The University of Queensland

School of Electrical Engineering and Computer Science (EECS)

Semester 2, 2024

CYBR3000

Information Security

Assignment 1

Due: 3:00pm, 20th September 2024

Marks: 100
Weight: 25% of your final grade.
Version: 1.3

Revision history:

- Version 1.0: 19 Aug 2024 release
- Version 1.1: 8 September 2024.
  - Fix a typo: NIDS.py to IDS.py
- Version 1.2: 16 September 2024.
  - Add "+" into the "flags" parameter for Part B IDS rules.
  - Add some clarifications for Part B IDS rules.
- Version 1.3: 19 September 2024.
  - Fix the inconsistency in submission structure

Introduction

This assignment is divided into two parts: Part A and Part B. Part A requires you to write multiple iptables rule sets based on provided scenarios. Part B requires you to use Python3.9 to code an Intrusion Detection System (IDS).

1. **Part A: Intrusion Prevention (25 marks total)**

You are required to write iptables rule sets for each of the following tasks. Each rule set for each of the tasks needs to be written in a separate .txt file. Each line in the .txt file refers to one single iptables rule. The name of the .txt file must match the name of the corresponding task (e.g. task1.txt). Please note that no marks will be given if the name of the .txt file is incorrect or mislabelled. The rules in each .txt file are supposed to be ready to use, which means others could copy-paste and run each rule of the rule set in the terminal to meet the requirements of the task. One example of a single rule in a rule set is:

```
sudo iptables -A INPUT -s 192.168.1.100 -j DROP
```

Please do not have any empty lines between each rule in a rule set.

To use iptables, you need to test your rules in a Linux-based system. Tools like Docker or a VM using software like VirtualBox or VMware would allow you to run a Linux environment where iptables can be used.

You will have to write a rule set for each of the following tasks:

- **Task 1 (5 marks):**

Allow incoming SSH traffic (which is using port 22) from a specific IP address (192.168.1.100) and deny all other incoming SSH connections.

- **Taks 2 (5 marks):**

Allow incoming HTTP (port 80) and HTTPS (port 443) traffic but drop all other incoming traffic.

- **Task 3 (5 marks):**

Limits incoming ICMP ping requests (ICMP echo-request) to only two per second.

- **Task 4 (5 marks):**

Allow only established and related connections for incoming traffic, blocking new incoming connections except for SSH.

- **Tasks 5 (5 marks):**

Block all incoming and outgoing traffic by default but allow SSH access from a specific IP (192.168.1.100), HTTP/HTTPS traffic, and DNS queries.

2. Part B: Intrusion Detection (75 marks in total)

In this part, you are required to write a Python program called IDS.py that mimics the behaviour of an Intrusion Detection System (IDS). This program would read two files: 1) one file includes intrusion detection rules (like rules used in Snort) and 2) the other file is the .pcap file that contains all the packets that your program would go through to check if any or some of them violates the rules. Both files will be passed into your Python program (IDS.py) through the Command-Line Argument. The start of IDS.py would be:

```
$python3 IDS.py <path_to_the_pcap_file> <path_to_the_IDS_rules>
```

Both paths need to be absolute paths.

Python3.9 is the version required for this part and the final test for marking this assignment will use Python3.9 as well. We will not allow the use of any other Python versions, and no marks will be given if unexpected behaviours happen due to the wrong Python version. Scapy is the library allowed in this part, do not use any other external library in your code.

Examples for both IDS_rules.txt and .pcap files are given on the BlackBoard. You need to read and parse each rule and use the rule to monitor and detect packets in the .pcap file. Please be aware that the provided .pcap files are only a subset of the .pcap file which will be used in the final marking of this assignment.

The format of the IDS rules is like the format of snort rules with slight differences. One simple example of your IDS rules is:

```
alert tcp 192.168.102.132 any -> any any (msg: "receive a TCP packet";)
```

This rule would raise an alert when the packet is an incoming TCP packet that comes from any port from IP address 192.168.102.132 and is sent to any port number in any IP address. If a packet like that is found, the IDS.py would log the message into a log called IDS_log.txt that has the following format:

```
2024-08-18 11:47:53 - Alert: receive a TCP packet

2024-08-18 11:47:53 - Alert: receive a TCP packet

... (if more packets are found)
```

Each line in the IDS_log.txt represents that the IDS finds a packet that meets at least one of the rules. Each line starts with a time stamp and then follows with `Alert: <content_in_msg>`

The IDS_log.txt must match the format given above since you will be marked based on this.

A more complicated example is:

```
alert tcp 192.168.102.132 any -> 131.171.127.1 25 (content: "malicious"; msg: "multiple malicious TCP syn packets found"; flags: S; detection_filter: count 10, seconds 2;)
```

In this example, the IDS.py would raise an alert if the IDS found more than 10 TCP syn packets within 2 seconds is sent from any port number from IP address 192.168.102.132 to port 25 on IP address 131.171.127.1 that has a content that contains string “malicious”.

Your IDS.py should ignore any line that starts with “#” in the rule set.

To clarify some details:

1. The IDS.py only has alert as the action after detecting a packet
2. The IDS.py supports four different protocols: ip, icmp, tcp, udp.
3. The IDS.py does not support IP range, all test cases will have one source IP address and one Destination IP address.
4. The IDS.py only considers incoming traffic, the “->” symbol would remain the same across all test cases.
5. Each rule option inside the brackets in the rule should be separated by “;”. The last option in the rule should finish with a “;” follows with a “)”.
6. The “flags” option supports four different types: A, S, F, R
   - A: ACK (Acknowledgement)
   - S: SYN (Synchronize sequence numbers)
   - F: FIN (Finish)
   - R: RST (Reset the connection)
   - A/S/F/R+: “+” means matching packets with a certain flag set, regardless of any other flags. For example, A+ will match any TCP packet with the ACK flag set, even if other flags (such as SYN, FIN, or PSH) are also set.
7. “detection_filter” would only and always have two options: “count <the number of packets>”, and “seconds <the time window in second>”. Please follow the exact format provided in the example.
8. All rules used in this Assignment are valid and match the format shown in the provided test scenarios. There is no need to do error checking and error handling for rules.
9. The parameters in the rules can appear in any order (such as msg, detection filter, flags, count).

Your IDS.py will be tested in the following scenarios (Tasks):

1. Detect multiple TCP packets (5 marks).
2. Detect multiple ICMP packets (5 marks).
3. Detect multiple IP packets (5 marks).
4. Detect multiple UDP packets (5 marks).
5. Detect a mix of TCP, ICMP, UDP and IP packets (5 marks).
6. Detect one packet with malicious content within all other benign packets (5 marks).
7. Detect packets with malicious content across different protocols (5 marks).
8. Detect TCP syn packets, fin packets, rst packets and ack packets (10 marks).

9. Detect TCP flooding (10 marks).
10. Detect TCP syn scan (10 marks).
11. Detect multiple TCP ack packets with malicious content within a short period of time (10 marks).

A more detailed description of each scenario will be provided in the rule set documents.

3. Submission

You should submit your assignment using the submission link in the Assignment 1 folder in the course blackboard (under Assessment).

In summary, you need to submit the following:

- Part A:

Place all your iptables rule files in the same folder named YourStudentID_A1_A

- Part B:

Place your Python code ~~and all your IDS rules files in the same~~ in a folder named YourStudentID_A1_B.

Compress the two folders from the tasks into one zipped (compressed) file with YourStudentID_A1.zip (e.g., 12345678_A1.zip):

```
e.g.
12345678_A1.zip
        - 12345678_A1_A
                - task1.txt
                - task2.txt
                - task3.txt
                - task4.txt
                - task5.txt
        - 12345678_A1_B
                - IDS.py
```

Please make sure that the code you submit is the final version. Also please make sure to complete your submission by clicking the submission button. Note that it is not allowed to manually change any part of your submitted Python code and text files, including modifying lines, whether they are commented out or added for testing purposes.

4. Integrity

All code submissions will be checked for similarity and plagiarism. All code in your submission that was not written by you (or was written by you for a previous assignment) must be correctly referenced in IEEE format in your code. This includes code written by generative AI.
Note that *the course coordinator and casual academic may interview some students if there are any suspicious behaviours.*