Introduction to Passive Network Traffic Monitoring

CS459 ~ Internet Measurements Spring 2015

Despoina Antonakaki antonakd@csd.uoc.gr
Active Monitoring

- Inject test packets into the network or send packets to servers and applications & … measure the response

- Unfortunately… extra generated traffic
  - influences the measurement

- Induced traffic can be controlled/adjustable

- Explicit control on generation of packets, nature of traffic generation, sampling techniques, timing, frequency, packet sizes, path & function to be monitored
  - You can control what you want, when you need it
Active Monitoring

- Can measure:
  - Delay/loss
  - Topology/routing behavior
  - Bandwidth/throughput

- Internet Control Message Protocol (ICMP) message used for diagnostic or control purposes or generated in response to errors in IP operations.
  - TTL field: ping & traceroute use it in attempt to reach a given host computer or to trace a route to that host.
  - Traceroute intentionally sends a packet with a low TTL value so that it will be discarded by each successive router in the destination path.
  - The time between sending the packet and receiving back the ICMP message that it was discarded is used to calculate each successive hop travel time
Passive Monitoring

• Watch passing by traffic. How?

• Special purposed device on the network to capture packets for analysis:
  • Sniffer (software/hardware), OCxMon, build on routers/switches/end node hosts.

• Build capabilities on switches or other network devices:
  • Remote Monitoring (RMON),
  • Simple Network Monitoring Protocol (SNMP)
  • Netflow capable devices

• Collect information periodically
Passive Monitoring

- No extra traffic induced
- Measures real traffic
- But generated traffic can be enormous, hard for analysis
- The passive approach extremely valuable in network trouble-shooting, however limited in ability to emulate error scenarios or isolating exact fault location
- May require viewing all packets on the network...raises privacy/security issues
Passive Monitoring

- Different ways to perform passive network monitoring:

- Capturing done by **packet capture filters** (tcpdump)
  - Requires access to wire
  - **promiscuous** mode network ports to see other traffic
    - pass ALL traffic it receives to CPU rather passing only frames the controller is intended to receive

- flow-level, packet-level data on routers
  - Simple Network Management Protocol (SNMP). Three key components:
    - managed devices,
    - agents,
    - network-management systems (NMSs).
      - A managed device is a node that has an SNMP agent and resides on a managed network. These devices can be routers and access servers, switches and bridges, hubs, computer hosts, or printers.
      - An agent is a software module residing within a device. This agent translates information into a compatible format with SNMP.
      - An NMS runs monitoring applications. They provide the bulk of processing and memory resources required for network.

- Cisco NetFlow
Passive Monitoring

Different ways to perform passive network monitoring:

- **Ad hoc packet analysis**: capturing packets & analyzing:
  - file manipulation tools (deep understanding of packet structure/network protocols, grep, sort, uniq)
  - a protocol analyzer: WireShark (supports both live/offline analysis, a GUI & support for analyzing multiple protocols)

- **Simple Network Management Protocol (SNMP)-based** analysis. A simple protocol helpful for monitoring conditions at the device level & helps have info about device interactions, especially the flow of traffic between devices. Three key components:
  - **Managed devices**: a node that has an SNMP agent and resides on a managed network. These devices can be routers and access servers, switches and bridges, hubs, computer hosts, or printers
  - **Agents**: software module residing within a device that translates information into a compatible format with SNMP
  - **Network-management systems** (NMSs): runs monitoring applications & provide the bulk of processing and memory resources required for network.
Passive Monitoring

- Netflow-based analysis: protocol for collecting information at a flow of traffic level
  - Packets with same src & dst, ports, IP protocol and type of service considered part of same network flow.
  - Developed by Cisco
  - Allows you identify apps & services:
    - consuming inordinate amounts of bandwidth
    - should not be on the network, such as malicious software or application
  - Capturing done by packet capture filters (tcpdump)
    - Requires access to wire
    - promiscuous mode network ports to see other traffic
      - pass ALL traffic it receives to CPU rather passing only frames the controller is intended to receive
Passive Monitoring tools

- Tcpdump & libpcap (limited protocol decoding but available on most *NIX platforms)
  - [http://www.tcpdump.org/](http://www.tcpdump.org/)

- Wireshark (powerful sniffer which can decode lots of protocols, lots of filters)
  - [http://www.wireshark.org/](http://www.wireshark.org/)

- Tshark (command line version of wireshark)
  - dumpcap (part of wireshark) - can only capture traffic and can be used by wireshark / tshark

- ssldump (an SSLv3/TLS network protocol analyzer-decrypts HTTPS)

All tools use libpcap (on windows winpcap) for sniffing. Wireshark / tshark / dumpcap can use tcpdump filter syntax as capture filter.
Passive Monitoring tools

- Microsoft Network Monitor

- Tcpflow
  - [https://github.com/simsong/tcpflow](https://github.com/simsong/tcpflow)

- dSniff
  - [http://www.monkey.org/~dugsong/dsniff/](http://www.monkey.org/~dugsong/dsniff/)

- Tcptrace
  - [http://www.tcptrace.org/](http://www.tcptrace.org/)
tcpdump

tcpdump is the premier network analysis tool for information security professionals.

- Syntax:

  tcpdump [options] [filter expression]

- Filters:
  - We are often not interested in all packets flowing through the network
  - Use bpf filters to capture only packets of interest to us
Berkeley Packet Filter

- On some Unix provides a raw interface to data link layers, allowing raw link-layer packets to be sent and received.
  - If promiscuous mode available:
    - all packets (&destined to other hosts) can be received

- Supports filters on packets (keep only interesting packets)

- Filters implemented as interpreter for a machine language for the BPF virtual machine:
  - Program gets data from packets, performs arithmetic operations and compare results with constants or against data in the packet or test bits in the results, accepting or rejecting packet.

Berkeley Packet Filter

- So... Allows a user-space program to attach a filter onto any socket and allow or disallow certain types of data to come through the socket.

- BPF is sometimes used to refer just to the filtering mechanism, rather than to the entire interface.

- The human readable filter is converted to a bytecode send to kernel -> efficient!

Demo

- Capture only udp packets
  - tcpdump “udp”

- Capture only tcp packets
  - tcpdump “tcp”
Demo

- Capture only UDP packets with destination port 53 (DNS requests)
  - tcpdump "udp dst port 53"

- Capture only UDP packets with source port 53 (DNS replies)
  - tcpdump "udp src port 53"

- Capture only UDP packets with source or destination port 53 (DNS requests and replies)
  - tcpdump "udp port 53"
tcpdump output

- Different output formats for different packet types

This is an IP packet

Source port number (22)
The standard TCP port 22 has been assigned for contacting SSH servers

Destination host name

Source host name

01:46:28.808262 IP danjo.CS.Berkeley.EDU.ssh >


Destination port number

TCP specific information
Useful tcpdump commands

- See the list of interfaces on which tcpdump can listen:
  - tcpdump -D

- Listen on interface eth0:
  - tcpdump -i eth0

- Be verbose while capturing packets:
  - tcpdump -v

- Limit the capture to 100 packets:
  - tcpdump -c 100

- Record the packet capture to a file called capture.cap:
  - tcpdump -w capture.cap

- Display the packets of a file called capture.cap:
  - tcpdump -r capture.cap

- Display IP addresses and port numbers instead of domain and service names when capturing packets:
  - tcpdump -n

- Capture any packets where the destination host is 192.168.1.1. Display IP addresses and port numbers:
  - tcpdump -n dst host 192.168.1.1

- Capture any packets where the source host is 192.168.1.1. Display IP addresses and port numbers:
  - tcpdump -n src host 192.168.1.1

- Capture any packets where the source or destination host is 192.168.1.1:
  - tcpdump host 192.168.1.1
Useful tcpdump commands

- Capture any packets where the destination network is 192.168.1.0/24:
  - `tcpdump dst net 192.168.1.0/24`

- Capture any packets where the source or destination network is 192.168.1.0/24:
  - `tcpdump net 192.168.1.0/24`

- Capture any packets where the destination port is 23:
  - `tcpdump dst port 23`

- Capture any packets where the destination port is between 1 and 1023 inclusive:
  - `tcpdump dst portrange 1-1023`

- Capture only TCP packets where the destination port is between 1 and 1023 inclusive:
  - `tcpdump tcp dst portrange 1-1023`

- Capture only UDP packets where the destination port is between 1 and 1023 inclusive:
  - `tcpdump udp dst portrange 1-1023`

- Capture any packets with destination IP 192.168.1.1 and destination port 23:
  - `tcpdump "dst host 192.168.1.1 and dst port 23"`

- Capture any packets with destination IP 192.168.1.1 and destination port 80 or 443:
  - `tcpdump "dst host 192.168.1.1 and (dst port 80 or dst port 443)"`

- Capture any ICMP packets:
  - `tcpdump -v icmp`

See more at: [http://www.tcpdump.org/manpages/tcpdump.1.html](http://www.tcpdump.org/manpages/tcpdump.1.html)
Online Packet Capturing example

• Start capturing traffic (needs sudo):

login milo csd machine
  • $: tcpdump –D (find the appropriate interface e.g. eth0)
  • $: tcpdump –n –i eth0 –w onlineTest.pcap

• Produce traffic:
  • Read your email
  • Watch a YouTube video for 5 minutes
  • Download torrent of an official Linux distribution
Online Packet Capturing example

Stop packet capture and report:

- **Average packet size:**

- **get ip packets:**
  
  $ tcpdump –q –n –r onlineTest.pcap ip > onlineTest.out

- **search for ‘length’ string, get size, sum bytes, print sum/n:**

  $: cat onlineTest.out \\
  | awk -F'length' '{print $2}' \\
  | awk '{ sum += $1; n++ }' \\
  | awk END { if (n > 0) print sum / n; }'
Online Packet Capturing example

Number of packets per destination port:

1. Get the destination port for each packet
   $: cat onlineTest.out \
   | awk '{print $5}' \
   | awk -F'.' '{print $5}' \
   | awk -F':' '{print $1}' > destPorts

2. Remove blank lines if any

   $: sed '/^$/d' destPorts > destPorts.log

3. Sort ports and measure the occurrences of each port

   $: sort -n destPorts | uniq -c > numbOfpackets

4. Plot numbOfpackets
Online Packet Capturing example

- Plot CDF of packet sizes:
  - Before you make cdf plot you should prepare your data:

```bash
$: cat onlineTest.out \  
| awk -F'length' '{print $2}' \  
| awk -F ':' '{print $1}' \  
| sort -n | uniq -c
```
MAWI traces

• The MAWI Working Group is a joint effort of Japanese network research and academic institutions to study the performance of networks and networking protocols in Japanese wide area networks.

• MAWI is an acronym for the Measurement and Analysis of Wide-area Internet.

See more at: http://mawi.wide.ad.jp/mawi/
Offline packet capturing example

Download a trace (e.g. trace of 1/3/2014):

- $\text{wget http://mawi.nezu.wide.ad.jp/mawi/samplepoint-F/2014/201403011400.dump.gz}$

- Analyze trace with tcpdump:
  1. Number of packets in trace:

    $\text{tcpdump -n -r 201301071400.dump | wc -l}$
    gives 100,933,314 packets
Offline packet capturing example

2. CDF of packet sizes, show the 95th and 99th percentile:
   - A percentile (or a centile) is a measure used in statistics indicating the **value below which a given percentage of observations** in a group of observations fall.
     e.g. the 20th percentile is the value (or score) below which 20 percent of the observations may be found.

   - Get the data from trace and measure the occurrences:
     ```bash
     cat 201301071400.out | awk -F'length' '{print $2}'
     | awk -F ':' '{print $1}'
     | sort -n
     | uniq -c
     ```

     How this work in awk:
     Line from trace file looks like this:
     07:00:01.351778 IP 141.83.1.224.21011 > 128.33.137.252.64071: UDP, length 1423
     `awk -F'length' '{print $2}'` sets length as a delimiter and returns 1423
Offline packet capturing example

```
set terminal postscript enhanced eps defaultplex
blacktext\dashed dashlength 2.0 linewidth 3.0 butt\palfuncparam 2000,0.003 "Helvetica" 24
set out "cdf.eps" set grid y lt 0 lw 1 lc rgb "#B8B8B8"
set auto x
set nokey
set xlabel "Packet size"
set ylabel "CDF"
set style data lines
set boxwidth 0.9

perc95=1441
perc99=1499
set arrow from perc95,0 to perc95,0.95 nohead lt 1 lc rgb "purple"
set arrow from graph(0,0),0.95 to perc95,0.95 nohead lt 1 lc rgb "purple"
set arrow from perc99,0 to perc99,0.99 nohead lt 1 lc rgb "purple"
set arrow from graph(0,0),0.99 to perc99,0.99 nohead lt 1 lc rgb "purple"
plot 'datacdf.log'
```

Draw percentiles
Offline packet capturing example
Offline packet capturing example

3. Traffic percentage per destination port

- Get number of packets per destination port

```
cat 201301071400.out |
   | awk '{print $5}' |
   | awk -F'.' '{print $5}' |
   | awk -F':' '{print $1}' |
   | sort -n |
   | uniq -c > packets
```

- Divide each port’s number of packets by the total packets that you calculate in (1) e.g.

```
cat packets | awk '{print $1/<totalnum>" "$2}'
```
4. All traffic to/from a given IP address per minute (let’s say 201.57.174.30)

- Get delta (micro-second resolution) between current and first line on each dump line for a specific host:

```
$: tcpdump -q -tttt -nr 201301071400.dump host 201.57.174.30 > specIP.out
```

Why is it zero???
- length 0 this is the length of the packet
- Because we sent just a SYN packet, and a SYN packet contain only the header of a TCP packet and doesn’t contain any data.
Offline packet capturing example

5. Get all UDP conversations of the trace:

$ tcpdump -n -w udpConvers.pcap -r 201301071400.dump udp