Specification for Lab assignment 1

DV1457: Programming in the UNIX environment

Created by Stefan Axelsson
Modified by Martin Boldt
Department of Computer Science and Engineering
Blekinge Institute of Technology

2019-09-05

1 Background

In this lab you will write a script that processes a log file and extracts useful information summaries from it, a task that is common in system administration work. This is also a task that scripting languages shine at (Indeed, the Perl language is named after this task "Practical Extraction and Report Language). If you want consideration for a pass with distinction grades, i.e. A or B, you will also have to implement an optional feature in the log analyzer. For more information see subsection 2.2

The log file in question is a web server log file from the small, portable and secure webserver thttpd[1]. It follows a standard format that is common in the UNIX world, that is, it's a flat text file with one record per line and fields delimited by whitespace or other characters.

Your task is to write a script that answers the following questions, the idea is that this script will be used by sys admins to extract and summarise relevant information about the operation of the webserver so that they can check for anomalies, i.e. things that stand out from the ordinary, and investigate further if necessary. This is in fact a very simple and common form of intrusion detection (just reading the logs won't work, there's just too much).

The logfile consists of lines where each line represent one access request. It pretty much follows the unified Apache combined log format. The fields, describe the following: IP number the request originated from; the ident answer from the originator (always '-'); the username of the requester as determined by http authentication; date and time the request was processed; the request line as it was received from the client in double quotes; http status code that was sent to the client; number of bytes that was transferred to the client; referer page (from the client); user agent string (from the client). See e.g. the Apache documentation for more details (`http://httpd.apache.org/docs/1.3/logs.html`). Note that the fields are not well delimited (i.e. there is no reserved character to separate fields), this is unfortunately common when it comes to log files and a problem you have to contend with.

[1] `http://www.acme.com/software/thttpd`

2 Your task

You will write a shell script that reads a log file and answers the following questions:

- Which IP addresses makes the most number of connection attempts?
- Which IP addresses makes the most number of successful connection attempts?
- What are the most common result codes and where do they come from (IP number)?
- What are the most common result codes that indicate failure (no auth, not found etc) and where do they come from?
- Which IP number get the most bytes sent to them?

For all the above the answer should be in the form of a sorted list with the largest number first (i.e. topmost). The user should be able to cap all answers, i.e. by asking for only the 'x' topmost results. With no cap all available answers should be presented.

The arguments to the shell script should be coded as such:

```
log_sum.sh [-n N] (-c|-2|-r|-F|-t) <filename>
-n: Limit the number of results to N
-c: Which IP address makes the most number of connection attempts?
-2: Which address makes the most number of successful attempts?
-r: What are the most common results codes and where do they come
from?
-F: What are the most common result codes that indicate failure (no
auth, not found etc) and where do they come from?
-t: Which IP number get the most bytes sent to them?
<filename> refers to the logfile. If '-' is given as a filename, or
no filename is given, then standard input should be read. This
enables the script to be used in a pipeline.
```

In the above '|' denotes choice, one of. Furthermore, '[]' denotes that all within the square brackets is optional, while normal brackets '()' denote a mandatory parameter. So, e.g., "[-n N]" means that the parameter -n can be given, but does not have to, while '(-c|-2|-r|-F|-t)' indicates that exactly one of the parameters -c, -2, -r, etc. must be given. If a mandatory parameter is not given, this is an error condition.

The output format should be in the form:

```
-c: xxx.xxx.xxx.xxx yyy where yyy is the number of connection attempts
-2: xxx.xxx.xxx.xxx yyy where yyy is the number of successful attempts
-r: yyy xxx.xxx.xxx.xxx where yyy is the result code, one ip per line
-F: yyy xxx.xxx.xxx.xxx where yyy is the result code indicating failure,
one ip per line
-t: xxx.xxx.xxx.xxx yyy where yyy is the number of bytes sent from
the server
```

xxx.xxx.xxx.xxx is the IP address in dotted quad form. Yyy is an integer (with between 1 and the required number of digits, i.e. not just three digits). You are allowed to insert tabs and or whitespace between the elements of the output according to taste. (But not within the elements, i.e. within an IP address). For -F and -r you are allowed to output multiple lines with the same result code or count, but the groups must still be sorted. There should still only be one ip address per line.

2.1 Example usage

- **Example 1**

```
$log_sum.sh -n 10 -c thttpd.log
213.64.237.230    2438
213.64.225.123    1202
213.64.141.89      731
213.64.64.53       591
213.64.214.124     480
213.64.55.182      429
213.64.246.37      400
213.64.153.92      336
213.64.100.52      336
213.64.19.224      272
```

That is to say, we're asking for a sorted list of the top ten most connection-intensive IP addresses, most prolific first. Example 1 can be used as a test case as it is correct given the provided thttpd.log file.

- **Example 2**

```
$log_sum.sh -r -n 3 thttpd.log
404    127.0.0.1
404    xxx.xxx.xxx.xxx
404    xxx.xxx.xxx.xxx

200    xxx.xxx.xxx.xxx
200    xxx.xxx.xxx.xxx
200    127.0.0.1

403    xxx.xxx.xxx.xxx
403    xxx.xxx.xxx.xxx
```

Example 2 is only a fictitious case. Here, the status code groups have to be sorted by which status code group appears more often and the IP addresses have to be sorted by occurrence within each of the groups. Further, IP address can show up several times in the output for different status codes. However, within each status code group, unique IP addresses have to be presented. Note that these formatting rules also apply to the output of -F.

- **Example 3**

```
$log_sum.sh -n 3 -t thttpd.log
xxx.xxx.xxx.xxx    19109572
xxx.xxx.xxx.xxx    18043610
xxx.xxx.xxx.xxx     1915720
```

Example 3 is also a fictitious case. The results for -t have to be sorted based on the sum of bytes.

2.2 Optional part for A and B grades

If you want to be able to get the higher grades A or B you need to fulfil the following optional task. That is to get the DNS name associated with each IP number and then double check that DNS name against the DNS blacklist available at the course page (filename is `dns.blacklist.txt`). This feature should be available as argument `-e` in the script's argument list and it should be possible to use `-e` in combination with other arguments '(-c|-2|-r|-F|-t)'. If argument `-e` is specified all DNS names (related to the IPs in the log) should be compared with each entry in the blacklist. If a match is found, then the end of the output of such a line output should be Blacklisted". Otherwise, no additional output is needed.

Hint: only check one IP address once and save that value for later.

- **Example**

```
$log_sum.sh -n 200 -c -e thttpd.log
213.64.237.230    2438
213.64.225.123    1202
213.64.141.89      731
.                    .
.                    .
xxx.xxx.xxx.xxx    yyy  *Blacklisted!*
.                    .
.                    .
xxx.xxx.xxx.xxx    yyy
```

2.3 Presentation and limitations

The assignment should be presented in person with both group members in attendance on a lab slot. You will be required to show the code, let me run it with the test file you have been given and be prepared to discuss your work. Your lab report should be about one (1) A4 and state the name of the group members, any problems you have encountered, and how you went about solving the lab assignment, e.g. what major commands did you use etc. Write short, but to the point.

You're allowed to use all tools available to the shell script programmer, except you're not allowed to use awk for any major processing, i.e. using awk for printing, field selection etc. is OK, writing the "shell script" part of the lab completely in awk and just call that from a Bourne shell script is not OK.

2.4 Grading

You should demonstrate a clear understanding of the problem and detail your strategy of how to solve it, including drawbacks and advantages of the solution chosen. Your code should be clear, concise and to the point.

You are allowed to discuss problems and solutions with other groups, but the code that you write must be your own and cannot be copied, gleaned, downloaded from the internet etc. (You may of course copy idioms and snippets, such as `awk '{print $1}'` verbatim). However, you must be able to clearly describe each part of your script solution.

Good luck, and may the source be with you.

Typeset by LaTeX.