Requirements for the Implemetation of the Verification Service

Ursa#1337

Contents

1 Hosting

This bot is able to be hosted on any operating system that has python 3 installed. Installation instructions for each technology are included to assist in handoff for new maintainers of this project. This document outlines the requirements for each technology; the reliant deliverables so to speak.

Remote hosting If a remote hosting solution is chosen, there should be an allowance for an:

- SMTP Server (Optional)
- MariaDB or MySQL Server
- Python3 Installation

In the default iteration of this project, a *Gmail* account is used to send emails; no functionality is included for reading emails. Keeping with this trend will not require a server to be configured, and the email account can be handed over to the next maintainer with relative ease. A MariaDB server is used, however any database which uses MySQL syntax can be utilized in it's place. The database and tables will be illustrated in the respective section. Finally, Python3 is used for the creation in the project because Python2 has been sunsetted at the start of 2020, and the APIs used in this project are available in the maintained version used (*3.8*).

2 User Permissions

A new user should be created on the host for security purposes, regardless of whether it's remote or local. Every opportunity has been taken to create the most secure application possible, and every additional step should also be taken by the end user to make their own system secure as well. An additional user, **harlequin**, should be created on the host with minimal privileges; a home directory isn't required.

3 Database

User and Database When the host is setup initially, the mysql and mariadb services should be installed for the database. After these services are verified to be working, A database named **harlequin** should be created. In turn, a database user (also **harlequin**) with a different password from the system user should be created to access this database. The permissions should allow all privileges to this database only, since giving access to any other database is insecure.

Auth Table The first table to be created is the **auth** table. Any name is sufficient, but will require changes to the program. Below is an illustration of the auth table: The first column, *email*, is variable up to 100 characters, required

Field	Type	NULL	Key	Default
email	VARCHAR(100)	NO	PRIMARY	NULL
username	VARCHAR(100)	NO	UNIQUE	NULL
code	CHAR(6)	NO		NULL
expiry	DATETIME	NO		NULL

Figure 1: Auth's columns and their respective properties

not to be null, and is the primary key for this table. All input is sanitized beforehand so that only proper emails can be included, and any attempt at inserting a record where this field passes the maximum allotted length will be denied. The second column, *username*, takes a discord username up to 100 characters, and is taken directly from an API. This field is sanitized in the case that any user attempts remote code execution or SQL injection by changing their username. In turn, the *code* column is populated internally by a variable-length pseudo-randomly generated password, so no injection-based attacks can occur on this parameter; a possible bruteforce attack could occur with a paltry range of ~12.5 million possible passwords, therefore the *expiry* column exists to deter those from attempting such a task. The expiry date is set to 24 hours from initial verification request as to deter bruteforcing methods of authentication; 100 requests per second would take around 33 hours, as reference. This column, along with the code column, are subject to change as security requirements increase.

Verified Table The **verified** table hosts the email addresses and usernames of those already verified. This table will be checked before any new verification process to ensure that no two users are tied to the same email address. The attributes of each column are listed below:

Field	Type	NULL	Key	Default
email	VARCHAR(100)	NO	PRIMARY	NULL
username	VARCHAR(100)	NO	UNIQUE	NULL

Figure 2: Verified's columns and their respective properties

4 Environment Variables

The constants used in execution aren't stored in the program, they're stored in an external *.env* file in the same directory; this allows the new maintainer to easily change email accounts, guild IDs, and other vital pieces of information that are used to run the bot. Again, it should be reiterated that *no passwords should be the same between services.* Names in the environment file are descriptive and succinct, but additional information will be given in the project's code documentation file as well. The final piece of the puzzle deals with bot permissions on the server; the only permission it requires, and the only permission that should be allocated to it, is the **'Manage Roles'** permission.

This document will be present alongside the code documentation, the main program and a template *.env* file for use. If any questions, comments, or suggestions arise, feel free to contact **Ursa#1337** on discord or by **email**. Other points of contact are **Bennie Duncan** (WGU Cyber Club President) and **Justin Sherley** (WGU Cyber Club Discord Administrator).