CertStream Developer Guide

Welcome to CertStream! Choose a section from the table of contents below to find the details on how CertStream works!

Table of Contents

Introduction to Certstream

CertStream is an easy-to-deploy Python Script designed for Cybersecurity Researchers. It seamlessly captures newly-registered domains that matches your capture regexes.

The CertStream User Guide acquaints you with the application's functionality, enabling you to maximize its potential.

Key Features:

- Retrieve domains from Certificate Transparency's vast network of monitors.
- Filters for domains of interest with one or more capture regexes.
- Integrates seamlessly with Group IB's Digital Risk Protection Tools.
- Stores domains of interest into a SQLite database.

> 💡 CertStream only requires one command to start. CertStream is user-friendly!

We are confident that CertStream will enhance your efficiency as Cybersecurity Researchers. Enjoy your experience with CertStream! 😃

Setting up & Getting started

Refer to the User Guide.

Design

Architecture

Given below is a quick overview of main components and how they interact with each other.

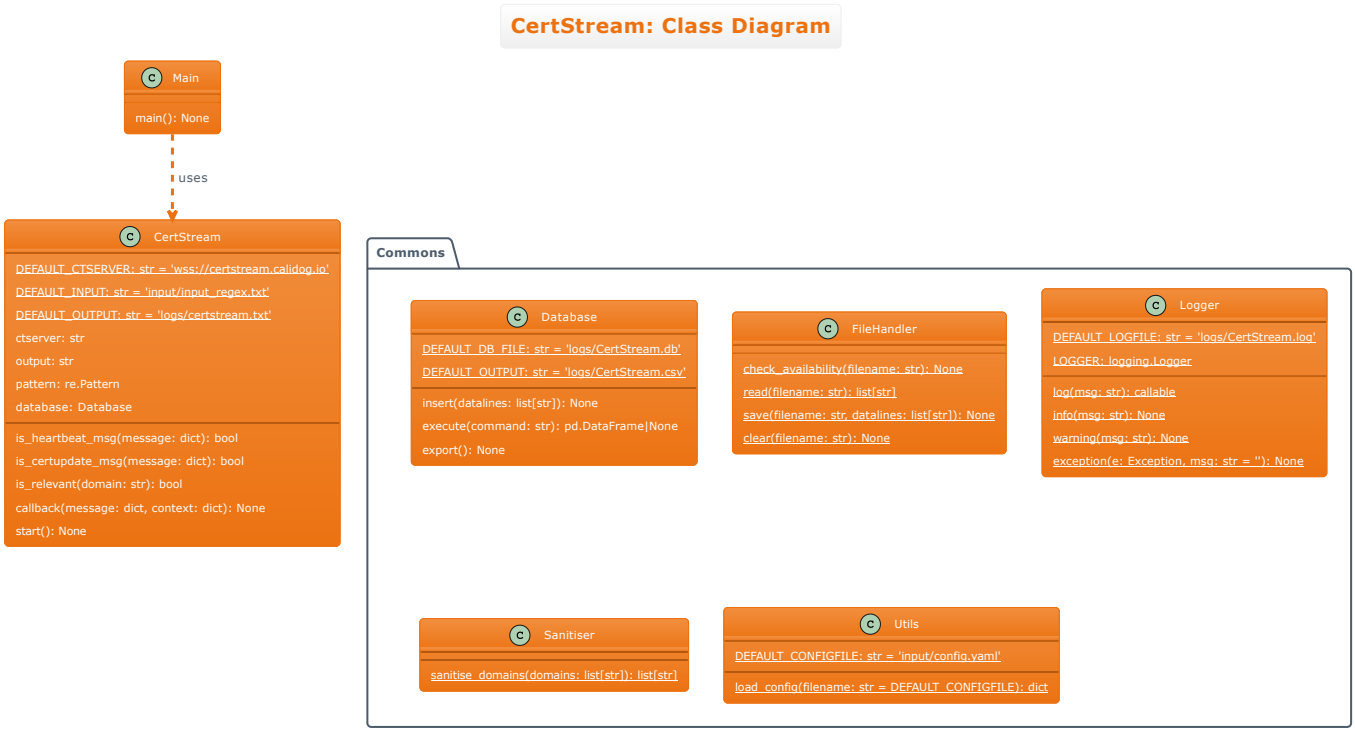
Main components of the architecture

At DomainChecker launch, `Main` calls `CertStream` to listen for Certificate Transparency logs.

`Commons` represents a collection of classes used by multiple other components.

How the architecture components interact with each other

The *Class Diagram* below shows how the components are structured.



The sections below give more details of each component.

CertStream component

API: `CertStream.py`

The CertStream component handles CertStream operations.

The CertStream component:

- uses Long Polling to listen for updates from Certificate Transparent monitor network (aggregated by Calidog's CertStream).
- logs each relevant domain and stores them in a SQLite database.

Design Considerations

Database: Why not use Text File as a simple datastore?

- **Alternative 1 (current choice):** Use a well-implemented database (i.e. SQL database).
    - Pros: Reliable, follows ACID principles. Provides additional features like disallowing duplicates.
    - Cons: Additional overhead and performance cost.

- **Alternative 2:** Use a simple Text/CSV file.
    - Pros: Simple implementation, fewer bugs and higher performance.
    - Cons: Hard to manage data once it is written into the Text/CSV file. Less flexible, unless auxillary code is written.

Common classes

API : `Commons.py`

Classes used by multiple components are in the `Commons` package.

Implementation of Features

This section describes some noteworthy details on how certain features are implemented.

Running CertStream `python3 Main.py`

Implementation

The execution of CertStream is facilitated by `Main`. `Main` sets up the runtime environment (specifically: Change runtime working directory & Initialise logging) & initialises `CertStream`.

Behaviour

Given below is a scenario of how running Main.py behaves at each step.

Step 1. The user first launches CertStream by executing `python3 Main.py` on the Terminal/Command Prompt.

Step 2. `Main` calibrates the working directory to the directory where `Main.py` is located, and initialises `CertStream`.

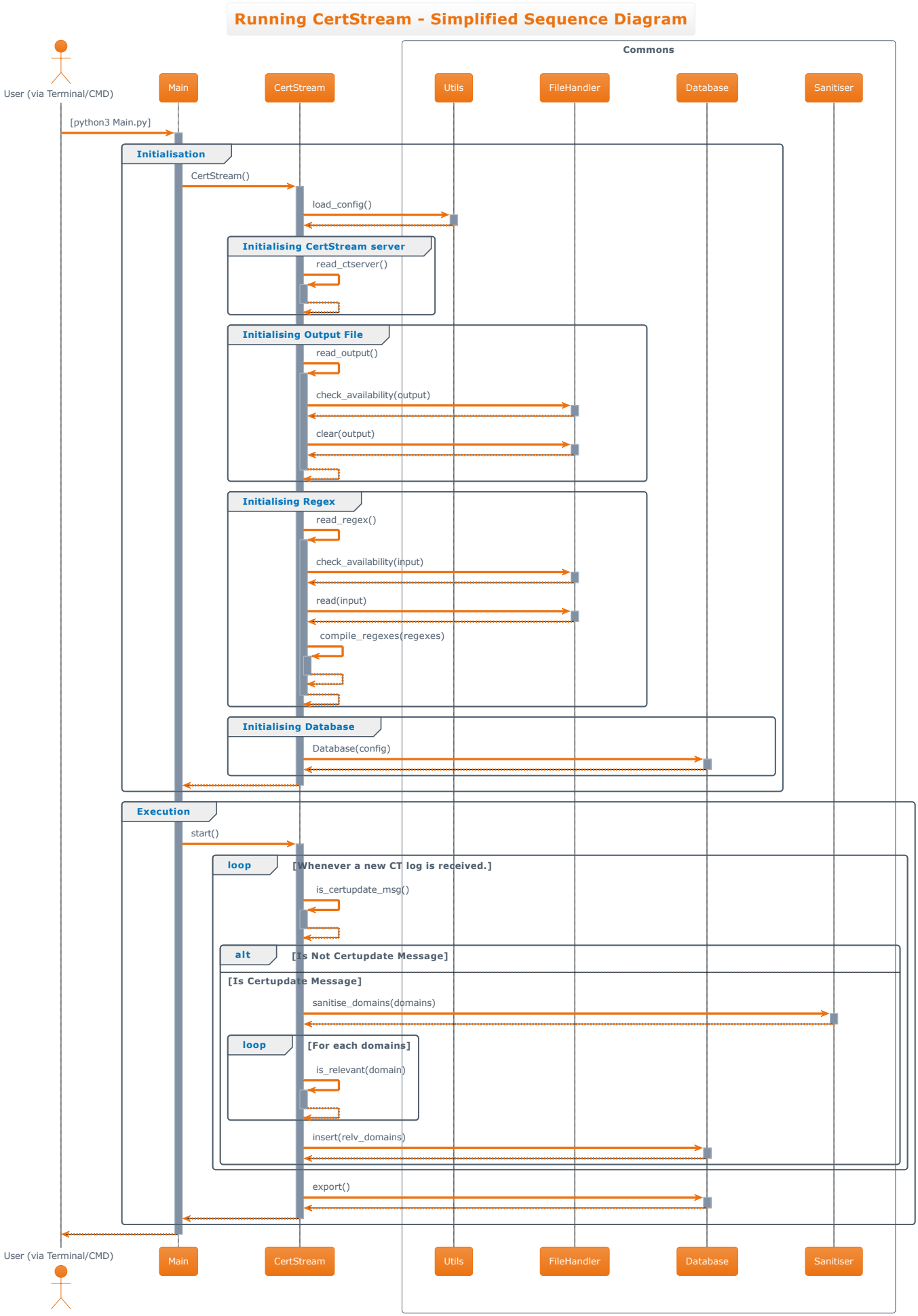
Step 3. `Facade` grabs the configuration fields from the configuration file (default: `config.yaml`), and initialises the Certificate Transparency monitor server, output filename, regexes to monitor and `Database`.

Step 4. `Main` starts Long Polling, by calling `certstream.start()`.

Step 5. For each new CT log, `certstream.callback()` is called. It first verifies if the log received is a Certificate Update message. Then, it sanitises the domains, filters for relevant domains and insert these domains into the database.

Step 6. When the user terminates CertStream (e.g. `CTRL`+`C`), `certstream.start()` will export all stored domains in the database to an output text file. Finally, the program terminates.

The following sequence diagram shows how the CertStream runs:


Running CertStream - Simplified Sequence Diagram
Commons
User (via Terminal/CMD)
Main
CertStream
Utils
FileHandler
Database
Sanitiser
[python3 Main.py]
Initialisation
CertStream()
load_config()
Initialising CertStream server
read_ctserver()
Initialising Output File
read_output()
check_availability(output)
clear(output)
Initialising Regex
read_regex()
check_availability(input)
read(input)
compile_regexes(regexes)
Initialising Database
Database(config)
Execution
start()
loop
[Whenever a new CT log is received.]
is_certupdate_msg()
alt
[Is Not Certupdate Message]
[Is Certupdate Message]
sanitise_domains(domains)
loop
[For each domains]
is_relevant(domain)
insert(relv_domains)
export()
User (via Terminal/CMD)
Main
CertStream
Utils
FileHandler
Database
Sanitiser

Feedback

CertStream is a pilot program. Any feedback is appreciated while we develop CertStream. To deposit ideas and comments, create a new Issue on Github!

Authors

This User Guide is written by Choon Yong.