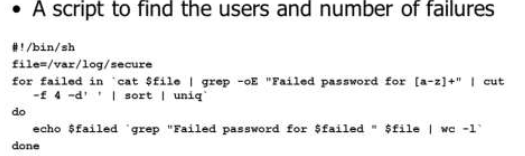
# BASH Lab 6 Solutions

The file that contains the logs we were given is auth.log. I put it in my home directory, so I just used access.log where the slides used /var/log/secure (location of the CentOS log file.)

## Failed SSH Login

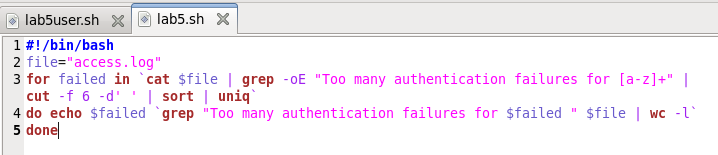
To find accounts that failed with the “Too many authentication failures” message, we can use the script from slide 5 almost exactly.

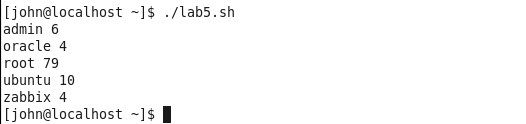


In line 2, file needs to be changed to the location of access.log.

In line 3, “Failed password for [a-z]+” needs to be changed to "Too many authentication failures for [a-z]+".

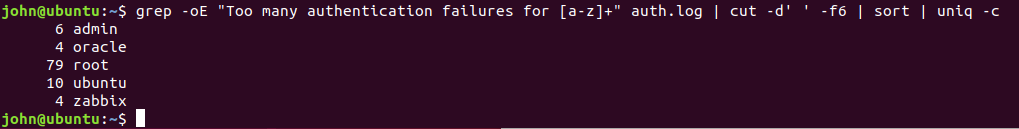
In line 4, the field number for the cut command changes from 4 to 6.

Here’s the new version.  
 

This is the result.  
 

Note: This is an actual log file containing a real SSH brute force attack. The first 4 user names are common. From [www.zabix.com](http://www.zabix.com), “Zabbix is the ultimate enterprise-level software designed for monitoring availability and performance of IT infrastructure components. Zabbix is open source and comes at no cost.”

# NOTE

I’m not sure why the slides used a for loop to solve this. It could have been solved just as easily with a single line of BASH.  


## Failed SSH Login invalid users

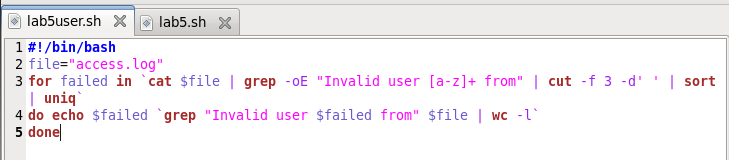
The next step was to see what other user names the bad guys tried, that weren’t on the server. The log entry for a user name the server doesn’t recognize is this.

Dec 11 16:51:25 a2bsensor sshd[2283]: Invalid user kyle from 221.194.57.246

Dec 11 16:51:25 a2bsensor sshd[2283]: input\_userauth\_request: invalid user kyle [preauth]

Dec 11 16:51:25 a2bsensor sshd[2258]: fatal: Read from socket failed: Connection reset by peer [preauth]

Dec 11 16:51:25 a2bsensor sshd[2283]: Received disconnect from 221.194.57.246: 11: Bye Bye [preauth]

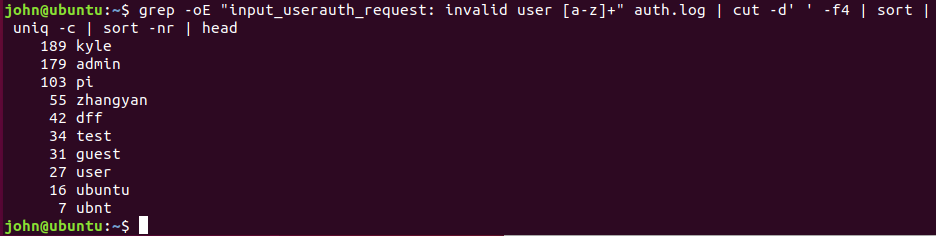
Let’s use the first line, since it contains both the user name the bad guy used, plus the IP address the bad guy came from. Again, minor modifications to slide 5 give us this. 

Just showing the first few results…

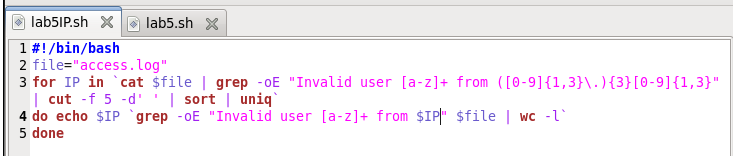
## 

# NOTE

Again, a one-liner will work.

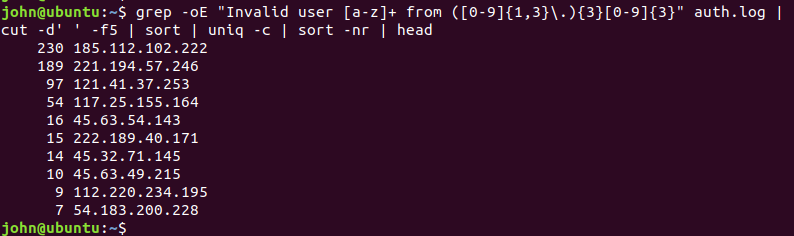


## Where are the attacks coming from?

Again, minor modifications to the script from slide 5 will solve the problem for us. The regex, ([0-9]{1,3}\.){3}[0-9]{1,3} was taken from slide 8 of Scripting & Parsing, and finds IP addresses.

Just showing the first few results… 

Or, as a one-liner,



Note: The “o” in grep -oE tells grep to only output the part that matched. I used it so there would be less to count when finding the field number for the cut command.

