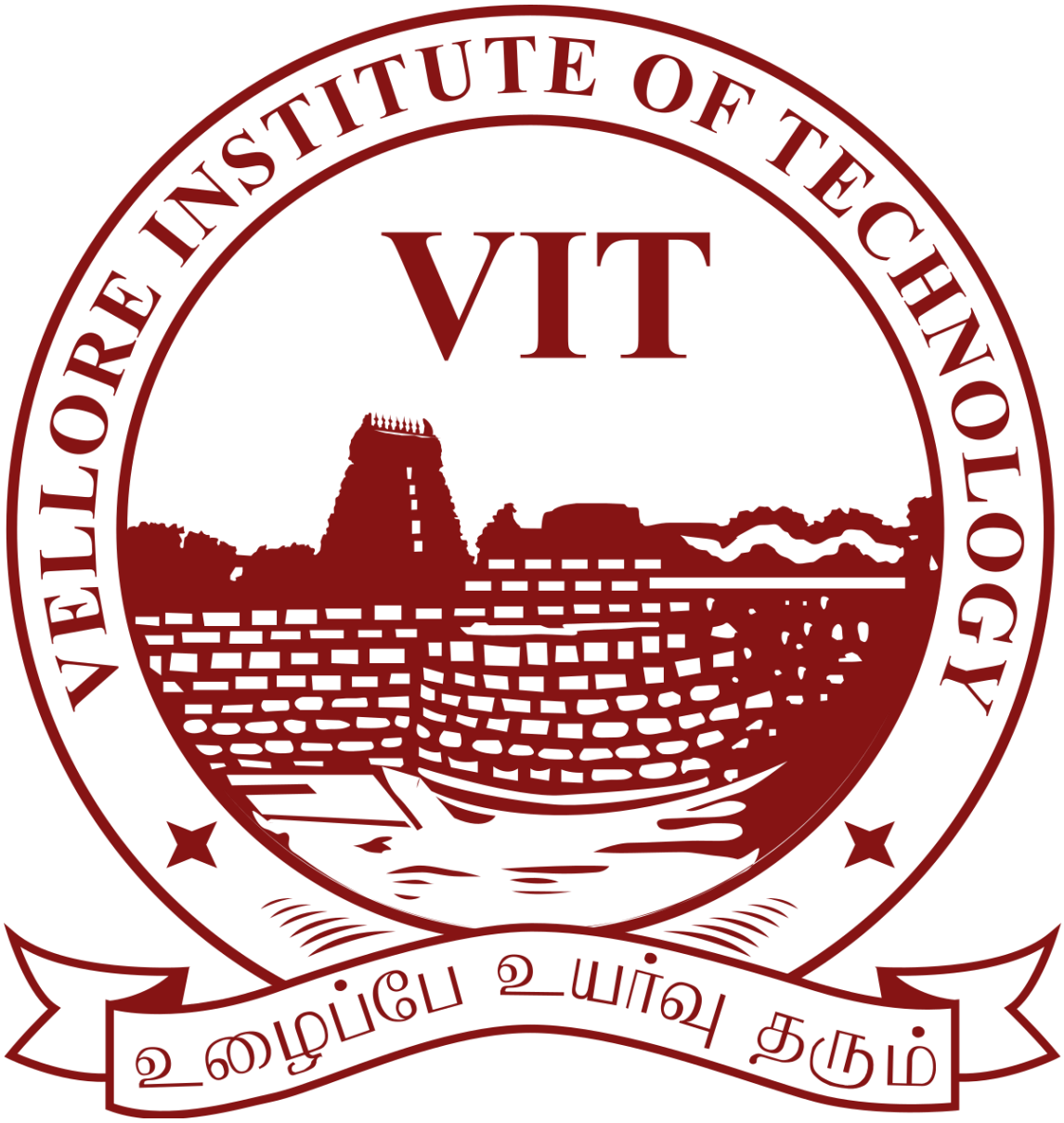

CYBER SECURITY AND DIGITAL FORENSICS - PROJECT

Faculty: Dr. D. Santhadevi (SCOPE)
Slot: A2
Name: M.Rupesh - 22BCE9097
Dev Kumar - 22BCE9115

Port and Vulnerability Scanner

Introduction

This Python script is designed to scan a target IP address for common open ports and identify potential vulnerabilities associated with the services running on these ports. It provides a straightforward way to assess basic security risks in a network environment.

Features

- Scans a target IP for a predefined set of common ports.
- Identifies services running on open ports.
- Highlights known vulnerabilities for the detected services.
- Uses Python's `socket` library for efficient port scanning.

How It Works

1. The script defines a list of common ports and their associated services and vulnerabilities.
2. The user provides the target IP address to scan.
3. The script iterates through the list of ports, attempting to establish a connection to each one.
4. Open ports are identified, and the corresponding services and vulnerabilities are displayed.
5. A summary of potential vulnerabilities is provided at the end of the scan.

Code:

```
import socket
from datetime import datetime

# Expanded list of common ports with corresponding services and
vulnerabilities
common_ports = {
    21: ("FTP", ["Anonymous access", "Plain-text passwords"]),
    22: ("SSH", ["Weak algorithms", "Password-based authentication"]),
    23: ("Telnet", ["Unencrypted", "Easily intercepted"]),
    25: ("SMTP", ["Open relay", "Weak authentication"]),
    53: ("DNS", ["Cache poisoning"]),
    69: ("TFTP", ["No authentication", "Data interception"]),
    80: ("HTTP", ["Directory traversal", "Outdated versions"]),
    110: ("POP3", ["Plain-text transmission"]),
    119: ("NNTP", ["Open access", "No encryption"]),
    135: ("MS RPC", ["Remote code execution"]),
    139: ("NetBIOS", ["Sensitive information leakage"]),
    143: ("IMAP", ["Unencrypted connections"]),
    161: ("SNMP", ["Default community strings", "No encryption"]),
    389: ("LDAP", ["Anonymous access", "Weak authentication"]),
    443: ("HTTPS", ["Weak SSL/TLS configurations"]),
    445: ("SMB", ["Exploitation of EternalBlue vulnerability"]),
    465: ("SMTP over SSL", ["Weak encryption", "Open relay"]),
    514: ("Syslog", ["Sensitive information leakage"]),
    543: ("PostgreSQL", ["Weak passwords", "Unsecured access"]),
    631: ("IPP", ["Printer vulnerabilities"]),
    993: ("IMAPS", ["Weak encryption"]),
    995: ("POP3S", ["Weak encryption"]),
    1433: ("MSSQL", ["Weak passwords", "Remote access"]),
    1521: ("Oracle DB", ["Default credentials"]),
    2049: ("NFS", ["Unrestricted access"]),
    3306: ("MySQL", ["Weak passwords", "Remote access"]),
    3389: ("RDP", ["Exposed to brute-force", "Weak encryption"]),
    5432: ("PostgreSQL", ["Weak credentials", "Misconfigured access"]),
    5900: ("VNC", ["Weak passwords", "No encryption"]),
    8080: ("HTTP Proxy", ["Open proxy", "Unsecured access"]),
    8443: ("HTTPS Alt", ["Weak SSL/TLS configurations"]),
    8888: ("HTTP Proxy Alt", ["Open proxy", "Unsecured access"]),
}
```

```
# Function to scan ports on the target IP
def port_scan(ip):
    open_ports = []
    print(f"\nStarting scan on {ip} at {datetime.now()}")
    for port in common_ports:
        try:
            # Initialize a socket and attempt connection to port
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.settimeout(1)  # Short timeout for fast scanning
            result = sock.connect_ex((ip, port))  # 0 means open
            if result == 0:
                service, vulnerabilities = common_ports[port]
                print(f"Port {port} ({service}) is open.")
                open_ports.append((port, service, vulnerabilities))
            sock.close()
        except socket.error:
            pass
    return open_ports

# Function to check for known vulnerabilities based on open services
def check_vulnerabilities(open_ports):
    print("\nChecking for known vulnerabilities...\n")
    for port, service, vulnerabilities in open_ports:
        print(f"Potential vulnerabilities for {service} on Port {port}:")
        for vuln in vulnerabilities:
            print(f" - {vuln}")
        print()
# Main execution function
def run_vulnerability_scan():
    target_ip = input("Enter the IP address to scan: ")
    print(f"\n[INFO] Scanning {target_ip} for open ports and known
vulnerabilities...")

    # Perform the port scan
    open_ports = port_scan(target_ip)

    if not open_ports:
        print("No open common ports found.")
    else:
        # Check for known vulnerabilities on detected services
        check_vulnerabilities(open_ports)
    print("\n[INFO] Vulnerability scan complete.")

# Execute the scan immediately
run_vulnerability_scan()
```

Output:

```
rupeshmalisetty@Rupeshs-MacBook-Pro ~ % /opt/homebrew/bin/python3 "/Users/rupeshmalisetty/Documents/import socket.py"
Enter the IP address to scan: 115.244.41.200

[INFO] Scanning 115.244.41.200 for open ports and known vulnerabilities...

Starting scan on 115.244.41.200 at 2024-11-06 20:11:22.769371
Port 53 (DNS) is open.

Checking for known vulnerabilities...

Potential vulnerabilities for DNS on Port 53:
 - Cache poisoning


[INFO] Vulnerability scan complete.
rupeshmalisetty@Rupeshs-MacBook-Pro ~ %
```

Code Documentation

1. Common Ports

The `common_ports` dictionary maps port numbers to their respective services and known vulnerabilities. For example:

```python
common_ports = {
    21: ("FTP", ["Anonymous access", "Plain-text passwords"]),
    22: ("SSH", ["Weak algorithms", "Password-based authentication"]),
    # Additional ports...
}
```

2. Port Scan Function

The `port_scan(ip)` function scans the target IP for open ports. It:
- Iterates through the `common_ports` dictionary.
- Attempts to establish a socket connection to each port.
- Collects information on open ports and the services running on them.

Example Code:

```python
def port_scan(ip):
    open_ports = []
    for port in common_ports:
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.settimeout(1)
            result = sock.connect_ex((ip, port))
            if result == 0:
                open_ports.append((port, *common_ports[port]))
            sock.close()
        except socket.error:
            pass
    return open_ports
```

3. Vulnerability Check Function

The `check_vulnerabilities(open_ports)` function identifies and displays known vulnerabilities for each detected service on open ports.

Example Code:
```python
def check_vulnerabilities(open_ports):
    for port, service, vulnerabilities in open_ports:
        print(f"Potential vulnerabilities for {service} on Port {port}:")
        for vuln in vulnerabilities:
            print(f" - {vuln}")
```

4. Execution Flow

The `run_vulnerability_scan()` function orchestrates the execution by:
- Prompting the user for the target IP address.
- Calling `port_scan()` to detect open ports.
- Calling `check_vulnerabilities()` to highlight risks.
- Displaying a summary of results.

Example Code:
```python
def run_vulnerability_scan():
    target_ip = input("Enter the IP address to scan: ")
    open_ports = port_scan(target_ip)
    if open_ports:
        check_vulnerabilities(open_ports)
```

Conclusion

This script provides a simple and effective way to scan for open ports and identify potential vulnerabilities. While it is useful for initial assessments, deeper security analysis requires more sophisticated tools and techniques.