Assignment 2: Network Intrusion Detection

This project is due on **October 1, 2021** at **5 pm**. You may work with a partner on this assignment and submit one project per team. You may NOT work with the same partner you have worked with on a previous assignment. Submit your solutions electronically.

Background

Network operators actively monitor their networks to protect against various intrusion attacks. Network attackers often perform random "portscans" of IP addresses to find vulnerable machines. Network Intrusion Detection Systems (NIDS) attempt to detect and flag such behavior as malicious.

In this assignment, you will analyze NetFlow network measurement data from a campus border router to identify potentially malicious traffic sent to the campus network. You will then simulate an online algorithm for identifying malicious remote hosts.

Objectives

- Gain familiarity with network measurement data
- Implement basic approaches for identifying malicious traffic
- Reason about the network security threats faced by universities

Provided Files

- `Readme.pdf`: This file
- `netflow.csv`: Pre-processed Netflow data from a campus network border router. The data is "unsampled," i.e. it compiles flow statistics for every packet that traverses any interface on the border router. Each row of the `netflow.csv` file, except for the first-row header, logs the following information for a traffic flow:
  - Date first seen, Time first seen (m:s), Date last seen, Time last seen (m:s), Duration (s), Protocol, Src IP addr, Src port, Dst IP addr, Dst port, Packets, Bytes, Flags, Input interface, Output interface
- `TestNIDS.py`: Test functions to check your intrusion detection code by hashing your results and comparing them to the hash of the correct results
- `NIDS.py`: Starter code
- `Questions.txt`: Written questions. Fill your answers in this file (see Deliverables #2 below)

Part 1: TCP SYN Scan Detection

Attack traffic is often difficult to detect among the legitimate traffic that is typically found in a large enterprise network. Malicious hosts might not show up in lists of the top senders (either by bytes or by flows), and it can be difficult to determine the exact nature of attack traffic in advance.

However, certain types of attacks have characteristic features that simplify their detection. For example, many attackers attempt to detect hosts running vulnerable services by scanning the address space using TCP SYN packets. If a NetFlow record indicates that a flow contained a TCP SYN packet but no packets with the ACK flag set, we can conclude that this flow never completed and thus may be part of a TCP SYN scan—particularly if there are a lot of these flows.

When an attacker sends a TCP SYN packet as part of a TCP SYN scan, there are three possible outcomes:

1. There is no active host on the network at the destination IP address. In this case, we will only see TCP SYN flag. This is very different from the legitimate traffic behavior. It should be possible to identify malicious scanners by identifying SYN-only flows.
2. There is an active host at the destination IP address that is listening on the destination port of the SYN packet. In this case, the NetFlow record will contain both the TCP SYN and ACK flags in the flow record. These flows are difficult to distinguish from legitimate flows, because a host actually answered the initial SYN.
3. There is an active host at the destination IP address, but the port to which the SYN is sent is closed so the server sends back a RST/ACK packet. According to normal TCP implementation guidelines, the scanner will immediately stop any TCP connection attempts once it receives a RST. Like in the first case, the NetFlow record will show SYN requests from the scanner host but no SYN/ACK packets in response.

Instructions:

1. If the matplotlib library is not already installed, run the command `pip3 install matplotlib`
2. Complete the `detect_syn_scan()` function in `NIDS.py`. The function should calculate and print the percentage of TCP flows in the netflow data that have SYN packet but no ACK packet. A flow has a SYN packet if there is an "S" in the "Flags" field. A flow has an ACK packet if there is an "A" in the "Flags" field. Be sure to only consider TCP traffic (use the "Protocol" field to filter out traffic from other protocols).

Part 2: Portscan Detection

Another method of identifying attacks involves TCP ports that are known to correspond to vulnerable services. For this assignment, we will consider ports 135, 139, 445, and 1433 as "known bad" ports. These correspond to Windows RPC, NetBIOS, SMB, and SQL-Snake attacks. We expect that most of the traffic to "known bad" ports are scanners, but that is not necessarily true for all traffic to these ports.

We want to test whether flows to known bad ports on the campus network are more likely to be TCP SYN scans then flows to other ports.

Instructions:

1. Complete the `detect_portscan()` function in `NIDS.py`. The function should first divide all TCP flows in the netflow data into the following categories:

    - SYN-only flows to known bad ports
    - Other flows to known bad ports
    - SYN-only flows to NOT known bad ports
    - Other flows to NOT known bad ports

2. The function should then calculate and print the following 3 percentages from the categorized flows:

    - Precent of flows to known bad ports (135, 139, 445, and 1433) out of all TCP flows
    - Percent of SYN-only flows out of all TCP flows to known bad ports
    - Percent of SYN-only flows out of all TCP flows to NOT known bad ports

Part 3: Malicious Host Detection

In addition to identifying potentially malicious flows, a network operator would like to identify potentially malicious hosts and block future traffic from these hosts. At the same time, network operators want to avoid blocking benign traffic. The potential for blocking benign traffic depends on the overlap between hosts that send malicious-looking traffic and hosts that send normal traffic.

Instructions:

Complete the `detect_malicious_hosts()` function in `NIDS.py`. The function should

1. Only consider hosts with source IP addresses **external** to the campus network (not `128.112.*.*`). The provided function `is_internal_IP()` returns True if the argument IP address is internal
2. Find the set of all external hosts (IP addresses) that sent SYN-only or known-bad-port flows. We will call these the "malicious hosts"
3. Find the set of all external hosts that sent non-SYN-only flows to NOT known bad ports. We will call these the "benign hosts"
4. Find the intersection of the malicious hosts with the benign hosts. We will call this intersection the "questionable" hosts
5. Remove these "questionable" hosts from the "malicious hosts" and "benign hosts" sets
6. Calculate and print 3 values:
   - The number of remaining malicious hosts
   - The number of remaining benign hosts
   - The number of questionable hosts

Part 4: Online Portscan Detection

In Part 3, we used offline analysis of pre-recorded traffic to identify potentially malicious hosts. A real network operator would not want to wait for offline analysis in order to detect potential threats. Instead, they would configure a network intrusion detection system (NIDS) to use an online port scan detection algorithm to classify hosts as benign or malicious, updating these classifications in real-time as traffic flows occur.

The "Bro" network intrusion detection algorithm was published in 1999 by Vern Paxson, et al. and builds on the same intuition that failed connection attempts are good indicators for identifying scans. The goal of Bro is to identify malicious hosts (by IP address) and block future traffic from these hosts.

The Bro algorithm should do the following:

1. For each sending host (external IP address), count the number of unique internal destination IP addresses contacted with failed TCP connection attempts on these "known good" ports: HTTP (80), SSH (22), Telnet (23), SMTP (25), IDENT (113), FTP (20), and Gopher (70). A failed connection attempt is one that does not have both a SYN and an ACK. (We expect many connections to be initiated to these ports, so we should only count failed connection attempts as potentially indicating a malicious sending host.)

2. For each sending host (external IP address), also count the number of unique internal destination (IP) addresses contacted with failed **or** successful TCP connection attempts (SYN packets) to all other ports. (We do not expect benign hosts to contact these ports, so any connection attempt may potentially indicate a malicious sending host.)

3. If the total number of unique internal destination IP addresses counted in 1 & 2 exceeds a parameter threshold, T, for a given external host, assume that host is malicious and add its IP address to a set of blocked hosts.

Instructions:

1. Complete the Bro class in NIDS.py to implement the Bro online port scan detection algorithm.

2. Run NIDS.py to plot (and save) the sensitivity curve for the Bro function for T in the range [1, 120]

Deliverables

One partner should submit the following files:

- `NIDS.py` with your implementations from Parts 1-4 above
- `sensitivity_curve.png` with your plot of Bro's sensitivity curve for T in [0, 120]
- `Questions.txt` completed with your **concise** answers

Extra Credit "Bug Bounty"

If you find a bug anywhere in this assignment, please inform Prof. Apthorpe. The first student (or partners) to find any particular bug will be given a small amount of extra credit. This is an incentive to start the assignment early and will help make the course better for students in future years.