SERVER TYPE PREDICTION SCRIPT - DOCUMENTATION

This script is designed to perform server type prediction based on running applications and open ports using PowerShell commands and Paramiko library for SSH connections. It includes functions for self-assessment on the local machine and checking a range of IP addresses for server type prediction.

Prerequisites

Python 3.x

Paramiko library (pip install paramiko)

Code Structure

The script consists of the following functions:

self_assess(): Performs server type prediction on the local machine.

check_range_ip(start_ip, end_ip, username, password): Performs server type prediction for a range of IP addresses.

connection(hostname, username, password): Establishes an SSH connection to a given hostname and retrieves running applications and open ports using PowerShell commands.

predict_server_type(running_applications, open_ports): Predicts the server type based on the running applications and open ports.

Function Details

self_assess()

This function executes PowerShell commands on the local machine to gather information about the operating system version, running processes, and open TCP connections. It then calls the predict_server_type function with the obtained data to predict the server type. The predicted server type is returned as the output.

```
def self_assess():
    cmd="powershell -Command [System.Environment]::OSVersion.VersionString"
    os=subprocess.run(cmd, shell=True, capture_output=True, text=True).stdout

    cmd = "powershell -Command Get-Process"
    proc = subprocess.run(cmd, shell=True, capture_output=True, text=True).stdout

    command = "Get-NetTCPConnection | Select-Object
LocalAddress,LocalPort,RemoteAddress,RemotePort,State"
    open_ports = subprocess.run(['powershell.exe', '-Command', command],
capture_output=True, text=True).stdout

    proc1={}
    proc=proc.split("\n")
    for i in range(3,len(proc)-3):
        t=proc[i].split()
        proc1[t[len(t)-1]]=1
    proc=list(proc1.keys())


    splitted = open_ports.split('\n')
    open_ = []
    for port in splitted:
        if 'LocalPort' in port:
            open_.append(port.split(':')[1])


    predicted = predict_server_type(proc,open_)
    return predicted
```

check_range_ip(start_ip,end_ip,username,password)

This function takes a range of IP addresses, a username, and a password as input. It iterates over the IP addresses within the specified range and establishes an SSH connection to each IP using the connection function. It retrieves the running applications and open ports for each IP and calls the predict_server_type function to predict the server type. The prediction results for each IP address are stored in a dictionary and returned as the output.

```
def check_range_ip(start_ip,end_ip,username,password):
    ip_range = ipaddress.summarize_address_range(ipaddress.IPv4Address(start_ip),
ipaddress.IPv4Address(end_ip))
    ips_within_range = []

    for ip_network in ip_range:
        ips_within_range.extend(str(ip) for ip in ip_network)

    result_data = {}

    for ip in ips_within_range:
        try:
            results = connection(ip,username,password)
            prediction = predict_server_type(results[1],results[2])
            result_data[ip] = prediction
        except:
            pass
    return result_data
```

connection(hostname, username, password)

This function establishes an SSH connection to a given hostname using the Paramiko library. It executes PowerShell commands to retrieve the operating system version, running processes, and open TCP connections. The obtained data is parsed and returned as a list containing the operating system version, running processes, and open ports. Running applications (Index 1 of result list) and open ports (Index 2 of result list) should be used as parameters for predict_server_type function.

```
def connection(hostname,username,password):

    client=paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    client.connect(hostname,username=username,password=password)

    stdin,stdout,stderr=client.exec_command('powershell -Command
\"[System.Environment]::OSVersion.VersionString\"')
    os=stdout.read().decode().splitlines()

    stdin,stdout,stderr=client.exec_command('powershell -Command \"Get-
Process\"')
    proc=stdout.read().decode().splitlines()
```

```
    stdin,stdout,stderr=client.exec_command('powershell -Command \"Get-
NetTCPConnection | Select-Object
LocalAddress,LocalPort,RemoteAddress,RemotePort,State\"')
    open_ports=stdout.read().decode().splitlines()
    open_ = []
    for port in open_ports:
        if 'LocalPort' in port:
            open_.append(port.split(':')[1])


    client.close()
    return [os,proc,open_]
```

predict_server_type(running_applications, open_ports)

This function takes a list of running applications and a list of open ports as input. It compares the running applications with predefined mappings of applications to server types. It also checks if any of the open ports match the predefined port lists associated with server types. Based on this analysis, it predicts the server type. The predicted server type is returned as the output.

This will output the predicted server type based on the provided running applications and open ports, as well as the predicted server type for the local machine based on self-assessment.

```
def predict_server_type(running_applications, open_ports):
    application_server_mapping = {
        "web": [" 80", " 443"],
        "database": [" 3306", " 5432"],
        "file_server": [" 21", " 22"],
        "mail": [" 25", " 143", " 110"],
    }


    apps_application = []

    apps =
{('nginx','apache','tomcat'):"web",('mysql','postgresql','mongodb'):"database",("
filezilla","samba", "nfs", "ftp"):"file_server",("postfix", "exim",
"sendmail"):"Mail"}
```

```
    for app in running_applications:
        app = app.lower()
        for possible in apps:
            for app_ in possible:
                if app_ in app:
                    apps_application.append(apps[possible])




    apps_port = []

    for application, ports in application_server_mapping.items():
        if set(ports).intersection(set(open_ports)):
            apps_port.append(application)

    result = [apps_port,apps_application]

    result_percentages = []

    result[0].extend(result[1])
    percentage_set = set(result[0])

    for i in percentage_set:
        result_percentages.append((i,(result[0].count(i)/len(result[0]))*100))

    sorted_result_percentages = sorted(result_percentages, key = lambda x:x[1],
reverse = True)


    return sorted_result_percentages if len(result_percentages) > 0 else
"generic_server"
```

Limitations

The script assumes that the target machines are running Windows operating system and have PowerShell installed.

The script relies on predefined mappings of applications to server types and predefined port lists associated with server types. If a new application or port needs to be considered, the mappings and port lists should be updated accordingly in the predict_server_type function.

The script assumes that the SSH connection to the target machines can be established using the provided username and password. If SSH key-based authentication is required, the script needs to be modified to accommodate that.

The script does not handle errors or exceptions in a comprehensive manner. It may need additional error handling to enhance its robustness in real-world scenarios.

Conclusion

The server prediction script contains multiple functions for different tasks, including assessing the local computer, assessing a range of IP addresses, and predicting possible server types given a list of applications and open_ports, each returning a list or dictionary with required data.