

Permissions Manager

<https://github.com/bennettyardley/permissions-manager>

Bennett Yardley

The Problem

You are a web developer that needs temporary access to the `/var/www/html` folder so you can make a change to a file that your current account only has read access to.

You contact the IT department and they give you the username and password to an account that has read and write access to the `/var/www/html` folder.

After you make the changes you need, you are supposed to logout of the account, however you realize it would be much easier to continue using the elevated account.

Now if an attacker breaches your system they have the elevated permissions.

The Solution

Instead of requiring usernames and passwords to be shared my program will assign certain permissions directly to the user's account and remove them after they are no longer needed.

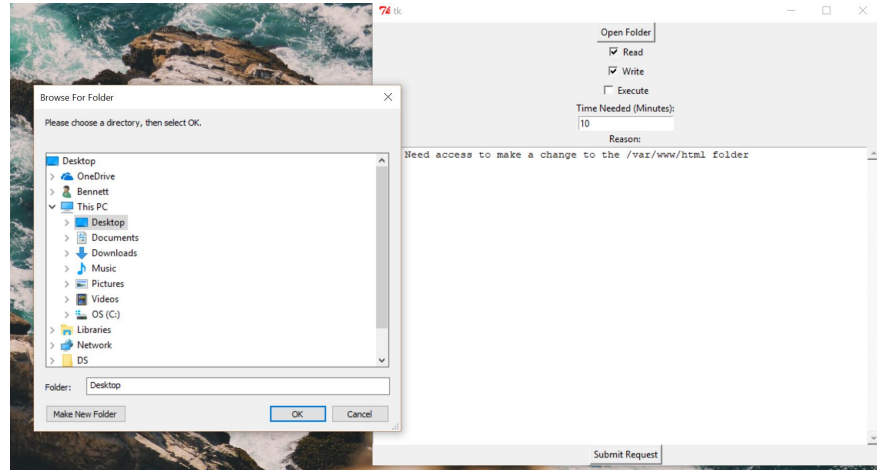
Prerequisites

1. Linux and Windows Compatible
2. No additional dependencies
3. Python

Client

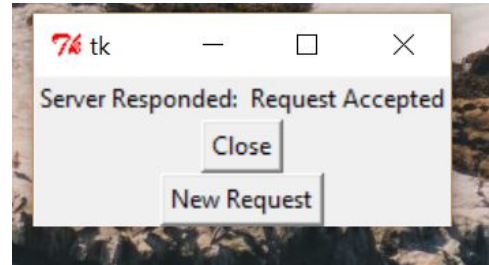
GUI - TkInter

- File Browser
- Text Boxes
- Check Boxes
- Sends to the server and waits for a response



CLI

- Manual Input
- Sends to the server and waits for a response



Server

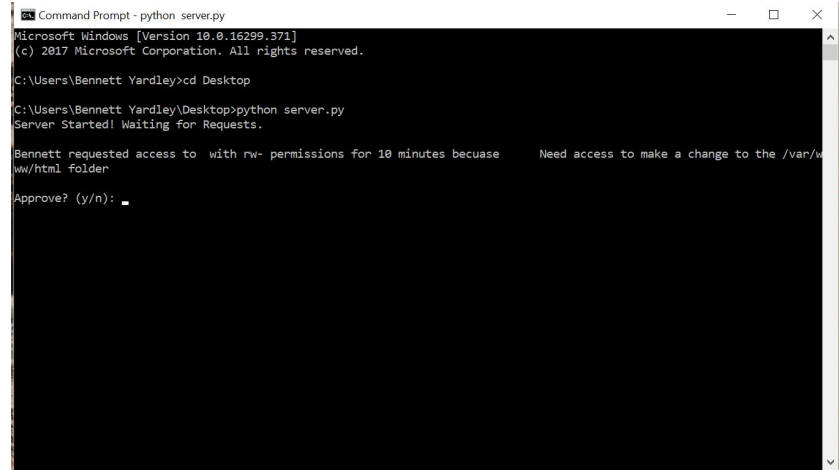
Creates the Database

Listens for a request for a permission change

Responds in two ways:

- Approve: Tells the client, makes the change in permission, and eventually reverts the permission
- Deny: Tells the client the reason

Deletes the row once it is finished



```
Command Prompt - python server.py
Microsoft Windows [Version 10.0.16299.371]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Bennett Yardley>cd Desktop
C:\Users\Bennett Yardley\Desktop>python server.py
Server Started! Waiting for Requests.

Bennett requested access to  with rw- permissions for 10 minutes because  Need access to make a change to the /var/www/html folder

Approve? (y/n): _
```

Database

How the server and client communicate

Columns:

- Folder that client needs access to
- Level of permission client needs
- Duration of time that client needs access
- Username of the client
- Reason for access
- ID number
- Status of request

Fun Stuff

Error Checking - Valid Inputs (Time/File Structure)

Permission Commands

- Linux:
 1. `setfacl -m u:USERNAME:PERMS -R DIRECTORY`
 2. `setfacl -x u:USERNAME:PERMS -R DIRECTORY`
- Windows:
 1. `icacls "PATH" /grant:r USERNAME:PERMS`
 2. `icacls "PATH" /deny:r USERNAME:PERMS`

`threading.Timer`

Future Challenges

Currently the computers have to have direct access to each other (same file system or network file access)

Creating an authorization system that could only be run if the server approves

Brings up challenges in security and requiring admin access

MySQL would need to be used because SQLite3 only works locally

Would require a dependency

Ability to create more specific read and write permissions for client/server

Use PyInstaller to compile an executable for Windows and Linux

