netaxis SOLUTIONS

Security Hardening Guide

Technical Note

Table of Contents

1 Introduction

This technical note provides an overview of the different security mechanisms available on SRE. Procedures are also detailed to further enhance the overall security of the platform.

2 Standard Security Features

SRE implements several features to expose a secure environment to the external world. While the SRE is typically deployed deep inside the core network and not exposed to the Internet, in some contexts, according to the company internal policies, some of these features may be enabled/configured.

- Web GUI:
  - General:
    * Possibility to enable HTTPS with the built-in self-signed certificate or any certificate configured
    * Possibility to deploy the Web GUI behind a reverse proxy, providing an additional layer of security
    * Configurable listen address
  - Authentication:
    * Built-in Username & password authentication
    * Brute-force detection & black-listing
    * Configurable password strength
    * Alternative LDAP-based authentication
    * Alternative OpenID-based authentication

  - Authorization:
    * Fine-grained access management
    * Role-based access rights
  - Session management:
    * Cookie samesite, httponly & secure mechanisms
    * CSRF counter-measures
  - Audit log
- REST API:
  - Token-based API authentication
  - Username & password authentication
  - Audit log
- Database:
  - Access rights management
- OS:
  - Unprivileged user for processes

3 Hardening Guide

3.1 NGINX Reverse Proxy

It is possible to deploy an NGINX reverse proxy in front of SRE to further enhance the security by taking advantage of the NGINX flexibility to limit the TLS ciphers or add security headers on SRE responses.

Here is a sample */etc/nginx/conf.d/sre-proxy.conf* file to set up such proxying. Adapt *FQDN* to match the GUI FQDN and the *proxy_pass* directive to the localhost port the process *sre-gui* listens on.

Warning

Even if the SRE GUI runs behind an HTTPS reverse proxy, it is recommended to keep the SRE GUI running in HTTPS to have the session cookies managed in advance secure mode.

```
server {
    listen 80;
    server_name <FQDN>;
    return 301 https://<FQDN>$request_uri;
}
```

```

server {
    listen 443 ssl;
    server_name FQDN;
    access_log  /var/log/nginx/sre_access.log;
    error_log  /var/log/nginx/sre_error.log debug;
    ssl_certificate    /opt/sre/etc/cert.pem;
    ssl_certificate_key /opt/sre/etc/privkey.pem;

    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:
    ↪ ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-
    ↪ GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:
    ↪ ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:
    ↪ ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:
    ↪ ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-
    ↪ AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA
    ↪ :!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK";
    ssl_ecdh_curve secp384r1;
    ssl_session_cache shared:SSL:10m;
    ssl_session_tickets off;
    #ssl_stapling on;
    #ssl_stapling_verify on;

    gzip                on;
    gzip_min_length     2000;
    gzip_proxied        expired no-cache no-store private auth;
    gzip_types          *;

    #Version Disclosure in server section
    server_tokens off;

    #Clickjacking: DENY or SAMEORIGIN
    add_header X-Frame-Options "DENY";

    #enable security-Headers
    add_header X-Content-Type-Options "nosniff";
    add_header Content-Security-Policy "frame-ancestors 'none'";
    add_header Referrer-Policy "no-referrer";
    add_header X-XSS-Protection "1; mode=block";

    #STS
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";

    location / {
        proxy_pass   https://localhost:8443;
        proxy_redirect     off;
        proxy_set_header   Host                 $host;
        proxy_set_header   X-Real-IP            $remote_addr;
        proxy_set_header   X-Forwarded-For      $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto    $scheme;
    }
}
```

3.2 PostgreSQL Database

You should remove all unauthenticated access, and set the method to md5, both for localhost and remote connections in *pg_hba.conf*:

```
host    all             all             127.0.0.1/32            md5
host    all             sre             10.0.161.0/24           md5
```

This is a sample config file:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 ident
# Allow replication connections from localhost, by a user with the
# replication privilege.
host    replication     postgres        ::1/128                 ident
host    replication     repmgr          127.0.0.1/32            trust
local   repmgr          repmgr                                  trust
host    all             sre             10.0.161.0/24           md5
host    replication     repmgr          10.0.161.62/32          trust
host    repmgr          repmgr          10.0.161.62/32          trust
host    replication     repmgr          10.0.161.64/32          trust
host    repmgr          repmgr          10.0.161.64/32          trust
host    replication     repmgr          10.0.161.63/32          trust
host    repmgr          repmgr          10.0.161.63/32          trust
host    replication     repmgr          10.0.161.60/32          trust
host    repmgr          repmgr          10.0.161.60/32          trust
```

By default, the SRE will use the user postgres to perform low-level tasks, such as creating a new database in case of new datamodel creation through the datamodel editor. As local connection with user postgres is not authorized anymore, the solution is to grant the right to create DB to user sre:

```
postgres=# ALTER USER sre CREATEDB;
ALTER ROLE
```

And to instruct the SRE to use the another user than the default postgres user, adapt the file */opt/sre/etc/sre.cfg* with parameter *dme_superuser* under section [db] with username:password to use. Example:

```
[db]
dme_superuser=sre:secretpassword
```

3.3 OS Firewall Configuration

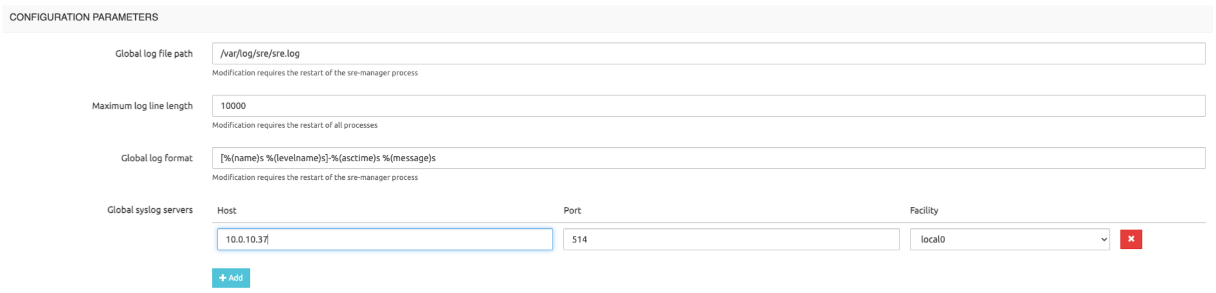
Here is a sample *iptables-save* config file that can be loaded with *iptables-restore*:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT

#allow SSH in
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT

#allow ports for GUI (adapt port if GUI listens on another port)
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT

#REST API, adapt source subnet
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5000 -s 10.0.0.0/16 -j
    ↪ ACCEPT

#in these following directives, adapt the source subnet to cover the different
    ↪ SRE nodes
#PostgreSQL (replication)
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5432 -s 10.0.0.0/16 -j
    ↪ ACCEPT
#MongoDB (service + replication)
-A INPUT -m state --state NEW -m tcp -p tcp --dport 27017 -s 10.0.0.0/16 -j
    ↪ ACCEPT
#logs & stats from any node to EM
-A INPUT -m state --state NEW -m tcp -p tcp --dport 10000 -s 10.0.0.0/16 -j
    ↪ ACCEPT
#acconting events
```

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 10002 -s 10.0.0.0/16 -j ACCEPT
#acconting replication
-A INPUT -m state --state NEW -m tcp -p tcp --dport 10003 -s 10.0.0.0/16 -j ACCEPT

#SIP interface (UDP example), adapt source subnet
-A INPUT -p udp --dport 5060 -s 10.0.0.0/16 -j ACCEPT

#HTTP service logic processor interface, adapt source subnet
-A INPUT -m state --state NEW -m tcp -p tcp --dport 6000 -s 10.0.0.0/16 -j ACCEPT

#ENUM service logic processor, adapt source subnet
-A INPUT -p udp --dport 53 -s 10.0.0.0/16 -j ACCEPT

#reject
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited

COMMIT
```

3.4 Logs Management

It is possible to configure one or more Syslog servers to forward SRE logs to an external system where encryption can be performed at rest and prevent any tampering with the local log files. To do so, head over to the System/Settings/Logging page and set one or more Syslog receivers, as in this screenshot.



CONFIGURATION PARAMETERS

Global log file path: /var/log/sre/sre.log
Modification requires the restart of the sre-manager process

Maximum log line length: 10000
Modification requires the restart of all processes

Global log format: [%(name)s %(levelname)s]-%(asctime)s %(message)s
Modification requires the restart of the sre-manager process

Global syslog servers

Host	Port	Facility
10.0.10.37	514	local0

These logs can be forwarded to one or more Syslog receivers:

- Global log
- GUI audit log
- REST API audit log
- Service logic execution log
- Accounting log