

Spring 2015



Information Security Project

Instructor:

Prof Alvaro Cardenas
University of Texas at Dallas
Department of Computer Science
Spring 2015

Team:

Nilesh Gupta – nxg130030
Partha De – pxd141430
Shazia Aftab – sxa141930
Swapnil Hasabe – sdh140430

Index

1. Problem No. 1-A
2. Problem No. 1-B
3. Problem No. 1-C
4. Problem No. 1-D
5. Problem No. 1-E
6. Problem No. 1-F
7. Problem No. 2
8. Problem No. 3-A
9. Problem No. 3-B
10. Problem No. 3-C

1.Learning how to use Bro

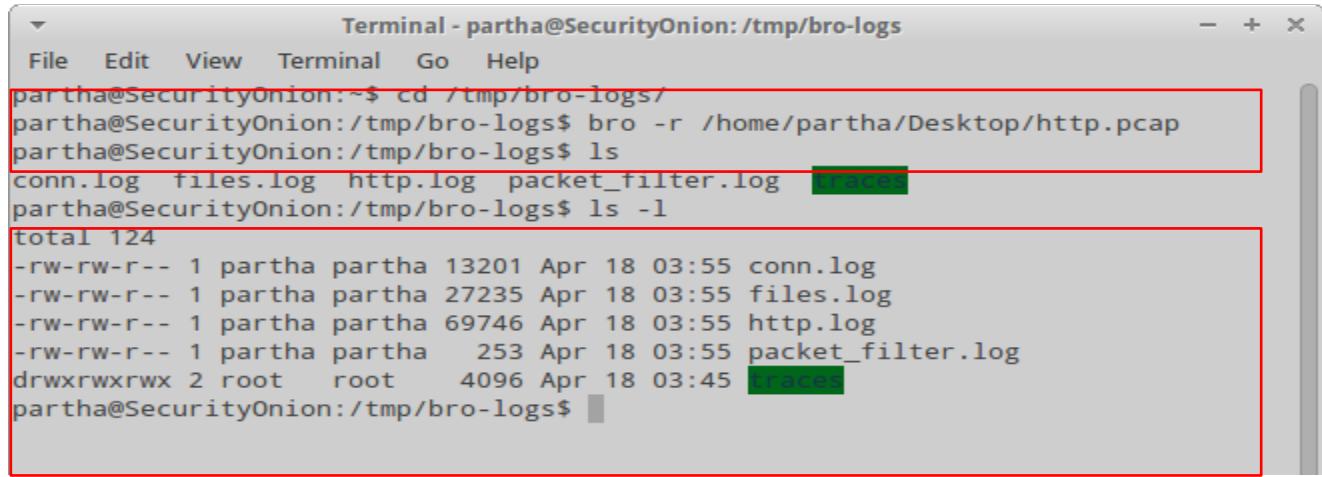
Ans. 1-A: <https://www.bro.org/bro-workshop-2011/solutions/logs/index.html>

Exercise: Understanding and Examining Bro Logs: While operating on and protecting a system, Bro produces huge amount of log files. The administrators may find it difficult sometimes to gather useful information as the enormous amount of data is written in file system. In this section we have learned about some very useful utilities that can be used to retrieve data as desired.

Part-1: Generating Logs: We are provided with an `http.pcap` file generated by a Bro system. ‘`bro -r`’ option reads from the `tcpdump` file and generates multiple log files.

Without a meaningful readable log files, users can't do much in monitoring the system.

```
Terminal - partha@SecurityOnion: /tmp/bro-logs
File Edit View Terminal Go Help
-rw-rw-r-- 1 partha partha 253 Apr 18 03:55 packet_filter.log
drwxrwxrwx 2 root root 4096 Apr 18 03:45 traces
partha@SecurityOnion:/tmp/bro-logs$ bro -h
bro version 2.3.2
usage: bro [options] [file ...]
  <file>                                | policy file, or read stdin
  -a|--parse-only                         | exit immediately after parsing scripts
  -b|--bare-mode                          | don't load scripts from the base/ directory
  -d|--debug-policy                       | activate policy file debugging
  -e|--exec <bro code>                   | augment loaded policies by given code
  -f|--filter <filter>                    | tcpdump filter
  -g|--dump-config                        | dump current config into .state dir
  -h|--help|?                            | command line help
  -i|--iface <interface>                 | read from given interface
  -p|--prefix <prefix>                   | add given prefix to policy file resolution
  -r|--readfile <readfile>                | read from given tcpdump file
  -y|--flowfile <file>[=<ident>]        | read from given flow file
  -Y|--netflow <ip>:<prt>[=<id>]       | read flow from socket
  -s|--rulefile <rulefile>              | read rules from given file
  -t|--tracefile <tracefile>            | activate execution tracing
  -w|--writefile <writefile>             | write to given tcpdump file
  -v|--version                           | print version and exit
  -x|--print-state <file.bst>          | print contents of state file
  -z|--analyze <analysis>               | run the specified policy file analysis
```

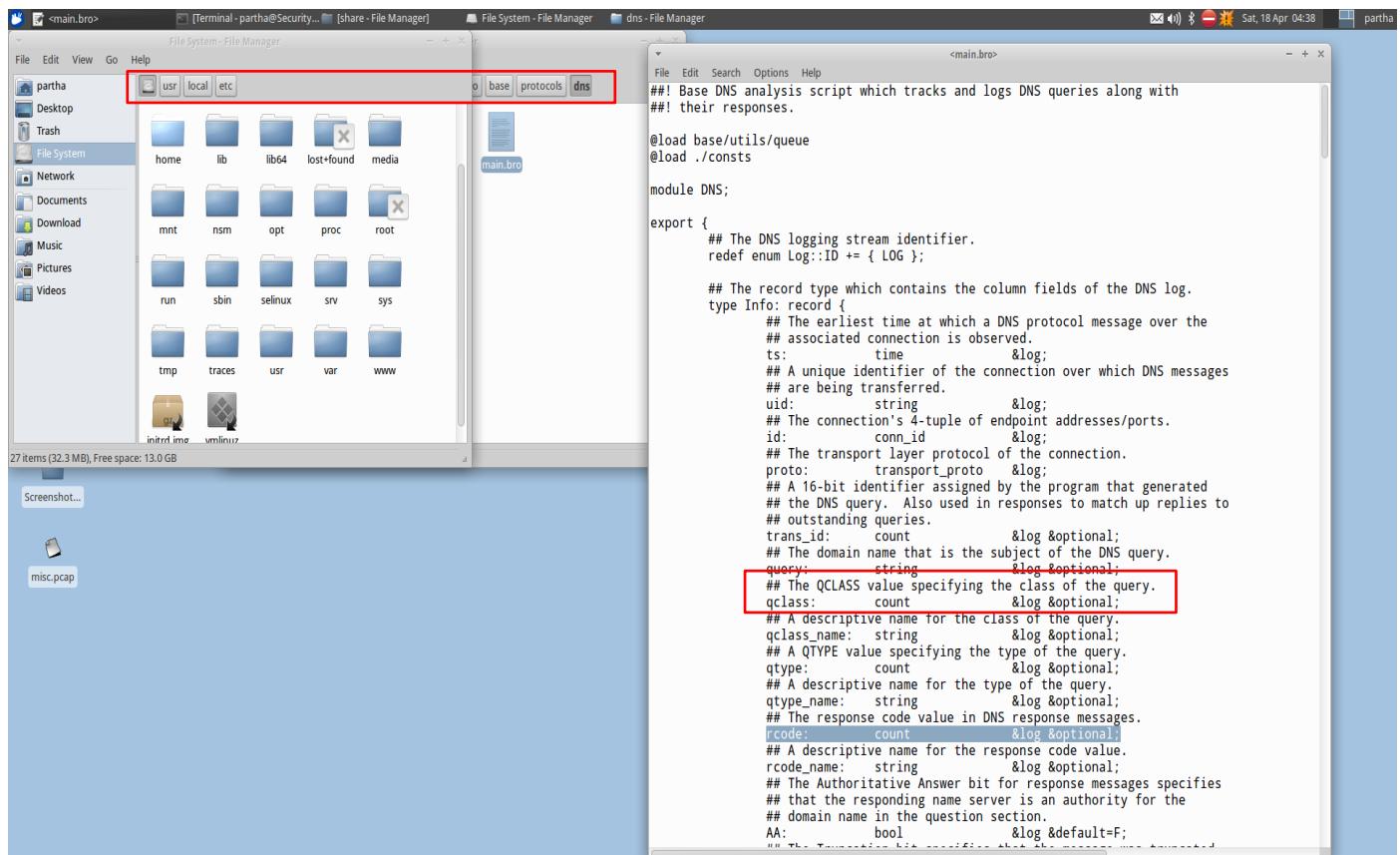


```

Terminal - partha@SecurityOnion:/tmp/bro-logs
File Edit View Terminal Go Help
partha@SecurityOnion:~$ cd /tmp/bro-logs/
partha@SecurityOnion:/tmp/bro-logs$ bro -r /home/partha/Desktop/http.pcap
partha@SecurityOnion:/tmp/bro-logs$ ls
conn.log files.log http.log packet_filter.log
partha@SecurityOnion:/tmp/bro-logs$ ls -l
total 124
-rw-rw-r-- 1 partha partha 13201 Apr 18 03:55 conn.log
-rw-rw-r-- 1 partha partha 27235 Apr 18 03:55 files.log
-rw-rw-r-- 1 partha partha 69746 Apr 18 03:55 http.log
-rw-rw-r-- 1 partha partha 253 Apr 18 03:55 packet_filter.log
drwxrwxrwx 2 root root 4096 Apr 18 03:45 misc
partha@SecurityOnion:/tmp/bro-logs$ 

```

Part-2: Matching Records to Log Fields: Extract the given file **misc.pcap** using the command ‘**bro -r**’. All the fields in the obtained **.log** files both in the Part-1 and Part-2 are explained in the following directory of our installation. **/usr/local/etc/share/bro/base/protocols/<Protocol_Name>/main.bro**



The screenshot shows a desktop environment with several windows open. On the left, a file manager window displays a file tree for the directory **<main.bro>**. The **dns** tab is selected. The tree includes **base**, **protocols**, and **dns** sub-directories under **<main.bro>**. The **dns** directory contains a file named **main.bro**. On the right, a terminal window shows the contents of the **main.bro** file. The code is a Bro analysis script for DNS. A red box highlights the section where the **qclass** field is defined. The code snippet is as follows:

```

##! Base DNS analysis script which tracks and logs DNS queries along with
##! their responses.

@load base/utils/queue
@load ./consts

module DNS;

export {
    ## The DNS logging stream identifier.
    redef enum Log::ID += { LOG };

    ## The record type which contains the column fields of the DNS log.
    type Info: record {
        ## The earliest time at which a DNS protocol message over the
        ## associated connection is observed.
        ts:           time          &log;
        ## A unique identifier of the connection over which DNS messages
        ## are being transferred.
        uid:          string         &log;
        ## The connection's 4-tuple of endpoint addresses/ports.
        id:           conn_id       &log;
        ## The transport layer protocol of the connection.
        proto:        transport_proto &log;
        ## A 16-bit identifier assigned by the program that generated
        ## the DNS query. Also used in responses to match up replies to
        ## outstanding queries.
        trans_id:     count         &log &optional;
        ## The domain name that is the subject of the DNS query.
        query:        string         &log &optional;
        ## The QCLASS value specifying the class of the query.
        qclass:       count         &log &optional;
        ## A descriptive name for the class of the query.
        qclass_name:  string         &log &optional;
        ## A QTTYPE value specifying the type of the query.
        qtype:        count         &log &optional;
        ## A descriptive name for the type of the query.
        qtype_name:   string         &log &optional;
        ## The response code value in DNS response messages.
        rcode:        count         &log &optional;
        ## A descriptive name for the response code value.
        rcode_name:   string         &log &optional;
        ## The Authoritative Answer bit for response messages specifies
        ## that the responding name server is an authority for the
        ## domain name in the question section.
        AA:           bool          &log &defaultF;
    }
}

```

It is very important to know the meanings and data types of each field to get some meaningful information from the log file.

```

Terminal - partha@SecurityOnion:/tmp/bro-logs
File Edit View Terminal Go Help
partha@SecurityOnion:/tmp/bro-logs$ bro -r /home/partha/Desktop/misc.pcap
partha@SecurityOnion:/tmp/bro-logs$ ls -l
total 44
-rw-rw-r-- 1 partha partha 5347 Apr 18 04:04 conn.log
-rw-rw-r-- 1 partha partha 11027 Apr 18 04:04 dns.log
-rw-rw-r-- 1 partha partha 815 Apr 18 04:04 files.log
-rw-rw-r-- 1 partha partha 1956 Apr 18 04:04 http.log
-rw-rw-r-- 1 partha partha 253 Apr 18 04:04 packet_filter.log
-rw-rw-r-- 1 partha partha 581 Apr 18 04:04 ssh.log
drwxrwxrwx 2 root root 4096 Apr 18 03:45 bro
-rw-rw-r-- 1 partha partha 538 Apr 18 04:04 weird.log
partha@SecurityOnion:/tmp/bro-logs$
```

```

Terminal - partha@SecurityOnion:/tmp/bro-logs
File Edit View Terminal Go Help
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path weird
#open 2015-04-18-04-04-30
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p name addl notice pee
#types time string addr port string string string bool string
1308930691.235561 CZdCw91ypcOKSC3PJj 172.16.238.1 5353 224.0.0.251 5353 dns_unmatched_reply
- F bro
1308930724.550812 - - - - dns_unmatched_msg -
1308930724.550812 - - - - dns_unmatched_msg -
1308930724.550812 - - - - dns_unmatched_msg -
#close 2015-04-18-04-04-30
-
```

```

Terminal - partha@SecurityOnion:/tmp/bro-logs
File Edit View Terminal Go Help
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#open 2015-04-18-04-04-30
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto service duration orig_bytes resp_bytes conn_state local_orig missed_bytes history orig_pkts orig_ip_bytes
#types time string port enum string interval count count string bool count count count set[string]
1308930691.235561 CZdCw91ypcOKSC3PJj 172.16.238.1 5353 224.0.0.251 5353 udp dns 0.003840 43 SF - 0 Dd 1 71 1 71 (empty)
1308930691.235561 CZdCw91ypcOKSC3PJj 172.16.238.1 5353 224.0.0.251 5353 udp dns 0.003840 43 SF - 0 Dd 1 71 1 71 (empty)
1308930691.235561 CZdCw91ypcOKSC3PJj 172.16.238.1 5353 224.0.0.251 5353 udp dns 0.003840 43 SF - 0 Dd 1 71 1 71 (empty)
1308930716.700542 CZdCw91ypcOKSC3PJj 172.16.238.131 45126 172.16.238.2 22 SF ssh 9.959807 2405 2687 SF - 0 Dd 1 40 4497 30 4455 (empty)
1308930716.702726 CytGdk1CPATQPTiuu 172.16.238.131 45126 172.16.238.2 22 SF ssh 0.007219 43 43 SF - 0 Dd 1 71 1 71 (empty)
1308930694.548964 CYdznsig2wkyHs67mk 172.16.238.1 49657 172.16.238.131 80 tcp http 20.001547 842 714 SF - 0 ShAdaff 8 1270 6 1034 (empty)
#close 2015-04-18-04-04-30
-
#open 2015-04-18-04-04-30
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto service duration orig_bytes resp_bytes conn_state local_orig missed_bytes history orig_pkts orig_ip_bytes
#types time string port enum string interval count count string bool count count count set[string]
1308930691.235561 CZdCw91ypcOKSC3PJj 172.16.238.131 5353 224.0.0.251 5353 udp dns 14.950443 258 0 S0 - 0 D 6 426 0 0 (empty)
1308930691.235561 CZdCw91ypcOKSC3PJj 172.16.238.131 5353 224.0.0.251 5353 udp dns 14.950381 410 0 S0 - 0 D 5 550 0 0 (empty)
1308930691.235561 CZdCw91ypcOKSC3PJj 172.16.238.131 5353 224.0.0.251 5353 udp dns 0.005956 33 219 SF - 0 Dd 1 51 1 247 (empty)
1308930716.700542 CZdCw91ypcOKSC3PJj 172.16.238.131 51970 172.16.238.2 22 SF ssh 0.003354 34 448 SF - 0 Dd 1 62 1 476 (empty)
1308930716.707076 CfZf4q20JewM81r3q2 172.16.238.131 54300 172.16.238.2 22 SF ssh 0.004850 33 270 SF - 0 Dd 1 61 1 298 (empty)
1308930716.701198 CMfMDy258NCCTQy9X6 172.16.238.131 44555 172.16.238.2 53 udp dns 0.006195 33 270 SF - 0 Dd 1 61 1 298 (empty)
1308930716.704559 ChabUD335vpEBPfM4f 172.16.238.131 33103 172.16.238.2 53 udp dns 0.003905 33 260 SF - 0 Dd 1 61 1 288 (empty)
1308930716.707350 Cok6N2144CKSzXeJX 172.16.238.131 50206 172.16.238.2 53 udp dns 0.003211 38 451 SF - 0 Dd 1 66 1 479 (empty)
1308930716.707350 Cok6N2144CKSzXeJX 172.16.238.131 50206 172.16.238.2 53 udp dns 0.003211 38 451 SF - 0 Dd 1 66 1 479 (empty)
1308930716.707350 Cq5J7z2kRLJ7hyHck 172.16.238.131 57272 172.16.238.2 53 udp dns 0.005699 34 447 SF - 0 Dd 1 62 1 475 (empty)
1308930716.707350 CyTgdk1CPATQPTiuu 172.16.238.131 38116 172.16.238.2 53 udp dns 0.005596 36 276 SF - 0 Dd 1 64 1 304 (empty)
1308930716.711033 CyTgdk1CPATQPTiuu 172.16.238.131 38140 172.16.238.2 53 udp dns 0.005956 37 276 SF - 0 Dd 1 67 1 304 (empty)
1308930716.712596 Car9244QyAs1v1t3 172.16.238.131 55365 172.16.238.2 53 udp dns 0.005067 33 459 SF - 0 Dd 1 61 1 487 (empty)
1308930716.715770 Cvhjqy3u95U29HNM2 172.16.238.131 53103 172.16.238.2 53 udp dns 0.007727 38 456 SF - 0 Dd 1 66 1 484 (empty)
1308930716.717405 CuU8p83tpox3xThm9 172.16.238.131 59579 172.16.238.2 53 udp dns 0.007712 33 425 SF - 0 Dd 1 61 1 453 (empty)
1308930716.719059 CZK1zA4XU7J2MK8pdB 172.16.238.131 52956 172.16.238.2 53 udp dns 0.009015 34 447 SF - 0 Dd 1 62 1 475 (empty)
1308930716.723548 CyBep3ACR5p38k5pDl 172.16.238.131 48621 172.16.238.2 53 udp dns 0.005263 35 290 SF - 0 Dd 1 63 1 318 (empty)
1308930716.723548 CyBep3ACR5p38k5pDl 172.16.238.131 48621 172.16.238.2 53 udp dns 0.005263 35 290 SF - 0 Dd 1 63 1 318 (empty)
1308930716.723548 CyBep3ACR5p38k5pDl 172.16.238.131 48621 172.16.238.2 53 udp dns 0.005263 35 290 SF - 0 Dd 1 63 1 318 (empty)
1308930726.862115 CZwab2D0XsyHeB7u 172.16.238.131 54925 172.16.238.2 53 udp dns 0.003309 35 91 SF - 0 Dd 1 63 1 119 (empty)
1308930726.862014 CZwab2D0XsyHeB7u 172.16.238.131 53624 172.16.238.2 53 udp dns 0.001906 35 91 SF - 0 Dd 1 63 1 119 (empty)
1308930726.950298 GH9853iaju5WnHRI 172.16.238.131 56215 172.16.238.2 53 udp dns 0.002653 35 91 SF - 0 Dd 1 63 1 119 (empty)
1308930726.953286 Ca70q11Kv0416gUxh 172.16.238.131 38118 172.16.238.2 53 udp dns 0.002358 45 163 SF - 0 Dd 1 73 1 191 (empty)
1308930726.955937 CJLqlq1f2ZKE66gdq 172.16.238.131 37934 172.16.238.2 53 udp dns 0.002205 45 163 SF - 0 Dd 1 73 1 191 (empty)
1308930726.958792 Co4cb0d8K8PzAu1j 172.16.238.131 36684 172.16.238.2 53 udp dns 0.002474 35 91 SF - 0 Dd 1 63 1 119 (empty)
1308930726.964858 CvtJm1n6oGRVWzFte 172.16.238.131 58367 172.16.238.2 53 udp dns 0.002808 45 163 SF - 0 Dd 1 73 1 191 (empty)
1308930726.968415 Cvt11YrHK8S2z12Wqj 172.16.238.131 42269 172.16.238.2 53 udp dns 0.002871 35 91 SF - 0 Dd 1 63 1 119 (empty)
1308930726.971640 Chk17128u0ff1kg3 172.16.238.131 56485 172.16.238.2 53 udp dns 0.002908 45 163 SF - 0 Dd 1 73 1 191 (empty)
1308930726.974759 C5npkr3c55bcC0dx108 172.16.238.131 39723 172.16.238.2 53 udp dns 0.002609 45 163 SF - 0 Dd 1 73 1 191 (empty)
1308930694.549219 CB7ewf2p48mzCvvSLd 172.16.238.1 49658 172.16.238.131 80 tcp - 30.001593 0 0 SF - 0 ShAfFa 5 272 4 224 (empty)
1308930726.984150 CQ1Q18nPH1XnJx4Ed 172.16.238.131 55515 74.125.225.81 80 tcp http 0.282829 1434 13871 SF - 0 ShAdaff 16 2094 15 14475 (empty)
1308930697.075345 CCCCCbe117RJ768lyBc 172.16.238.131 45908 141.142.192.39 22 tcp ssh 1.362081 1479 1724 SF - 0 ShAdaff 13 2019 16 2368 (empty)
1308930727.236071 CuuIn0q4MVYQE1wqzq 172.16.238.1 123 69.50.219.51 123 udp - 30.008549 218 0 S0 - 0 D 2 274 0 0 (empty)
#close 2015-04-18-04-04-30
```

Part-3: Connection Statistics: Bro summarizes each TCP and UDP connection as a single line in the conn.log. We can filter the connections depending on different conditions.

First, these connection summaries are quite detailed, so we can extract plenty useful statistics from it. Moreover, we can concentrate only on those records on which we are interested. The set size will become much smaller compared to the whole dump file.

(i) List the connections by in increasing order of duration, i.e., the longest connections at the end. The following command is used to achieve the goal. By ‘NR > 4’ the first 4 commented lines are ignored and the list is sorted by the 9th column which is by default holds the ‘interval’.

```

File Edit View Terminal Go Help
partha@SecurityOnion:/tmp/bro-logs$ awk 'NR > 4 < conn.log | sort -t$"\t" -k 9 -n
clobbered 2015-04-18-04-04-30
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto service duration orig_bytes resp_bytes conn_state local_orig missed_bytes history orig_pkts orig_ip_bytes
#types time string addr port enum string interval count count string bool count string count count count set[string]
1308930726.862014 C2wab8ZDPsIe8u7 172.16.238.131 33624 172.16.238.2 53 udp dns 0.001906 35 91 SF - 0 Od 1 63 1 119 (empty)
1308930726.955937 CGJG1q1f2KCKE649pd 172.16.238.131 37934 172.16.238.2 53 udp dns 0.002205 45 163 SF - 0 Od 1 73 1 191 (empty)
1308930726.955286 Ca70Jq1lKVQ416guKh 172.16.238.131 38118 172.16.238.2 53 udp dns 0.002358 45 163 SF - 0 Od 1 73 1 191 (empty)
1308930726.958986 Co4cdh8f8KPz4u1j 172.16.238.131 36862 172.16.238.2 53 udp dns 0.002474 35 91 SF - 0 Od 1 63 1 119 (empty)
1308930726.959750 CSmpAr4XmCpNkxt8 172.16.238.131 39223 172.16.238.2 53 udp dns 0.002609 45 163 SF - 0 Od 1 73 1 191 (empty)
1308930726.959958 CH95S1abnRw5d 172.16.238.131 56634 172.16.238.2 53 udp dns 0.002853 35 91 SF - 0 Od 1 63 1 119 (empty)
1308930726.061792 CdMr4q1NqghUj1Jte 172.16.238.131 55552 172.16.238.2 53 udp dns 0.002908 45 163 SF - 0 Od 1 73 1 191 (empty)
1308930726.964588 CvFjap2JmR9KDrpO 172.16.238.131 58367 172.16.238.2 53 udp dns 0.002837 45 163 SF - 0 Od 1 73 1 191 (empty)
1308930726.968415 CwtiiLvhHKM52z1DwI 172.16.238.131 42267 172.16.238.2 53 udp dns 0.002871 35 91 SF - 0 Od 1 63 1 119 (empty)
1308930726.971646 Chk17l2dbuIf1khuH3 172.16.238.131 56487 172.16.238.2 53 udp dns 0.002908 45 163 SF - 0 Od 1 73 1 191 (empty)
1308930716.707350 CKDK6N2144CK52x0Jxj 172.16.238.131 50205 172.16.238.2 53 udp dns 0.003211 38 451 SF - 0 Od 1 66 1 479 (empty)
1308930716.707354 CGp03XqBz8lHv8uGK 172.16.238.131 54065 172.16.238.2 53 udp dns 0.003259 35 91 SF - 0 Od 1 63 1 119 (empty)
1308930716.593745 Clz21Fr6sf1Wx0zY 172.16.238.131 58485 172.16.238.2 53 udp dns 0.003332 33 219 SF - 0 Od 1 61 1 227 (empty)
1308930716.700542 CoJNJL3g1On5oxXmI 172.16.238.131 51970 172.16.238.2 53 udp dns 0.003354 34 448 SF - 0 Od 1 62 1 476 (empty)
1308930691.130401 CSUUn3oYgSHTh3cvh3 172.16.238.131 37972 172.16.238.2 53 udp dns 0.003840 43 43 SF - 0 Od 1 71 1 71 (empty)
1308930716.704559 cba3u035vEBPrM41f 172.16.238.131 33109 172.16.238.2 53 udp dns 0.003905 33 260 SF - 0 Od 1 61 1 288 (empty)
1308930716.700708 Cf1fd4p20wsMsB13rq2 172.16.238.131 54304 172.16.238.2 53 udp dns 0.004850 33 270 SF - 0 Od 1 61 1 298 (empty)
1308930716.713596 Car92qdaQaoYaSiVfT3 172.16.238.131 55368 172.16.238.2 53 udp dns 0.005067 33 459 SF - 0 Od 1 61 1 487 (empty)
1308930716.561545 CGoPqf3PjUuL17HvK5D9 172.16.238.131 48018 172.16.238.2 53 udp dns 0.005063 35 290 SF - 0 Od 1 63 1 380 (empty)
1308930716.702544 Cog5J9Lb6Wl17Hyak 172.16.238.131 52772 172.16.238.2 53 udp dns 0.005609 34 447 SF - 0 Od 1 62 1 475 (empty)
1308930716.708921 Cxtyhyzh20t6r88A7ge 172.16.238.131 33818 172.16.238.2 53 udp dns 0.005811 36 276 SF - 0 Od 1 64 1 304 (empty)
1308930716.711033 CmcyOUDjMoq0000Q8 172.16.238.131 45140 172.16.238.2 53 udp dns 0.005956 39 276 SF - 0 Od 1 67 1 304 (empty)
1308930716.701199 CMMDp258Nm6RQyXp6 172.16.238.131 44555 172.16.238.2 53 udp dns 0.006195 33 270 SF - 0 Od 1 61 1 298 (empty)
1308930703.072726 CyGdk1PQ7PT1uu1 172.16.238.131 45126 172.16.238.2 53 udp dns 0.007219 43 43 SF - 0 Od 1 71 1 71 (empty)
1308930716.717405 CUu89mTpoyX3Thnn 172.16.238.131 59572 172.16.238.2 53 udp dns 0.007712 33 425 SF - 0 Od 1 61 1 453 (empty)
1308930716.717160 Cw6I1JzZuLhDm1o 172.16.238.131 53102 172.16.238.2 53 udp dns 0.008246 56 SF - 0 Od 1 66 1 397 (empty)
1308930716.7119059 CXZkTA4Xu7j3Bk8p8d 172.16.238.131 50002 172.16.238.2 53 udp dns 0.009015 34 447 SF - 0 Od 1 62 1 475 (empty)
1308930727.236071 CuAm0qAMYTgYElwq 172.16.238.131 123 69.50.219.51 123 dnn 0.006727 48 48 SF - 0 Od 1 76 1 76 (empty)
1308930716.457950 C1Q1lb1HfTf1AjwAed 172.16.238.131 55155 74.125.225.81 80 tcp http 0.282829 1434 13871 S1 - 0 ShADad 16 2094 15 14475 (empty)
1308930726.861650 CpFpyw3TT.JmAH1HmC 172.16.238.131 45905 141.142.192.39 22 tcp ssh 1.362031 1479 1724 SF - 0 ShAdBaFf 13 2019 16 2368 (empty)
1308930691.035040 CO5aT4C406KMa314 172.16.238.1 49656 172.16.238.121 22 ssh ssh 9.953807 2405 2887 SF - 0 ShAdBaFf 40 4497 30 4455 (empty)
1308930691.235561 Cz2CwHkCnLCKOpBpj 172.16.238.1 5353 224.0.0.251 5353 udp dns 14.950381 410 0 S0 - 0 D 5 550 0 0 (empty)
1308930691.235570 Cz4CwopG4PP9T0e 172.16.238.131 5353 224.0.0.251 5353 udp dns 14.950443 258 0 S0 - 0 D 6 426 0 0 (empty)
1308930691.235258 CauSPf311Uh1M1o69 169.250.200.125:ff:febd:ff:ff:ff:5353 dnn 14.950464 258 0 S0 - 0 D 6 426 0 0 (empty)
1308930703.068148 CAw3D059Qp2WeC81327 172.16.238.1 49659 172.16.238.131 21 tcp ftp 18.866337 55 394 SF - 0 ShAdBaFf 18 1003 12 1078 (empty)
1308930694.548964 CYdzw52pRyKhSw67mk 172.16.238.1 49657 172.16.238.131 80 tcp http 20.001547 842 714 SF - 0 ShADadFf 8 1003 13 1078 (empty)
1308930694.549219 CB7ewf2p4BmzCcvlsd 172.16.238.1 49658 172.16.238.131 80 tcp - 30.001593 0 0 SF - 0 ShAfFa 5 272 4 224 (empty)
1308930697.075345 CCS8e117Rj768VMyBc 172.16.238.1 17500 172.16.238.255 17500 udp - 30.006549 218 0 S0 - 0 D 2 274 0 0 (empty)
partha@SecurityOnion:/tmp/bro-logs$ partha@SecurityOnion:/tmp/bro-logs$
```

(ii) Find all connections that are last longer than 15 seconds. 9th field is checked whether the value is > 15 sec.

```

File Edit View Terminal Go Help
partha@SecurityOnion:/tmp/bro-logs$ awk 'NR > 4 && $9 > 15' conn.log
clobbered 2015-04-18-04-04-30
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto service duration orig_bytes resp_bytes conn_state local_orig missed_bytes history orig_pkts orig_ip_bytes
#types time string addr port enum string interval count count string bool count string count count count set[string]
1308930694.548964 CVdzw52pRyKhSw67mk 172.16.238.1 49657 172.16.238.131 80 udp dns 20.001547 842 714 SF - 0 ShAdadFf 8 1270 6 1034 (empty)
1308930694.549219 CB7ewf2p4BmzCcvlsd 172.16.238.1 49658 172.16.238.131 80 tcp - 30.001593 0 0 SF - 0 ShAfFa 5 272 4 224 (empty)
1308930697.075345 CCS8e117Rj768VMyBc 172.16.238.1 17500 172.16.238.255 17500 udp - 30.006549 218 0 S0 - 0 D 2 274 0 0 (empty)
partha@SecurityOnion:/tmp/bro-logs$ partha@SecurityOnion:/tmp/bro-logs$
```

(iii) Find all IP addresses of web servers that send more than more than 1 KB back to a client.

```

partha@SecurityOnion:/tmp/bro-logs$ bro-cut service resp_bytes id.resp_h < conn.
> | awk '$1 == "http" && $2 < 1000000 { print $3 }' \
> | sort -u
172.16.238.131
74.125.225.81
partha@SecurityOnion:/tmp/bro-logs$
```

(iv) Are there any web servers on non-standard ports (i.e., 80 and 8080)?

```
partha@SecurityOnion:/tmp/bro-logs$ bro-cut service id.resp_p id.resp_h < conn.log \
>     | awk '$1 == "http" && ! ($2 == 80 || $2 == 8080) { print $3 }' \
>     | sort -u
74.125.225.81
partha@SecurityOnion:/tmp/bro-logs$
```

Here, by \$1, \$2, \$3 the fields in the conn.log file are indicated.

(v) Show a breakdown of the number of connections by service.

```
partha@SecurityOnion:/tmp/bro-logs$ 
partha@SecurityOnion:/tmp/bro-logs$ bro-cut service < conn.log | sort | uniq -c
| sort -n
 1 ftp
 2 http
 2 ssh
 3 -
30 dns
partha@SecurityOnion:/tmp/bro-logs$
```

Here, ‘uniq –c’ takes the count for each unique type of service and ‘sort –n’ sorts the list according to the numerical value.

(vi) Show the 5 top destination ports in descending order.

```
partha@SecurityOnion:/tmp/bro-logs$ bro-cut id.resp_p < conn.log | sort | uniq -c
| sort -rn | head -n 5
 27 53
 3 5353
 2 80
 2 22
 1 88
partha@SecurityOnion:/tmp/bro-logs$
```

(vii) What are the top 10 hosts (originators) that send the most traffic?

```
partha@SecurityOnion:/tmp/bro-logs$ bro-cut id.orig_h orig_bytes < conn.log
    | sort
t != $1) {
        if (size != 0)
            print $1, size;
        host=$1;
        size=0
    } else
        size += $2
}
END {
    if (size != 0)
        print $1, size
}
| sort -k 2
| head -n 2
172.16.238.131 2623
172.16.238.1 2802
partha@SecurityOnion:/tmp/bro-logs$
```

Part-4: HTTP Statistics: As many of the attacks and suspicious activities are disguised under the HTTP request-reply, some special cares are needed to keep a closer eye on them. So Bro always gather different kinds of information about the hosts, like- Browser, Version, Operating System, etc.

(i) What are the distinct browsers in this trace? What are the distinct MIME types of the downloaded URLs?

Here we have selected only the desired columns by the command ‘bro-cut’ and then restrict the output to the distinct values. The “mime_type” field was absent in our file so it returned an “unknown field” message.

```
partha@SecurityOnion:/tmp/bro-logs$ bro-cut user_agent < http.log | sort -u
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.100 Safari/534.30
Mozilla/5.0 (X11; Linux i686; rv:5.0) Gecko/20100101 Firefox/5.0
partha@SecurityOnion:/tmp/bro-logs$ bro-cut mime_type < http.log | sort -u
bro-cut error: unknown field "mime_type"
partha@SecurityOnion:/tmp/bro-logs$
```

(ii) What are the two most commonly accessed web sites?

```
partha@SecurityOnion:/tmp/bro-logs$ bro-cut host < http.log | sort | uniq -c | sort -n | tail -n 2
2 172.16.238.131
2 www.google.com
partha@SecurityOnion:/tmp/bro-logs$
```

(iii) What are the top referred hosts?

```

Terminal - partha@SecurityOnion:/tmp/bro-logs
File Edit View Terminal Go Help
partha@SecurityOnion:/tmp/bro-logs$ bro-cut referrer < http.log
| awk 'sub(/[:alpha:]+:\//, "", $1) \
{
    split($1, s, /\//);
    print s[1]
}'
| uniq -c
| sort | head -n 1
| sort -rn
1 www.google.com
partha@SecurityOnion:/tmp/bro-logs$ 

```

Note: We worked with smaller tcpdump file due to performance reasons and that is why we searched only the top website as we do not have 10 unique site referred.

Part-5: Tweaking Log Output: After writing the script `new_separator.bro` script which modifies the field separator in the `.log` files during generation from tcpdump, we reprocessed `http.pcap`. The new files were generated with comma as the field separator. In the line number 2 in the commented line the change was also reflected.

```

Terminal - partha@SecurityOnion:/tmp/bro-logs-modify
File Edit View Terminal Go Help
partha@SecurityOnion:/tmp/bro-logs-modify$ bro -r /home/partha/Desktop/http.pcap
/home/partha/Desktop/new_separator.bro
partha@SecurityOnion:/tmp/bro-logs-modify$ ls
conn.log  files.log  http.log  packet_filter.log
partha@SecurityOnion:/tmp/bro-logs-modify$ vim conn.log
partha@SecurityOnion:/tmp/bro-logs-modify$ 

```

Note: To include the bro script while extraction the location of the bro script should be provided as follows.

`partha@SecurityOnion:~$ bro -r /home/partha/Desktop/ssh.pcap local /home/partha/Desktop/my_basic_policy.bro`

See the body of the bro script.

```

*new_separator.bro
File Edit Search Options Help
# Modifies the logfile separator character to be a comma instead of a tab.
# See: base/frameworks/logging/writers/ascii.bro for details.
redef LogAscii::separator = ",";

```

Next one is the log files regenerated with new field separator comma.

```

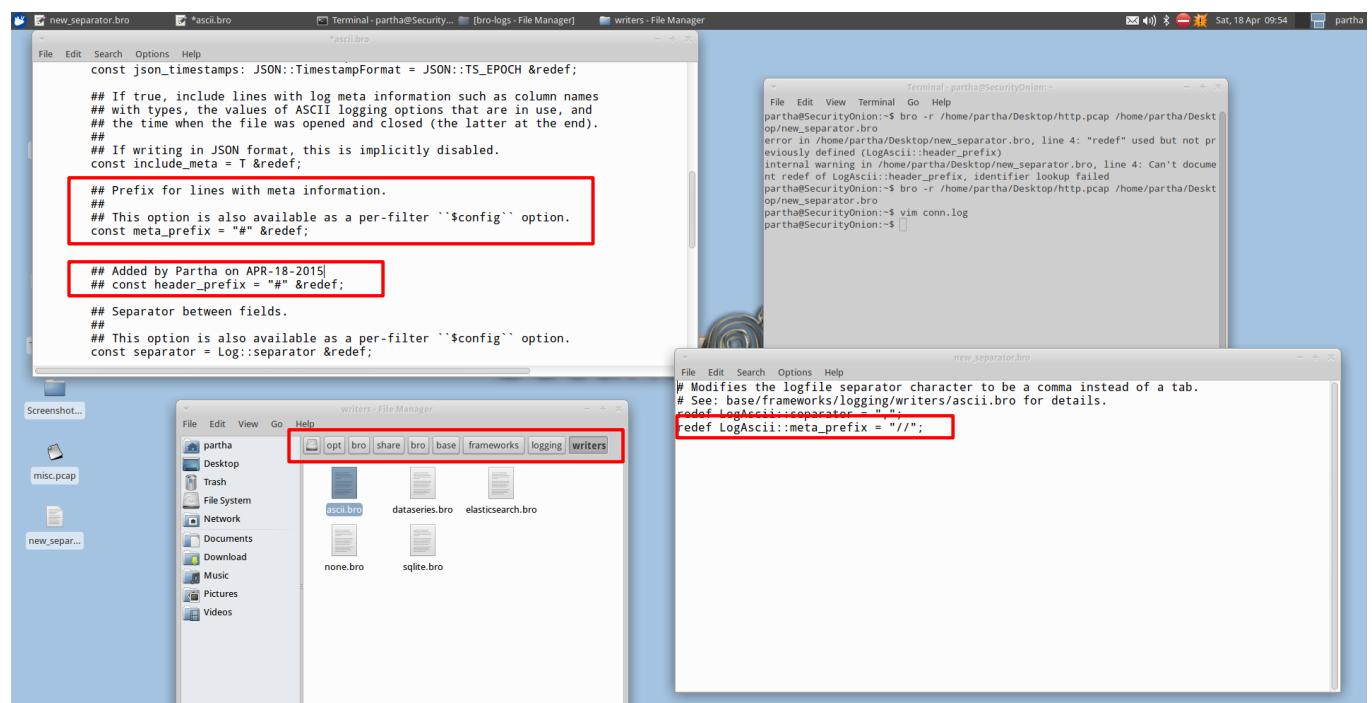
Terminal - partha@SecurityOnion: /tmp/bro-logs-modify
File Edit View Terminal Go Help
separator;
set_separator(',');
empty_reldb(empty)
unset_field,-
path.com
#2015-04-18-09-35-07
#fields,ts,uid,id.orig_h.id.orig_p.id.resp_h.id.resp_p.proto.service.duration,orig_bytes,resp_bytes,conn_state,local_orig,missed_bytes,history,orig_pkts,orig_ip_bytes,resp_pkts,resp_ip_bytes,tunnel_parents
1320279554,496300,Cpmell18439y-sf1Me,192,168,2,76,52025,208,85,42,28,80,tcp,-,0,125850,0,1092421,SF,-,0,daffa,400,20800,756,1131733,(empty)
1320279567,181431,CfGdfn2P46DEtghPf3,192,168,2,76,52034,174,129,249,33,80,tcp,http,0,082899,389,1495,SF,-,0,ShAd0ff,5,613,4,1667,(empty)
1320279567,181431,Cu2w93MKScyvNP8,192,168,2,76,52035,184,72,234,3,80,tcp,http,2,561940,905,731,SF,-,0,ShAd0ff,9,1289,8,1063,(empty)
1320279567,181050,C7bCyT3usin16Nb0d2,192,168,2,76,52033,184,72,234,3,80,tcp,http,3,345539,1856,1445,SF,-,0,ShAd0ff,15,2480,13,1969,(empty)
1320279572,537165,C4D0232xKlghP1r0s,192,168,2,76,52014,132,225,215,117,80,tcp,-,0,005881,0,0,SF,-,0,Ffa,2,104,1,52,(empty)
1320279572,886650,CbfXEWfUzdd1ja6h,192,168,2,76,52052,63,241,108,124,80,tcp,http,0,498720,156,2543,SF,-,0,ShAd0ff,6,1830,5,2747,(empty)
1320279577,453637,C9VjV3opbsWcJ98,192,168,2,76,52046,216,34,181,48,80,tcp,http,0,077548,596,576,SF,-,0,ShAd0ff,6,920,5,848,(empty)
1320279581,284239,C1s4duqsd0hYCh8g,192,168,2,76,52059,207,171,163,23,80,tcp,-,0,056486,0,0,SF,-,0,ShAFT,4,184,2,92,(empty)
1320279581,507914,Cee2e34pAXBwDjS5,192,168,2,76,52046,216,34,181,45,80,tcp,http,11,154832,2603,181933,SF,-,0,ShAd0ff,6,6775,134,188913,(empty)
1320279590,173878,CvmGay1xAlF86d7P,192,168,2,76,52077,125,225,78,80,tcp,-,0,048740,0,0,SF,-,0,ShAFT,4,220,2,112,(empty)
1320279600,532392,CmHd0232xKlghP1r0s,192,168,2,76,52081,199,159,187,43,80,tcp,http,0,233472,442,31853,0,0,ShAd0ff,6,1226,26,32773,(empty)
1320279600,826685,C0Vkgp1x70xhblLM3,192,168,2,76,52083,192,150,187,43,80,tcp,http,15,233472,442,31853,0,0,ShAd0ff,6,1226,26,32773,(empty)
1320279600,826441,Cn0HwB8x0xelAccf,192,168,2,76,52081,192,150,187,43,80,tcp,http,5,233763,446,24258,0,0,ShAd0ff,6,1166,21,2558,(empty)
1320279600,826004,CuFbHbyRwogTPUh,192,168,2,76,52079,192,150,187,43,80,tcp,http,5,496459,1309,17849,0,0,ShAd0ff,14,1626,17,17469,(empty)
1320279600,825492,CDjWV03XdcGzYAPkj,192,168,2,76,52080,192,150,187,43,80,tcp,http,5,515977,144,14412,0,0,ShAd0ff,14,2486,16,15252,(empty)
1320279600,581672,CvQyTIm1eAa03H6,192,168,2,76,52027,192,150,187,43,80,tcp,http,5,825503,1599,80801,0,0,ShAd0ff,37,3353,63,84085,(empty)
1320279607,998777,Cr0RFPhvF161nmwA7,192,168,2,76,52022,74,125,225,68,80,tcp,-,0,021505,0,0,0,0,ShAd0ff,2,104,1,52,(empty)
1320279607,998577,Cw588F1jDHZVBFSf2d,192,168,2,76,52023,209,85,145,101,80,tcp,-,0,031533,0,0,0,0,ShAd0ff,2,104,1,52,(empty)
1320279612,495344,CfQ3hN4gIgsLHr43Vl,192,168,2,76,52093,199,59,148,201,80,tcp,http,0,279806,907,1070,0,0,ShAd0ff,6,1231,5,1338,(empty)
1320279613,968096,C191d129k7G5zxc608,192,168,2,76,52094,199,59,148,201,80,tcp,http,0,486591,902,1070,0,0,ShAd0ff,6,1226,5,1338,(empty)
1320279611,171273,CmHd0232xL1N3Emm3,192,168,2,76,52091,192,150,187,43,80,tcp,-,0,081864,0,0,0,ShAFT,5,272,3,172,(empty)
1320279611,552622,CPUa231AQ510Ax2,192,168,2,76,52096,199,59,148,201,80,tcp,http,15,200059,4078,9556,0,0,ShAd0ff,12,4714,13,10240,(empty)
1320279610,744212,CSILAY3Q9gAnzbgnJ1,192,168,2,76,52090,192,150,187,43,80,tcp,http,6,499438,1669,37688,0,0,ShAd0ff,26,3033,31,3908,(empty)
1320279610,826604,CuFbHbyRwogTPUh,192,168,2,76,52089,192,150,187,43,80,tcp,http,5,496459,1309,17849,0,0,ShAd0ff,14,1626,17,17469,(empty)
1320279630,486420,CgDrVn7Zn145r62xP4,192,168,2,76,52097,199,59,148,201,80,tcp,http,0,166289,903,1070,0,0,ShAd0ff,6,1227,5,1338,(empty)
1320279630,021607,CgDrVn7Zn145r62xP4,192,168,2,76,52096,192,150,187,43,80,tcp,http,5,199366,427,15397,0,0,ShAd0ff,13,1115,15,16165,(empty)
1320279637,215536,C26t0nCkObjBnT7j,192,168,2,76,52100,199,59,148,201,80,tcp,http,0,264911,67,1066,0,0,ShAd0ff,7,1281,5,1336,(empty)
1320279577,687091,CxhNLc39ru1eEUWbh,192,168,2,76,52051,184,29,211,172,80,tcp,http,61,298320,1465,22567,0,0,ShAd0ff,19,2465,21,23667,(empty)
1320279639,698701,CrtWfg3u91BLnjqOka,192,168,2,76,52110,199,59,148,201,80,tcp,http,0,283987,901,1067,0,0,ShAd0ff,6,1225,5,1335,(empty)

```

Now we wanted to change the comment character to ‘ // ’. For that we consulted the file `ascii.bro` in the directory `/base/frameworks/logging/writers/` and found that character is defined by the key word “**meta_prefix**”. We overrode that value using Bro script.

Note: We also tried to add an entry with the key word “**header_prefix**” (As suggested in the training material). But unfortunately that did not work though we do not have any clue regarding the reason. We tried again by disabling the “**meta_prefix**”, still no luck. We got an error message saying **internal error: internal variable**

LogAscii::meta_prefix missing. Newly generated .log file sample screenshot is below.



```

Terminal - partha@SecurityOnion: ~
File Edit View Terminal Go Help
separator
//set_separator..
//unset_field,(empty)
//path_conn
//open 2015-04-18-09-51-50
//fields ts,src_ip,src_port,dst_ip,dst_port,proto,service,duration,orig_bytes,resp_bytes,conn_state,local_orig,missed_bytes,history,orig_pkts,orig_ip_bytes,resp_pkts,resp_ip_bytes,tunnel_parents
//time_string,src_ip,src_port,dst_ip,dst_port,enum_string,interval,count,conn_count,conn_count,seq(string)
1320279567,406300,CxAt5JhMwqkFB23,192.168.2.76,52026,208.85.43.28,80,tcp,-,2,135850,0,120431,SF,-,0,ShADaFF,400,20800,756,1121733,(empty)
1320279567,181431,CzL7vK5ybVmIO23,192.168.2.76,52024,174.129.249.33,80,tcp,http,0.02899,389,1495,SF,-,0,ShADaFF,5,613,4,1667,(empty)
1320279567,452735,CfRTw125oAdeqM6,192.168.2.76,52035,184,72,234.3,80,tcp,http,2,561940,905,731,SF,-,0,ShADaFF,9,1289,8,1063,(empty)
1320279567,181050,CuE8nBvKLPSznmk,192.168.2.76,52033,184,72,234.3,80,tcp,http,3,345539,185,1445,SF,-,0,ShADaFF,15,2480,13,1969,(empty)
1320279572,537165,CryFV4WmZCh2d2,192.168.2.76,52014,132,235,215,117,80,tcp,-,0.005881,0,0,FFA,2,104,1,52,(empty)
1320279572,181050,CuE8nBvKLPSznmk,192.168.2.76,52033,184,72,234.3,80,tcp,http,3,345539,185,1445,SF,-,0,ShADaFF,15,2480,13,1969,(empty)
1320279572,452735,CfRTw125oAdeqM6,192.168.2.76,52035,184,72,234.3,80,tcp,http,2,561940,905,731,SF,-,0,ShADaFF,9,1289,8,1063,(empty)
1320279581,284239,CzL7vK5ybVmIO23,192.168.2.76,52059,207,171,163,23,80,tcp,-,5.058468,0,0,FFA,4,184,2,92,(empty)
1320279577,507914,C477fcv151StjyuH,192.168.2.76,52045,2603,181933,SF,-,0,ShADaFF,80,6775,134,188913,(empty)
1320279599,558878,CxxFu10InJcBjMwyl,192.168.2.76,52077,74,125,225,78,80,tcp,-,5.048744,0,0,FFA,4,220,2,112,(empty)
1320279601,552309,CFKL8g22edGmlyN,192.168.2.76,52086,199,59,148,201,80,tcp,http,0.237418,883,1071,SF,-,0,ShADaFF,6,1207,5,1339,(empty)
1320279600,826685,C5b572ooytEo9,192.168.2.76,52083,192,150,187,43,80,tcp,http,5.233472,442,31933,SF,-,0,ShADaFF,20,1494,26,32713,(empty)
1320279600,826686,C5b572ooytEo9,192.168.2.76,52083,192,150,187,43,80,tcp,http,5.233472,442,31933,SF,-,0,ShADaFF,20,1494,26,32713,(empty)
1320279600,826604,CPuPhuZhjyCnFcY,192.168.2.76,52089,192,150,187,43,80,tcp,http,5.404390,886,16577,SF,-,0,ShADaFF,14,1636,17,17469,(empty)
1320279600,826592,C5b572ooytEo9,192.168.2.76,52079,192,150,187,43,80,tcp,http,5.496459,1309,17849,SF,-,0,ShADaFF,16,2153,18,18793,(empty)
1320279600,826607,CRCNq1E2h2IdKTKEw7,192.168.2.76,52082,192,150,187,43,80,tcp,http,5.515177,1746,14412,SF,-,0,ShADaFF,14,2486,16,15252,(empty)
1320279600,581672,CsRiuP1vvKnTtR,192.168.2.76,52078,192,150,187,43,80,tcp,http,5.825503,1599,80081,SF,-,0,ShADaFF,37,3535,63,84085,(empty)
1320279607,998777,CfL5s2Wk5Na1Pl24b,192.168.2.76,52022,74,125,225,68,80,tcp,-,0.021509,0,0,FFA,2,104,1,52,(empty)
1320279607,998577,Cx0t6G31w1y7sN4,192.168.2.76,52028,209,85,145,101,80,tcp,-,0.021533,0,0,FFA,2,104,1,52,(empty)
1320279612,493844,CsL0N6G31w1y7sN4,192.168.2.76,52093,199,59,148,201,80,tcp,http,0.279806,907,1070,SF,-,0,ShADaFF,6,1226,5,1338,(empty)
1320279613,968096,CouX2c3QAkoyCnVnPg,192.168.2.76,52094,199,59,148,201,80,tcp,http,0.486591,902,1070,SF,-,0,ShADaFF,6,1226,5,1338,(empty)
1320279611,171273,Cz9Cux2byk32uymsk,192.168.2.76,52098,199,59,148,20,80,tcp,http,15,200059,4078,9556,SF,-,0,ShADaFF,12,4714,10,20240,(empty)
1320279612,444212,C6H5z2vWm51W4Kc,192.168.2.76,52099,192,150,187,43,80,tcp,http,5.299438,1669,3768,SF,-,0,ShADaFF,26,3033,31,39308,(empty)
1320279616,742239,CDj7e4W15j7P9rCe,192.168.2.76,52095,208,85,43,42,80,tcp,http,0.604819,346,59445,SF,-,0,ShADaFF,29,2066,45,61793,(empty)
1320279616,742239,CDj7e4W15j7P9rCe,192.168.2.76,52095,208,85,43,42,80,tcp,http,0.604819,346,59445,SF,-,0,ShADaFF,29,2066,45,61793,(empty)
1320279630,021607,Ctxxx2pd61zPho3b,192.168.2.76,52096,192,150,187,43,80,tcp,http,5.199360,421,15397,SF,-,0,ShADaFF,13,111,22,16165,(empty)
1320279637,215536,CkaSC73AMUkOr1h,192.168.2.76,52100,199,59,148,201,80,tcp,http,0.264911,905,1068,SF,-,0,ShADaFF,7,1281,5,1336,(empty)
1320279577,687091,Cxet8Ctndw3qD2q,192.168.2.76,52051,184,29,211,172,80,tcp,http,61.298320,1465,22567,SF,-,0,ShADaFF,19,2465,21,23667,(empty)
1320279633,698701,CZQVU12xiq4V1ajWoB,192.168.2.76,52110,199,59,148,20,80,tcp,http,0.283987,963,1067,SF,-,0,ShADaFF,6,1225,5,1335,(empty)
1320279638,450681,CsDj9Abcbv2LYPKab,192.168.2.76,52101,192,150,187,43,80,tcp,http,5.709781,758,19809,SF,-,0,ShADaFF,16,1602,20,20857,(empty)
1320279638,954157,CsKE13q4830TmN,192.168.2.76,52106,192,150,187,43,80,tcp,http,5.228420,379,494,SF,-,0,ShADaFF,7,747,5,766,(empty)
1320279638,954157,CsKE13q4830TmN,192.168.2.76,52106,192,150,187,43,80,tcp,http,5.228420,379,494,SF,-,0,ShADaFF,7,747,5,766,(empty)
1320279638,955996,Cs2af0210087Jfd1g,192.168.2.76,52103,192,150,187,43,80,tcp,http,5.243925,338,24829,SF,-,0,ShADaFF,18,123,22,22981,(empty)
1320279639,349306,CtB8n028L0wygtG0e,192.168.2.76,52109,192,150,187,43,80,tcp,http,4.862785,400,7004,SF,-,0,ShADaFF,9,880,8,7428,(empty)
1320279639,147746,CyyHK83qyrbnvnNb,192.168.2.76,52107,192,150,187,43,80,tcp,http,5.066841,404,491,SF,-,0,ShADaFF,6,728,4,707,(empty)
1320279639,205080,CANXf73J31Wq5EzD,192.168.2.76,52106,192,150,187,43,80,tcp,-,5,.009511,0,0,FFA,5,272,3,172,(empty)
1320279639,052091,CABUUm3dx5Id1lxh,192.168.2.76,52105,192,150,187,43,80,tcp,-,5.162501,0,0,FFA,5,272,3,172,(empty)
1320279639,052091,CABUUm3dx5Id1lxh,192.168.2.76,52105,192,150,187,43,80,tcp,http,5.019984,493,491,SF,-,0,ShADaFF,12,728,4,707,(empty)
1320279639,052091,CABUUm3dx5Id1lxh,192.168.2.76,52105,192,150,187,43,80,tcp,http,5.019984,493,491,SF,-,0,ShADaFF,12,728,4,707,(empty)
1320279630,486859,CLUsOo2yKVMyz1Ak,192.168.2.76,52099,199,59,148,20,80,tcp,http,15.198762,2050,4776,SF,-,0,ShADaFF,23,24,20185,(empty)
1320279637,118128,Cm0XmB35e1Yq52Pwh,192.168.2.76,52112,199,59,148,201,80,tcp,http,0.351267,902,1068,SF,-,0,ShADaFF,6,1226,5,1336,(empty)
1320279637,273571,C80k37315BPr8x4cNj,192.168.2.76,52111,192,150,187,43,80,tcp,http,5.564817,419,48038,SF,-,0,ShADaFF,23,1627,38,50022,(empty)
"conn.log" 101L, 13200C

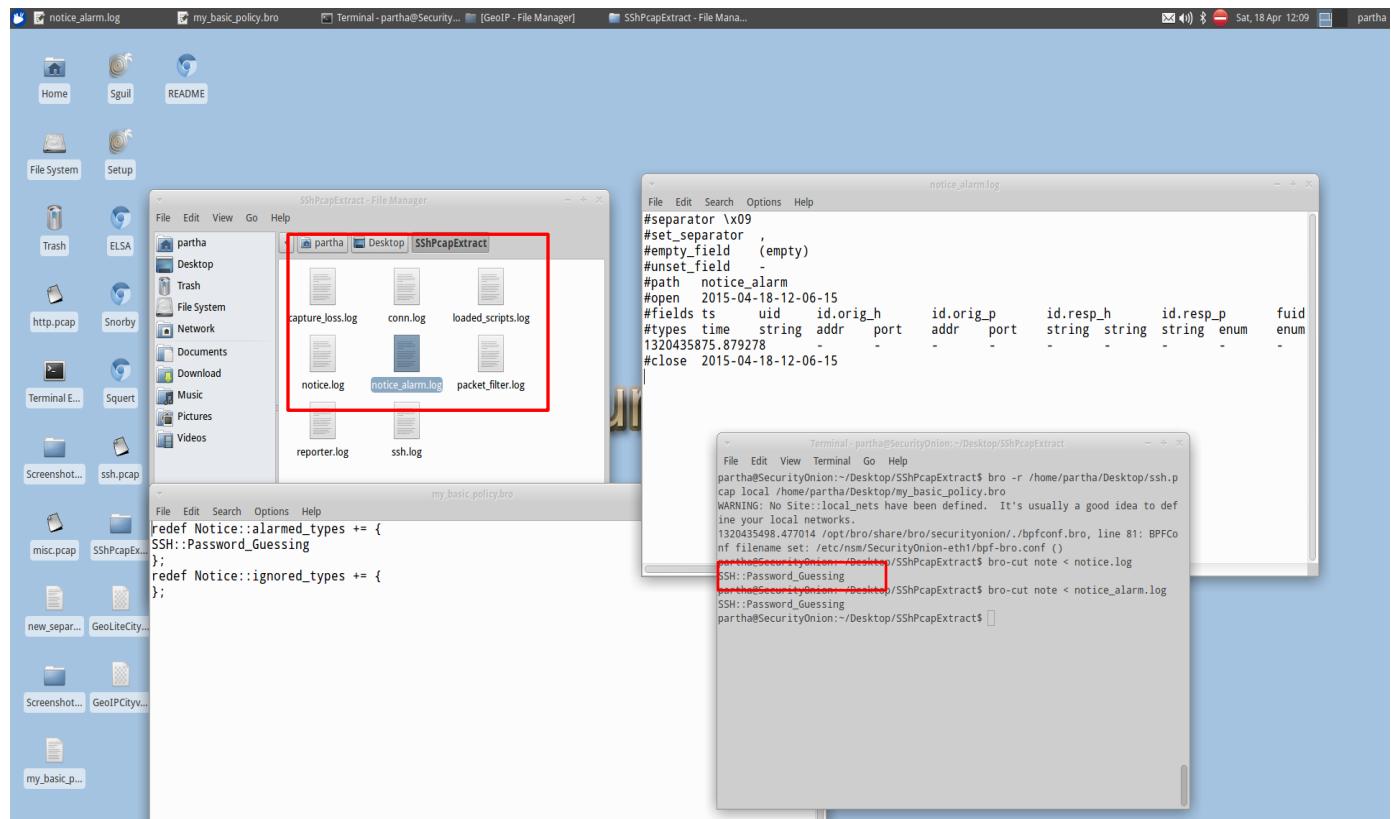
```

1,1 Top

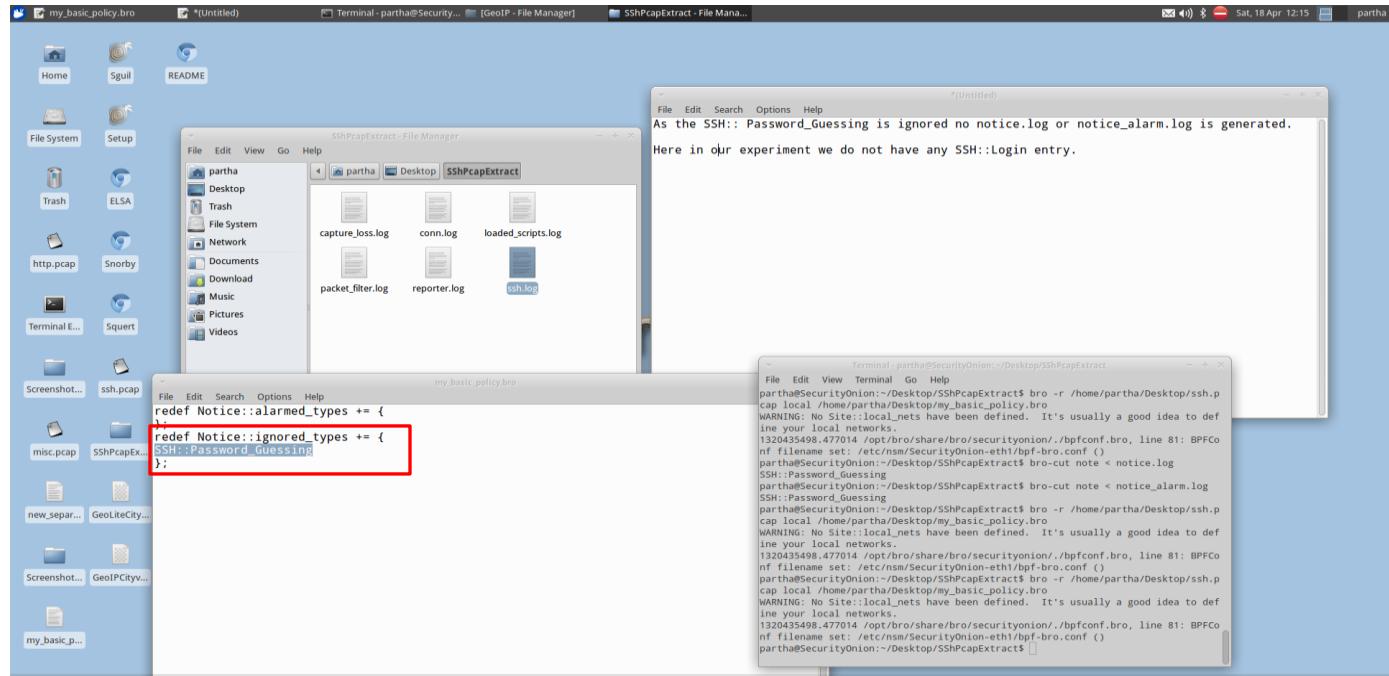
Ans. 1-B: <https://www.bro.org/current/solutions/notices/>

Exercise: Handling Notices: Bro by default, does some analysis on traffic and can suggest potentially interesting network activity to investigate. These suggestions are known as “notices”.

Part 1: Notice Log: Run Bro on a network trace and check if it suggests anything interesting to look at. We have downloaded **ssh.pcap** for this exercises. We have used the command ‘**bro -r ssh.pcap local**’ that will extract 8 files from the .pcap file. Now we have checked the file ‘**notice.log**’ file to see any interesting report and found somebody tried to brute forced to crack password.



Part 2: Basic Filtering on the Notice Log: Now, we have written custom policy using Bro script. We have to pass the custom script while extracting the .pcap file. We have used ‘`my_basic_policy.bro`’ as our custom Bro script. In the script we set a policy that if there is any password guessing brute force attack alarm should be raised. We have seen that after the extraction using the Bro script it generated a new file ‘`notice_alarm.log`’. By default Bro checks the policies from the file ‘`local.bro`’ in the directory “`<Installation_Dir>/share/bro/site`”. Using Bro script we can also send email to the administrators in case of any alarm.



Part 3: More Advanced Notice Policy: More complex policies can be written according to our need. We have used a Bro script ‘`my_advanced_policy.bro`’ to implement a policy like when password guessing is done from only some particular IP address happens, then only it will raise alarm. We can add the IP address 172.16.238.129

```
module SSH;

redef enum Notice::Type += {
    Login,
};

const watched_servers: set[addr] = {
    172.16.238.136,
    172.16.238.168,
} &redef;

event SSH::heuristic_successful_login(c: connection)
{
    NOTICE([$note=Login, $msg="Possible SSH login success", $conn=c]);
}

hook Notice::policy(n: Notice::Info)
{
    if ( n$note == SSH::Login && n$id$resp_h in watched_servers )
        add n$actions[Notice::ACTION_ALARM];
}
```

Part 4: Extra Credit: Till now the custom scripts we have seen are working only on one IP address, in other words, it works with one line in the log files. Now we can write some even more complex policies that catches all the IP addresses that attempted SSH brute force attack for last 24 hours duration and raise alarm if such an IP address does a successful login assuming that the attacker become successful to break in the system.

```
my_extracredit_policy.bro
File Edit Search Options Help
module SSH;

redef enum Notice::Type += {
    Login,
};

global brute_forcers: set[addr] &write_expire=24hrs;

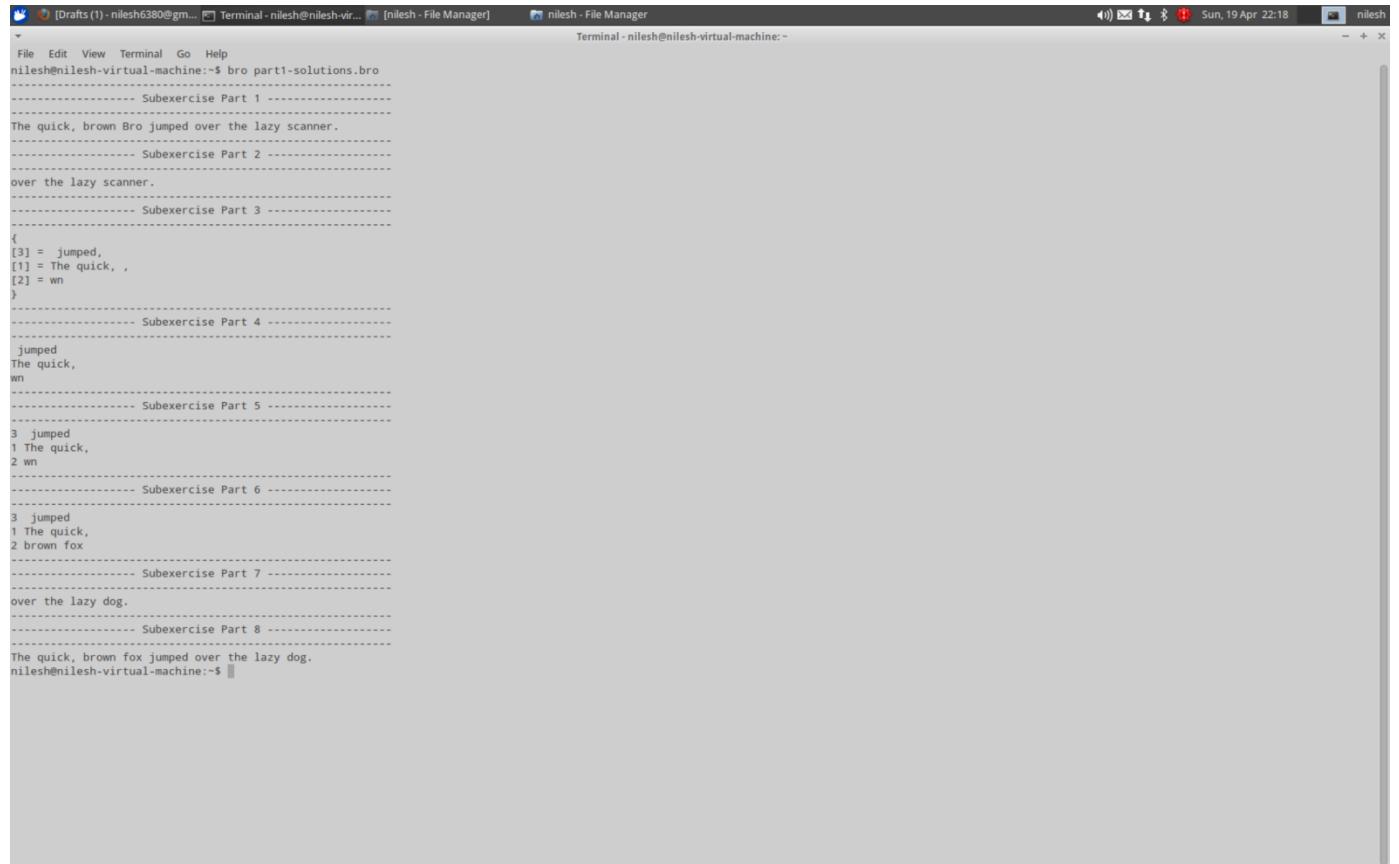
event SSH::heuristic_successful_login(c: connection)
{
    NOTICE([$note=Login, $msg="Possible SSH login success", $conn=c]);
}

hook Notice::policy(n: Notice::Info)
{
    if ( n$note == SSH::Password_Guessing )
        add brute_forcers[n$src];
    else if ( n$note == SSH::Login && n$id$orig_h in brute_forcers )
    {
        add n$actions[Notice::ACTION_ALARM];
        n$sub = "Was previously a password guesser";
    }
}
```

Ans. 1-C: <https://www.bro.org/bro-workshop-2011/solutions/prog-primer/index.html>

Exercise: Bro Programming Primer: Bro contains a programming language designed specially to be able to represent network-related abstraction (e.g. addresses and ports) and as such offers a great deal of functionality and flexibility in terms of helping you accomplish your network monitoring goals. . In this section we have learned about some very useful utilities that can be used to retrieve data as desired.

Part-1: String manipulations and patterns: We are provided with a [part1.bro](#) file with sub exercises inside it. This exercise will help us to learn about String manipulations and patterns.



```
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~$ bro part1-solutions.bro
----- Subexercise Part 1 -----
The quick, brown Bro jumped over the lazy scanner.
----- Subexercise Part 2 -----
over the lazy scanner.
----- Subexercise Part 3 -----
{
[3] = jumped,
[1] = The quick,
[2] = wn
}
----- Subexercise Part 4 -----
jumped
The quick,
wn
----- Subexercise Part 5 -----
3 jumped
1 The quick,
2 wn
----- Subexercise Part 6 -----
3 jumped
1 The quick,
2 brown fox
----- Subexercise Part 7 -----
over the lazy dog.
----- Subexercise Part 8 -----
The quick, brown fox jumped over the lazy dog.
nilesh@nilesh-virtual-machine:~$
```

Part-2: Tables and set and vectors: We are provided with a file [part2.bro](#). There are several containers/collections types that are commonly used in Bro for accumulating/storing state information. Here we are learning how to store data in tables, sets or vectors and manipulate/retrieve from them.

```

[Drafts (1)] - nilesh6380@gmail.com [Terminal - nilesh@nilesh-vir... [File Manager] nilesh - File Manager Terminal - nilesh@nilesh-virtual-machine:~]
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~$ bro part2-solutions.bro
----- Subexercise Part 1 -----
Albrous Huxley owns 192.168.1.102
George Brorwell owns 192.168.1.100
Brorson Scott Card owns 192.168.1.101
Broprah Winfrey owns 192.168.1.103
----- Subexercise Part 2 -----
Albrous Huxley owns 192.168.1.102
George Brorwell owns 192.168.1.100
Brorson Scott Card owns 192.168.1.101
----- Subexercise Part 3 -----
The mapping is gone.
----- Subexercise Part 4 -----
Albrous Huxley owns 192.168.1.102
George Brorwell owns 192.168.1.100
Brorson Scott Card owns 192.168.1.101
Maybe Another Broauthor owns 192.168.1.103
----- Subexercise Part 5 -----
{
  1985/tcp,
  2540/udp,
  1984/tcp,
  632/udp
}
----- Subexercise Part 6 -----
{
  1985/tcp,
  1984/tcp,
  2011/tcp
}
----- Subexercise Part 7 -----
[Ego, SuperEgo, Id]
----- Subexercise Part 8 -----
Ego
SuperEgo
Id
Albrous Huxley
George Brorwell
Brorson Scott Card
Maybe Another Broauthor
nilesh@nilesh-virtual-machine:~$ 

```

Part-3: Records and functions: We are provided with a file part3.bro with sub exercise in it. A record is a collection of values, with each have a name type pair. There is no restriction on allowed type, which means it can contain other, nested records or even functions as part of record. These exercises will help us learn creating the record, manipulation and deletion on the record and also about function inside it.

```

nilesh@nilesh-virtual-machine:~$ bro part3-solutions.bro
----- Subexercise Part 1 -----
192.168.1.111:ssh
192.168.1.222:13131/tcp
192.168.1.111:http
192.168.1.222:ssh
----- Subexercise Part 2 -----
192.168.1.111:ssh
192.168.1.222:ssh
192.168.1.111:http
192.168.1.222:13131/tcp
----- Subexercise Part 3 -----
*192.168.1.111:ssh
*192.168.1.222:ssh
*192.168.1.111:http
192.168.1.222:13131/tcp
----- Subexercise Part 4 -----
*192.168.1.111:ssh
*192.168.1.222:ssh
*192.168.1.111:http
192.168.1.222:13131/tcp
nilesh@nilesh-virtual-machine:~$ 

```

Part-4: Events: We have provided with a file part4.bro and tcpdump browse.pcap. Bro provides an event engine as the primary facilitator of network analysis. And bro can tap into the engine is at the scripting layer through even handler. In this exercise, we will learn about event and multi handlers.

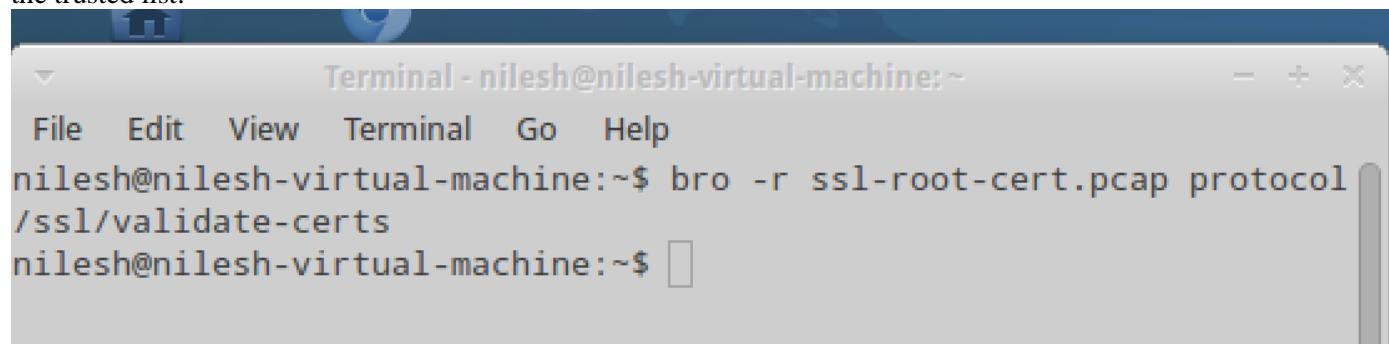
```
nilesh@nilesh-virtual-machine:~$ bro -r browse.pcap part4-solutions.bro
Hello, world!
Hello, Brogrammer!
192.150.187.43 -> /
192.150.187.43 -> /css/pygments.css
192.150.187.43 -> /css/960.css
192.150.187.43 -> /css/bro-ids.css
192.150.187.43 -> /js/jquery.cycle.all.min.js
192.150.187.43 -> /js/jquery.fancybox-1.3.4.pack.js
192.150.187.43 -> /js/jquery.tweet.js
192.150.187.43 -> /js/jquery.rssfeed.js
192.150.187.43 -> /js/jquery.tableofcontents.js
192.150.187.43 -> /js/superfish.js
192.150.187.43 -> /js-hoverIntent.js
192.150.187.43 -> /js/general.js
192.150.187.43 -> /js/jquery.collapse.js
192.150.187.43 -> /css/print.css
192.150.187.43 -> /documentation/index.html
192.150.187.43 -> /js/breadcrumbs.js
192.150.187.43 -> /documentation/reporting-problems.html
The following users might be experiencing Broblems:
  10.0.2.15
nilesh@nilesh-virtual-machine:~$
```

Ans. 1-D: <http://www.bro.org/bro-workshop-2011/solutions/extending/index.html>

Exercise: Extending a Script's Functionality:

Part 1: Customize which certificates to trust: pcap file has been given to us and asks us to perform certificate validation. The pcap file has a self-signed certificate. This is the same file is shown in the log files.

Importance: The admin might have gotten a new certificate and might be interested in adding this new certificate to the trusted list.



A screenshot of a terminal window titled "Terminal - nilesh@nilesh-virtual-machine:~". The window shows the following command being run:

```
nilesh@nilesh-virtual-machine:~$ bro -r ssl-root-cert.pcap protocol
/ssl/validate-certs
nilesh@nilesh-virtual-machine:~$
```

```
is subject issuer client_subject client_issuer validation_status
:ring
? (empty) CN=Brometheus,OU=CSD,O=NCSA,ST=IL,C=US CN=Brometheus,OU=CSD,O=NCSA,ST=IL,C=US - - - self signed certificate
```

This is ssl.log file

Certificate hex has been given in this exercise and using it, we have to give instructions to Bro to trust the self-signed Certificate.

For this we created a mytrust.bro file as

```
redef SSL::root_certs += {
["My First Certificate"] = "hex file contents";
And ran this command
```

```
nilesh@nilesh-virtual-machine:~$ bro -r ssl-root-cert.pcap protocol
s/ssl/validate-certs mytrust.bro
nilesh@nilesh-virtual-machine:~$
```

then Bro accepted the certificate as we can see the validation status shown below in the log file.

```
n_fuids subject issuer client_subject client_issuer validation_status
ng string
ttGxj9 (empty) CN=Brometheus,OU=CSD,O=NCSA,ST=IL,C=US CN=Brometheus,OU=CSD,O=NCSA,ST=IL,C=US - - - ok
```

The third exercise asks us to create the certificate file that we used in the last file.

I typed the following command:

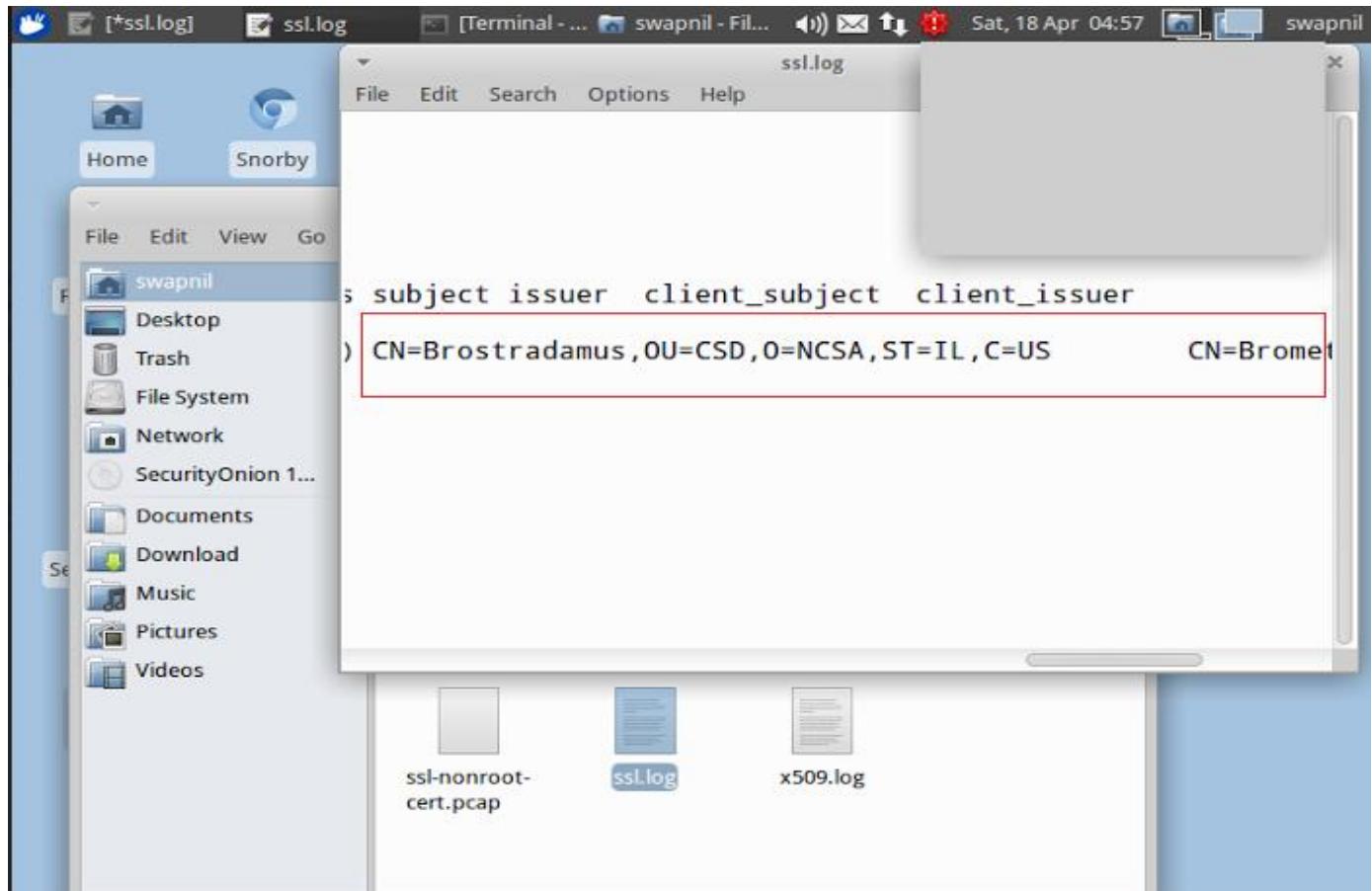
```
nilesh@nilesh-virtual-machine:~$ openssl x509 -in cert.der -inform
DER -outform DER | hexdump -v -e '1/1 "\\\x"' -e '1/1 "%02X"' > my-
cert-hexesc.der
nilesh@nilesh-virtual-machine:~$
```

Part 2: Log More Than Just Server Cert.Subject:

In the first exercise, we include the issuer of the server's certificate information. So, we create a new .bro file called rootissuer.bro. We need to redefine the SSL logging unit (SSL::Info) and handle x509_certificate event. rootissuer.bro file contains:

```
redef record SSL::Info += {
root_issuer: string &log &optional;
};
event x509_certificate(c: connection, cert: X509, is_server: bool, chain_idx: count, chain_len: count, der_cert:
string)
{
c$ssl$root_issuer = cert$issuer;
}
```

And then run the following command:
bro -r ssl-nonroot-cert.pcap rootissuer.bro



Part 3: Using the Notice Framework:

This exercise asks us to log the event when someone is using sslv2. We use given a pcap file and we just need to run Bro to check if some client is using SSLv2 and if so, create log.

```
module SSL;
redef enum Notice::Type += {
SSLv2_Client_Hello
};
event ssl_client_hello(c: connection, version: count, possible_ts: time, client_random:string, session_id: string,
ciphers: vector of count)

{
if ( version == SSLv2 )
{
local message = fmt("SSL client %s sent v2 hello", c$id$orig_h);
local ident = fmt("%s", c$id$orig_h);
NOTICE ([$note=SSLv2_Client_Hello,
$msg=message,
$conn=c,
$identifier=ident]);
}
}
```

The screenshot shows a network traffic analysis interface. A specific packet is highlighted with a red border. The packet details are as follows:

- Protocol: TCP
- SSLv2 Client Hello
- Client IP: 141.142.228.5
- Port: 443
- Server IP: 74.125.225.83
- Port: 443

The status bar at the bottom right indicates "bro Notice::ACTION LOG".

Ans. 1-E:

<https://www.bro.org/bro-workshop-2011/solutions/incident-response/index.html>

Exercise: Intelligent based incident response: A typical incidents response tasks begin with a piece of obtained intelligence that security analyst uses as starting point in any investigation. The workshop session features several scenario of same kind.

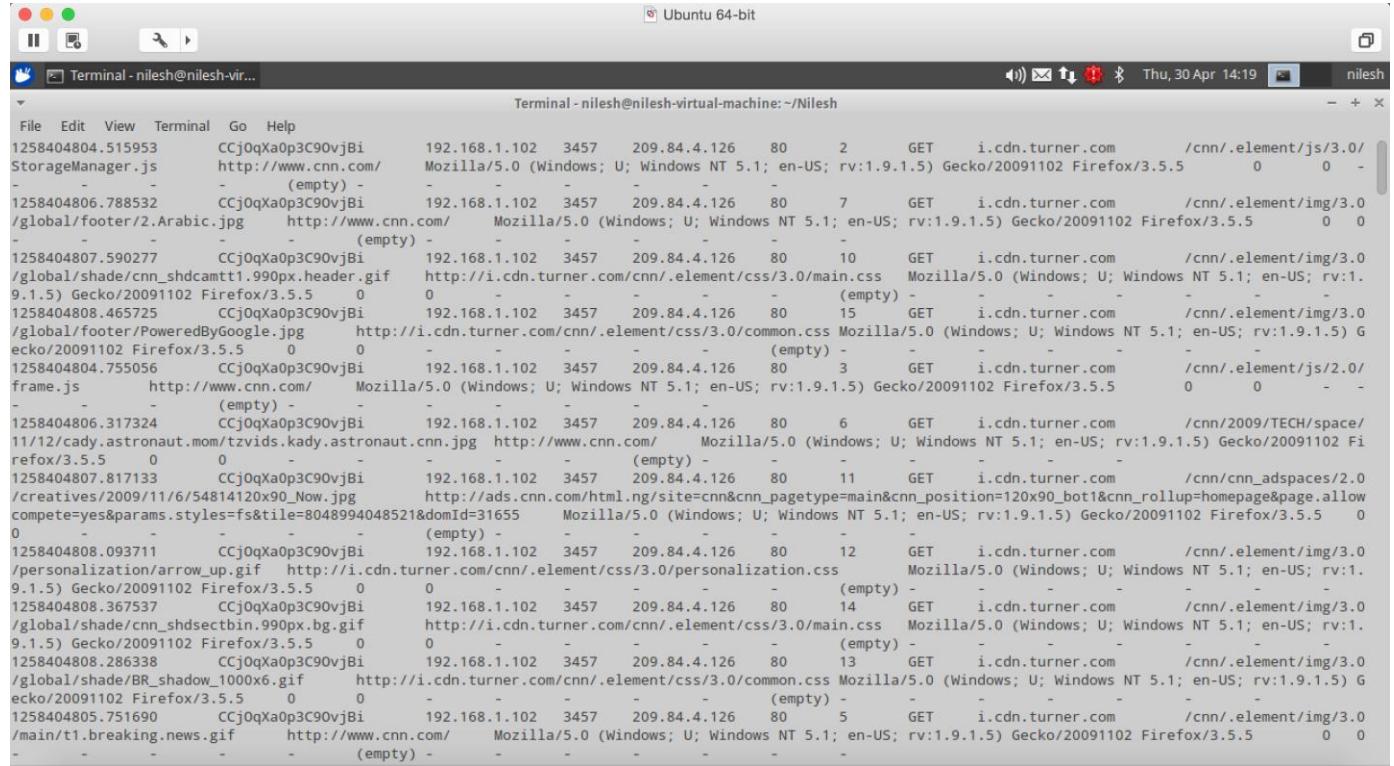
Part-1: Ill consideration authorization: It has been observed that many users will use same password for different website, which may have weak security scheme. From this exercise, we are trying to determine that which webserver will have weak security scheme.

(1). Firstly, we will download the tcpdump named illauth.pcap file and try to generate log files from that using bro scripts. We are looking for http header which will contain Basic and Digest authentication and for basic authentication, it will contain username and password in the header. The header will contain Basic and string and a encoded string which contain username and password. Bro will fetch all the basic authentication header information in HTTP requests and store it in http.log file. Below are the snapshots.

The screenshot shows a VMware Fusion terminal window titled "Terminal - nilesh@nilesh-vir...". The terminal output is as follows:

```
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~$ cd Niles
nilesh@nilesh-virtual-machine:~/Niles$ bro -r /home/nilesh/illauth.pcap
1258358417.399292 warning in /opt/bro/share/bro/base/misc/find-checksum-offloading.bro, line 54: Your trace file likely has invalid IP checksums, most likely from NIC checksum offloading.
nilesh@nilesh-virtual-machine:~/Niles$ ls
conn.log  dhcp.log  dns.log  dpd.log  files.log  http.log  packet_filter.log  reporter.log  smtp.log  ssl.log  weird.log
nilesh@nilesh-virtual-machine:~/Niles$
```

Below is the snapshot of http.log file, which will have all the information regarding all IPs.



```
Ubuntu 64-bit
Terminal - nilesh@nilesh-vir...
Terminal - nilesh@nilesh-virtual-machine:~/Nilesh
File Edit View Terminal Go Help
1258404804.515953 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 2 GET i.cdn.turner.com /cnn/.element/js/3.0/
StorageManager.js http://www.cnn.com/ Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404806.788532 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 7 GET i.cdn.turner.com /cnn/.element/img/3.0
/global/footer/2.Arabic.jpg http://www.cnn.com/ Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404807.590277 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 10 GET i.cdn.turner.com /cnn/.element/img/3.0
/global/shade/cnn_shdcantbin.990px.header.gif http://i.cdn.turner.com/cnn/.element/css/3.0/main.css Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404808.465725 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 15 GET i.cdn.turner.com /cnn/.element/img/3.0
/global/footer/PoweredByGoogle.jpg http://i.cdn.turner.com/cnn/.element/css/3.0/common.css Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404808.755056 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 3 GET i.cdn.turner.com /cnn/.element/js/2.0/
frame.js http://www.cnn.com/ Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404806.317324 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 6 GET i.cdn.turner.com /cnn/2009/TECH/space/
11/12/cady.astronaut.mom.tzvids.kady.astronaut.cnn.jpg http://www.cnn.com/ Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404807.817133 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 11 GET i.cdn.turner.com /cnn/cnn_adspaces/2.0
/creatives/2009/11/6/54814120x90_Now.jpg http://ads.cnn.com/html.ng/site=cnn&cnn_pagetype=main&cnn_position=120x90_bot1&cnn_rollup=homepage&page.allow
compete=yes&params.styles=fs&tile=8048994048521&domId=31655 Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0
0 - (empty) -
1258404808.093711 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 12 GET i.cdn.turner.com /cnn/.element/img/3.0
/personalization/arrow_up.gif http://i.cdn.turner.com/cnn/.element/css/3.0/personalization.css Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404808.367537 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 14 GET i.cdn.turner.com /cnn/.element/img/3.0
/global/shade/cnn_shdsectbin.990px.bg.gif http://i.cdn.turner.com/cnn/.element/css/3.0/main.css Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404808.286338 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 13 GET i.cdn.turner.com /cnn/.element/img/3.0
/global/shade/BR_shadow_1000x6.gif http://i.cdn.turner.com/cnn/.element/css/3.0/common.css Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
1258404805.751690 CCj0qXa0p3C90vjbI 192.168.1.102 3457 209.84.4.126 80 5 GET i.cdn.turner.com /cnn/.element/img/3.0
/main/t1.breaking.news.gif http://www.cnn.com/ Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 0 0 -
- (empty) -
```



```
Terminal - nilesh@nilesh-vir...
Terminal - nilesh@nilesh-virtual-machine:~/Nilesh
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut id.orig_h id.resp_h username password < http.log | awk '$3 != "-"' 
192.168.121.175 192.168.121.176 hbdairey4
192.168.121.175 192.168.121.176 hbdairey4
nilesh@nilesh-virtual-machine:~/Nilesh$
```

The output of the files shows which IP has weak authentication.

(2). In this part, we are trying to determine the user password for that IP. We will run the script with different parameter as argument so that it will convert blank details to password. Below are the snapshot for the same.

```
Applications Menu Terminal - nilesh@nilesh-virtual-machine: ~/Nilesh
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~/Nilesh$ bro -r illauth.pcap "HTTP::default_capture_password=T"
nilesh@nilesh-virtual-machine:~/Nilesh$ fatal error in <params>, line 1: bro: problem with trace file illauth.pcap - illauth.pcap: No such file or directory
nilesh@nilesh-virtual-machine:~/Nilesh$ ls
conn.log dhcp.log dns.log dpd.log files.log http.log packet_filter.log reporter.log smtp.log ssl.log weird.log
nilesh@nilesh-virtual-machine:~/Nilesh$
```

```
nilesh@nilesh-virtual-machine:~/Nilesh$ bro -r /home/nilesh/illauth.pcap "HTTP::default_capture_password=T"
1258358417.399292 warning in /opt/bro/share/bro/base/misc/find-checksum-offloading.bro, line 54: Your trace file likely has invalid IP checksums, most likely
from NIC checksum offloading.
nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ ls
conn.log dhcp.log dns.log dpd.log files.log http.log packet_filter.log reporter.log smtp.log ssl.log weird.log
nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut id.orig_h id.resp_h username password < http.log | awk '$3 != "-"'>
192.168.121.175 192.168.121.176 hbdairy4 cheesecake
192.168.121.175 192.168.121.176 hbdairy4 cheesecake
nilesh@nilesh-virtual-machine:~/Nilesh$
```

Here we can see that the password for the IP 192.168.121.175 is cheesecake.

Part-2:Information theft from web server: Here, we are determining about theft, which occur in the web server.

(1). In this part we are determining the logs generated by bro script on tcpdump theft.pcap. In this case also, we are checking http.log in which we will determine that what type of attack, an attacker is trying to do.

```
nilesh@nilesh-virtual-machine:~/Nilesh$ bro -r /home/nilesh/theft.pcap
1258133127.449275 warning in /opt/bro/share/bro/base/misc/find-checksum-offloading.bro, line 54: Your trace file likely has invalid IP checksums, most likely
from NIC checksum offloading.
nilesh@nilesh-virtual-machine:~/Nilesh$ ls
conn.log dhcp.log dns.log dpd.log files.log http.log packet_filter.log reporter.log ssl.log weird.log
nilesh@nilesh-virtual-machine:~/Nilesh$
```

```
nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut host uri < http.log \
> | awk '$1 ~ /hbdairy/' \
> | head
www.hbdairy.com /index.php?file=.../cheddar.pdf
www.hbdairy.com /index.php?file=.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../.../.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../.../.../.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../.../.../.../.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../.../.../.../.../.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../.../.../.../.../.../.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../.../.../.../.../.../.../cheddar.pdf
www.hbdairy.com /index.php?file=.../.../.../.../.../.../.../.../cheddar.pdf
nilesh@nilesh-virtual-machine:~/Nilesh$
```

As we can see from the logs, it is trying to get cheddar.pdf by trying to access all the directories on the server.

(2). In this we are trying to determine whether he has successfully accessed the file or not. For that we are again check the http.log and status of every request and show the data on console to determine.

```
nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut host status_code < http.log \
> | awk '$1 ~ /hbdairy/ { print $2 } '
> | sort \
> | uniq -c
 1 200
 90 404
nilesh@nilesh-virtual-machine:~/Nilesh$
```

```
nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut host uri status_code < http.log \
> | awk '$1 ~ /hbdairy/ && $3 != "404"'
www.hbdairy.com /index.php?file=\\x0\\xae\\xc0\\xae\\xc0\\xaf\\xc0\\xae\\xc0\\xaf\\xc0\\xae\\xc0\\xafcheddar.pdf      200
nilesh@nilesh-virtual-machine:~/Nilesh$
```

From above, we have seen that it has tried 91 times in which 90 times attacker has faced failure and for one time, he gets success.

Part-3:Email Leakage: As part of this exercise, we are trying to determine that someone has shared the username to attacker and now attacker is trying to send unencrypted email.

(1). We are trying to find out the source of the attacker if he sent the unencrypted file or not. Below are the snapshot to find out the one.

```
nilesh@nilesh-virtual-machine:~/Nilesh$ bro -r /home/nilesh/email.pcap
1302419241.847851 warning in /opt/bro/share/bro/base/misc/find-checksum-offloading.bro, line 54: Your trace file likely has invalid TCP checksums, most likely from NIC checksum offloading.
nilesh@nilesh-virtual-machine:~/Nilesh$ ls
conn.log dhcp.log dns.log files.log packet_filter.log reporter.log smtp.log weird.log
nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ grep hbdairy smtp.log
1302419719.507237    CGtIod2gnNrfoYEH6    192.168.121.179 51158    192.168.121.176 25      1      [192.168.121.179]      <lesharq@dchlaw.com>      <mond
o.cheeze@hbdairy.com>  Sun, 17 Apr 2011 04:44:08 -0400 lesharq <lesharq@dchlaw.com>  mondo.cheeze@hbdairy.com      -      <1303029848.21831.7.camel@see
d-desktop>      -      [Confidential] advice      -      -      -      250 2.0.0 Ok: queued as 1774A73E60      192.168.121.176,192.168.121.179 Evolution 2.2
6.1      F      FMqjvz2vl2w74IDnA6,FAhpVH1rNrOi4Mm9uf
nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut uid from to subject < smtp.log | awk '$1 == "WVVfjCeD7"'
```

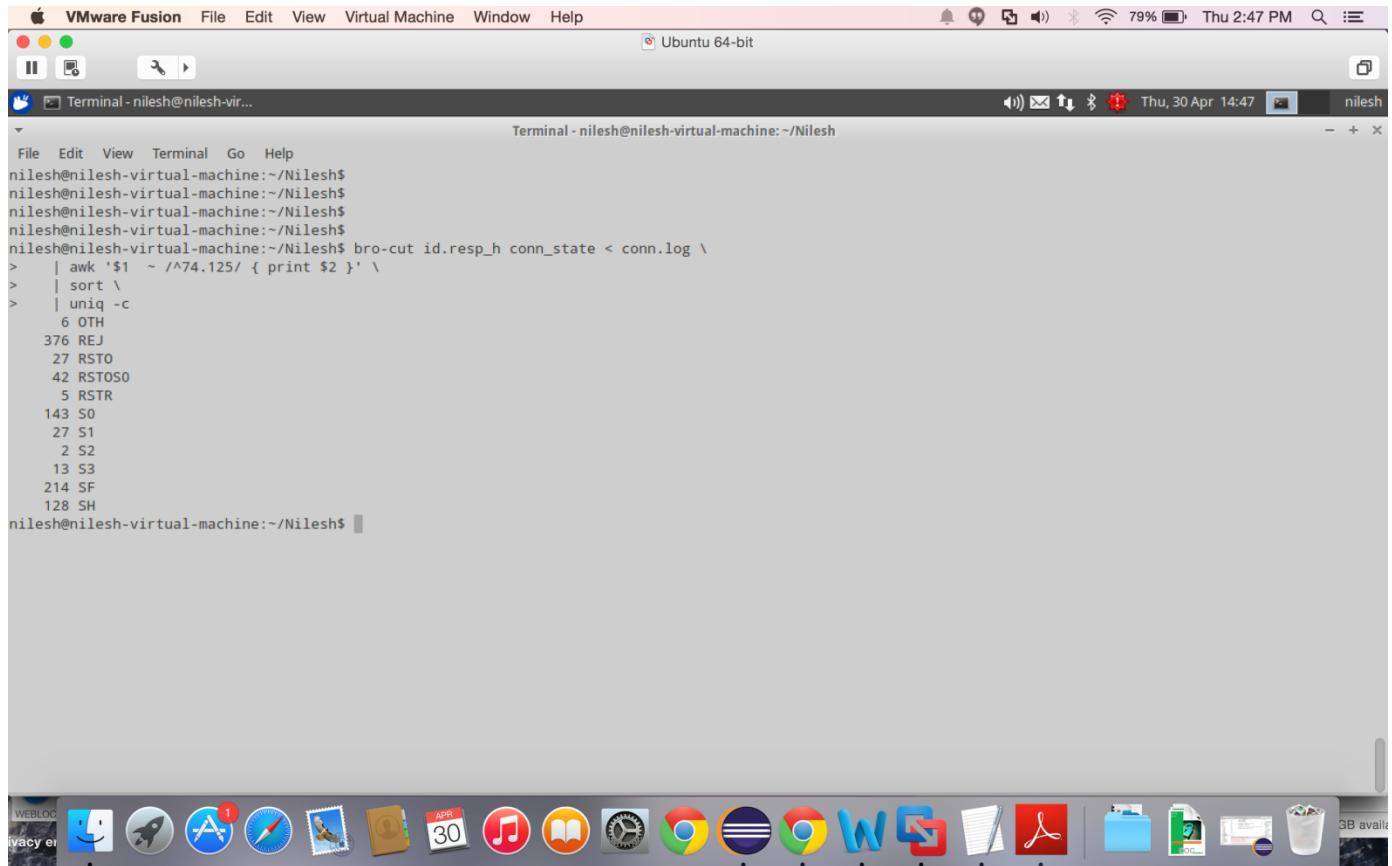
As part of this file no one is trying to send the unencrypted file to any email address. But he is sending, then we will be able to determine the email of the sender who is sending unencrypted file.

(2). Now, we are trying to determine who authored the document and what are the linked inside the pdf document. This can be achieved by reading the pdf file line by line and store the data into disk. Bro will not read the file on its own but there is an option to do the same in bro. By using **file_type**, we can read the document by below statement.

```
bro -r email.pcap -e 'redef SMTP::extract_file_types = /application/pdf/;'
```

This will smtp-entity file, which we can read and check.

Part-4:YouTube becomes NoTube: When you are giving access to someone to a particular site. And that person will remove/revoke the access of the other users of the same group.



```
VMware Fusion File Edit View Virtual Machine Window Help
Ubuntu 64-bit
Terminal - nilesh@nilesh-virtual-machine:~/Nilesh
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut id.resp_h conn_state < conn.log \
> | awk '$1 ~ /^74.125/ { print $2 }' \
> | sort \
> | uniq -c
6 OTH
376 REJ
27 RSTO
42 RSTOSO
5 RSTR
143 S0
27 S1
2 S2
13 S3
214 SF
128 SH
nilesh@nilesh-virtual-machine:~/Nilesh$
```

Here we have seen that 376 failed request has been sent to server. This might be because of an attacker sees the pattern of the IP address and block the request from it.

(2). We are determining the downtime of youtube connection. We will search on the basis of connection state which is equal to REJ in our case.



```
Terminal - nilesh@nilesh-virtual-machine:~/Nilesh
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut id.resp_h conn_state ts < conn.log \
> | awk '$1 ~ /^74.125/ && $2 == "REJ" { print $3 }' \
> | sort \
> | head -n 1
1258409589.906464
nilesh@nilesh-virtual-machine:~/Nilesh$ nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut id.resp_h conn_state ts < conn.log \
> | awk '$1 ~ /^74.125/ && $2 == "REJ" { print $3 }' \
> | sort \
> | tail -n 1
1258411809.009357
nilesh@nilesh-virtual-machine:~/Nilesh$ echo "1258411809.009357-1258409589.906464" | bc
2219.102893
nilesh@nilesh-virtual-machine:~/Nilesh$
```

(3). Finding out who all the affected from this outage. For this we will check in the connection log file and check the rejection state of the request, which we have already determined. Now we will fetch the id.orig_h corresponding to the rejection status. Snapshot is below.

```
Terminal - nilesh@nilesh-virtual-machine:~/Nilesh
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut id.resp_h conn_state id.orig_h < conn.log \
> | awk '$1 ~ /^74.125/ && $2 == "REJ" { print $3 }' \
> | sort | uniq -c
 176 192.168.121.147
   64 192.168.121.148
   38 192.168.121.149
   98 192.168.121.150
nilesh@nilesh-virtual-machine:~/Nilesh$
```

Part 5:The Mysterious DairyStock Transaction: In this part, we are trying to check, who the unauthorized/bogus transaction with the Internet security issues. For this, we were given a tcpdump dairystock.pcap. By this part, we are learning how to identify bogus transaction over TCP.

```
Terminal - nilesh@nilesh-virtual-machine:~/Nilesh
File Edit View Terminal Go Help
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$ bro -r /home/nilesh/dairystock.pcap
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$ bro-cut id.orig_h id.orig_p id.resp_h method host uri < http.log \
> | awk '$4 == "POST" && $5 ~ /dairy/ { print $1, $2, $3, $5, $6 }'
192.168.121.147 48205 85.47.63.142 www.dairystock.com /index.php
192.168.121.177 53796 85.47.63.142 www.dairystock.com /transfer.php
192.168.121.184 56436 85.47.63.142 www.dairystock.com /stock.php
192.168.121.167 33447 85.47.63.142 www.dairystock.com /stock.php
192.168.121.157 51135 85.47.63.142 www.dairystock.com /stock.php
192.168.121.147 48207 85.47.63.142 www.dairystock.com /stock.php
192.168.121.177 53796 85.47.63.142 www.dairystock.com /stock.php
192.168.121.157 51136 85.47.63.142 www.dairystock.com /stock.php
192.168.121.167 33448 85.47.63.142 www.dairystock.com /transfer.php
192.168.121.157 51137 85.47.63.142 www.dairystock.com /transfer.php
192.168.121.184 56469 85.47.63.142 www.dairystock.com /transfer.php
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$ 
nilesh@nilesh-virtual-machine:~/Nilesh$
```

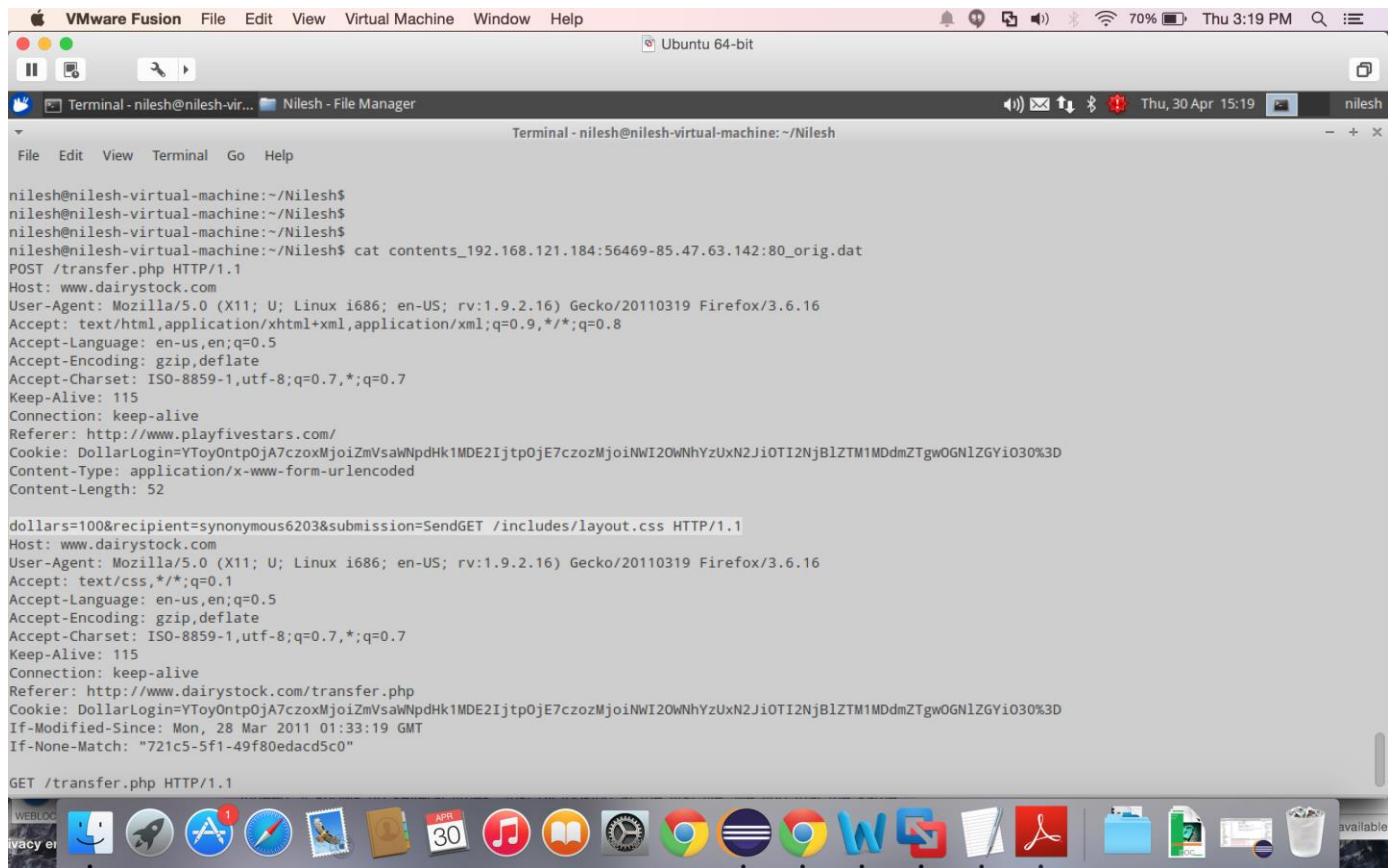
After extracting http log file, we are trying to check, who send the POST request from dairystock.com and send some data file over it.

Now, we have id.orig-h (sender ID) from http log. Now we are creating a script extract.bro in which we are writing IPs, which are generated in previous results.

Now, after generating script, we are again extracting logs from pcap file using origin ID. By doing this, we will get dat file for each corresponding IP in the script. And the connection that involves sending transfer.php, we will have there IP and port number. Now, if we check the file with that IP and port number, we will see the transaction details inside it.

```
Terminal - nilesh@nilesh-virtual-machine:~  
File Edit View Terminal Go Help  
@load base/protocols/http  
  
event connection_established(c: connection)  
{  
    if ( (c$id$orig_h == 192.168.121.147 ||  
          c$id$orig_h == 192.168.121.157 ||  
          c$id$orig_h == 192.168.121.167 ||  
          c$id$orig_h == 192.168.121.177 ||  
          c$id$orig_h == 192.168.121.184) &&  
          c$id$resp_h == 85.47.63.142 )  
    {  
        c$extract_orig = T;  
        c$extract_resp = T;  
    }  
}  
-  
-
```

```
Terminal - nilesh@nilesh-virtual-machine:~/Nilesh  
File Edit View Terminal Go Help  
nilesh@nilesh-virtual-machine:~/Nilesh$  
nilesh@nilesh-virtual-machine:~/Nilesh$  
nilesh@nilesh-virtual-machine:~/Nilesh$ bro -r /home/nilesh/dairystock.pcap /home/nilesh/extract.bro  
nilesh@nilesh-virtual-machine:~/Nilesh$  
nilesh@nilesh-virtual-machine:~/Nilesh$  
nilesh@nilesh-virtual-machine:~/Nilesh$ ls  
conn.log  
contents_192.168.121.147:48205-85.47.63.142:80_orig.dat  
contents_192.168.121.147:48205-85.47.63.142:80_resp.dat  
contents_192.168.121.147:48206-85.47.63.142:80_orig.dat  
contents_192.168.121.147:48206-85.47.63.142:80_resp.dat  
contents_192.168.121.147:48207-85.47.63.142:80_orig.dat  
contents_192.168.121.147:48207-85.47.63.142:80_resp.dat  
contents_192.168.121.157:45032-85.47.63.142:80_orig.dat  
contents_192.168.121.157:45032-85.47.63.142:80_resp.dat  
contents_192.168.121.157:51135-85.47.63.142:80_orig.dat  
contents_192.168.121.157:51135-85.47.63.142:80_resp.dat  
contents_192.168.121.157:51136-85.47.63.142:80_orig.dat  
contents_192.168.121.157:51136-85.47.63.142:80_resp.dat  
contents_192.168.121.157:51137-85.47.63.142:80_orig.dat  
contents_192.168.121.157:51137-85.47.63.142:80_resp.dat  
contents_192.168.121.167:33447-85.47.63.142:80_orig.dat  
contents_192.168.121.167:33447-85.47.63.142:80_resp.dat  
contents_192.168.121.167:33448-85.47.63.142:80_orig.dat  
contents_192.168.121.167:33448-85.47.63.142:80_resp.dat  
nilesh@nilesh-virtual-machine:~/Nilesh$
```



Ans. 1-F: <http://www.bro.org/bro-workshop-2011/solutions/advanced-http/index.html>

Exercise: Advanced HTTP Traffic Analysis:

Part 1: Building a Sidejacking Detector:

This part of exercise demonstrates us how to create a sidejacking detector. Sidejacking is when someone steals your cookies and uses it to impersonate you on a particular website. The tutorial gives the example of Brooke using Brody's cookies to impersonate him on Twitter. Network operators and incident responders clearly would like to find out when their users' security and privacy is compromised. How do we detect it?

At a very basic level, we have to monitor each session cookie and observe whether it appears in more than one user context. That is, if one IP address uses a cookie previously used by another IP address, we have likely witnessed a sidejacking incident.

The second thought is that of session cookie reuse where we reuse the same cookie on different browsers.

Exercise1: It asks us to add the cookie information to the http log file, since it is not added by default. We create a .bro file to do this.

```
GNU nano 2.2.6          FILE: sidejack-detect.bro

module HTTP;

export {
    redef record Info += {
        # Extend this record with an optional, logged field named "cookie".
        cookie: string &log &optional;
    };
}

# Track the cookie value inside HTTP.
event http_header(c: connection, is_orig: bool, name: string, value: string)
{
    if ( ! is_orig )
        return;

    # Write the cookie value into the HTTP state of the connection.
    if ( name == "COOKIE" )
        c$http$cookie = value;
}
```

Then Used following command:

bro -r twitter.pcap sidejack-detect.bro

Now, the cookie field has been added to the log file.

```

.vvirtual-machine:~/sidejack$ grep "cookie" http.log
uid      id.orig_h      id.orig_p      id.resp_h      id.resp_
depth    method host      uri      referrer      user_agent      r
response_body_len      status_code      status_msg      info_cod
filename      tags      username      password      p
uids      orig_mime_types      resp_fuids      resp_mime_types      cookie
.vvirtual-machine:~/sidejack$ cat http.log

```

Next exercise is to check whether the cookie has been seen with different user agents.
We write the below .bro file.

```

module HTTP;

export {
    redef enum Notice::Type += {
        ## Cookie reuse by a different user agent
        SessionCookieReuse
    };

    # We track the cookie inside the HTTP state of the connection.
    redef record Info += {
        cookie: string &log &optional;
    };
}

const twitter_cookie_keys = set("_twitter_sess", "auth_token");

# Map cookies to user agents.
global cookies: table[string] of string;

```

It keeps track of which cookie is using what browser and as soon as the same cookie uses a different browser, it detects it.

Used following commands:

nano solution.bro

bro -r twitter.pcap solution.bro

cat notice.log

```

#empty_field    (empty)
#unset_field   -
#path  notice
#open  2015-04-15-09-34-05
#fields ts      uid      id.orig_h      id.orig_p      id.resp_h      id.res
p      fuid     file_mime_type   file_desc      proto      note      msg       sub
rc     dst      p          n          peer_descr   actions  suppress_for  dropped
emote_location.country_code  remote_location.region remote_location.city
emote_location.latitude  remote_location.longitude
#types  time      string     addr      port      string      string     string
num     enum      string     string     addr      port      count      string   set[er
]      interval   bool      string     string     string      double    double
1320257580.095086     C0cH1123JafADEbB53     192.150.187.147 51742   199.59
49.198  80      -         -        -          tcp      HTTP::SessionCookieReuse
eused Twitter session via cookie _twitter_sess=BAh7DzoVaw5fbmV3X3VzZXJfZmxvdzA

```

Part 2: Building a Facebook Chat Analyzer:

This part tells us to recover the transcript from a Facebook chat. In first part of this exercise, we create a data structure ChatMessage and write a function for it.

```

## Reassemble the HTTP body of replies and look for Facebook chat messages.
event http_body(c: connection, is_orig: bool, data: string, size: count)
{
    # Only consider chat messages for now.
    if (/^for \(;;\); \{\\"t\":\"msg\".*text\\":\"/ !in data)
        return;

    local msg = parse_fb_message (data);

    local i: Info;
    i$timestamp = msg$timestamp;
    i$chat_from = msg$from;
    i$chat_to = msg$to;
    i$chat_msg = msg$text;

    Log:::write(Facebook::LOG, i);
}

```

Running this .bro file is the next part.

```
bro -r url.pcap facebook-solution.bro
```

```
cat facebook.log
```

```

#path  facebook
#open  2015-04-15-09-53-29
#fields timestamp      chat_from      chat_to      chat_msg
#types  string        string        string
1303218454567  Mondo Cheeze   Udder Kaos    So I need the URL, dude. What is it?
1303218465938  Udder Kaos     Mondo Cheeze   the URL?
1303218474259  Mondo Cheeze   Udder Kaos    Yeah for the secret image

```

1303218454567	Mondo Cheeze	Udder Kaos	So I need the URL, dude. What is it?
1303218465938	Udder Kaos	Mondo Cheeze	the URL?
1303218474259	Mondo Cheeze	Udder Kaos	Yeah for the secret image
1303218481721	Udder Kaos	Mondo Cheeze	ok lemme see
1303218495626	Mondo Cheeze	Udder Kaos	Someone could be sniffing this conversation, be sure to send it safely
1303218503972	Udder Kaos	Mondo Cheeze	?
1303218570782	Mondo Cheeze	Udder Kaos	Cmon we talked about this. Encrypt it with WondersCipher-92 and send me the Base64 encoding of the hex. Usual key.
1303218587568	Udder Kaos	Mondo Cheeze	'k. So here it is:
1303218595067	Udder Kaos	Mondo Cheeze	NmQwMDJjZDdhZTdlYmYxNTc5MGVjZDc1YTYxNDk10GE0ZTRhYjAzOTVi
1303218618252	Mondo Cheeze	Udder Kaos	What's the IV
1303218624712	Udder Kaos	Mondo Cheeze	huh?
1303218637197	Mondo Cheeze	Udder Kaos	Initialization vector, you maroon. WC-92 is a stream cipher, you know
1303218667601	Udder Kaos	Mondo Cheeze	oh yeah. I used my birthday, all as one number.
1303218685436	Udder Kaos	Mondo Cheeze	you *do* remember it, right?
1303218700515	Mondo Cheeze	Udder Kaos	yeah your an April Fool, not hard to remember
1303218710402	Udder Kaos	Mondo Cheeze	heh
1303218718486	Mondo Cheeze	Udder Kaos	K gimme a sec to decrypt then.
1303218733463	Mondo Cheeze	Udder Kaos	Hey idiot this isn't the secret, it's Google's home page.
1303218745028	Udder Kaos	Mondo Cheeze	whoops hang on, blew my cut&paste
1303218767633	Udder Kaos	Mondo Cheeze	okay, here's the right one:
1303218776922	Udder Kaos	Mondo Cheeze	NmQwMDJjZDdhZTdlYmYwMDY3MGRjZDdlYjA1NDlhODQ0ZjA1YmEyNDRm
1303218800303	Mondo Cheeze	Udder Kaos	And?
1303218807537	Udder Kaos	Mondo Cheeze	and what
1303218815022	Mondo Cheeze	Udder Kaos	What's the IV
1303218824330	Udder Kaos	Mondo Cheeze	huh?

In this way, we have used Bro to extract the transcript of the Facebook chat.

Extra credit:

Can you reconstruct the encrypted URL?

Yes, we can. using the same IV twice is not safe.

We define some variables below:

B64D:= function that decodes a Base64 encoded string

P1:= "http://www.google.com"

P2:= (unknown)

C1:= B64D (NmQwM...zOTVi)

C2:= B64D (NmQwM...mNDRm)

\oplus := XOR operation

If R is the keystream used,

$$C1 \oplus C2 = (P1 \oplus R) \oplus (P2 \oplus R) = P1 \oplus P2$$

Which is same as

$$P2 = C1 \oplus C2 \oplus P1$$

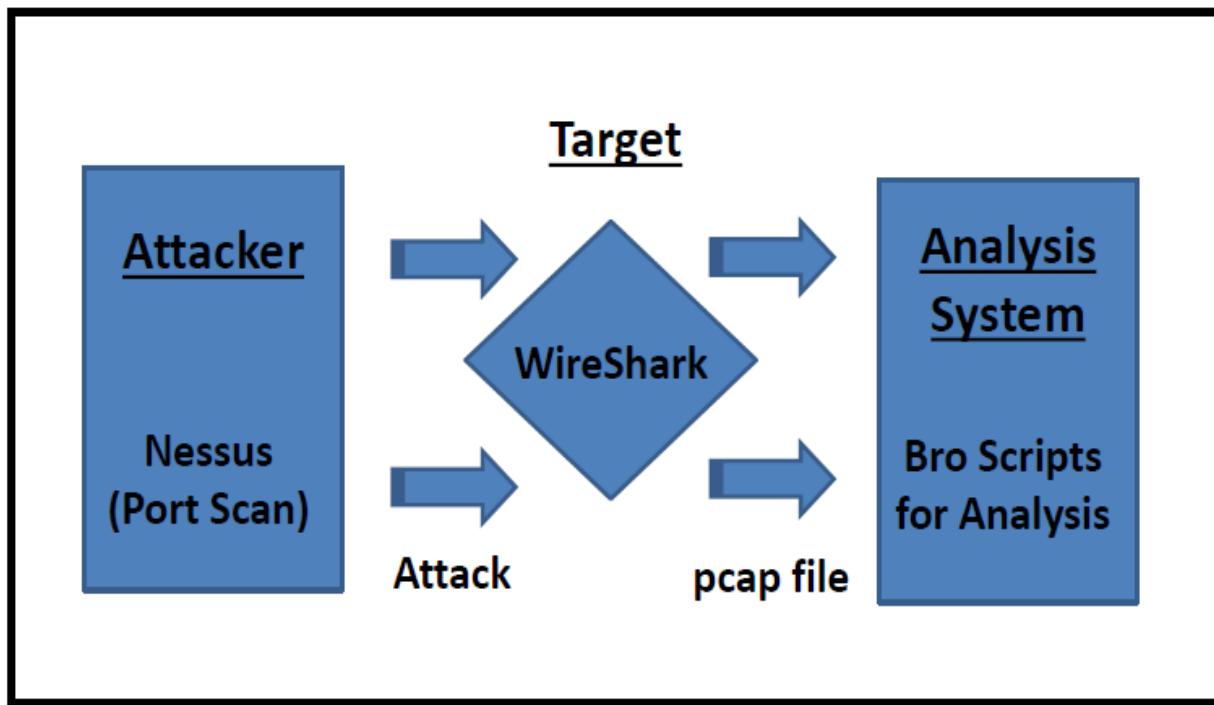
We can find the value of P2 by plugging in the values of C1, C2 and P1. In the above example it is <http://bit.ly/hbdairy>

Ans. 2:

2. Port Scan Attack Using Nessus

Most attacks start with a recognition phase, where an attacker looks for attack vectors in one or several victim systems. Port scanning is arguably the most widely used technique by both worms and human attackers to probe for vulnerabilities in Internet systems. In our project we attempt to address and implement the problem of how to efficiently and reliably port scans can be detected without too many false positives.

The below figure briefly explains what we are going to implement in our project, the main agenda is going to be to detect the port scan attack that we will perform using the penetration testing tool called Nessus and Wireshark by running a bro script on security onion.



The below section gives us a brief understanding of functionality of each of the components mentioned above.

Port Scan Attack

Port scanning can be defined as hostile Internet searches for open doors or ports, through which intruders gain access to computers. This technique consists of sending a message to a port and listening for an answer. The received response indicates the port status and can be helpful in determining a host's operating system and other information relevant to launching a future attack.

Ports are like little doors on our systems. Most packets leaving the machine come out of a certain door. They are destined for another door on another system. There are two different protocols that use ports: TCP and UDP.

An attacker launches a port scan to see what ports are open, with a listening service, on the machine. A port scan attack, therefore, occurs when an attacker sends packets to your machine, varying the destination port. The attacker can use this to find out what services are running and to get a pretty good idea of the operating system it has.

Hackers can compromise the vulnerable hosts and can either take over their resources or use them as tools for future attacks. With so many different protocols and countless implementations of each for different platforms, the launch of an effective attack often begins with a separate process of identifying potential victims.

Port scans can be classified into **three basic types** based on the pattern of target destinations and ports the scan explores, and they are:

Vertical Scans: The vertical scan is a port scan that targets several destination ports on a single host. Naively executed, this scan is among the easiest to detect because only local (singlehost) detection mechanisms are required.

Horizontal Scans: A horizontal scan is a port scan that targets the same port on several hosts. Most often the attacker is aware of a particular vulnerability and wishes to find susceptible machines. One would expect to see many horizontal scans for a particular port immediately following the publicizing of a vulnerability on that port.

Block Scans: Some attackers combine vertical and horizontal scanning styles into large sweeps of the address-port space.

Nessus

The penetration testing tool we have chosen is Nessus. Nessus is a remote security scanning tool, which scans a computer and raises an alert if it discovers any vulnerabilities that malicious hackers could use to gain access to any computer you have connected to a network. It does this by running over 1200 checks on a given computer, testing to see if any of these attacks could be used to break into the computer or otherwise harm it. If you are an administrator in charge of any computer (or group of computers) connected to the internet, Nessus is a great tool help keep their domains free of the easy vulnerabilities that hackers and viruses commonly look to exploit.

Why we used Nessus??

Unlike other scanners, Nessus does not make assumptions about your server configuration (such as assuming that port 80 must be the only web server) that can cause other scanners to miss real vulnerabilities.

Nessus is very extensible, providing a scripting language for you to write tests specific to your system once you become more familiar with the tool. It is also provides a plug-in interface, and many free plug-ins are available from the Nessus plug-in site. These plugs are often specific to detecting a common virus or vulnerability.

Up to date information about new vulnerabilities and attacks. The Nessus team updates the list of what vulnerabilities to check for on a daily basis in order to minimize the window between an exploit appearing in the wild, and you being able to detect it with Nessus.

Wireshark

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color-coding and other features that let you dig deep into network traffic and inspect individual packets. We will use Wireshark to capture the the traffic while the port scan by Nessus is running and then the resulting **.pcap file** will be analyzed using bro script on security onion.

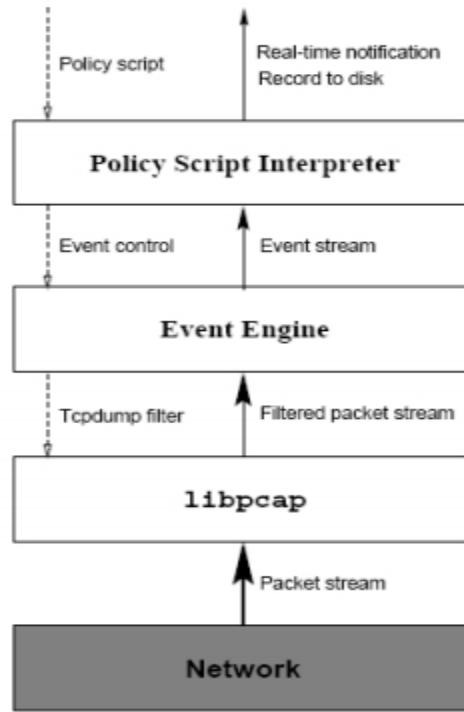
Bro

Bro is an intrusion detection system. Bro was intentionally a stand alone system for detecting network intruders in real time. Bro is open source software, written in C and is capable of deep & stateful packet inspection at very high speed. Bro IDS is a powerful real-time network traffic analyzing tool.

Bro - Internal Architecture:

Bro internal architecture consists roughly of the following elements.

1. libpcap
2. an event engine and
3. a policy script interpreter.



Network: Bro needs a physical network connection to get a copy of the network traffic it will analyze. This is normally done by the use of port mirroring functionality in switches/routers or a TAP device. **libpcap :**In order to get traffic data from the physical network, a so-called Application Program Interface (API) is needed. Bro uses libpcap. libpcap is a C/C++ library for network traffic capture. This is the abstract layer between the physical network medium and the operating system. With the use of libpcap Bro filters traffic that is sent to Bro Event Engine.

Event Engine :The filtered network data packages from libpcap are then fed into the next level; the Event engine. This event engine tries to reassemble all the network traffic it gets to known events/patterns as high as possible in the OSI ISO Model.

Policy Script Interpreter :The policy script interpreter processes events from the event engine. For every event handled to the policy script interpreter, it performs the following steps:

1. Look up the corresponding event handler's (semi-)compiled code/script
2. Bind the value(s) of the event(s) to the argument of the handler
3. Interpret the actual event code/script

Bro Log Files:

conn.log — consists of the complete connection log during Bro's run time. The file consists of twenty columns (timestamp, connection ID (unique), source IP, dest IP, source port, dest port etc.).

loaded_scripts.log — shows Bro scripts (*.bro) that were loaded during Bro startup.

notice_policy.bro — shows the current Bro Notice policy.

communication.log — logs for Bro's internal communication between remote and central instances, clusters etc.

conn-summary.log — generated when Bro is terminated. Post processing connection summaries

known_hosts.log — hosts that have performed complete TCP handshake
notice.log — notices that Bro rises
reporter.log — internal messages/warnings/errors for troubleshooting.

Implementation

1.Nessus

The attacker uses Nessus penetration testing tool to implement the Port Scan attack. Below screen shots show how this is performed.

```
root@kali:~# /etc/init.d/nessusd start
$Starting Nessus : .
root@kali:~#
```



After Starting Nessus we will create a policy by running the port scan and giving the ip address of the system we are trying to find the vulnerability.

A screenshot of the Nessus web interface. The top navigation bar shows 'Applications', 'Places', and the date 'Fri May 1, 4:31 PM'. The main title is 'Nessus Home / Policies - Iceweasel'. The URL in the address bar is 'https://kali:8834/html5.html#/policies'. The page displays a 'Policies' section with a 'New Policy' button. A 'Policy Wizards' dialog box is open, listing several pre-defined scanning policies: 'GHOST (glibc) Detection', 'Bash Shellshock Detection', 'PCI Quarterly External Scan' (marked as 'DRAFT'), 'Host Discovery', 'Basic Network Scan', and 'Credentialed Patch Audit'. The 'Host Discovery' policy is highlighted. The bottom right of the dialog shows the copyright notice 'Tenable Security® All Rights Reserved. Nessus Home Version: 5.2.7'.

New Scan / Basic Settings

Basic Settings

- Name: portScan
- Description:
- Policy: portScan (highlighted with a red box)
- Folder: My Scans
- Targets: 192.168.67.131 (highlighted with a red box)

Upload Targets Add File

Below screen shot shows the port scan in happening.

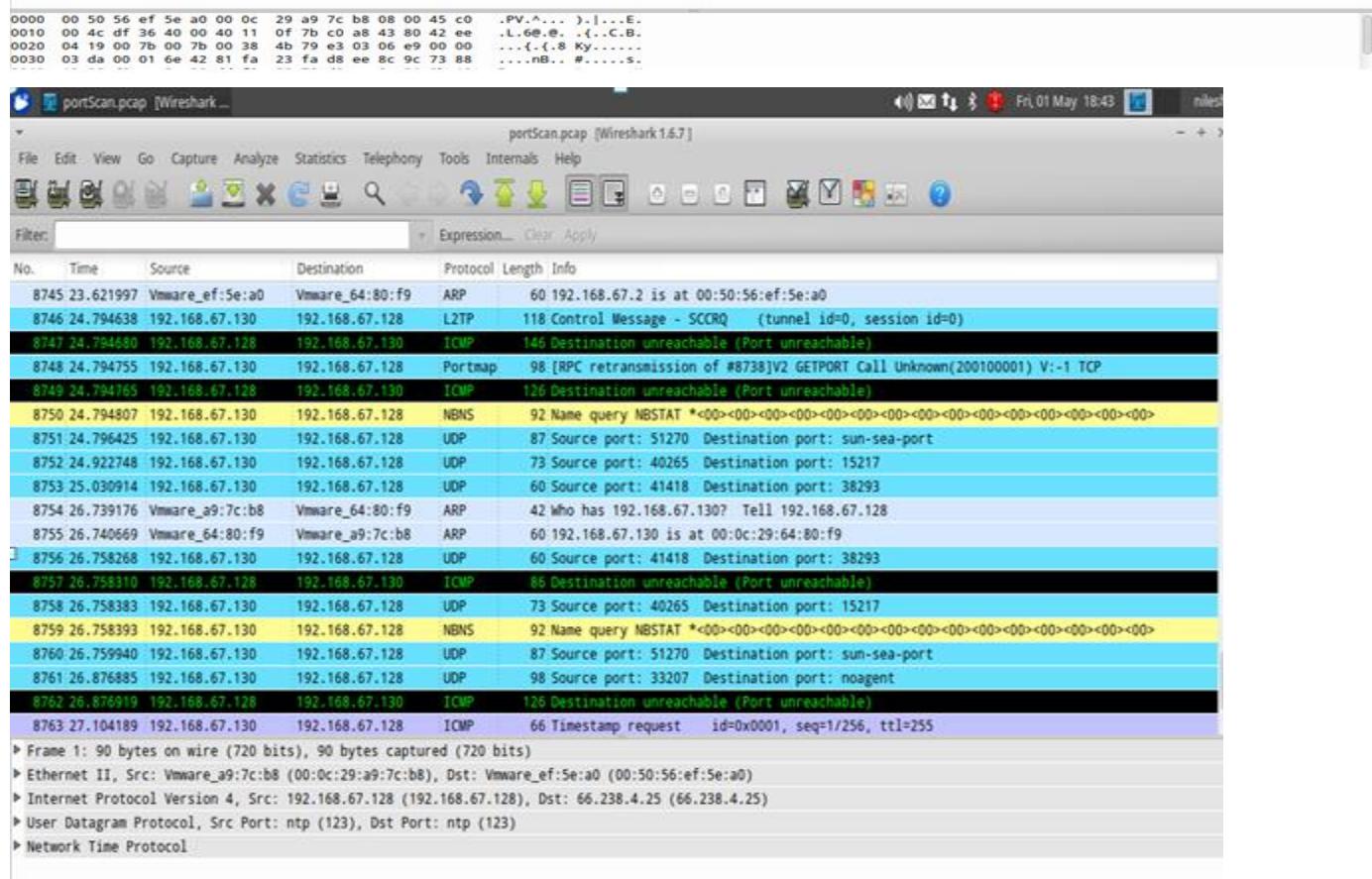
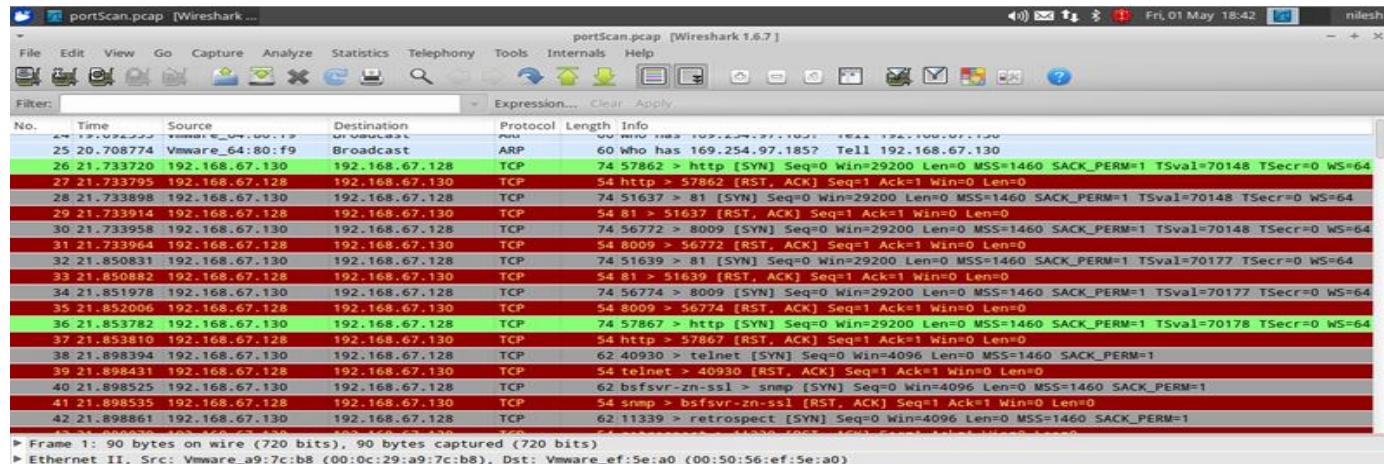
Severity	Plugin Name	Plugin Family	Count
MEDIUM	SSL Certificate Cannot Be Trusted	General	1
MEDIUM	SSL RC4 Cipher Suites Supported	General	1
MEDIUM	SSL Self-Signed Certificate	General	1
LOW	SSH Server CBC Mode Ciphers Enabled	Misc.	1
LOW	SSH Weak MAC Algorithms Enabled	Misc.	1
INFO	Nessus SYN scanner	Port scanners	4
INFO	Service Detection	Service detection	3
INFO	Web Server SSL Port HTTP Traffic Detection	Web Servers	2
INFO	Backported Security Patch Detection (SSH)	General	1
INFO	Common Platform Enumeration (CPE)	General	1
INFO	Device Type	General	1
INFO	Ethernet Card Manufacturer Detection	Misc.	1
INFO	ICMP Timestamp Request Remote Date Disclosure	General	1
INFO	Nessus Scan Information	Settings	1
INFO	Open Port Re-check	General	1

Vulnerabilities

Severity	Count
Info	4
Low	1
Medium	1
High	0
Critical	0

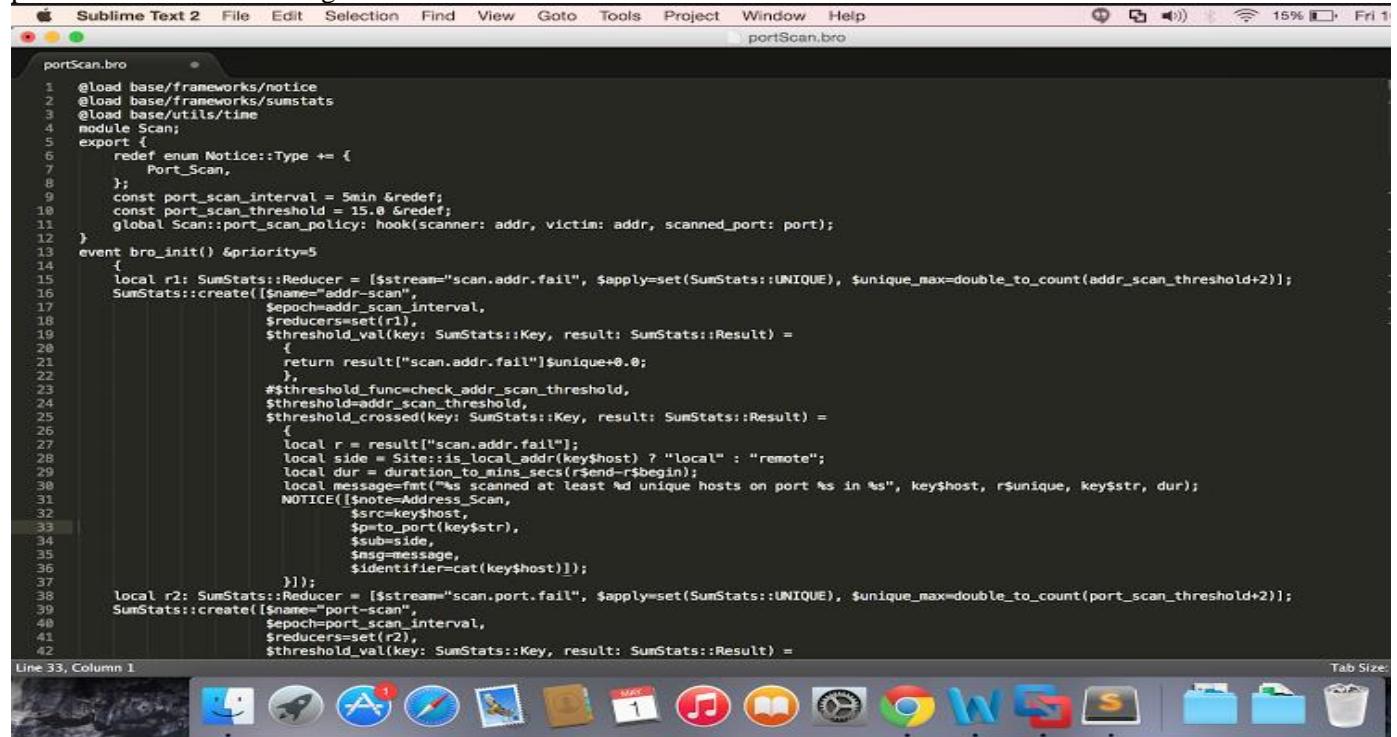
2.WireShark

Now we are going to start the capture on the Wireshark to capture the network activity on the victims system. This is going to produce the .pcap file which we are going to use for bro analysis, checking the tcpdump in Wireshark for port scan.



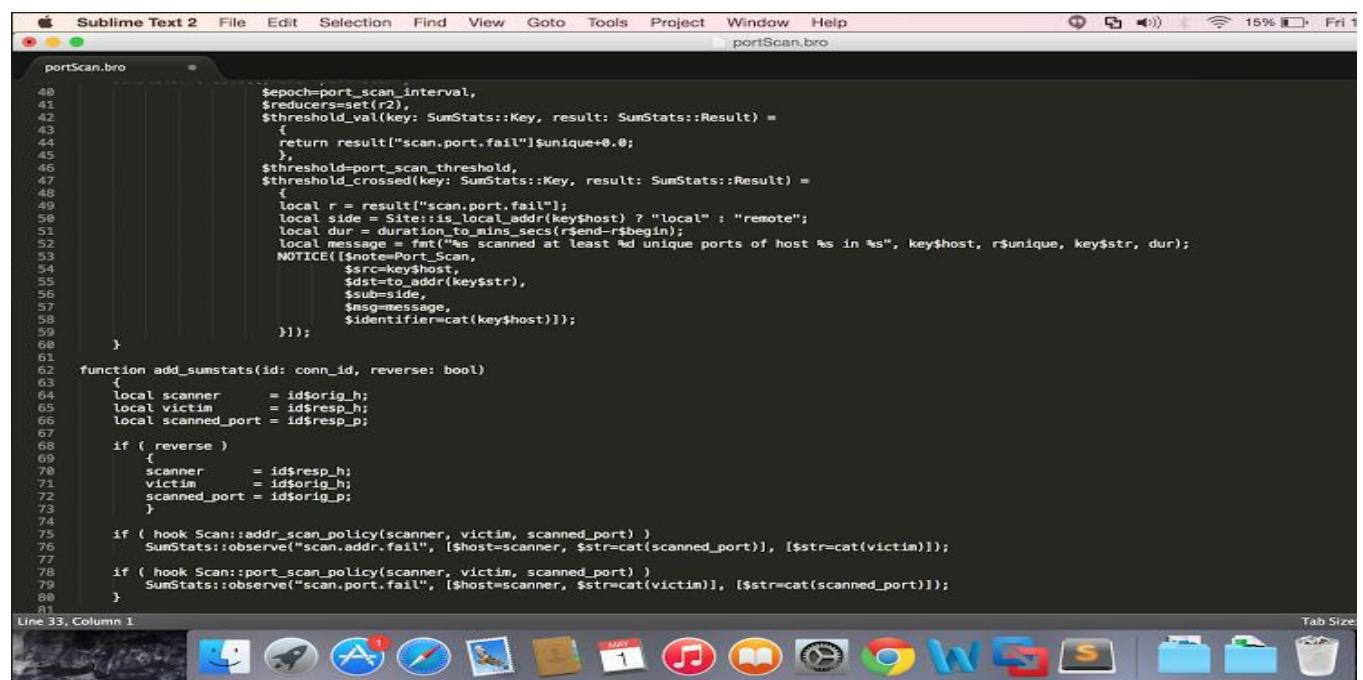
3. Bro

So here we have written a script that detects for the portscans occurring on the network, and it creates an entry in the notice.log file. Below screenshot shows the code we have implemented. This code keeps track of the number of portscans that are occurring in the network.



The screenshot shows the Sublime Text 2 interface with the file 'portScan.bro' open. The code implements a Bro script to detect port scans and log them to the notice.log file. It uses the base/frameworks/notice module and defines a new type 'Port_Scan'. It sets up reducers for address and port fail streams to count unique hosts. It also hooks into the Scan module's port scan policy to log the scan details. The code includes logic to determine if a scan is local or remote based on the source IP.

```
1 @load base/frameworks/notice
2 @load base/frameworks/sumstats
3 @load base/utils/time
4 module Scan;
5 export {
6     redef enum Notice::Type += {
7         Port_Scan,
8     };
9     const port_scan_interval = 5min &redef;
10    const port_scan_threshold = 15.8 &redef;
11    global Scan::port_scan_policy: hook(scanner: addr, victim: addr, scanned_port: port);
12 }
13 event bro_init() &priority=5
14 {
15     local r1: SumStats::Reducer = [$stream="scan.addr.fail", $apply=set(SumStats::UNIQUE), $unique_max=double_to_count(addr_scan_threshold+2)];
16     SumStats::create([$name="addr-scan",
17         $epoch=addr_scan_interval,
18         $reducers=set(r1),
19         $threshold_val(key: SumStats::Key, result: SumStats::Result) =
20             {
21                 return result["scan.addr.fail"]$unique+0.0;
22             },
23             #$threshold_func=check_addr_scan_threshold,
24             $threshold=addr_scan_threshold,
25             $threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
26                 {
27                     local r = result["scan.addr.fail"];
28                     local side = Site::is_local_addr(key$host) ? "local" : "remote";
29                     local dur = duration_to_mins_secs(r$end-r$begin);
30                     local message=fmt("%s scanned at least %d unique hosts on port %s in %s", key$host, r$unique, key$str, dur);
31                     NOTICE([$note=Address_Scan,
32                         $src=key$host,
33                         $proto=port$key$str,
34                         $sub=side,
35                         $msg=message,
36                         $identifier=cat(key$host)]);
37                 });
38     local r2: SumStats::Reducer = [$stream="scan.port.fail", $apply=set(SumStats::UNIQUE), $unique_max=double_to_count(port_scan_threshold+2)];
39     SumStats::create([$name="port-scan",
40         $epoch=port_scan_interval,
41         $reducers=set(r2),
42         $threshold_val(key: SumStats::Key, result: SumStats::Result) =
```



The screenshot shows the continuation of the Sublime Text 2 interface with the file 'portScan.bro' open. The code adds a function 'add_sumstats' which takes an ID, connection ID, and reverse flag. It extracts scanner, victim, and scanned port information from the ID and observes the 'scan.addr.fail' and 'scan.port.fail' streams to update the sumstats module. The logic for determining if a scan is local or remote is also present in this part of the script.

```
40     $epoch=port_scan_interval,
41     $reducers=set(r2),
42     $threshold_val(key: SumStats::Key, result: SumStats::Result) =
43         {
44             return result["scan.port.fail"]$unique+0.0;
45         },
46         $threshold=port_scan_threshold,
47         $threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
48             {
49                 local r = result["scan.port.fail"];
50                 local side = Site::is_local_addr(key$host) ? "local" : "remote";
51                 local dur = duration_to_mins_secs(r$end-r$begin);
52                 local message = fmt("%s scanned at least %d unique ports of host %s in %s", key$host, r$unique, key$str, dur);
53                 NOTICE([$note=Port_Scan,
54                     $src=key$host,
55                     $dst=to_ipaddr(key$str),
56                     $sub=side,
57                     $msg=message,
58                     $identifier=cat(key$host)]);
59             });
60         }
61     function add_sumstats(id: conn_id, reverse: bool)
62     {
63         local scanner      = id$orig_h;
64         local victim       = id$resp_h;
65         local scanned_port = id$resp_p;
66
67         if ( reverse )
68         {
69             scanner      = id$resp_h;
70             victim       = id$orig_h;
71             scanned_port = id$orig_p;
72         }
73
74         if ( hook Scan::addr_scan_policy(scanner, victim, scanned_port) )
75             SumStats::observe("scan.addr.fail", [$host=scanner, $str=cat(scanned_port)], [$str=cat(victim)]);
76
77         if ( hook Scan::port_scan_policy(scanner, victim, scanned_port) )
78             SumStats::observe("scan.port.fail", [$host=scanner, $str=cat(victim)], [$str=cat(scanned_port)]);
79
80     }
```

Below screenshot shows that after running our bro script it made an analysis on the .pcap file to find that 15 ports of the victim were probed. Hence our script was successful at detecting the portscan that occurred on the network.

```
Terminal - nilesh@nilesh-vir... portScan - File Manager portScan.pcap [Wireshark ...] Fri, 01 May 21:38 nilesh
```

```
File Edit View Terminal Go Help
-rw-r--r-- 1 nilesh nilesh 235 May 1 21:03 mybro.bro
nilesh@nilesh-virtual-machine:~/portScan$ pwd
/home/nilesh/portScan
nilesh@nilesh-virtual-machine:~/portScan$ cd ..;/final
nilesh@nilesh-virtual-machine:~/final$ ls -lrt
total 1816
-rw-r--r-- 1 root root 1851392 May 1 21:36 portScan.pcap
-rw-r--r-- 1 nilesh nilesh 6734 May 1 21:37 portScan.bro
nilesh@nilesh-virtual-machine:~/final$ bro -r portScan.pcap portScan.bro
nilesh@nilesh-virtual-machine:~/final$ ls -lrt
total 2788
-rw-r--r-- 1 root root 1851392 May 1 21:36 portScan.pcap
-rw-r--r-- 1 nilesh nilesh 6734 May 1 21:37 portScan.bro
-rw-rw-r-- 1 nilesh nilesh 13812 May 1 21:38 x509.log
-rw-rw-r-- 1 nilesh nilesh 1178 May 1 21:38 weird.log
-rw-rw-r-- 1 nilesh nilesh 473 May 1 21:38 tunnel.log
-rw-rw-r-- 1 nilesh nilesh 106961 May 1 21:38 ssl.log
-rw-rw-r-- 1 nilesh nilesh 3463 May 1 21:38 ssh.log
-rw-rw-r-- 1 nilesh nilesh 1153 May 1 21:38 snmp.log
-rw-rw-r-- 1 nilesh nilesh 253 May 1 21:38 packet_filter.log
-rw-rw-r-- 1 nilesh nilesh 805 May 1 21:38 notice.log
-rw-rw-r-- 1 nilesh nilesh 15317 May 1 21:38 files.log
-rw-rw-r-- 1 nilesh nilesh 8393 May 1 21:38 dns.log
-rw-rw-r-- 1 nilesh nilesh 935 May 1 21:38 dhcp.log
-rw-rw-r-- 1 nilesh nilesh 807130 May 1 21:38 conn.log
nilesh@nilesh-virtual-machine:~/final$ cat notice.log
#separator \x09
#set_separator
#empty_field (empty)
#unset_field -
#path notice
#open 2015-05-01-21-38-18
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p fid file_mime_type file_desc proto note msg sub s
rc dst p n peer_descr actions suppress_for dropped remote_location.country_code remote_location.region remote_location.cityr
emote_location.latitude remote_location.longitude
#types time string addr port string string string enum enum string string addr addr port count string set[e
num] interval bool string string double double
430532826.003010 - - - - - - - - - - - - - - - - - - - - - - Scan::Port_Scan:192.168.67.130 scanned at least 15 unique por
ts of host 192.168.67.131 in 0m0s      remote 192.168.67.130 192.168.67.131 - - - - - bro Notice::ACTION_LOG 3600.000000 F - - -
#close 2015-05-01-21-38-19
nilesh@nilesh-virtual-machine:~/final$
```

Conclusion

Port Scanning means that a system has targeted one system, connecting to one port, then another, until it has scanned a range of ports. Bro labels unusual or exceptional events as weird, they include a wide range of events, including malfunctioned connections, attackers attempts to confuse or evade being detected, malfunctioning hardware, or misconfigured system or network devices such as routers and offcourse attacks like port scans. Some weird events might be an attack or some anomalies. The better we know our system the more likely we are going to be to detect the events to be genuine or an attack.

So as future work of our project we could next implement where weird activities will be detected and an alarm will be raised , care should be taken that genuine events are not mistaken to be attack attempts hence reducing false positives. We could write policies that will differentiate attacks from regular genuine events. Every system is vulnerable to port scanning. Before a system is placed online, port scan should be performed against the system. If a system is compromised, attacker may try to open more ports on the system so they can more easily exploit the port weaknesses.

To secure the unauthorized access on a port, when a request is coming on the port, we can first check whether the host is allowed for the port access. If there is no access, then it cannot send the request to port.

3. Password Security

Ans. 3-A: Describe the format of the file (e.g., userid: hashed password: etc...)

crack01:mrmOpRH5FyJAo:526:531::/home/crack01:/bin/bash

Each field is separated with a ":" colon character, and are as follows:

- **crack01**→Username, up to 8 characters. Case-sensitive, usually all lowercase
- **mrmOpRH5FyJAo**→in the password field. Passwords are stored in the ``/etc/shadow" file.
- **526**→Numeric user id. This is assigned by the ``adduser" script. Unix uses this field, plus the following group field, to identify which files belong to the user.
- **531**→Numeric group id. Red Hat uses group id's in a fairly unique manner for enhanced file security. Usually the group id will match the user id.
- **:** “:**”**→Full name of user. I'm not sure what the maximum length for this field is, but try to keep it reasonable (under 30 characters).
- **home**→User's home directory. Usually /home/username (eg. /home/smithj). All user's personal files, web pages, mail forwarding, etc. will be stored here.
- **:/bin/bash**→User's "shell account". Often set to ``/bin/bash" to provide access to the bash shell (my personal favorite shell).

"crack-these-please"

After installing john-the-ripper and placing the crack-these-please into the run folder of john, ran the command

```
# john crack-these-please
```

1.crack01:bike	14.crack14:hello	27.crack27:stir	40.crack28:tall
2.crack02:bloody	15.crack15:into	28.crack29:test	41.crack47:wwwwww
3.crack03:blue	16.crack16:japan	29.crack30:usa	
4.crack04:bonjour	17.crack17:kaput	30.crack36:hackme	
5.crack05:bread	18.crack18:1337	31.crack37:09112001	
6.crack06:bueno	19.crack19:linux	32.crack38:R2D2	
7.crack07:cowboy	20.crack20:mind	33.crack39:nauj	
8.crack08:ddd	21.crack21:money	34.crack41:w	
9.crack09:dejavu	22.crack22:more	35.crack42:WW	
10.crack10:dog	23.crack23:abcdefgh	36.crack43:www	
11.crack11:perro	24.crack24:pass	37.crack44:www	
12.crack12:fido	25.crack25:really	38.crack45:www	
13.crack13:Clinton	26.crack26:smc	39.crack46:wwwwww	

Below are the screenshots of the same:

```

Command Prompt - john crack-these-please
Loaded 50 password hashes with 50 different salts <Traditional DES [128/128 BS S>
SE21>
monev          <crack21>
cowboy         <crack07>
hello          <crack14>
test           <crack29>
blue            <crack03>
japan           <crack16>
bonjour        <crack04>
dog              <crack10>
pass             <crack24>
www              <crack43>
www             <crack44>
wwwwww         <crack47>
more            <crack22>
ddd              <crack08>
mind            <crack20>
bike            <crack01>
tall             <crack28>
1337            <crack18>
w                <crack19>
bread           <crack05>
servo           <crack11>
smc              <crack26>
stir              <crack22>
bloody          <crack02>
ww              <crack42>
usa              <crack30>
linux            <crack19>
really          <crack25>
fido             <crack12>
into             <crack15>
bueno            <crack06>
kaput            <crack17>
hackme          <crack36>
dejavu          <crack09>
nauj             <crack39>
www             <crack15>
wwwD2           <crack38>
guesses: 37    time: 0:01:29:13 <3> c/s: 929557   trying: lyuz6v - lyzfix
guesses: 39    time: 0:01:31:23 <3> c/s: 930560   trying: 0312it - 031?so
guesses: 37    time: 0:01:32:52 <3> c/s: 931502   trying: gom4318 - gon4301
wwwwww         <crack46>
guesses: 38    time: 0:01:49:44 <3> c/s: 908352   trying: siXd8e - siff6w
guesses: 38    time: 0:03:23:39 <3> c/s: 566460   trying: 342nge99 - 342ngalt
guesses: 38    time: 0:03:23:51 <3> c/s: 566675   trying: 308sepue - 308sei5#
guesses: 38    time: 0:03:27:28 <3> c/s: 572168   trying: stdard? - stdarol
guesses: 38    time: 0:03:30:46 <3> c/s: 577345   trying: $Asra* - $Asrpl
guesses: 38    time: 0:03:40:58 <3> c/s: 594742   trying: liamilk - liamsy
guesses: 38    time: 0:03:45:44 <3> c/s: 602697   trying: 17#aksdf - 17#akth3
guesses: 38    time: 0:03:49:19 <3> c/s: 608672   trying: dixs7d - dixitt
guesses: 38    time: 0:03:51:39 <3> c/s: 612460   trying: sllibdiv8 - sllibdoie
guesses: 38    time: 0:04:06:26 <3> c/s: 634679   trying: gnskugh - gnsrtls
guesses: 38    time: 0:04:08:29 <3> c/s: 664428   trying: gizeaze - gizeekm
guesses: 38    time: 0:04:31:51 <3> c/s: 668351   trying: f186766 - f1867nt
abcdeFgh       <crack23>
guesses: 39    time: 0:15:43:05 <3> c/s: 294389   trying: hEsinO - hEsit3

```

```

Command Prompt - john crack-these-please
guesses: 38    time: 0:03:27:28 <3> c/s: 572168   trying: stdard? - stdarol
guesses: 38    time: 0:03:30:46 <3> c/s: 577345   trying: $Asra* - $Asrpl
guesses: 38    time: 0:03:40:58 <3> c/s: 594742   trying: liamilk - liamsy
guesses: 38    time: 0:03:45:44 <3> c/s: 602697   trying: 17#aksdf - 17#akth3
guesses: 38    time: 0:03:49:19 <3> c/s: 608672   trying: dixs7d - dixitt
guesses: 38    time: 0:03:51:39 <3> c/s: 612460   trying: sllibdiv8 - sllibdoie
guesses: 38    time: 0:04:06:26 <3> c/s: 634679   trying: gnskugh - gnsrtls
guesses: 38    time: 0:04:30:29 <3> c/s: 664428   trying: gizeaze - gizeekm
guesses: 38    time: 0:04:33:51 <3> c/s: 668351   trying: f186766 - f1867nt
abcdeFgh       <crack23>
guesses: 39    time: 0:15:43:05 <3> c/s: 294389   trying: hEsinO - hEsit3
guesses: 39    time: 0:16:19:38 <3> c/s: 311328   trying: RjnB0o - RjnB0q
guesses: 39    time: 0:16:48:11 <3> c/s: 329557   trying: mettati - mettfek
guesses: 39    time: 0:17:00:36 <3> c/s: 335423   trying: amodrif - amodrive
guesses: 39    time: 0:17:46:29 <3> c/s: 323384   trying: asiyans - asiyash
guesses: 39    time: 0:17:58:12 <3> c/s: 329934   trying: 18ki84be - 18kiYOK9
guesses: 39    time: 0:18:08:23 <3> c/s: 335669   trying: ccoxjuft - ccoxjlde
guesses: 39    time: 0:18:08:24 <3> c/s: 335680   trying: ccrogog?? - ccrgoii5
guesses: 39    time: 0:18:08:26 <3> c/s: 335686   trying: ccipitro - ccipicto
guesses: 39    time: 0:18:15:50 <3> c/s: 339742   trying: 05tepey - 05tepix
guesses: 39    time: 0:18:15:51 <3> c/s: 339753   trying: 0b0nig0 - 0b0n138
guesses: 39    time: 0:18:15:52 <3> c/s: 339763   trying: 04tu24 - 04ttudy
guesses: 39    time: 0:18:15:53 <3> c/s: 339774   trying: 04tnlia - 04tnma5
guesses: 39    time: 0:18:15:54 <3> c/s: 339785   trying: 09titm2 - 09t156y
guesses: 39    time: 0:18:15:56 <3> c/s: 339790   trying: 0aatc03 - 0aatow3
guesses: 39    time: 0:18:24:37 <3> c/s: 344483   trying: fd17t1 - fdbLOW
guesses: 39    time: 0:18:51:12 <3> c/s: 352467   trying: 33139ay5 - 331375ed
guesses: 39    time: 0:18:53:44 <3> c/s: 360929   trying: 51572tin - 51572t27
09112001       <crack37>
Clinton        <crack13>
guesses: 41    time: 0:21:06:28 <3> c/s: 416558   trying: J0ddzd1 - J0dxsu

```

=====
Ans. 3-B:

The previous file was an old password file as it used TripleDES for password hashing, a broken scheme that does not provide good security. These hashes are created with sha512crypt, which is a password-based hashing tool, where \$6\$ is the type of password hash (SHA512 many rounds), the next string between \$ \$ is the salt, and the remaining string is the hash. Use John The Ripper (or Hashcat) to break these four passwords.

- \$6\$NShHCRTL\$1Ae9dI1rtpAXQkiMPqncpCQ69gE7Y25TgKRDvtfIOdLVTIG4cMAp9LQE9eEZuboS4tO6ippBnOIFE8zgqOvGP0-->**soccer**

```
Command Prompt
Input.Mode: Dict <D:\wordlists\Abbreviations.dic>
Index....: 1/1 <segment>, 483815 <words>, 4079187 <bytes>
Recovered.: 0/1 hashes, 0/1 salts
Speed/sec.: 51 plains, 51 words
Progress...: 308436/483815 <63.75%>
Running...: 00:01:40:39
Estimated.: 00:00:57:18

$6$NShHCRTL$1Ae9dI1rtpAXQkiMPqncpCQ69gE7Y25TgKRDvtfIOdLVTIG4cMAp9LQE9eEZuboS4tO6
ippBnOIFE8zgqOvGP0:soccer

All hashes have been recovered

Input.Mode: Dict <D:\wordlists\Abbreviations.dic>
Index....: 1/1 <segment>, 483815 <words>, 4079187 <bytes>
Recovered.: 1/1 hashes, 1/1 salts
Speed/sec.: - plains, 50 words
Progress...: 313464/483815 <64.79%>
Running...: 00:01:42:44
Estimated.: 00:00:56:47

Started: Sun Apr 19 11:40:57 2015
Stopped: Sun Apr 19 13:34:11 2015

C:\hashcat>=
```

- \$6\$ssMb25ys\$yuyoQKJaaGeRVhwsklDAvWnJLcgZxiTX7mrXH.8xCslnGcCbB3S0gLic3qlOGWCZImFI3KW29p1Ht7ny9Jwo-->**joshua**
- \$6\$sH2VWpHm\$cEvtk3IffFilT73amGGv7/6j2LRWHQ7df4vjgoSuOSEt8QZDeDDYxCqlly.cU8/AfL/uLYmX/42QI.etA8fdV1-->**wizard**

```

C:\hashcat>hashcat -m 1800 C:\hashcat\hash1.txt C:\hashcat\wordlists\ C:\hashcat\cracked.txt
Initializing hashcat v0.49 with 2 threads and 32mb segment-size...
Added hashes from file C:\hashcat\hash1.txt: 2 <2 salts>
NOTE: press enter for status-screen
$6$ssMb25ys$yuvoQKJaaGeRUhwsk1DAvWnJLcgZxiTX7mxH.8xCsInGcCbB3S0gLic3q1yOGWCZimF
13Kw29p1Ht?ny9Jwo/:joshua

Input.Mode: Dict <C:\hashcat\wordlists\AT_T.dic>
Index....: 1/1 <segment>, 64571 <words>, 657999 <bytes>
Recovered.: 1/2 hashes, 1/2 salts
Speed/sec.: 150 plains, 150 words
Progress..: 19508/64571 <30.21%>
Running...: 00:00:02:09
Estimated.: 00:00:05:00

$6$ssH2UWpHm$cEvtk3IffFilT73amGGu7/6.j2LRWHQ7df4vjgoSu0SEt8QZDeDDYxCq1ly.cU8/afL/u
1YmX/42Ql.etA8fdU1:wizard

All hashes have been recovered

Input.Mode: Dict <C:\hashcat\wordlists\AT_T.dic>
Index....: 1/1 <segment>, 64571 <words>, 657999 <bytes>
Recovered.: 2/2 hashes, 2/2 salts
Speed/sec.: - plains, 67 words
Progress..: 64572/64571 <100.00%>
Running...: 00:00:16:03
Estimated.: > 10 Years

Started: Thu Apr 23 19:31:52 2015
Stopped: Thu Apr 23 19:47:55 2015
C:\hashcat>_

```

- \$6\$E5s/79nO\$HLNy0xElppb7Dx4537KCsaIAER.wULMLLS1vzgmkVyp1ZK/fK/.td819Ea1RFhMBLfsQ
XvFM0HfMW3k3oF4ob-->**phantom**

```

C:\ Command Prompt - hashcat -m 1800 hash.txt D:\wordlists\ C:\hashcat\cracked... - 
Estimated.: 00:09:26:45

Input.Mode: Dict <D:\wordlists\Abbreviations.dic>
Index....: 1/1 <segment>, 483815 <words>, 4079187 <bytes>
Recovered.: 0/3 hashes, 0/3 salts
Speed/sec.: 28 plains, 9 words
Progress..: 179564/483815 <37.11%>
Running...: 00:05:20:03
Estimated.: 00:09:23:25

$6$E5s/79nO$HLNy0xElppb7Dx4537KCsaIAER.wULMLLS1vzgmkVyp1ZK/fK/.td819Ea1RFhMBLfsQ
XvFM0HfMW3k3oF4ob.:phantom

Input.Mode: Dict <D:\wordlists\Abbreviations.dic>
Index....: 1/1 <segment>, 483815 <words>, 4079187 <bytes>
Recovered.: 1/3 hashes, 1/3 salts
Speed/sec.: 17 plains, 8 words
Progress..: 189152/483815 <39.10%>
Running...: 00:05:54:12
Estimated.: 00:10:13:52

```

Ans. 3-C:

Repeating these instructions, add 10 more users with 10 different passwords of your choosing and copy the last 10 lines of the shadow file to crack1.hash. You can use dictionary words, only letters, only symbols, four-character passwords, 10-character passwords----you don't need to crack them all, just leave them running overnight as a maximum---and summarize your impressions on password cracking.

Created 10 users as listed below with their respective passwords:

Username	Password
alice	alice
bob	bob123
dravid	password
mario	Super1390
king	kruskal
reena	\$\$reena12
tony	\$bat#1#
rajesh	!123
kapil	20dev&&
sony	@@sony.!@

```
Terminal
[04/23/2015 18:11] seed@ubuntu:~$ sudo adduser alice
[sudo] password for seed:
Adding user `alice' ...
Adding new group `alice' (1012) ...
Adding new user `alice' (1011) with group `alice' ...
Creating home directory `/home/alice' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for alice
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
[04/23/2015 18:12] seed@ubuntu:~$ sudo adduser bob
Adding user `bob' ...
Adding new group `bob' (1013) ...
Adding new user `bob' (1012) with group `bob' ...
Creating home directory `/home/bob' ...
Copying files from `/etc/skel' ...
```

```
alice:x:1011:1012:,:/home/alice:/bin/bash
bob:x:1012:1013:,:/home/bob:/bin/bash
dravid:x:1013:1014:,:/home/dravid:/bin/bash
mario:x:1014:1015:,:/home/mario:/bin/bash
king:x:1015:1016:,:/home/king:/bin/bash
reena:x:1016:1017:,:/home/reena:/bin/bash
tony:x:1017:1018:,:/home/tony:/bin/bash
rajesh:x:1018:1019:,:/home/rajesh:/bin/bash
kapil:x:1019:1020:,:/home/kapil:/bin/bash
sony:x:1020:1021:,:/home/sony:/bin/bash
[04/23/2015 18:20] seed@ubuntu:~$
```



After copying the last 10 line of the /etc/shadow file into crack2.hash, started john for password cracking.

Left it running overnight and in this span of time the only pass words it could crack were as shown in the screenshot below

Username	Password
alice	alice
bob	bob123
dravid	password

```
[04/27/2015 22:04] seed@ubuntu:~$ john crack2.hash
Loaded 10 password hashes with 10 different salts (generic crypt(3) [?/32])
alice      (alice)
bob        (bob)
password   (dravid)
guesses: 3  time: 0:02:07:08 27% (2)  c/s: 39.74  trying: olivier5 - skibum5
```

So the observation we have made is few passwords couldn't be decoded may be because they are protected by a strong password like a password with a combination of letters(Uppercase & Lowercase), digits(0,1...9) also with special characters. Such kind of passwords become very difficult to decode even using tools like john.

And on the other hand we could decode few passwords since these passwords are easy to decrypt as we can see it is very easy for john to guess these passwords as it is a commonly occurring word in all common dictionaries and can be easily guessed by dictionary based attacks. And it is not strong enough to resist hacking. This password would have been stronger if it had a mix of digits, letters and special characters which could have made it very difficult for john to guess the password.