Systems Programming Project 2

April 11, 2024

1 Hashassin

Now that we have some basics of hashing, etc. down, we are going to make a full blown rainbow table implementation.

You can accept the GitHub classroom assignment at `https://classroom.github.com/a/nZLy9P4v`.

Assignment Project Exam Help

This is a group project with a maximum of *three* students per group.

https://powcoder.com

2 Instructions

Add WeChat powcoder

For this project, you will be extending your solution from project 1 to include a full blown implementation of a rainbow table and corresponding password cracker.

As with Project 1, your binary crate should be in a directory called `cli` and it should compile to a binary named `hashassin`.

Your library crate should be in a directory called `core` and it should be available called `hashassin-core` and usable by other crates via `use hashassin_core::Whatever`.

Your CLI will will support several additional commands: 1) `gen-rainbow-table`, 2) `crack`, and 3) `server`.

`gen-rainbow-table` will generate a rainbow table with chains starting from a list of preexisting passwords. `gen-rainbow-table` *must* have several options that can be set:

1. `--num-links`, which specifies the number of links generated chains should have. `--num-links` must be greater than zero and should have a default value of 5.
2. `--threads`, the number of threads to use when generating the rainbow table (format specified below).

3. `--out-path`, which is where the generated rainbow table should be saved.
4. `--threads`, the number of threads to use to generate passwords. This value *must* be greater than zero and the maximum value should be the maximum length of an array on whatever system the program is being run on. `--threads` must have a default value of 1.
5. `--password-length`, which is number of characters in the passwords for this rainbow table. `--password-length` must be greater than zero and have a default value of 4.
6. `--algorithm`, the hashing algorithm that should be used for this rainbow table. `--algorithm` should have a default value of md5 (i.e., the default hashing algorithm should be md5).

By default, the valid character set for passwords is the same as Project 1: valid ASCII, both caps and lower case, and including punctuation and spaces but *not* including non-printable characters like newline or tabs or bells.

The on disk format for your rainbow table must (by default) have the start and end points of each chain, **separated by a tab**, with one chain per line.

For example, a rainbow table for 8 character passwords using md5:

```
Vty.whm5\tC;hLO*Fg
P@3h?~Jz\tm1.s4hE&
6_wUZNF\thATA!Kpk
'm~BMxT\t-FV}|[ka
Dni_2/TU\tZBFUpjZw
v~y^2i$e\tYy(7.I44
EO<QE~b$\tcfY^MEf]
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

where

```
\t
```

is the ASCII tab character.

NB: You can add additional functionality that can restrict or expand the set of characters to be used in generated passwords, but by default, generated passwords should adhere to the above guidelines.

`crack` will, given a hash, produce the password corresponding to that hash using a pre-computed rainbow table. `crack` *must* have several options that can be set:

1. `--rainbow-table`, which specifies the path to read the rainbow table from.
2. `--out-path`, which, *if present*, will write the output of the command to the specified file, with one pair of hash *hex encoded* and corresponding password separated by the tab character, per line,. If not present, results should be written to `stdout`

3. `--threads`, the number of threads to use to crack a password. This value *must* be greater than zero and the maximum value should be the maximum length of an array on whatever system the program is being run on. `--threads` must have a default value of 1.

4. `--algorithm`, the hashing algorithm to use. `--algorithm` must have a default value of md5.

5. `--num-links`, which specifies the number of links generated chains should have. `--num-links` must be greater than zero and should have a default value of 5.

6. `--password-length`, which is number of characters in the passwords for this rainbow table. `--password-length` must be greater than zero and have a default value of 4.

7. `--in-path`, which is a path to a set of hashes to crack.. The input path *must* (by default) have hashes stored as contiguous bytes. **NB: This is somewhat different than project 1 where we stored hashes one per line.**

You are free to add other options and functionality, but whatever you add should not be required to be set by a user. I.e., it should either be optional or have sensible defaults.

By default, the valid character set for passwords is the same as Project 1: valid ASCII, both caps and lower case, and including punctuation and spaces but *not* including non-printable characters like newline or tabs or bells.

Assignment Project Exam Help

Output should be *hex encoded* hash and its corresponding password, separated by a tab character, with one pair per line.

https://powcoder.com

If there were no passwords found for the input hashes, then `crack` should *error* out with an error message saying "No passwords found."

Add WeChat powcoder

NB: `crack` should *not* panic if there is no password found. It should *error* out.

`server` will provide a multi-user, interactive version of `crack`. `server` *must* have several options that can be set:

1. `--host`, which sets the IP address the server should listen on. Host must be a valid IP address and should default to `127.0.0.1`.

2. `--port`, which sets the port the server should listen on. Host must be a valid port and should default to `4515`.

3. `--rainbow-table`, which is a path to a rainbow table the server will be using.

4. `--algorithm`, which is the algorithm that should be used for hashing and should default to md5.

5. `--password-length`, which must be greater than zero and corresponds to the length of passwords.

6. `--threads`, the number of threads to use for commands run by users. This value *must* be greater than zero and the maximum value should be the maximum length of an array on whatever system the program is being run on. `--threads` must have a default value of 1.

Additionally, `server` should allow clients to interact with it in the following way:

1. `CRACK hash` where `hash` is a hex encoded. This command should produce the associated password for the given hex encoded hash, in the same way that the `crack` command line does. If no password is found the user should be returned the message "Password not found."

`server` *must* allow multiple clients to connect and interact at the same time!

3 Grading

You will receive points according to the following list. Please note that there are more than 100 points available. Also please note that some items on the list are *required* to be implemented by groups with at least one graduate student and do not grant additional points (undergrad only groups can complete these for additional points).

- -1,000 points: If you do not update `CREDITS.md` with the names of your group members, as well as an honest break down of the work each group member did, you will receive **NEGATIVE ONE THOUSAND** points. I.e., it would be next to impossible to get anything higher than a zero on this project.

Assignment Project Exam Help

- -1,000 points: If running `cargo fmt` results in any change to your repository, you will get `NEGATIVE ONE THOUSAND` points. Be sure to run `cargo fmt`!!!!!!!!!

https://powcoder.com

- 25 points: Program compiles, as well as the items noted below as required for graduate groups.

Add WeChat powcoder

- 15 points: **All** required functionality of `gen-rainbow-table` is implemented (e.g., setting `--threads` actually uses multiple threads, etc.)
- 15 points: **All** required functionality of `crack` is implemented (e.g., setting `--threads` actually uses multiple threads, etc.).
- 15 points: **All** required functionality of `server` is implemented (e.g., setting `--threads` actually uses multiple threads, etc.).
- 5 points: Comprehensive documentation. Points will be determined by looking at the documentation built when running `cargo doc --document-private-items --no-deps`.
- 10 points: No warnings from `cargo check` with no use of directives that would suppress default warnings (e.g., `#[allow(dead_code)]`)
- 10 points: No warnings from `cargo clippy` with no use of directives that would suppress default warnings (e.g., `#[allow(dead_code)]`)
- 5 points: **REQUIRED FOR GRADUATE GROUPS.** Support Sha256
- 0.25 points: Support additional hashing algorithms besides md5 (0.25 points per algorithm)

- 5 points: **REQUIRED FOR GRADUATE GROUPS.** You support passwords that are longer than 128 bits. I.e., you must have a reduction function that operates on arbitrary precision integers instead of just primitive types (e.g., `u128`).
- 5 points: **REQUIRED FOR GRADUATE GROUPS.** Proper error handling. Create and use proper error types in library code and appropriate handling in any user code (e.g., CLI).
- 5 points: No unwraps or expects. Add `#![deny(clippy::unwrap_used, clippy::expect_used)]` to the top of your `main.rs` and `lib.rs`. If running `cargo clippy` doesn't result in any errors, then you get the 5 points.
- 5 points: Add proper logging. Using the `tracing` crate to generate logs. Points will be determined on comprehensiveness of logs, as well as the usage of different log levels.
- 10 points: **REQUIRED FOR GRADUATE GROUPS.** Performance report. Create a report that shows the performance characteristics of your program. E.g., how does it scale with respect to length of passwords? What performance differences are there between md5 and sha256 (and any other algorithms you might test)? How does threading affect performance? This report is not expected to be massive, but it should explain your experimental setup and have plots. The report is not required to conform to any particular formatting, but please try to make it look decent.

Assignment Project Exam Help

- 10 points: Make `server` handle networking using async. Only the networking component needs to be async, and in fact you probably want to make sure that any actual work is being done on a thread where blocking is ok (check out `https://docs.rs/tokio/latest/tokio/task/fn.spawn_blocking.html`).

https://powcoder.com

- Unlimited points: Cool factor. Points are determined subjectively by me, but if I find anything about your project to be particularly unique, difficult, clever, etc., you can get some extra points. Be sure to point out anything we should pay special attention to in your `README.md` file.

Add WeChat powcoder

In addition to the above, you must have a README.md to summarize what you did and give any and all instructions on how to run things! I'm going to go off what's in the README.md to grade, so make sure you tell us everything you did and test all the instructions, etc.

3.1 Allowed Crates

Generally speaking, you are free to use any crates listed on `https://blessed.rs`.

You can (and probably should) also use the `hex` crate (`https://docs.rs/hex/latest/hex/index.html`) as well as the `num` crate (`https://docs.rs/num/latest/num/`).

You can also use the `async-channel` crate (`https://docs.rs/async-channel/latest/async_channel/`) if you are going to go for the async `server` points.

Any additional crates not on Blessed.rs need to be approved by Jeremy.

Be sure to note what crates you use in your `README.md`!

3.2 Due Date

April 30th 11:59PM

NB: Write access to your repositories will be revoked at the deadline!

3.3 Academic Honesty Statement

In addition to the code in your repository, you *must* include an academic honesty statement as per the class syllabus. If you do not include this academic honesty statement in your repository, you will receive a *ZERO* on the project.

REMEMBER THAT IF YOU USE A MACHINE LEARNING TOOL THAT IS CHEATING, YOU WILL RECEIVE A ZERO ON THE PROJECT (i.e., your max grade in the class will be a 66%, which is an F), AND THE CLASS WILL REVERT TO IN CLASS TESTS AND QUIZZES!!!!!

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder