SecureDNA

Synthclient quickstart

SecureDNA

Contents

SecureDNA

Introduction

This is a *minimal* set of instructions for starting to screen. It does not attempt to describe the rationale, threat model, or architecture of the entire system, nor does it delve into advanced options for provisioning or use.

`synthclient` is the primary tool for interacting with SecureDNA's screening infrastructure. It provides a REST API for automated use, and provides a web UI for visualizing results and for debugging. The REST API should be what other order automation uses to do screening.

See the API description for how to call on the `synthclient` endpoint and interpret its results.

This documents how to create tokens using a web browser. For automated generation of many tokens, it's probably easier to create them using the command line interface.

Obtaining certificates

Generating a certificate request

`synthclient` is free to use, but requires a certificate for authentication to the screening servers to prevent various types of abuse.

To generate a certificate request, fill out the form at https://securedna.org/cert-request/ and download the three files it generates.

- `companyname-leaf.certr`: This is a *certificate request.* You will send this file to SecureDNA as detailed below.
- `companyname-leaf.priv`: This is a *private key,* protected by your passphrase. Do *not* send this to SecureDNA, do not lose it, and do not share it. (If you send it to SecureDNA by mistake, we will ask you to generate a new certificate request and to throw away the existing request.)
- `companyname-leaf.pub`: This is a *public key,* which can be freely shared. You do not need it for the scenarios described in this quickstart, but you should preserve it for future use in more-complicated scenarios, not described here, in which you might reuse an existing keypair. (One such example concerns *certificate cross-signing,* used to establish multiple paths back to a parent cert for redundancy in case an intemediate cert is lost, or to enable phased rollover of certificate expirations. None of these are necessary for typical use of `synthclient`.)

The name and email in the above command are baked into the resulting certificate. If we later detect some problem with the use of your certificate (such as abuse), we will send mail to that email address.

The passphrase you enter in that form is used by your browser to create your certificate request, but it is *never* sent anywhere and never leaves your browser. We shall refer to this as *passphrase A.* This passphrase is used to protect the private key associated with your certificate, and should be cryptographically strong. Anyone who obtains the private key and your passphrase may use your certificate as if it is their own, so protect it well and do not share it. (Your passphrase may be as long as you wish, and may include spaces and punctuation as you desire, however note that any *leading* or *trailing* whitespace [spaces, tabs, or newlines] will be removed before use.)

Sending your certificate request

Go to https://securedna.org/start/ and fill out the form. Upload `companyname-leaf.certr` where indicated.

A human will review your request for reasonability. This aim is to prevent thousands of spurious certificates being automatically issued to a botnet and similar scenarios. If your message is sent from an address different from the

email embedded in the certificate, we may confirm the email in the certificate to guard against typos before taking action on the certificate request.

We will reply via email with an attached certificate named `companyname-leaf.cert`, e.g., your original name, minus the `r`.

Generating synthesizer tokens

Once you have your certificate, you may use it to make *synthesizer tokens.* How many you make depends on how your business is structured.

- *If you are a centralized provider,* you can make a single token for all of your screening, and supply that token with every request.
- *If you are a benchtop manufacturer,* you should make one token per printer. (As mentioned above, if your printers are dumb peripherals, then make one per controller, or one per customer site, or (if cloud-based) one per customer account, etc.)
- If you are a biosafety officer in charge of *approving synthesis exemptions*, you may need a synthesizer token to approve certain exemption requests.

To create a synthesizer token, fill out the form at https://securedna.org/synth-token/ using your cert and private key. Neither the private key, nor the old or new passphrases, will leave your browser: the token is generated by code running in your browser, reading the contents of the private key without uploading it anywhere. We shall refer to the new (token's) passphrase as *passphrase B,* and it is this passphrase which you will supply to `synthclient` when you run it, along with the token. For best security, it should be different from *passphrase A.*

Select the purpose of your token, then fill out the remaining fields.

For centralized providers:

- *Rate limit* is the maximum number of nucleotides per day that you expect to screen across all the devices associated with this token.

For benchtop synthesizers:

- *Model* should be the name of this particular model of synthesizer, so it's possible for a third party to verify that the provided rate limit is reasonable for that model. (This can be any Unicode string.)
- *Serial* is the benchtop's serial number. (This can be any Unicode string.)
- *Rate limit* is the maximum number of nucleotides per day that your synthesizer can produce.

For biosafety officers:

- *Rate limit* is the maximum number of nucleotides per day that you expect to screen across all exemption requests you handle.

In all cases:

- *Domain* should match the domain of your company (e.g. yourcompany.com). Typically, this should also match the (apparent) domain of your email, but this is evaluated by a human if the need arises.
- *Days valid* controls the *maximum* longevity of the synthesizer token. If a parent token in the certificate chain expires sooner than this, the child token will become invalid.

The resulting `token.st` file is what you provide to `synthclient`. `synthclient` will use this token for every screening request it makes.

SecureDNA

Supported distributions

SecureDNA provides its software in several different ways:

- As an Ubuntu PPA compiled for:
  - `amd64` (for Ubuntu 22.04 and later) and
  - `arm64` (for Debian Bookworm and later)
- As a Red Hat RPM, compiled for:
  - `amd64` (for CentOS Stream release 9 and later)
- As docker/podman `amd64` images, in two flavors:
  - minimal-size (containing just `synthclient`), or
  - also including certificate-management tools (especially useful for provisioning multiple benchtop synthesizers via scripting)

Thus, you may run `synthclient` either

- as a standalone binary, or
- inside a docker/podman container.

Please note:

- Before running `synthclient`, you must obtain authorization certificates from SecureDNA:
  - The simplest method, suitable for centralized providers or (when necessary) biosafety officers, is to use our web-based form at https://securedna.org/cert-request/.
  - However, benchtop synthesis vendors should have one token per machine, and it's simplest to automate that using simple scripting and the certificate-manipulation binaries. These available either:
    * from the PPA, as shown in the section *Installing the PPA*, or
    * from the RPM, as shown in the section *Installing the RPM*, or
    * via the docker/podman `synthclient-tools` container image, as shown in the section *Installing the container image*.
- Follow the instructions in *Obtaining certificates*.
- Once you have obtained certificates, you may run the `synthclient` binary as detailed in the section *Running synthclient*.

Installing the PPA

These instructions are for Debian distributions, such as Ubuntu.

To run `synthclient` or the certificate-handling tools directly on your machine, without a container, you should download the PPA:

```
$ curl -sSL https://securedna.github.io/ppa/deb/securedna-keyring.gpg | \
  sudo tee /usr/share/keyrings/securedna-keyring.gpg > /dev/null
$ echo "deb [signed-by=/usr/share/keyrings/securedna-keyring.gpg] \
  https://securedna.github.io/ppa/deb ./" | \
  sudo tee /etc/apt/sources.list.d/securedna.list > /dev/null
$ sudo apt update
$ sudo apt install synthclient
```

This will install the `synthclient` binary, as well as several ancillary binaries whose names all start with `sdna` and are used to handle certificate provisioning.

SecureDNA

Installing the RPM

These instructions are for Red Hat distributions, such as CentOS.

To run `synthclient` or the certificate-handling tools directly on your machine, without a container, you should download the RPM:

```
$ sudo rpmkeys --import https://securedna.github.io/ppa/rpm/securedna-keyring.asc
$ sudo yum install -y \
  https://securedna.github.io/ppa/rpm/synthclient-latest.x86_64.rpm
```

Installing the container image

These instructions are suitable when the host software is neither Debian- nor Red Hat-derived, or is some unusual CPu architecture, or the host OS is too old to run the `synthclient` binary directly.

Two container images are available:

- a minimal-size image containing just `synthclient`:
  - `docker pull ghcr.io/securedna/synthclient`
- an image containing both `synthclient` and the tools required to request a certificate and generate authorization tokens:
  - `docker pull ghcr.io/securedna/synthclient-tools`

Where to run `synthclient`

`synthclient` may be run in several places:

- For centralized synthesis providers, `synthclient` may be run either on-premises or in the cloud. More than one copy may be run for load-sharing, failover redundancy, or because the provider is geographically distributed.
- Benchtop manufacturers should run one `synthclient` per installation, which may depend on the deployed architecture:
  - For benchtop synthesizers which have a centralized ”smart” controller on the customer premises, but synthesizers/printers with little or no intelligence or UI, `synthclient` should probably run on the ”smart” controller.
  - If the benchtop *does* have more intelligence than a basic microcontroller, `synthclient` should run inside the benchtop if possible. (Even a Raspberry Pi 4B is plenty fast.)
  - For benchtops which are standalone devices without a per-customer on-site controller, `synthclient` should run inside the benchtop itself.
  - For benchtops which are entirely controlled via some connection from the cloud, `synthclient` should run in the cloud as part of that controller.

Running `synthclient`

You may run `synthclient` as either a standalone binary or as a docker/podman container image. In both cases, note that `synthclient` is designed to run as a persistent server. `synthclient` finds appropriate SecureDNA backend servers on startup, and this process may take a few seconds; connecting to its API during that interval will simply wait, rather than being rejected. After startup, connections are accepted instantly. If running standalone,

you may find that `systemd` is useful for starting the binary, or restarting it after upgrading; similar services exist for docker/podman.

`synthclient` communicates with SecureDNA's servers using TLS (https), but accepts local connections to its API on port 80 (http). If you intend to use `synthclient` in an environment where machines not on `localhost` may communicate with its API, we suggest proxying the connection through a TLS-capable web proxy such as `apache` or `nginx`, using your TLS certificates in your own certificate hierarchy, to protect communications across your local network. SecureDNA can provide example configurations if requested.

Running as a standalone binary

To start `synthclient` by hand, using the binary from the PPA:

```
$ sudo synthclient \
  --token-file token.st \
  --keypair-file token.priv \
  --keypair-passphrase-file token.passphrase &
```

Note that the `token.passphrase` file should contain *passphrase B.* (The contents of the file may end in a newline, because all *leading* and *trailing* whitespace [spaces, tabs, or newlines] will be removed from the passphrase before use.) If the tokens you produce are being provisioned into individual benchtops at customer premises, it is especially important that (a) they all use different passphrases, and (b) none of these passphrases match *passphrase A.* This is because, without taking special precautions, a machine under customer control is subject to a variety of attacks which may be able to read its local filesystem.

Note also that the example above assumes you are running `synthclient` as root, which is necessary if you intend to serve traffic on the default port (80), because binding that port requires elevated privileges. If you don't need to use port 80 (e.g., `synthclient` is having its connections proxied via an https server on port 443, or your internal software can be configured to use an unprivileged port), you can also start `synthclient` using whatever other port you desire, such as:

```
$ synthclient --port 8080 ...
```

Running as a container image

You can always get the latest image via `docker pull ghcr.io/securedna/synthclient`.

You must put the token you have generated from the cert provided by SecureDNA somewhere the image can find it. For the example below, we assume you have put it in `/usr/share/securedna-certificates/`, but the particular directory is your choice.

To start `synthclient`:

```
$ docker run --name synthclient \
  --env SECUREDNA_SYNTHCLIENT_TOKEN_FILE="/certs/token.st" \
  --env SECUREDNA_SYNTHCLIENT_KEYPAIR_FILE="/certs/token.priv" \
  --env SECUREDNA_SYNTHCLIENT_KEYPAIR_PASSPHRASE_FILE="/certs/token.passphrase" \
  --volume /usr/share/securedna-certificates:/certs/:z \
  --detach \
  -p 80:80 \
  ghcr.io/securedna/synthclient \
  ./synthclient
```

Be careful about punctuation in the `--volume` parameter; in particular, ensure that you are using `/certs/` (an absolute path) and not `certs/`; the latter is a relative path and is likely not the location you expect.

Note that this example assumes you are running the container as root. If you are not, then you either need to `sudo` inside the container to run `synthclient` as root, or use `systemd` or some similar tool to start as root, or you should use an unprivileged port, e.g., `--port 8080`.

Connecting to `synthclient`

Once you have `synthclient` running locally, you can talk to it by submitting requests to the API (see the Synthclient API) or using the web interface on the SecureDNA demo page.

Automated use

The primary mode of operation for `synthclient` is automated use: Some portion of your order-processing workflow should make calls on the endpoint described in the Synthclient API, sending it FASTA files to screen, and returning the results to your workflow. If your workflow sees `synthesis_permission: "denied"`, then your customer has asked you to synthesize a hazard for which an appropriate exemption list token has not been provided. Additional fields in the response detail what sorts of hazards were found, and where in the order they occurred.

Using the visualizer

As a lower-volume alternative, designed for human use, `synthclient` includes a built-in web UI endpoint, which takes the normal output from `synthclient` and renders it into a human-readable visualization of where hazards were found in the submitted order. This is called the *demo page,* and can be pointed at any `synthclient` instance. Note that this demo page is simply a connector between your local `synthclient` and a human-comprehensible visualization of its results: Even though your browser initially fetches this page from SecureDNA, the page itself is only rendering results from the URL you provide, and is not sending data anywhere else.

To use the demo page, enter the URL of your `synthclient`, e.g. `http://localhost:1234`. You should see the ”client” and ”database” fields at the top of the page take on the values reported by your `synthclient` instance. Now you can submit FASTA strings and see a visualization of hazard hits when the result is denied. The total light gray strip represents the submitted record; the colored strips represent regions recognized as matching the organisms listed at the bottom. You can hover the mouse over an organism name or accession number to highlight its hit regions. You can also click an organism name to copy its hit regions as a FASTA file.

Rate limits

`synthclient` sends your synthesizer token along with each screening request, and this token authorizes up to a certain number of nucleotides per day to be screened. You pick what that limit is using the `--rate-limit` command-line parameter above. *However*:

- If you generate a token with an absurdly high limit for your business, SecureDNA may deny and/or revoke the token on first use, regardless of the actual number of nucleotides requested.
- If you generate a very low limit, you are likely to exceed it accidentally. However, you can fix this yourself, by generating a new token with a more appropriate limit.
- SecureDNA does not necessarily cap the number of nucleotides screened exactly by the limit you specify; the limit is more properly a hint to abuse-detection automation. In particular, sometimes an order will be

flagged as containing a hazard, and the customer may have to resubmit the order to the provider either with an appropriate exemption list, or without the hazard. (For example, in some cases, an accidental single hit in a large oligo library can simply be dropped with no effect on the overall lab procedure.) When this happens, the customer may immediately resubmit a fixed order to the provider, and occasional screening rejections of this sort typically should not be taken into account as far as potential abuse goes. This means that you need not necessarily overestimate your total rate just to handle the occasional screening rejection; SecureDNA will apply appropriate adjustments based on the incoming requests.

In the unlikely event that SecureDNA detects what appears to be deliberate abuse, such as an attempt to execute a denial-of-service attack or compromise the database, SecureDNA may take one or all of the following actions, at its discretion:

- SecureDNA may return a ”too many requests” sort of error. The rate limit is computed as a window over all requests in the last 24 hours, so exceeding the rate limit may require that you wait until the oldest request in that window has aged out.
- SecureDNA may revoke the synthesizer token corresponding to this order, either temporarily or permanently. While this token is revoked, all attempts to screen using it will be rejected immediately.
- In the event that the token is revoked for obvious abuse, and a new token is generated from the same certificate and (for benchtops) for the same machine serial number, SecureDNA will likely treat the new token as a fresh start. *However*, if the new token is also abused, SecureDNA may conclude that your token-issuance infrastructure is itself compromised, and may revoke your certificate. This will deny you the ability to use *any* tokens created from that certificate. If this occurs, you will have to contact SecureDNA to obtain a new certificate after determining how your certificate private key and passphrase were misused.

This quickstart does not detail the procedure, but advanced providers may obtain and provision their own intermediate certificates (one level up from leaf certificates), thus enabling themselves to have several different trees of certificates. This can enable large providers to handle situations such as:

- A bug in one particular product or company department accidentally tries to screen billions of nucleotides forever
- One particular model of benchtop is compromised and is drafted into a botnet which constantly tries to screen fraudulent orders

In these cases, having subtrees means that when SecureDNA revokes the cert of one set of malfunctioning tokens, the rest of them still function, limiting the effects and allowing easier recovery.