martini
security

Martini Security
Deploying STIR/SHAKEN with Kamailio

Version 1.1

Deploying STIR/SHAKEN with Kamailio

Background

This document outlines various methods for implementing STIR/SHAKEN using Kamailio, in conjunction with services provided by Martini Security.

Starting from version 5.6, Kamailio natively supports two modules for STIR/SHAKEN implementation. This guide will provide instructions on deploying these modules, along with Olive, an ACME client developed by Martini Security that facilitates ordering and auto-renewal of STIR/SHAKEN signing certificates.

The recommended approach involves using Vermouth, an application that operates as a separate service alongside Kamailio. Vermouth provides a REST API for signing and verifying SIP calls within Kamailio and includes its own lightweight ACME client for automatic certificate renewal.

Sample Configuration Notes

The information provided in this document has been tested on Ubuntu Server 22 LTS, using the latest packages available as of April 4, 2023. While not guaranteed, it is assumed that other Debian-based distributions or RHEL/Fedora systems may also be compatible.

Configuration snippets were validated using Kamailio 5.6 configured with MySQL 8. Tests were conducted with both static Debian packages and binaries compiled from the source. Alternative database engines can be used in place of MySQL, as Kamailio supports several options. However, a database engine is necessary for this setup.

The source code was cloned using:

```
git clone --branch 5.6 https://github.com/kamailio/kamailio.git
```

To install static libraries, add the following lines to the /etc/apt/sources.list file:

```
deb http://deb.kamailio.org/kamailio56 jammy main
deb-src http://deb.kamailio.org/kamailio56 jammy main
```

This guide assumes that the reader possesses a strong understanding of Kamailio, its ecosystem, and SIP. Familiarity with the ATIS STIR/SHAKEN specifications is also beneficial.

Getting Started

Before deploying, you must first create an account with Martini Security and obtain an API key. This process involves four steps:

1. Register for an account with Martini Security and acquire an API key.
2. Submit your FCC 499 filer ID.
3. Select a subscription plan and complete payment.
4. Have the registered 499 filer approve the request to represent their organization.

Account creation typically takes only a few minutes.

After registering, creating ACME credentials can be done in just a few steps.

1. Click ACME Clients on the left navigation bar.
2. Click to + to add a new client.
3. Give it a label meaningful to you.
    - Ex: “signing-node-1”

4. Locate the sections labeled “Key ID” and “API Key” and copy these somewhere. You will need them soon regardless of which mechanism you will use to implement STIR/SHAKEN.

For more information, watch this video: https://www.youtube.com/watch?v=CXvR-jyJVx4&t=1s

Additionally, you will need your STI-PA API credentials and OCN, which can be found in the credentials you created at iConectiv’s STIR/SHAKEN provider portal.

General Kamailio + STIR/SHAKEN Guidance

All STIR/SHAKEN implementations require the following information:

- X5U
  - Public STIR/SHAKEN certificate HTTP URL
- Attestation Level
  - A, B, or C as defined by ATIS
- OrigTN
  - Calling party
- DestTN
  - Called party
- OrigID
  - Call identifier specified by signer (unspecified OrigIDs cause most signing engines to generate and use a UUID)
- Private Key Path
  - Signing key used to sign calls

Kamailio offers numerous built-in pseudo variables. In the examples provided, from user `($fU)` and to user `($tU)` are used for signing requests. Note that for US-based calling and called parties, signing requests must use the 11-digit format –i.e., “1NXXNXXXXXX”.

Testing was conducted with SIP To and From User data formatted as “+1NXXNXXXXXX”, so string manipulation was performed to remove the “+”. Keep in mind that the sources of origTN and destTN are specific to your VoIP environment.

The request to add the identity header should be made just before sending the INVITE to an external peer. The request to verify the identity should be made as soon as a SIP INVITE is received from an external SIP peer. These practices apply consistently across all STIR/SHAKEN deployments.

Using Vermouth

About

Vermouth is a system service that listens on a configured IP and port for signing and verification requests, providing appropriate responses. It offers various services, including ACME operations for requesting and renewing signing certificates.

Installation

Vermouth can be installed from a package or built from the source. After downloading the current release (or manually building the package), install it using `dpkg -i vermouth-X.X.X.deb`. By default, `/etc/vermouth` is the YAML master configuration file that must be modified before starting Vermouth.

Vermouth Configuration

First, open `/etc/vermouth` in your preferred text editor. Most of the default fields already contain the recommended values and can be left unchanged. However, the following lines need to be modified:

```
sti-pa:
  api_user_id: <STI-PA API Username>
  api_password: <STI-PA API Password>
sti_as:
  carrier_ocn: <your OCN>
  acme_eab_key_id: <ACME Key ID from Martini Security>
  acme_eab_key_secret: <ACME Key from Martini Security>
```

After making the required changes, use `systemctl start vermouth` to initiate STIR/SHAKEN services.

Kamailio Configuration

Kamailio will be configured to make synchronous HTTP requests to the Vermouth API endpoints. The returned values can be used for further decision-making.

Your VoIP environment will determine the necessary values and how they need to be cleaned up before making requests to Vermouth. For instance, in signing requests, you may prefer using the outbound P-Asserted-Id instead of the "from" field. This is because the "from" field might contain "Anonymous" with the Privacy:id header set, while the P-Asserted-Id has the actual origTN to be used in a signing request.

The simple test environment used to create these examples had SIP To and From data formatted as "+1NXXNXXXXXX", so the "+" was removed before making signing requests.

```
...
# If using TLS, this line must come after TLS is loaded
loadmodule "http_client.so"
...
# The default timeout is infinite. We will set this to 2 seconds
modparam("http_client", "connection_timeout", 2)
...
request_route {
...
# The verification request accepts two possible origTN values and two
# possibles destTN values. It is recommended to use the From/PAID and
# To/R-URI

$var(cleanFrom) = $(fU{s.substr,1,0});
$var(cleanTo) = $(tU{s.substr,1,0});
$var(paiduser) = $(ai{uri.user});
$var(cleanRURI) =  $(ai{uri.user}){s.substr,1,0});


http_client_query("http://127.0.0.1:8085/ezstir/v1/verification/$var(cleanFrom)/\
$var(paiduser)/$var(cleanTo)/$var(cleanRURI)/$hdr(Identity)","$var(result)");

# The result variable will contain VALID-A, VALID-B, or VALID-C for all
# valid cases. BAD-SIGNATURE or NO-SIGNATURE will appear otherwise.

if ($rc == "200") {
     xlog("L_INFO","$var(result)");
}
```

```
...
$var(cleanFrom) = $(fU{s.substr,1,0});
$var(cleanTo) = $(tU{s.substr,1,0});

# The Request format is: /origTN/destTn/attestestation/origID
# origId is always optional. A random UUID will be generated.
# If you just need an A attestation, then providing just from the origTN and
#   destTN is sufficient.



http_client_query(\
    "http://127.0.0.1:8085/ezstir/v1/signing/$var(cleanFrom)/$var(cleanTo)", \
    "$var(result)" \
);
if ($rc == "200") {
     # A fully formed SIP header beginning "Identity: ... " is returned
     #  from the HTTP request and can be appended directly to the INVITE
     append_hf("$var(result)\r\n");
}
...
}
...
```

Using Olive and Kamailio Modules

First, we need to install Olive. This is the Martini Security-developed command-line ACME client that will be used for ordering and renewing STIR/SHAKEN certificates.

```
wget https://storage.googleapis.com/martini-security/download/olive
chmod a+x olive
```

To simplify the process, let's configure two variables for the ACME credentials. These can be obtained after creating a new ACME client within the Martini Security web application.

```
export ACME_KEY_ID=<acme key id>
export ACME_KEY=<acme key>
```

Next, we need to register our ACME account.

```
olive acme register --key-id ${ACME_KEY_ID} --key ${ACME_KEY} \
--contact <your email> --agree-tos
```

This will create a hidden directory in your `$HOME` directory where we will need to store our STI-PA credentials for the certificate we are about to request.

```
cat <<EOF >> ~/.mrtsec/.stipa.yaml
user_id: <STI-PA API userid>
password: <STI-PA API password>
ocn: <your OCN>
EOF
```

Now we can request an SPC token and then order our first STIR/SHAKEN certificate.

```
olive stipa token --key-id ${ACME_KEY_ID} \
--config ~/.mrtsec/.stipa.yaml --out=/tmp/spc.json

olive acme order --key-id ${ACME_KEY_ID} --spc /tmp/spc.json
```

Within moments, we should have the certificate, key, and cert repository (X5U URL). Note that subsequent calls to order will automatically renew when the validity period is less than 7 days until expiry. An SPC token should always be requested before ordering or renewing a certificate.

Later on, we will script the auto-renewal and reload of signing cert data.

Base Kamailio Config

Both modules discussed in this document require information about the X5U publication URL and the private key to sign the call. This guide uses Kamailio's htable module (hash table) as a quick and convenient method to store this information.

The following example can be easily adapted to SQL syntax, but the provided example uses MySQL.

First, add a new htable reference to the Kamailio DB and then add the required keynames. Using the information from the certificate ordering process, UPDATE those records with the correct path. For scripting the automatic renewal of the certificate, you will need an UPDATE structured like this:

```
CREATE TABLE `stirshaken_config` (
`id` int(11) NOT NULL,
`key_name` varchar(64) NOT NULL DEFAULT '',
`key_type` int(11) NOT NULL DEFAULT '0',
`value_type` int(11) NOT NULL DEFAULT '0',
`key_value` varchar(128) NOT NULL DEFAULT '',
`expires` int(11) NOT NULL DEFAULT '0',
PRIMARY KEY (`id`) );

INSERT INTO stirshaken_config (id, key_name) VALUES (1,'priv_key_path');
INSERT INTO stirshaken_config (id, key_name) VALUES (2,'repo_url');

UPDATE stirshaken_config \
    SET key_value='https://p.mtsec.me/XXXX/001122334455.pem' \
    WHERE key_name='repo_url';

UPDATE stirshaken_config \
    SET key_value='/root/.mrtsec/certificates/001122334455667788.key' \
    WHERE key_name='priv_key_path';
```

Now, we need to add the appropriate kamailio config. Change your #!define as necessary and ensure DBURL is set correctly within the config.

```
#!define WITH_MYSQL

loadmodule "htable.so"

modparam("htable", "db_url", DBURL)
modparam("htable", "htable","stirshaken=>size=2;dbtable=stirshaken_config;")
```

Use `kamctl` to restart Kamailio and reload the configuration. At any time, you can use the following commands to reload or view the stored STIR/SHAKEN values, which will be used by the Kamailio modules.

```
kamcmd htable.reload stirshaken
kamcmd htable.dump stirshaken
```

At any point within the Kamailio config, `$sht(stirshaken=>priv_key_path)` will be used to reference the private key file, and `$sht(stirshaken=>repo_url)` will be used to reference the public certificate repository URL (X5U).

Scripting Cert Renewal and STIR/SHAKEN Config Reload

When the certificate is close to expiring, we can automate the renewal and configuration reload using a shell script. Here is an example of such a script.

```
#!bin/bash

# You could choose to pass this in as a param...
ACME_KEY_ID=<acme key id>

# Clear and prep for ACME result
echo "" > /tmp/acme_result
ACME_RESULT=/tmp/acme_result

# Get an SPC Token and Proceed to attempt to renew cert
olive stipa token --key-id ${ACME_KEY_ID} --config ~/.mrtsec/.stipa.yaml \
    --out=/tmp/spc.json \
    && olive acme order --key-id ${ACME_KEY_ID} --spc /tmp/spc.json > $ACME_RESULT

# (continued on next page)
# If this failed, we will try again tomorrow
if [ $? -ne 0 ]
then
    exit
Fi

# Parse the output from Olive
while IFS=":" read -r key value; do
    case "$key" in
      "found valid certificate") valid=1 ;;
```

```
      "key") key="$value" ;;
      "repository") repo="$value" ;;
    esac
done < "$ACME_RESULT"

# If we are outside of 7 days of expiration, Olive informs us that we already have a key
# A renewal can be forced with --renew on our "acme order" command
if [ $valid -eq 1 ]
then
    exit
Fi

# Olive will overwrite the cert and private key in the .mrtsec directory.
# Copy the private key somewhere else to make this transition seamless
MYDATE=`date +%F`
NEWKEY=`mktemp --suffix=.$MYDATE /etc/kamailio/XXXXXXXXXX` || exit 1
cp -f "$key" "$NEWKEY"

mysql kamailio -e "UPDATE stirshaken_config SET key_value='$NEWKEY' \
    WHERE key_name='priv_key_path';"
mysql kamailio -e "UPDATE stirshaken_config SET key_value='$repo' \
    WHERE key_name='repo_url';"

# kamctl db exec would also function more generically,
# but a simple expect script would be needed for RW db exec privileges with default
 ↪  kamctl behavior

kamcmd htable.reload stirshaken
```

Module SecSIPId

Installation

The secsipid module can be quickly installed via:

```
apt install kamailio-secsipid-modules
```

Configuration

To verify incoming SIP calls, the secsipid module utilizes the system root store to determine the validity of the cert chain. However, since the STIR/SHAKEN CA roots are not included in any default OS store, they must be added. For this purpose, Martini Security has provided a mirror.

Here are the steps to add the concatenated certs (PEM encoded) from the calist file to the root store in Ubuntu:

```
cd /usr/local/share/ca-certificates
wget https://wfe.prod.martinisecurity.com/v1/stipa/calist
csplit -q -z -f "cert-" -b '%02d.crt' calist \
'/-----BEGIN CERTIFICATE-----/' '{*}'
```

```
update-ca-certificates
```

To validate incoming SIP calls, the secsipid module uses the system root store to check the certificate chain's validity. However, since the STIR/SHAKEN CA roots are not included in any default OS store, they need to be added. For this purpose, Martini Security has provided a mirror.

Follow these steps to add the concatenated certs (PEM encoded) from the calist file to the root store in Ubuntu:

```
loadmodule "secsipid.so"

...
request_route {
...
if(secsipid_check_identity("")) { ... } else { ... }
...
$var(cleanFrom) = $(fU{s.substr,1,0});
$var(cleanTo) = $(tU{s.substr,1,0});

secsipid_add_identity(\
    $var(cleanFrom), $var(cleanTo), "A", "", \
    $sht(stirshaken=>repo_url), $sht(stirshaken=>priv_key_path)\
);
...
}
...
```

Several parameters should be configured while working with the secsipid module. Please refer to the documentation to customize its behavior.

Module Stirshaken

Installation

As of writing this document, installing the stirshaken module can be quite cumbersome. It must be built from source and has several dependencies that also need to be built. In addition, the kamailio source code is required to build the module itself after building the underlying library known as libstirshaken.

To begin, prepare the development environment:

```
apt install build-essential autoconf libtool pkg-config \
libssl-dev libcurl4-openssl-dev cmake uuid-dev bison flex
```

We need jansson, libjwt, and libks to build libstirshaken. Once built, the Kamailio Stirshaken module can be copied into place.

```
cd /usr/local/src
wget https://github.com/akheron/jansson/releases/download/v2.14/jansson-2.14.tar.bz2
```

```
tar xvf jansson-2.14.tar.bz2
cd jansson-2.14
./configure && make && make install

cd /usr/local/src
wget https://github.com/benmcollins/libjwt/releases/download/v1.15.2/libjwt-1.15.2.tar
↪  .gz
tar xvf libjwt-1.15.2.tar.gz
cd  libjwt-1.15.2
./configure && make all && make install

cd /usr/local/src
git clone https://github.com/signalwire/libks.git
cd libks
cmake . && make && make install

cd /usr/local/src
git clone https://github.com/signalwire/libstirshaken.git
cd libstirshaken
./bootstrap.sh
./configure
make CFLAGS="-Wno-error=deprecated-declarations"
make install

cd /usr/local/src
git clone --branch 5.6 https://github.com/kamailio/kamailio.git
cd kamailio
make modules modules=src/modules/stirshaken/
cp src/modules/stirshaken/stirshaken.so /usr/lib/x86_64-linux-gnu/kamailio/modules/
```

Configuration

Similar to what was required for SecSIPId, stirshaken needs to know which CA roots to use for verifying calls. The difference is that stirshaken expects a directory of certificates instead of relying on the OS's root store, which is beneficial.

To make the concatenated certs (PEM encoded) from the calist file available to the module, follow these steps:

```
mkdir ~/.mrtsec/ca
cd ~/.mrtsec/ca
wget https://wfe.prod.martinisecurity.com/v1/stipa/calist
csplit -q -z -f "cert-" -b '%02d.pem' calist \
'/-----BEGIN CERTIFICATE-----/' '{*}'
openssl rehash .
# Optionally
rm -f calist
```

Next, modify the Kamailio configuration file.

```
loadmodule "stirshaken.so"
...
modparam("stirshaken", "vs_ca_dir", "/root/.mrtsec/ca")
...
request_route {
...
if (1 == stirshaken_check_identity()) { ... } else { ... }
...
$var(cleanFrom) = $(fU{s.substr,1,0});
$var(cleanTo) = $(tU{s.substr,1,0});

stirshaken_add_identity_with_key(\
    $sht(stirshaken=>repo_url), "A", $var(cleanFrom), \
    $var(cleanTo), "", $sht(stirshaken=>priv_key_path) \
);
...
}
...
```

When working with the stirshaken module, several parameters should be configured. Please refer to the documentation to customize its behavior.