Gmail marsupial seven <marsupialseven@gmail.com>

# bootcon:: auto_nmapscan_rev9

1 message

**marsupial seven** <marsupialseven@gmail.com> Mon, Jan 13, 2025 at 10:47 PM
To: marsupial seven <marsupialseven@gmail.com>

```
import subprocess
import json
import time
from ipaddress import ip_network

# Function to run the simple Nmap scan
def run_simple_nmap_scan(target_ip):
 try:
 print(f"Running simple Nmap scan on {target_ip}...")

 # Nmap command for scanning the most common ports
 nmap_command = ["nmap", "-T4", target_ip]

 result = subprocess.run(nmap_command, capture_output=True, text=True)

 if result.returncode == 0:
 print("Nmap scan completed successfully:")
 print(result.stdout)
 return result.stdout
 else:
 print("Error in scanning:", result.stderr)
 return None

 except Exception as e:
 print(f"An error occurred: {e}")
 return None

# Function to run the enhanced Nmap scan
def run_enhanced_nmap_scan(target_ip):
 try:
 print(f"Running enhanced Nmap scan on {target_ip}...")

 # Nmap command for:
```

```
    # 1. Scanning all ports (-p-)
    # 2. Service version detection (-sV)
    # 3. OS detection (-O)
    nmap_command = ["nmap", "-p-", "-sV", "-O", target_ip]

    result = subprocess.run(nmap_command, capture_output=True, text=True)

    if result.returncode == 0:
    print("Nmap scan completed successfully:")
    print(result.stdout)
    return result.stdout
    else:
    print("Error in scanning:", result.stderr)
    return None

    except Exception as e:
    print(f"An error occurred: {e}")
    return None

# Function to run the aggressive Nmap scan
def run_aggressive_nmap_scan(target_ip):
    try:
    print(f"Running aggressive Nmap scan on {target_ip}...")

    # Nmap command for:
    # 1. Aggressive scan (-A), which includes OS, version, script scanning, and traceroute
    nmap_command = ["nmap", "-A", target_ip]

    result = subprocess.run(nmap_command, capture_output=True, text=True)

    if result.returncode == 0:
    print("Nmap scan completed successfully:")
    print(result.stdout)
    return result.stdout
    else:
    print("Error in scanning:", result.stderr)
    return None

    except Exception as e:
    print(f"An error occurred: {e}")
```

```
 return None

# Function to determine the local network
def get_local_network():
 try:
 print("Determining local network...")
 # Run ip a to get the local IP and network range
 result = subprocess.run(["ip", "a"], capture_output=True, text=True)
 if result.returncode == 0:
 ip_info = result.stdout
 print("Local IP Information:\n", ip_info)
 return ip_info
 else:
 print("Error retrieving IP information:", result.stderr)
 return None
 except Exception as e:
 print(f"An error occurred while getting local network: {e}")
 return None

# Function to parse Nmap results and extract services/ports
def parse_nmap_for_services(nmap_output):
 services = []
 lines = nmap_output.splitlines()
 for line in lines:
 if "open" in line: # A line with open ports typically contains the word "open"
 parts = line.split()
 port = parts[0].split("/")[0] # Extract port number
 service_name = parts[2] # Extract service name
 version = None
 # Check if version information is available
 if len(parts) > 3 and '(' in parts[3]:
 version = parts[3].strip('()')
 services.append((port, service_name, version))
 return services

# Function to load the CVE data from the JSON file
def load_cve_data():
 try:
 with open("cve_vuln_data.json", "r") as file:
 return json.load(file)
```

```
 except Exception as e:
 print(f"Error loading CVE data: {e}")
 return []

# Function to check and display CVEs based on service name
def display_cve_data_for_service(service_name, cve_data, report_file):
 report_file.write(f"Checking CVEs for service: {service_name}\n")
 for entry in cve_data:
 if service_name.lower() in entry["service"].lower():
 report_file.write(f"CVE ID: {entry['cve_id']}\n")
 report_file.write(f"Description: {entry['description']}\n")
 report_file.write("-" * 50 + "\n")

# Main driver function
def main():
 # Determine local network information
 local_network = get_local_network()

 # Open the report file for writing
 with open("scan_report.txt", "w") as report_file:
 if local_network:
 # Ask if user wants to scan the entire network or a specific host
 scan_choice = input("Do you want to scan the entire network (y/n)? ")

 if scan_choice.lower() == 'y':
 # Get the network range (you may extract this automatically based on ip a)
 #target_ip = "192.168.1.0/24" # You can automate this based on local network
detection
 target_ip = "192.168.47.132/24"
 else:
 target_ip = input("Enter the specific IP address to scan: ")

 # Provide options for scan types
 print("\nSelect the type of Nmap scan:")
 print("1. Simple Nmap scan")
 print("2. Enhanced Nmap scan (with version and OS detection)")
 print("3. Aggressive Nmap scan (includes script scanning and traceroute)")

 scan_type = input("Enter your choice (1, 2, or 3): ")
```

```
    if scan_type == '1':
    scan_results = run_simple_nmap_scan(target_ip)
    elif scan_type == '2':
    scan_results = run_enhanced_nmap_scan(target_ip)
    elif scan_type == '3':
    scan_results = run_aggressive_nmap_scan(target_ip)
    else:
    print("Invalid choice. Exiting...")
    return

    if scan_results:
    # Write the Nmap scan results to the report file
    report_file.write(f"Nmap scan results for {target_ip}:\n")
    report_file.write(scan_results)
    report_file.write("\n" + "="*50 + "\n")

    # Load CVE data
    cve_data = load_cve_data()

    # Parse Nmap results to detect services
    services = parse_nmap_for_services(scan_results)

    # Display CVEs related to the detected services and write them to the report file
    for port, service, version in services:
    display_cve_data_for_service(service, cve_data, report_file)

    # Sleep to respect API rate limits
    time.sleep(2)

if __name__ == "__main__":
    main()
```