Networks and Systems Security
Week 07
Penetration Testing

Aims of the Seminar

This workshop is designed to provide a hands-on introduction to some core concepts of Penetration Testing (Pen Testing). We will use Python to simulate and understand the initial stages of a penetration test, focusing on reconnaissance and vulnerability assessment. Pen testing is not a "magic bullet" and should supplement a comprehensive security strategy.

Learning Objectives:

- Understand the purpose and basic principles of Penetration Testing.
- Explore fundamental pen testing phases like reconnaissance and scanning using Python.
- Gain familiarity with common tools and techniques used in ethical hacking.

Important Note (Ethics & Legality):

- ALWAYS obtain explicit, written permission before testing ANY system that you do not own.
- Conduct all activities within a designated, isolated test environment (e.g., a local VM, a dedicated test lab, or platforms like HackTheBox/OverTheWire).
- Unauthorized testing is illegal and unethical. The scripts and tools here are for educational purposes only, within your own controlled environment.

Feel free to discuss your work with peers, or with any member of the teaching staff.

Whois domain lookup

A Whois domain lookup allows you to trace the ownership and tenure of a domain name. Similar to how all houses are registered with a governing authority, all domain name registries maintain a record of information about every domain name purchased through them, along with who owns it, and the date till which it has been purchased.

The Whois database contains details such as the registration date of the domain name, when it expires, ownership and contact information, nameserver information of the domain, the registrar via which the domain was purchased, etc.

Practical

Demonstrate passive reconnaissance with Python to gather public domain info safely.

```python
import socket
import requests

def get_domain_info(domain):
    try:
        # Get IP address (active but low-risk)
        ip = socket.gethostbyname(domain)
        print(f"IP Address: {ip}")

        # Get public WHOIS-like info (passive, using a free API)
        response = requests.get(f"https://ipapi.co/{ip}/json/")
        if response.status_code == 200:
            data = response.json()
            print(f"Organization: {data.get('org', 'Unknown')}")
            print(f"City: {data.get('city', 'Unknown')}")
            print(f"Country: {data.get('country_name', 'Unknown')}")
        else:
            print("Could not fetch WHOIS data.")
    except Exception as e:
        print(f"Error: {e}")

# Example: Use a public domain (never use without permission!)
get_domain_info("python.com")
```

Types of Penetration Tests

Test Basis:

- **Black Box (Opaque)**: Minimal info, like external attacker. Realistic for outsiders, shows external posture.
- **White Box (Transparent)**: Full access (code, architecture). Thorough, efficient.
  Test Types:
  1. **Bespoke Software**: For custom apps, e.g., web; provides coding feedback.
  2. **Scenario-Driven**: Tests risks like lost devices or insider threats.
  3. **Detection and Response**: Assesses vulnerabilities plus detection/response.
     Select based on goals—black box for realism, white box for depth. Supplements routine security.

Practical

Simulate black box on a web page. Note: If "Server: Unknown", servers can hide headers for security; dig deeper with other tools like Nmap for behaviour.

```
import requests

def black_box_recon(url):
    try:
        response = requests.head(url)
        print("Black Box Findings:")
        print(f"Server: {response.headers.get('Server', 'Unknown')}")
        print(f"Content-Type: {response.headers.get('Content-Type',
'Unknown')}")
    except Exception as e:
        print(f"Error: {e}")


url = "http://python.com"
known_info = {"server": "Apache 2.4", "vulns": "Check CVE-2021-1234"}
black_box_recon(url)
```

Penetration Testing Methodology and Tools

Methodology steps:

1. **Reconnaissance**: Gather info (passive/active).
2. **Scanning/Enumeration**: Identify hosts/ports/services.
3. **Vulnerability Assessment**: Find weaknesses.
4. **Exploitation**: Attempt breaches.
5. **Post-Exploitation**: Maintain/expand access.
6. **Reporting**: Document findings.
   Tools: Nmap (scanning), Nessus (vulns), Metasploit (exploitation), etc. Get permission always. Trends: AI, cloud testing, ethical considerations.

Practical

Basic port scanner simulation (localhost only) using pure Python. Then, an advanced part using python-nmap to simulate real Nmap functionality.

```python
import socket

def scan_ports(host, ports):
    open_ports = []
    for port in ports:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)
        result = sock.connect_ex((host, port))
        if result == 0:
            open_ports.append(port)
        sock.close()
    return open_ports

host = "127.0.0.1"
ports = [80, 443, 22, 8080]
open_ports = scan_ports(host, ports)
print(f"Open ports on {host}: {open_ports}")
```

Using Nmap in Python

Nmap ("Network Mapper") is a free and open-source utility used by network administrators and security professionals for **network discovery, security auditing, and inventory management**. It sends raw IP packets and analyses responses to identify active hosts, open ports, operating systems, and services on a network.

Key Features and Uses

- **Host Discovery**: Identifies which hosts are online and available on the network.
- **Port Scanning**: Determines which ports are open on target systems, a key step in identifying potential entry points.
- **Service and Version Detection**: Identifies the application name, version number, and protocol of services listening on ports (e.g., a web server, DNS server).
- **Operating System (OS) Detection**: Uses TCP/IP stack fingerprinting to determine the OS and its version running on a target host.
- **Vulnerability Detection**: The Nmap Scripting Engine (NSE) uses Lua-based scripts to automate a wide range of tasks, including advanced service detection and vulnerability checks.
- **Firewall and IDS Evasion**: Includes various techniques to bypass packet filters and intrusion detection systems.
- **Cross-platform Support**: Runs on major operating systems, including Linux, Windows, and macOS.

More information can be found here:

1. https://nmap.org/
2. https://www.freecodecamp.org/news/what-is-nmap-and-how-to-use-it-a-tutorial-for-the-greatest-scanning-tool-of-all-time/

To use nmap with python you can use python package for it:

https://pypi.org/project/python-nmap/

Practical

This uses the python-nmap library to perform a more comprehensive scan, including service detection—similar to the Nmap tool mentioned in the lecture. Run only on localhost or authorized hosts.

```
import nmap

def nmap_scan(host, port_range='1-1024'):
    nm = nmap.PortScanner()
    try:
        nm.scan(host, port_range, arguments='-sV')  # -sV for service version
detection
        for host in nm.all_hosts():
            print(f"Host: {host} ({nm[host].hostname()})")
            print(f"State: {nm[host].state()}")
            for proto in nm[host].all_protocols():
                print(f"Protocol: {proto}")
                lport = nm[host][proto].keys()
                for port in sorted(lport):
                    service = nm[host][proto][port]
                    print(f"Port: {port}\tState: {service['state']}\tService:
{service.get('name', 'unknown')} {service.get('version', '')}")
    except Exception as e:
        print(f"Error: {e}")

# Example: Scan localhost
nmap_scan('127.0.0.1', '1-10')
```

The end 😊