Network Mapping

Network scanner assignment 2

Lesson Description / Topic

As a penetration tester, your very first step is always to gather information about the environment in which you are performing your assignment, including the network. Fortunately, there are many tools available that allow you to do this (such as Nmap, Scapy, Netcat, Masscan, etc.). However, a well-trained security expert does not rely blindly on other people's tools alone. It is important to also understand how the underlying mechanisms work, so that you are able to build your own solution—even without tools—for the problems you will encounter during your career.

The network that needs to be scanned can be accessed via different ports on D2.60 and D2.70. These are connected to the network, and a DHCP server is running on this network to provide your laptops with IP addresses.

You will receive IP addresses in the range **192.168.0.50 – 192.168.0.250** (Systems within this range **do not need to be scanned**.)

Patch Information

The devices that you must scan are located in the IP range: **192.168.0.1 – 192.168.0.49**

Patched table outlets for the scan

D2.60:	B2.70
Tafel 1: 6A13	Tafel 1: 5A23
Tafel 1: 6A14	Tafel 1: 5A24
Tafel 2: 6A7	Tafel 2: 5A17
Tafel 2: 6A8	Tafel 2: 5A18
Tafel 3: 6A22	Tafel 3: 5A11
Tafel 3: 6A23	Tafel 3; 5A12
Tafel 4: 6B11	Tafel 4: 5B5
Tafel 4: 6B12	Tafel 4: 5B6
Tafel 5: 6B1	Tafel 5: 5B9
Tafel 5: 6B4	Tafel 5: 5B10

This week, we will begin by building a **network scanner**.

Submission

You must submit the following items:

- A **PDF of your Lab Journal**
  - In the discussion section, also address the following:
    - Did the final code match the pseudocode well?
    - What caused any differences between them?
- A file containing your answers to **questions 1a and 1b** (preferably in PDF format)
- A **`.py` file** containing your code
- A **link to your Git repository**
  - The code in your repository must be identical to the code in your `.py` file

Assignment

In this assignment, you will write a **network scanner** that is capable of mapping a network environment. Follow the steps outlined in the assignment carefully and ensure that you submit all required subtasks.

1. Design

The most important step when creating your own tool is the design phase. This is the stage where you can save the most time later on. Review the requirements listed later in this document that your network scanner must meet, and think carefully about how your code will be structured. The more effort you invest in the design of your code, the easier the actual programming will be.

a. Use Case

Sketch the use case of your tool. You may do this by describing a set of use cases, creating a flow diagram, or using simple comments. Explain how the user is expected to interact with the tool and what output the tool should generate. Consider the following:

- Which functions will you need to write?
- How will you call these functions?
- What input and output will each function have?

b. Pseudocode

Based on the requirements and your use case description, write the structure of your program in **pseudocode** and include it in your lab journal. If necessary, you may revise this step later.

2. Implementation

Once you have a solid design for your tool, you can begin with development and testing. Pay close attention to naming conventions. Below are several tips to make this process easier.

Tips

- Write and test your code **function by function**.
- Write comments and docstrings as early as possible and keep them up to date. The better you do this, the easier it will be to maintain an overview of your own code, and the smaller the chance that you will get stuck in long and frustrating debugging sessions.
- In Year 1, you have already encountered code using the **Scapy** and **socket** modules; these modules can also be very useful for this assignment.
- Make good use of **pair programming**. Working in pairs allows you to reason through problems much faster (and ultimately, both students must fully understand the complete code in order to meet the requirements).
- The internet can be a valuable resource for writing code, provided that you use it in a targeted way. A general search query such as *"python network scanner"* will likely return an answer that is too large to work with effectively (and may result in plagiarism).

  Therefore, search for **small and specific sub-questions**. For example, a more focused query such as *"Python scapy ARP ping"* will provide a much more specific solution to a subproblem (such as *"How do I use Scapy to discover a single host on a network?"*), which you can then interpret and apply relatively easily in your own solution.

Network Scanner Requirements

1. The user must be able to scan both:
   - a complete subnet, and
   - a single, specific host.
2. The scanner must be able to identify the following information about hosts in the network:
   - a. The IP address
   - b. The MAC address
   - c. Open ports
   - d. The service associated with an open port
   - e. The hostname
   - f. The operating system
3. The output of the scanner must be well-structured, easy to read, and neatly formatted (for example, by using **f-strings**).

Code Requirements

1. It is **not allowed** to use Python's `nmap` module, **except** for:
   - detecting running services, and
   - detecting the operating system

   (requirements d and f).

2. The code must be clearly written and include clear comments and **docstrings**.
   The preferred language for code and docstrings is **English**.
3. The code must follow the **PEP 8** coding standards.

Bonus

1. The user can run the scanner via the **command line**, and control all scan options using flags.
2. In addition to scanning an entire network or a single host, the user can also scan:
   - different subnets, or
   - individual hosts
     (also via the command line).
3. The scan output makes elegant use of **color** to improve readability.
4. While running, the scanner displays useful **progress reports**, and provides an overview of **metrics** at the end of the scan.