IT and Network Security
WiSe 2020/21

Practical Assignment No. 4

Nils Aschenbruck

Leonhard Brüggemann, Alexander Tessmer, Stefanie Thieme

Release	08.12.2020
Submission	04.01.2021

General information for the practical exercise:

- Read the exercise sheet **thoroughly from top to bottom**, before you start working!
- For the admission to the exam, you have to acquire at least 8 points.
- You have to work on all exercises **on your own**.
- **(Code) plagiarism will not be tolerated!**
  This includes the use of code from other sources without reference or unusual amounts of code with references and applies to all parties involved.
- Submission is **no later than 9:00 a.m.** on the respective submission date via **StudIP**. Related files must be packed as a ZIP archive and uploaded in the exercise area of the lecture. The archive name must conform to the following pattern:

  ITS_WiSe_202021_PA4_*MatrNo*

  where ***MatrNo*** has to be replaced with your matriculation number. On unpacking the archive, a **subfolder with the same naming pattern** must be created automatically. Please note that potential changes to this submission method can occur for individual exercises.
- **Comply with all stated specifications** for file names, folder structure, program parameters, etc. Otherwise the time for exercise evaluation is unnecessarily extended and the result announcement will be delayed.
- Please use the **mailing list** of the lecture for questions of general interest. Alternatively you can attend the **Q&A tutoring session**. An application by mail to the teaching assistant (**brueggemann@uos.de**) is optional, but recommended for extensive questions.

Practical Assignment 4: Port Scanning

In the lecture, we discussed the port scanner nmap [1]. In this task, you will implement the basic functions of such a port scanner yourself. Since the port scan is often the first step in penetrating a system, administrators or security officers may check for port scans and exclude an attacker. Therefore the detection of port scans is often part of an intrusion detection system (IDS) or an intrusion prevention system (IPS) such as Snort [2]. Besides programming a straightforward and improved port scanner, you will also implement a simple detector for port scans in this task. As usual, the virtual machine created in PA0 and Python 3.8.4 must be used.

Exercise 1: Simple Port Scanner (1 Point)

First, write a simple port scanner that does *not* require root rights. The port scanner should connect to all ports in the specified area one after the other using full open TCP connections. The output should only be generated for open ports. Your program call must look like this:

```
./simple_scanner.py TARGET_HOST START_PORT END_PORT
```

The output must look like this:

```
Port NUMBER is open.
```

Exercise 2: Simple Detector (1 Point)

Now write a simple detector that will detect such a scan using the python library `asyncio` [3]. The detector must also work *without* root rights. Implement the following simple heuristic in which a list of all TCP connections is constantly examined to distinguish potential port scans or usual or accidental connection attempts:

1. Examine for each established TCP connection:
   - 1.1. Did the source IP connect for the first time?
     Then register source IP, port, and the time of the connection.
   - 1.2. Did the source IP connect during the past 10 seconds? Then:
     - 1.2.1. Compute how many ports had already been used for the source IP.
     - 1.2.2. If more than ten ports have been scanned already, then the source IP is regarded as a port scan. A source IP, once regarded as a port scan, should not be reported multiple times.
   - 1.3. Did the last established connection with the source IP was made more then 10 seconds ago? Then disregard the source IP as a port scanner for now.

If you come across a resource limit in the number of ports, you may limit the number of ports to a reasonable size. As with a honeypot, it is acceptable if no other services can run on the ports at the same time. Your program must start using the following pattern:

```
./simple_detector.py START_PORT END_PORT
```

The output must apply to the following format like this:

```
Scan detected from IP 127.0.0.1 in Scanned Ports [2001,2002,2003].
```

Note: Using the method *loop.create_server* allows to create a TCP server listening on a port on the host address and *more*.

Exercise 3: Improved Port Scanner (1 Point)

Now write an improved port scanner (this time with root rights) that the previous simple detector (without root rights) does not recognize. Your program must start using the following pattern:

```
sudo ./advanced_scanner.py TARGET_HOST START_PORT END_PORT
```

The output must look like this:

```
Port NUMBER is open.
```

Use the Python files specified in `PA_Blatt04_files.zip` for creating raw sockets or to process IP- and TCP packets. The `pa6 folder` in the zip file must be in the directory with your source files. Then you can access the provided classes as follows:

```
from pa6.ip import SimpleIPv4Packet
ip_packet = SimpleIPv4Packet(...)
```

Note: The file `ip_tcp_frame.txt` contains information about header formats. It is also worth looking at the classes provided.

Happy Coding!

Submission:

- upload the ZIP-archive *ITS_WiSe_202021_PA4_MatrNo.zip* to **StudIP**: *IT- und Netzwerksicherheit* → *Tasks* → *PA4*
- ZIP-archive named *ITS_WiSe_202021_PA4_MatrNo.zip* must create the corresponding folder *ITS_WiSe_202021_PA4_MatrNo* on unpacking (cf. general information on page 1).
- This folder must have the following structure and contain the following files:

```
ITS_WiSe_202021_PA4_MatrNo
├── simple_scanner.py
├── simple_detector.py
├── advanced_scanner.py
└── pa6
    ├── __init__.py
    ├── ip.py
    ├── rawsocket.py
    └── tcp.py
```

Sources

[1] https://nmap.org

[2] https://snort.org

[3] https://docs.python.org/3/library/asyncio.html