Project 4

Network Monitoring

Spring, 2021

Introduction

Goals:

The goal of this project is to introduce students to the techniques that help to differentiate malicious and legitimate network traffic. This is a task that network operators perform frequently. In this project, the students are provided with samples of malicious and legitimate traffic. They can observe how each type of traffic looks like. In the project folder, there is a pcap file that contains network traffic that originates from multiple hosts in the same network. This pcap file is a mixture of legitimate and malicious traffic. The students are asked to investigate the pcap file in network tools such as WireShark. Finally, the students are asked to use Snort and write their own Snort rules, which will differentiate malicious and legitimate traffic. In summary, the students are introduced to:

- Observing pcap samples of legitimate and malicious network traffic
- Using Snort and writing Snort rules to differentiate legitimate traffic from malicious traffic

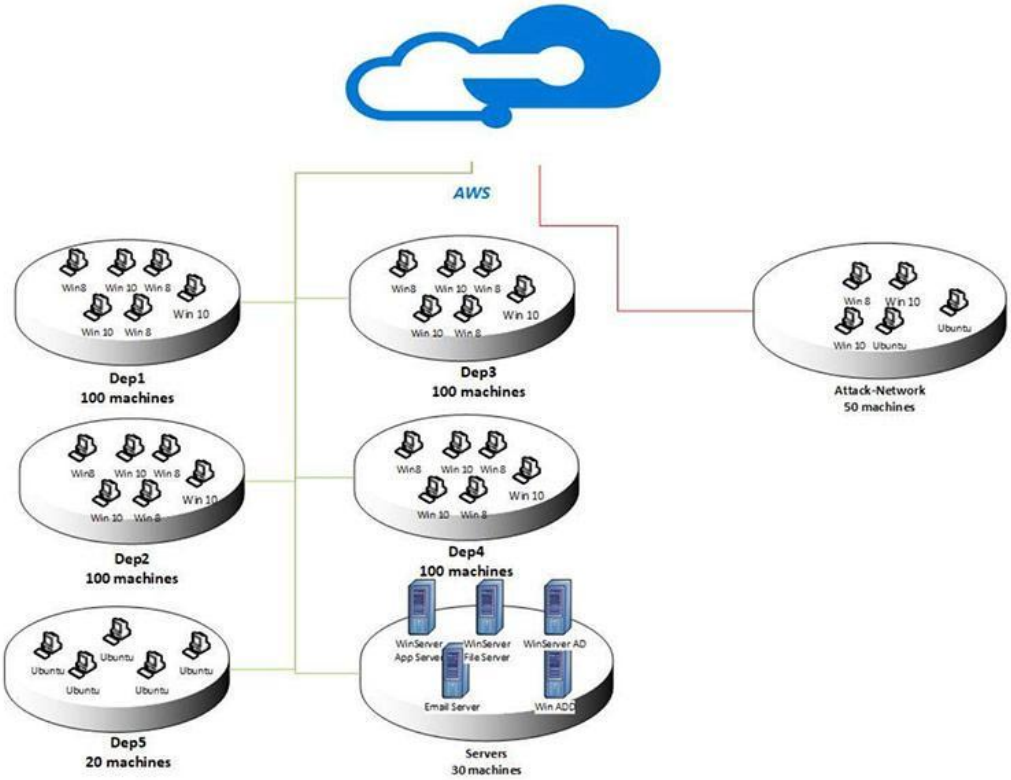

Figure 1: *Network setup for traffic collection.*

Definitions and Traffic Collection Set-up:

In this assignment, there are four attack scenarios. For each attack, a scenario is defined based on the implemented network topology, and the attack is executed from one or more machines outside the target network. Figure 1 shows the implemented network, which is a common LAN network topology on the AWS computing platform. The hosts are behind a NAT, and their IP addresses belong to a single /16: 172.31:0:0:/16. It also shows a visual representation of the network and our traffic collection set-up.

Types of attacks:

(i) Denial of Service (DoS):

In DoS, attackers usually keep making full TCP connections to the remote server. They keep the connection open by sending valid HTTP requests to the server at regular intervals but also keep the sockets from closing. Since any Web server has a finite ability to serve connections, it will only be a matter of time before all sockets are used up and no other connection can be made.

It is your task to find out how the DoS attack is present in the evaluation *pcap given to you.*

(ii) Bruteforce:

FTP/SSH is attacked via a Kali Linux machine(the attacker machine), and Ubuntu 14.0 system is the victim machine. There is a large dictionary that contains 90 million words that were used for the list of passwords to brute force.
It is your task to identify which one of them is present in the evaluation *pcap given to you.*

(iii) Web Attacks:

There are 3 possible web attacks, one of which would be present in your pcap.
(a) DVWA-based: Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is vulnerable. An attacker might try to hijack it.
(b) XSS-based: An attacker might try to launch an XSS attack.
(c) SQL Injection: An attacker might try an SQL injection attack.
It is your task to identify which ones of them are present in your evaluation *pcap.*

(iv) Botnet:

Zeus is a trojan horse malware that runs on Microsoft Windows. It might be presented in the pcap. It can be used to carry out many malicious and criminal tasks and it is often used to steal banking information by man-in-the-browser keystroke logging and form grabbing. It is used to install the Crypto-Locker ransomware as well. Zeus spreads mainly through drive-by downloads and phishing schemes. **The Ares botnet** might also be presented in the pcap. It is an open-source botnet and has the following capabilities:
(a) remote cmd.exe shell
(b) persistence
(c) file upload/download
(d) screenshot
(e) keylogging

Either Zeus and Ares could be present in your evaluation *pcap, it is your task to identify which one.*

Notes: the traffic doesn't have to cover all the attacks, and they can also cover multiple attacks for one category. For example, for web attacks, we can have both SQL injection and XSS. You need to find those in the evaluation pcap.

Sample traffic: For each type of traffic 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 in the evaluation pcap, which the students will need to label.

- **Legitimate background traffic:**
  For this exercise, we assume normal traffic to include HTTP, DNS. An example of normal (attack free) traffic can be found in:
  - sample_background.pcap
- **BruteForce:**
  - sample_bruteforce_ssh.pcap
  - sample_bruteforce_ftp.pcap
- **Botnet:**
  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_bot.pcap
- **Web Attack:**
  - sample_web.pcap
  - sample_xss.pcap
  - sample_sqlinjection.pcap.

  You should use multiple rules to cover all these attacks.
- **dos:**
  - We do **not** provide a sample. Please look at the example Snort rules on dos in the resources section.

Introduction Video (optional):

We made a short video about wireshark and the project(about 15 mins):
https://bluejeans.com/s/EiWzm3BxScx/

You will need to log in with your GaTech login information. When viewing the video, please slide right at the bottom of the screen to see the second screen in full screen mode.

We recommend that you read over the project description before viewing the video.
There are probably more filters(such as the filtering on the http method etc) that you can apply. We encourage you to read over the wire shark related links at the end of the project description to learn more about it.

Project Tasks (100 points):

The goal is to:
(i) Explore the given pcaps in Wireshark and identify the attack traffic patterns.
(ii) Write Snort rules to raise alerts to identify the attacks.

Towards this goal, please follow the tasks below:

- **Install Wireshark** in your local machine (we provide a VM but we recommend inspecting the pcaps via Wireshark on your local machine – instead of the VM as it is very CPU and RAM intensive).
- **Download**: The vm from this link.
  In case you are doing the project on your local machine. We also provide the evaluation pcap in the link so you don't need to scp it.
  MD5 hash of 2021SP4.ova: ee14a57afceb03046a4e7f524b3aac12

- **Import** the VM from this link. **Login to the VM using: login: student, password: project4**
- **Locate** the pcap files on your desktop. In this directory, you will find the sample pcaps and the evaluation pcap evaluation.pcap.
- **Make observations** on the pcaps:

Observe the sample pcaps to get an idea about how each type of malicious traffic looks like. You can use Wireshark or tshark to isolate some traffic. 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 is destined to this IP address). Also, you can combine filters using or/and.
You should use the attack descriptions above – to understand how these attacks should look like in network traffic.

- **Write Snort rules** – keep in mind, we are using **Snort3**, and not Snort2 – please make sure you use the Snort version installed in the VM.

You can write your Snort rules in any file. As an example, we'll write them in ~/Desktop/eval.rules

- You can now **run these snort rules** on the evaluation pcap using:
  sudo snort -c /usr/local/etc/snort/snort.lua -r ~/Desktop/evaluation.pcap -R ~/Desktop/eval.rules -s 65535 -k none -l . (The result will be in `alert_json.txt`. The dot at the end means the result will be generated in the current directory)
  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 the source IP 10.0.0.1 and any port to any destination IP and any destination port.
- You can then **view the Snort alert** log using sudo vim alert_json.txt.

- **Use EXACTLY ONE of the following strings as the alert message in the Snort rule:**
  1. DoS,
  2. Bruteforce,
  3. WebAttack,
  4. Botnet.

  For example, if you are writing a rule to detect ssh brute force, then the alert message should be "Bruteforce". **This will be used to grade your result – getting this part wrong can lead to a point loss.**

Deliverable/Rubric:

For this project, you should submit **two files**

- **eval.rules** - your Snort rules file. You are not allowed to hardcode a single IP in their rules. Instead, you should specify subnets and use the features of the attacks to capture them.
- **connections.txt** - the result file generated by running `python3 cal_unique_connection_2021.py alert_json.txt`. The script can be downloaded here

Notes: Don't zip the file. Just upload them as separate files. Make sure the filename is correct.

How to validate your answer:

1. We consider a connection to be "src_ip:src_port:dest_ip:dest_port". You can utilize the `cal_unique_connection.py` to check the unique connections of your alert_json.txt. You can compare the number of Dos/BruteForce/WebAttack/Botnet you got with the statistics above. If the number is close, you are likely on the right track.
2. You can view the pcap file in Wireshark to confirm you are finding the right connections.
3. Last, you can verify your result by submitting your answer to the gradescope (See steps 4&5). As the number of trials is limited, perform step 2 first!
4. We have provided you a way to verify your results on the Gradescope. You need to upload your **connections.txt** and you will see your current score. Please note that this is only for you to verify the results and you still need to upload your result on canvas. And you can verify your results for **at most**

10 times! Uploading more than 5 times in gradescope will tell you that you cannot verify your result anymore.

5. Running a single snort rule against the evaluation pcap can get a different result when you run it along with other snort rules. This is related to the limitation of Snort. So please run all your rules together to get the result we want.

Grading:

- The snort rules file you write would be run in the autograder.
- There are 4 attack categories as described above, each of which carries 25% grade weight.
- For each attack category, your grade = #correct alerts / (#real correct alerts + #incorrect alerts). Therefore, if you raise alerts for packets that are benign as one of the attack categories (false positives), you will lose points for that attack category. Also, if you miss raising an alert for an attack packet, you will lose points for that category.
- We will calculate your final grade for each category (which has 25 points as a full mark) by #final grade = min(25, #grade-for-one-category * 125%), which means if you catch **80%** of the attacks correctly, you can get full marks for the category.

Statistics for each type of unique connections (Important!):
Bruteforce: 3673
DoS: 8095
WebAttack: 40
Botnet: 47621
(The number might be a little different when you try to find it in Wireshark. Use the number that Snort gives you)

We consider a connection to be "src_ip:src_port:dest_ip:dest_port". run "**python3 ~/Desktop/cal_unique_connection_2021.py yourAlertFile**" to check the unique connections of your alert_json.txt and generate the results in `connections.txt`. If your alert JSON file is generated in the home directory, you might need to add sudo in front of your command.

FAQs

1. Do I get partial credits if the snort rule is not 100% correct?

- Partial credit is awarded according to how many false positives are detected. Please refer to the grading algorithm above.

2. Do you provide sample pcaps for dos?

- No, we don't provide sample pcaps for dos because you can observe dos from the evaluation pcap. A good start is reading some existing snort rules for detecting dos attacks to get a sense of it.

3. Do we care about UDP packets?
-You can ignore UDP packets. You can look at UDP as well, but it won't make much of a difference - UDP is mostly for negotiation protocols/DNS, etc.

4. After grading this assignment, will you release the correct answers?
-No, we don't, since similar projects run across semesters.

5. Should we ignore instances of ICMP and IP?
- You can ignore ICMP & IP

6. Is Tshark/Wireshark installed on the virtual machine?
- Yes. If the VM is too slow for you, I would recommend you install pcap reading tools on your local machine and use the VM only for Snort.
evaluation.pcap is a very large file so it can take some time to load. You may need to increase the

amount of RAM available to the system to get it to display properly.

7. Looks like I can complete the assignment without using VM?
   -Correct, you can complete without the VM but make sure your snort file works with the version installed in the given VM.

8. What constitutes a connection?
   -A "Connection" is identified by its Source Address, Destination Address, Source Port, and Destination Port.

9. Any hints on what we should look for when trying to identify SPAM-ing? SMTP connections?
   -You may want to look at packets within a certain time frame.

10. Can attacks be only from one or two machines or it should be from massive bot machines?
    -It can be from any number of machines. The exact number is not relevant.

11. Should we include an IPV6 connection? or only IPV4?
    - Only IPV4

12. We don't need to zip the file when we submit right?
    -No, you just need to submit the two files separately: eval.rules and connections.txt.

13. Is it possible to get 100% just using rules derived from samples?
    -We can't confirm or deny. :-)
    The sample pcaps are there to give you a good idea, but we don't claim they're representative. You should learn the pattern in the sample pcap and try to find related or similar patterns in the evaluation pcap. Always manually verify that what you find is correct.

14. The instructions state "Do not use any preprocessors." So that means we cannot use "flow" or "flowbits" in our rules because those keywords come from the Stream preprocessor. Correct?
    - Yes. You should not need flow or flowbits in this project.

15. How can I make sure that I am not using a preprocessor?
    -If you don't change anything in your config file you will be fine.

16. I am getting some json error/json decoder error
    - Yes that means your json file is too big and maybe you need to put more constraints on your snort rule to generate a smaller json file.

Resources:

Readings on botnets behavior: Please read through the following papers, to get an understanding of 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, http://faculty.cs.tamu.edu/guofei/paper/Gu_NDSS08_botSniffer.pdf
- *"BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection"*, G. Gu, R. Perdisci, J. Zhang, W. Lee, https://www.usenix.org/legacy/event/sec08/tech/full_papers/gu/gu.pdf

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 only to get you familiar with how Snort rules look like. You are expected to write your own Snort rules.

- https://usermanual.wiki/Document/snortmanual.760997111/view
- https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/596/original/Rules_Writers_Guide_to_Snort_3_Rules.pdf

Example: Writing Snort rules to detect dos traffic: This is an example to give you an idea about how we can use our understanding of an attack, and write Snort rules with potentially long shelf life, to detect this attack. Intro reading for dos: https://en.wikipedia.org/wiki/Denial-of-service_attack. Snort for dos: 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 dos 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

Useful tools/commands:

- You can SCP the files from the VM to your local machine and view them using Wireshark.
- SCP: http://www.hypexr.org/linux_scp_help.php
- Redirecting a program's output: http://linuxcommand.org/lc3_lts0070.php
- You can install Wireshark from here: https://www.wireshark.org/
- Wireshark display filters to view part of the traffic: https://wiki.wireshark.org/DisplayFilters
- How to scp a file named `file` to the VM: `scp file student@<VM's ip>:/home/student`. 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: `ip a`.
- The above scp command is just an example. Modify it accordingly. Resource for scp syntax: http://www.hypexr.org/linux_scp_help.php

Subnet:

- **Why is 172.31.0.0/16 a subnet?**
  Because it uses CIDR notation. CIDR and subnetting are virtually the same thing.
- **What's CIDR?**
  CIDR is Classless inter-domain routing. It is the /number representation. In this case, we have /16
- **What does /16 mean again?**
  /16 represents the **subnet mask** of 255.255.0.0

  If you convert 255.255.0.0 into binary, you will see 16 1's and that's where the number 16 comes from.
  Of course, I can't remember all those conversions for all netmask. There is a cheat sheet:

	Hosts	Netmask	Amount of a Class C
/30	4	255.255.255.252	1/64
/29	8	255.255.255.248	1/32
/28	16	255.255.255.240	1/16
/27	32	255.255.255.224	1/8
/26	64	255.255.255.192	1/4
/25	128	255.255.255.128	1/2
/24	256	255.255.255.0	1
/23	512	255.255.254.0	2
/22	1024	255.255.252.0	4
/21	2048	255.255.248.0	8
/20	4096	255.255.240.0	16
/19	8192	255.255.224.0	32
/18	16384	255.255.192.0	64
/17	32768	255.255.128.0	128
/16	65536	255.255.0.0	256

Wait, what's a subnet mask?

Feel free to read this link if you want to know more:

https://avinetworks.com/glossary/subnet-mask/

Important Notes

Disclaimer for background traffic. Please note that the traffic that is found in the evaluation pcap, and/or at the Sample pcaps is not generated by us. The dataset closely resembles 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 do we ask you) to click on URLs found inside http payloads.

Additional tools are not allowed. For 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 preprocessors that may be publicly available, pre-compiled Snort rules, detection tools. etc. **You are expected to write your own Snort rules.**