COMS 4180 Network Security Written Assignment 4

Due Wed, April 18, 2018, 10:00pm Eastern time.
50 points
The homework is to be done individually.
NO LATE HOMEWORK WILL BE ACCEPTED. The answers will be provided on the 20th, which is the last lecture before the exam.

What to submit: A text/pdf/word file with your answers. Submit via courseworks.

1. 20 points (5 for part a, 10 for part b, 5 for part c)

Scenario: there is a Linux system on which runs 10 processes performing some task along with typical operating system processes. The server is up 24 x 7. The only user on the system is the administrator, who logs in as root periodically to perform routine checks and maintenance. The administrator sometime uses a terminal connected to the system and other times logs in remotely. The system is connected to the internet and once every 24 hours sends data to a server at another location. The internet connection is also used to download and install updates on the system.

a. First do a small experiment with top and netstat on your VM or any Linux system you have.

Create a shell script repeat.sh containing the following. Replace the /bin/bash line with the path to bash on your system if the path differs. "which bash" will give you the location. wget should be present by default. If it is not installed on your system, you will need to install wget.

```
#!/bin/bash

# command line args are <number of times send get> <sleep
interval between gets>
cnt=1
max=$1
sleepinterval=$2

# send get to www.example.com max times
# sleeping sleepinterval seconds between each
# the response is saved to index.html.<number> files, these
# are immediately removed each time
while [ $cnt -le $max ]
do
   echo $cnt
   wget www.example.com
   ((cnt++))
   echo "retrieved webpage, removing index.html files"
   rm index.html*
   sleep $sleepinterval
done
echo "done"
```

repeat.sh will send a get to www.example.com max times, sleeping sleepinterval seconds between each time. max and sleepinterval are command line arguments.

Example usage: The following will send 5 get messages with 4 seconds between each message.

./repeat.sh 5 4

In two other terminal windows, run the following, where <interval> is in seconds. The smallest interval watch allows is 0.1 seconds.

watch –n <interval> “netstat –an”

watch -n <interval> "ps -aef | grep "wget"

What indication do you see that web pages were requested?

If the script was able to close and teardown the socket immediately after each wget, would you see any indication of wget running?

b, Using only the data available from the netstat, ps and top commands, describe how an IDS can monitor the Linux system for suspicious connections and usage. Include how often to collect data, what data (from that produced by netstat and top) to use and what statistics to compute. Does this data by itself allow an IDS to detect any malicious process (why or why not)? Also discuss the likeliness of false positives and false negatives.

c. Suppose the developers of the software running on the system are given logins to the system. How would this impact the IDS collection and analysis of the data and how may it impact the accuracy of the results?

2. 5 points

4 snort rules are shown below. EXTERNAL_NET and HOME_NET are defined variables and indicate anything external to the network being protected and the network being protected, respectively. SSH_PORT is also a defined variable and indicate the port used for SSH connections. Their specific values do not matter for this problem.

a. (1 point each) What traffic (in terms of addresses, ports, direction) is each of the 4 rules alerting on?

b. (1 point) What content is rule 4 looking for in the packet?

Rule 1

alert udp $EXTERNAL_NET 3345 -> $HOME_NET 3344 (msg:"Rule 1"; sid:162; rev:10;)

Rule 2

alert tcp $HOME_NET 666 -> $EXTERNAL_NET any (msg:"Rule 2"; sid:158; rev:10;)

Rule 3

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"Rule 3"; sid:362; rev:20;)

Rule 4

alert tcp $EXTERNAL_NET any -> $HOME_NET $SSH_PORT (msg:"Rule 4"; content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|"; rawbytes;sid:1325; rev:10;)

4. 10 points

In the paper oakland2013-peekaboo.pdf *Peek-a-boo I See You: Why Efficient Traffic Analysis Countermeasures Fail*, what are the authors able to identify (what is it that a client receives can be identified) via traffic analysis? Do any of the countermeasures they discuss prevent the identification? For

each countermeasure they tested, briefly describe (1 to 3 sentences) how they are able to thwart the countermeasures or how the countermeasure prevents the identification. For example, if they are able to thwart the countermeasure, what in the traffic were they able to measure to perform the identification.

5. 5 points (points for each part are all or nothing, no partial credit)

In wright06a.pdf:

a. (1 point) What transport layer protocol was used in the traffic the authors analyzed?

b. (2 points) What three features from each packet did the authors use in their analysis?

c. (2 points) What were the 8 applications/protocols for which they collected traffic?

6. 10 points total The following involves writing simple iptables rules.

a. (4 points) Write the answer to the following, do not do it on your Ubuntu VM. What would you type on the command line to add rules to iptables to allow DNS traffic (port 53) and block all other UDP traffic? You do not need to include how to save the changes so they persist past a reboot.
b. (6 points) Do the following on your Ubuntu VM. You may want to take a snapshot of your VM before beginning and revert back to the snapshot after finishing this exercise in case you make a mistake in the iptables rule or cannot clear the iptables rules. Copy what you typed on the command line and the output and include it into your answer for this question.

Use the netstat command to check if anything is using a port in the 40000 to 41000 range. If yes, use a different port range than what is listed below (it can be a smaller range that doesn't not include any ports currently in use but it must include at least 3 ports). Do not include the netstat output in your answer.

Before making any edits, show (use the verbose option when listing the rules) what is currently in iptables.

In the order listed below, add rules via the command line to do the following:

i) block all incoming traffic to ports in the range 40000-41000.
ii) allow all incoming traffic to port 40200 (or one of the ports in the range you used if your range differs). You must make sure the final order of your rules allows traffic on port 40200 and blocks it on all others in the range. One of the examples at https://help.ubuntu.com/community/IptablesHowTo can be used as guidance on how to insert a new rule in the correct order.

When done adding the 2 rules, show (use the verbose option when listing the rules) what is in iptables.

When you are done, either flush iptables (if it was empty when you started), individually delete the rules or revert to the snapshot of your VM.