**Scripting Project 2 – Due 3/31 11:59 pm CST**

The sample output video is the definitive word on what to do. Your script must look like and run like mine. Remember that your script must run to be graded, so it’s better to fully finish a function rather than halfway complete them all. You have done everything required for this project in previous tutorials and homework exercises.

The file alert\_full\_short.pcap.tgz is available for you to git pull into your home directory. The goal of your script will be to parse that file into a more usable file and provide some reporting on that data.

The script will unpack the tgz file in the home directory and parse the data into the file alert\_full\_short\_cleaned.csv that you will create.

Start with project\_2\_starter.sh that has the string of major descriptors. Be sure to put your code in this order:

Shebang

Descriptor string

Menu choice 1 function

Menu choice 2 function

Menu choice 3 function

Menu choice 4 function

Main code body:

Create major descriptors array

Name strings

Data file check

Menu

1. Parse alert data

2. Major descriptors

3. Classifications

4. Clean up and exit

**Name strings**: make a static variable MY\_NAME and set it to your first and last name (e.g. ‘Jim Burkman’). Split that variable to make a variable for your last name and for your first name. Use those variable when making the final tgz file.

**Data file check**: actively check to see if alert\_full\_short.pcap.tgz is in the home directory. If it is not, tell the user to put the file there and to restart the script. Exit the script. If the file is there, unzip the contents to the home directory.

**Menu**: Use a single echo statement for the menu then IF/ELIF/ELSE for the options. Catch invalid options and have the user try again.

**Main code body:** Make the , name strings and check for the tgz file.

**Parse alert data**: Our goal here is to read each line of data from the pcap file and pull out data for variables of interest. We will then write that good data to a new file in your student directory. The data file is messy. Be sure to follow my guidance. Start with a message to use user asking them to be patient. Open your good data file in your student directory and name it alert\_full\_short\_cleaned.csv. Write the header, which consists of Date, Time, Priority, Classification, Description, Packet Type, Source IP, Source Port, Destination IP, and Destination Port.da

The parsing flow follows Project 1. When comparing the file line to certain descriptors be sure to escape characters \* [ ] > by putting a \ in front of them. Like if [[ $cat == \*\[\\*\>\* would find the string \> in any location of the line. Use cut, tr and slices to get the data without any extra spaces. For checking two conditions the format is if [[ $cat == \*a\*]] && [[ $dog == \*b\* ]];then. For ip addresses without ports use the word “unspecified” for the data. Write the header with the words Date, Time, Priority, Classification, Description, Packet Type, Source IP, Source

Port, Destination IP, Destination Port. The cleaned file has a header, a blank line at the end, and 21,612 lines of data. It is about 22 KB in size. Be sure that every data element with spaces has those replaced with underscores.

**Major descriptors:** The goal of this function is to allow the user to enter one or more beginning characters of a major descriptor to show some statistics about alerts related to that major descriptor (not any letters anywhere). The major descriptor is part of the capitalized start of the description. We will match user input with an array of major descriptors that you will build from the string of major descriptors. If multiple descriptors are returned then we will ask the user to try again. If no major descriptors are returned we will inform the user then have them try again. If only one major descriptor is returned we will read the clean data file and collect all the full descriptions that match into a list. This list must only have unique values. We will tell the user the number of matches and wait for input to show those matches. Be sure that everything the user sees and inputs allows for spaces but that your array and clean data uses underscores instead of space. Follow the Project 1 solution for the general flow. Use the command “exit” to end the script.

**Classifications:** Ensure that your output matches mine.

**Clean up and exit:** Set a variable equal to your last name, comma, first name. Replace the comma with an underscore then add .tgz. Use the resulting variable to check to see if the (Burkman\_Jim.tgz) file exists. Remove it if it does exist. Then use that same variable to create a tgz file with the cleaned data and alert\_full\_short.pcap in the home directory. Delete alert\_full\_short.pcap and alert\_full\_short\_cleaned.csv. Leave alert\_full\_short.pcap.tgz in the home directory.

The guidance here is made in good faith by reviewing my solutions. Errors are likely. All questions about the veracity of these instructions must be posted to the appropriate discussion forum. My answers will only be posted in that same forum.

Helpful facts:

* Look at my output closely. Your numbers must match.
* Use the Project 1 solution to help with logic flow.
* I recommend opening your good data file in Excel every so often then sorting columns to quickly eyeball and weird problems.

Post any questions needed to clarify the project requirements to the Project 2 discussion forum. Do not post “how do I” or “I solved it by”. This is an exam.