED OPTION: --cipher set to 'AES-128-CBC' but missing in --data-ciphers (AES-256-GCM:
VPN ignores --cipher for cipher negotiations.
2023-05-31 12:41:15 WARNING: file 'OpenVPN_Charles.p12' is group or others accessible
2023-05-31 12:41:15 OpenVPN 2.6.1 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO]
2023-05-31 12:41:15 library versions: OpenSSL 3.0.8 7 Feb 2023, LZO 2.10
Enter Auth Username: Charles
Enter Auth Password: *****
2023-05-31 12:41:19 TCP/UDP: Preserving recently used remote address: [AF_INET]100.100.0.2:1194
2023-05-31 12:41:19 UDPv4 link local (bound): [AF_INET][undef]:0
2023-05-31 12:41:19 UDPv4 link remote: [AF_INET]100.100.0.2:1194
2023-05-31 12:41:19 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option to
2023-05-31 12:41:20 [server-cert] Peer Connection Initiated with [AF_INET]100.100.0.2:1194
2023-05-31 12:41:20 TUN/TAP device tun0 opened
2023-05-31 12:41:20 net_iface_mtu_set: mtu 1500 for tun0
2023-05-31 12:41:20 net_iface_up: set tun0 up
2023-05-31 12:41:20 net_addr_ptp_v4_add: 100.100.253.13 peer 100.100.253.14 dev tun0
2023-05-31 12:41:20 net_iface_mtu_set: mtu 1500 for tun0
2023-05-31 12:41:20 net_iface_up: set tun0 up
2023-05-31 12:41:20 net_addr_v6_add: 2001::fffd:2/112 dev tun0
2023-05-31 12:41:20 Initialization Sequence Completed
^C2023-05-31 12:41:22 event_wait : Interrupted system call (fd=-1,code=4)
2023-05-31 12:41:22 net_addr_ptp_v4_del: 100.100.253.13 dev tun0
2023-05-31 12:41:22 net_addr_v6_del: 2001::fffd:2/112 dev tun0
2023-05-31 12:41:22 SIGINT[hard,] received, process exiting

┌──(kali㉿kali)-[~/Desktop/potentissimo/OpenVPN_Charles]
└─$ 
```

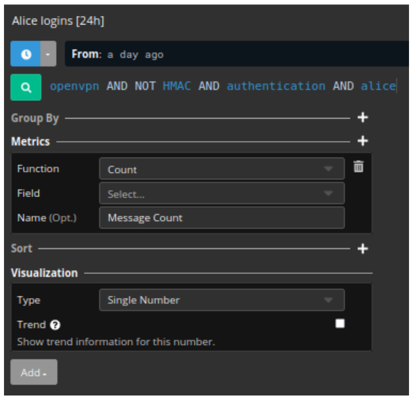
Figure 22: Dashboard 4 after test

To test it, we simply connected to the VPNs, and we noticed that Graylog updates the information correctly.

2.5.5 Alice logins [24h], Bob logins [24h], Charles logins [24h]

The next three dashboards are simple counters that display the number of OpenVPN authentications for each account: Alice, Bob and Charles. The queries are:

```
openvpn AND NOT HMAC AND authentication AND alice

openvpn AND NOT HMAC AND authentication AND bob

openvpn AND NOT HMAC AND authentication AND charles
```



Alice logins [24h]

From: a day ago

```
openvpn AND NOT HMAC AND authentication AND alice
```

Group By

Metrics

Function: Count
Field: Select...
Name (Opt.): Message Count

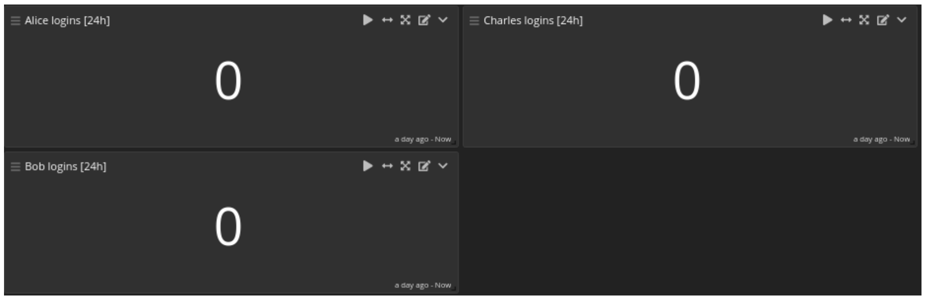
Sort

Visualization

Type: Single Number
Trend
Show trend information for this number.

Figure 23: Dashboard 5 characteristics

This is a simple one, we have only a Metric that is Count and Visualization is set to a Single Number (the same for Bob and Charles).



Alice logins [24h]
0
a day ago - Now

Charles logins [24h]
0
a day ago - Now

Bob logins [24h]
0
a day ago - Now

Figure 24: Dashboard 5 before test

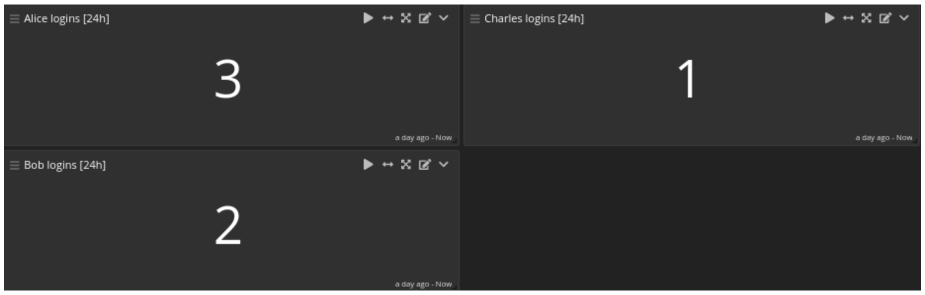

Alice logins [24h]
3
a day ago - Now
Charles logins [24h]
1
a day ago - Now
Bob logins [24h]
2
a day ago - Now

Figure 25: Dashboard 5 after test

To execute the tests we connected to the VPNs and as we can see from the image 25 the numbers correctly update.

2.5.6 SSH failed login to Web Server [24h]

The next dashboard lets us analyse all the SSH connections to the Web Server, in particular every time a user inputs the wrong password. As we know, only the hosts of the subnet Client network can connect to it (firewall rule). To do this we installed SSH in the Web Server (we do not remember if it was installed before) and we modified the SSH configuration to accept authentication to the root account, and we also made it possible to use authentication using a password which was normally deactivated, and the only possibility was using public and private keys. The query used is:

```
sshd AND Failed password
```

This dashboard shows us every single failed SSH authentication and from which IP that attempt was made from (thanks to the ip_attacker extractor).

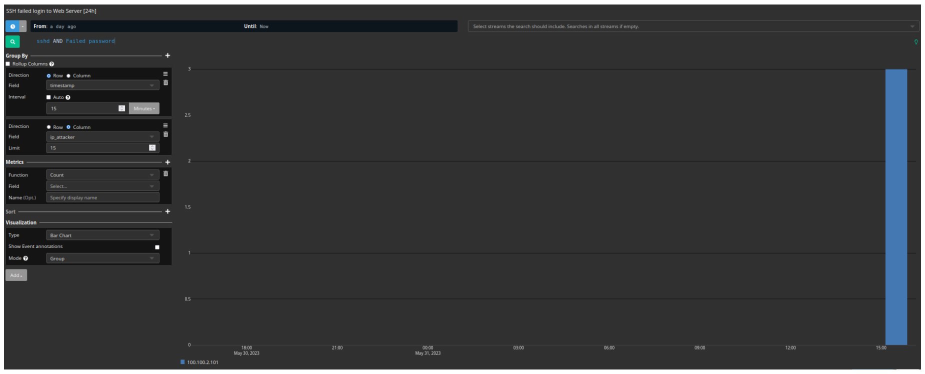
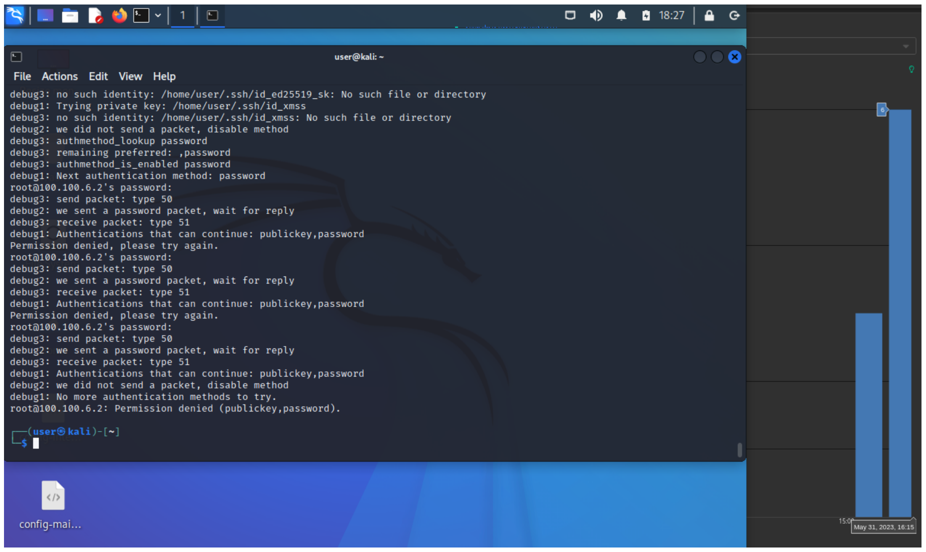

SSH failed login to Web Server [24h]
From: a day ago Until: Now
sshd AND Failed password
Group By
Direction: Row
Field: timestamp
Interval: 15 Minutes
Direction: Column
Field: ip_attacker
Limit: 15
Metrics
Function: Count
Field: Select...
Name (Opt.): Specify display name
Visualization
Type: Bar Chart
Mode: Group

Figure 26: Dashboard 6 before test

As we can see from the image 26 there are two Group By: a timestamp Row

(15 minutes candles) and a Column with ip_attacker as field (extractor). There is a single Metric Count and the Visualization is a Bar Chart in Group Mode.



```
debug3: no such identity: /home/user/.ssh/id_ed25519_sk: No such file or directory
debug1: Trying private key: /home/user/.ssh/id_xmss
debug3: no such identity: /home/user/.ssh/id_xmss: No such file or directory
debug2: we did not send a packet, disable method
debug3: authmethod_lookup password
debug3: remaining preferred: ,password
debug3: authmethod_is_enabled password
debug1: Next authentication method: password
root@100.100.6.2's password:
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
Permission denied, please try again.
root@100.100.6.2's password:
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
Permission denied, please try again.
root@100.100.6.2's password:
debug3: send packet: type 50
debug2: we sent a password packet, wait for reply
debug3: receive packet: type 51
debug1: Authentications that can continue: publickey,password
debug2: we did not send a packet, disable method
debug1: No more authentication methods to try.
root@100.100.6.2: Permission denied (publickey,password).

┌──(user㉿kali)-[~]
└─$ 
```

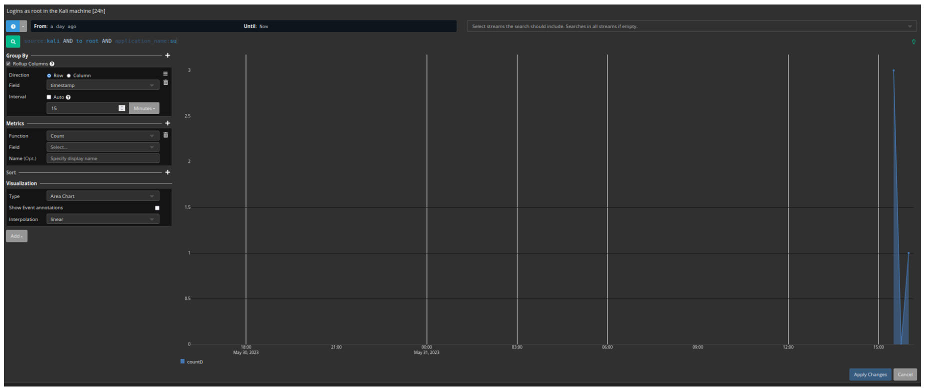
Figure 27: Dashboard 6 after test

To test it, we connected from kali (in Clients network) to the Web Server via SSH, and we typed a wrong password. From the picture 27 we see that Graylog correctly received the logs.

2.5.7 Logins as root in the Kali machine [24h]

The last aggregation we made is used to keep track of who logs in as root in the kali (client-ext1) machine, in other terms, every time someone uses the commands "sudo su" or "sudo su -". The query used is:

```
source:kali AND to root AND application_name:su
```



Logins as root in the Kali machine [24h]

From: a day ago Until: Now

```
source:kali AND to root AND application_name:su
```

Group By
Rollup Columns
Direction: Row
Field: timestamp
Interval: 15 Minutes

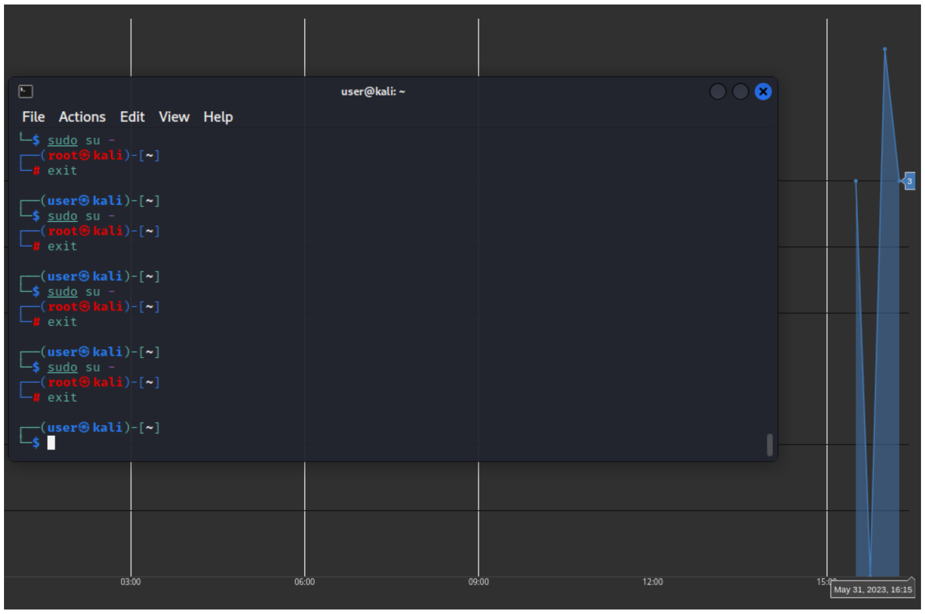
Metrics
Function: Count
Field: Select...
Name (Opt.): Specify display name

Sort

Visualization
Type: Area Chart
Show Event annotations
Interpolation: linear

Figure 28: Dashboard 7 before test

To generate the graph we set a Group By with the timestamp (15 minutes candles), a Metric that is the Count and the Visualization is set to a linear Area Chart.



```
└─$ sudo su -
┌──(root㉿kali)-[~]
└─# exit

┌──(user㉿kali)-[~]
└─$ sudo su -
┌──(root㉿kali)-[~]
└─# exit

┌──(user㉿kali)-[~]
└─$ sudo su -
┌──(root㉿kali)-[~]
└─# exit

┌──(user㉿kali)-[~]
└─$ sudo su -
┌──(root㉿kali)-[~]
└─# exit

┌──(user㉿kali)-[~]
└─$ 
```

Figure 29: Dashboard 7 after test

To test it we executed the command “sudo su -” in kali (Client ext 1) and as we can see from the picture 29 the graph correctly updates itself.

3 Final remarks

We effectively learned how to use Graylog fairly easily to manage packets and logs. We found the interface quite intuitive and accessible, thanks also to the presence of numerous tutorials on the Web. The pros of using this system are an overview of the network's overall situation, the possibility of alerting developers in the case of problems and, for example, the possibility of defending oneself against accusations in the case of an attack by having security and event logs. The cons are the fact that it slows down the network by generating numerous packets and has a cost to maintain. One negative aspect (bug) that we noticed is that every time a packet is sent for example to a port (in the first three dashboards) and this port is not recorded in the bar chart, the candle of said packets began to overlap with the candle behind it. The problem fixes on its own when the next candle closes.

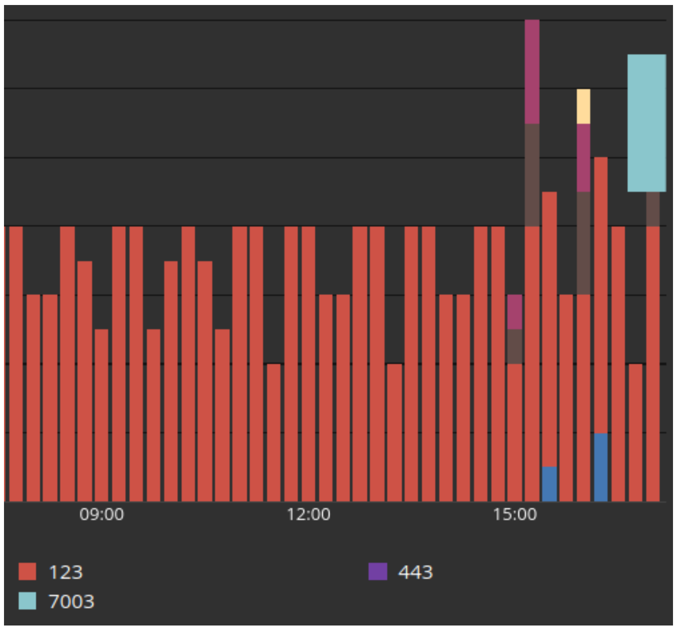


Figure 30: Bar Chart visual bug