Advanced Computer Networking

CYBR 230 – Jeff Shafer – University of the Pacific

Project 2

Schedule

This Week

- Mon September 18
  - *Project 1 Work*
- Wed September 20
  - *Project 1 Testing (Grading)*
- Fri September 22
  - *Start Project 2*

Next Week

- Mon September 25
  - *Project 2 Work*
- Wed September 27
  - *TBD*
- Fri September 29
  - *TBD*

Project 2

Design Goals

- *Nothing shall escape our view!*
  - Monitoring, Monitoring, Monitoring!
- Part 1: Capture
  - Full packet capture
  - NetFlow capture
  - Protocol-specific capture
- Part 2: Analysis
  - Data indexing, searching, and analysis tools

Part 1 - Capture

Capture *All of the Things*!

Advantages

- 100% visibility
  (of non-encrypted traffic)
- Able to re-create historic events with full fidelity
- Happy feeling that you aren't missing anything ☺

Disadvantages

- **Massive** data sets
  - Challenging to store
  - Challenging to search
  - Challenging to analyze
- Google calculator trick:
  "50 megabit per second to gigabytes per day"
  - *540GB/day*

Design Tradeoffs

- **How much data do I capture?**
  - Do I capture all packet data?
    - Contains all information possible, but comes with privacy considerations, massive storage requirements for busy networks, and slow analysis and processing times
  - Do I capture only flow summary data?
    - Smaller data sets are easier to process, but you won't be able to reconstruct all details of network events

Design Tradeoffs

- **How long do I retain data for?**
  - Does this answer vary depending on the data type?
- **What are the *exact* ports and links I am monitoring?**
  - At the public-facing router? Which port? At key internal switches? Which ports?
  - Depending on the choices made, key network events may either be visible or invisible.
- **Where is the captured data being stored?**
  - How is this storage secured?

Design Tradeoffs

Prior to implementation, you need to:

(a) Propose a design that answers these questions
(b) Discuss the design with me

Deadline? Let's discuss at end of class

Capture Requirements

- Full packet capture
  - `.pcap` or `.pcapng` files archived on server
- NetFlow capture
  - Archived on server
- DNS Monitoring
  - PassiveDNS tool

Capture Requirements

- "Nice to have" list
  - ARPWatch tool (on the bridges?)
  - Log files from Mikrotik (firewall and DHCP server)
  - Wireless monitoring

Full Packet Capture

- Packet capture file types: `.pcap`, `.pcapng`
  - **PCAP** – Classic format
    - Broad compatibility
  - **PCAP<u>NG</u>** – "Next Generation"
    - Capture from multiple interfaces
    - Higher timestamp resolution
    - Metadata on capture source
    - Application compatibility is work in progress
      - Default file format in Wireshark/tshark now!
    - https://pcapng.github.io/pcapng/

Full Packet Capture

- Common applications and libraries
  - **Tcpdump** - http://www.tcpdump.org/
  - **Wireshark** (GUI) and **tshark** (command-line) and **dumpcap** (cmd, capture-only) - https://www.wireshark.org/
- Common libraries
  - **Libpcap** - https://github.com/the-tcpdump-group/libpcap
  - **WinPcap** - https://www.winpcap.org/

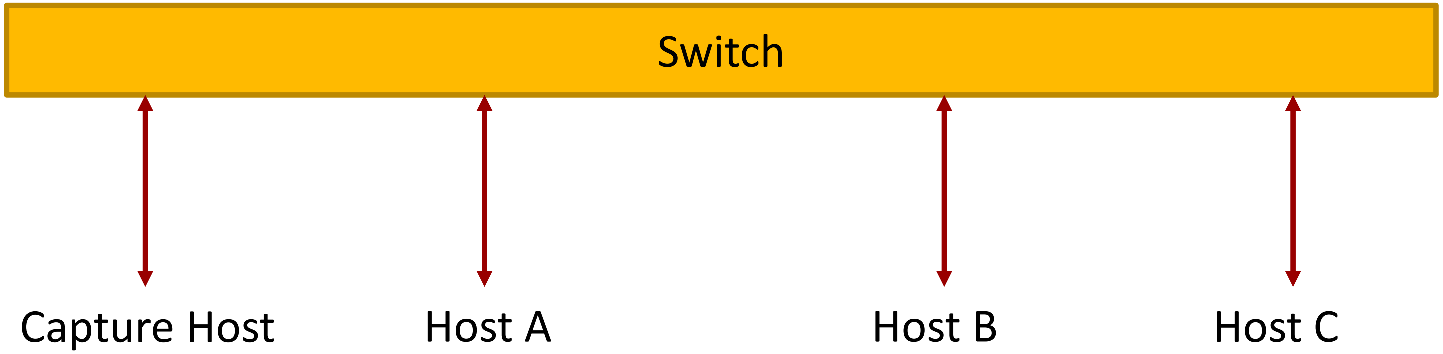
Full Packet Capture

```
sudo tcpdump -i <interface> -s 65535 -w <some-file>
```

- `-i`: Interface
- `-s`: Max packet size (tradeoff size vs fidelity)
- `-w`: Output file name
- *Tip: Look at the `-W`, `-C`, and `-G` options if you want a set of files that rotates based on date or size, rather than one infinitely-growing file*

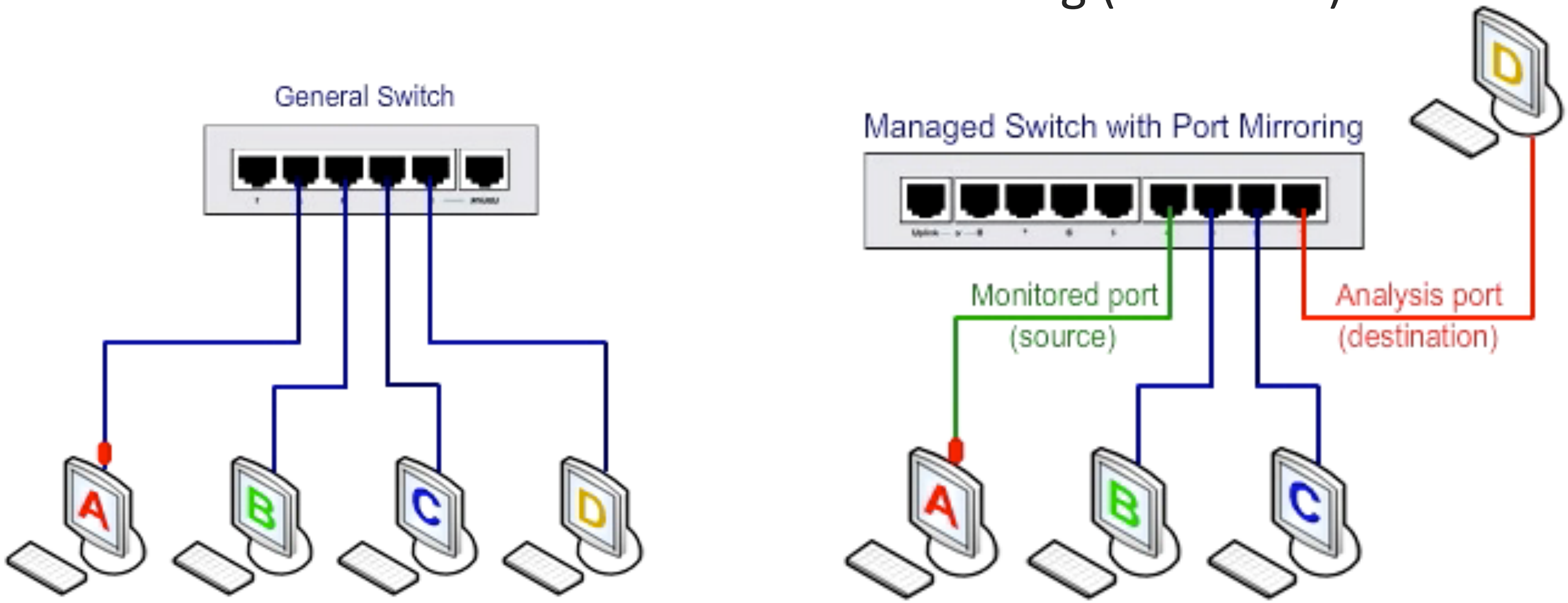
Full Packet Capture

- **Design Problem:** Does this NIC actually *see* the data we want to capture?
  - Broadcasts (ARP, DHCP) are sent out all switch ports
  - Other traffic is point-to-point!
  - And what about other switches?



Full Packet Capture

- Hardware switch: Port mirroring (aka SPAN)



https://www.miarec.com/faq/what-is-port-mirroring

Full Packet Capture

- Problem: Our CCR **router** lacks the hardware **switch chip** that (in Mikrotik-land) provides hardware port mirroring functionality ☹
- Plan B.1 – Add switches and configure mirroring?
  - The HP switch is for student+Q subnet
  - Use the Cisco switch for instructor subnet
- Plan B.2 – Software?
  - Tazmen Sniffer Protocol (TZSP) stream

Full Packet Capture

On Router:

```
tool sniffer set streaming-enabled=yes streaming-
server=a.b.c.d filter-stream=yes filter-interface=bridgeXYZ

tool sniffer start
tool sniffer print
tool sniffer stop
```

On Capture Device:
use tshark capture tool and filter on ‘tzsp’ protocol

```
tshark -i extif -f "udp port 37008" -n -d udp.port==37008,tzsp
```

Full Packet Capture

- Pending issues **(for you to solve!)**
  - Will tshark unwrap the TZSP packets and save only the inner data into the `.pcap[ng]` file? (required)
  - Will Linux complain about all these UDP packets arriving at a port that is closed, and send back ICMP Destination Unreachable messages?
    - Do we need a second IP address / fake NIC that tshark captures on, but Linux is not running a network stack?
  - CPU overhead of this method vs raw tcpdump?
- Other option besides tshark
  - https://github.com/thefloweringash/tzsp2pcap

NetFlow

- Provides *summary* of connections (streams of packets)
- Different product names from different vendors
  - Cisco: **NetFlow** (original developer)
  - HP: “NetStream”
  - Juniper: “Jflow”
  - MikroTik: “Traffic Flow”
- Common versions of NetFlow:
  - **v5** (very common, but IPv4 only)
  - **v9** (adds support for IPv6, MPLS)

NetFlow

- **Q: What is a flow?**
- **A:** Packets that share the same attributes
  - Ingress interface
  - Source IP address
  - Destination IP address
  - Source port (TCP/UDP)
  - Destination port (TCP/UDP)
  - IP Protocol
  - IP Type of Service

NetFlow

- What data is captured on a per-flow basis?
  - *Ingress interface*
  - *Source IP address*
  - *Destination IP address*
  - *Source port (TCP/UDP)*
  - *Destination port (TCP/UDP)*
  - *IP Protocol*
  - *IP Type of Service*
  - Start time
  - Stop time
  - Total number of packets
  - Total number of bytes
- **The payload is <u>NOT</u> captured**

NetFlow Probe (on Mikrotik)

```
ip traffic-flow set interfaces=etherX,etherY,etherZ
ip traffic-flow set cache-entries=16k
ip traffic-flow target add dst-address=a.b.c.d port=9995
version=9
ip traffic-flow set enabled=yes

ip traffic-flow print
ip traffic-flow target print detail
ip traffic-flow monitor
```

NetFlow Capture

```
nfcapd -p 9995 -b a.b.c.d -l /path/to/storage
```

- `-p` : Port (9995)
- `-b` : Bind to specific IP (should be YOUR IP, the IP that the probe is sending flow data to)
- `-l` : Base directory to store output files
- http://nfdump.sourceforge.net/
- Tip: Look at –w and –S options for file rotation, automatic directory structure creation, etc…

DNS Monitoring

- **PassiveDNS** tool can sniff DNS queries and aggregate results for analysis
  - https://github.com/gamelinux/passivedns
- `passivedns -h` to see help menu
- Use the `-i` option to listen on an interface (live capture) **or** the `-r` option to parse a PCAP file (post-capture analysis)
  - Post-capture analysis probably better for project

Part 1

- **Division of Labor for Part 1?**
- **Deadline for Part 1?**
  - How much data do I capture?
  - How long do I retain data for?
  - What are the *exact* ports and links I am monitoring?
  - Where is the captured data being stored?

Part 2 - Analysis

Analysis

- *Coming Soon!*