Single URL configuration

Configuration of single URL

Install python

yum install nginx python3 python3-pip

pip3 install flask

Now make directory

mkdir /zimbra-login

Now go in to the dir

cd /zimbra-login/

List the dir

ls

python3 -m venv venv

source venv/bin/activate

pip3 install flask requests

Make dir.

mkdir static

mv single.conf /root/

Unzip the tar file

tar xvfz single_url.tar.gz

ls

cd zimbra-login/

ls

Copy the app.py to zimbra-login

cp app.py /zimbra-login/

cp -pvr static templates /zimbra-login/

cd /zimbra-login/

cp app.py app.py.org

cat app.py

—----------

[root@jalindia zimbra-login]# cat app.py

from flask import Flask, request, jsonify, redirect, render_template

import requests

from xml.etree import ElementTree as ET

```
import logging

import urllib.parse

import threading

import time


app = Flask(__name__)


# Set up logging

logging.basicConfig(level=logging.DEBUG)


# Optionally suppress InsecureRequestWarning

requests.packages.urllib3.disable_warnings(requests.packages.urllib3.exceptions.InsecureRequestWarning)


# Global variables for user lists

ne_users = []

ose_users = []


# Thread-safe lock to protect access to the user lists

user_list_lock = threading.Lock()


# Load users from text files
```

```
def load_users(file_path):
    with open(file_path, 'r') as f:
        return [line.strip() for line in f.readlines()]

# Function to refresh the user lists every 10 minutes
def refresh_user_lists():
    global ne_users, ose_users
    while True:
        with user_list_lock:
            ne_users = load_users('/zimbra-login/ne_users.txt')  # List of users for ne
            ose_users = load_users('/zimbra-login/ose_users.txt')  # List of users for ose
            logging.debug("User lists reloaded.")
        time.sleep(600)  # Sleep for 10 minutes (600 seconds)

# Start the background thread for refreshing user lists
refresh_thread = threading.Thread(target=refresh_user_lists)
refresh_thread.daemon = True  # Daemonize thread to exit when the main program exits
refresh_thread.start()

@app.route('/')
```

```
def index():
    return render_template('login.html')


@app.route('/zimbra_auth', methods=['POST'])
def zimbra_auth():
    try:
        xml_data = request.data.decode('utf-8')


        if not xml_data:
            return jsonify({"error": "XML payload is required"}), 400


        # Extract email from the XML data to determine the server
        email_start = xml_data.find("<account by=\"name\">") +
len("<account by=\"name\">")
        email_end = xml_data.find("</account>")
        email = xml_data[email_start:email_end]


        # Access user lists with thread-safe locking
        with user_list_lock:
            if email in ne_users:
                url = "https://mta2.jalindia.co.in/service/soap"
            elif email in ose_users:
```

```
            url = "https://mta3.jalindia.co.in/service/soap"
        else:
            return jsonify({"status": "failed", "error": "User not found on any server"}), 400

    headers = {
        'Content-Type': 'text/xml; charset=utf-8',
        'SOAPAction': ''
    }

    logging.debug(f"Sending XML data to {url}")
    response = requests.post(url, data=xml_data, headers=headers, verify=False)

    logging.debug(f"Response Status Code: {response.status_code}")
    logging.debug(f"Response Text: {response.text}")

    if response.status_code == 200:
        response_xml = response.text
        root = ET.fromstring(response_xml)
        auth_token = root.find('.//{urn:zimbraAccount}authToken').text
```

```
        # URL encode auth_token
        encoded_auth_token = urllib.parse.quote(auth_token)

        if email in ne_users:
            zimbra_webmail_url =
f"https://mta2.jalindia.co.in/service/preauth?isredirect=1&autht
oken={encoded_auth_token}"
        else:
            zimbra_webmail_url =
f"https://mta3.jalindia.co.in/service/preauth?isredirect=1&autht
oken={encoded_auth_token}"

        logging.debug(f"Redirect URL: {zimbra_webmail_url}")

        # Return redirect URL as JSON
        return jsonify({"status": "success", "redirect_url":
zimbra_webmail_url})
    else:
        return jsonify({"status": "failed", "error":
response.text}), response.status_code

  except Exception as e:
    logging.error(f"Exception: {str(e)}")
```

```
        return jsonify({"error": str(e)}), 500


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)


[root@jalindia zimbra-login]#
```

—----------

vi login.html

—----------

```
[root@jalindia templates]# cat login.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Zimbra Login</title>
    <style>
        /* Resetting default styles */
```

```
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: Arial, sans-serif;
            background: url('/static/image.png') no-repeat center
center fixed;
            background-size: cover;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        .container {
            background-color: rgba(255, 255, 255, 0.9); /* White with
transparency */
            padding: 2em;
            border-radius: 8px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
```

```
    max-width: 400px;
    width: 100%;
    text-align: center;
}

.container h2 {
    color: #333;
    margin-bottom: 20px;
}

.logo {
    width: 120px;
    height: auto;
    margin-bottom: 20px;
}

.form-group {
    margin-bottom: 15px;
    text-align: left;
}

.form-group label {
```

```
    font-size: 1rem;
    color: #333;
}

.form-group input {
    width: 100%;
    padding: 10px;
    font-size: 1rem;
    border: 1px solid #ccc;
    border-radius: 4px;
    margin-top: 5px;
}

.btn {
    width: 100%;
    padding: 10px;
    font-size: 1rem;
    color: #fff;
    background-color: #007bff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
```

```
            transition: background-color 0.3s ease;
        }

        .btn:hover {
            background-color: #0056b3;
        }

        #response {
            margin-top: 15px;
            font-size: 1rem;
        }
    </style>
</head>
<body>
    <div class="container">
        <!-- Logo Section -->
        <img src="/static/test.jpg" alt="CRPF Logo" class="logo">

        <h2>Zimbra Login</h2>
        <form id="loginForm">
            <div class="form-group">
                <label for="username">Email Address:</label>
```

```
            <input type="text" id="username" name="username"
required>
        </div>
        <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" id="password"
name="password" required>
        </div>
        <button type="submit" class="btn">Login</button>
    </form>
    <div id="response"></div>
  </div>

  <script>

document.getElementById('loginForm').addEventListener('sub
mit', function(event) {
        event.preventDefault();

        const username =
document.getElementById('username').value;
        const password =
document.getElementById('password').value;
```

```
        fetch('/zimbra_auth', {
            method: 'POST',
            headers: {
                'Content-Type': 'text/xml',
            },
            body: `<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
                <soap:Body>
                    <AuthRequest xmlns="urn:zimbraAccount">
                        <account
by="name">${username}</account>
                        <password>${password}</password>
                    </AuthRequest>
                </soap:Body>
            </soap:Envelope>`
        })
        .then(response => response.json())
        .then(data => {
            if (data.status === 'failed') {
                document.getElementById('response').innerHTML
= 'Login failed: ' + data.error;
                document.getElementById('response').style.color =
'red';
```

```
            } else {
                // Redirect to Zimbra URL with token
                window.location.href = data.redirect_url;
            }
        })
        .catch(error => {
            console.error('Error:', error);
            document.getElementById('response').innerHTML =
'An error occurred: ' + error;
            document.getElementById('response').style.color =
'red';
        });
    });
  </script>
</body>
</html>
```

[root@jalindia templates]#

—--

Go into tmux

Run python script

Come out from tmux

Ctrl + b then d

vi ne_ldap-import.sh

Make two files for NE and NE2

mv ldap-import.sh ne-ldap-import.sh

cp ne-ldap-import.sh ne2-ldap-import.sh

vi ne2-ldap-import.sh