FRAFOS ABC Monitor Handbook

Release 5.4.13

FRAFOS GmbH

Mar 17, 2025

Contents

Chapter 1

5.4 Release Notes

For a detailed list of all changes, please check the *changelog*.

Chapter 2

ABC Monitor changelog

2.1 v5.4.0

2.1.1 Added

- monitor disk inodes usage too (G#8288)
- New charts for CA calls (G#7717)
- add ca name to registration lost event' reason (G#8496)
- Added alerts- and frafos-rtp-stats- indices cleanup (G#9224)

2.1.2 Fixed

- use latest elasticsearch and logstash 7.x versions (7.17.24) (G#8337)
- change logstash template name to avoid conflict with default logstash template (G#8336)
- fixed the new CA chart name to be Sessions by CA (G#7717)
- Monitor cannot be accessed via SSH tunnel (G#8637)
- allow Mon gui listening on ipv6 address too (G#8580)
- call old events cleanup on "days to keep" setting change, to prevent running out of shards until periodical cleanup takes effect (G#9013)
- fixed: specified mapping type of *_tm fields in call-end event (G#9011)
- allow less then 10% of memory allocation for LS and ES (G#126)

2.1.3 Changed

- Prefere ipv4 to ipv6 to internal logstash comunication (G#8322)

2.1.4 Removed

- unused monit removed (G#470)

Chapter 3

Installation

The ABC Monitor is monitoring application distributed in form of a container (systemd or OCI type). It can be run on operating system of customer choice, if the OS supports running of those container types. The recommended and tested host OS is Debian 12 (Bookworm).

In case of problems please contact the FRAFOS support at support@frafos.com.

3.1 Recommended server configuration

Minimum server configuration for ABC Monitor is:

- 1 x CPU with 4 cores
- 8GB memory
- 256GB disk
- 1 x 1Gb NIC interface

Recommended server configuration:

- 2 x CPU with 4 cores
- 32 GB memory
- 1TB SAS SATA SSD
- 2 x 1Gb NIC interface

Recommended file system to use: `ext4`. Note that depending on use case, if traffic logs and audio recordings are being synced from Sbc(s) to the Monitor and the retention policy is configured to keep them for long time, there may be many small files stored on the Monitor filesystem. In that case, it is recommended to tune the blocks / inodes ratio when formatting the filesystem, to allow storing of higher number of files then the default ext4 values allow.

3.2 Container Installation

The installation steps are similar to installation of Cluster Config Manager container, please refer to Sec-Install section for details.

Note: it is also recommended to use separate host server directory for the `/data` container volume path holding persistent data. It will allow smooth container replacement with newer version, without need to do backup / restore.

Starting 5.4.0, Frafos ABC Monitor allows connection to an external Elasticsearch database. As such, the Elasticsearch process can be externalized to an dedicated container. An existing Elasticsearch database can be used.

If none available, Frafos recommends the use of the official OCI image one. Refer to *Elasticsearch container (recommended)* for more information.

3.2.1 Podman

ABC Monitor OCI images can be downloaded from Frafos' docker registry (registry.frafos.net) using the following:

```
% podman pull registry.frafos.net/abc/mon:5.4
```

The access to container registry requires authentication using username and password. The account can be self-created at https://registry.frafos.net/, but the account has to be then assigned to proper project by Frafos. Please contact Frafos support, if your account is not approved yet for access to the ABC Sbc repository project.

The podman command to pull container will prompt for username and password, or this commandline option can be added to pass that:

```
% --creds=[username[:password]]
```

The ABC Monitor container can be created similarly to Cluster Config Manager, using proper capabilities, mounting `/data` directory and using the management network with explicitly specified IP address and DNS server:

```
% mkdir -p /var/data/mon
% podman run \
    --name mon \
    --hostname mon \
    -d \
    --tty \
    --network=mgmt:ip=172.22.31.101 \
    --dns=172.22.31.2 \
    -v /var/data/mon:/data \
    --cap-add=AUDIT_WRITE \
    --cap-add=AUDIT_CONTROL \
    registry.frafos.net/abc/mon:5.4
```

Upgrade Procedure

For a container upgrade, please start by pulling the newest images:

```
% podman pull registry.frafos.net/abc/mon:5.4
```

then stop, remove and re-create the container:

```
% podman stop mon
% podman rm mon
% podman run \
    --name mon \
    --hostname mon \
    -d \
    --tty \
    --network=mgmt:ip=172.22.31.101 \
    --dns=172.22.31.2 \
    -v /var/data/mon:/data \
    --cap-add=AUDIT_WRITE \
    --cap-add=AUDIT_CONTROL \
    registry.frafos.net/abc/mon:5.4
```

3.3 Initial Configuration

The ABC Monitor GUI can be accessed using web browser at https://<ABC Monitor IP addr>:445/.

Note that the ABC Monitor gui https access uses non-standard port 445 by default, to prevent possible port conflict if the container is running on host together with CCM container (which uses port 443 for gui access). The (nginx) port can be tuned in ABC Monitor settings. Please pay attention while setting up the port forwarding of on the containerization tool.

The ABC Monitor GUI now authenticate user using CCM. On first login, it asks for hostname of CCM and for the Elasticsearch database URL. It shall be possible to resolve the CCM hostname by the ABC Monitor and also by browser of the user accessing the GUI. To make single-sign-on and logout working reliably, both CCM and ABC Monitor shall be accessed via a FQDN in a single domain (e.g. ccm.example.com and monitor.example.com).

To make the authentication working, you have to also configure a pair of RSA keys on the CCM GUI. See the ccmloginparameters.

Use the defined Elasticsearch database URL on the initial screen > 'URL of Elastic Search' field. The value can be edited later under Config > 'Logstash and ElasticSearch' > 'ElasticSearch server address'

Please refer to the *Elasticsearch container (recommended)* chapter for more information regarding the expected Elasticsearch database URL.

Once the ABC Monitor initial configuration done, login into the GUI via HTTPs (port 445 by default), and set some basic settings under the Settings / General page.

PLEASE SET at least memory options for the logstash processes, to correspond to half of the total available system memory.

- Enable receiving events only via encrypted input: by default, ABC Monitor accepts events and replicated files (pcaps, recordings) coming both via normal unencrypted channel (which is default and has better performance) or encrypted using TLS. The way events are sent either non-encrypted or encrypted is configured on ABC SBC side. If this option is enabled, the ABC Monitor will accept only events coming via the TLS encrypted channel.
- Peer certificate verification level, if TLS used: if the events are being sent to ABC Monitor encrypted using TLS, this option sets the level of verification of the ABC SBC side certificate. Possible values are: 0 - ignore peer certificate, 1 - verify peer certificate if present, 2 - verify peer certificate, 3 - verify peer with locally installed certificate, 4 - ignore CA chain and only verify peer certificate. Default value is 0.
- Monitor GUI https port: sets the https port on which ABC Monitor GUI is accessible. Default value is 445. If the container is going to be started with port mapping (*–port 445:445*), please update that mapping according to this settings value.
- Number of days to keep old events before deleting them: this sets the time period, for how long ABC Monitor will keep the monitoring data collected from ABC SBC nodes. The data is stored in /data/elasticsearch directory. Default setting is 30 days. Amount of disk space used highly depends on the monitored nodes traffic. This setting is used also to delete old log files.
- Minimum partition free space percentage before deleting old events: if the free space on the partition where events are stored (that is partition holding /data/elasticsearch directory) drops below the value set, oldest events will be automatically deleted until the free space percentage is again above limit. Default value is 20. Set to 0 to disable the automatic deletion based on free space. Note: it is highly recommended to use separate partition for the /data/elasticsearch directory, otherwise old events may be deleted even if something else occupied the disk space. On SSD disks it is recommended to keep about 20% of free disk space for better performance and longer disk life.
- Time in minutes to keep old traffic pcap files before deleting them: this sets retention period for traffic pcap files that are synced from ABC SBC nodes to ABC Monitor, if it is enabled in ABC SBC global config. The setting must be greater or equal than the retention setting on SBC. (**"Global Config → Events → Number of days to keep old traffic log files"**), failure to dimension the period correctly may cause ABC SBC to keep uploading the files repeatedly. The data is stored in /data directory.

- Time in minutes to keep old recordings files before deleting them: similar setting like the previous one, but for recordings file, if replicating enabled in ABC SBC global config. The setting should be also equal to corresponding setting on ABC SBC. The data is stored in /data directory.
- Firewall: source IP addresses or subnets allowed to access Monitor GUI over https: limits access to ABC Monitor GUI using system iptables rules only from specified list of IP addresses, IP subnets or hostnames. You can specify more items separated by spaces. For IP subnets, use the CIDR notation like 192.168.0.0/24. It is highly recommended to limit access only from particular addresses or network subnets, to minimize security risks.
- Firewall: source IP addresses or subnets allowed to access Monitor over ssh: similar to the previous setting, but limiting access to ssh daemon.
- Firewall: source IP addresses or subnets allowed to push events to Monitor: limits access to pushing events from ABC SBC to ABC Monitor only from specified list of IP addresses or subnets. More items can be separated by spaces. It is highly recommended to limit access to ABC Monitor only from particular addresses, to minimize security risks.
- Monitor name: sets ABC Monitor name that will be displayed in GUI dashboards heading.
- E-mail TO for alerts: sets recipient e-mail address for sending alerts. Use empty value to disable sending of alerts.
- E-mail FROM for alerts: sets e-mail From address to be used when sending e-mail alerts.
- E-mail address for reports: sets recipient e-mail address for sending reports. (Note: the reports are not existing yet in ABC Monitor 4.3 and 4.4 releases, this option is prepared for later releases.)
- SMTP server address for sending emails: sets the SMTP server address that will be used as e-mail relay. If empty, defaults to "localhost". Important: there is no SMTP server running on Monitor container, it has to be provided by customer.
- SMTP server port: sets the SMTP server port. Default value is 25.
- SMTP server - use TLS: if enabled, TLS will be used on the connection to the SMTP server. Default is disabled.
- SMTP server authentication method: specifies the authentication method used for connection to the SMTP server. Default value empty means no authentication. You can use values like "plain", "login".
- SMTP server authentication username: if SMTP authentication is used, this option specifies the authentication username.
- SMTP server authentication password: if SMTP authentication is used, this option specifies the authentication password.
- Logstash heap memory size percentage: sets the amount of memory to use as heap for the Logstash process, which is the ABC Monitor part receiving and filtering events, as percentage from total system memory. It is recommended to use 20% of system memory. Use value without the "%" suffix.
- Elasticsearch heap memory size percentage: sets the amount of memory to use as heap for the Elasticsearch process, which is the ABC Monitor database part storing events. It is recommended to use 40% of system memory. Use value without the "%" suffix. Only works with local Elasticsearch database.

On the ABC SBC side, to enable pushing of monitoring data to the ABC Monitor, you have to configure the ABC Monitor IP address in "ABC Monitor address" setting of Global Config, under Events tab.

By default, the pcap and traffic recordings files are copied from ABC SBC nodes to ABC Monitor using basic rsync protocol, which works without any extra configuration. If needed, rsync over TLS can be used if enabled in Global config on ABC SBC side.

The https access to ABC Monitor GUI uses automatically created self-signed certificate by default. If real trusted certificate is going to be used, it can be imported using the gui Settings.

3.4 Elasticsearch container (recommended)

Starting 5.4.0, Frafos ABC Monitor allows connection to an external Elasticsearch database. As such, the Elasticsearch process can be externalized to an dedicated container.

Frafos ABC Monitor 5.4 still ships an Elasticsearch process inside of the Mon container, but it may be removed in future release. On the initial login screen admin can choose whether the local Elasticsearch provided as part of ABC Monitor container should be started and used, or if external Elasticsearch provided by separate container will be used. This option can be also modified later under gui Setting page "Local Elasticsearch database" option.

For the management of the dedicated Elasticsearch container, the official_documentation should **always** be prioritized to the Frafos one. Frafos suggestions and example are **not** suited to all use cases, and shouldn't be used as so for production. Nevertheless, they can be used as base to iterate from.

For Elasticsearch data export from a ABC Monitor < 5.4 to a detached container, please refer to *Export Elasticsearch data via snapshot*

3.4.1 Monitor settings value

If the Elasticsearch and ABC Monitor containers should be run on the same host or inside of the same pod, the 'http://127.0.0.1:9200' value should be used for the 'URL of Elastic Search' configuration field (initial / configuration screens). If the container is running on a remote server, please use that host IP instead of localhost.

Ex: the Elasticsearch container is reachable on the '10.1.1.42' remote IP, the value 'http://10.1.1.42:9200' is used for 'URL of Elastic Search' field.

3.4.2 Test setup

A running Elasticsearch server can be queried from the host to ensure the node startup. In the following example, curl was installed on the host:

```
% curl -X GET 'localhost:9200/_cat/nodes?&pretty'
ip          heap.percent ram.percent cpu load_1m load_5m load_15m node.role   master␣
↪name
172.42.0.89           37          70  10    1.24    0.73     0.55 cdfhilmrstw *      ␣
↪a536f00cb2e6
```

3.4.3 OCI image

Note: The following examples use `podman`, but `docker` can be substituted pretty easily.

- As the ABC Monitor container needs to access the Elasticsearch server, the service port (*9200*) needs to be exposed (`-p 9200:9200`)
- For most ABC Monitor use-case, a single node cluster is enough (`-e "discovery.type=single-node"`)
- The ulimits must be increased, as per ulimits_documentation (`--ulimit memlock=-1:-1 --ulimit nofile=65535:65535`)
- Swapping needs to be disabled for performance and node stability (`-e "bootstrap.memory_lock=true" --ulimit memlock=-1:-1`) – see the swap_documentation for more informations

To allow data persistence, the content of the container' `/usr/share/elasticsearch/data` directory must be preserved. The following example can be used if no volume were created yet.

While using an OCI volume is pretty straight forward:

- create the volume `podman volume create data_es`
- attach it to the container `-v data_es:/usr/share/elasticsearch/data`

Mounting a directory from the host is a bit trickier (volume_documentation):

- create an directory `mkdir -p /data/elasticsearch`
- set the permission of the directory `chmod g+rwx /data/elasticsearch ; chgrp 0 /data/elasticsearch`
- attach it to the container `-v /data/elasticsearch:/usr/share/elasticsearch/data`

As Elasticsearch' data is safely persisted across container restart and replacement, the restart policy can be set to `always`.

Regarding memory consumption, multiple approach are doable:

- OCI run time limitation (`-m 2G` or `--memory 2G`), causing OOM killing
- Elasticsearch environment (`-e ES_JAVA_OPTS="-Xms1g -Xmx2g"`), does not survive configuration changes
- Mounting of a custom configuration file into the container

The last one requires a few preparation steps:

An *heap.options* file mus be created with the following content:

```
% cat heap.options
-Xms1g
-Xmx2g
%
```

The file must then be mounted into the */usr/share/elasticsearch/config/jvm.options.d/* location

The final command to start the Elasticsearch container via `podman` looks like:

```
% podman run \
  --name elasticsearch \
  --rm \
  --restart=always \
  --port 9200:9200 \
  --ulimit nofile=65535:65535 \
  --ulimit memlock=-1:-1 \
  -e bootstrap.memory_lock=true \
  -e discovery.type=single-node \
  -v /path/to/heap.options:/usr/share/elasticsearch/config/jvm.options.d/heap.options␣
↪\
  --volume data_es:/usr/share/elasticsearch/data \
  elasticsearch:7.17.24
```

Chapter 4

Upgrade Procedure

4.1 Container

The ABC Monitor deployed as container can be replaced with newer version.

The new ABC Monitor can be either configured again via GUI, or the old configuration is used automatically in persistent "/data" is used, or the configuration can be transferred from old container to new container. Please refer to *Backup and Restore Operations* section for details about the configuration backup and restore. The recommended way is to use persistent "/data".

The old events data can survive container replacement with newer version, if the "/data" directory of the container is either mounted externally (and not erased while setting up the new container), or if the "/data" directory content is copied or moved (while keeping ownership and access rights) from old stopped container to new unpacked container, before starting it for the first time.

Note: the "mounting" of persistent host directory to the "/data" directory of the container can be achieved using `"--bind"` option of "systemd-nspawn" command used to start the container.

Note: older ABC Monitor releases up to 4.4 used different location "/var/lib/elasticsearch" for storing the events data, while starting with release 4.5 the directory was moved to "/data/elasticsearch" to allow possible persistence across container replacement, if the whole "/data" directory is mounted externally to the container.

Note 2: If data format has changed, some charts or Monitor function maybe will not work correctly with old data format.

4.2 Export Elasticsearch data via snapshot

The following procedure should always be prioritized over *Export Elasticsearch data via system copy*. The examples used a 5.3.3 ABC Monitor (Elasticsearch 7.17.24), run trough `podman` and named `mon_53`.

The text is mostly an path adaptation of snap_documentation. In case of needs or questions, please, contact support.

Note: ABC Monitor 5.3.3+ is strongly advised as some related settings where pre-set into it. Nevertheless, Elasticsearch data export from ABC Monitor older than 5.3.3 is still doable.

Note 2: Elasticsearch data backups are strongly recommended, and can be generated by the following the described procedure

4.2.1 Create Backup

To export Elasticsearch data from a running 5.3.3 ABC Monitor container, an Elasticsearch data export (aka `snapshot`) must be generated. The snapshot is then exported to the host to allow import into the new container.

Setup snapshot directory

The directory where Elasticsearch backups are to be dumped (snapshot location) isn't explicitly set by the default Elasticsearch configuration. The following commands set the `/data/snap` directory as snapshot location:

```
% mkdir -p /data/local-templates/elasticsearch/
% cp /etc/abc-monitor/templates/elasticsearch/elasticsearch.yml.tmpl /data/local-
↪templates/elasticsearch/
% echo 'path.repo: ["/data/snap"]' > /data/local-templates/elasticsearch/
↪elasticsearch.yml.tmpl
```

Note: not required for 5.3.3+ ABC Monitor, setting already injected. The location is given as example, any valid location inside of the container (mounted system volume, OCI dynamic volume) can be used.

Prior to restarting the Elasticsearch service, the snapshot location must be created and the `elasticsearch` user and group ownership must be set:

```
% mkdir -p /data/snap
% chown -R elasticsearch:elasticsearch /data/snap
```

The Elasticsearch service can then be safely restarted (the operation can take several minutes):

```
% abc-monitor-activate-config
```

Register snapshot directory

Once the snapshot location configuration loaded, the following query must be made against the Elasticsearch API to register the location:

```
% podman exec -it mon_53 curl -XPUT 'http://localhost:9200/_snapshot/snap?pretty' \
  -H 'Content-Type: application/json' \
  -d \
  '{
     "type": "fs",
     "settings": {
       "location": "snap",
       "compress": true
     }
   }'
{
  "acknowledged":true
}
% podman exec -it mon_53 curl 'http://localhost:9200/_snapshot/snap?pretty'
{
  "snap" : {
    "type" : "fs",
    "settings" : {
      "compress" : "true",
      "location" : "snap"
    }
  }
}
```

Create snapshot

The snapshot 'es_export' can then be created, using the following curl requests:

```
% podman exec -it mon_53 curl -XPUT 'http://localhost:9200/_snapshot/snap/es_export?
↪pretty' \
  -H 'Content-Type: application/json' \
  -d '{}'
{
  "accepted": true
}
```

Note: the *es_export* name is given as example, any string value is valid

The created snapshot can be inspected as follow:

```
% podman exec -it mon_53 curl 'http://localhost:9200/_snapshot/snap/ex_export?pretty'
{
  "snapshots": [
    {
      "snapshot": "ex_export",
      "uuid": "Fn5I9cOyQNyYTAUe6guI8A",
      "repository": "snap",
      "version_id": 7171599,
      "version": "7.17.24",
      "indices": [
        ".geoip_databases",
        "data_persistent",
        "logstashh-2024.01.25",
        "lastlog-2024.1",
        "status-2024.01.25",
        ".ds-.logs-deprecation.elasticsearch-default-2024.01.25-000001"
      ],
      "data_streams": [
        ".logs-deprecation.elasticsearch-default"
      ],
      "include_global_state": true,
      "state": "SUCCESS",
      "start_time": "2024-01-25T11:04:18.698Z",
      "start_time_in_millis": 1706180658698,
      "end_time": "2024-01-25T11:04:20.299Z",
      "end_time_in_millis": 1706180660299,
      "duration_in_millis": 1601,
      "failures": [],
      "shards": {
        "total": 7,
        "failed": 0,
        "successful": 7
      },
      "feature_states": [
        {
          "feature_name": "geoip",
          "indices": [
            ".geoip_databases"
          ]
        }
      ]
    }
```

(continues on next page)

(continued from previous page)

```
  ],
  "total": 1,
  "remaining": 0
}
```

Wait for all states to be "SUCCESS" before attempting to export the snapshot.

Note: the returned value **will** vary from one run to another as they're using timestamps

Export snapshot to host

To allow the created snapshot to be restored on a external Elasticsearch container, the whole Elasticsearch snapshot directory is copied to the host:

```
% podman cp mon_53:/data/snap ~/
```

If the container directory is mounted through a system volume, that snapshot location may be copied from that host volume directly.

4.2.2 Restore snapshot

If you're not familiar with Elasticsearch containerization, please first **read** the *Elasticsearch container (recommended)* chapter.

The started Elasticsearch container version **must** match (or be greater than) the ABC Monitor one where the snapshot was created (`5.3.3` ship the `7.17.24` Elasticsearch version).

The following example assume that the new Elasticsearch container is started on the same host than the one where the snapshot was exported.

Setup snapshot directory

The snapshot directory, previously copied from the 5.3.3 ABC Monitor to the host (`~/snap` in our example), needs to get it's permissions set to the one expected by the `elasticsearch` container:

```
% chown -R 1000:1000 ~/snap
```

Setup Elasticsearch configuration

The Elasticsearch container doesn't allow any service restart post configuration file change. Instead, the container should be started with a custom *.options* suffixed file mounted into the `/usr/share/elasticsearch/config/jvm.options.d` location.

The snapshot root directory setting can be injected into an *options* configuration file the following way:

```
% echo 'path.repo: ["/data/snap"]' > ~/snapshot.options
```

Start restore instance

An one-shot Elasticsearch instance can then be started to restore the 5.3 snaphsot.

Multiple extra volume with requirements are mounted into that restore instance:

- to allow the restored snapshot data persistence across container restart, the Elasticsearch container data directory must be mounted.
- the custom configuration file must be mounted in place and the snapshot root directory must be mounted into that restore instance The snapshot directory
- the snapshot root directory must be mounted in place and necessary permissions

The following podman commands are used:

```
% podman volume create data_es
% podman run \
  --name elasticsearch \
  --rm \
  --port 9200:9200 \
  -e bootstrap.memory_lock=true \
  -e discovery.type=single-node \
  --volume data_es:/usr/share/elasticsearch/data \
  --volume ~/snapshot.options:/usr/share/elasticsearch/config/jvm.options.d/snapshot.
↪options \
  --volume ~/snap:/data/snap \
  elasticsearch:7.17.24
```

Note: The following assume that a podman volume will be used for the future production Elasticsearch container. refer to *Elasticsearch container (recommended)* for local volume usage or un-referenced podman option usage.

Register snapshot directory

Once the container started, the snapshot directory can be registered on the new Elasticsearch instace using the host' `curl` binary:

```
% curl -XPUT 'http://localhost:9200/_snapshot/snap?pretty' \
  -H 'Content-Type: application/json' \
  -d \
  '{
     "type": "fs",
     "settings": {
       "location": "snap",
       "compress": true
     }
   }'
{
  "acknowledged":true
}
```

To ensure that the snapshot directory was registered, the following command can be used:

```
% curl 'http://localhost:9200/_snapshot/snap?pretty'
{
  "snap" : {
    "type" : "fs",
    "settings" : {
      "compress" : "true",
      "location" : "snap"
```

(continues on next page)

(continued from previous page)

```
    }
  }
}
```

Restore snapshot

The snapshot `es_export`, created previously on the 5.3.3 ABC Monitor container, exported to the host `es_snap` directory, which is mounted into the Elasticsearch restore instance, can finally be restored using the following command from the host:

```
% curl -XPOST 'http://localhost:9200/_snapshot/snap/es_export/_restore?pretty' \
  -H 'Content-Type: application/json' \
  -d '{ \
   "indices": ".geoip_databases,data_persistent,logstashh-2024.01.25,lastlog-2024.1,
↪status-2024.01.25", \
   "ignore_unavailable": true,  \
   "include_global_state": true \
  }'
  {
    "accepted":true
  }
```

Note: the following command used some value with timestamp inside of them (`2024-01.25`). These value **must** match the `indices` one returned when inspecting the created snapshot.

Once the snapshot restored, the Elasticsearch restore instance can safely be stopped. Please note that the same Elasticsearch data volume from the restore instace must be used on the production instance one (`--volume data_es:/usr/share/elasticsearch/data`).

4.3 Export Elasticsearch data via system copy

Note: the following procedure is not recommended, as using snapshots is the official approach It can be followed as it should work in most cases. In case of start time failure from the new Elasticsearch container instance regarding data loading, please refer to the following: *Export Elasticsearch data via snapshot*.

To allow data export from a existing ABC Monitor container to a detached Elasticsearch one, a copy of an the existing container data volume can be used.

Stop the `elasticsearch` process from the running ABC Monitor container before any other step:

```
% systemctl stop elasticsearch
```

On 5.2+ ABC Monitor container, the Elasticsearch data is stored at the following location: `/data/elasticsearch`. The content of that directory must be placed into a volume that will be used for the future Elasticsearch container.

If the ABC Monitor container is running with a system volume mounted at the `/data` location, use the following:

```
% mkdir /data
% cp -rf /host/path/mon/data/elasticsearch /data/elasticsearch
```

If the running ABC Monitor container has an OCI volume mounted at the``/data`` location, or no volume mounted there, use the following:

```
% mkdir /data
% podman copy [monitor container name]:/data/elasticsearch /data/elasticsearch
```

The permissions of the future Elasticsearch container data volume need to be set accordingly:

```
% chmod g+rwx /data/elasticsearch
% chown -R 1000:1000 /data/elasticsearch
```

An detached Elasticsearch container instance can then be started, using the exported volume as data partition:

```
% podman run -v /data/elasticsearch:/usr/share/elasticsearch/data elasticsearch:7.17.
↪24
```

Chapter 5

Backup and Restore Operations

5.1 Configuration Backup

A backup of the ABC Monitor main configuration files can be created from command line using "abc-monitor-backup-config" command.

By default, the file name of the backup is created automatically using current date and time and the file is saved to "/data/backups/configs" directory.

Note: starting with 5.0 release, the backup file is a gzipped tarball, holding the main ABC Monitor config files inside. In pre 5.0 releases, the backup file format was just plain JSON file. It is still supported to restore the older format on >= 5.0 release.

The configuration backup contains only ABC Monitor application settings: main "monitor.json" and "monitor-layout.json" config files, and gui access credentials "htpasswd" file. It does not contain any system settings (like hostname, network interfaces settings and similar).

The following options can be used:

```
%  --file <filename>
```

Specify full path of the file to which the configuration backup should be saved. If not used, the file name is created automatically and default directory used.

```
%  --quiet
```

Do not write any info messages. If used, only errors are written, plus a message like the following: "Config backup saved to: /data/backups/configs/monitor-2020-01-16_14-54-18.tgz".

5.2 Configuration Restore

A backup of the ABC Monitor main configuration can be restored from command line using "abc-monitor-restore-config" command. It checks the configuration file, restores it as the main config file and optionally also activates the new configuration.

The following options can be used:

```
%  --file <filename>
```

Sets full path to the file with the ABC Monitor configuration backup to be restored. This option is mandatory.

```
%  --noactiv
```

By default, the restored ABC Monitor config is activated, meaning that services which are affected by the new config are restarted. If this option is used, the config backup is restored but no activate done. Admin can review the restored config in ABC Monitor GUI and use Save from there to activate it.

```
%  --quiet
```

Do not write any info messages. If used, only errors are written.

Chapter 6

Overview

The ABC Monitor provides administrators with an aggregated view of user activity based on usage reporting data collected from the ABC SBC/WebRTC gateways. This highly interactive, near real-time view can be used for trending, analysis of both short-term and long-term use patterns, troubleshooting, auditing server policies and identifying misconducting users. The reporting data comes using events from inside of the SBCs. This "insider view" allows the ABC Monitor administrators to inspect SIP traffic encrypted on the way from and to the SBCs, correlate calls "separated" by topology hiding, and report internal ABC SBC context such as traffic shaping decisions.

If there are multiple SBCs organized in a hot-standby pair, or a cloud the ABC Monitor will collect data from all of them and its centralized nature provides a global view of the whole system. An ABC SBC may also send its data to two Monitors in parallel. This is useful for various organization with multiple isolated teams, easy-to-start virtualized trials and migration scenarios.

The ABC Monitor user interface is organized in several dashboards. The opening Home Dashboard shows the most important data in a single comprehensible page, such as shown in Figure *ABC Monitor Home Dashboard*. All of the data relates to the period of time chosen in the top right corner. This page can also be sent to administrator on a daily basis by email to report on previous 24 hours.

The data in the home dashboard is structured in several rows. The first row shows various call metrics, such as number of completed and attempted calls, total number of minutes, etc. The second shows how frequent were events of the various types in the observed period of time. Dark fields represent many events of a kind. In this example we see that greylisting events were dominating at a time slot, a situation that often occurs when a SIP scan is launched on a public SIP service.

The next two rows show history of number of parallel calls and registrations, also compared with data from previous day shown using a thin line. In the boxes on the right hand side there are current numbers.

The last row shows number of security-related events. The timeline is divided in buckets and the number of events relates to each of the buckets. The bucket length grows proportionally with the time window. The number on the right-hand side shows number of security events in the most recent bucket. The number's background color changes with the number and is green when smaller or equal to five, orange bellow ten, and red above.

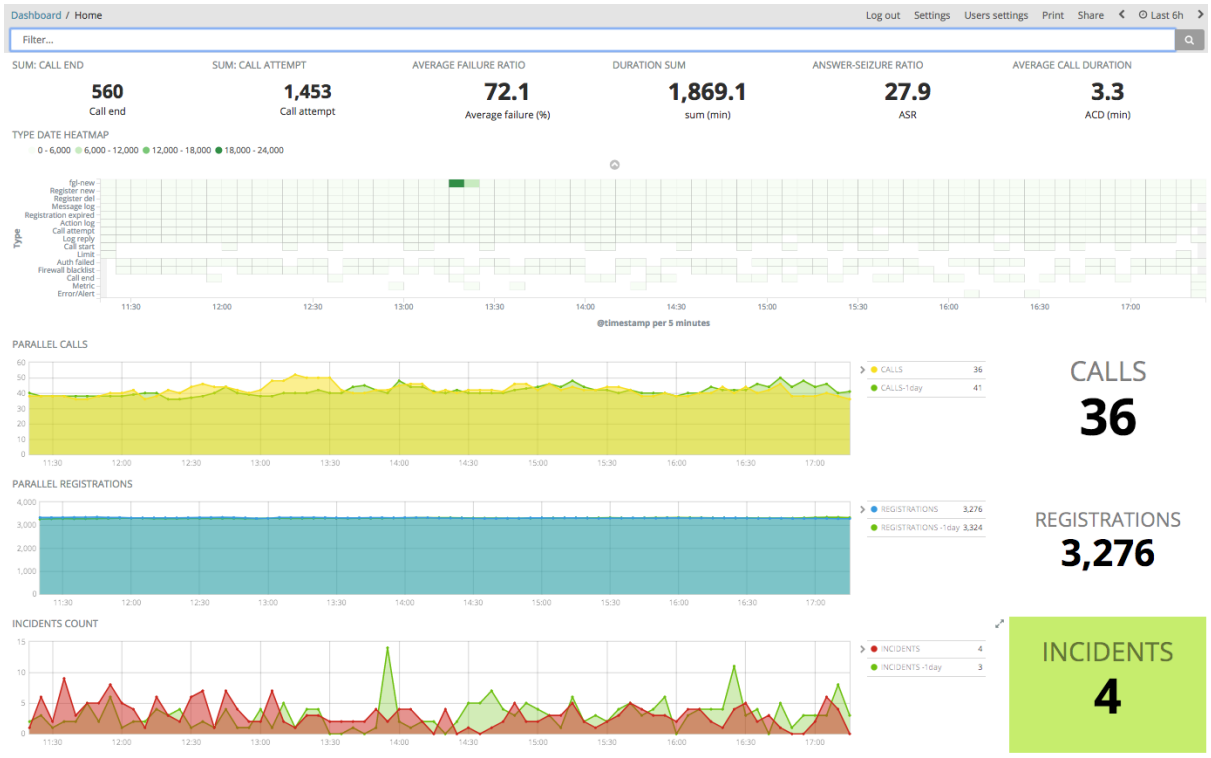

Fig. 1: ABC Monitor Home Dashboard

A snapshot of the home dashboard can be produced and sent as PDF attachment by Email every morning if a recipient email address is configured under "Settings → Reports: e-mail address for everyday reports".

All other dashboards are similarly organized. While they are organized along different aspects of SIP operation and show therefore different data, the visual structure follows the same pattern. In the very top, there are filters that allow to limit the events based on various criteria. Using the filters is described later in Chapter *Using Filters*. In the mid part, there are various graphical elements showing either some aggregated values, or their history over time. In the very bottom, there is a list of the actual events. The events can be clicked on to unfold and view all details.

The most frequently used dashboards are the following:

- "Call Dashboard" shows history of calls, analysis of failures and QoS reports. This is helpful to identify call trends, volumes, reasons of call failures, and troubleshooting specific calls. It provides both aggregate view of the situation as well as the possibility to review call details. More details are shown in Chapter *Calls Dashboard*.
- "Security" shows all Security events, also organized by the most active IP addresses and geo-locations.
- "Toplists" shows most active users by various metrics: call attempts, call minutes, number of short calls, etc.
- "Exceeded Limits" helps to alert on abnormal situations, such as excessively many phone calls from a single IP address. See Chapter *Exceeded Limits Dashboards* for more details.
- "Overview" holds all events relating to a user. There are really many. This is mostly useful when an administrator begins to suspect a problem and wants to see full user's history. Often administrators get alerted on a user when reviewing security dashboard, toplists, or exceeded limits, set up filter for such user, and inspect then his full event history. Additional details are shown in Chapter *Overview Dashboard*.
- "Connectivity CA" operates at topology layer and shows how good peering between calls from one Call Agent to another Call Agent is. This helps to identify Call Agents with poor performance, and/or Call Agents that have troubles making calls with each other. See Chapter *Connectivity CA Dashboard*.
- "Registration" shows registrations: new, expired and deleted, which transport is being used, from which part

of the world are the registrations coming, and what SIP equipment is being used. This dashboard is useful for troubleshooting SIP user's connectivity. More information can be found in Chapter *Registration Dashboard*.

There are even more dashboards that provide less aggregated data that is useful when trying to understand a low-level problem.

- "Diagnostics" collects troubleshooting information. If an SBC administrator chose to record PCAP files, WAV files, produce custom event, or an unusual OS situation is reported, it appears here. See Section *Diagnostics Dashboard* for more information. Diagnostic events relating to layer-3 and layer-4 are separated in "Transport" dashboard and provide useful information to detect frequent retransmissions, or various TLS handshake failures.
- "Connectivity" is a dashboard operating at URI level that shows the most active caller-callee pairs.
- "Network Statistics" shows low-level data such as number of parallel calls, active registrations, bytes sent and received, etc. See Section *Network and Statistics Dashboard*. "Realms Stats" is a subset of network statics, broken down by realms.
- "Systems" shows how SBCs are doing in terms of memory and CPU. Useful to identify overload situations. See Section *System Dashboard*.

6.1 Events (optional)

Note that producing events is an optional feature that requires an additional licensing option and the ABC Monitor software.

The events collect a detailed history of user activities. With this data, it is possible to review in detail the history of a specific user as well as the whole SIP service. The events are produced by the ABC SBC in course of processing SIP and RTP traffic whenever some relevant action occurs: a call was attempted / established / terminated, a bandwidth threshold was applied, etc. Administrators may also choose to generate their own custom events.

The events are keyed by SIP address and IP address so that history of specific users can be easily established.

All events include three types of fields: mandatory, type-specific and call variables.

Every event includes mandatory fields identifying which SBC reported on which activity and when ("timestamp", "event-type" and "sbc"). The timestamp signifies the instant of time when an event was received by the ABC Monitor. This helps to overcome situations with multiple SBCs with unsynchronized clocks or different time-zone and may be slightly different than SBC-recorded time. A typical time sequence of events for a call is shown in Figure *Event of Call Timelines*: when an INVITE is arrived and PCAP file is recorded, "message-log" appears instantly. "Call-attempt" event appears as soon as a positive final answer comes back. Eventually "call-stop" appears upon receipt of a BYE from either party.

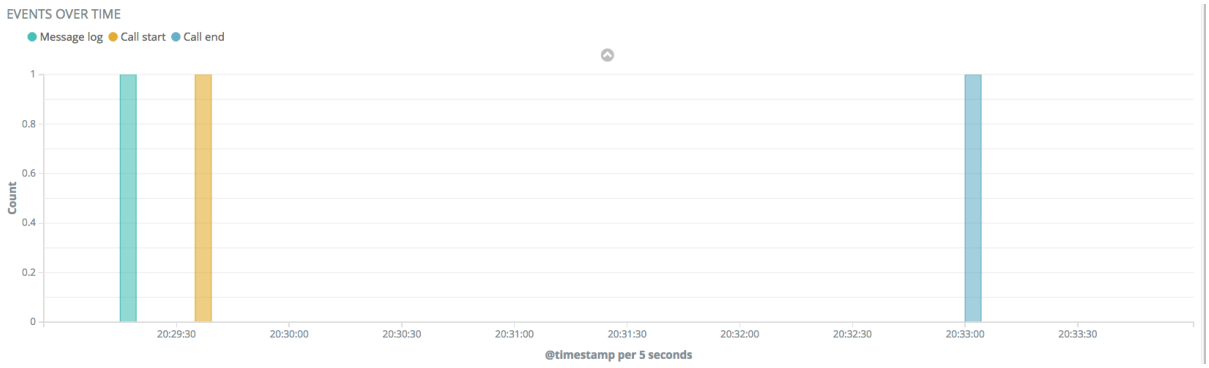

Fig. 2: Event of Call Timelines

Events relating to a SIP message include SIP addresses ("attrs.from", "attrs.to") and source IP address ("attrs.source"). Call-end events include "attrs.duration" expressing call length in seconds, whereas reg-new events show the address of a newly registered SIP device in "attrs.contact".

Last but not least, script processing variables can be passed along with the call processing events – this can be for example useful to "label" the events with "tags" assigned to a call during call processing, such as "domestic", "long-distance" or "emergency". Some visualizations in ABC Monitor specifically require that an administrator sets well-known variables, as shown in Figure *ABC Rule for Setting a Destination Country Code by Request URI Prefix* and *Setting Minute Counter Call Variable*.

The ABC Monitor also enhances the events by additional fields used for security level assessment, geographical location, QoS information, and other data used for further analysis.

The following table shows content of a call-start event that is always produced when a successful INVITE transaction sets up a call. Internal fields with a leading underscore in name not shown.

Field	Value	Comment
@timestamp	2016-03-31T12:17:02.000Z	GMT event timestamp
@version	1	internal version number
type	call-start	event type
attrs.call-id	65601f8e625e0fb6484…	SIP Call-ID of the call
attrs.dst_ca_name	pstn_gateway	name of Call Agent to whom the call is forwarded
attrs.dst_rlm_name	sipgate	name of destination realm
attrs.sbc	3e440ca4-00ee	id of the reporting SBC
attrs.src_ca_name	users	Call Agent from which the request came
attrs.src_rlm_name	public	realm from which the request came
attrs.from	sip:0000@172.27.10.114	SIP From URI
attrs.from-ua	VQM 0.4	User Agent Client type
attrs.method	INVITE	SIP Request method – always INVITE for call-start
attrs.r-uri	sip:echo@free.tel	SIP Request URI
attrs.sip-code	200	Numerical code of SIP reply always 200 for call-start
attrs.sip-reason	OK	Human-reasonable reason phrase in SIP reply
attrs.receiver_ip	192.168.0.111	SBC's IP address at which it received the INVITE
attrs.ruriip	free.tel	host part of request URI
attrs.scenario	call	this is a scripting variable chosen to be passed along with the event
attrs.source	10.0.0.10	source IP address of the request
attrs.src-port	1085	source port number of the request
attrs.to	sip:echo@free.tel	To URI as in the INVITE request
attrs.to-ua	F-PBX 2.3	Signature of the called party's UA Server
attrs.transport	udp	transport protocol used for signaling
id	483B139F-56FD153E…	internal ID. useful for correlating multiple related events

The rest of this Section is structured by the event types that the ABC SBC produces:

- *Call Processing Events* – these are events that describe SIP calls and are mostly used to observe user habits, reasons for call failures, and QoS
- *Registration Events* – these are events that describe how SIP devices register with the SIP service. The events show the reachability of the SIP users in time.
- *Diagnostics Events* – these events help to identify unusual traffic patterns, misconfiguration of the service and other irregular situations.
- *Security Events* – these events report on SIP traffic which may possibly indicate attempts to compromise security of some SIP users or the SIP service as whole

6.1.1 Call Processing Events

These events are generated automatically at different stages of the SIP call establishment process, see Fig. *SIP call processing events*.

- call-start: generated after a successful call establishment. The method is always INVITE and sip-code 200.
- call-attempt: generated after an unsuccessful attempt to establish a call due to caller canceling the call, callee declining it, or a timeout. Failed authentication attempts are reported on in separate events. The events always include the SIP code with which the call attempt was rejected.
- call-end: generated after an established call is terminated. They include a full report on how the call completed. The From and To event fields take the same values as call-start event – they signify who initiated the call (and not who initiated the call termination). The event-specific fields include:
  - The field "originator" specifies who caused the call termination and can take the following values: "caller-terminated", "callee-terminated", "call-length-terminated" (SBC terminated upon exceeding the maximum call length limit), "no-ack" (SBC terminated due to missing ACK), "rtp-timer-terminated" (SBC terminated upon RTP inactivity), "session-timer-terminated" (SBC terminated upon session timer expiration), "admin-control-terminated" (administratively terminated from GUI), "internal-disconnect" (call terminated due to a internally transferred call), "reply" (negative response received on an established dialog: 404, 408, 410, 416, 480, 482, 483, 484, 485, 502, 604), "server-shutdown" (server process terminated due to a SIGUSR1 or SIGUSR2 signal), "srtp-failure" (SRTP key negotiation failure), "internal-error" (internal error).
  - The field "duration" specifies the length of call in seconds.
  - The fields "rtp-stats-a" and "rtp-stats-b" represent the RTP statistics for the media streams on each call leg. Each call leg contain one or more media streams, each of which offer incoming and outgoing information.

The following table shows the information for the incoming media streams.

Field	Value	Comment
ssrc	413934793	incoming SSRC value
src_ip	192.168.0.155	source IP address
src_port	37454	source port
dst_ip	192.168.0.155	destination IP addres
dst_port	46920	destination port
payload	PCMU/8000	media payload
packets	52832	received packets
bytes	9087104	received bytes
last_seq_nr	54428	last received sequence number
max_delta	3	maximum delta between two packets
max_delta_seq	21397	sequence number of the packet with the maximum delta
max_burst	69	maximum number of packets per second
lost_percentage	0	lost percentage
jitter	6	jitter
dropped	0	packets dropped
seconds_since_last_received_packet	0	seconds since last received packet

The following table shows the information for the outgoing media streams.

Field	Value	Comment
ssrc	473964392	outgoing SSRC value
src_ip	192.168.0.149	source IP address
src_port	27054	source port
dst_ip	192.168.0.155	destination IP addres
dst_port	26120	destination port
payload	PCMU/8000	media payload
packets	52832	received packets
bytes	8710432	received bytes
last_seq_nr	24326	last received sequence number
lost_percentage	0	lost percentage
rtt_min	5	minimum round trip time
rtt_max	172	maximum round trip time
rtt_avg	26	average round trip time
jitter	6	jitter
seconds_since_last_sent_packet	0	seconds since last sent packet

The call-end and call-start events have the same ID which can be used for correlation. This is however more often used for correlation with other events, like recording for example, because there is no additional data in call-start beyond call-end.

From-URI	To-URI	SIP callid	Result code	Response phrase	Duration	User agent (from)	User agent (to)	AoR	Contact
sip:alice@test.com		w0PtInpA3u3xqXGLOI0C60COqPWR3Ifb		ctrl-cmd	00:00:20				
sip:alice@test.com		3zoHtsOwykozxiw2MZU5cXpBDNTTzFrl		timeout	00:01:29				
sip:bob@test.com	sip:alice@test.com	5002B22A-51B99C4900013F38-F327A700		bye	00:00:05				

Fig. 3: SIP call processing events

An important capability is passing call variables (Section Sec-Call-Var) in the call processing events. This can be used to "tag" calls with various hints, for example to discriminate domestic and international calls. In some cases, there are predefined call variables that have a specific meaning to the ABC Monitor. Such are currently **dst_cc** (Figure *ABC Rule for Setting a Destination Country Code by Request URI Prefix*) and **minute_counter** (Figure *Setting Minute Counter Call Variable*).

To turn variable passing on for call events, turn on the SEMS Global Configuration Option **Add call variables into events**. Care needs to be applied so that some possibly sensitive information is not passed along with the variables to the ABC Monitor.

The call variables appear in the events with the "attrs." prefix. If a variable name is the same a name of a mandatory event field, the call variable overrides the field value. This can be useful for example when identity of a SIP user should be presented in a different way (altered, anonymized) than seen in the SIP traffic. We however recommend to use this overriding capability with caution as it leads to the loss of the original traffic view. Example of such a rule and the resulting event is shown in Figures *A Rule for Rewriting an Event Field with a Call Variable* and *An Event with a Field Modified by a Call Variable*.

Fig. 4: A Rule for Rewriting an Event Field with a Call Variable

Fig. 5: An Event with a Field Modified by a Call Variable

6.1.2 Registration Events

The registration events are generated automatically at different stages of processing SIP REGISTER requests when register caching is enabled (see Section Sec-Register).

- reg-new. This event is produced when a SIP User Agent registers a new contact through the ABC SBC.
- reg-del. This event is generated when a SIP User Agent deregisters a previously registered contact using the RFC3261 procedures. This is typically the case when a softphone is shut-down and it unregisters gracefully. Some clients are also implemented a way that they unregister and re-register newly instead of periodically renewing one registered binding. See an example of such in the ABC Monitor snapshot in Figure *Event Timeline for a SIP Device Registering and Unregistering Periodically* – unregistered bindings are immediately followed by new registrations.

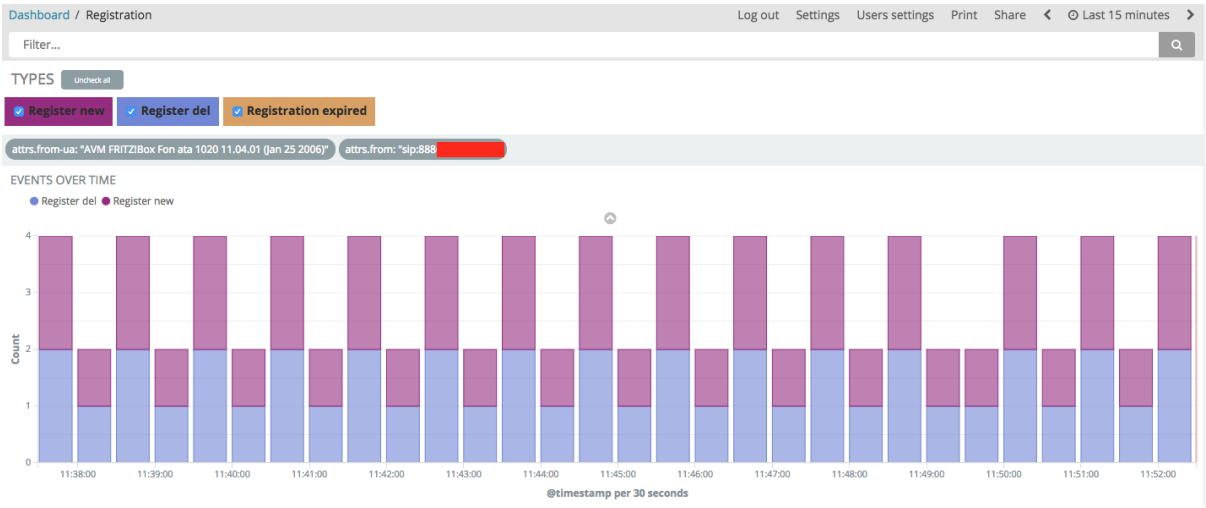

Fig. 6: Event Timeline for a SIP Device Registering and Unregistering Periodically

- reg-expired: Indicates an expired registration binding. This happens if an upstream SIP client fails to renew its contact within the window re-registration window agreed upon between the ABC SBC and downstream registrar. When this timer expires, the binding will be deleted and no incoming requests can be forwarded using the particular binding. This often happens with clients that don't comply to RFC 3261 by not respecting the server-side-imposed registration renewal

interval or vary the contacts inadequately. Example of a timeline for such a devices is shown in Figure *Event Timeline for a SIP Device Failing to Re-register Timely*. Such a devices remains unreachable in the periods of time between the orange expiration and green re-registration bars.

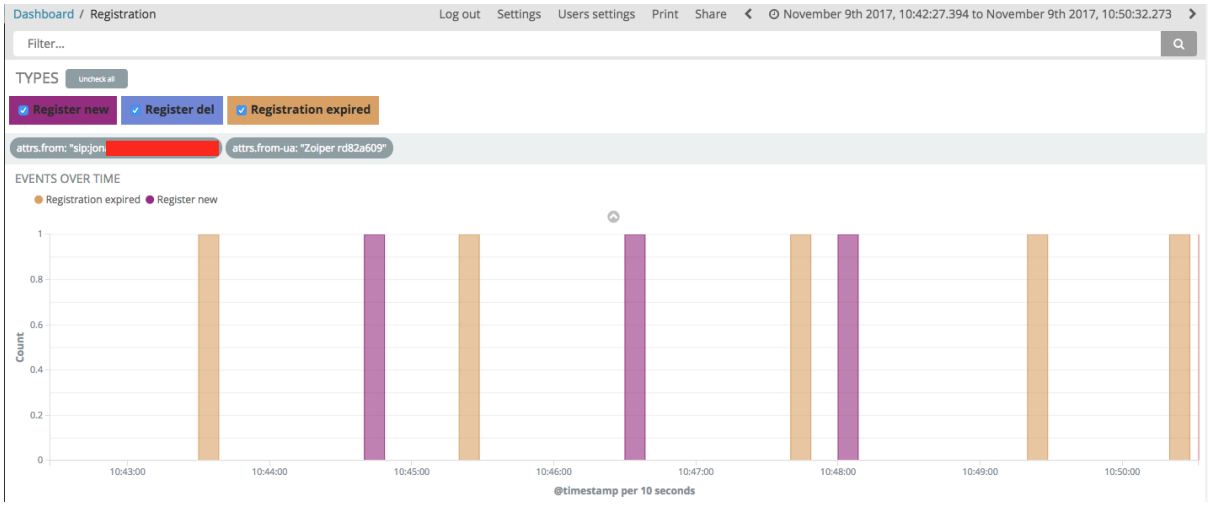

Fig. 7: Event Timeline for a SIP Device Failing to Re-register Timely

6.1.3 Diagnostics Events

The diagnostic events are used to identify conditions that provide additional diagnostics information and sometimes alert on conditions an administrator shall verify whether they are normal. These events are triggered by minor errors, completed playing of or recording of WAV/PCAP files, change in status of transport blacklisting, and custom events.

- *Custom events* (action-log): This is one of the most important diagnostic events available in the system. It is triggered from the ABC SBC rule-base using the "Log Event" rule action. Typically it is used when an administrator wants to see if a specific rule is indeed evoked and how often. Also administrators use the custom events when they begin to suspect some undesirable traffic and don't want to drop it yet. This action allows them to observe the suspicious traffic before making a further action. The conditions can be any that the rule definitions allow and often includes tests if a SIP device is registered, shows a suspicious User Agent type, tries to call a premium phone number or otherwise falls in the "suspicious category". For example an administrator may choose to observe all SIP messages that come to the ABC SBC without a username in To header field URI. Example of such a "Log event" rule is shown bellow in Figure *Rule for Reporting on SIP Requests with Empty Username in From*.

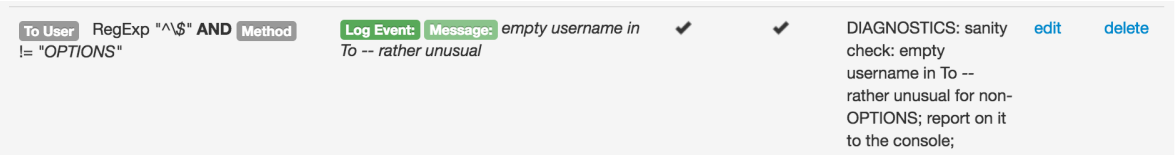

Fig. 8: Rule for Reporting on SIP Requests with Empty Username in From

The action includes a parameter where the administrator can specify additional text describing the event. The parameter can include replacement expressions providing additional information about the processed SIP message. These should be used only if necessary – when varying elements are present in event description ABC Monitor software cannot group the events by the same description.

- *recording events* (recording): These events are generated if voice recording was enabled for a particular call. The events include HTTP reference to a file with the recorded WAV file. See also Section recording.

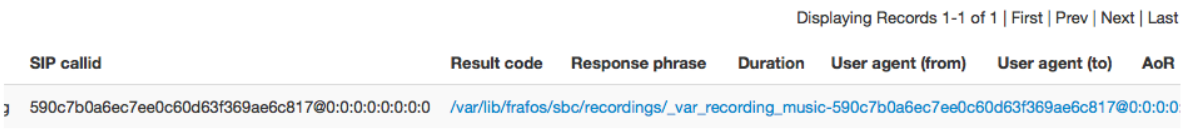

Fig. 9: Recording Event

- *SIP traffic logging events* (message-log): These events are generated only if a rule was set up to record SIP/RTP traffic using the **Log received traffic** action (see Section *Diagnostics Dashboard*). The event includes references to the recorded PCAP file and a ladder chart displaying the recorded traffic. The PCAP files appears always later than the event: the event appears when recording begins, the file when it recording ends and upload to ABC Monitor completes in background. Figure *Ladder Diagram for the Suspicious User* shows an example of such ladder-chart rendered by the ABC Monitor and showing both sides of a SIP message – incoming and outgoing.
- *Error/Alert* (error): These events are always produced when no route matches for a SIP request, a TLS connection is refused, terminated or other unspecified error occurs. The ABC SBC reports these because TLS credential management is often misconfigured and needs to be fixed for TLS clients to be able to connect. Alerts may also appear if the system is under-dimensioned and include messages like "/data disk usage above 80%" or a misconfiguration has been encountered in rule-base that was detected run-time (for example: routing failed: can't parse outbound proxy URI: 192.168.0.85).
- *Notice* (notice): These events are produced when some layer-3 or layer-4 conditions change. This is currently the case when a TLS connection opens or closes successfully, or when health status of a Call Agent changes so that it is either removed from or added to a transport blacklist. (See Section Sec-adaptive-blacklisting). *Too many SIP retransmissions event* notice events are generated if a SIP transaction reaches the defined number of retransmissions. The retransmission number triggering the event is globally configured under **"Config → Global Config → Events → Generate an event if a SIP transaction reaches .."**. The default value is 0, which disables the event notification. Notice events are also produced when traffic reaches a shaping soft-limit (see Section Sec-Traffic).
- *Prompt*: These events are always produced when a caller's attempt is handled using local audio announcements (see Section playingaudioannouncement).
- *dest_monit*: The ABC SBC reports these events when availability monitoring is enabled for a Call Agent (see Section Sec-adaptive-blacklisting). In ABC Monitor the events are visualized in the CA Availability diagram in the "Connectivity CA" dashboard as shown in Figure *Call Agent Availability Lanes*.

6.1.4 Security Events

This Section discusses events that have relevance to security of a SIP service. These security events are generated when messages are dropped because of failing to accommodate a security policy. This can be because the traffic has exceeded traffic limits, a drop action has been applied, authentication failed or an unfavorable SIP answer came from downstream in response to a SIP request.

Counter-measures to fend off security attacks are discussed in a separate Section Securing. The event types are the following:

- **limit**: These events are generated if some of the traffic constrains (see Section Sec-Traffic) has been exceeded. For example an administrator may choose to ban signaling traffic from an IP address if it sends more than 10 requests per a minute. See Figure *Limit Events* for example of limits reporting on traffic shaping in effect. The limit event type is also generated when current traffic volume exceeds limits set by the ABC SBC software license.

SIP callid	Result code	Response phrase	Duration
c76553ac-14cbc0ad-6772cee2@192.168.178.29		RTP bandwidth limit towards callee reached	00:00:00
c76553ac-14cbc0ad-6772cee2@192.168.178.29		RTP bandwidth limit towards caller reached	00:00:00
e1130136-ff37103f-5506921c@192.168.178.29		RTP bandwidth limit towards callee reached	00:00:00
e1130136-ff37103f-5506921c@192.168.178.29		RTP bandwidth limit towards caller reached	00:00:00
a8cad0100f70c974a8aaa908b07cf670@0:0:0:0:0:0:0:0		CAPS limit reached for counter 6ee06b93-06cc-84e8-fbc5-0000288d644f	00:00:00

Displaying Records 1-5 of 5 | First | Prev | Next |

Fig. 10: Limit Events

- **message-dropped**: These events are generated if a message was silently discarded using the *drop* action in A or C rules. Sender of the discarded traffic will not see any answer to his request. Note that if he is probing the service using TCP he will still be able to find out if there is running service. (Section Sec-uablacklisting provides more details on blocking traffic using the *drop* action).
- **auth-failed**: This event is triggered always when a SIP request authentication fails. Note that the way the SIP protocol works, an initial request is always challenged by server using the 401/407 replies. This initial challenge does NOT trigger the event. Only when the subsequently re-submitted request with credentials fails to authenticate and yields a 401/407 answer, the auth-failed event is generated. The reason why a request fails to authenticate may be multi-fold and needs deeper examinations. A SIP phone user may fail to configure his device with proper SIP URI and/or password. It may be administrative mistake on the server side such as deactivating a user account. However it may be also a password-guessing attack, such as shown in Figure *ABC Monitor Displaying How a Brute-Force Password Guessing Attack Ramps Up*. The sudden increase in number of authentication failure clearly indicates an attempt to breach security of the SIP service.

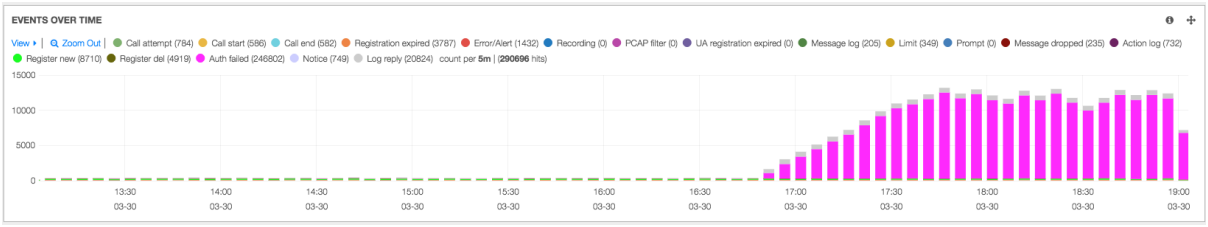

Fig. 11: ABC Monitor Displaying How a Brute-Force Password Guessing Attack Ramps Up

- **log-reply**: The log-reply event allows an administrator of the ABC SBC to identify traffic that apparently irritates downstream SIP equipment. If for example a downstream server chooses to send a 604 for requests that use non-existent SIP URIs, the ABC SBC may be configured to report on such using the *log replies* action as shown in Figure *Rule for Reporting All 604-replied Responses*. Similarly the events can be generated on receipt of any other specific reply codes, such as 403 (Forbidden).

Fig. 12: Rule for Reporting All 604-replied Responses

An example of events captured during a scanning attack is shown in Figure *Events Produced During a Scanning Attack*. In the "To" column you actually see that the attacker was trying to register under different numbers beginning with 12: 122667, 12554, 122562, etc about every two seconds. When he hit a non-existing account (12554 for example), a 604 came back and triggered a "log-reply" event. However as he

tried 12667, a 401 came back revealing to him that he had "pinged" an existing account, just without proper credentials.

Timestamp	Type	Reason	To
2016-03-31T18:36:30.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:36:29.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:36:28.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:36:20.000+02:00	log-reply	this From URI not served here	sip:12554@sip.
2016-03-31T18:35:51.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122562@si
2016-03-31T18:35:50.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122562@si
2016-03-31T18:35:50.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122562@si
2016-03-31T18:35:48.000+02:00	log-reply	this From URI not served here	sip:12563@sip.
2016-03-31T18:35:48.000+02:00	log-reply	this From URI not served here	sip:12561@sip.
2016-03-31T18:35:39.000+02:00	log-reply	this From URI not served here	sip:12556@sip.
2016-03-31T18:35:27.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:35:27.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:35:27.000+02:00	auth-failed	Authorizing REGISTER request failed	sip:122667@si
2016-03-31T18:35:19.000+02:00	log-reply	this From URI not served here	sip:12554@sip.

Fig. 13: Events Produced During a Scanning Attack

- **firewall-blacklist**: These events identify blocked IP addresses. They are are generated when an ABC SBC chooses to drop traffic from an offending IP address. See Section Sec-Abuse for configuring criteria for automated IP address blocking in an ABC SBC.
- **firewall-greylist**: These events are generated when an ABC SBC chooses to drop traffic from an IP address that sent the ABC SBC some initial traffic but has not managed to establish trust. See Section Sec-greylisting for configuring blacklisting in an ABC SBC.

6.2 HOWTO Find a Needle in the Haystack: Iterative Event Filtering

The ABC Monitor combines both aggregated view of event data as well as the actual data details. This is instrumental in finding problems quickly: the aggregated view helps to detect a trend or anomaly which would be hard to find in the vast amount of SIP data. Once a situation worth further investigation is detected, the administrator can apply different filters consecutively until the root cause is identified using event details. The details can go as low as bits of SIP messages.

In this chapter, we will show an example how to iteratively proceed from detecting a high-level problem to finding the low-level bits triggering it. The examples are taken from a real operation and therefore many of the elements in the screenshots are shaded.

For example, administrator may find in Call Dashboard that average call failure ratio over 50% is too high, see that most frequently occurring call failure reason is 480 ("User Offline" in SIP specification), and start nailing the problem down by applying event filters.

This is the situation shown in Figure *Example of a High Call Failure Rate Situation*. Average failure rate is at 68%, the most massive error code is 480.

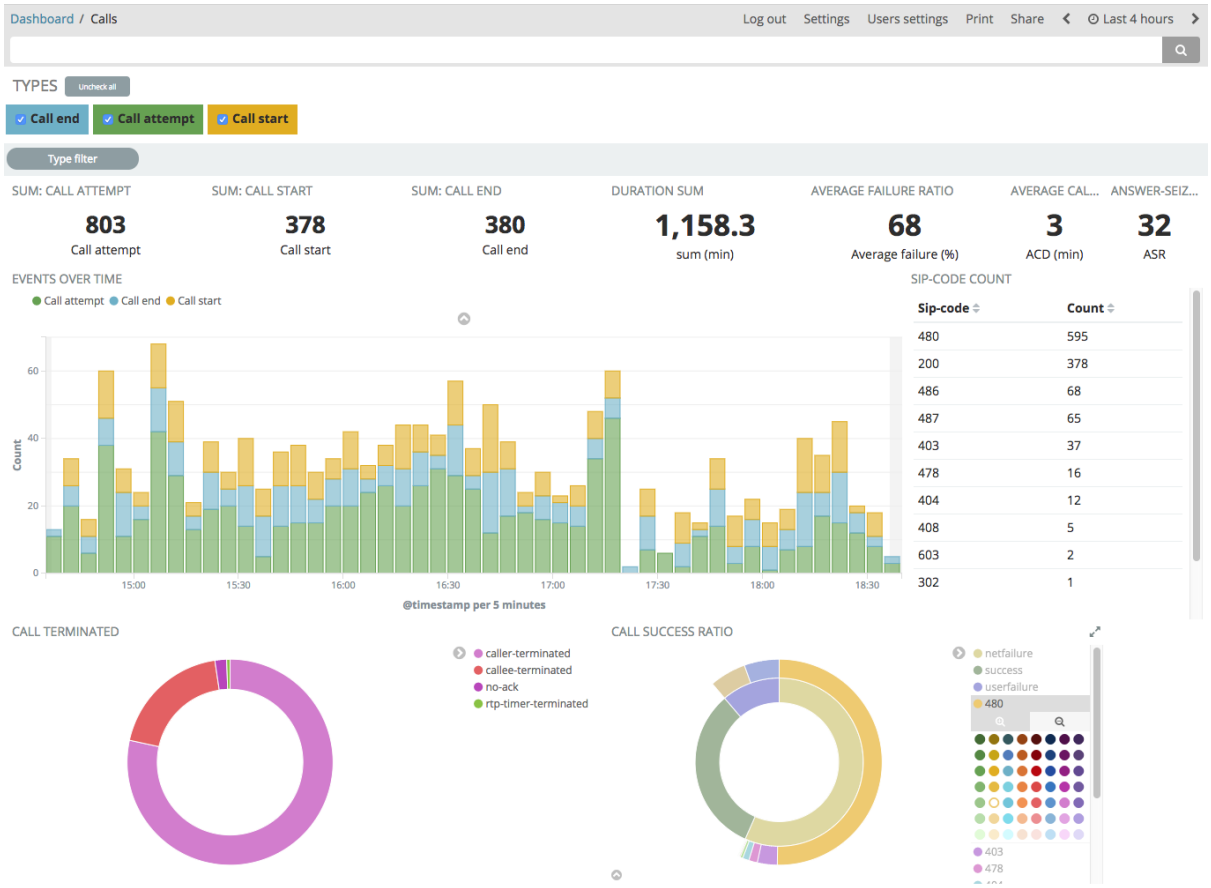

Fig. 14: Example of a High Call Failure Rate Situation

What the administrator typically does in such a situation is to spot the unusual trend, and inspect its details. Here the abnormality is the unusual number of 480s, in fact the typical top error codes are Busy (486) and Canceled (487). Therefore the administrator will limit the events to those with the SIP code of 480. He does so by clicking the "plus magnifying glass" icon underneath the 480 code, and "pinning" the filter using the pin icon in the top bar so that the filter can be transferred to a Dashboard like "TopLists". The filter looks like in Figure *Filter for Further Inspection of too Many 480s*. The statistics have changed because the filter limited inspected events only to call attempts failing with 480 code, and also because the time window has slightly advanced in the meantime.

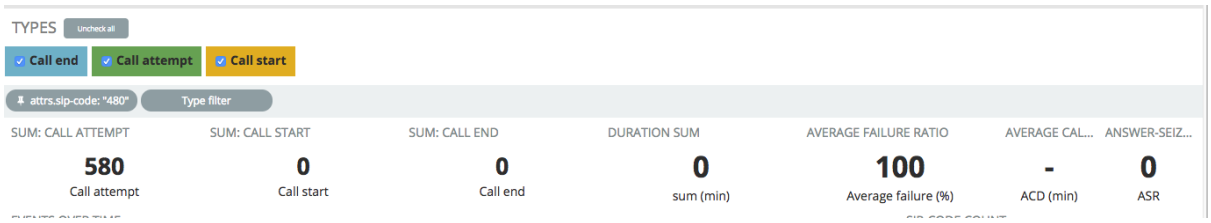

Fig. 15: Filter for Further Inspection of too Many 480s

When we now switch to the Toplist, we will find out that vast majority of the 480-terminated call attempts is coming from a single domain, and inside this domain an anonymous user is dominating. (Figure *The Top 480-er*) While we do not know, if it is a user trying to desperately reach an offline called party, or someone scanning calls, we can pin a filter by caller, deactivate the filter by error code and see the full history of the suspicious user in Overview.

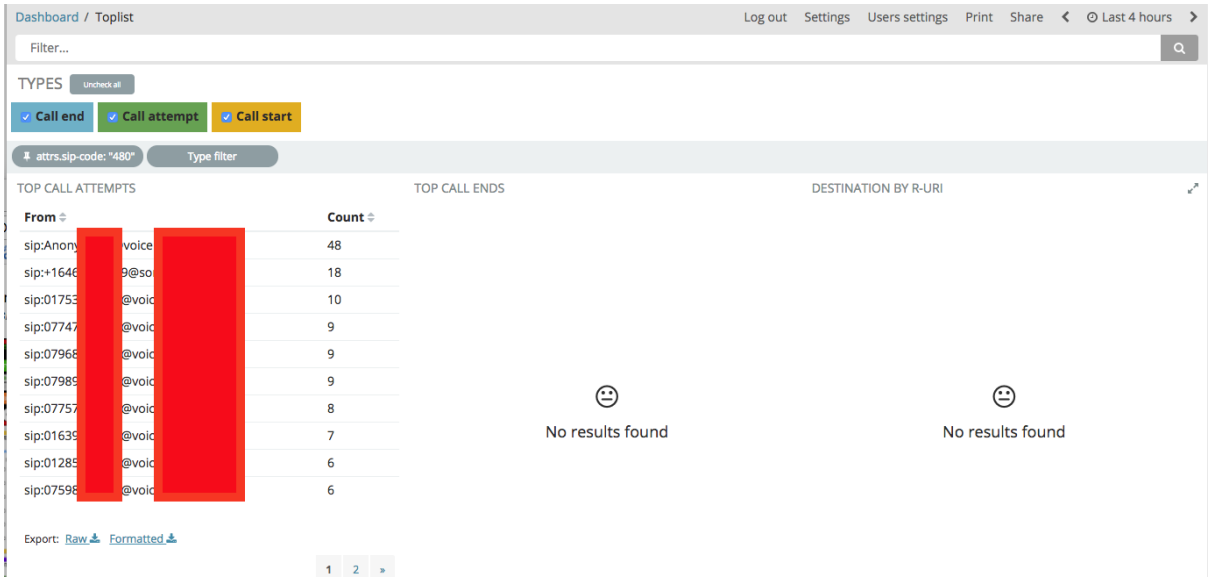

Fig. 16: The Top 480-er

The Overview gives us a picture of a suspicious user who keeps making call attempts at a high rate without success and also without attempt to register. This is seen in *Complete Event History of a Suspicious User*.

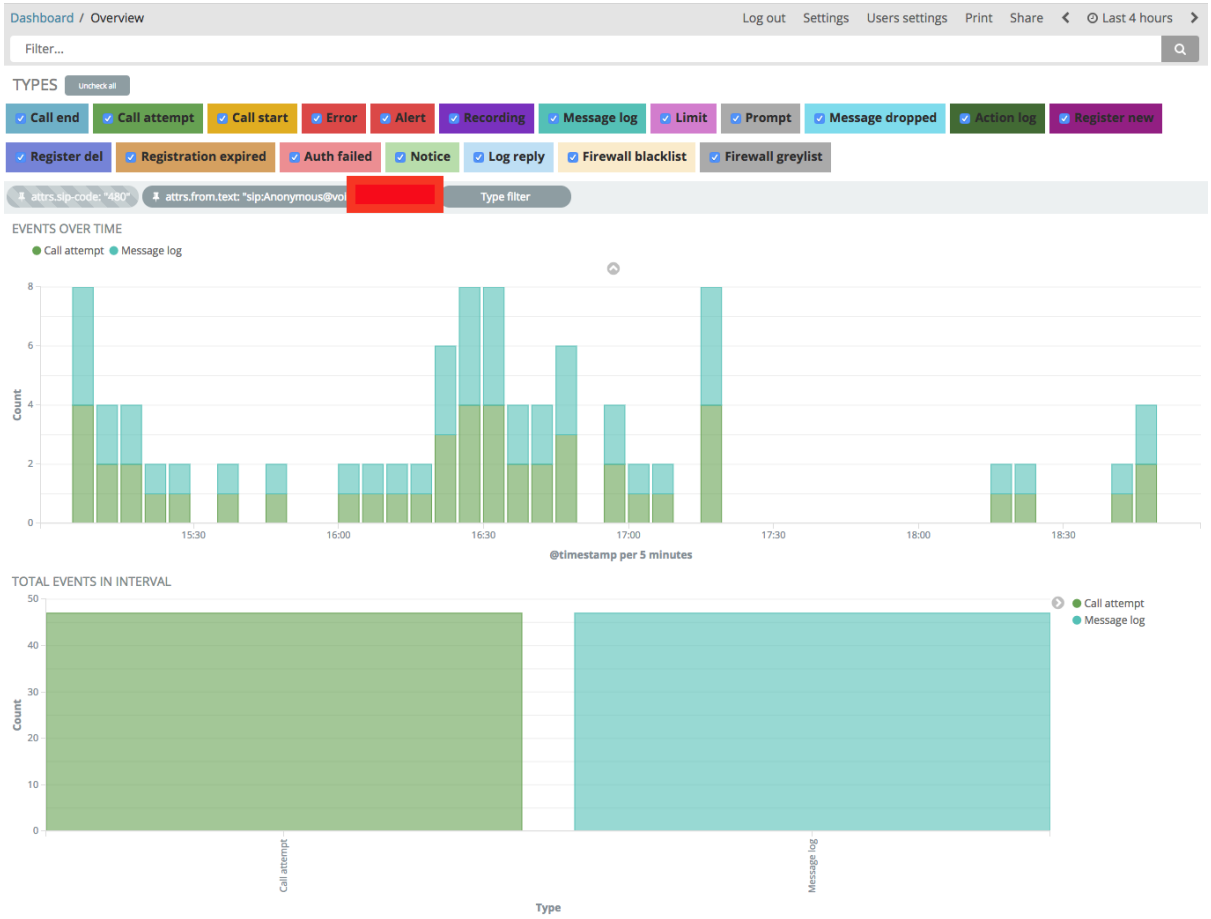

Fig. 17: Complete Event History of a Suspicious User

It pays off therefore to scroll down and look at event details for this particular user. Not only is there a detailed

report on the call attempt, but under “View Messages” there is a link to ladder diagram showing the actual SIP message exchange.

Fig. 18: Complete Event Details

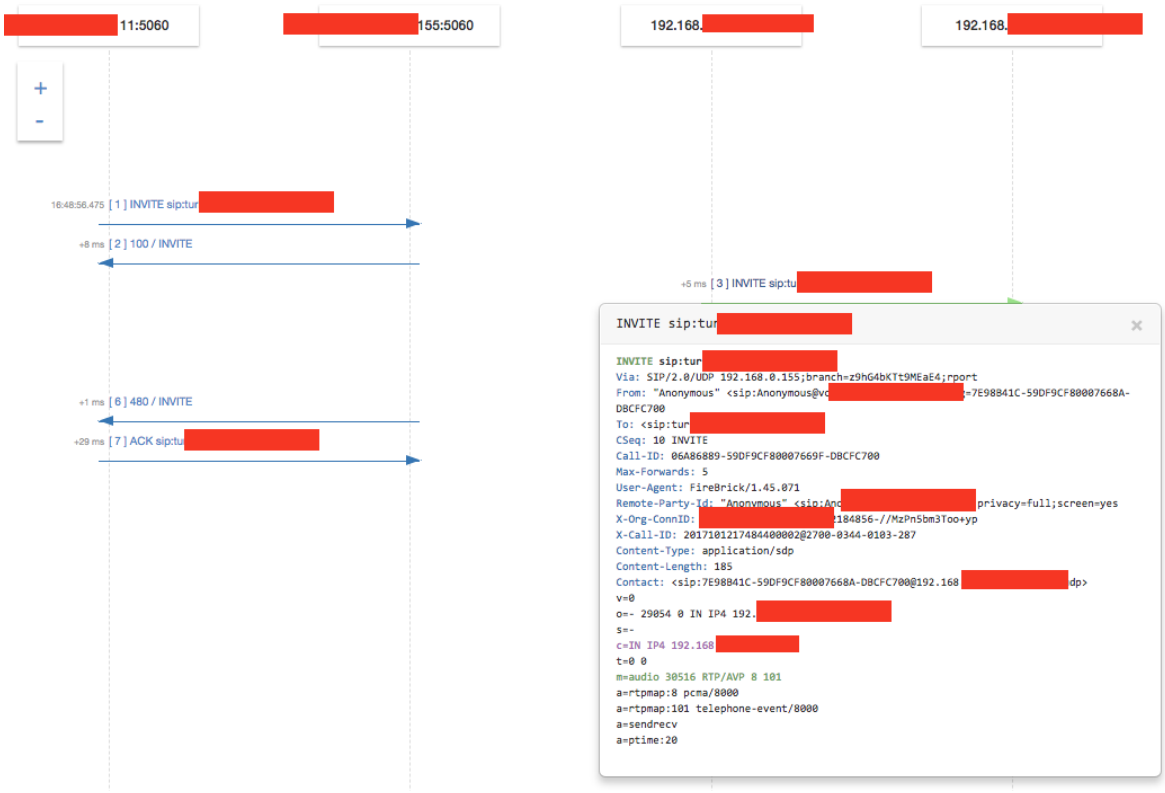

Fig. 19: Ladder Diagram for the Suspicious User

The ladder diagram shows the message flow and its timing, as well as details of SIP message, including From and To identities, type of SIP User Agent, and SDP media negotiation payload. The capability to view internal perspective of traffic is very valuable – if traffic is encrypted, it is difficult for a troubleshooter to inspect its content. Also if the SIP traffic is obfuscated by use of topology hiding (see Sec-Tolplogy), it would be difficult to relate incoming to outgoing SIP dialogs without the internal perspective.

In summary, we have shown in this example how to detect unusual situations using aggregated views (too many 480s), filter out events specific to the situation, find a user that has caused most of them and inspect in detail his

gap-free history and even SIP message details. This iterative process gives every administrator powerful tools to find out what is going on in a SIP service, and have good information to decide if he is dealing with an abnormally active user, malicious attack or a network misconfiguration.

6.3 Using Filters

As shown in the previous chapter, filters are the essential instrument for finding out what is going on. In this chapter we describe all of the filter types available in ABC Monitor. There are data filters, type filters, time filter, and full-text filters. Multiple data filters can be combined, in which case events will be sorted out that match **ALL** of them. All of the filters appear in the most upper part of the dashboards. For example, events can be restricted to all but registration events, as long as they relate to an IP address, lead to a "480" SIP failure code and are for a given SIP user, as shown in Figure *Example of a Combined Filter*.

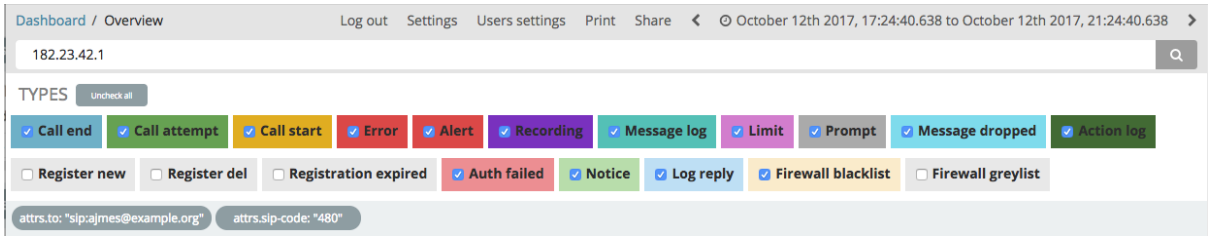

Fig. 20: Example of a Combined Filter

Data filters are used to filter out all events with the same data field values. They can be created from many elements shown in the dashboards. When user hovers over most of the data elements shown in the dashboards, two magnifying glass icons with plus and minus symbol will appear. By clicking on either icon, a filter is created that restricts events to those that either have (plus icon) or do NOT have (minus icon) the same value.

For example, one can visit the Call Dashboard, review the most frequent SIP error codes, and filter out call attempts relating only to 403 (Forbidden) as shown in Figure:

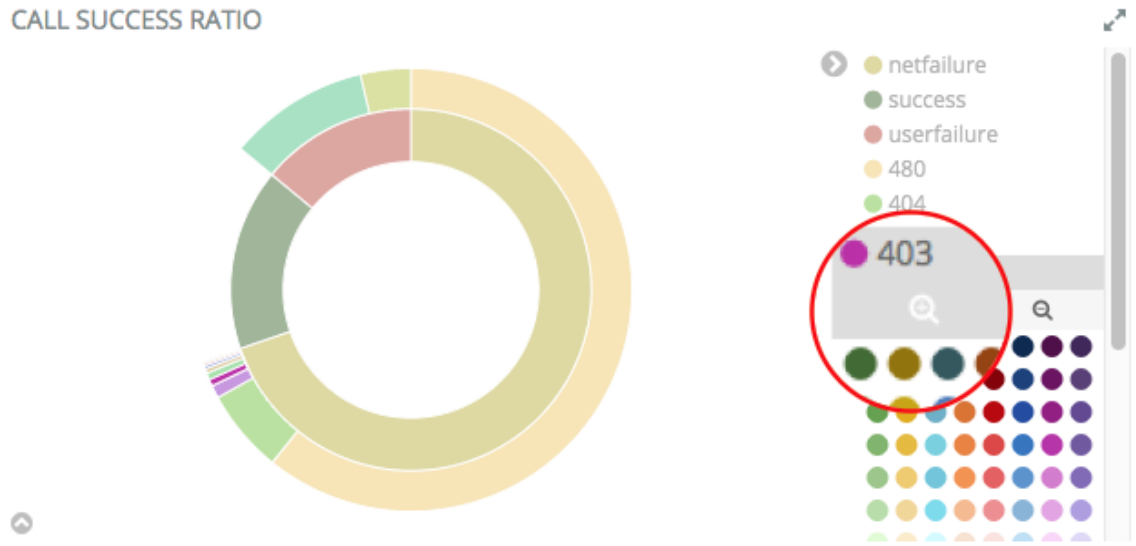

Fig. 21: ABC Monitor filtering out 403-ed call attempts in Call Dashboard

Every data filter can be deleted, deactivated, and importantly pinned – as shown in the example in the previous chapter, a pinned filter can be transferred to another dashboard where some specific aspect of the filtered data is easier to find. To pin a filter, hover over it and click on the pin icon. Unpinning is done the same way.

Other possibilities to adjust filters include temporary deactivation using the checkbox icon (filter appears then dimmed), permanent deletion using the thrash bin icon, and filter negation using magnifying glass icon (negated filters appear in red).

Fig. 22: Filter Alternations: pinned, pinned and negated, deactivated

Type filters checkboxes are shown in the top-bar and allow to easily restrict events by their respective types. This is particularly useful in dashboards with many event types, like in Overview, when administrator wishes to filter out events unrelated to his case.

Time filter sets the window of inspect time either absolutely, or relatively to current time. Of course, it can only cover the time period for which events are stored, as configured during the ABC Monitor installation.

Last but not least, the top search box allows to add **full-text filter** that looks for a pattern in multiple fields of the available events. By default, the full value, such as Call-ID must be included. Special terms can be used as follows:

- *****, asterisk, stands for a wildcard and can substitute for any number of any characters. Use of wildcards slows down the search.
- ****, backlash, means that the subsequent character will be interpreted literally
- a **colon**-separated <name>:<value> pair means that the searched value is looked for only in a field of the given name
- combinations of the terms are possible: **AND** allows to introduce multiple conditions, all of which must be met; **OR** allows to introduce multiple conditions, any of which must be met; **NOT** allows negation
- also the syntax "attrs.source:[from_ip TO to_ip] allows matching the event source IP address against an IP address **range**

Therefore if there is a user Wesley making calls using his SIP address sip:wesley@frafos.net to reach the SIP address sip:123456@example.net and he makes the calls from an IP address 192.168.0.85 belonging to the Call Agent "wesley-net", the following search expressions will match:

- **sip\:wesley@frafos.net** will match all calls from/to Wesley; note that colon must be preceded by backslash, otherwise the ABC Monitor would attempt to search through a field named sip
- ***wesley*** will match all previous records, and probably some more as well, such as wesley.home and wesley.office. It will also match all calls from and to the Call Agent "wesley-net".
- **attrs.dst_ca_name:wesley-net** will match all calls towards the Call Agent **wesley-net**
- **192.168.0.85** will match all events relating to that IP address.
- **attrs.source:[192.168.0.0 TO 192.168.0.255]** will also match because Wesley's IP address is inside the range
- **487** will match all call attempts that failed with 487 SIP code
- **attrs.sip-code:487 OR attrs.sip-code:486** will match all call attempts that failed because of 487 (canceled) or 486 (busy)

The following search expressions will not match:

- **wesley** will not match, because full-match is attempted without wildcards
- **sip:wesley@frafos.net** will not match because of the colon
- **NOT attrs.source:[192.168.0.0 TO 192.168.0.255]** will certainly match many events but not Wesley's as his IP address is in the negated IP range
- **attrs.duration:[500 TO *]** will filter out all calls exceeding 500 seconds in duration

6.4 Overview Dashboard

The Overview Dashboard displays events of all types. This is often used, when inspecting a gap-free history of a specific IP address or user identified by a URI.

In the following example, we look at traffic generated in the frafos.net domain. We let a user to register using wrong password (failed-authentication event), then retry using a correct password (register-new), make a call to an announcement (prompt, and also message-log because administrator chose to store all SIP traffic on this SBC, and action-log because administrator chose to issue a custom event for calls to a specific destination).

Fig. 23: Example: gap-free history of all events in a domain

Some other interesting chart in the Overview dashboard is that depicting total number of events by time. Especially finding a disproportionally high number of a specific event type indicates an unusual situation. For example a high number of greylisting events failures as shown in Figure *Total Events with Disproportionally High Number of Greylisted IPs* typically signifies a security attack.

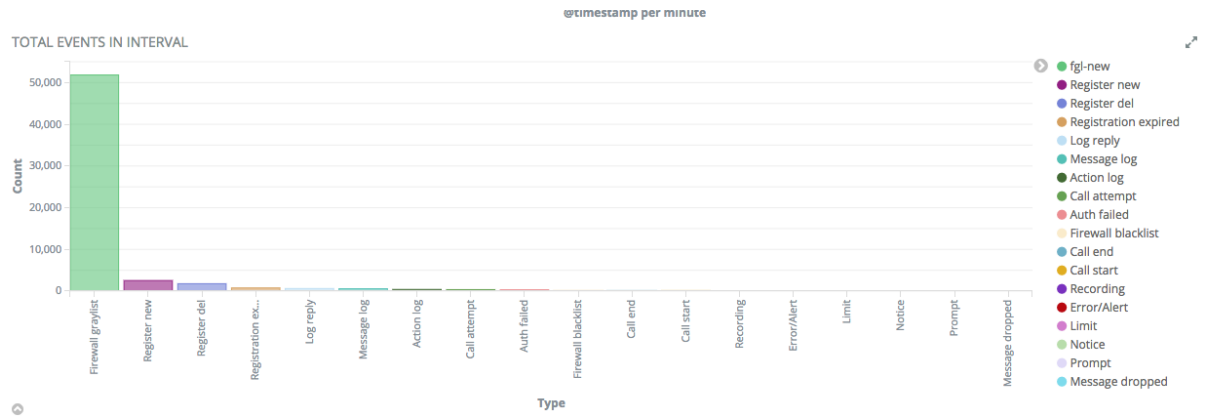

Fig. 24: Total Events with Disproportionally High Number of Greylisted IPs

6.5 Calls Dashboard

The Calls Dashboard analyzes call-related events (see Section *Call Processing Events*) to summarize processed SIP calls and how their quality was.

A screenshot of the top part of the dashboard has been already shown in Figure *Example of a High Call Failure Rate Situation*. From top to the bottom, there are call statistics, call events timeline, and breakdowns of successful calls by termination party and final status.

The pie chart breakdowns help to identify in detail why calls are being terminated. The left-hand side pie chart shows who terminated established calls. The normal termination types are "caller-terminated" and "callee-terminated". However calls could have been also terminated by the ABC SBC for a variety of reasons. These include "no-ack" when a caller failed to deliver the SIP ACK request, "rtp-timer-terminated" when RTP media stopped flowing without clean SIP session termination. See the Section *Call Processing Events* for the full list. The right-hand side pie chart shows both successfully established calls and failed call attempts and structures them by status code in the outer ring. The status codes are categorized in the inner circle intro three groups: success (200-answered INVITEs), userfailure (486 Busy and 487 Canceled) and network failure (everything else). Clicking on a pie chart segment allows to introduce a filter for the events of the same kind.

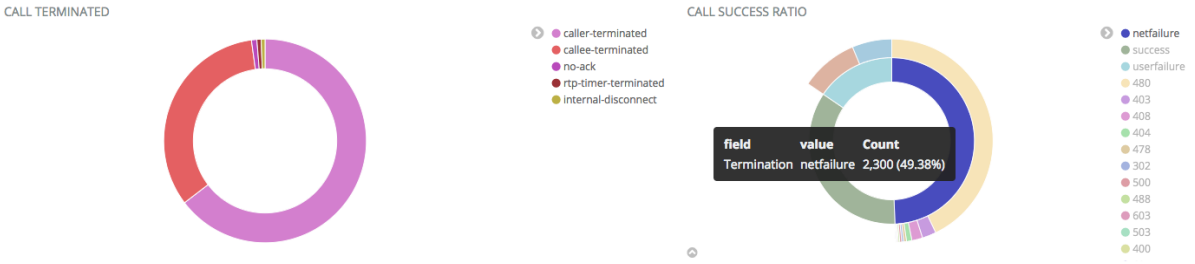

Fig. 25: Call Completion Status Breakdown

The lower dashboard part is shown in Figure *Screenshot: Lower Part of the Call Dashboard* and includes call durations, break-down of calls by countries, and eventually quality details of QoS-troubles calls and the call event details.

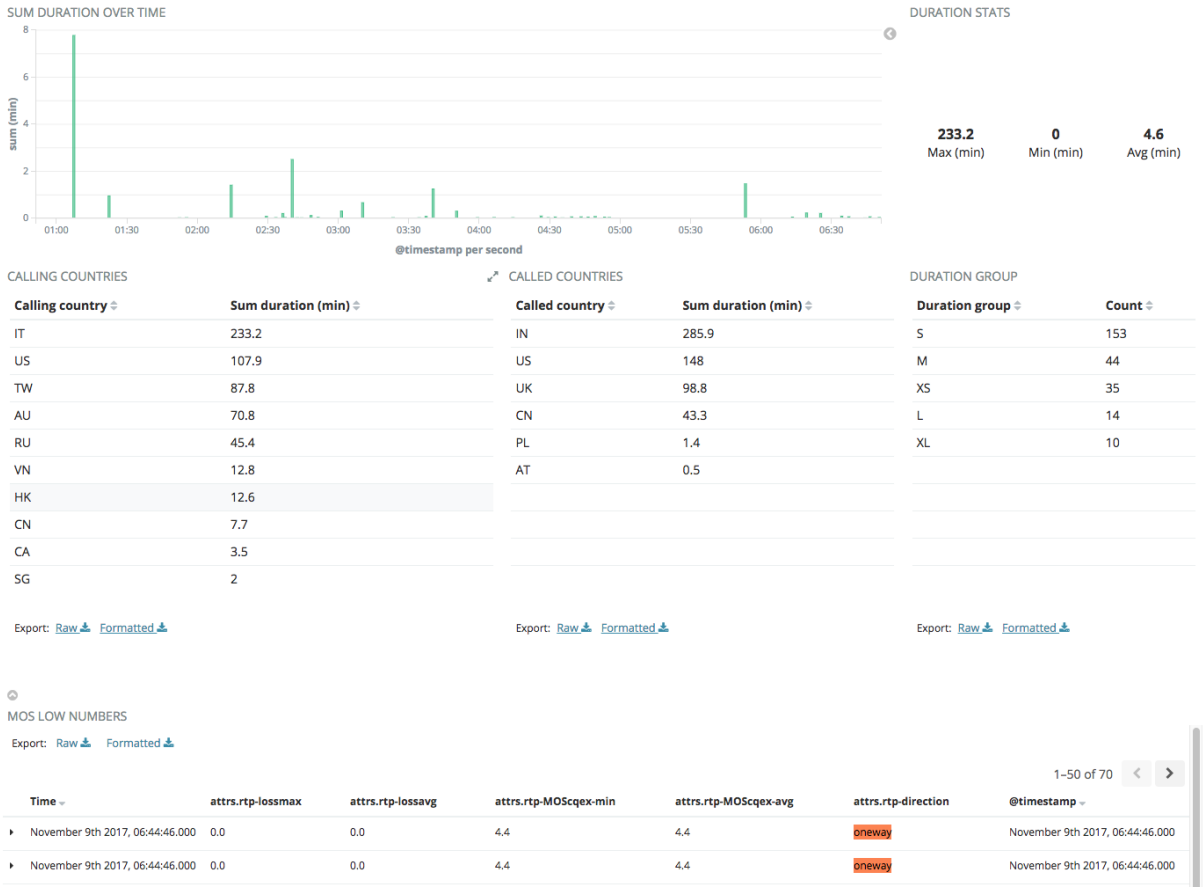

Fig. 26: Screenshot: Lower Part of the Call Dashboard

Note that break-down of calls is calculated differently for source and destination. The source country is determined using the request source address, whereas the destination country is determined from request-URI using knowledge of the SIP service's dialing plan. To accomplish the latter an administrator must tag the calls by a country tag in the ABC SBC. To do so, he must have knowledge of used dialing plans and set the call variable "dst_cc" in ABC SBC rules to proper country codes using the "Set Call Variable" Action, see Figure *ABC Rule for Setting a Destination Country Code by Request URI Prefix*.

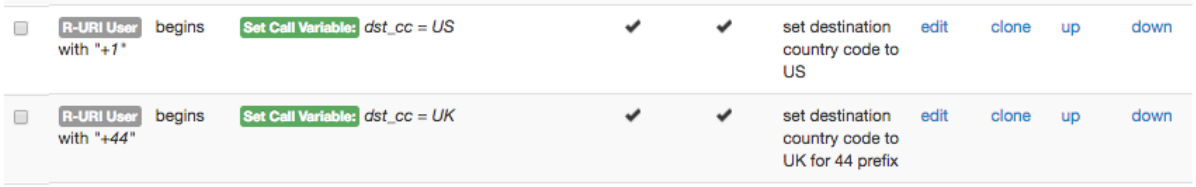

Fig. 27: ABC Rule for Setting a Destination Country Code by Request URI Prefix

The next section highlights calls with suboptimal VoIP quality. Especially of importance are calls with the attribute "attrs.rtp-direction" set to "oneway". That means that for that call, media has been received only in one direction. This is an irritating VoIP phenomena which typically occurs when there are NAT connectivity problems for the affected user.

Last but not least in this dashboard, there is the list of call details. If PCAP and/or WAV recording was enabled for the respective calls in ABC SBC rules, the files or ladder diagrams (see example in Figure *Ladder Diagram for the Suspicious User*) can be downloaded from the unfolded event details. QoS reports are included in the call-stop events in JSON format. The reports include two parts, one for the media streams from and to the caller, and another one for the streams from and to the called party. The values have the following meaning:

- max_delta stands for the maximum interarrival packet gap of received packets. Values above 120 ms for audio are already high and indicate a gap which could have occurred due to muting or voice inactivity detection without marking it as such.
- loss percentage shows relative number of packets lost[2]. Values above 1% show lossy networks, values bellow 5% can be often tolerated by listeners.
- jitter[3] shows variation in packet transit delay. High value above 120 ms typically indicates network congestion and results in dropping of late-arrived packets.

The following example shows such a QoS report. The first bracket pair encloses records about packet streams from caller as seen by ABC SBC and to caller as reported by caller's RTCP reports. The second bracket pair reports quality on streams from and to callee both of them with perfect QoS:


```
[
  {
    "dir": "in",
    "ssrc": "1864183198",
    "src_ip": "192.168.0.155",
    "src_port": "20518",
    "dst_ip": "192.168.0.155",
    "dst_port": "17342",
    "payload": "PCMU/8000",
    "packets": "15670",
    "expected": "16204",
    "bytes": "2695240",
    "last_seq_nr": "42433",
    "max_delta": "14851",
    "max_delta_seq": "41514",
    "gaps": "8",
    "lost_percentage": "3.295482596889657",
    "jitter": "13",
    "dropped": "0",
    "seconds_since_last_received_packet": "0",
    "MOScqex": "3.420"
  },
  {
    "dir": "out",
    "ssrc": "1618416588",
    "src_ip": "192.168.0.155",
    "src_port": "17342",
    "dst_ip": "192.168.0.155",
    "dst_port": "20518",
    "packets": "16271",
    "bytes": "2798612",
    "last_seq_nr": "16280",
    "lost_percentage": "6",
    "jitter": "39",
    "rtt_min": "114",
    "rtt_max": "1150",
    "rtt_avg": "164",
    "seconds_since_last_sent_packet": "0"
  }
]
```


[2] For in-depth discussion of packet loss we recommend the following article: http://www.voiptroubleshooter.com/problems/packetloss.html

[3] For in-depth discussion of jitter and its sources we recommend the following article: http://www.voiptroubleshooter.com/indepth/jittersources.html

6.6 Registration Dashboard

The registration dashboard helps to figure out if registration works for SIP users and also identify where they are coming from by analyzing registration events (see Section *Registration Events*). Events reporting on expired registrations are of particular concern because often they mean a user cannot be reached by signaling messages. Most often this is caused by broken home routers, corporate firewalls with a too strict policy, or imperfect SIP client implementations that ignore some important nuances of the SIP RFC3261 contact registration handshake.

The dashboard is structured in several parts shown in Figure *Registration Dashboard*. Bellow the statistic is also a list of the actual registration details (not shown in the Figure).

The first row shows a timeline of the registration events. Our screenshots shows a usual situation in which in every time bucket the number of new registration is about the same as number of deleted and expired registrations. Unusual situations that can be captured here are connectivity outages demonstrated by an increase in expired registrations. Note that SIP devices that register properly and keep re-registering do not produce events as they cause neither a new registration, nor a deleted/expired one.

The second row shows a geographic map of registration events. This gives a rough idea how the users are distributed in the world, even though it is not perfect. That's because the map really shows the events. As mentioned in previous paragraph, not every registered user must be producing registrations events in the examined period of time.

The next row shows use of transport protocols as reported in the registration events, these may include UDP, TCP, TLS and Websockets.

Finally we see the breakdown of SIP User-Agents, here FritzBox being the device producing more registration events, and user accounts that expire most often – probably as result of some NAT traversal difficulties.

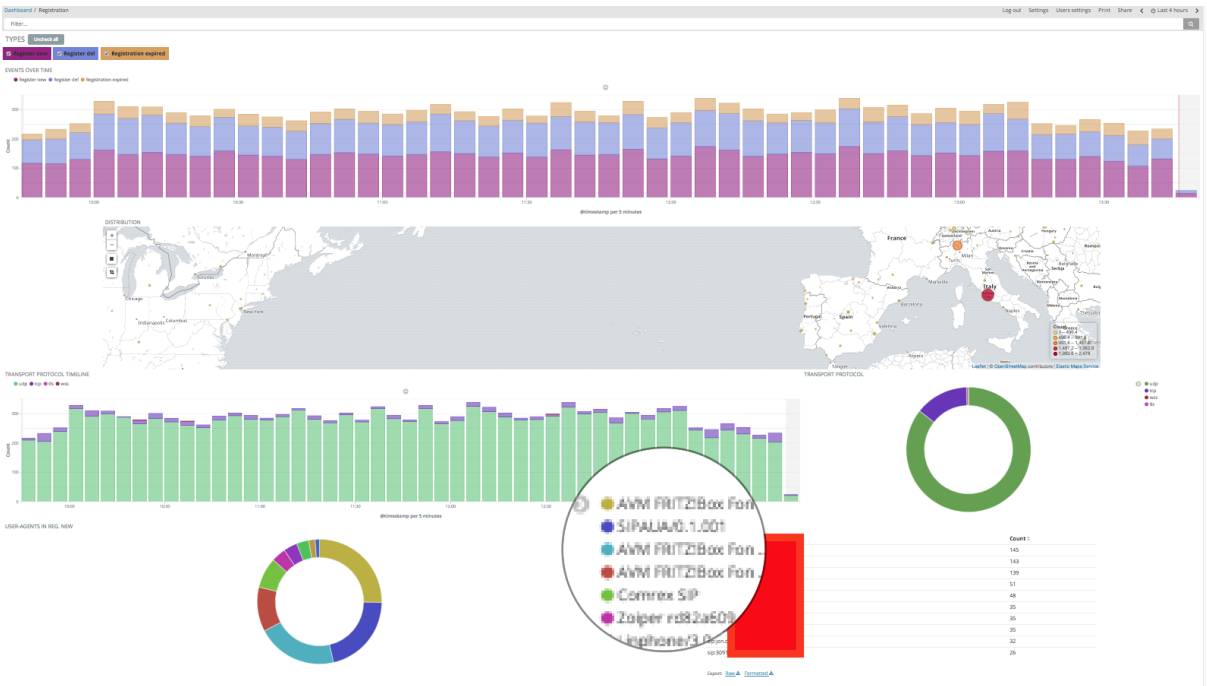

Fig. 28: Registration Dashboard

6.7 Connectivity CA Dashboard

This dashboard that came with 4.0 release focuses on topology and visualizes statistics for calls between Call Agents. This helps to discover situations such as a destination Call Agent failing abnormally often to complete calls, or SIP compatibility issues on a link from one CA to another. The numbers visualized in this dashboard refer to the currently chosen time window, as is the case with all other dashboards. The screenshots shown here visualize a situation with five call-agents.

There are two graphs in the top row that provide a quick glance at the situation. The first chart is a directed cyclic relationship graph showing how events, typically call-start, call-attempt and call-end, flow between Call Agents. The thicker the vertices, the more events were generated for traffic on this route. Also the nodes are colored from green to red – the closer to the red side of spectrum the more events refer to traffic from or to the respective node.

The table on the right-hand side shows statistics for calls by destination call agent. This table allows to quickly find out signaling performance of the CA.

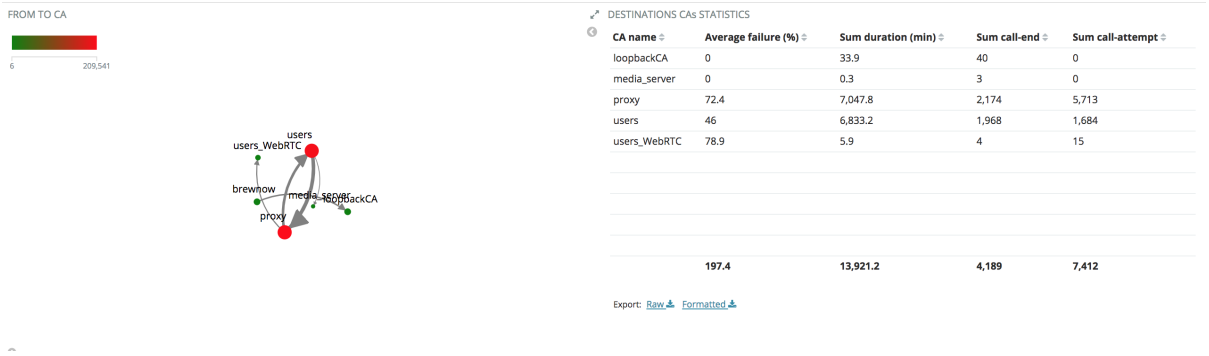

Fig. 29: Top Part of the Connectivity CA Dashboard

In the next rows there are several CAxCA matrixes that visualize the following characteristics of calls between source CA (Y axis) and destination CA (X axis):

- number of call attempts
- connection failure ratio, i.e. number of call attempts divided by sum of call-attempts and call-starts
- duration of calls between CAs
- number of completed calls

Darker colors represent higher numbers, hovering with a mouse over a field shows the actual numbers. In the example screenshot, the darkest failure ratio of 78.9% is shown for the pair proxy->users_WebRTC.

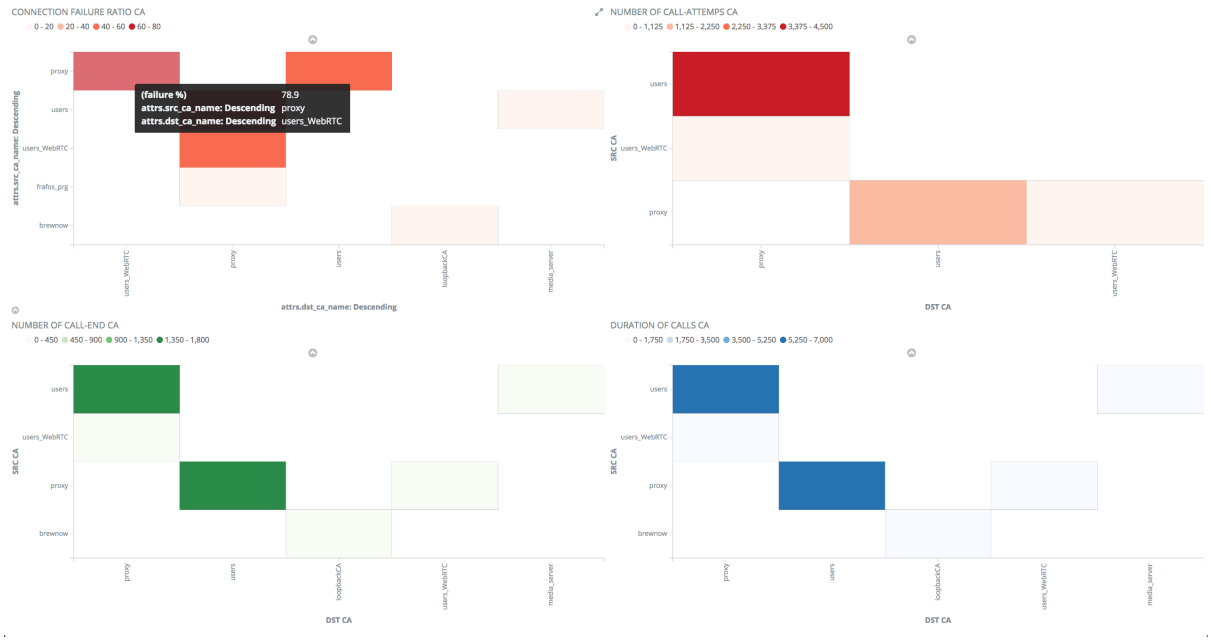

Fig. 30: Bottom Part of the Connectivity CA Dashboard

The CA Connectivity Dashboard can be used for traffic analysis the same way as laid out in the Section *HOWTO Find a Needle in the Haystack: Iterative Event Filtering*. Administrator starts by finding out some aggregated value which appears worth investigating. It can be for example an unusually high failure rate for a SIP connection from Call Agent "proxy" to Call Agent "users_webRTC" as shown in Figure *A CA-CA connection Matrix with High Failure Rate*. This connection shows 78.9% failure rate. It pays off therefore to investigate it in detail.

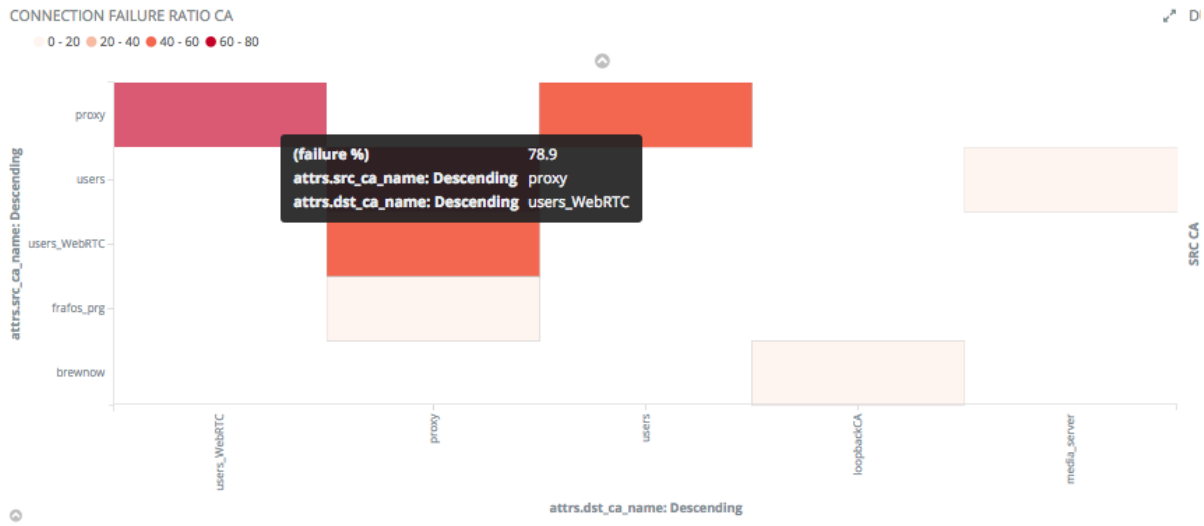

Fig. 31: A CA-CA connection Matrix with High Failure Rate

Narrowing events down to those concerning this connection is as easy as clicking on that particular field in connection matrix and confirm the resulting filter as shown in Figure *Applying a CA-CA filter*.

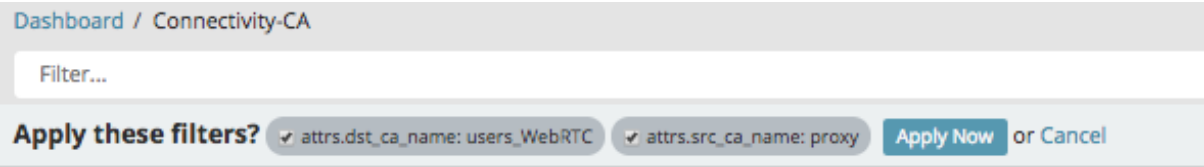

Fig. 32: Applying a CA-CA filter

After applying and pinning the filters, one can switch to the Call Dashboard and inspect the failures for this particular connection in details. Here, one can find that 500 SIP responses dominate and inspect the details of the respective events.

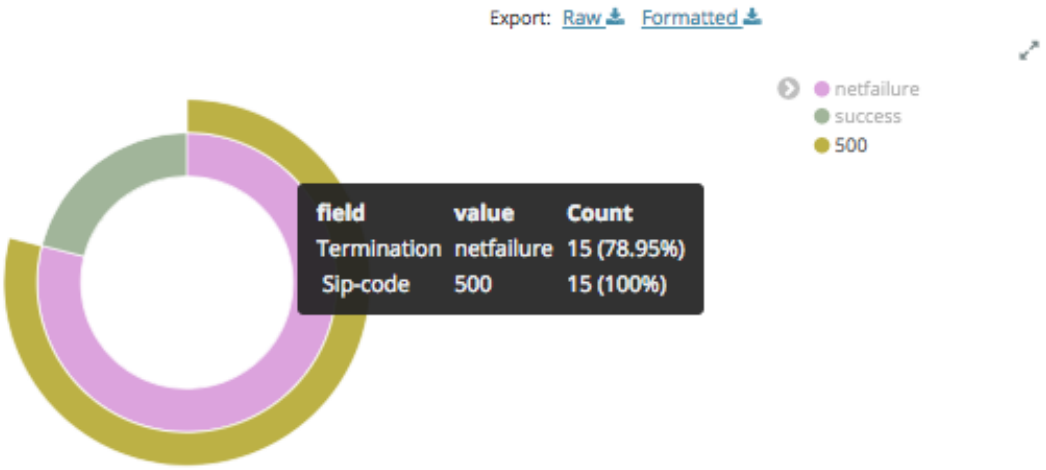

Fig. 33: Finding out the Root Cause of High CA-CA Failure rate

The bottom-most Connectivity Dashboard lane shows availability of the monitored Call Agents. Only Call Agents for which monitoring has been enabled are shown (See Section Sec-adaptive-blacklisting). The 0 status represents a Call Agent that is reachable, all other values represent some kind of connectivity issues (unreachable, DNS-unresolvable, overloaded or returning a negative response).

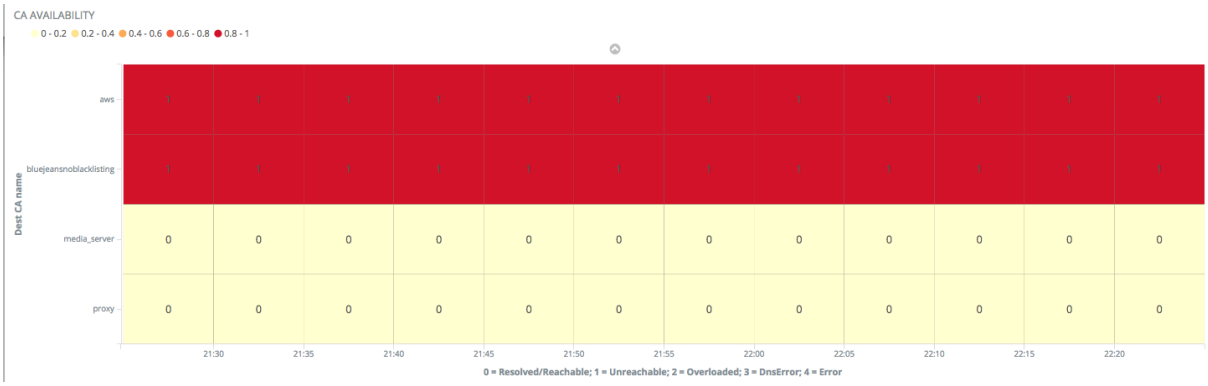

Fig. 34: Call Agent Availability Lanes

6.8 Security Dashboard

Security Dashboard is perhaps one of the most important ones as it tracks events relating to security as explained in Section *Security Events* and attempts to answer the question where the attacks are coming from. Occurrence of such events may indicate an attack that can compromise security of a SIP user or of the whole service. A detailed debate of security techniques recommended to fortify a SIP service against attackers is provided in Section Securing.

The security dashboard comes with three important charts in the Toplist section: the most frequent offenders by originating IP address, /24 netmask and geographic region.

Unless an attacker is mounting a sophisticated distributed attack, the top-list shows which IP address is causing the most of offending traffic. It is as easy as a single click of button to limit all events to those caused by the offending IP address, inspect these and undertake some appropriate security measures, blocking the IP address typically. Even if some more sophisticated attackers can send small batches of traffic from multiple IP addresses in the same subnet – they will appear on the /24 subnet toplist.

The geographic map is also very important from the security point of view. Even though most attackers don't avail of many IP addresses, they sometimes do use more than one subnet to stay under radar screen. As long as they do not use VPNs, these can be tied down by their geographic region.

An example of a situation in a public SIP service is shown in Figure *Example of the Security Dashboard*. It shows the most active IP addresses violating the SIP site's policies, and also their break-down by subnet and country, China being the most active source of offending traffic.

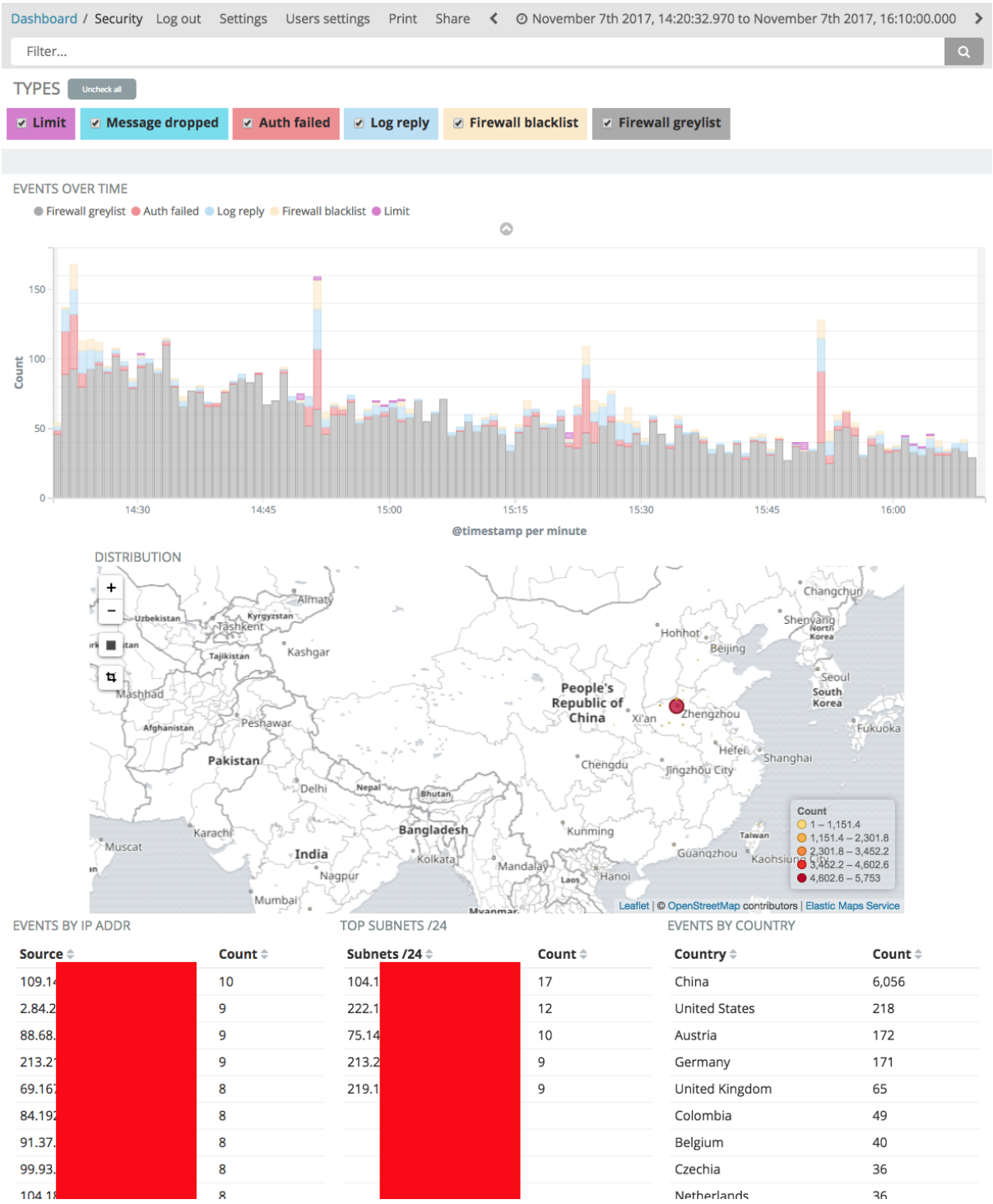

Fig. 35: Example of the Security Dashboard

6.9 Exceeded Limits Dashboards

Trying to find some unusual patterns may be sometimes a repetitive task. Therefore it is possible to raise alerts when some abnormal conditions repeat too often. The ABC Monitor allows to configure such alerts under "Settings" and inspect the alerts in the Exceeded Limits Dashboard. There are several types of the alerts, that are described in subsequent subsections.

The dashboard is only of advisory nature: it highlights excessive traffic but does not take a further action. Administrator must act if he chooses to. The following example in Figure *Exceeded Limits Dashboard* shows various such alerts as they occur over time. The donut chart breaks down the number of alerts by their type, the most offending URIs are shown in the top-chart on the right-hand side.

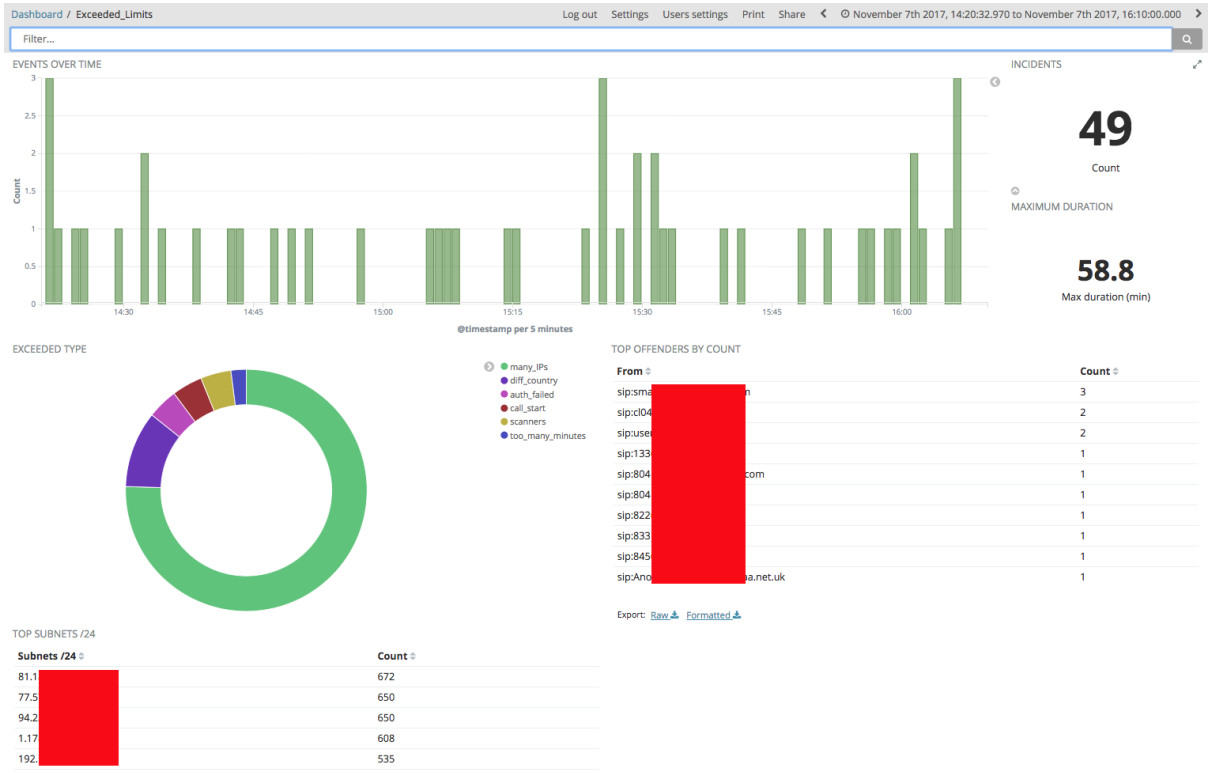

Fig. 36: Exceeded Limits Dashboard

Threshold for the respective alert types can be configured from the **Settings** Menu. The actual values can vary from site to site as what is legitimate for a SIP service may be alarming for another service. For example, in a domestic VoIP service reporting a change of user's country may be worth inspecting, whereas the same report is usual in a global public SIP service.

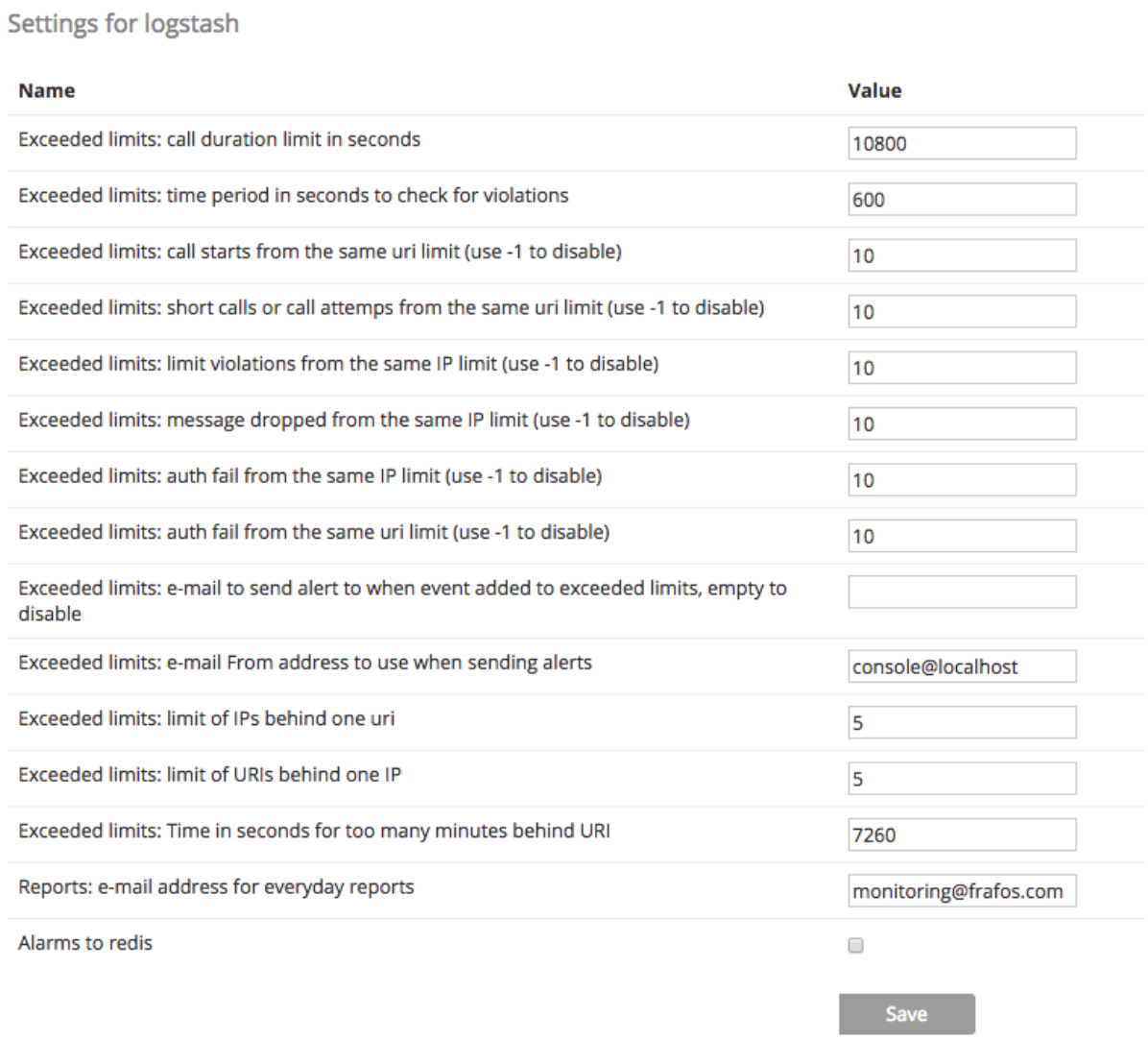

Fig. 37: Alert Threshold Settings

6.9.1 Maximum Call Duration (max_duration)

This alert is raised when a call is completed that exceeded a maximum call length threshold. The threshold value is configurable under “Settings”. Default value is 10800 seconds (3 hours).

6.9.2 Too Frequent Call Attempts from a URI (call_start)

This alert is raised when a user identified by his URI makes too many call attempts. This can be caused for example by a SIP scanner. The number of attempts and the time-span are configurable under settings and default to 10 attempts for previous 10 minutes.

6.9.3 Too Frequent Call Attempts or Short Calls from a URI (scanners)

This is similar to the previous alert except very short calls bellow 0.5 seconds count towards the limit as well.

6.9.4 Repeated Traffic Shaping Violations from an IP (limit)

This alert is raised when too many limit events originate from a single IP address over a period of time. By default, 10 such limit event occurrences over past 10 minutes will raise the alert.

6.9.5 Repeated Drop for an IP Address (message-drop)

This alert is raised when the rule action **drop** in an SBC drops an incoming SIP request from an IP address too often, by default 10 times in the past 10 minutes.

6.9.6 Too Many Authentication Failures from an IP Address (auth_failed)

This alert is raised when an authentication fails too many times from a single IP address. By default, 10 attempts in past 10 minutes from the same IP address will raise the alert.

6.9.7 Too Many Authentication Failures from a URI (auth_failed)

This alert is raised when an authentication fails too many times from a single URI. By default, 10 attempts in past 10 minutes from the same IP address will raise the alert.

6.9.8 Rapid Growth in Number of Security Events (security_metrics)

This alert is raised when the number of security events (drop, limit, auth-failed, log-reply) begins to grow too quickly.

6.9.9 A URI Active from behind too many IP addresses (many_IPs)

This alert allows to detect situations in which a user as identified by his From URI is spotted at too many IP addresses. This may be caused by both legitimate and illegitimate behavior. Sometimes users like to be reachable under the same URI at multiple destinations (office, home, second-home) or multiple call agents may be registered under the same call center's SIP AoR. However it may be also a case of identity theft. The alert includes number of IPs found, and the actual IP addresses if there are fewer than five of them. The alert doesn't repeat until next day.

6.9.10 Too Many Users behind a single IP Address (many_URIs)

This alert is triggered when events from too many users appear coming from a single IP address. This may be often legitimate when there are multiple users behind a home NAT, carrier NAT, or a PBX. The URI count is shown in the alert (countURI field) and so are the actual URIs if there are fewer than five of them (URIs field). The alert doesn't repeat until next day.

6.9.11 Changed Country Alert (diff_country)

This alert is raised when a new registration, call attempt or call with the same From URI comes from a different country than previously in the past 24 hours. The alert may identify both legitimate cases (users or call-centers with presence in multiple countries) as well as identity theft. The field **firstCountry** shows the country that was encountered previously, **geoip.country_name** show the current event's country name.

6.9.12 Too Many Minutes from a User (too_many_minutes)

This alert is raised for tagged calls when a user identified by his From URI address makes too many call minutes in the observed period of time. By default 7260 minutes in the past two hours will trigger the alert. The field "durationSum" shows the offending number of seconds.

To tag calls to count against the many-minutes alert, set the call variable **minute_counter** to the value **enabled**. This can be particularly useful in topologies when a call on a way to and from a PBX passes the SBC more than once, see Figure *Setting Minute Counter Call Variable*.

Fig. 38: Setting Minute Counter Call Variable

6.9.13 Underperforming Destination Call Agent (poor_failure_ratio_ca)

This alert is raised when the number of call attempts is relatively high to the number of successful calls. This may often be the case when a destination Call Agent begins to be overloaded. The alert is raised when the failure ratio exceeds 90%.

6.10 System Dashboard

The system dashboard shows utilization of the ABC SBC Linux operating system: system load, memory and CPU.

The example screenshot shown in Figure *System Dashboard* shows utilization of an SBC. The situation here is normal as all the values keep oscillating within a fixed range.

Fig. 39: System Dashboard

6.11 Network and Statistics Dashboard

The network statistics dashboard shows amount of traffic processed by all of the managed SBCs, both at high-level (number of calls and registrations) and low-level (number of bytes and packets). It also shows statistics of automated blacklisting.

The example in the Figure *Network Statistics Dashboard Capturing a Failover Situation* shows a typical situation on a public SIP service. The number of registrations remains fixed over time at about 3 thousands. Parallel calls peak at 8 PM, and a moderate number of auto-blacklisted IP addressed reaches slightly above one hundred. The number of greylisted IP addresses is quite high though: at 70,000 and constantly increasing. That's a clear evidence that the public SIP service is continuously subject to SIP scanning. The number of IP addresses that have passed greylisting is slightly higher than number of registrations: obviously some registrations and re-registrations occur in about same quantity, leaving the number of current registrations constant, and increasing number of IP addresses that have been accepted over time.

Fig. 40: Network Statistics Dashboard Capturing a Failover Situation

Particularly the number of calls and registrations is important – a dip often almost always indicates some abnormal network conditions. For example, if the SIP services loses its IP connectivity, SIP re-registrations will not reach it and subsequently the number of current registrations sinks down.

6.12 Diagnostics Dashboard

The diagnostics dashboard collects details that help with troubleshooting of low-level problems. Such may include SIP and SDP interoperability problems, or QoS problems. The dashboard shows events described in Section *Diagnostics Events*. ABC Monitor visualizes these events in several dashboards: Diagnostics, Transport and Connectivity CA.

Most of these events appear only when activated by administrator in the ABC SBC rules. The key diagnostics feature the ability to store PCAP files of SIP/RTP and WAV audio files. Being able to retrospectively inspect these allow administrators to find a problem which appears only transiently and is hard to reproduce.

In order for these files to appear in the dashboard, the ABC SBC must be configured to produce them. Once configured for selected calls, as soon as they complete, ABC SBC uploads the resulting WAV and PCAP files to ABC Monitor. Eventually administrator can download them from the diagnostics dashboard by clicking on the respective event details. There is also a possibility to see the SIP traffic in from of a ladder chart as shown in Figure *Ladder Diagram for the Suspicious User*.

It is worth noting that this ABC SBC capability to report on the traffic as seen "from inside" is superior to the capability of external snooping-based monitoring equipment. The "insider view" allows to analyze such SIP traffic even when it is encrypted when on the net, or obfuscated using Topology Hiding (see Section Sec-Tolplogy.)

To activate recoding of the SIP traffic one must use the action “Log received traffic”. When this action is called for a SIP call, the SIP signaling is recorded in PCAP file, optionally including RTP traffic.

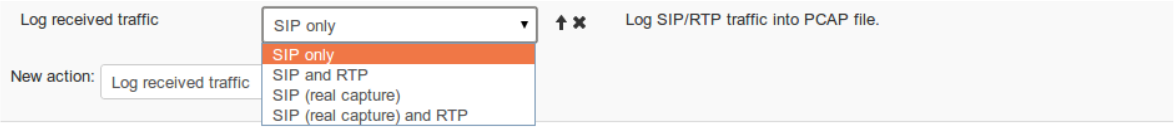

Fig. 41: Configuring traffic capturing

When recording completes, a “message-log” event is produced that includes a references to the stored PCAP file.

The parameter “PCAP file name” allows administrators to define their own filename for the PCAP files. Using Replacement expressions (see Sec-Replacement) one can include SIP message elements in the filename that may make identification and sorting the recorded files easier. If no filename is chosen, the ABC SBC chooses its own ephemeral filenames. A fixed name may result in mixed PCAPs for different transactions. If custom filename is being used, it is recommended to not use fixed filename but include some date or time variable replacement in the filename to make it unique. In any case, the filename is relative to the path /data/traffic_log to avoid conflicts with the filesystem. Use a filename with .pcap extension.

Note that this action cannot be used multiple times for the same call meaningfully. In such case an error is reported in the SBC process log and only the first used logging action takes effect. Also due to the “inside view” nature how the packets are captured, they may display some minor differences from the actual traffic as seen on the net. Specifically TCP headers are not shown for SIP traffic sent using TCP.

Custom events also appear in this dashboard and also require a proper configuration on the SBC side as shown in the Section *Diagnostics Events*.

6.13 Monitor Troubleshooting

Should the ABC Monitor itself become a bottleneck in a network, it is a good idea to check its status. To see the status page, open its URL with path “/status” as shown in the screenshot Figure *Displaying ABC Monitor Status*. If the status shown on the page is not “green”, collect the statistics and contact FRAFOS support.

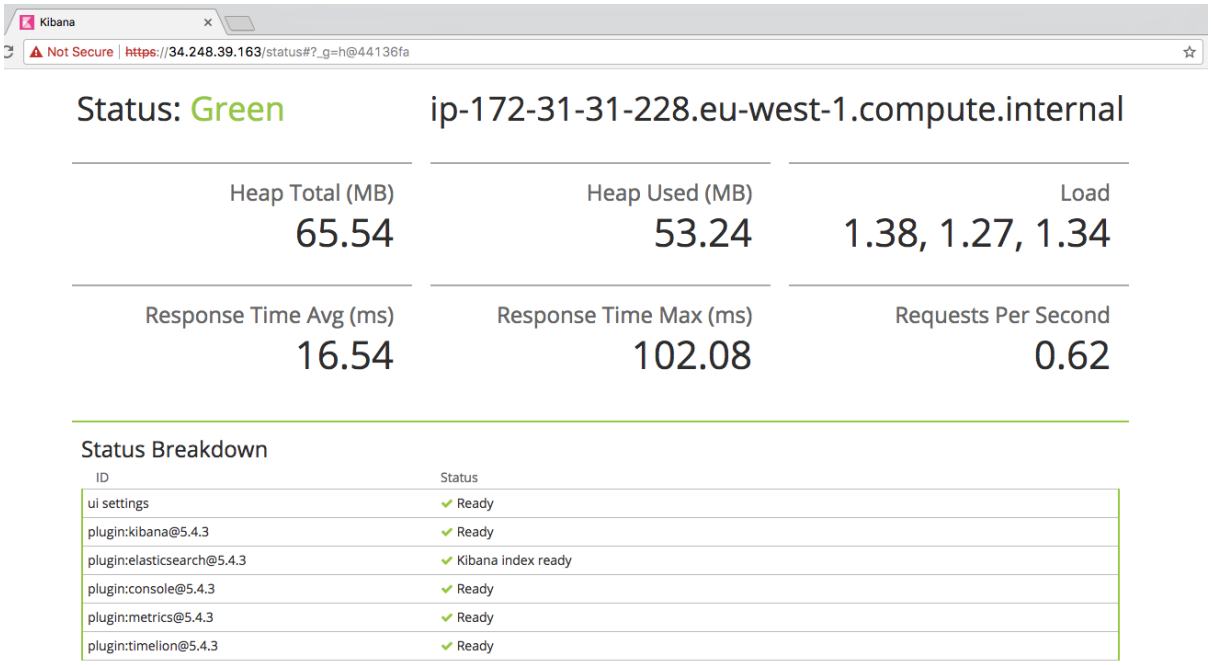

Fig. 42: Displaying ABC Monitor Status

Chapter 7

Analyze: Finding Patterns in Events using the ABC Monitor

See and keep silent. Sir Francis Walsingham, Queen Elizabeth's Principal Secretary

This section shows how the Monitor can be used when looking for security threats to a SIP service. Detecting presence of attacks and understanding their nature is a pre-requisite for devising proper policies that eliminate harmful and permit legitimate traffic.

Regular monitoring of a SIP service is proven to be the best way to keep operation smooth. Many administrators practice the simple and powerful habit to check their monitoring equipment as the very first thing in their working day. This section shows what to look for in the Monitor. Once a suspicious pattern is identified, the procedure is simple: keep filtering all regular events out until the events causing the pattern remain. Inspect their details and devise a proper policy.

In the following paragraphs we will walk you through the typical "stops" an administrator shall visit during his routing service check.

A good starting point is the **"Overview dashboard"**. Here a DoS attack can be discovered quickly as the security-related events gain dominance rapidly. Even Distributed DoS attacks can be spotted there because increased aggregate number of security events will reveal their presence, regardless how well masqueraded they are. This is shown in two 10-minutes examples in Figures *Total Number of Events in a Usual Situation* and *Total Number of Events when a Peak in IP Address Occurs Greylisting*.

Under normal operation, as shown in the upper part, the event types are balanced. There is a quite high number of greylisting events which is typical for a public SIP service exposed to scans from the public Internet. However, the number of these events still remains in the same order of magnitude as the other events. It is followed by the three registration events, also a typical situation for a public VoIP service in which telephones are turned off and on.

What should catch attention though is a situation in which an event type begins to dominate. This is the case in the lower diagram where greylisting events appear in unusual high quantities. That's good time to visit the Security Dashboard and find out more specifics.

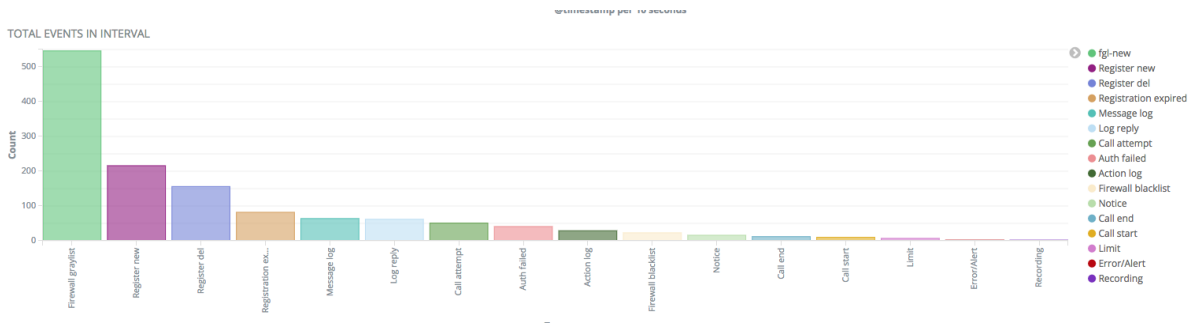

Fig. 1: Total Number of Events in a Usual Situation

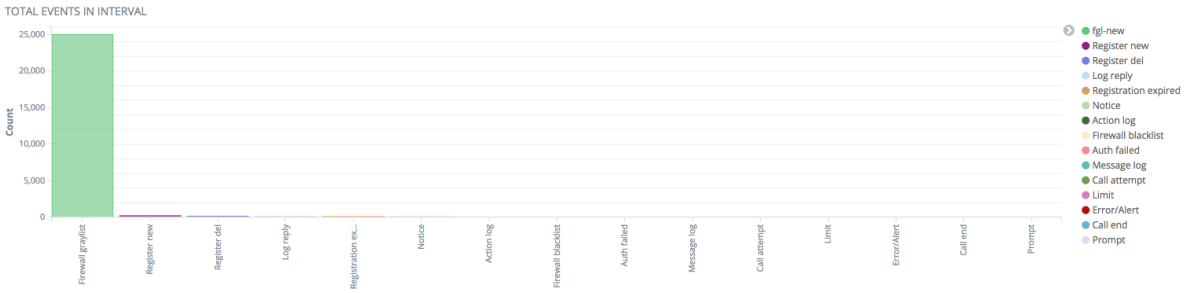

Fig. 2: Total Number of Events when a Peak in IP Address Occurs Greylisting

The **Security Dashboard** is introduced in more details in the Section *Security Dashboard*. The dashboard aggregates events that have relevance to security of a SIP service. Presence of authentication events points at password-guessing attacks (Section *Password Guessing Attacks*), presence of log-reply events at scanning attacks (Section *Scanning Attacks*), and presence of dropped-message events at rejected attempts to violate SIP site's security policies. So do Limit events unless they limit violation reaches an extend of a Denial of Service Attack (Section *Denial-of-Service Attacks*).

Not all SIP attacks have the ambition to ruin or gain control of a SIP service or a SIP user identity – some have the very simple motivation to find policy gaps that mistakenly permit phone calls. Such attempts are best spotted in the **Toplist Dashboard** as discussed in the Section *Dial-out Attempts*.

The following subsections detail on how to identify typical threats. Note that real incidents as recorded by FRAFOS are shown here that cannot be easily reproduced and may therefore include screenshots of previous ABC Monitor version with slightly different graphics.

7.1 Password Guessing Attacks

How would you feel if someone stole your password and was able to initiate and receive your phone calls, and all of it at your expense?

Password guessing attacks are really irritating: they are aiming at acquiring a user's password by guessing all thinkable variants of trivial passwords. The guesses are performed by machines. Once successful, the attacker can impersonate his victim and make phone calls on his behalf. The good thing is these attacks relatively easily manifest themselves by an abnormally high number of failed authentication attempts. If an attacker tries to stay under the radar, the total number of failures may not be apparent. Even then, however, the failures become easily visible when tied up to the attacker's IP address or geographic region. Such a situation is easily discovered in the Security Dashboard. A snapshot of the dashboard under attack is shown in the Figure *Events Reported during anEvents Reported during an Authentication Attack*. The toplist reveals that three of the ten most offending IP addresses come from the same subnet beginning with 200 and they are all clustered in South America.

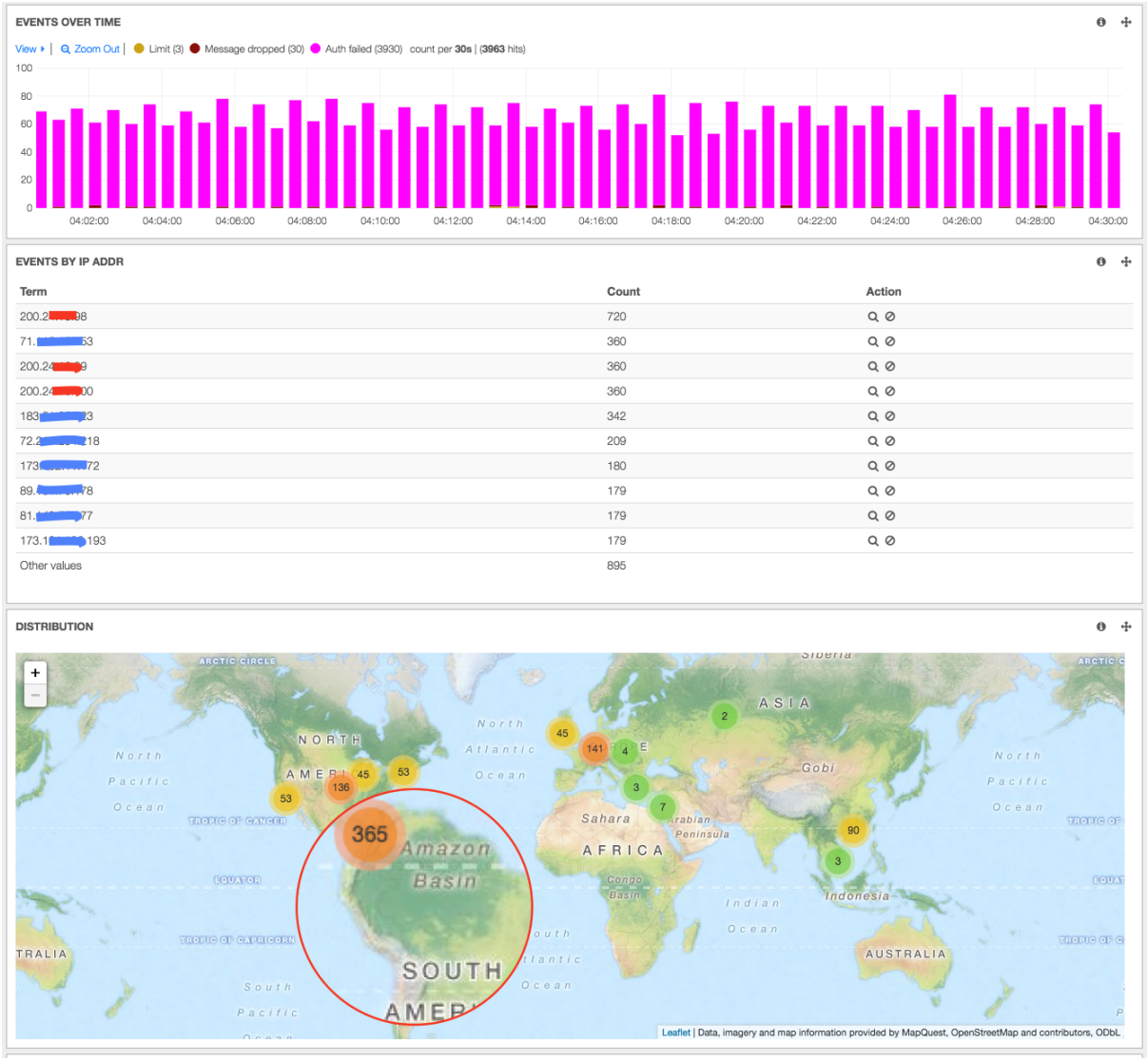

Fig. 3: Events Reported during anEvents Reported during an Authentication Attack

This intelligence is good enough to take counter-measures and lock the offenders by using some blocking techniques introduced in the Section Sec-security-rules.

An investigative administrator may go further and study the attacker's background. He may look at event details, see if these are attempts to register or make a phone call, what URIs are being used, or even return to the Home Dashboard, filter events by IP address and find out about all other activities of the offender. He may also filter the events from this offender out to see if this massive attack hasn't overshadowed another less intense one.

7.2 Scanning Attacks

Would you be irritated if you found someone fiddling with your house door's lock? Then get some nerves with SIP services operating on the public Internet: This is happening every second and is called "scanning attacks".

Scanning attacks are attempts to send SIP messages to various SIP addresses to find out if the server is connecting calls to them. Very often, attackers dial-out a telephone number many times with various prefixes in attempt to find a gap in dialing plan that will let them through. The destination number is often a premium number that generates revenue for its owner. Scanning attempts typically result in an increased number of failed authentications when the SIP service policy requires authentication, or some 403s (Forbidden)/480s(off-line)/604s(no such URI) when the

service is not serving a particular destination. Alone this information may be useful for an attacker – finding out that a destination exists may encourage him to mount a password-guessing attack against it.

Therefore it makes sense to check "Log Reply" events in the Security Console and find senders who are trying to reach many non-existing addresses or whose request often fail for other reasons. Such are often indeed originators of scanning attacks. We clearly see in the Monitor (Figure *Scanning Attacks Shown on Security Dashboard*) that the in the observed period the log-reply events peak up shortly.

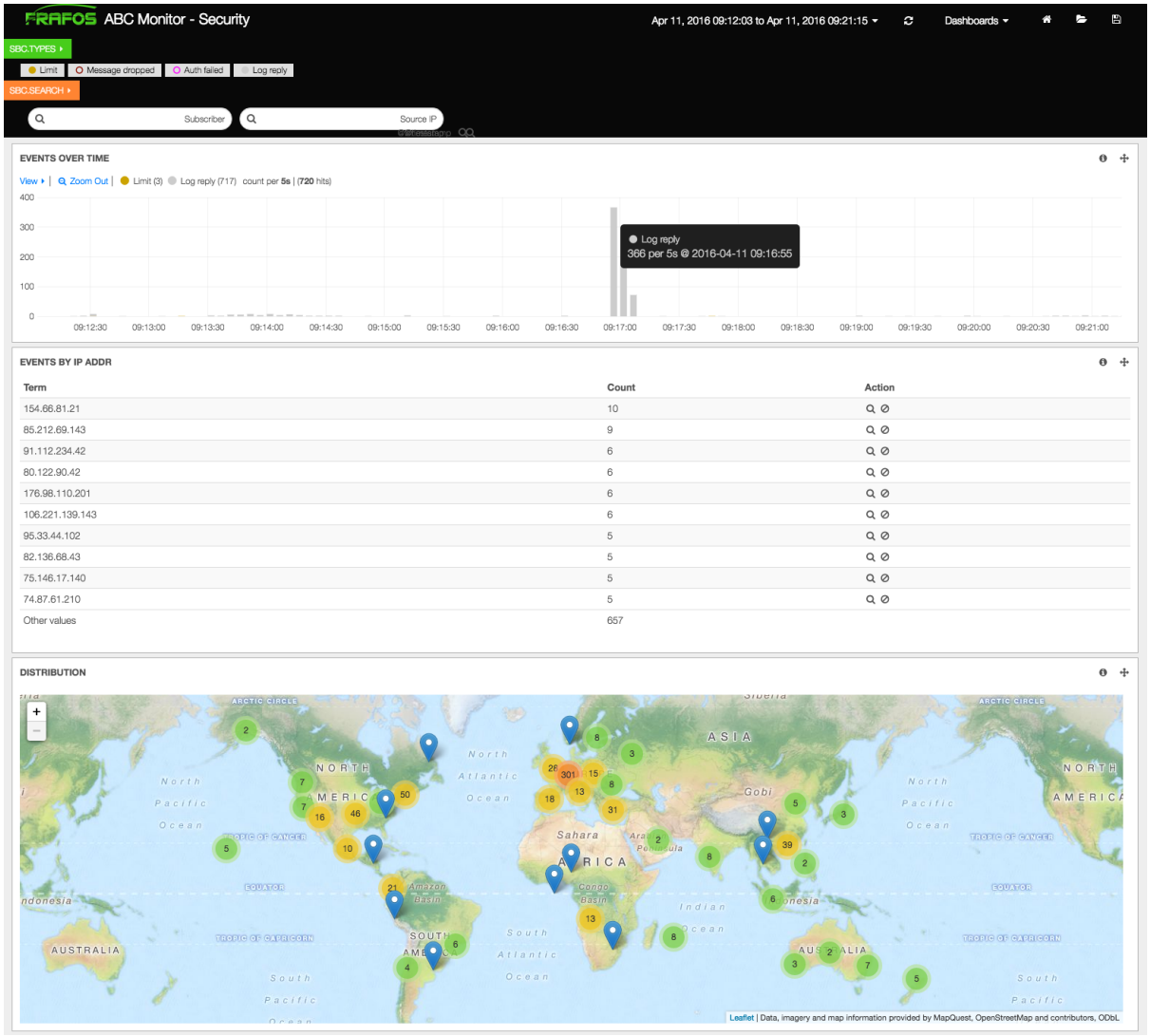

Fig. 4: Scanning Attacks Shown on Security Dashboard

The relatively flat distribution across IP address leaves us with a question whether that was a coincidence or an orchestrated distributed attack. In such cases administrator's insight is needed to judge the "Friend or Foe" dilemma. We discuss this further in the Section *Distributed Attacks*.

7.3 Denial-of-Service Attacks

Denial of service attacks (DoS) are simply excessive amounts of traffic targeted on a site with the sole purpose of impairing its service partially or completely. The ABC-SBC includes various counter-measures, probably the most effective one being the automated blocking described in Sections Sec-Abuse.

Presence of a DoS attack is best detected by the shaping actions that set traffic limits and report on their violations using *limit* events as documented in Section Sec-Traffic. Adequate care is needed when devising the limits to match legitimate traffic patterns otherwise legitimate traffic could be shaped and reported on as abuse.

Once the limits are exceeded the offenders appear on the Security Dashboard. If the excessive traffic recurs and Auto-blocking is turned on, the offending IP addresses will be blocked.

An example of such a situation is shown in Figure *Excessive Traffic Captured on a Security Dashboard*. The *limit* event peaks come in quantities on timeline around 11:30 and in Hong Kong in the map. Further analysis, which is not shown in the Figure, reveals that a single User Agent Type is sending total of 5 requests per second using several distinct URIs. An administrator may then judge if his limits are set correctly and revealed an actual DoS attender, or if his limit criteria have been to strict.

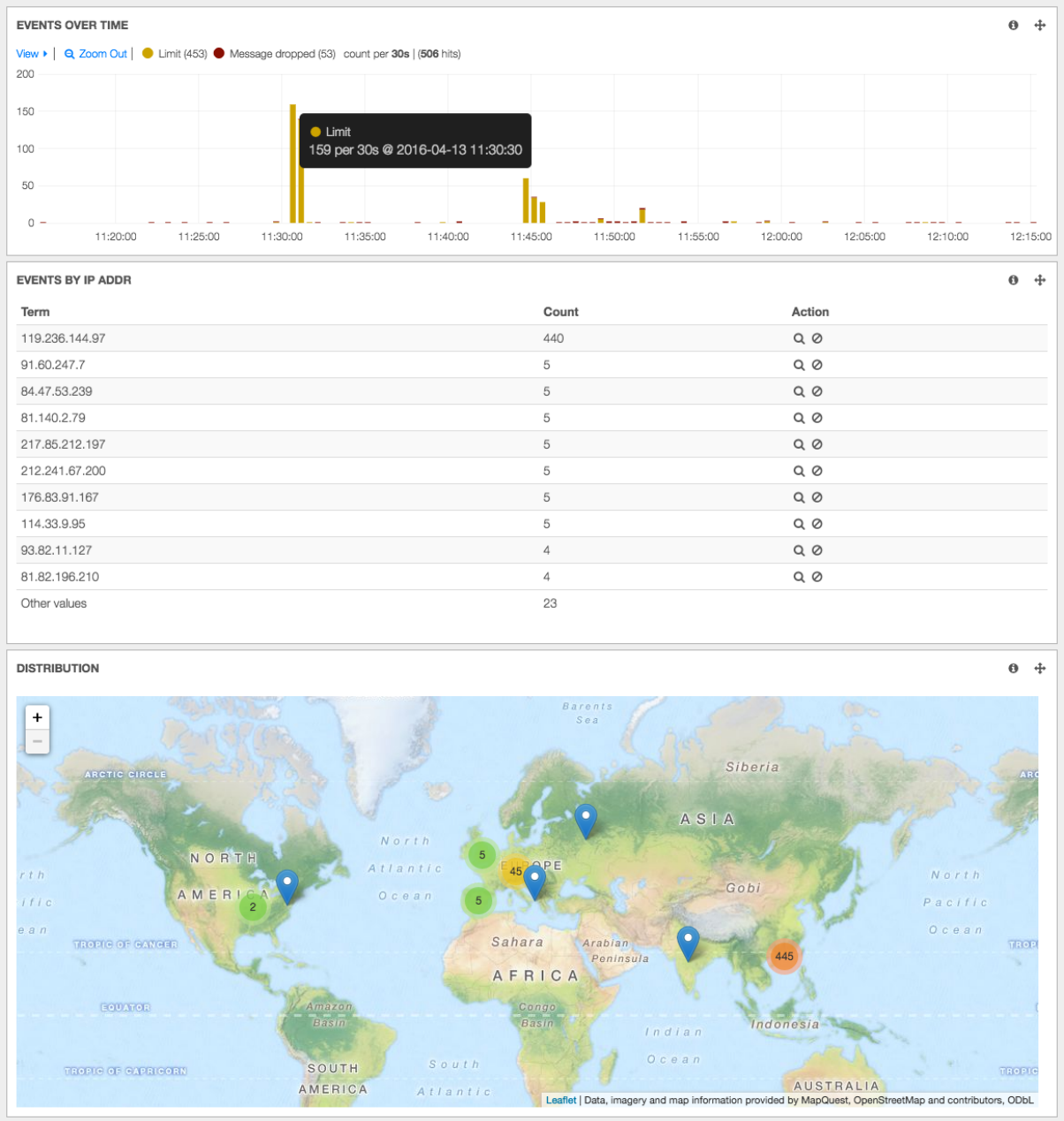

Fig. 5: Excessive Traffic Captured on a Security Dashboard

7.4 Distributed Attacks

Probably the closest non-computer analogy of a distributed attack is cross-fire when a target is exposed to an attack from many sides.

Distributed networking attacks are indeed a sophistication of other attack types that attempt to gather destructive force while remaining under radar screen. To obtain this explosive mix, the attacks are mounted from multiple sources. The most popularized distributed attack is the distributed denial of service attack (DDoS). However any other attack forms can be mounted in the distributed form and gain in invisibility and strike power.

The SBC can actually filter out substantial parts of a distributed attack on its own as shown in Section Sec-greylisting. The greylisting technique filters traffic which does not appear to do explicit harm yet it could have been probes prior to a real attack. It may be therefore worth to keep eye on the whitelisting and greylisting statistics. See the Figure *Network Statistics Dashboard Capturing a Failover Situation* in Section *Network and Statistics*

Dashboard.

Still it may be useful to detect presence of a distributed attack. A way to examine validity of a situation is to compare the number of registrations (over 3 thousands in the example shown) and the number of greylisted IP addresses (around 70,000). The order of magnitude difference here shows there are many more IP addresses causing useless if not harmful traffic than those originating legitimate traffic.

Another place worth looking at is the Overview Dashboard and break-down of events by type, see Figure *Total Number of Events when a Peak in IP Address Occurs Greylisting*. While distributed attacks are more or less good in hiding their source, their presence can still be detected by their intensity, i.e. by number of certain event types that depart from values experienced under normal situation.

7.5 Dial-out Attempts

One of the very common attack forms is dial-out attempts. Technically it is the simplest possible attack and yet it can provide high gains to the attacker. It is simply trying to dial telephone numbers to see if the attacker can connect to them without authorization.

We have shown techniques that can discover such attacks in a generic way. If the attacker tries too hard and fast, his efforts will be exposed by the limit checks that serve the purpose of DoS detection (Section *Denial-of-Service Attacks*). If the attacker is not making good guesses about the service's numbering plan, his attempts will be discovered using the response-checking technique used also for discovery of scanning attacks (Section *Denial-of-Service Attacks*). The attacker may be smarter though: he can seek for unprotected telephone numbers at low pace and using some educated guesses about dialing plans.

So if the attacker manages to remain "under radar" we still have to find out that someone is trying. The simplest way is to visit the "Top Lists Dashboard" – it shows a number of attempts and completed calls sorted by URI. Abnormal number of call attempts or even completed calls will appear here, even if the signaling rate remained moderate and the caller "hit" valid phone numbers. Therefore the daily-check routine for an administrator shall include visiting the "toplist Dashboard" and inspecting events of the most active users.

The most active users appear in the "toplist dashboard" sorted in descending order. The example screenshot in Figure *Example: Toplist of Attempted and Completed Calls* gives us a quite clear picture: a user whose name begins with "fmat" attempts many more times calls than anyone else in the observed period of time. It is also unusual that the same user does not appear in the top-list of completed calls.

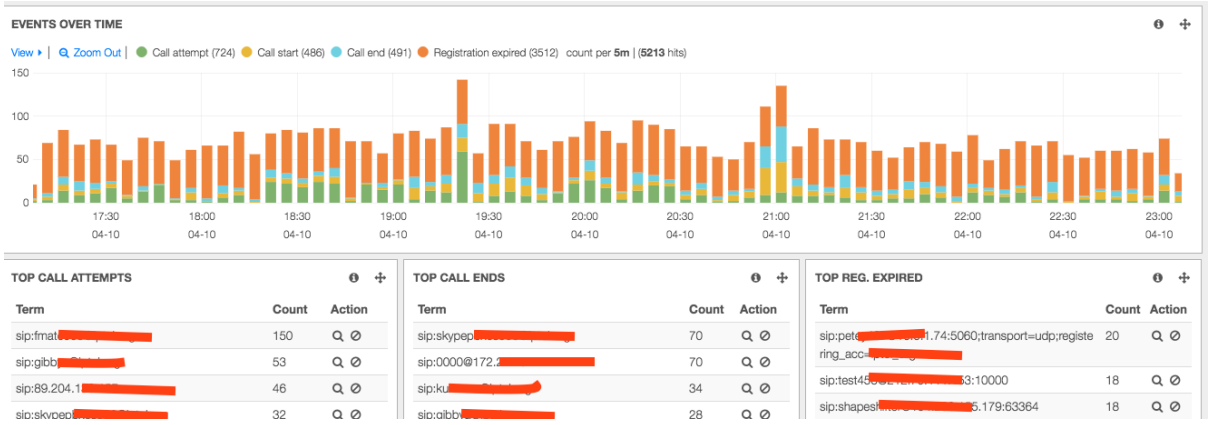

Fig. 6: Example: Toplist of Attempted and Completed Calls

To verify if this is a legitimate user, the same method can be used as introduced in Section *HOWTO Find a Needle in the Haystack: Iterative Event Filtering*: iterative filtering. When you narrow down the events to those originated by a suspicious From URI, the other toplists will show values relating to the specific originator. If for example, the destination toplists shows the same destinations only with varying prefixes, we know this was an attempt to break through. Example of such call attempt details are shown in Figure *Example: List of Attempted Destinations*. Obviously someone is trying to use a SIPCLI tool to call a number 48957372266 with varying prefixes (011, +,

9011, etc). The attempts are slow to remain under radar screen: they attempt every five minutes. They always yield the SIP response code 403 (Forbidden).

1–8 of 8

Time	type	attrs.from	attrs.to	attrs.source	attrs.from-ua	attrs.sip-code
November 11th 2017, 23:22:10.000	call-attempt	sip:55505@52.16.175.5	sip:56601148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 11th 2017, 23:18:08.000	call-attempt	sip:55505@52.16.175.5	sip:54401148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 11th 2017, 23:14:22.000	call-attempt	sip:55505@52.16.175.5	sip:53301148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 11th 2017, 23:10:58.000	call-attempt	sip:55505@52.16.175.5	sip:52201148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 10th 2017, 23:10:05.000	call-attempt	sip:55505@52.16.175.5	sip:80114895737266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 10th 2017, 23:06:08.000	call-attempt	sip:55505@52.16.175.5	sip:90114895737266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 10th 2017, 23:02:07.000	call-attempt	sip:55505@52.16.175.5	sip:+48957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403
November 10th 2017, 22:58:10.000	call-attempt	sip:55505@52.16.175.5	sip:01148957372266@52.16.175.5	158.69.248.156	sipcli/v1.8	403

Fig. 7: Example: List of Attempted Destinations

Chapter 8

Reference

8.1 Reference of Default Port Numbers

The reference lists port numbers the ABC Monitor uses. It is particularly useful when considering firewall policies for firewalls placed in front of the ABC Monitor. The reference lists default port numbers, transport protocols, and service listening on that port.

Port	Description
445 (default)	Access to the HTTPs web monitoring application. Value is configurable. Containerization port forwarding should use matching value.
873, 1873	(HTTPS / TCP) Allows ABC SBC to upload PCAP and WAV files to the ABC Monitor. 1873 is used for secure connection, 873 otherwise.
5044, 5045	(beat) ELK beats interface, allowing ABC SBC to send events to the ABC Monitor. 5045 is used for secure connection, 5044 otherwise.
9200	(HTTPS) Elasticsearch API (if enabled).

8.2 Command Line Reference

The administrative GUI is the preferred way to manage the ABC Monitor. However there are cases like the initial configuration and/or automation when accessing the ABC Monitor via Command Line is useful.

8.2.1 Configuration Management

CLI	Purpose	Reference
abc-monitor-backup-config	create backup of ABC Monitor config	*Backup and Restore Operations*
abc-monitor-restore-config	restore backup of ABC Monitor config	*Backup and Restore Operations*
abc-monitor-set-gui	configures ABC Monitor GUI https port	
abc-monitor-reset-access	Reset access to ABC Monitor and show the initial screen again.	

8.3 Reference of Used Open-Source Software

The ABC Monitor is based on the "ELK" stack which consists of the following components:

- logstash, Apache2 License
- Elastic Search, Apache2 License
- nginx, BSD-like

Additionally it relies on the Linux operating systems and numerous accompanying libraries and components provided by third parties under the following license terms:

- bash , GPLv3+
- cronie , MIT and BSD and ISC and GPLv2+
- crontabs , Public Domain and GPLv2
- dialog , LGPLv2
- dmidecode , GPLv2+
- ethtool , GPLv2
- expat (XML parser): MIT https://sourceforge.net/p/expat/code_git/ci/master/tree/expat/COPYING
- js , GPLv2+ or LGPLv2+ or MPLv1.1
- json-c: MIT (https://github.com/json-c/json-c/blob/master/COPYING)
- libcap , LGPLv2+
- libcurl: MIT/X derivate license https://curl.haxx.se/docs/copyright.html
- libevent: BDS-like http://libevent.org/LICENSE.txt
- libosip2 , LGPLv2+
- libpcap , BSD with advertising
- libtiff , BSD-like (http://www.libtiff.org/misc.html)
- libxml2 , MIT http://www.xmlsoft.org/FAQ.html
- monit , AGPLv3
- nginx, BSD-like
- ntp , (MIT and BSD and BSD with advertising) and GPLv2
- openssl, BSD-like https://www.openssl.org/source/license.html
- pciutils , GPLv2+
- python , Python
- python-jinja2 , BSD
- rsync , GPLv3+
- spandsp (g722, DTMF): LGPL
- speex , BSD
- sqlite , Public Domain
- stunnel, GPL
- tcpdump , BSD with advertising