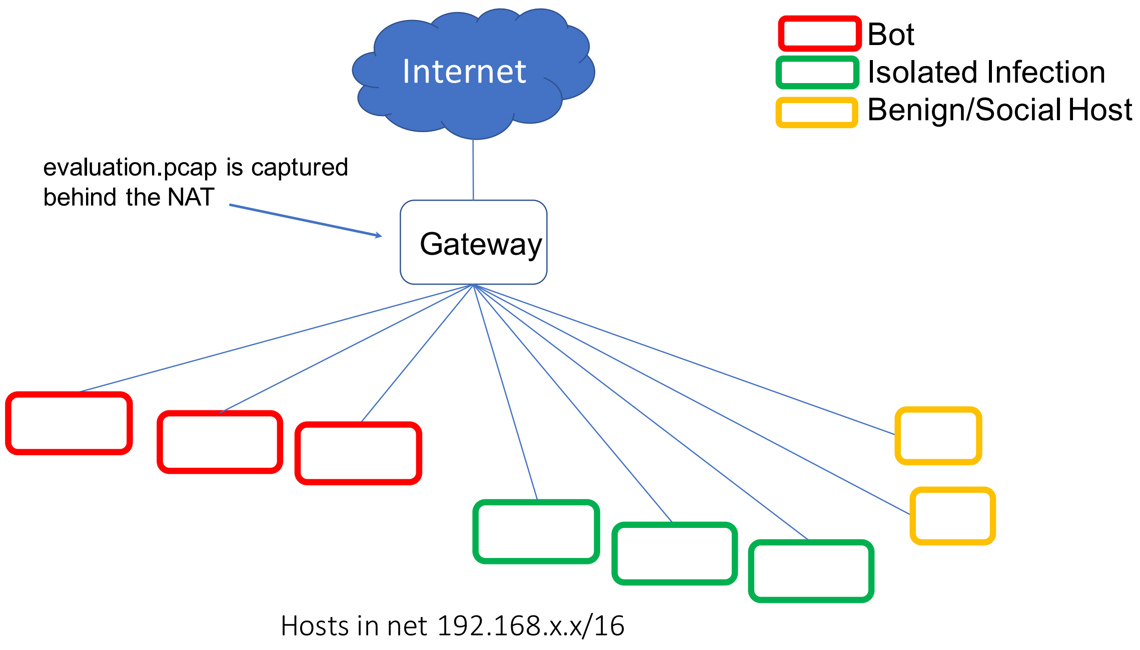
CS6262 - Network Security
Project 4

1 Introduction - Goal of the assignment

The goal of this project is to introduce students to techniques that help to differentiate between malicious and legitimate network traffic. This is a task that threat analysts and network operators are often called to perform. In this project, the students are provided with samples of malicious and legitimate traffic, so that they can observe how each type of traffic looks like. Then they are given a pcap file that contains network traffic that originates from multiple hosts in the same network. This pcap is a mixture of legitimate and malicious traffic. The students are asked to replay the pcap file against a network interface. Finally, the students are asked to use Snort and write their own Snort rules to differentiate between malicious and legitimate traffic. In summary the students are introduced to:

- Observing pcap samples of legitimate and malicious network traffic.
- Using TCPreplay to replay pre-captured network traffic.
- Using Snort and writing Snort rules to differentiate legitimate traffic from malicious traffic.



Internet
Bot
Isolated Infection
Benign/Social Host
evaluation.pcap is captured behind the NAT
Gateway
Hosts in net 192.168.x.x/16

Figure 1: *Network setup for traffic collection.*

2 Definitions and Traffic Collection Set-up

In this assignment, we have collected traffic from an enterprise network network. The hosts are behind a NAT, and their IP addresses belong to a single /16: 192.168.x.x./16. Figure 1 shows a visual representation of the network and of our traffic collection set-up. Each host in the network can only have one out of three distinct roles:

- **Benign host.** The host generates normal background traffic (*e.g.*HTTP). Also the host may be generating traffic due to social activities, *e.g.*the host communicates with a server to participate in a chat channel.
- **Isolated infection.** The host is infected and performs a malicious activity *e.g.*scanning, spam, DDoS, etc. The host generates normal background traffic (*e.g.*HTTP) and malicious activity traffic *e.g.*scanning, spam, DDoS, etc.
- **Bot.** The host communicates with a C&C server to receive commands **AND** it performs a malicious activity (sign of infection) *e.g.*scanning, spam, DDoS, etc. The host generates normal background traffic (*e.g.*HTTP), C&C

communication traffic and malicious activity traffic. Please read the paper "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation" which was published by Gu et al. at Usenix Security 2007. Specifically, for a complete discussion about what is a bot (definition) please see Section3. As the authors state: *A bot has to exhibit: Evidence of local host infection, AND evidence of outward bot coordination or attack propagation.* Reading the entire paper can give you a lot of information and background for this project. The paper can be found here: `http://faculty.cs.tamu.edu/guofei/paper/Gu_Security07_botHunter.pdf`. Also, please see the reading section below.

Samples of traffic. For each type of traffic that is mentioned above, we provide a sample of that category/type of traffic. These samples are only for **illustration** purposes. These samples are *only examples*, and they are **not** the same as the actual traffic that is included at the evaluation pcap, which the students will need to label.

- Legitimate background traffic. For the purpose of this exercise we assume normal traffic to include HTTP, DNS. For an example of normal (attack free) traffic please take a look at the sample file. **Sample pcap**:

  `SAMPLE_background_legit.pcap`

- Legitimate social activity traffic. This traffic is benign and it generated when a host participates in benign social activities *e.g.*chat. **Sample pcap**:

  `SAMPLE_benign_socialchat.pcap`

- Infection traffic: Scanning. **Sample pcap**:

  `SAMPLE_bad_scan.pcap`

- Infection traffic: HTTP DDoS. **Sample pcap**:

  `SAMPLE_bad_ddos.pcap`

- C&C communication. The host generates this traffic *explicitly* to communicate with a C&C server. The host communicates with the C&C server to receive commands, updates, *etc.* . **Sample pcap**:

  `SAMPLE_bad_CnC.pcap`

3 Tasks

The goal is to: a) write Snort rules, b) using the Snort rules to label each connection at the `evaluation.pcap` file as CnC-communication (communication between the host and a C&C server to receive commands), malicious activity (scanning or ddos), or other (if the connection does not belong to any of the previous two categories), and c) to label each host with a role (one of the distinct roles above).

Towards this goal, please follow the tasks below:

- Import the VM. Login to the VM using: `login:  mininet, password:  mininet`
- Locate the pcap files at the directory: `/home/mininet`. In this directory you will find the sample pcaps and the evaluation pcap `evaluation.pcap`.
- Observe the sample pcaps to get an idea about how each type of malicious traffic looks like.
- If for any of the pcap files you encounter header errors, you can issue this command on the command line:
  ```
  editcap -F libpcap -T ether old_file.pcap new_file.pcap
  mv new_file.pcap old_file.pcap
  ```
  - You can use wireshark or tshark to isolate some traffic, if you want. For example in wireshark you can apply display filters *e.g.*`tcp` (to display only TCP traffic), `ip.addr == 10.0.0.1` (to display traffic that originates from or it is destined to this IP address), also you can combine filters using `or` and `and`.
  - To view the files on your local host with wireshark, you will need to install wireshark: `https://www.wireshark.org/`
  - You can scp the files from the VM to your local machine, and view them using wireshark. See notes at the end (Useful commands), on how to scp the pcaps to your local machine.
- Write Snort rules. To write Snort rules, you will need to edit the file:

  ```
  sudo vim /etc/snort/rules/local.rules
  ```

  - Example Snort alert rule based on IP: `alert tcp 10.0.0.1 any -> any any (msg:"TCP traffic detected from IP 10.0.0.1"; GID:1; sid:10000001; rev:001;)` It creates an alert message:

```
"TCP traffic detected from IP 10.0.0.1"
```

  when there is a TCP connection from source IP `10.0.0.1` and any port to any destination IP and any destination port.

  - The above example Snort rule has been entered for demonstration at the file `/etc/snort/rules/local.rules`. Please edit this file as needed when you write your solution.
  - More Snort rules examples are provided here:

    `http://stuff.is-a-geek.net/OnlineDocs/Security/Snort/chap2.html`

- Test the new Snort configuration.

```
sudo snort -T -i eth0 -u snort -g snort -c /etc/snort/snort.conf
```

  Wait for a few seconds until the process finishes. If the configuration is OK, you will see the following:

```
Snort successfully validated the configuration! Snort exiting
```

- Start Snort at an interface. Snort will now listen for traffic that arrives at that interface.

```
 sudo snort -A console -u snort -g snort -c /etc/snort/snort.conf -i eth1
```

- You need to wait a few seconds until snort starts. It will display several messages and then:

```
Commencing packet processing
```

  . At that point Snort is ready to process traffic at the interface you have indicated (*e.g.*eth1 in this example).

- After you have started Snort, use TCPreplay to play the traffic against the same interface that Snort listens. Example:

```
sudo tcpreplay -i eth1 evaluation.pcap
```

  Or if you want to replay the pcap multiple times:

```
 sudo tcpreplay -i eth1 --loop 10 test.pcap
```

- To replay using a different speed, *e.g.*10 packets per second:

```
 sudo tcpreplay -i eth1 --pps 10 test.pcap
```

- If you want to watch your traffic being replayed against the interface, you can open a separate screen and check with:

```
 sudo tcpdump -i eth1 -v
```

- Snort should generate alert messages on the screen (the same screen where you started Snort), using your rules.
- Please label each connection as legitimate or malicious. You can process the output of Snort appropriately to create a `txt` file that labels each connection as legitimate or malicious. Please see *Deliverables* below for the format.
- Please label each host as malicious bot, isolated infection or normal host (social). Please see *Deliverables* below for the format.
- Please submit your file with the Snort rules that you wrote as well. Please see *Deliverables* below.

4 Deliverables

For this project, there are three deliverables.

A) [Points: 85] Please provide a `txt` file `connections.txt` to report a label for each connection (related to hosts in 192.168.x.x/16) in the evaluation pcap. Each line has to have the following fields:

- Source IP, *e.g.*`10.0.0.1`
- Source Port, *e.g.*`80`
- Destination IP, *e.g.*`11.10.9.8`
- Destination Port, *e.g.*`765`
- Label: a) cnc (traffic that is generated when a bot communicates with the CnC server to receive commands/updates), b) infection (traffic that is generated when a host performs a malicious activity *e.g.*scan, ddos, spam), c) other (the traffic does not belong to the other two categories). For complete definitions of each type of traffic and for samples, please see sections above.

Format: Please follow the format below to report a connection as: a) cnc, b) infection, or c) other:

```
|Source IP|Source port|Destination IP|Destination port|Label|
|10.0.0.1|81|11.10.9.9|766|cnc|
|10.0.0.2|81|11.10.9.9|766|infection|
|10.0.0.3|80|11.10.9.8|765|other|
```

You only need to report each connection once. You don't need to report both directions of the connection. Each connection can only have one label. For example the following two lines are equivalent. You don't need to report both. They should have the same label.

```
|Source IP|Source port|Destination IP|Destination port|Label|
|10.0.0.2|81|11.0.9.9|766|cnc|
|11.0.9.9|766|10.0.0.2|81|cnc|
```

B) [Points: 5] Please label each host (in 192.168.x.x/16), in the evaluation pcap with a distinct role. Each host can only have one role. Please provide a `txt` file `hosts.txt`

- Host IP. *e.g.*`10.0.0.1`
- Role. *e.g.*Benign, Bot, or IsolatedInfection.

Format: Please follow the format below to report a distinct role for each host:

```
|12.13.14.15|Benign|
|12.13.14.16|Bot|
|12.13.14.17|IsolatedInfection|
```

C) [Points: 10] Please provide a `txt` file `rules.txt`, with the Snort rules that you wrote to perform the above tasks.

5 Important Notes

Readings on botnets behavior: Please read through the following papers, to get an understanding about what is a bot, and how botnets behave. Please note that we are not asking you to implement the proposed methodologies, *e.g.*a machine learning method to detect bots.

- "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation", Gu et. al. `http://faculty.cs.tamu.edu/guofei/paper/Gu_Security07_botHunter.pdf`
- "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic", G. Gu, J. Zhang, W. Lee, `https://www.usenix.org/legacy/event/sec08/tech/full_papers/gu/gu.pdf`
- "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection", G. Gu, R. Perdisci, J. Zhang, W. Lee, `http://faculty.cs.tamu.edu/guofei/paper/Gu_NDSS08_botSniffer.pdf`

Additional tools not allowed. For the purpose of the assignment you are not allowed to use any available tools, related to Snort or others. For example you are not allowed to use Snort pre-processors, that maybe publicly available, pre-compiled Snort rules, detection tools. etc. **You are expected to write your own Snort rules**.

Snort resources. Here you can find some examples of Snort rules, and some resources so that you get familiar with Snort rules. The purpose of these resources is to only to get you familiar with how Snort rules look like. If a Snort-preprocessor or any other tool is mentioned in these resources, you are not allowed to use it for this assignment. You are expected to write your own Snort rules, only.

- `http://archive.oreilly.com/pub/h/1393`
- `http://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm`
- `http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node27.html`
- `http://catalogue.pearsoned.co.uk/samplechapter/0131407333.pdf`

Writing Snort rules: You are expected to write your Snort rules that: 1) are not based on artifacts (*e.g.*block a specific port or IP or a specific protocol) and 2) have potentially long shelf life (*e.g.*if you were to deploy your rules in a real scenario you would not need to change your rules very frequently). For example, it is not OK to create a Snort rule based exclusively on artifacts of the sample traffic, *e.g.*a specific port number, or a specific IP address, or specific protocol. It is not OK, to flag a connection as malicious by writing a Snort rule that just flags a connection based on the src or dest IP address, or flags a connection just because it is HTTP, port 80. The reason behind this is that artifacts may change *e.g.*a port number or a specific IP address may change, Also, artifacts are not enough to capture the attack, the impact of the attack, the logic behind the attack and so on. It will help, if you think as a threat analyst. For example, would you ever create a Snort rule that flags the incoming traffic to a network, based exclusively on a port number or exclusively on an IP address? The answer is no, and this is because, these may easily change, and they may produce lots of false positives.

Example: Writing Snort rules to detect DDoS traffic. This is an example to give you an idea about how we can use our understanding of an attack, and write Snort rule with potentially long shelf life, to detect this attack. Intro reading to DDoS: `https://en.wikipedia.org/wiki/Denial-of-service_attack` Snort for DDoS: Please read this to get a general idea about how Snort can be used for this purpose. Please focus on sections 3 and 4. `http://www.ijeert.org/pdf/v2-i9/3.pdf`. After reading the above, one way to detect DDoS traffic is to monitor the rate of incoming traffic. Here is an example Snort rule based on traffic rate: `http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node35.html`

Disclaimer for background traffic. Please note that part of the traffic that is found in the evaluation pcap, and/or at the Sample pcaps is not generated by us. Specifically the legitimate background traffic is taken from publicly released datasets that are found here: `http://www.unb.ca/cic/research/datasets/flowmeter.html#ISCXFlowMeter`. In this link you can find the datasets and the academic publications that describe how the datasets were generated. The datasets closely resemble realist traffic. Part of this traffic might contain inappropriate content or language. We have taken extra measures and we have performed considerable effort to filter all traffic, based on commonly used inappropriate words. We have filtered the http payload, and URIs. Nevertheless, it might still be possible that some inappropriate content or words might have not been filtered entirely. In case you locate such content, we are letting you know, that it is not intentional, and we are not responsible for it. Also, to complete this assignment, you do not need (nor we ask you) to click on urls found inside http payloads.

Useful tools/commands.

- You can SCP the files from the VM to your local machine and view them using wireshark.
- You can install wireshark from here: `https://www.wireshark.org/`
- Wireshark display filters to view part of the traffic: `https://wiki.wireshark.org/DisplayFilters`
- Isolate traffic from the command line: `tcpdump -r file.pcap -n "host 192.168.4.120 and/or/not host 192.168.2.110" -w new.pcap`
- Merge two pcap files: `mergecap -w merged.pcap` $part_a.pcap$ $part_b.pcap$
- Fix header problems of pcap files: `editcap -F libpcap -T ether old.pcap new.pcap`
- How to scp a file named $file$ to the VM: `scp file mininet@192.168.56.101:/home/mininet`. If your VM has a different IP address than the above then you can find it by starting the VM, then log-in and then do: `sudo config`.
- The above scp command is just an example. Modify it accordingly. Resource for scp syntax: `http://www.hypexr.org/linux_scp_help.php`
- Python script to read/explore the pcaps: `view_pcaps.py`. Please download it from the $t - square$ project release page.