

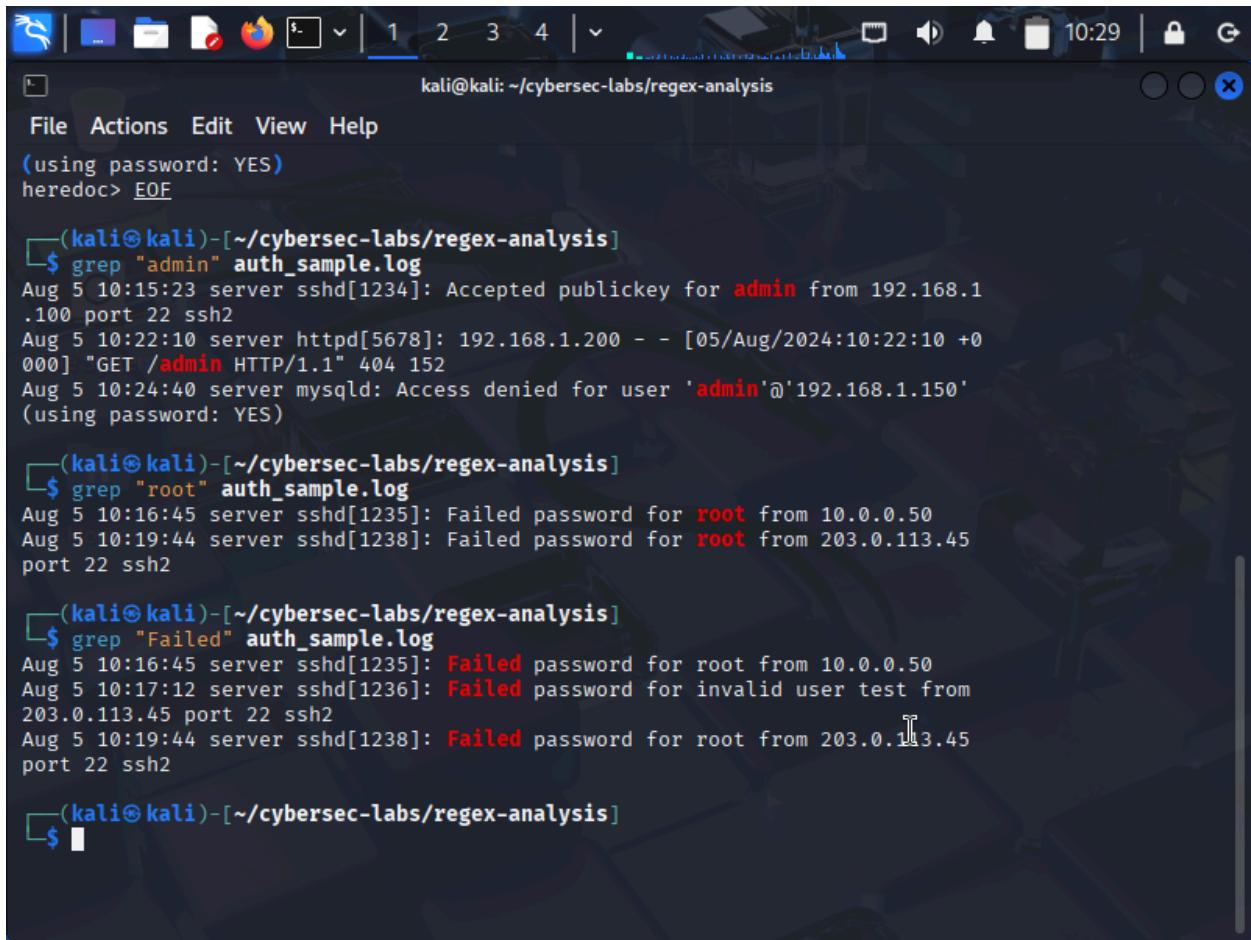
Lab 3: Linux Basics Part 3 - Regular Expressions and Log Analysis

Matthew Cox, COMP325-01, term (Fall 2025)

Introduction:

This lab focuses on practical regex applications, emphasizing real-world log analysis scenarios you'll encounter during incident response activities. In this lab, I will be able to understand and construct, and apply regex with grep for advanced log analysis and threat detection. I will be able to identify security-relevant patterns in log files and system outputs.

Body:



The screenshot shows a terminal window titled "kali@kali: ~/cybersec-labs/regex-analysis". The terminal displays several commands using the grep command to search for specific patterns in a log file named "auth_sample.log".

```
(using password: YES)
heredoc> EOF

[(kali㉿kali)-[~/cybersec-labs/regex-analysis]]
$ grep "admin" auth_sample.log
Aug 5 10:15:23 server sshd[1234]: Accepted publickey for admin from 192.168.1
.100 port 22 ssh2
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 - - [05/Aug/2024:10:22:10 +0
000] "GET /admin HTTP/1.1" 404 152
Aug 5 10:24:40 server mysqld: Access denied for user 'admin'@'192.168.1.150'
(using password: YES)

[(kali㉿kali)-[~/cybersec-labs/regex-analysis]]
$ grep "root" auth_sample.log
Aug 5 10:16:45 server sshd[1235]: Failed password for root from 10.0.0.50
Aug 5 10:19:44 server sshd[1238]: Failed password for root from 203.0.113.45
port 22 ssh2

[(kali㉿kali)-[~/cybersec-labs/regex-analysis]]
$ grep "Failed" auth_sample.log
Aug 5 10:16:45 server sshd[1235]: Failed password for root from 10.0.0.50
Aug 5 10:17:12 server sshd[1236]: Failed password for invalid user test from
203.0.113.45 port 22 ssh2
Aug 5 10:19:44 server sshd[1238]: Failed password for root from 203.0.113.45
port 22 ssh2

[(kali㉿kali)-[~/cybersec-labs/regex-analysis]]
$
```

Here I was able to create a sample log file. What you are currently seeing is just some basic character matching.

The screenshot shows a terminal window titled "kali@kali: ~/cybersec-labs/regex-analysis". The terminal displays several log entries from a server, including SSH, HTTP, and MySQL logs. The user runs a command to filter these logs using grep, specifically targeting lines where the timestamp contains digits and the log message contains digits. The grep command is: `grep "[0-9][0-9]:[0-9][0-9]" auth_sample.log`. The output of the grep command shows the same log entries as the full log, but only those that match the timestamp pattern. The terminal prompt at the bottom is `$`.

```
2 ssh2
Aug 5 10:18:33 server sshd[1237]: Accepted password for user1 from 192.168.1.101 port 22 ssh2
Aug 5 10:19:44 server sshd[1238]: Failed password for root from 203.0.113.45 port 22 ssh2
Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port 45123 [preauth]
Aug 5 10:21:30 server kernel: TCP: eth0: link up
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 -- [05/Aug/2024:10:22:10 +0000] "GET /admin HT
TP/1.1" 404 152
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 -- [05/Aug/2024:10:23:25 +0000] "POST /login.ph
p HTTP/1.1" 200 341
Aug 5 10:24:40 server mysqld: Access denied for user 'admin'@'192.168.1.150' (using password: YE
S)

└─(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "[0-9][0-9]:[0-9][0-9]" auth_sample.log
Aug 5 10:15:23 server sshd[1234]: Accepted publickey for admin from 192.168.1.100 port 22 ssh2
Aug 5 10:16:45 server sshd[1235]: Failed password for root from 10.0.0.50
Aug 5 10:17:12 server sshd[1236]: Failed password for invalid user test from 203.0.113.45 port 2
2 ssh2
Aug 5 10:18:33 server sshd[1237]: Accepted password for user1 from 192.168.1.101 port 22 ssh2
Aug 5 10:19:44 server sshd[1238]: Failed password for root from 203.0.113.45 port 22 ssh2
Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port 45123 [preauth]
Aug 5 10:21:30 server kernel: TCP: eth0: link up
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 -- [05/Aug/2024:10:22:10 +0000] "GET /admin HT
TP/1.1" 404 152
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 -- [05/Aug/2024:10:23:25 +0000] "POST /login.ph
p HTTP/1.1" 200 341
Aug 5 10:24:40 server mysqld: Access denied for user 'admin'@'192.168.1.150' (using password: YE
S)

└─(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$
```

In this image, you can see a few examples of case-insensitive matching and digit matching.

```
$ grep "[A-Z]" auth_sample.log
Aug 5 10:15:23 server sshd[1234]: Accepted publickey for admin from 192.168.1.100 port 22 ssh2
Aug 5 10:16:45 server sshd[1235]: Failed password for root from 10.0.0.50
Aug 5 10:17:12 server sshd[1236]: Failed password for invalid user test from 203.0.113.45 port 22 ssh2
Aug 5 10:18:33 server sshd[1237]: Accepted password for user1 from 192.168.1.101 port 22 ssh2
Aug 5 10:19:44 server sshd[1238]: Failed password for root from 203.0.113.45 port 22 ssh2
Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port 45123 [preauth]
Aug 5 10:21:30 server kernel: TCP: eth0: link up
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 - - [05/Aug/2024:10:22:10 +0000] "GET /admin HTTP/1.1" 404 152
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 - - [05/Aug/2024:10:23:25 +0000] "POST /login.php HTTP/1.1" 200 341
Aug 5 10:24:40 server mysqld: Access denied for user 'admin'@'192.168.1.150' (using password: YES)

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "^[A-Z][a-z][a-z]" auth_sample.log
Aug 5 10:15:23 server sshd[1234]: Accepted publickey for admin from 192.168.1.100 port 22 ssh2
Aug 5 10:16:45 server sshd[1235]: Failed password for root from 10.0.0.50
Aug 5 10:17:12 server sshd[1236]: Failed password for invalid user test from 203.0.113.45 port 22 ssh2
Aug 5 10:18:33 server sshd[1237]: Accepted password for user1 from 192.168.1.101 port 22 ssh2
Aug 5 10:19:44 server sshd[1238]: Failed password for root from 203.0.113.45 port 22 ssh2
Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port 45123 [preauth]
Aug 5 10:21:30 server kernel: TCP: eth0: link up
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 - - [05/Aug/2024:10:22:10 +0000] "GET /admin HTTP/1.1" 404 152
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 - - [05/Aug/2024:10:23:25 +0000] "POST /login.php HTTP/1.1" 200 341
Aug 5 10:24:40 server mysqld: Access denied for user 'admin'@'192.168.1.150' (using password: YES)
```

In this image, I tried a different method of taking a screenshot, just to see if it's any better than what I have previously done. But within this image, I used a letter range command that matches lines starting with three-letter month abbreviations.

```
kali㉿kali: ~/cybersec-labs/regex-analysis

File Actions Edit View Help

Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port 45123 [preauth]
Aug 5 10:21:30 server kernel: TCP: eth0: link up
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 - - [05/Aug/2024:10:22:10 +0000] "GET /admin HTTP/1.1" 404 152
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 - - [05/Aug/2024:10:23:25 +0000] "POST /login.php HTTP/1.1" 200 341
Aug 5 10:24:40 server mysqld: Access denied for user 'admin'@'192.168.1.150' (using password: YES)

File System
(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "[0-9][0-9]*\.[0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" auth_sample.log
zsh: Input/output error: grep

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "[0-9][0-9]*\.[0-9][0-9]*\.[0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" auth_sample.log
zsh: Input/output error: grep

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ ^[[200~grep "[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" auth_sample.log
zsh: bad pattern: ^[[200~grep

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" auth_sample.log
zsh: Input/output error: grep

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" auth_sample.log
zsh: Input/output error: grep

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$
```

I have no clue what I did wrong here. I will try again once I can get my Kali Linux to not crash.

The screenshot shows a terminal window titled "kali@kali: ~/cybersec-labs/regex-analysis". The terminal displays several command-line sessions:

- The first session shows a failed MySQL login attempt for user 'admin' from IP '192.168.1.150':

```
+0000] "POST /login.php HTTP/1.1" 200 341
Aug 5 10:24:40 server mysqld: Access denied for user
'admin'@'192.168.1.150' (using password: YES)
```
- The second session demonstrates matching IP addresses using a regular expression pattern: `^([A-Z][a-z])[a-z]`. It lists various log entries, including successful SSH logins and failed password attempts:

```
(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "^[A-Z][a-z][a-z]" auth_sample.log
Aug 5 10:15:23 server sshd[1234]: Accepted publickey for admin from
Aug 5 10:16:45 server sshd[1235]: Failed password for root from 10.0.0.50
Aug 5 10:17:12 server sshd[1236]: Failed password for invalid user test
Aug 5 10:18:33 server sshd[1237]: Accepted password for user1 from
Aug 5 10:19:44 server sshd[1238]: Failed password for root from
Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port
Aug 5 10:21:30 server kernel: TCP: eth0: link up
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 - - [05/Aug/2024:10:22:10
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 - - [05/Aug/2024:10:23:25
Aug 5 10:24:40 server mysqld: Access denied for user
```
- The third session shows matching IP addresses using a more complex regular expression pattern: `[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*`. It highlights specific IP addresses in red:

```
(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" auth_sample.log
192.168.1.100 port 22 ssh2
Aug 5 10:16:45 server sshd[1235]: Failed password for root from 10.0.0.50
from 203.0.113.45 port 22 ssh2
192.168.1.101 port 22 ssh2
203.0.113.45 port 22 ssh2
Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 - - [05/Aug/2024:10:22:10
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 - - [05/Aug/2024:10:23:25
'admin'@'192.168.1.150' (using password: YES)
```
- The fourth session shows a blank command line prompt.

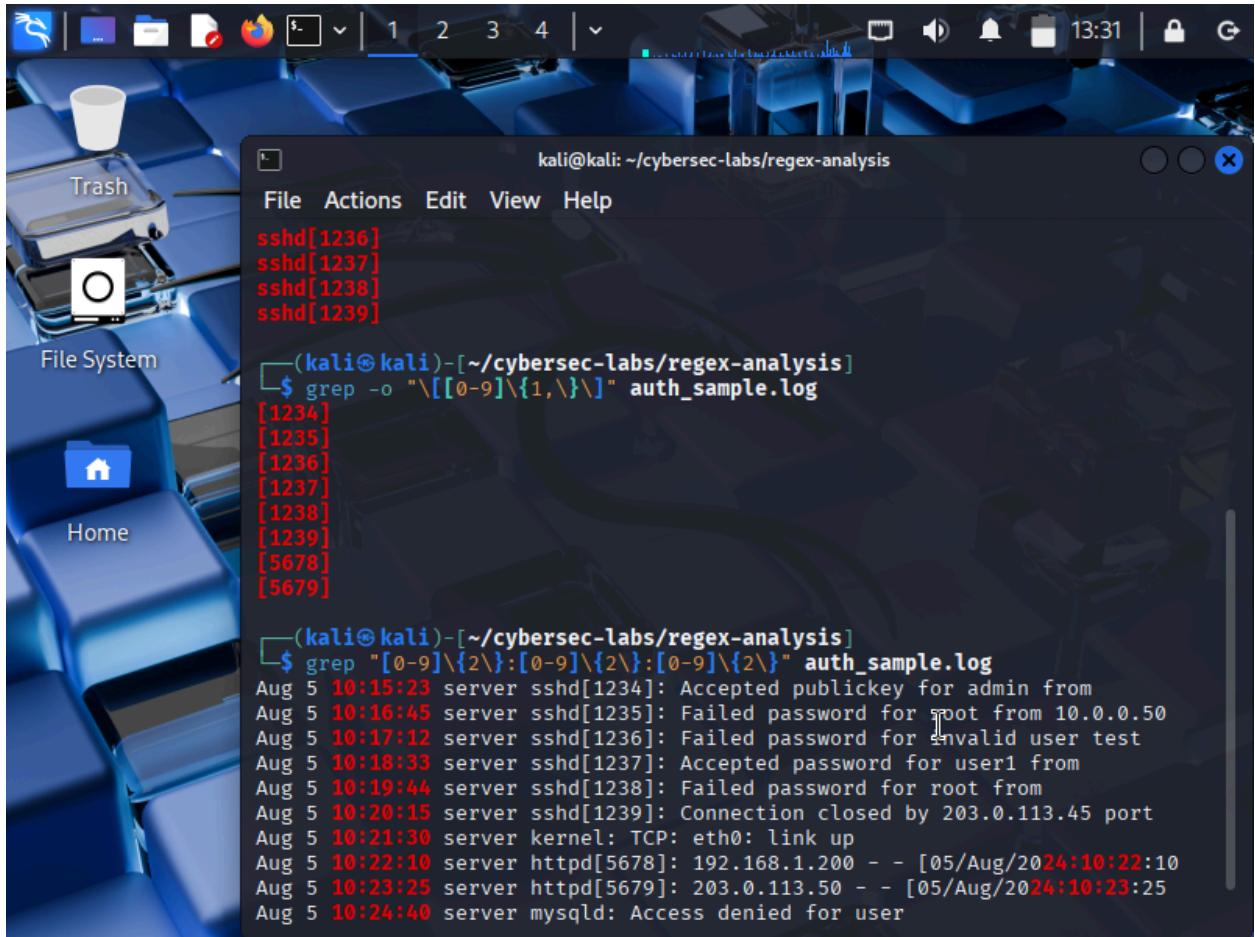
Following the previous image, I finally got it to work. I'd assume I might have missed a line within the sample log files. Or maybe it was an indentation error. But here is a successful IP address pattern matching example.

```
(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" auth_sample.log
192.168.1.100 port 22 ssh2
Aug 5 10:16:45 server sshd[1235]: Failed password for root from 10.0.0.50
from 203.0.113.45 port 22 ssh2
192.168.1.101 port 22 ssh2
203.0.113.45 port 22 ssh2
Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 -- [05/Aug/2024:10:22:10
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 -- [05/Aug/2024:10:23:25
'admin'@'192.168.1.150' (using password: YES)

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ echo "==== IP Address Analysis ===" > ip_analysis.txt
Home
(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ echo "All IP addresses found:" >> ip_analysis.txt

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -o "[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*" auth_sample.log
>> ip_analysis.txt
192.168.1.100
10.0.0.50
203.0.113.45
192.168.1.101
203.0.113.45
203.0.113.45
192.168.1.200
203.0.113.50
192.168.1.150
```

Here, I created an IP address. Now I can continue onto Exercise 3.



The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal window title is "kali@kali: ~/cybersec-labs/regex-analysis". The terminal content displays several grep commands and their results:

```
sshd[1236]
sshd[1237]
sshd[1238]
sshd[1239]

[(kali㉿kali)-[~/cybersec-labs/regex-analysis]] $ grep -o "[0-9]{1,\}\]" auth_sample.log
[1234]
[1235]
[1236]
[1237]
[1238]
[1239]
[5678]
[5679]

[(kali㉿kali)-[~/cybersec-labs/regex-analysis]] $ grep "[0-9]{2\}:[0-9]{2\}:[0-9]{2\}" auth_sample.log
Aug 5 10:15:23 server sshd[1234]: Accepted publickey for admin from
Aug 5 10:16:45 server sshd[1235]: Failed password for spot from 10.0.0.50
Aug 5 10:17:12 server sshd[1236]: Failed password for invalid user test
Aug 5 10:18:33 server sshd[1237]: Accepted password for user1 from
Aug 5 10:19:44 server sshd[1238]: Failed password for root from
Aug 5 10:20:15 server sshd[1239]: Connection closed by 203.0.113.45 port
Aug 5 10:21:30 server kernel: TCP: eth0: link up
Aug 5 10:22:10 server httpd[5678]: 192.168.1.200 - - [05/Aug/2024:10:22:10
Aug 5 10:23:25 server httpd[5679]: 203.0.113.50 - - [05/Aug/2024:10:23:25
Aug 5 10:24:40 server mysqld: Access denied for user
```

With the conclusion of Exercise 3, I was able to demonstrate basic qualifiers, process ID extraction, and time patterning successfully. I had some issues with successfully executing an improved IP address matching command. I seemed to get stuck in my terminal. But I was able to close it out and move on.

The screenshot shows a terminal window titled "kali@kali: ~/cybersec-labs/regex-analysis". The terminal displays several commands and their outputs related to grep and regular expressions:

```
(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "^$" test_file.txt
grep: unrecognized option '___'.
Usage: grep [OPTION] ... PATTERN [FILE] ...
Try 'grep --help' for more information.

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "^$" test_file.txt

Home
(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -v "^$" test_file.txt
Valid line
Another valid line

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -c "^$" test_file.txt
2

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -cv "^$" test_file.txt
2

(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$
```

In this assessment, I was able to grab the beginning and end lines. As I went to find specific lines, I entered the command, cat test_file.txt, which then froze my Kali Linux. I think it was just a goof. I was able to eventually close out Kali and reboot it. As you can see, I got it just fine the second time around.

A screenshot of a terminal window titled "kali@kali: ~/cybersec-labs/regex-analysis". The terminal shows several grep commands being run against a file named "security_events.log". The output includes logs from various sources like kernel, Firewall, and Suricata, and specific detections for failed password attempts, UNION SELECT attacks, and port scanning.

```
heredoc> Aug 5 10:40:30 server kernel: Firewall: IN=eth0 OUT= MAC=aa:bb:cc:dd:ee:ff SRC=203.0.113.200 DST=192.168.1.1 PROTO=TCP SPT=12345 DPT=1433
heredoc> Aug 5 10:41:15 server suricata: [1:2100498:7] GPL SQL Injection attack
[Classification: Web Application Attack] [Priority: 1]
heredoc> Aug 5 10:42:00 server fail2ban: WARNING [sshd] Ban 198.51.100.10
heredoc> EOF

└──(kali㉿kali)-[~/cybersec-labs/regex-analysis]
└─$ grep "Failed password.*from [0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}" security_events.log

└──(kali㉿kali)-[~/cybersec-labs/regex-analysis]
└─$ grep "GET.*\.\./..\.\.\." security_events.log
+0000] "GET /.../..../etc/passwd HTTP/1.1" 302 0

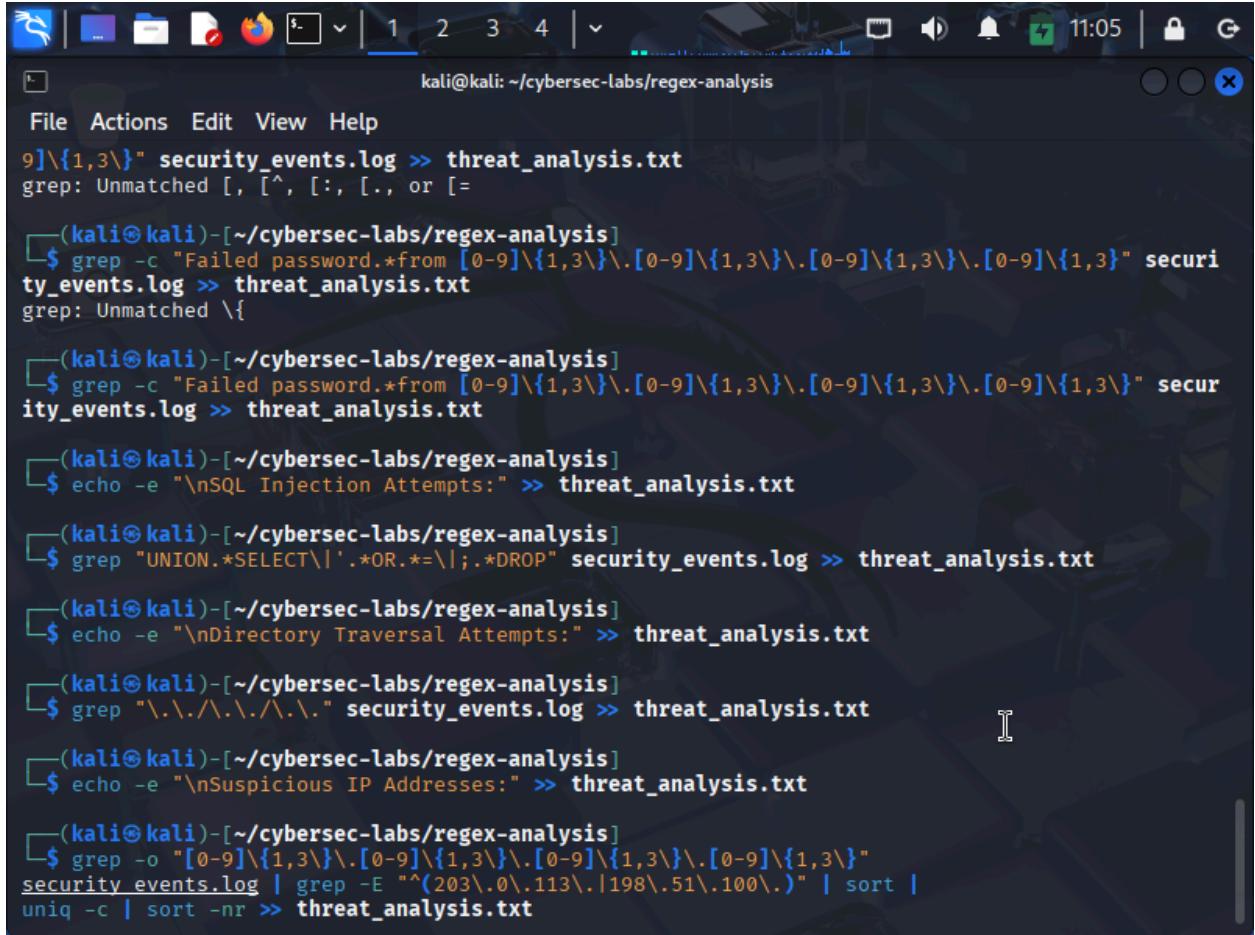
└──(kali㉿kali)-[~/cybersec-labs/regex-analysis]
└─$ grep "UNION.*SELECT" security_events.log
+0000] "GET /index.php?id=1' UNION SELECT * FROM users-- HTTP/1.1" 500 87

└──(kali㉿kali)-[~/cybersec-labs/regex-analysis]
└─$ grep "GET.*['\"]" security_events.log
+0000] "GET /admin/login.php HTTP/1.1" 200 1234
+0000] "GET /.../..../etc/passwd HTTP/1.1" 302 0
+0000] "GET /index.php?id=1' UNION SELECT * FROM users-- HTTP/1.1" 500 87

└──(kali㉿kali)-[~/cybersec-labs/regex-analysis]
└─$ grep "DPT=[0-9]\{4,5\}" security_events.log
SRC=203.0.113.200 DST=192.168.1.1 PROTO=TCP SPT=12345 DPT=1433

└──(kali㉿kali)-[~/cybersec-labs/regex-analysis]
└─$
```

I created a comprehensive security log successfully. Didn't seem to have any goofs this time. I got all of the correct outputs when running various detections and scans.



The screenshot shows a terminal window titled "kali@kali: ~/cybersec-labs/regex-analysis". The terminal contains a series of shell commands used to analyze security logs. The commands include grep, echo, and redirection operators (>>). The log files analyzed include "security_events.log" and "threat_analysis.txt". The analysis covers various threat types such as failed password attempts, SQL Injection Attempts, UNION SELECT queries, Directory Traversal Attempts, and suspicious IP addresses.

```
9]\{1,3\}" security_events.log >> threat_analysis.txt
grep: Unmatched [, [^, [:, [., or [=

  (kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -c "Failed password.*from [0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}" security_events.log >> threat_analysis.txt
grep: Unmatched \{

  (kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -c "Failed password.*from [0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}" security_events.log >> threat_analysis.txt

  (kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ echo -e "\nSQL Injection Attempts:" >> threat_analysis.txt

  (kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "UNION.*SELECT\|.*OR.*=\|;.DROP" security_events.log >> threat_analysis.txt

  (kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ echo -e "\nDirectory Traversal Attempts:" >> threat_analysis.txt

  (kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep "\.\./\.\.\.\.\." security_events.log >> threat_analysis.txt

  (kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ echo -e "\nSuspicious IP Addresses:" >> threat_analysis.txt

  (kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -o "[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}" security_events.log | grep -E "^(203\.0\.113\.|198\.51\.100\.)" | sort | uniq -c | sort -nr >> threat_analysis.txt
```

Above, I created a comprehensive threat analysis. I had a couple of spelling issues with the failed password command, but eventually figured it out.

```
kali@kali: ~/cybersec-labs/regex-analysis
File Actions Edit View Help

Aug 5 10:30:18 server sshd[1243]: Failed password for test from
Aug 5 10:30:19 server sshd[1244]: Failed password for guest from
+0000] "GET /admin/login.php HTTP/1.1" 200 1234
+0000] "POST /admin/login.php HTTP/1.1" 302 0

└─(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -E "(UNION|SELECT|DROP|INSERT|DELETE)" security_events.log
+0000] "GET /index.php?id=1' UNION SELECT * FROM users-- HTTP/1.1" 500 87
File System

└─(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -E "\(\.\.\.\|etc\|passwd\|proc\|sys\|\" security_events.log
+0000] "GET / ..../etc/passwd HTTP/1.1" 302 0

└─(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -E "DPT=(80|443|22|21|25|53|135|139|445|1433|3389)" security_events.log
SRC=203.0.113.200 DST=192.168.1.1 PROTO=TCP SPT=12345 DPT=1433

└─(kali㉿kali)-[~/cybersec-labs/regex-analysis]
$ grep -E "[0-9]{1,3}\.){3}[0-9]{1,3}" security_events.log
198.51.100.10 port 22 ssh2
Aug 5 10:35:22 server httpd[5680]: 203.0.113.100 -- [05/Aug/2024:10:35:22
Aug 5 10:35:23 server httpd[5681]: 203.0.113.100 -- [05/Aug/2024:10:35:23
Aug 5 10:36:45 server httpd[5682]: 203.0.113.100 -- [05/Aug/2024:10:36:45
Aug 5 10:36:45 server httpd[5682]: 203.0.113.100 -- [05/Aug/2024:10:36:45
Aug 5 10:37:12 server httpd[5683]: 203.0.113.100 -- [05/Aug/2024:10:37:12
SRC=203.0.113.200 DST=192.168.1.1 PROTO=TCP SPT=12345 DPT=1433
Aug 5 10:42:00 server fail2ban: WARNING [sshd] Ban 198.51.100.10
```

In this screenshot, I can present a few matching commands. I ran an alternative, multiple-stack pattern detection, port range, and complex IP pattern matching commands. These were the steps before creating an advanced log parsing and an incident response log.

The screenshot shows a terminal window titled "kali@kali: ~/cybersec-labs/regex-analysis". The terminal displays a log of network events and system activity. Key entries include:

- Firewall blocks from 203.0.113.50:12345 to 192.168.1.100:445 and 192.168.1.100:139.
- IDS alert for SQL Injection attack on port 203.0.113.75:45123.
- Webserver logs for GET /admin/config.php?id=1' OR '1'='1 HTTP/1.1 requests.
- Webserver logs for POST /login.php HTTP/1.1 requests.
- Authentication failures for root and admin users.
- Malware scanner detection of Trojan.Win32.Generic in /tmp/suspicious_file.exe.
- Malware scanner quarantine of /tmp/suspicious_file.exe.
- DNS queries for suspicious-domain.com and blocked request to malware-c2.evil.com from 192.168.1.150.
- Proxy blocked URL: http://phishing-site.com/steal-credentials.php requested by 192.168.1.175.

Here is my very extensive incident response log. Now it is time to see if I executed this correctly.

The screenshot shows a terminal window titled "kali@kali: ~/cybersec-labs/regex-analysis". The terminal contains a series of shell commands used for incident analysis and IOC extraction. The commands include:

- \$ grep -E "(THREAT|malware|Trojan|suspicious-domain|malware-c2)" incident_log.txt >> incident_analysis.txt
- \$ echo -e "\nAttacker IP addresses:" >> incident_analysis.txt
- \$ grep -oE "([0-9]{1,3}\.){3}[0-9]{1,3}" incident_log.txt | grep "203.0.113" | sort | uniq -c | sort -nr >> incident_analysis.txt
- \$ echo "==== IoC Extraction ====" > ioc_report.txt
- \$ echo "Malicious IP addresses:" >> ioc_report.txt
- \$ grep -oE "203\.0\.113\.[0-9]{1,3}" incident_log.txt | sort | uniq >> ioc_report.txt
- \$ echo -e "\nMalicious domains:" >> ioc_report.txt
- \$ grep -oE "[a-zA-Z0-9.-]+\.(com|net|org)" incident_log.txt | grep -E "[suspicious|malware|vill|phishing)" >> ioc_report.txt
- \$ echo -e "\nMalicious files:" >> ioc_report.txt
- \$ grep -oE "/[a-zA-Z0-9.-]+\.(exe|bat|com|scr|pif)" incident_log.txt >> ioc_report.txt

I have successfully created the incident analysis and IOC extraction. Therefore, my incident response log was successful. I had no issues. With the conclusion of that exercise, I have completed the lab.

Knowledge Assessment:

Question 1: Which regex pattern would match IP addresses in log files?

- b) [0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}

Question 2: True or False: The regex anchor ^ matches the beginning of a line.

True

Question 3: Which grep command would find all lines containing either "Failed" or "Denied"?

- b) grep -E "(Failed|Denied)" file.log

Question 4: What does the regex quantifier {2,5} specify?

- d) Both b and c are correct

Question 5: Which regex pattern would detect potential SQL injection attempts in web logs?

- d) All of the above

Conclusion:

With the completion of Lab 3, I have greatly developed my regex skills. I am now very familiar with running scans and detections such as incident response, threat hunting, and security operations. I now have more interest in specific jobs such as Incident Response Investigator, Digital Forensic Investigator, and Penetration Tester. These skills from this assessment will greatly impact my job search and desire to expand my interests.