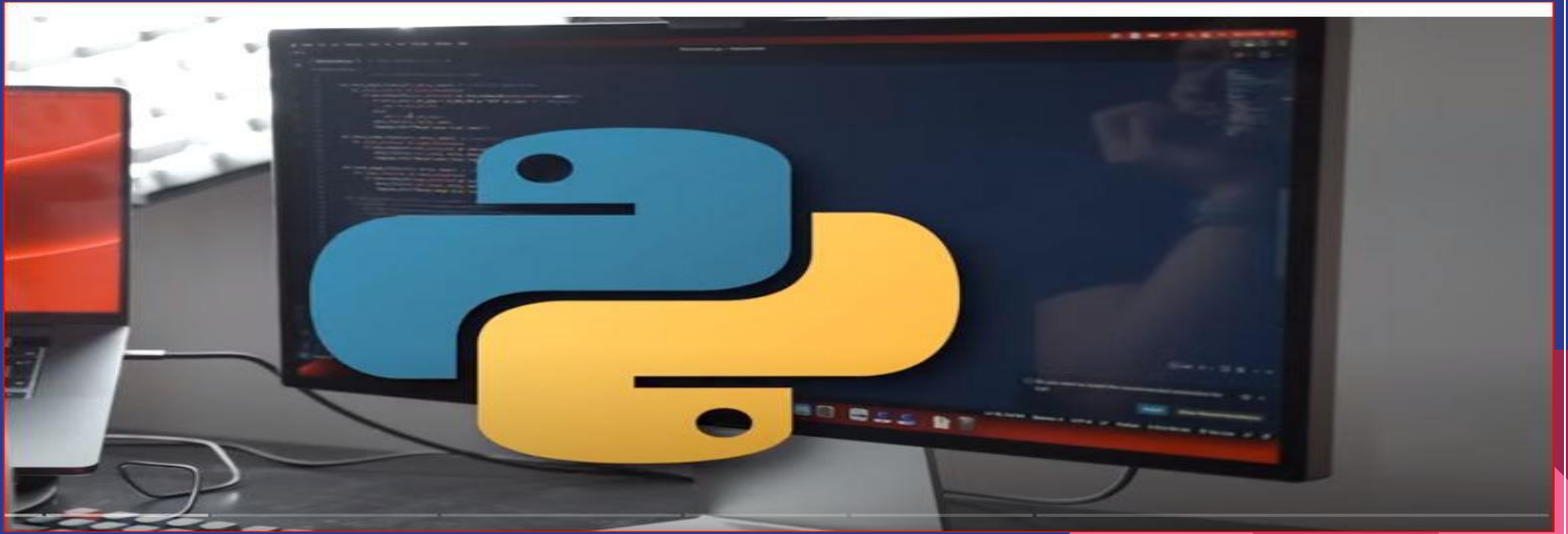
Project 4 BootCon Presentation

Automating Nmap Scans with Python

Subba, Abib *Dec 14, 2023*



Automating Nmap Scans with Python (selected the topic)

- Efficient, relevant, and practical for network security professionals, enhancing exploration and vulnerability identification in computer networks.
- Powerful scripting for customized, targeted Nmap scans, enhancing network security.
- Python's versatility integrates Nmap with diverse tools, creating a holistic network security strategy.
- Automating Nmap scans educates on scripting's role in cybersecurity.
- Showcase of automation's power in simplifying tasks and enhancing cybersecurity.
- Python doesn't just scan, it also makes sense of the results.

Concept Applied For

Networking Concepts:

- Utilize Nmap to discover and analyze network topology.
- Identify open ports, services, and potential vulnerabilities within the network.

Cryptographic Concepts:

- Emphasize secure communication during Nmap scans.
- Understand encryption protocols when dealing with sensitive data.

Security Concepts:

- Apply Nmap to enhance security processes and identify weaknesses.
- Discuss ethical considerations: proper authorization and adherence to legal standards.

Research Steps Taken

- **Define Objectives:** Clearly outline the goals and outcomes for automating Nmap scans in network security.
- **Research Nmap Automation Tools:** Explore existing Python libraries or tools for Nmap automation.
- **Develop Python Script:** Create a robust script with error handling, considering network issues and Nmap errors.
- **Testing and Validation:** Rigorously test the script across varied network scenarios, validating results against expectations.
- **Documentation:** Emphasize clear documentation and comments for enhanced understanding, maintenance, and collaboration.
- **Skill Empowerment:** Design training materials to empower network security professionals with the knowledge and skills for effective automation.
- **Feedback and Iteration:** Gather feedback, iterate on the script, and refine the automation process based on real-world testing.

Preview: A Practical Demonstration

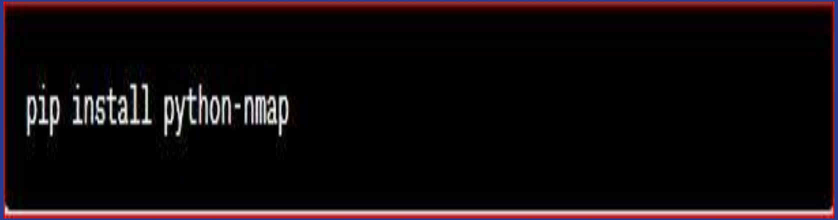
- Quick overview of Nmap and its significance in network scanning.
- Introduction to Python and its role in automating Nmap scans.
- Basic Nmap Commands and Python Interaction
- Understand how to customize Nmap scans using Python to cater to specific project requirements.
- Discussing essential security considerations when automating Nmap scans.
- Providing insights into best practices for secure scripting and risk mitigation.
- Applying the gained knowledge to real-world scenarios, such as cybersecurity assessments, network administration, and vulnerability management.
- Encouraging participants to share their experiences and insights.

Live demonstration

- Execute Python script to extend Nmap's functionality.
- Showcase scanning capabilities.
- Explain each step as you progress through the demonstration.

Download & Install Nmap & Python

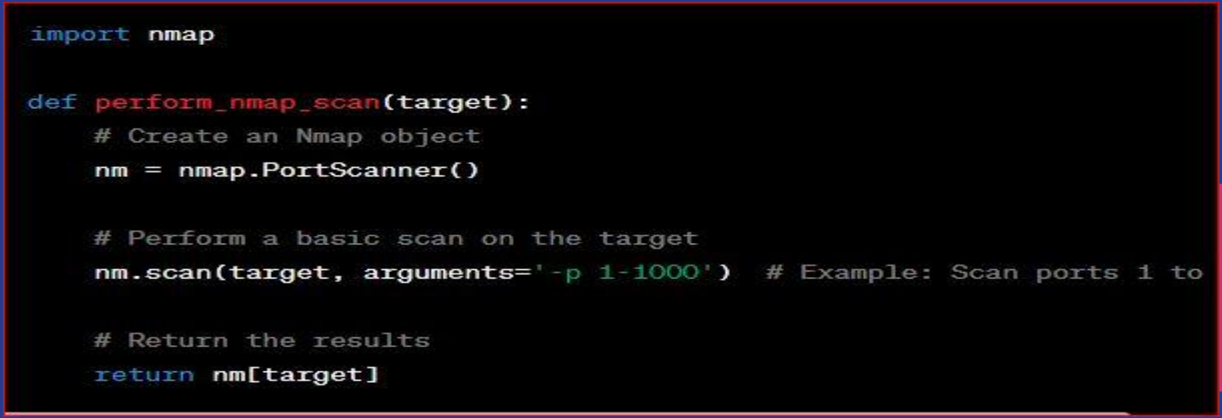
- Download and install `python-nmap` module from the official website: https://nmap.org/download.html.
- Download and Install Python official website: https://www.python.org/downloads/.
- Alternatively, use a package manager like `apt` for Linux or Homebrew for macOS to install Nmap.
- Open a terminal or command prompt and install the `python-nmap` module using `pip`:



```
pip install python-nmap
```

Nmap Port Scan

- Importing Libraries:
    - `import nmap`: This line imports the `python-nmap` library, which allows us to interact with Nmap functionality in Python.
- `perform_nmap_scan` Function:
    - `nm = nmap.PortScanner()` : This creates an instance of the Nmap PortScanner class, which will be used to perform scans.
    - `nm.scan(target, arguments='-p 1-1000')` : This line initiates a basic Nmap scan on the specified target IP address, scanning ports 1 to 1000.
    - `return nm[target]` : The function returns detailed scan results for the specified target.

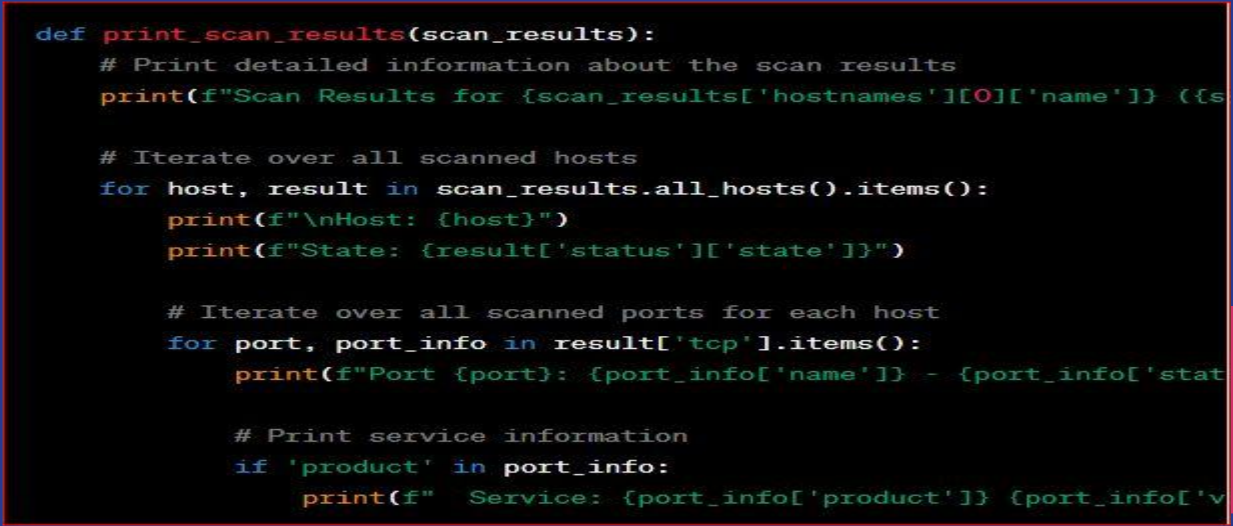


```python
import nmap

def perform_nmap_scan(target):
    # Create an Nmap object
    nm = nmap.PortScanner()

    # Perform a basic scan on the target
    nm.scan(target, arguments='-p 1-1000')  # Example: Scan ports 1 to

    # Return the results
    return nm[target]
```

Nmap Scan Result

`print_scan_results` Function:

- This function takes the detailed scan results as an argument and prints human-readable information.
- `print(f"Scan Results for {scan_results['hostnames'][0]['name']} ({scan_results['addresses']['ipv4']})")`: Prints the target's hostname and IP address.

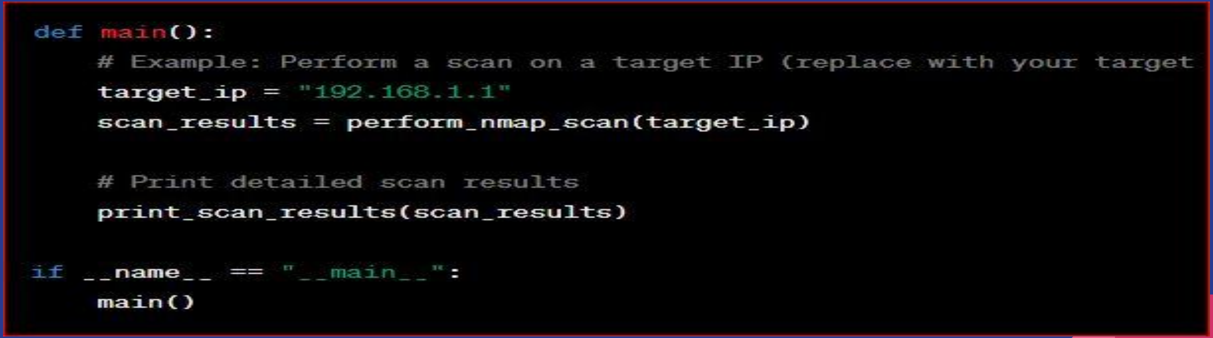


```
def print_scan_results(scan_results):
    # Print detailed information about the scan results
    print(f"Scan Results for {scan_results['hostnames'][0]['name']} ({s

    # Iterate over all scanned hosts
    for host, result in scan_results.all_hosts().items():
        print(f"\nHost: {host}")
        print(f"State: {result['status']['state']}")

        # Iterate over all scanned ports for each host
        for port, port_info in result['tcp'].items():
            print(f"Port {port}: {port_info['name']} - {port_info['stat

            # Print service information
            if 'product' in port_info:
                print(f"  Service: {port_info['product']} {port_info['v
```

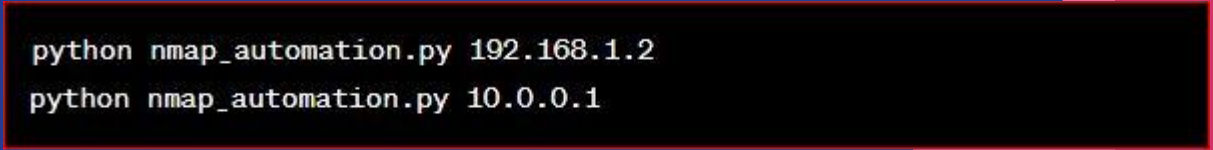
A basic scan on a target IP

`main` Function:

- `target_ip = "192.168.1.1"`: Replace this with the target IP address you want to scan.
- `scan_results = perform_nmap_scan(target_ip)`: Calls the `perform_nmap_scan` function to get detailed scan results for the specified target.
- `print_scan_results(scan_results)`: Prints the detailed scan results.



```
def main():
    # Example: Perform a scan on a target IP (replace with your target
    target_ip = "192.168.1.1"
    scan_results = perform_nmap_scan(target_ip)

    # Print detailed scan results
    print_scan_results(scan_results)

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

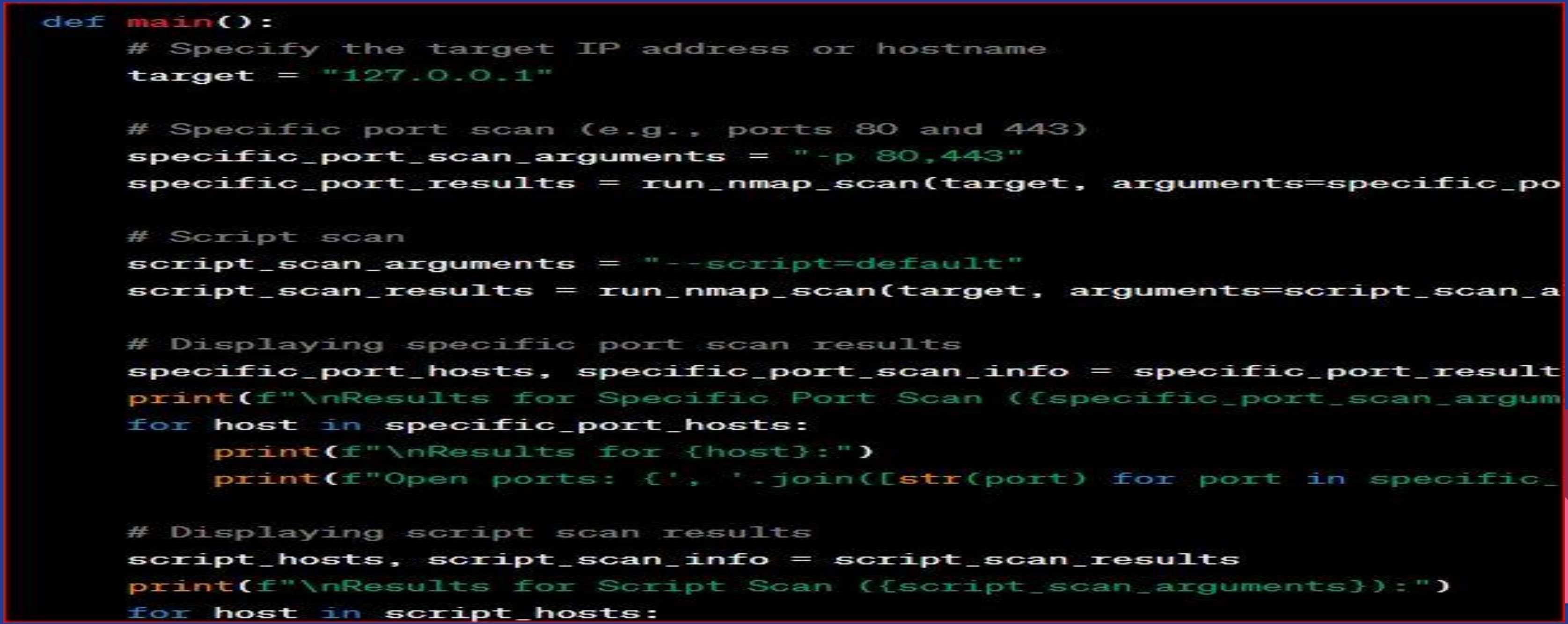


```
python nmap_automation.py 192.168.1.2
python nmap_automation.py 10.0.0.1
```

Main Function: The results of the script scan, including hosts and script scan outputs.

The `main` function performs the following steps:

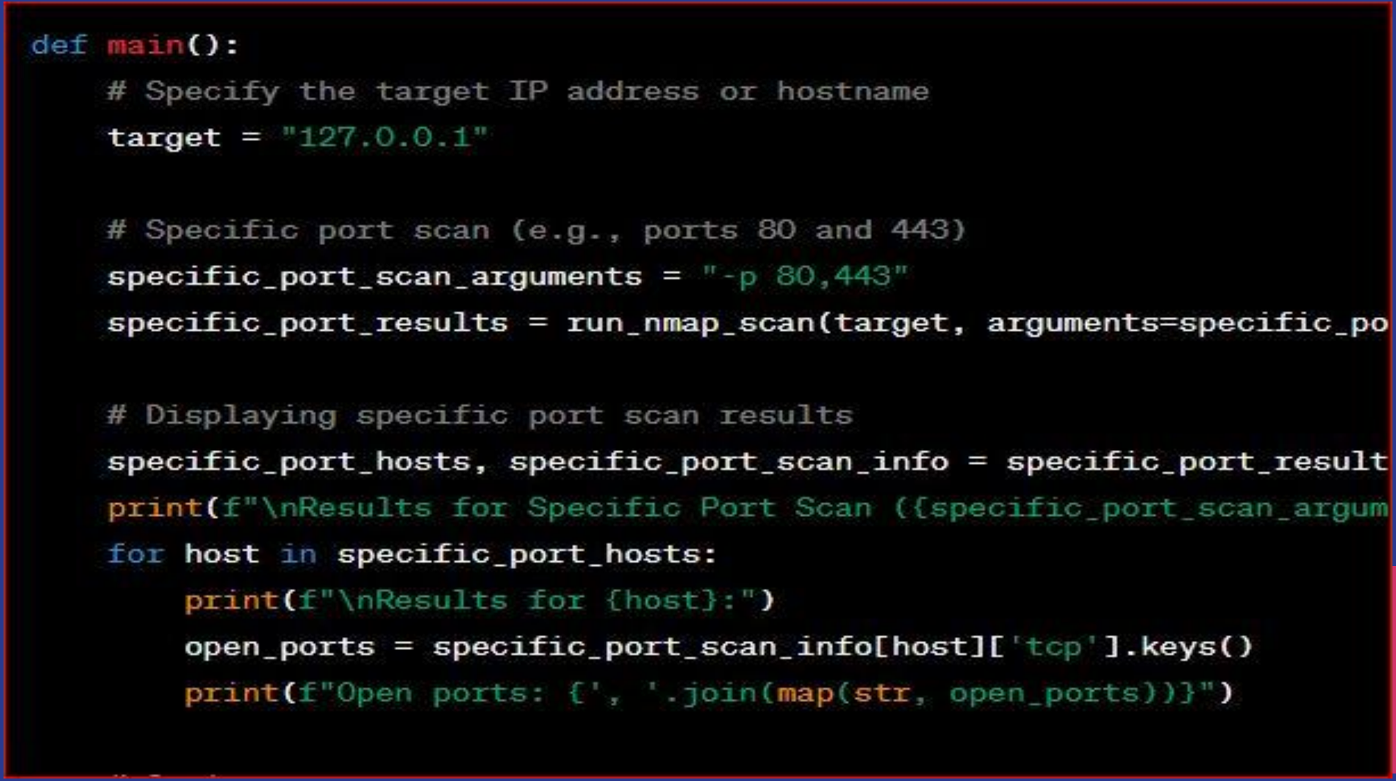
- Initiates a specific port scan with the `run_nmap_scan` function, targeting ports 80 and 443.
- Displays the results of the specific port scan, including hosts and open ports.
- Initiates a script scan with the `run_nmap_scan` function using the default script.



```
def main():
    # Specify the target IP address or hostname
    target = "127.0.0.1"

    # Specific port scan (e.g., ports 80 and 443)
    specific_port_scan_arguments = "-p 80,443"
    specific_port_results = run_nmap_scan(target, arguments=specific_po

    # Script scan
    script_scan_arguments = "--script=default"
    script_scan_results = run_nmap_scan(target, arguments=script_scan_a

    # Displaying specific port scan results
    specific_port_hosts, specific_port_scan_info = specific_port_result
    print(f"\nResults for Specific Port Scan ({specific_port_scan_argum
    for host in specific_port_hosts:
        print(f"\nResults for {host}:")
        print(f"Open ports: {', '.join([str(port) for port in specific_

    # Displaying script scan results
    script_hosts, script_scan_info = script_scan_results
    print(f"\nResults for Script Scan ({script_scan_arguments}):")
    for host in script_hosts:
```

Specific Ports 80 & 443 Scan

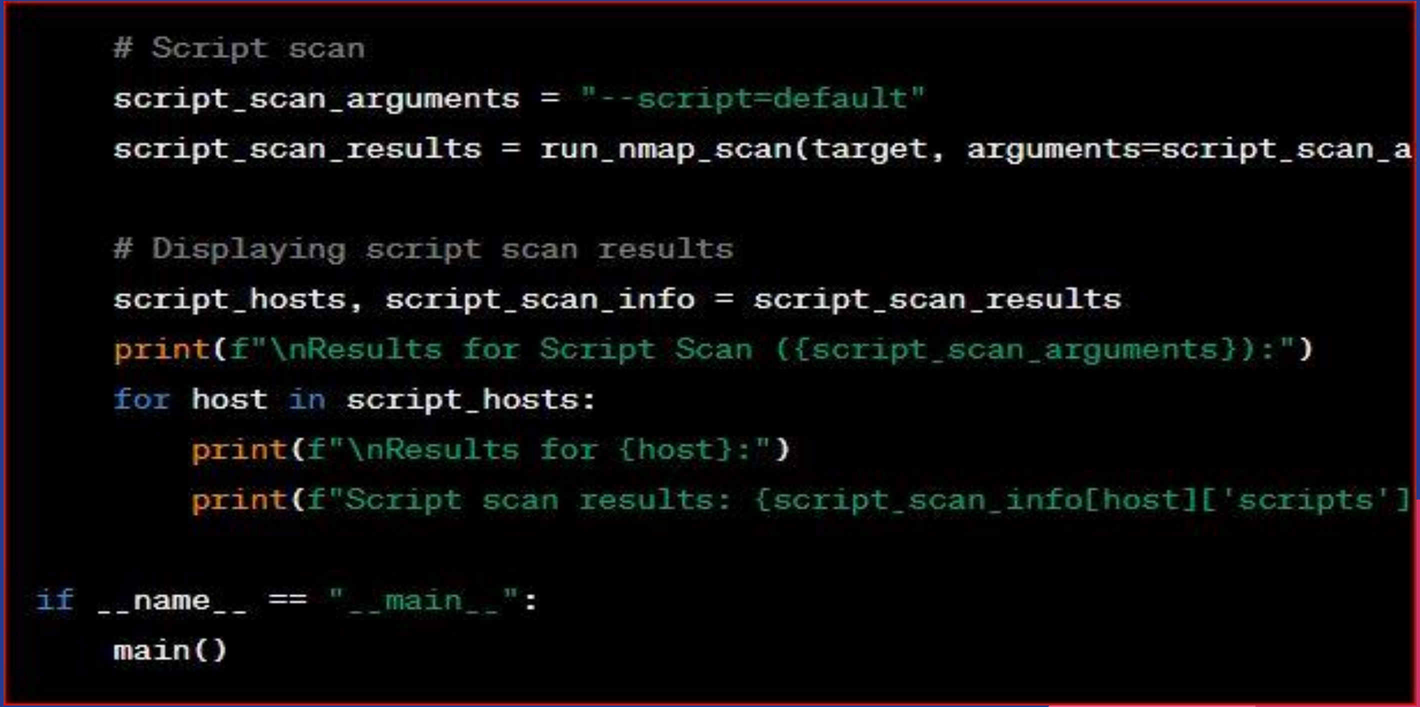
- Targets ports 80 and 443 using the `run_nmap_scan` function with specific port scan arguments.
- Prints results, showcasing hosts and open ports for the specified ports.



```
def main():
    # Specify the target IP address or hostname
    target = "127.0.0.1"

    # Specific port scan (e.g., ports 80 and 443)
    specific_port_scan_arguments = "-p 80,443"
    specific_port_results = run_nmap_scan(target, arguments=specific_po

    # Displaying specific port scan results
    specific_port_hosts, specific_port_scan_info = specific_port_result
    print(f"\nResults for Specific Port Scan ({specific_port_scan_argum
    for host in specific_port_hosts:
        print(f"\nResults for {host}:")
        open_ports = specific_port_scan_info[host]['tcp'].keys()
        print(f"Open ports: {', '.join(map(str, open_ports))}")
```

Script Scan:

- Utilizes the `run_nmap_scan` function with script scan arguments ("--script=default").
- Prints results, revealing hosts and outputs of the script-based scan.



```
    # Script scan
    script_scan_arguments = "--script=default"
    script_scan_results = run_nmap_scan(target, arguments=script_scan_a

    # Displaying script scan results
    script_hosts, script_scan_info = script_scan_results
    print(f"\nResults for Script Scan ({script_scan_arguments}):")
    for host in script_hosts:
        print(f"\nResults for {host}:")
        print(f"Script scan results: {script_scan_info[host]['scripts']

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

Python Script Overview: Script Walkthrough

Importing Libraries:

- Imports `nmap` library for Nmap functionality in Python.

perform_nmap_scan Function:

- Creates an instance (`nm`) of the Nmap `PortScanner` class.
- Initiates a basic Nmap scan on a specified target IP, scanning ports 1 to 1000.
- Returns detailed scan results for the specified target.

print_scan_results Function:

- Prints human-readable information about detailed scan results.
- Displays the target's hostname and IP address.
- Iterates over scanned hosts, printing host state.
- Iterates over scanned ports for each host, printing port details.

main Function:

- Sets the target IP address (`192.168.1.1`) for scanning.
- Calls `perform_nmap_scan`to get detailed scan results.

Specific Port Scan:

- Targets ports 80 and 443 using `run_nmap_scan` with specific port scan arguments.
- Prints results, showing hosts and open ports for the specified ports.

Script Scan:

- Utilizes `run_nmap_scan` with script scan arguments ("--script=default").
- Prints results, revealing hosts and script-based scan outputs.

Key Takeaways

Script Summary Overview:

- Combining Nmap with Python makes scans automatic, improving how fast they work and how results are managed.
- It works on different systems, can be customized easily, and handles errors well.
- Gives clear guidance on using Nmap effectively, adapting to changing security needs.
- It serves as an educational tool for Python and Nmap integration in cybersecurity tasks.
- Its crucial for network security professionals to identify potential vulnerabilities, open ports, and services running on the target system.

Project Summary: "Automating Nmap Scans with Python"

Title and Topic:

- Title: "Automating Nmap Scans with Python"
- Topic: Empowering network security professionals through automated Nmap scans.

End Goal:

- Empower attendees with knowledge and skills for smooth automation of network scanning processes.

Adapt to Best Security Practices Recommendations

Vulnerability Being Exploited:

- Regularly stay informed about Nmap updates, new features, and security trends to mitigate vulnerabilities.

Adjust Python Script:

- Modify the Python script to adapt to evolving security requirements.
- Add or adjust Nmap options based on changing needs and emerging threats.

Adapt to Best Security Practices & Devices Used

Continuously Learn:

- Encourage continuous learning by exploring Nmap documentation and engaging with security forums.
- Enhance understanding of scanning techniques to stay ahead of potential vulnerabilities.

Integrate with Security Practices:

- Incorporate the automated Nmap script into broader security practices and workflows.
- Seamlessly integrate automated scans to enhance overall security posture.

Devices and Technologies:

- Devices: Computers running Python and Nmap.
- Technologies: Python programming language and Nmap scanning tool.

Summary of Device and Technology Usage

- Utilize a text editor to create a Python script named `nmap_automation.py`.
- Combine the capabilities of Python and Nmap to automate and streamline network scanning.
- Enhance security practices by integrating automated Nmap scans into daily workflows.
- Empower professionals to adapt to evolving security requirements through continuous learning and script adjustments.
- Achieve the end goal of seamless automation and heightened network security awareness.

Conclusion

BootCon emphasizes practical skills. This presentation aims to leave attendees equipped with the knowledge and skills needed to implement effective Nmap automation using Python in their professional settings.

Q&A and Interactive Session:

- ???