Configuration & Integration

Introduction

This documentation contains the following solutions:

- The ability to find the categories of a given URL by searching the local database located on the user's terminal device (PC, router, embedded device). In this case, an Internet connection is not required and no network requests are made. This type of database is called the local database.
- The ability to find categories of a given URL using HTTPS access to the SafeDNS cloud. To find a category of domains an internet connection is required and a username/password is required as well to get access to the library. But there is no need to have a local database on the user's computer.
- The ability to save URL categories received from the SafeyDNS cloud in the local disk cache for faster access to them later. The data saved in the disk cache is not deleted after restarting the computer and restarting the application using this cache.
- The ability to cache the received categories of a URL in the computer's RAM to speed up their subsequent search. The cache sizes can be set during the library assembling.
- The ability to receive and install regular updates of the local database categories from the SafeDNS cloud. Downloading the entire database is available: all categories ~ 1.6GB; smaller custom builds are also possible.
- The performance of the library on average hardware is approximately 60k requests per second.

The solution is implemented in the programming language "C". It's provided with the user interface in the form of functions of the "C" programming language. The solution can also be integrated into a Python project.

SDK integration into the final product

The SDK is easy enough to integrate into the final software product developed in the C or C++ programming language. It is also possible to integrate it into the final Python solution.

To use the library, you need to get the library sources from repository here, configure the library, build and install it.

Configuring the url2cat library

Necessary utilities to assemble and configure the library:

- **cmake** (version 3.15 or higher)
- **make** (version 4.2 or higher)
- **gcc**
- **sphinx** (version 1.8.5 or higher) (module sphinx_rtd_theme)

Necessary libraries (Ubuntu 18 or higher):

- **openssl-1.1.1s (or 3.0.6)**
- **libssl-dev**
- **pkg-config**
- **libssl-dev**

Configuration

1. Go to the library source directory

2. Configure the library using the command

```
cmake -S . -B build -DCMAKE_BUILD_TYPE=Release -DURL2CAT_SERVER=yes -DURL2CAT_DATABASE=yes -DURL2CAT_LIBRARY=static -DURL2CAT_LOCALE=en
```

Or edit according to your needs and run the bash script located in `liburl2cat` directory - `create_build.sh`.

To run the script perform:

```
sudo chmod +x create_build.sh
./create_build.sh
```

Using the following cmake commands you can configure library functions:

- `URL2CAT_SERVER` - request to the SafeDNS server (yes, no)
- `URL2CAT_DATABASE` - request to the local database (yes, no)
- `URL2CAT_LIBRARY` - library assembling type (static & shared)
- `URL2CAT_MAX_NUMBER_CATEGORY` - number of defined categories (Default number: 5)
- `URL2CAT_HASH_TYPE` - Hash type (MD5, SHA256, SHA512. Default type - MD5)
- `URL2CAT_HASH_LEN`- Hash length (full - hash is not truncated before searching in the database. Half extract the first 8 bytes of the hash. Default length - half)
- `URL2CAT_CACHE` - category entry cache size (dynamic / static)

3. Build the project using the command:

```
cmake --build build
```

4. Copy the following library files to your project:

- `build_release/lib/liburl2cat.a` or `build_release/lib/liburl2cat.so`
- `include/url2cat.h`

Using the library in the project:

Before using the library, needs to initialize it using the `s_url2cat_setting` structure. The structure has the following fields:

- `cache_size` - cache size in bytes (if set to 0 the cache is not used)
- `db_name` - the name of the database
- `db_download_scheme` - protocol for updating the database
- `db_download_host` - database update host
- `db_download_port` - database update port
- `db_download_path` - database update path
- `db_download_user` - login to updating the database
- `db_download_password` - password to updating the database
- `server_scheme` - protocol for connecting to the SafeDNS server
- `server_host` - host of connection to the SafeDNS server
- `server_port` - port for connecting to the SafeDNS server
- `server_path` - path to connect to the SafeDNS server
- `server_user` - login to connect to the SafeDNS server
- `server_password` - password to connecting to the SafeDNS server

For initialization use the function - `url2cat_init(s_url2cat_setting * setting)`

To get a category use the function:
`url2cat_category(char * url, size_t len_url, s_url2cat_category * * category, size_t * number_category)`

where the structure s_url2cat_category has the following fields:

- `type` – the number of category
- `type_name` – the name of category

After finishing using the library you will need to deinitialize it with the command: `url2cat_deinit()`

The database can be updated while the library is running using the command:
`url2cat_database_update(s_url2cat_setting * setting)`

A request for recategorization can be sent while the library is running using the command:
`url2cat_recategory(char * url, size_t url_size, char * category_name, size_t category_name_size)`

Examples of integration the solution into simple projects running on "C" can be found in the /example directory.

Revision #11
Created 22 August 2022 11:44:18
Updated 10 February 2023 04:20:35