Homework 4 Report - Group number 27

Name	Student ID	email
Simone di Biasio	1823213	dibiasio.1823213@studenti.uniroma1.it
Leonardo Persiani	1795525	persiani.1795525@studenti.uniroma1.it

0. Brainstorming

For this assignment we wanted to have a final result similar to waht Enrico showed us with his Graylog setup. We tried our best to collect relevant logs regarding relevant services and all other system logs.

The chosen software was initially Graylog, but we couldn't set it up it on the `logserver` machine since `elasticsearch` wouldn't install due to unsufficient memory, so we turned to `splunk` which has a nice web interface, comprehensive documentation, high customizability and a quite large number of plugins to handle different types of logs.

To collect logs as requested, we first redirected all of them towards the `logserver` via `UDP`, adjusting firewall rules accordingly, then we proceeded to add new input sources to `splunk`.

To test the configuration, go to http://100.100.1.3:8000 (`splunk` web interface) and use credentials `root:Passw0rd.1`. (We set a firewall rule to enable access to Splunk interface from WAN)

1. Splunk installation

We installed splunk by creating a new account on its website and by downloading the `.deb` file. After installing the instance, we used the web interface on port `TCP 8000`.

2. Redirecting Logs

Now it was time to redirect the logs: some required configuration file tricks, some `rsyslog` redirection and some used a GUI (like opnsense). Let's recall that **all logs** are forwarded via `UDP`.

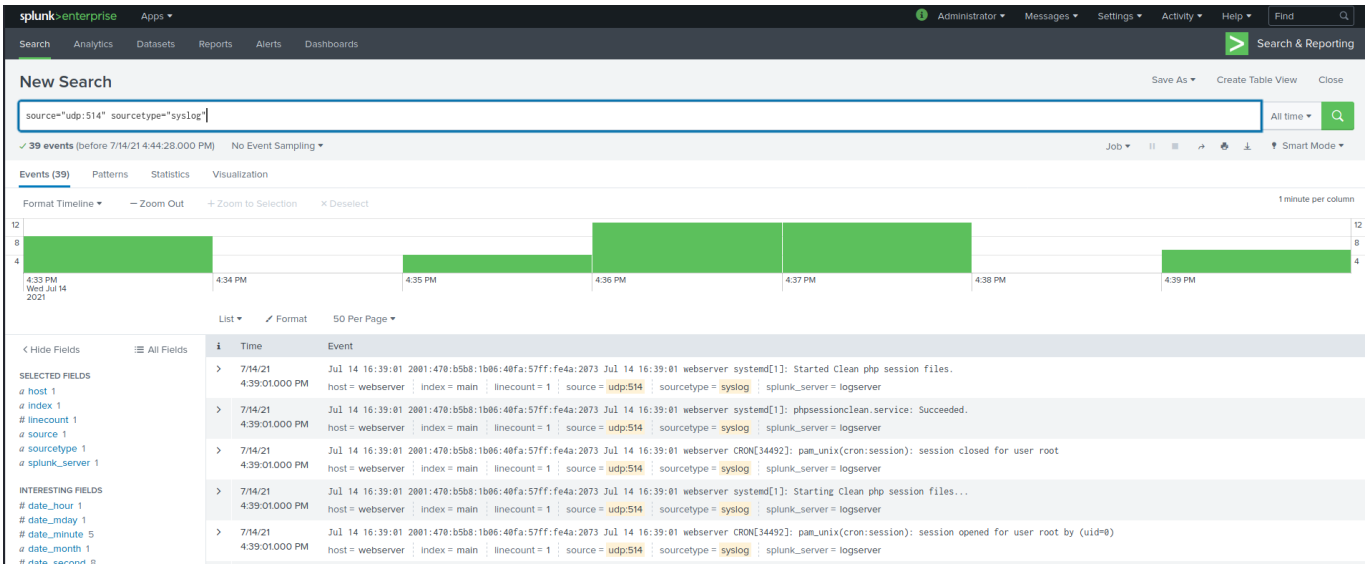
2.1 Machine Syslogs: port 514

As specified in the First Assignment, we redirected all but Client network hosts' logs to the central log using `rsyslog`: in `/etc/rsyslog.conf` we added the following line:

```
auth,authpriv,user,daemon.*             @log.acme27.com:514
```

We decided to collect system logs from machines hosting relevant services, which are basically the ones in the DMZ and Internal Servers Network; we couldn't access the `kernel` facility, probably because the machine are virtual instancies.

In splunk we set a UDP source on port UDP 514 with sourcetype `syslog`:



2.2 SQUID: port 515

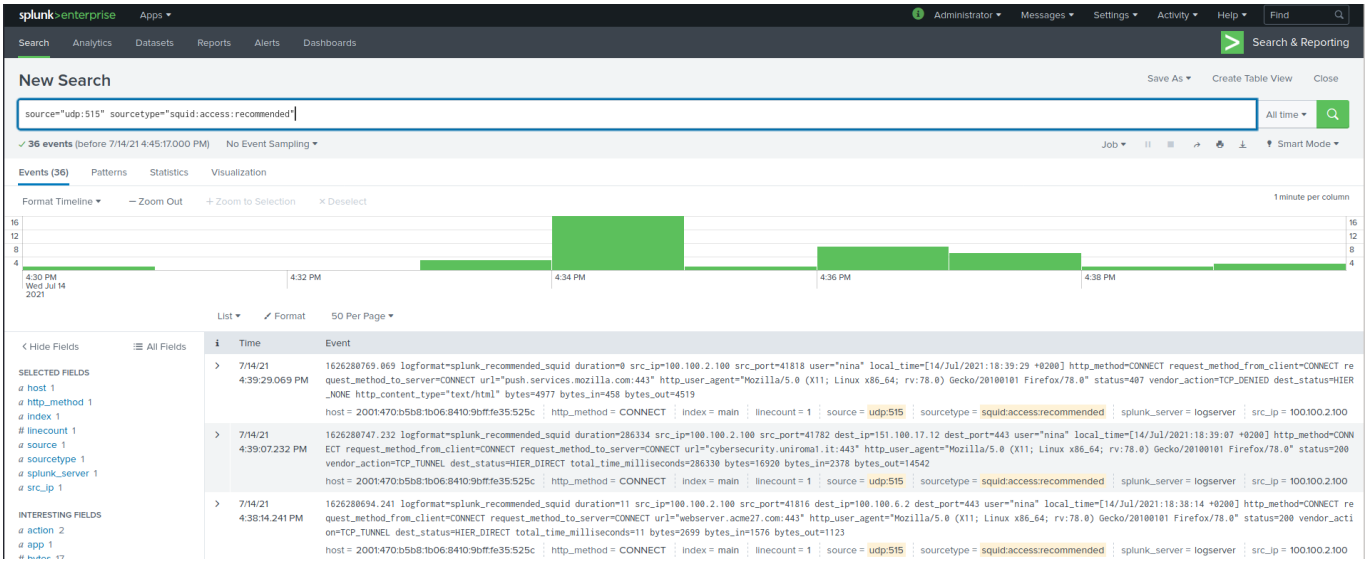
We installed the `Splunk Add-on for Squid Proxy (version 2.0.0R0b22769)` and, as suggested in the add-on documentation we changed the logging format before forwarding it:

```
logformat splunk_recommended_squid %ts.%03tu
logformat=splunk_recommended_squid duration=%tr src_ip=%>a src_port=%>p
dest_ip=%<a dest_port=%<p user_ident="%[ui" user="%[un" local_time=[%tl]
http_method=%rm request_method_from_client=%<rm
request_method_to_server=%>rm url="%ru" http_referrer="%{Referer}>h"
http_user_agent="%{User-Agent}>h" status=%>Hs vendor_action=%Ss
dest_status=%Sh total_time_milliseconds=%<tt http_content_type="%mt"
bytes=%st bytes_in=%>st bytes_out=%<st
```

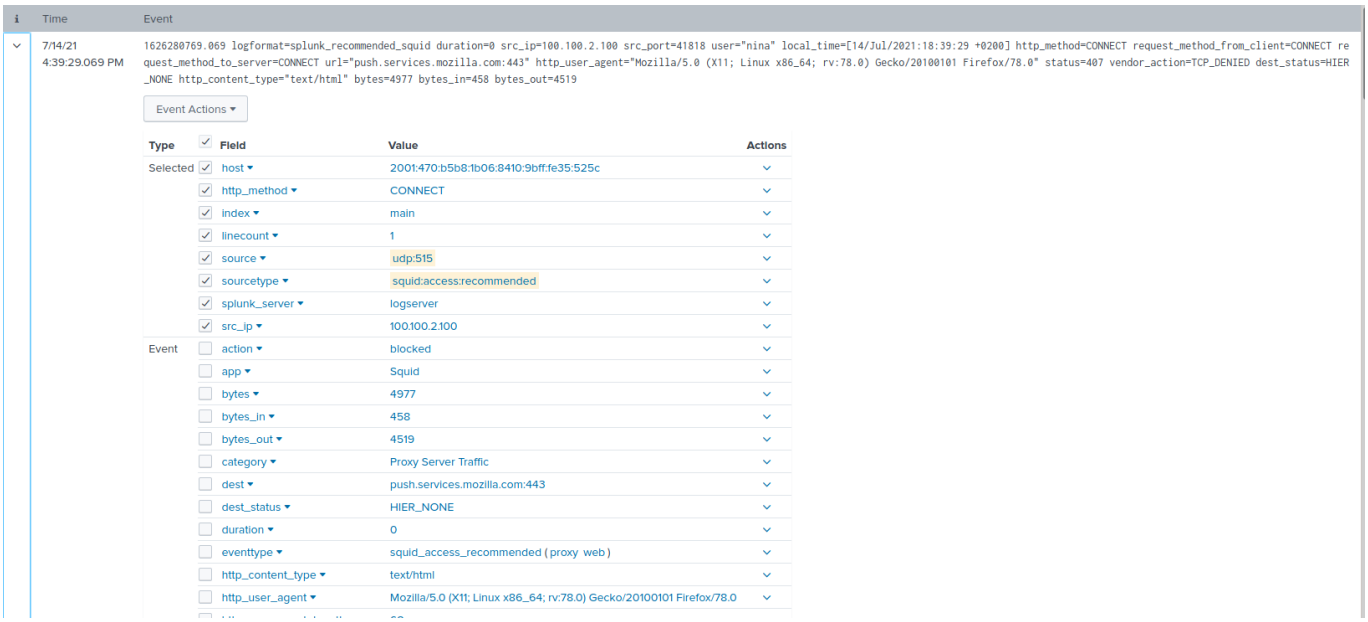
Note that we removed the last piece of it (the one regarding ssl: `sni="%ssl::>sni"`) since we're not SSLbumping. We then added this line to `squid.conf` to actually redirect the log:

```
access_log udp://log.acme27.com:515 splunk_recommended_squid
```

In splunk we set a UDP source on port UDP 515 with sourcetype `squid:access:recommended`:



Opening one of the event we can see that it is correctly parsed by the add-on:



2.3 Apache

It turns out that `squid` was the only service which didn't require any external 'help' from `rsyslog`. In this regard, `apache2` required to first redirect the log to a local logging facility and then to forward it using `rsyslog`. We also installed the `Splunk Add-on for Apache (version 2.0.0)`.

2.3.1 Reverse Proxy Access Log: port 516

We chose to not log the requests to `display.asp` generated by the `js` script inside the `index.html` because they filled up the log pretty quickly.

```
SetEnvIf Request_URI "^/display.asp$" dontlog
```

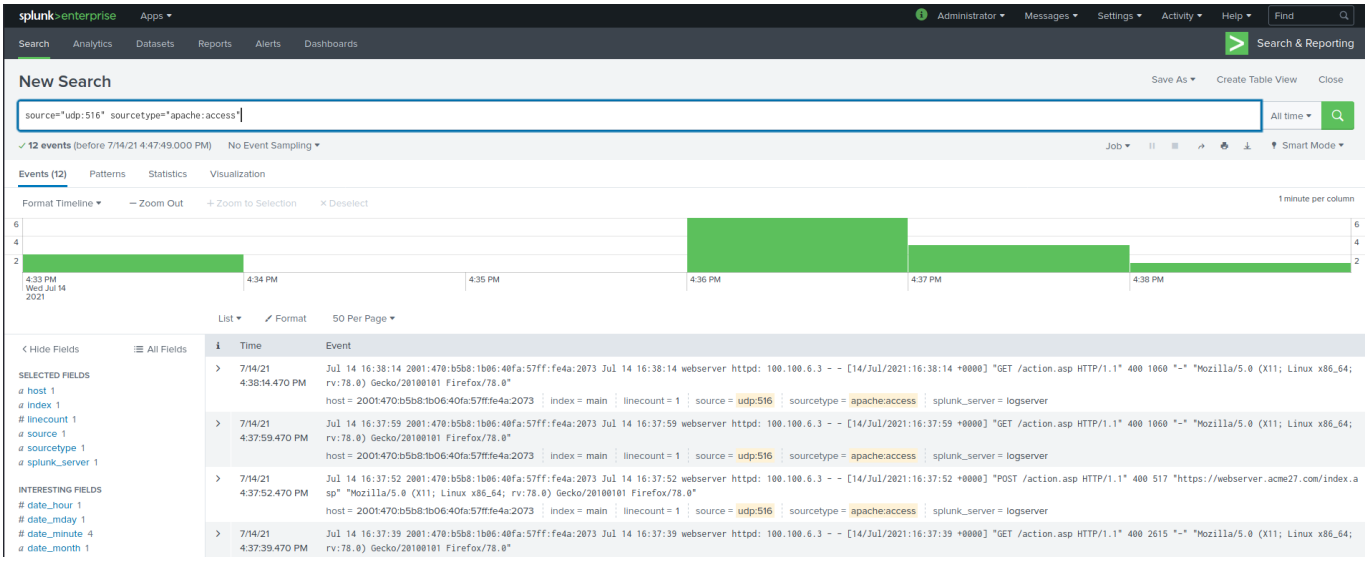
To redirect it, inside the website configuration file, we added:

```
CustomLog "|/usr/bin/logger -t httpd -p local0.info" combined env=!dontlog
```

and inside `/etc/rsyslog.d/access_apache.log` we forwarded the input:

```
local0.=info            @log.acme27.com:516
```

In splunk we set a UDP source on port UDP 516 with sourcetype `apache:access`



2.3.2 Reverse Proxy Error Log: port 517

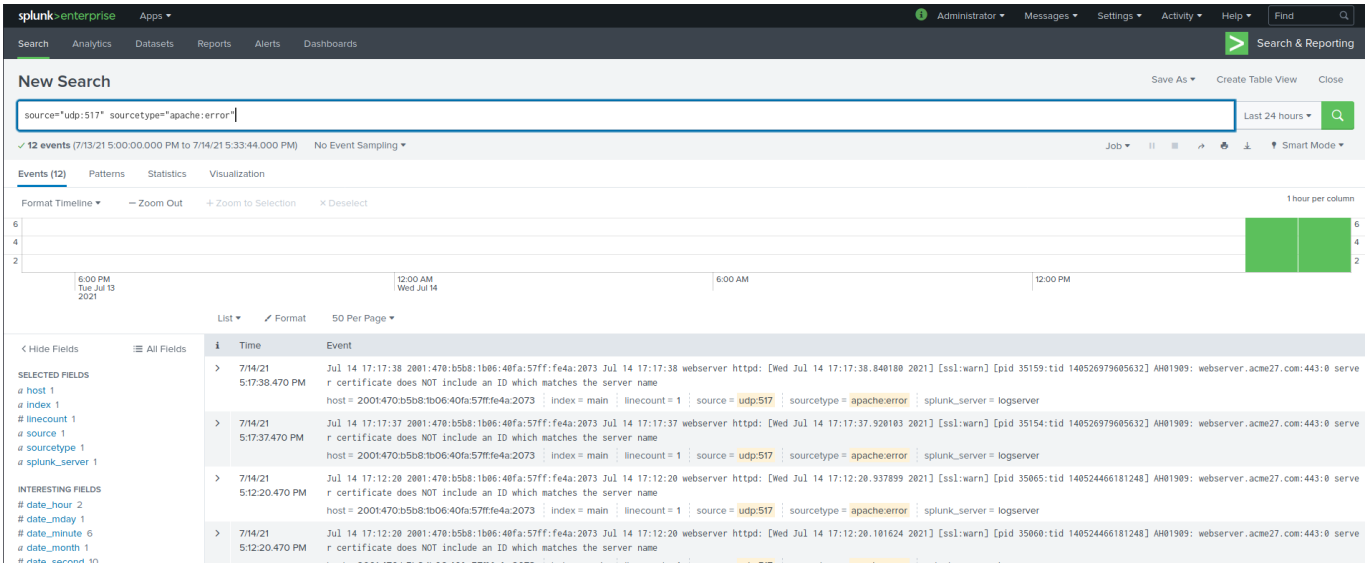
We forwarded the Error Log basically in the same way. In the conf file:

```
ErrorLog  "|/usr/bin/logger -t httpd -p local0.err"
```

and inside `/etc/rsyslog.d/error_apache.conf`:

```
local0.=err             @log.acme27.com:517
```

In splunk we set a UDP source on port UDP 517 with sourcetype `apache:error`



2.3.3 Modsecurity Audit Log: port 518

Before forwarding these logs, even though they're technically errors, we first had to made sure that the violated rules are logged exclusively to the `modsec_audit.log` file and not to the standard apache `error.log` in order to avoid duplicates and parsing errors.

In fact, the `log` action in rules definition by default writes to both files: it had to be replaced with the `auditlog` action which, as the name suggest, writes only to `modsec_audit.log` file.

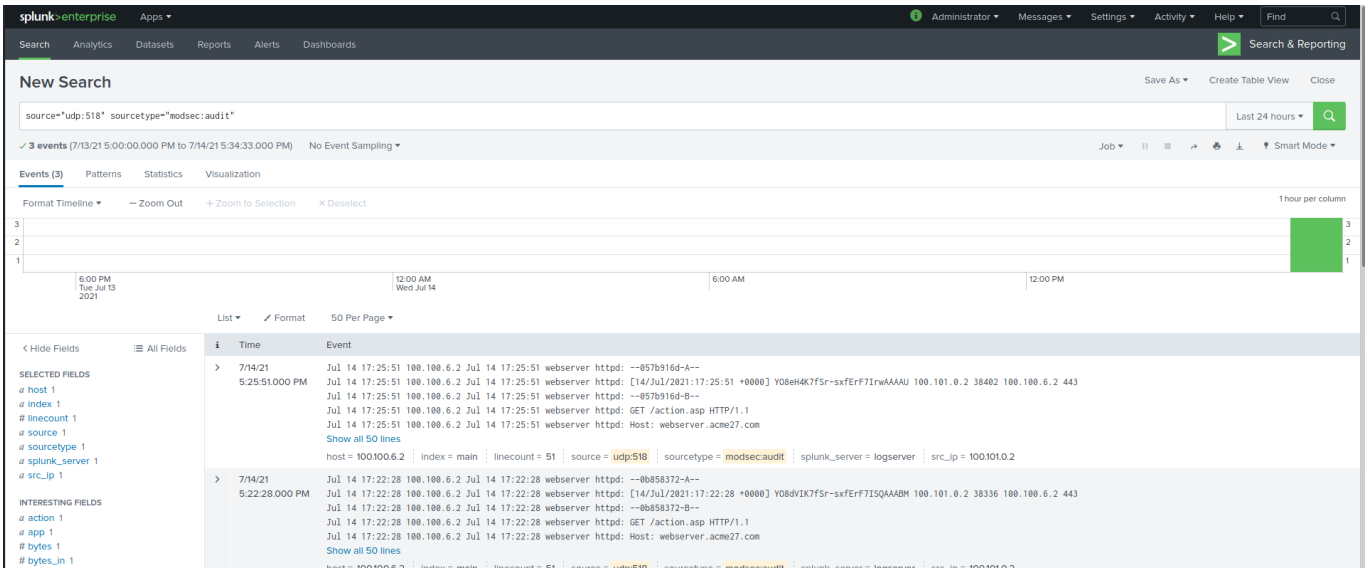
Then, in `/etc/modsecurity/modsecurity.conf`, edit the `SecAuditLog` directive using apache syntax:

```
SecAuditLog "|/usr/bin/logger -t httpd -p local6.info"
```

and finally in `/etc/rsyslog.d/modsecurity.conf`, forward the logs:

```
local6.*        @log.acme27.com:518
```

In splunk we set a UDP source on port UDP 518 with sourcetype `modsec:audit`



2.4 Bind DNS: ports 520-521

To redirect Bind DNS logs we first had to specify what to log: we settled on queries and query error. We created channels and redirected those info through them, then we forwarded via `rsyslog`.

In `named.conf.options` file, we added:

```
logging {

        channel queries_channel {
        syslog local0;
        print-time yes;
        print-category yes;
        print-severity yes;
        severity dynamic;
        };

        channel query_error_channel {
        syslog local1;
        print-time yes;
        print-category yes;
        print-severity yes;
        severity dynamic;
        };

        category queries { queries_channel; };
        category query-errors { query_error_channel; };
 };
```

and in `/etc/rsyslog.d/bind9.conf`:

```
local0.* @log.acme27.com:520
local1.* @log.acme27.com:521
```

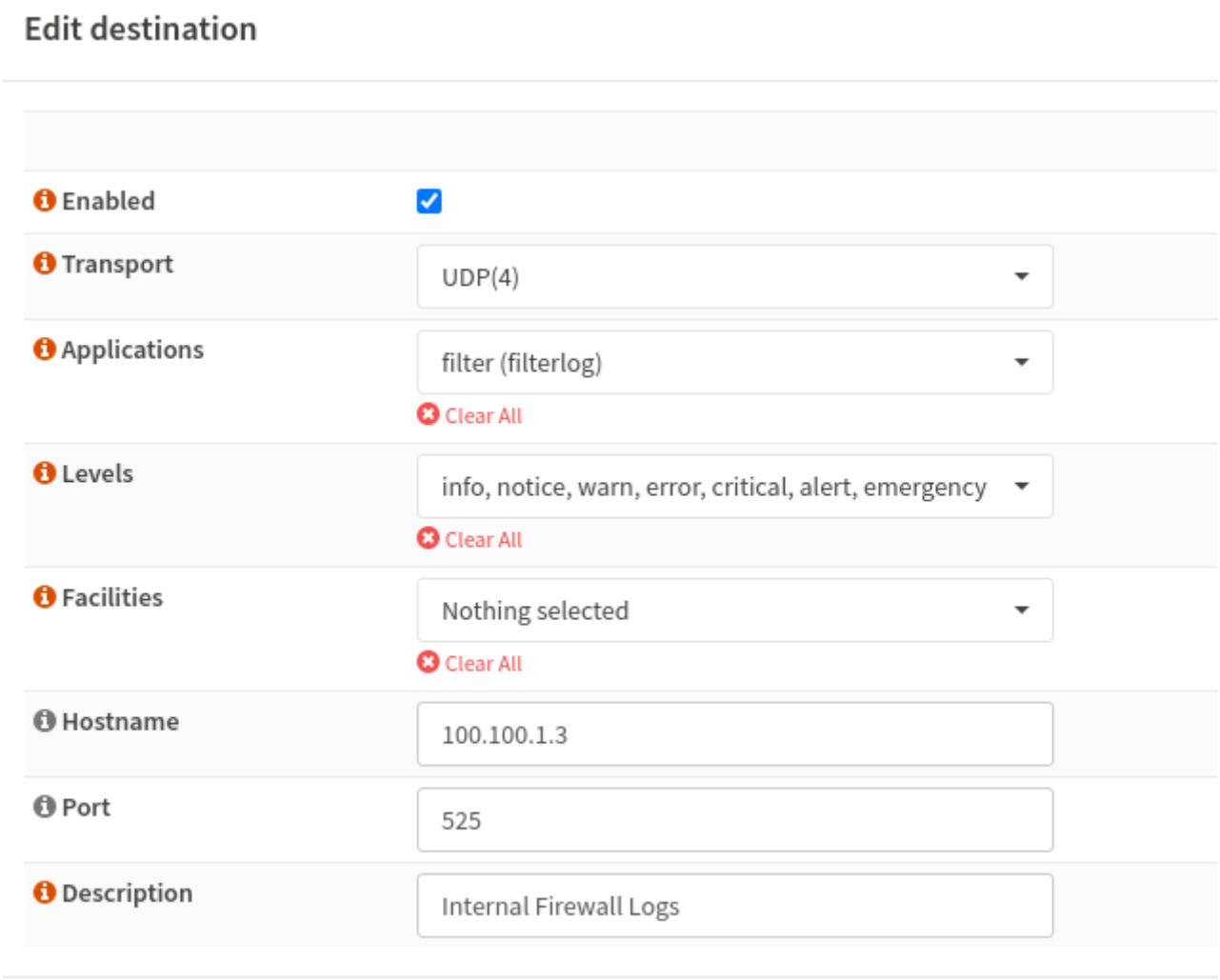
In splunk we installed the `Splunk Add-on for ISC BIND (version 2.0.0)`

2.5 OPNSense

We forwarded `OPNSense` routers log via its web interface, and we proceeded to install the `OPNsense Add-on for Splunk (version 1.4.3)` in order to correctly parse the logs.

2.5.1 Firewall Logs: port 525

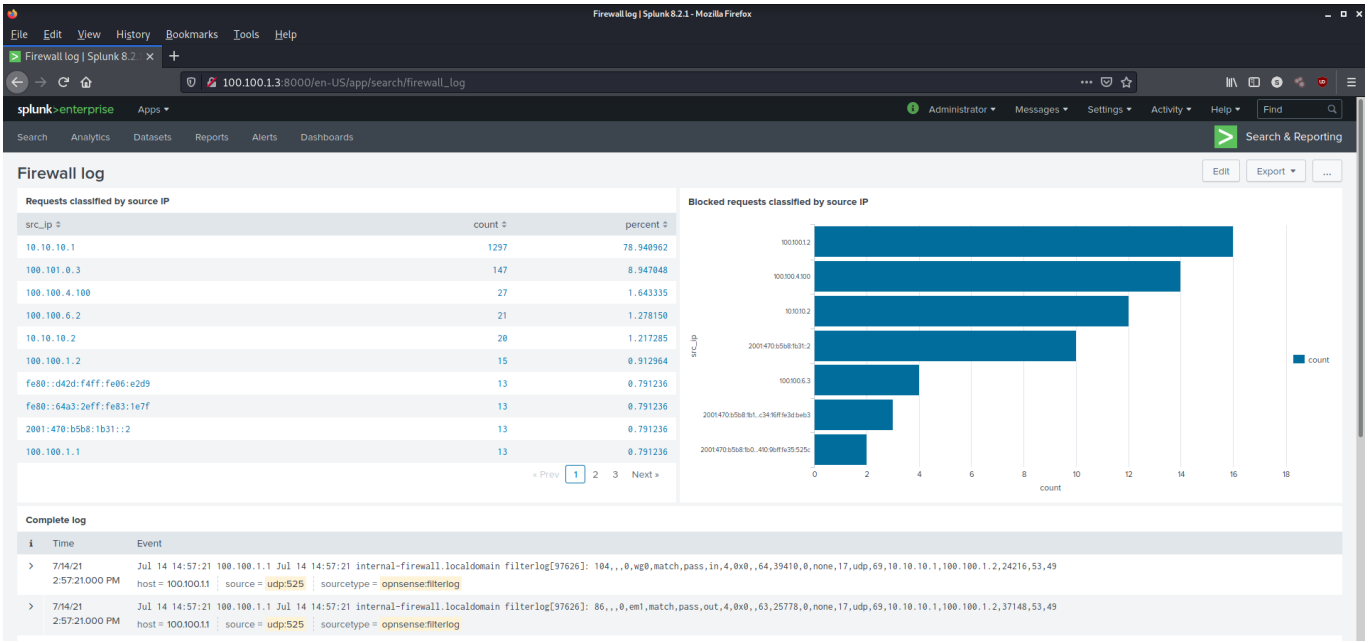
We forwarded firewall logs by selecting `filter` application in the Application section (in both routers):



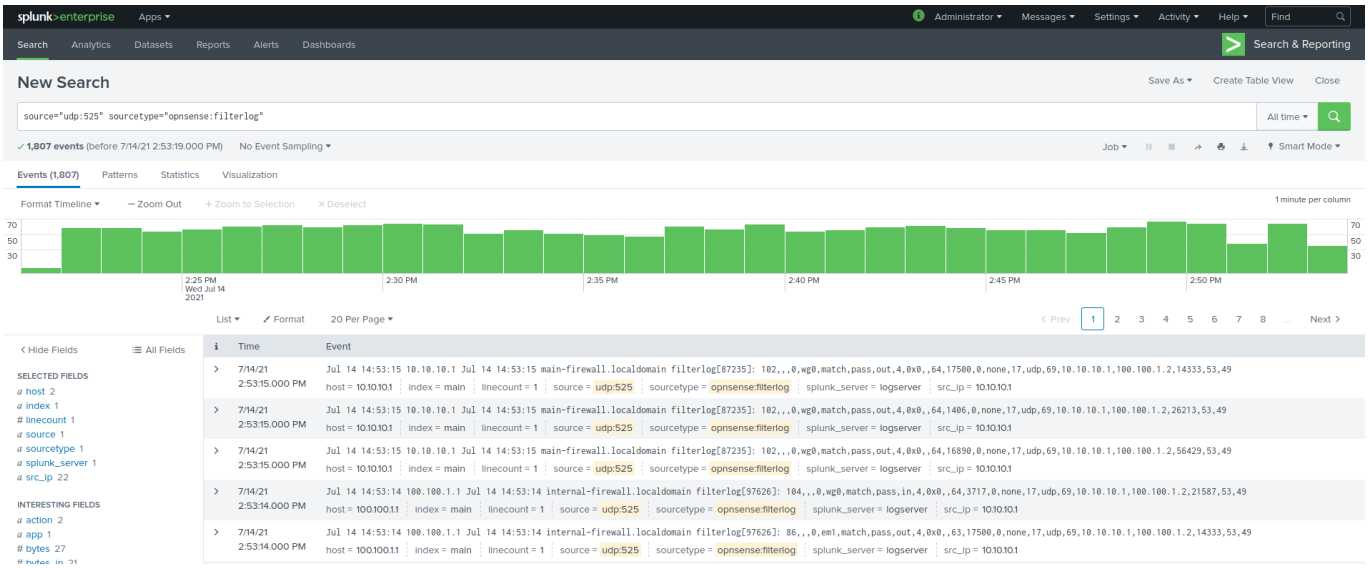
We have created a dashboard to better visualize the firewall logs.

- the top-left there is a statistics table with the requests sorted by source IP: `source="udp:525" sourcetype="opnsense:filterlog" | top limit=100 src_ip`

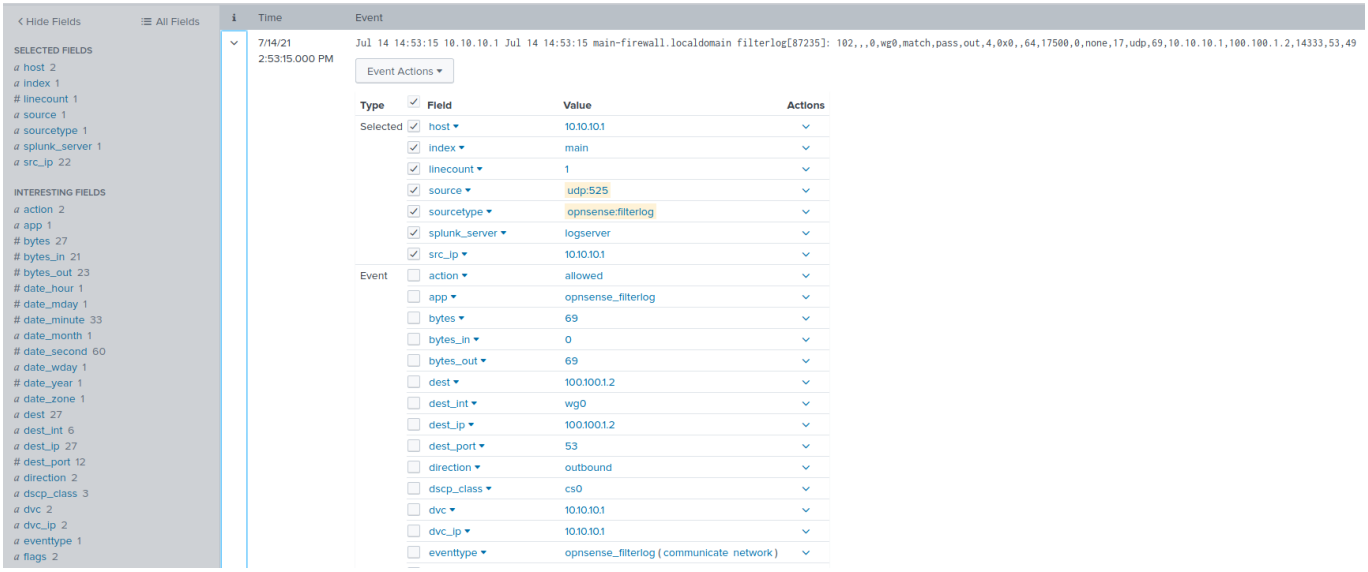
- on the top-right there is a bar chart with the blocked requests sorted by source IP: `source="udp:525" sourcetype="opnsense:filterlog" vendor_action=block | top src_ip`
- in the lower part instead, there is the complete log: `source="udp:525" sourcetype="opnsense:filterlog"`



By clicking on the table, on the bar chart or on the complete log, the search and reporting page opens with the relative filter applied:



We have installed `OPNsense Add-on for Splunk` to improve the parsing and inspection of fields in the firewall logs:



3. Test of the configuration

To test that the logs were correctly received and parsed from the Splunk inputs we simply used ACME 27 services:

- login/logout with ssh to test syslog
- proxied requests from the clients network to test squid logs
- dns queries to test bind logs
- bad requests to the reverse proxy to test modsecurity
- traffic blocked by the firewalls to test filterlog

The results of these tests are contained in the images seen so far.

4. Final remarks

Interestingly, only certain services managed to use IPv6 as internet protocol: if we put logserver's domain name in its configuration file, `rsyslog` would prioritize the IPv6 resolution over the IPv4 counterpart.

Since we didn't have a lot of time, and given the complexity of the software, we tried our best to make `Splunk` run at full power by installing add-ons, diversifying inputs and adding a dashboard which describes probably the most important aspect of our acme network, which is the firewall's traffic.

Overall, even though we spent *a lot* of time dealing with strange issues and obscure errors, we really enjoyed this hands-on approach which certainly has taught us a lot more than we could've imagined.