Level 1.1 - Intro

Let's build an NIDS (Network Intrusion Detection System) in Python!
In this level we will create the basic skeleton and implement the first detection.

Environment

- Install VMWare Workstation Pro / Fusion Pro
- Download `NIDS_workshop_x64.zip` or `NIDS_workshop_arm.zip` according to your computer's architecture.
- Extract the zip and double-click the .OVF file to deploy the VM to your computer
- Power on the VM, then minimize it and forget about it
- There should now be an internal private network running (subnet 10.0.0.0/24). Attacker: 10.0.0.7 Client: 10.0.0.13 And numerous servers. This is the virtual network you will be defending!
- You can't access the internal network, but you will get a mirror of all its traffic to interface `VMware Network Adapter VMnet1` on your computer - as we learned, a requirement for NIDS is getting a mirror of the traffic. **Notice**: On mac, this interface will be called `vmenet3`.
- Open both the attacker and client user interfaces in a browser: `http://attackui:5000`, `http://attackui:5001`. It might take up to a minute after powering up the VM for these sites to be available. **Notice**: If this doesn't work, try `http://attackui.local:5000` and `http://attackui.local:5001` instead
- Start the first simulation - you should see all the internal network traffic. Develop detections accordingly to this traffic.
- Good luck!

Simulation UI

1. The attacker and client both have a web interface running on port 5000. Make sure you can access them:

- `http://attackui:5000`
- `http://attackui:5001`

2. This is the simulation user interface from which you can initiate network simulations of attacks
3. Each level contains two triggers
   - Malicious trigger: Creates a network communication inside the network that should be detected by your NIDS
   - Non-malicious trigger: Similar, but the communication is legitimate and shouldn't trigger an alert by your NIDS
4. Rule of thumb: **Every malicious trigger should cause an alert to be displayed by your NIDS. No non-malicious trigger should cause an alert.**
5. Starting from the next level, you will be given a password to unlock each level
6. Most levels can be initiated only from the attacker or only from the client. For example, first level (1.1) can be found in the attacker UI

Project skeleton

1. Create a new directory for your solution
2. Download the starter code

3. Edit the detect*malicious*ip() function.
    - As a start, just make it print **all sniffed packets**.
    - At the same time, run Wireshark and make sure you see the packets
    - The network interfaces are usually `VMware Network Adapter VMnet1` (or `vmenet3` for mac), but may be different depending on your device
    - Edit the returned values in the get_interface() function if your interfaces are different

Identifying Malicious Traffic

1. Run your code, and Wireshark. Make sure both are capturing on the correct interface.
2. Then, click the first level's malicious trigger, which is on the attacker's simulation interface.
3. Threat intelligence has identified a C2 server's IP! - 4.2.2.2
    - Your script should be printing out packets from the attacker to IP 4.2.2.2 (as well as other packets).
    - You should also see them in Wireshark
4. Important: If you don't see the communication in wireshark and Scapy, you perhaps chose the wrong network interface, or for some reason it has a different name on your computer (this can happen with different versions of VMWare or operating systems). Find the correct interface on Wireshark, and go back to Project Skeleton Pt 3.
5. Now you know what you need to detect!

Adding the Detection

1. Add the identified C2 server IP to the list of malicious IPs, 'MALICIOUS_IPS'
   - We'll store the malicious IPs as a list (even though we only know one currently)
   - This allows definitions to be easily added as new threats are discovered
2. Now, edit the detect*malicious*ip() function
   - Add some detection logic to check if the packet's origin/destination is in the list of malicious IPs
   - Call the alert() function with a descriptive alert message when the malicious IP is detected
   - For example, if the NIDS detects a packet from the C2 (4.2.2.2) to 10.0.0.1, or vice versa: ` ``*ALERT* Communication between malicious IP 4.2.2.2 and 10.0.0.1``
3. Test that the malicious trigger triggers an alert on the NIDS
   - Also ensure the non-malicious trigger doesn't trigger an alert

Understanding the Code

1. Try to understand the rest of the provided starter code
2. Instead of printing out the entire packet, your NIDS just prints a message when such a packet is found.
   - This helps to focus the alerts on the alert message and its relevant details

3. Also, we've implemented an alert timeout.
    - If there are multiple packets, the NIDS *doesn't* display an alert for each.
    - Instead, we maintain an alert timeout - for example, don't repeat the same alert in a 10 second window.
    - The user doesn't care about each individual packet - they should only be informed that IP X in their network is communicating with a C2; doesn't matter if it's 1 packet or a 100.

To submit

Submit file `main.py`.

Sentinel