Citrix Automated Reporting

Code to create automated citrix acceptance reports

```
import requests
import json
from requests.auth import HTTPBasicAuth

# Citrix VPX credentials and Nitro API endpoint
citrix_vpx_url = 'https://<citrix-vpx-ip>/nitro/v1/config/'
username = '<your-username>'
password = '<your-password>'
auth = HTTPBasicAuth(username, password)

# Disable SSL warnings for self-signed certificates
requests.packages.urllib3.disable_warnings()

# Function to get data from the Nitro API
def get_data_from_vpx(endpoint):
    url = citrix_vpx_url + endpoint
    response = requests.get(url, auth=auth, verify=False)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Failed to fetch {endpoint}: {response.status_code}")
        return None

# Fetch system-level details
def get_system_report():
    system_info = get_data_from_vpx('nsconfig')
    if system_info:
        ns_ip_address = system_info['nsconfig'][0].get('IPAddress', 'N/A')
        hostname = system_info['nsconfig'][0].get('hostname', 'N/A')
        ha_status = system_info['nsconfig'][0].get('hacurstatus', 'N/A')
        print(f"System IP Address: {ns_ip_address}")
        print(f"Hostname: {hostname}")
        print(f"HA Status: {ha_status}\n")
```

```
# Fetch and classify backend health status for virtual servers, also calculate health score
def get_load_balancer_report():
    lb_vservers = get_data_from_vpx('lbvserver')
    if lb_vservers:
        print(f"Load Balancing Virtual Servers:\n{'-'*30}")

        # Separate servers by health status
        perfect_servers = []
        degraded_servers = []
        down_servers = []

        # Counters for health score calculation
        total_servers = 0
        perfect_count = 0
        degraded_count = 0

        for vserver in lb_vservers['lbvserver']:
            name = vserver.get('name', 'N/A')
            ip = vserver.get('ipv46', 'N/A')
            state = vserver.get('curstate', 'N/A')
            total_servers += 1

            # Fetch backend service health status
            services_bound = get_data_from_vpx(f'lbvserver_service_binding/{name}')
            if services_bound:
                backend_health = {'perfect': True, 'degraded': False}
                for service in services_bound.get('lbvserver_service_binding', []):
                    service_name = service.get('servicename', 'N/A')
                    service_health = get_data_from_vpx(f'service/{service_name}')
                    if service_health:
                        health_status = service_health['service'][0].get('svrstate', 'N/A')
                        if health_status == 'DOWN':
                            backend_health['perfect'] = False
                            backend_health['degraded'] = True
                        elif health_status != 'UP':
                            backend_health['perfect'] = False

                if backend_health['perfect']:
                    perfect_servers.append(f"Name: {name} | IP: {ip} | State: {state}")
                    perfect_count += 1
                elif backend_health['degraded']:
```

```
                    degraded_servers.append(f"Name: {name} | IP: {ip} | State: {state}")
                    degraded_count += 1
                else:
                    down_servers.append(f"Name: {name} | IP: {ip} | State: {state}")
            else:
                down_servers.append(f"Name: {name} | IP: {ip} | State: {state}")

        # Print out categorized servers
        print("Perfect (Green) Servers:\n" + "-" * 30)
        for server in perfect_servers:
            print(server)
        print("\nDegraded (Yellow) Servers:\n" + "-" * 30)
        for server in degraded_servers:
            print(server)
        print("\nDown (Red) Servers:\n" + "-" * 30)
        for server in down_servers:
            print(server)
        print('\n')

        # Calculate overall health score
        total_degraded = degraded_count * 0.5
        total_health_contrib = perfect_count + total_degraded
        overall_health_score = (total_health_contrib / total_servers) * 100 if total_servers >
0 else 0

        print(f"Overall Health Score: {overall_health_score:.2f}%\n")

# Fetch SSL certificate details, including expiration dates
def get_ssl_report():
    ssl_certificates = get_data_from_vpx('sslcertkey')
    if ssl_certificates:
        print(f"SSL Certificates:\n{'-'*30}")
        for cert in ssl_certificates['sslcertkey']:
            cert_name = cert.get('certkey', 'N/A')
            cert_alias = cert.get('cert', 'N/A')
            exp_date = cert.get('expirymonitor', 'N/A')  # This field might vary depending on
your Citrix version
            print(f"Cert Name: {cert_name} | Alias: {cert_alias} | Expiry Date: {exp_date}")
        print('\n')

# Fetch high availability details
```

```
def get_ha_report():
    ha_status = get_data_from_vpx('hanode')
    if ha_status:
        print(f"High Availability Details:\n{'-'*30}")
        for node in ha_status['hanode']:
            node_id = node.get('id', 'N/A')
            node_state = node.get('state', 'N/A')
            print(f"Node ID: {node_id} | State: {node_state}")
        print('\n')

# Main function to generate the report
def generate_report():
    print("Citrix VPX High-Level Report\n")
    get_system_report()
    get_load_balancer_report()
    get_ssl_report()
    get_ha_report()

if __name__ == '__main__':
    generate_report()
```

Revision #2
Created 20 December 2024 08:15:00 by Mesut Bayrak
Updated 20 December 2024 09:48:54 by Mesut Bayrak