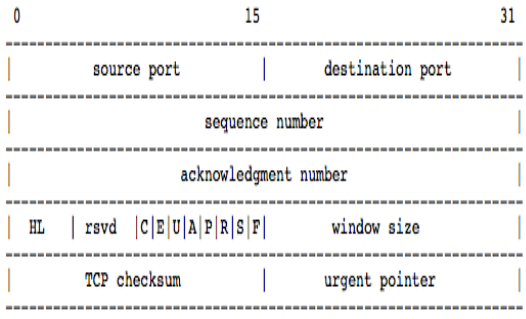
Exercise 1

A SYN flood occurs when a host becomes so overwhelmed by SYN segments of requests that it becomes incapable of handling legitimate segments of requests.
The communication occurs between two systems through a three-way handshake, which is Based on triple packets exchange. System A sends SYN packets to System B, B responds with SYN/ACK segment and A responds with ACK segment. When haphazardly, the frequency of SYN packets being sent by Sender A increases, for System B it start crating a backlog of SYN packets to handle. This results in network overload and can make the whole network Unresponsive.

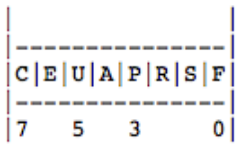
tcpdump is a command line network packet analyzer tool which prints the description of the contents of packets on the network. tcpdump captures based on source/destination IP address, time of packet flow, and also by protocol types such as TCP/IP, udp, ipv4/6 etc.

(b) The designing of filter according to the manual:

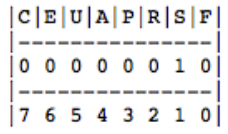
The tcpdump user manual gives a detailed information on how to capture TCP packets with particular flag combinations such as SYN-ACK. To understand the whole scenario, we should, firstly understand the construction of TCP header.



(F1)



(F2)



(F3)

A TCP header (Figure 1) usually holds 20 octets of data, unless options are present with the relevant TCP control bits are saved in the 13th octet (Figure 2).

In the condition where the packet is SYN type, the TCP header will be set as that shown in Figure 3 with the bit1 set to 1. If we convert the binary value obtained into corresponding unsigned 8-bit number, we will get the value 2 (decimal value of 00000010 is 2).

So, we can say that whenever the 13th octet of TCP header has the corresponding binary value as 2, the request contains a SYN packet, i.e,

tcp[13]==2

We can use the above statement as a filter for tcpdump to get only those packets which have their SYN as set. We can create the final statement as:

tcpdump –i x10 tcp[13]==2

(a) Conclusion: Although the filter can be created there are reasons why it will not be successful for any IDS. An attacker can use multiple fake IPs to flood and thus the tcpdump wont be able to detect the attack just like a distributed attack. tcpdump will consider each request as separate.

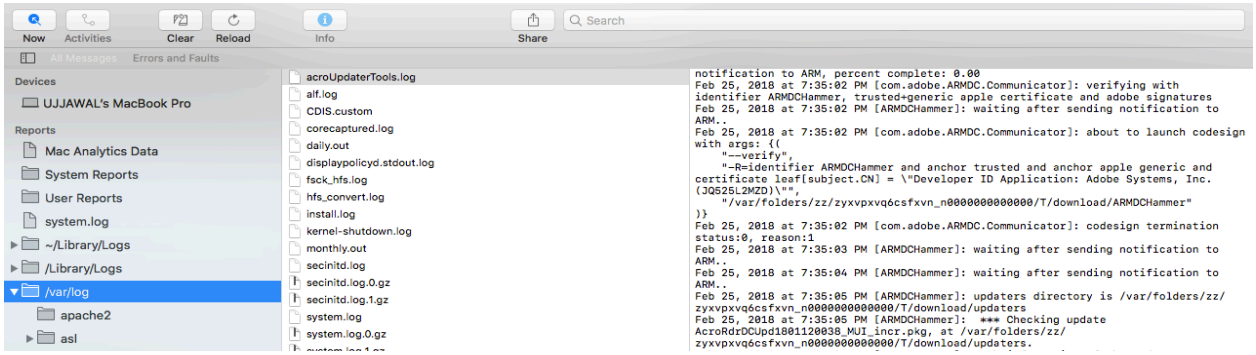
Exercise 2

My computer uses macOS operating system with the version of 10.13.1 as High Sierra. The BSM or the Basic Security Model developed by Apple is responsible for the framework of Audit Logs generation in MacOS. Current version is maintained by the Trusted BSD Project, and is known as OpenBSM.

(a)The audit logs in macOS gets generated at the location /private/var/audit. The audit logs are only visible to user having the root privileges.

```
UJJAWALs-MacBook-Pro:Logs ujjawalsharma$ sudo -s
Password:
bash-3.2# cd /private/var/audit/
bash-3.2# ls
20160114013717.crash_recovery   20170909100029.crash_recovery   20171108021425.crash_recovery   20171228191526.crash_recovery   20180201023346.crash_recovery
20160114173632.crash_recovery   20170916183333.crash_recovery   20171114020049.crash_recovery   20180109222758.crash_recovery   20180218024635.not_terminated
20170101000113.crash_recovery   20171011052510.crash_recovery   20171117092809.crash_recovery   20180109233825.crash_recovery   current
20170101000144.crash_recovery   20171016221717.crash_recovery   20171117113518.crash_recovery   20180112182000.crash_recovery
20170907105246.crash_recovery   20171031094018.crash_recovery   20171118084828.crash_recovery   20180128074813.crash_recovery
20170908004627.crash_recovery   20171104111336.crash_recovery   20171208203020.crash_recovery   20180131033327.crash_recovery
```

From the above image, we can see that there are few log files which have naming convention as **date.crash_recovery**. These files are the archived files of a day having the date part in YearMonthDayHourMinuteSecond format. Another file having format as **date.not_terminated** is the log file, which is currently recording the audit, logs of the system. At the end of the day, it will get saved in the **date.crash_recovery** format.
The third file **current** points to the **date.not_terminated** file. All these logs are saved in a binary format and can be read using the **praudit** command. As a UI feature, MacOS comes with an application **console.app**, using which we can see all these logs in a human readable format, as shown below:



The audit configuration files are located in **/etc/security** folder (root access required)

(b) In MacOS, the failed logins are stored in **private/var/audit/** directory into the file **/private/var/audit/todaysDate.not_terminated** file.
The above file can be seen using the **praudit** command, as shown below:

```
bash-3.2# praudit 20180218024635.not_terminated
header,102,11,audit crash recovery,0,Sat Feb 17 21:46:35 2018, + 929 msec
text,launchd::Audit recovery
path,/var/audit/20180201023346.crash_recovery
return,success,0
trailer,102
header,57,11,audit startup,0,Sat Feb 17 21:46:35 2018, + 929 msec
text,launchd::Audit startup
return,success,0
trailer,57
header,125,11,session start,0,Sat Feb 17 21:46:43 2018, + 109 msec
argument,1,0x0,sflags
argument,2,0x0,am_success
argument,3,0x0,am_failure
subject,-1,root,wheel,root,wheel,0,100003,0,0.0.0.0
return,success,0
trailer,125
```

(c) For designing the IDS, we can take the tuple parameters as:

X=(Time, Date)
A= (Event-Type, Error)
Y=(0 or 1) (O is for failure, 1 for success)

Below is the pseudo code, which can define the IDS for failure of more than 3 login in one hour:

```
Initializing start_hour=00              //(In Hours)
Initializing end_hour=00                //(In Hours)

FOR Each 1 Hour:
        start_hour=end_hour;
        end_hour++;
        FOR EachLine in /private/var/audit/log_file:     //Every Hour, extracting logs
            Initialize localLogs="";
            IF (time>start_hour && time<end_hour)
                localLogs= localLogs + EachLine;
            END IF
        END FOR
        Initialize fail_count=0;
        FOR EachLine in localLogs:                       //Iterating over the logs of the hour.
            IF (EachLine contains 'AUTH FAILED')
                fail_count++;                  //Increasing the count, if occurance of fail
                IF(fail_count>3)
                    Alert("Too Many Failed Logins, Chance of an intrusion !!")
                END IF
            END IF
        END FOR
END FOR
```