PY: NAT port knocking script with authentication

template login.html

login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f5f5f5;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .container {
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
            text-align: center;
        }
        h2 {
            margin-bottom: 20px;
        }
        .logo {
            max-width: 100px;
            margin-bottom: 20px;
        }
        form {
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        label {
            margin-bottom: 5px;
        }
        input[type="text"],
        input[type="password"] {
            width: 100%;
```

```
            padding: 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
            box-sizing: border-box;
        }
        input[type="submit"] {
            width: 100%;
            padding: 10px;
            background-color: #007bff;
            color: #fff;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }
        input[type="submit"]:hover {
            background-color: #0056b3;
        }
        .error {
            color: red;
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Login</h2>
        {% if error %}
        <p class="error">{{ narzedzia:error }}</p>
        {% endif %}
        <form action="/" method="post">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password">
            <input type="submit" value="Login">
        </form>
    </div>
</body>
</html>
```

Code of the program itself

auth.py

```
from flask import Flask, request, render_template
import paramiko
```

```
import time
import re
# MikroTik API credentials
ROUTER_IP = '192.168.1.1'
USERNAME = 'admin'
PASSWORD = 'PASS'

app = Flask(__name__)

# Configure Flask to trust X-Forwarded-For header
app.config['TRUSTED_PROXIES'] = '127.0.0.1'

def remove_port(ip_address_with_port):
    return re.split(r'[;,|:]',ip_address_with_port)

# Function to add IP address to the specified list with a timeout
def add_to_list(ip_address_arg):
    ip_address = remove_port(ip_address_arg)[0]
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(ROUTER_IP,port=22, username=USERNAME,
password=PASSWORD)

    # Send command to add IP address to address list
    command = f"/ip firewall address-list add list=port_knocking_stage1
address={ip_address} timeout=12h"
    stdin, stdout, stderr = ssh.exec_command(command)

    # Wait for the command to execute
    time.sleep(1)

    # Check for any errors
    if stderr.read().decode():
        print("Error:", stderr.read().decode())
    else:
        print("IP address added successfully. "+ip_address)

    ssh.close()
# Dummy database for demonstration (replace with your own
authentication mechanism)
users = {
    'admin': 'pass',
}

# Authentication route
@app.route('/', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
```

```
        if username in users and users[username] == password:
            user_ip = request.headers.get('X-Forwarded-For',
request.remote_addr)
            add_to_list(user_ip)
            add_to_list(user_ip)
            success_message = 'Authenticated successfully! Your IP
address <strong>{}</strong> has been added to the whitelist for 12
hours.'.format(remove_port(user_ip)[0])
            return '<div style="font-family: Arial, sans-serif; text-
align: center; margin-top: 50px;"><h2 style="color:
#4CAF50;">Success!</h2><p>{}</p></div>'.format(success_message)
        else:
            error = 'Invalid credentials. Please try again.'
    return render_template('login.html', error=error)

if __name__ == '__main__':
    app.run(debug=False)
```