

Flow Analysis and Network Hunting

Ben Actis & Michael McFail
netflowanalysis@gmail.com

Creative Commons License

- <http://creativecommons.org/licenses/by-sa/3.0/>
- All materials are licensed under a Creative Commons “Share Alike” license

You are free:



to Share — to copy, distribute and transmit the work



to Remix — to adapt the work

Under the following conditions:



Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

Additional Content/Ideas/Info Provided By

- David Wilburn: initial topic outline
- Willie Kupersanin: content review

About this class

- The intent of this class is to expose you to ...
 - Netflow data
 - Netflow tools
 - Labs with real live netflow found in the wild
 - Analytic tradecraft
 - Situational awareness analytics
 - Hunting analytics

Outline

- **Introduction**
 - What is Netflow?
 - Sensor Location
 - Sampling
- Tools
 - YAF
 - SiLK
 - iSiLK
 - Argus
 - Bro
- Analytics
 - Situational Awareness Analytics
 - Hunting Analytics
 - Data Fusion Analytics
- Wrap Up

Pcap Recap: IPv4

Bit offset	0-3	4-7	8-13	14-15	16-18	19-31
0	Version	Internet Header Length	Differentiated Services Code Point	Explicit Congestion Notification	Total Length	
32	Identification				Flags	Fragment Offset
64	Time to Live		Protocol		Header Checksum	
96	Source IP Address					
128	Destination IP address					
160	Options					
160 or 192+	Data					

Pcap Recap: TCP

Octet	0								1							2							3									
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port															Destination port																
32	Sequence number																															
64	Acknowledgment number (if ACK is set)																															
96	Data offset	Reserved					N S	C W R	E C R	U R G	A C K	P C H	R S H	S S T	Y N	I N	Window size															
128	Checksum															Urgent pointer (if URG is set)																
160	Options																															
...																																

Adapted from:

http://en.wikipedia.org/wiki/Transmission_Control_Protocol#TCP_segment_structure

Pcap Recap: UDP

Bit Offset	0-15	16-31
0	Source port	Destination port
32	Length	Checksum
64	Data	
...		

What is netflow?

- Feeling bloated and fatigued carrying around DVDs of pcap data...then netflow is for you 😊
- 80 GB of PCAP can be converted to 300 MB of netflow data



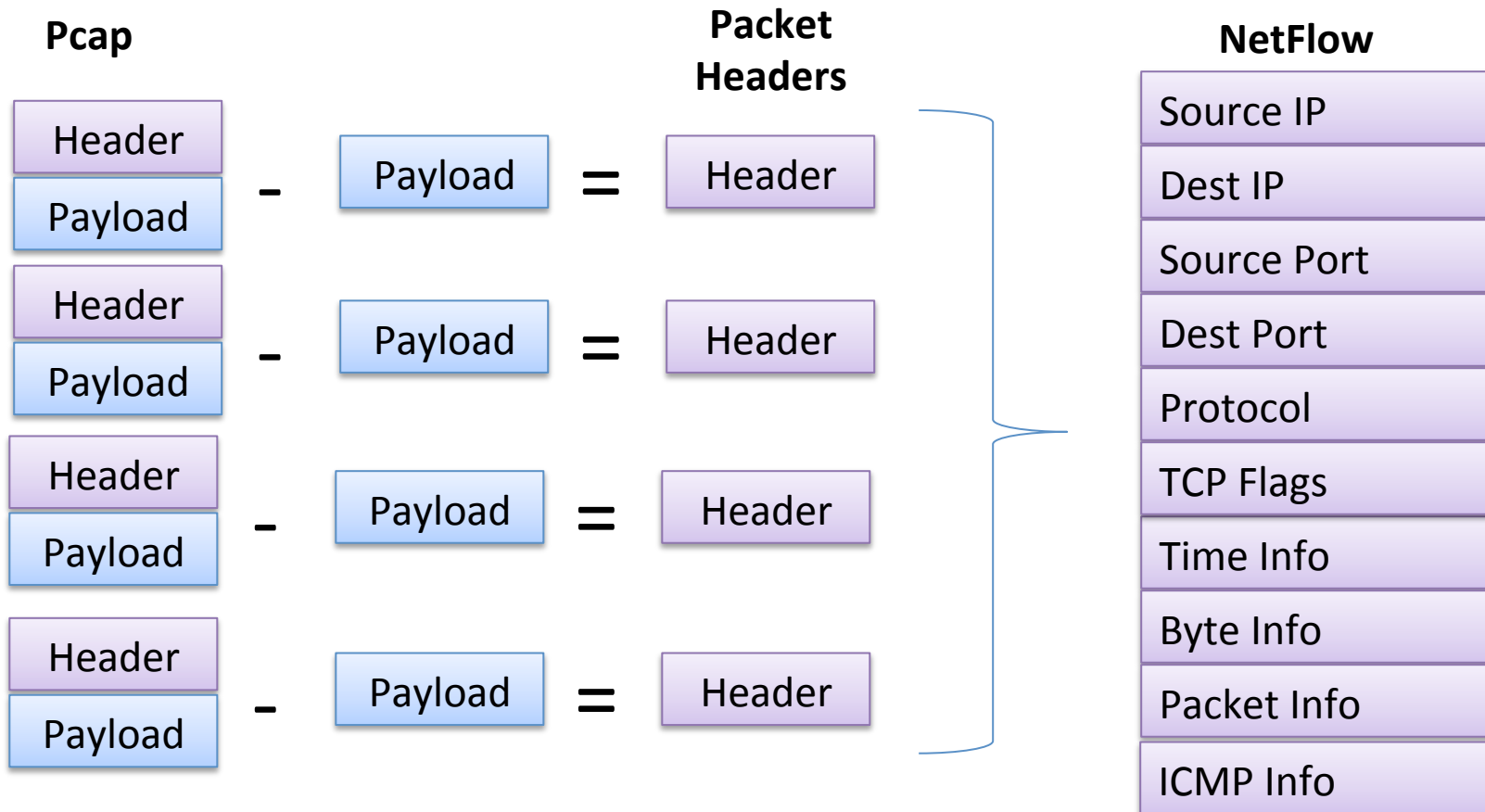
80 GB of PCAP

=



300 MB of Netflow

Netflow data, the “diet” pcap



What netflow data is not

- Replacement for full packet capture
- If you care about the content of the message continue to use full packet capture
- Netflow is like a phone bill
 - You know who called who, but not what was said

Netflow Versions

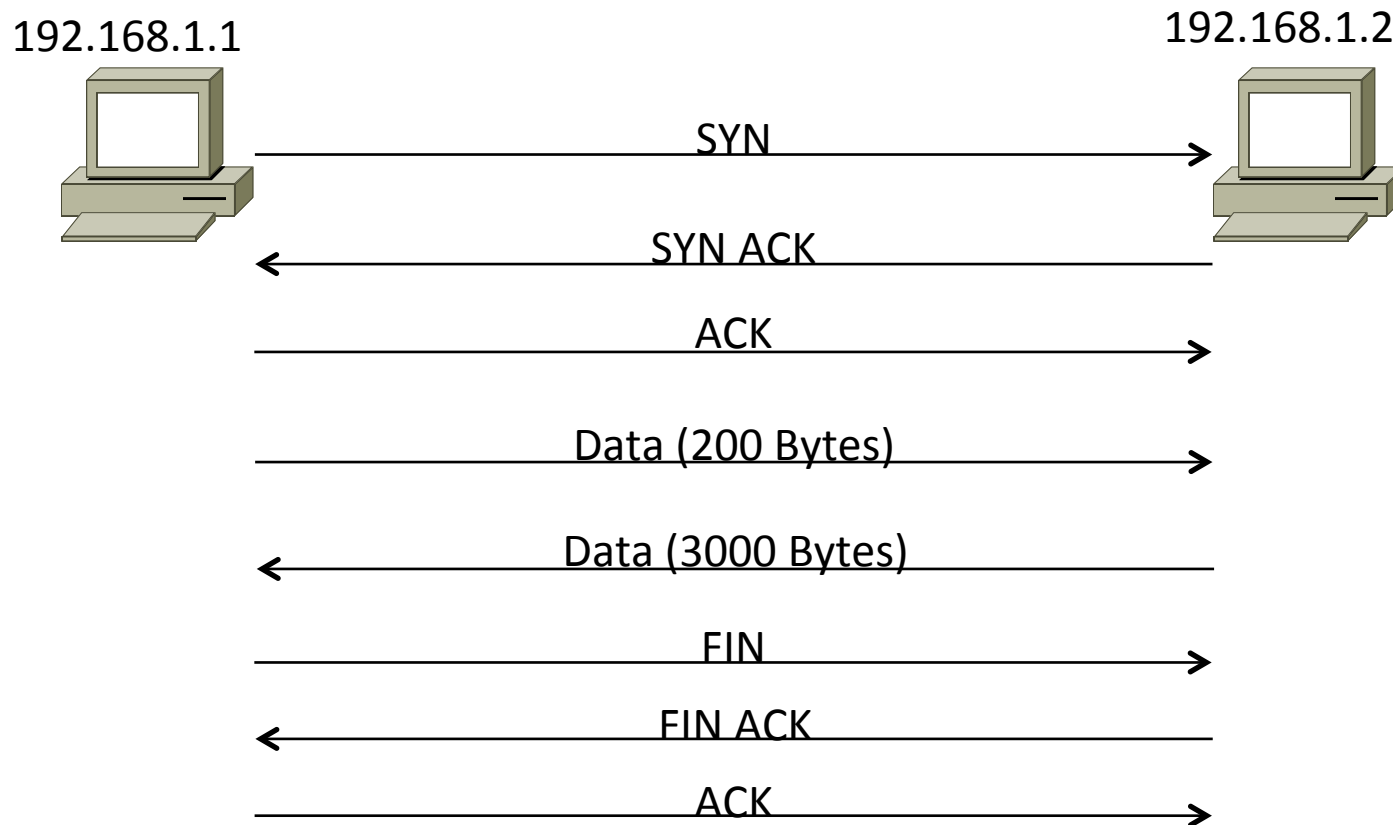
Version	Comment
v1	First implementation, now obsolete, and restricted to IPv4 (without IP mask and AS Numbers).
v2	Cisco internal version, never released.
v3	Cisco internal version, never released.
v4	Cisco internal version, never released.
v5	Most common version, available (as of 2009) on many routers from different brands, but restricted to IPv4 flows.
v6	No longer supported by Cisco. Encapsulation information (?).
v7	Like version 5 with a source router field. Used (only?) on Cisco Catalyst switches.
v8	Several aggregation form, but only for information that is already present in version 5 records
v9	Template Based, available (as of 2009) on some recent routers. Mostly used to report flows like IPv6, MPLS, or even plain IPv4 with BGP nexthop.
v10	aka IPFIX, IETF Standardized NetFlow 9 with several extensions like Enterprise-defined fields types, and variable length fields.

We really only care about v5 and v10 (IPFIX)

http://en.wikipedia.org/wiki/NetFlow#NetFlow_Versions

<http://datatracker.ietf.org/wg/ipfix/charter/>

TCP Connection Example



Uniflow vs Biflow

- Uniflow

192.168.1.1	56981	192.168.1.2	80	1/17/12 15:23:30	300 bytes
-------------	-------	-------------	----	---------------------	-----------

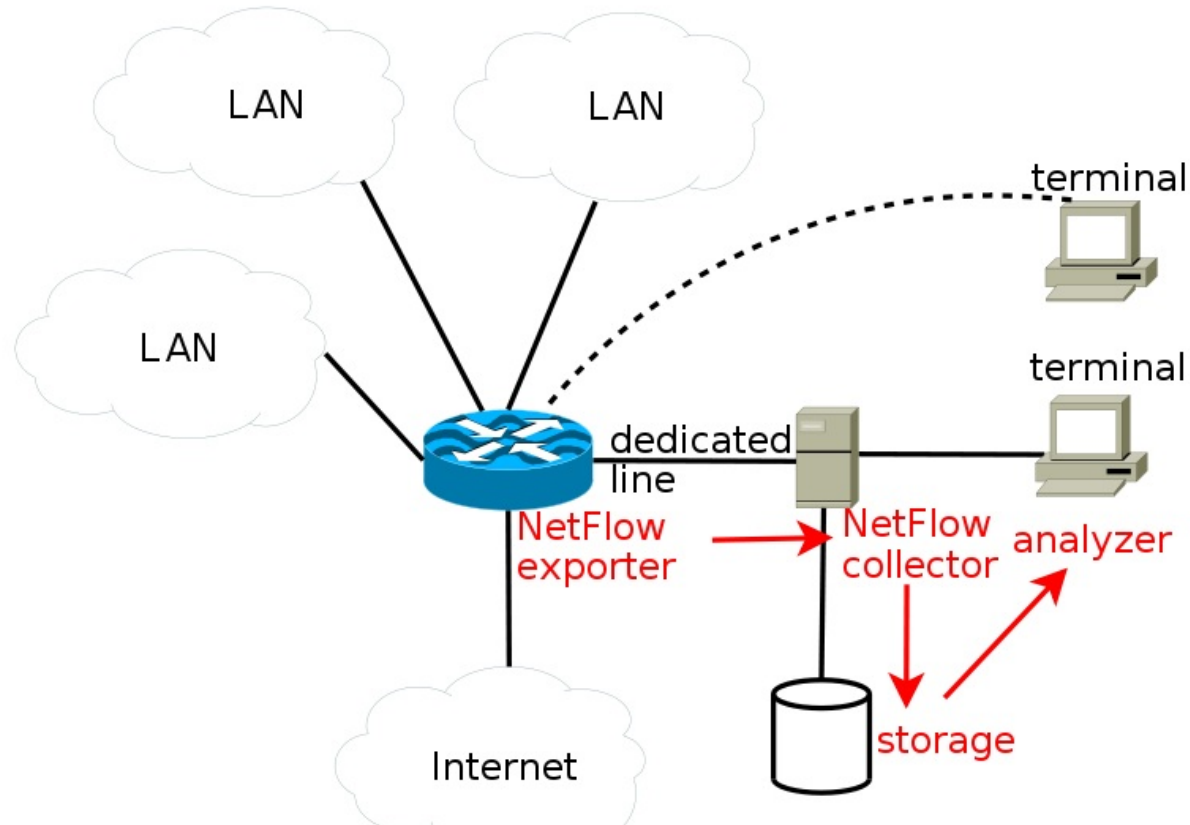
192.168.1.2	80	192.168.1.1	56981	1/17/12 15:23:30	3060 bytes
-------------	----	-------------	-------	---------------------	------------

- Biflow

192.168.1.1	56981	->	192.168.1.2	80	1/17/12 15:23:30	3360 bytes
-------------	-------	----	-------------	----	---------------------	------------

Why don't the sizes listed match up with the amount of data transferred?

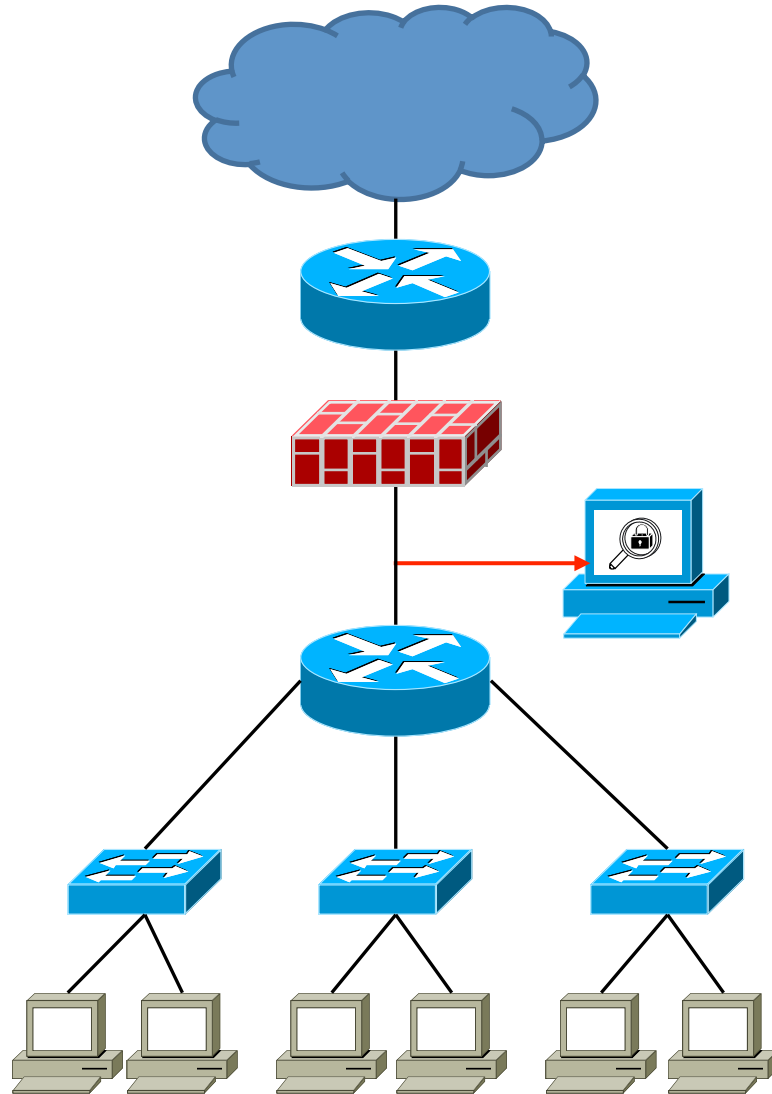
Architectural Discussion



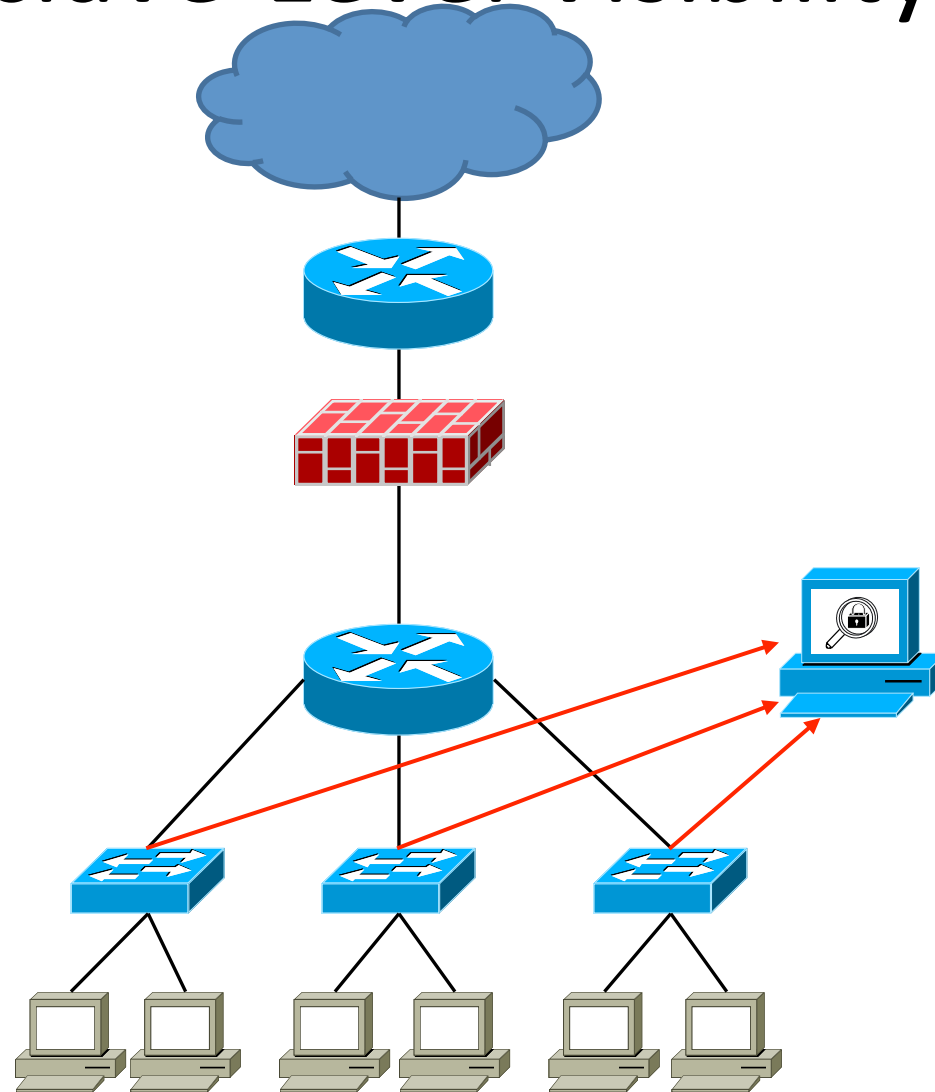
Sensor deployment

- Where are sensors deployed logically?
- Sensor hardware limitations
 - Relative to the bandwidth of the link being monitored
 - Straight collection vs. pushing analytics forward
- Bandwidth back to centralized processing and storage
- Passive stand-alone sensor vs. getting netflow off of routers

Perimeter Visibility



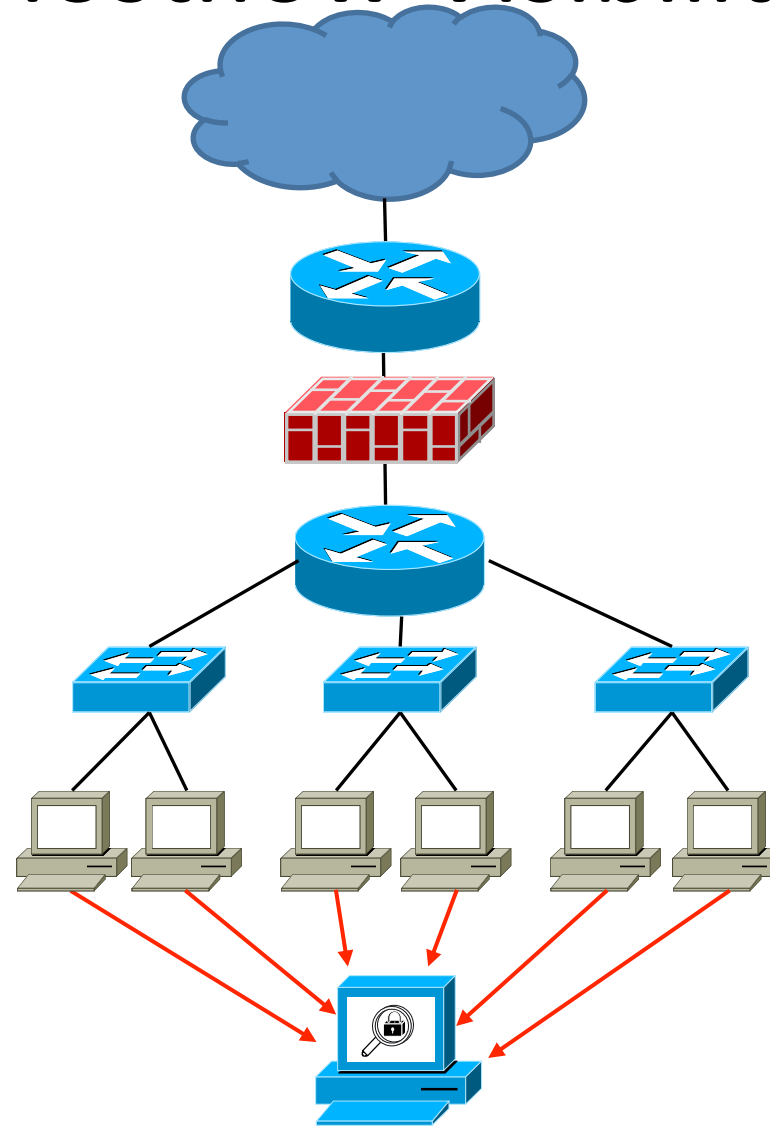
Enclave-Level Visibility



Hostflow

- MITRE developed tool
- Collects netflow-like data from the host
 - Requires deploying an agent on the device
- Bridges the gap between network and host
- Allows visibility when deploying network sensors at the access layer is cost prohibitive

Hostflow Visibility



Bonus: Flow Sampling

- Routers do flow sampling
 - Selecting one out of every n packets
 - Great for net ops, hard to use for security
- Enclave-level sensors produce a lot of flows
- Depending on your analytics you can do some intelligent sampling and aggregation
- Particularly important if your enterprise is geographically distributed

Bonus: Flow Sampling

- Collect only from n hosts in an area (e.g., subnet)
- Collect only for a limited period of time
- Sample only 1 in every n flows
- Ignore common servers and their ports
- Collect only for ports of interest
- Collect only for flows destined for same subnet
- Report only on new sip/sport or dip/dport pairs that haven't been seen in the last hour, day, etc.
- Aggregate into larger timespans

Outline

- Introduction
 - What is Netflow?
 - Sensor Location
 - Sampling
- **Tools**
 - YAF
 - SiLK
 - iSiLK
 - Argus
 - Bro
- Analytics
 - Situational Awareness Analytics
 - Hunting Analytics
 - Data Fusion Analytics
- Wrap Up

Tool Time



Tool Time

- Yaf, SiLK, iSilk, Argus, Bro
- Lab data is supplied from ITOC competition
 - Samples include port scanning, malware c2c behavior, SMTP, HTTP, FTP, SSH

High Level Tools Overview

- YAF: DPI and POf OS fingerprinting
- SiLK: best analysis documentation, files can stay in binary
- iSiLK: nice GUI front end, good for learning SiLK
- Argus: simple server/client install process, many 3rd party tools
- Bro: weird.log file is useful for picking up on strange evasion / misconfigurations

YAF

Yet Another Flowmeter

YAF Outline

- What is YAF?
- YAF optional features
- Where do you get it?
- Converting pcap to YAF netflow format
- What does YAF netflow format look like?

YAF (Yet Another Flowmeter)

- Pcap -> IPFIX netflow format
 - Can also be run on pcap files or on the wire
- Optional features
 - Biflow extension
 - Application labeling
 - OS detection
 - Deep packet inspection
- See
 - <http://tools.netsa.cert.org/yaf/yafdpi.html>
 - <http://tools.netsa.cert.org/yaf/applabel.html>
 - <http://tools.netsa.cert.org/yaf/>
 - <http://tools.netsa.cert.org/yaf/docs.html>

YAF Optional Feature: applabel

- YAF application labeling
 - Supports yafscii, rwflowpack, flowcap, and rwipfix2silk
 - Adds an additional column of data
- App Labeling Rules
 - Rules located in /usr/local/etc/yafApplabelRules.conf
 - label <N> regex <expression>
 - label <N> plugin <library> <function> <arg-list>
 - label <N> signature <expression>
 - label 80 regex HTTP/\d\.\d\b
 - label 53 plugin dnsplugin
dnsplugin_LTX_ycDnsScanScan

YAF Optional Feature: p0f

- P0f OS fingerprinting
 - Passive OS identification using libP0f
 - Packet size
 - Window size
 - Flags
 - More info `/usr/local/etc/p0f.fp`
- DHCP OS fingerprinting
 - `/usr/local/etc/dhcp_fingerprints.conf`
- Output viewed with `yaf-file-mediator`

<http://tools.netsa.cert.org/yaf/yafdhcp.html>

<https://tools.netsa.cert.org/confluence/display/tt/libp0f>

YAF Optional Feature: DPI

- Deep packet inspection
- App labeling needs to be used
 - Different per application (HTTP, DNS, etc.)
- In order to enable DPI in YAF:
 - `--plugin-name=/usr/local/lib/yaf/dpacketplugin.la`
- Specify which protocols to perform DPI:
 - `--plugin-opts="53 80 21"`
 - The above will perform DPI for DNS, HTTP, and FTP

YAF Optional Feature: DPI

- FTP, HTTP, IMAP, SSH, DNS, SSL/TLS, IRC, POP3, MySQL
 - FTP: commands/replies
 - HTTP: server response header, user agent, location response header, response code, cookie headers
 - IMAP: command and response, login and pass, authenticate mechanism, # of messages in mailbox
 - SSH: version number
 - DNS: query/response type, header field, etc.
 - More info available `/usr/local/etc/yafDPIRules.conf`
- Output viewed with `yaf-file-mediator`

YAF Optional Feature: DPI

- Examples (FTP)

label 21 yaf 131 (?i)(REST \d+|RETR \w+|STO[RU]
\w+)\b

label 21 yaf 132 (?i)USER (\w+)\b

label 21 yaf 133 (?i)PASS ([\w.@]+\b

label 21 yaf 134 (?i)TYPE (A|E|I)\b

label 21 yaf 135 (?i)([1-5][0-5][0-7] [\w\s]+\b

Acquiring YAF

- Easy way in a premade Linux
 - <http://tools.netsa.cert.org/livecd.html>
- RPMs are available for Fedora 15, and Redhat 5
 - <https://tools.netsa.cert.org/confluence/display/tt/RPMs+of+NetSA+Tools>
- Compiling from source
 - <http://tools.netsa.cert.org/yaf/docs.html>
 - <https://tools.netsa.cert.org/confluence/pages/viewpage.action?pageId=23298051>
 - Have fun, some dependencies are only available at <http://netsa.cert.org>
 - Remember, some of the optional features need flags when running `./configure` when building YAF

Converting pcap to IPFIX

- Standard conversion
 - `yaf -in filename.pcap --out filename.yaf`
- With application labeling
 - Add `--applabel-rules= /usr/local/etc/yafApplabelRules.conf --max-payload 300`
- POf
 - Add `--p0fprint --p0f-fingerprints /usr/local/etc/ --max-payload 300`
- DHCP finger printing
 - Add `--plugin-name=/usr/local/lib/yaf/dhcp_fp_plugin.la`
- Deep Packet Inspection
 - Add `--plugin-name=/usr/local/lib/yaf/dpacketplugin.la`
- Man `yaf` and `yafdpi`

Binary IPFIX to ASCII

- Necessary since many tools cannot display IPFIX data
- Convert from binary YAF to ascii with yafscii
 - yafscii --in filename.yaf
- POf and dpi data must use YAF 2.0 IPFIX file mediator
 - yaf_file_mediator --input in_file.yaf --out out_file.txt

YAF Output: TCP Example

2009-04-21 08:07:08.676 - 08:07:08.719 (0.043
sec)

tcp 10.1.10.65:49157 => 10.2.250.136:80

3b3aa589 S/APR vlan 014 (5/336 ->) applabel:
80

YAF Output: TCP Reset

2009-04-21 08:08:02.042

tcp 10.2.254.116:443 => 10.2.200.248:49387

00000000 AR/0 (1/40 ->)

YAF Output: UDP

2009-04-21 08:06:36.713 - 08:07:08.874
(32.161 sec)

udp 10.2.196.253:137 => 10.2.255.255:137
(12/1080 ->) idle applabel: 137

SiLK

System for Internet-Level Knowledge

SiLK Installation

- Time consuming to install from source, handbook here
<http://tools.netsa.cert.org/silk/install-handbook.html>
- There is a live cd
<http://tools.netsa.cert.org/livecd.html>
- Back in my day we didn't have Redhat/Fedora RPMS
<https://tools.netsa.cert.org/confluence/display/tt/RPMs+of+NetSA+Tools>

SiLK Optional Tools

- Country codes
- DNS lookups
- IPv6
- For more info
 - <http://tools.netsa.cert.org/silk/install-handbook.html>
 - <http://tools.netsa.cert.org/silk/install-handbook.html#x1-160002.3>

SiLK Familiarization

- Learning to crawl with rwcut
- Learning to walk with rwcut & rwfilter
- SiLK commands are piped to form a workflow
 - Same idea as Linux commands

Building SiLK

- Needs YAF, python-dev
- <http://tools.netsa.cert.org/silk/install-handbook.html#x1-160002.3>

SiLK Commands: rwfileinfo

- Provide basic metadata on a SiLK file

- Try it:

```
rwfileinfo 20120501-1400-1500.rwf
```

SiLK Commands: rwcut

- View flow records as text

```
rwcut --num-rec=10 20120501-1400-1500.rwf
```

- You can also specified fields to print

```
rwcut --num-rec=10 --  
fields=sip,dip,proto,sport,dport,stime  
20120501-1400-1500.rwf
```

SiLK Commands: rwtotal

- Count how much traffic matched specific keys
- What layer 4 protocols (TCP, UDP, etc) are running?

```
rwtotal --proto --skip-zero  
20120501-1400-1500.rwf
```

- Note: the above is identical to `rwuniq --field=proto --values=records,bytes,packets --sort-output 20120501-1400-1500.rwf`
- `rwtotal` runs faster and uses a fixed amount of memory, but has less functionality.

SiLK Commands: rwuniq

- Like rwtotal, but with more key options
- We want common servers and src ports with at least 50 flows. For each server/port pair display flows, total bytes and distinct dips.
- `rwfilter --proto=6,17 --pass=stdout 20120501-1400-1500.rwf | rwuniq --field=sip,sport --flows=50 --bytes --values=dip-distinct | sort -nr -k 3,3 -t '|' | head`

SiLK Commands: rwstats

- Generate top-N/bottom-N lists or overall stats
- What does the distribution of bytes, packets and bytes/packet look like?

```
rwstats --overall-stats  
20120501-1400-1500.rwf
```

SiLK Commands: rwstats (cont'd)

- How about the 10 top destination ports?

```
rwstats --fields=dport --count=10  
20120501-1400-1500.rwf
```

- And the top 10 source ports?

```
rwstats --fields=sport --count=10  
20120501-1400-1500.rwf
```

- We can also use percentage based cutoffs

```
rwstats --fields=dport --percentage=1  
20120501-1400-1500.rwf
```

SiLK Commands: rwcoun

- Examine traffic binned over time
- Let's chop our hour into five minute intervals

```
rwcoun --bin-size=300  
20120501-1400-1500.rwf
```

SiLK Commands: rwcoun (cont'd)

- Why are there bins after 15:00 if the traffic is from 14:00-15:00?
- How does this output differ?

```
rwcoun --bin-size=300 --load-scheme=1  
20120501-1400- 1500.rwf
```

- The above is nearly identical to

```
rwstats --fields=stime --  
values=records,bytes,packets --bin-time=300 --  
percentage=1 20120501-1400-1500.rwf | tail -n +4 |  
sort -t '|'
```

SiLK Commands: rfilter

- Swiss-army knife for filtering flows
- There are switches for every flow attribute
- Let's see what our top webservers are located

```
rfilter 20120501-1400-1500.rwf --  
sport=80,443,8080 -- protocol=6 --packets=4- --  
ack-flag=1 --pass=stdout | rwstats --fields=sip  
--percentage=1 --bytes
```

SiLK Commands: rwscore

- Employs two algorithms to detect scans

```
rwscore --fields=sip,proto,dip  
20120501-1400-1500.rwf | rwscore --  
scan-model=2
```

- The other scan model doesn't work well for this particular data

Other ways to look for scanning

```
rwfilter 20120501-1400-1500.rwf --  
bytes=0-2048 -- packets=1-3 --flags-all=/RF --  
pass=stdout | rwuniq --      fields=sip --  
values=dip-distinct,records | sort -k 3,3 -n -r -t '|'   
| head -n 30
```

- Meaning:
 - Size less than 2048 bytes
 - 1 to 3 packets
 - No RST or FIN flags
 - Per sip, |dip| and record count

SiLK Commands: IP Sets

- Useful for dealing with summaries of data
- Describe collections of arbitrary IP addresses
- Set operations are supported by `rwsettool`
 - Intersection
 - Difference
 - Union
- `Rwfilter` can take IP sets as parameters
- Example: you can build an IP set for all of your web servers and then profile their traffic

SiLK Commands: IP Sets

- Simple example:
 - Look for sources in 10.0.0.0/24
- `echo 10.0.0.0-255 > set_a.txt`
- `rwsetbuild set_a.txt a.set`
- `rwfilter 20120501-1400-1500.rwf -- sipset=a.set --pass=stdout | rwcuf --num-rec=10`

SiLK Commands: IP Bags

- No, not that



SiLK Commands: IP Bags

- Bags are like sets, but include volume
- Convert rwsan output to a bag with flow counts

```
rwsort --fields=sip,proto,dip  
20120501-1400-1500.rwf| rwsan -- scan-  
model=2 --no-titles | cut -d '|' -f 1,5 | rwbagbuild  
--bag-      input=stdin > rwsan-output.bag
```

- View the output

```
rwbagcat rwsan-output.bag | sort -t '|' -k  
2,2 -rn | head
```

SiLK Commands: IP Bags (cont'd)

- Capture our rfilter scanning output in a bag

```
    rfilter 20120501-1400-1500.rwf --  
bytes=0-2048 --      packets=1-3 --flags-all=  
RF --pass=stdout | rwbag --sip-  
    flows=rfilter-scan-output.bag
```

- What does the data look like?

```
    rwbagcat rfilter-scan-output.bag | sort  
-t '|' -k 2,2 -rn | head      -n 20
```

SiLK Commands: IP Bags (cont'd)

- Limit it to 10k flows (after eyeballing the data)

```
rwbagtool --mincounter=10000 rwfilter-  
scan-output.bag >  rwfilter-scan-  
output-10k.bag
```

SiLK Commands: IP Bags (cont'd)

- Combine bags from our two scanning methods (this could also be done with ipsets)

```
rwbagtool --maximize rwscan-  
output.bag rwfilter-scan- output-10k.bag >  
union-scan.bag
```

- View the output

```
rwbagcat union-scan.bag | sort -t '|' -k  
2,2 -rn | head
```

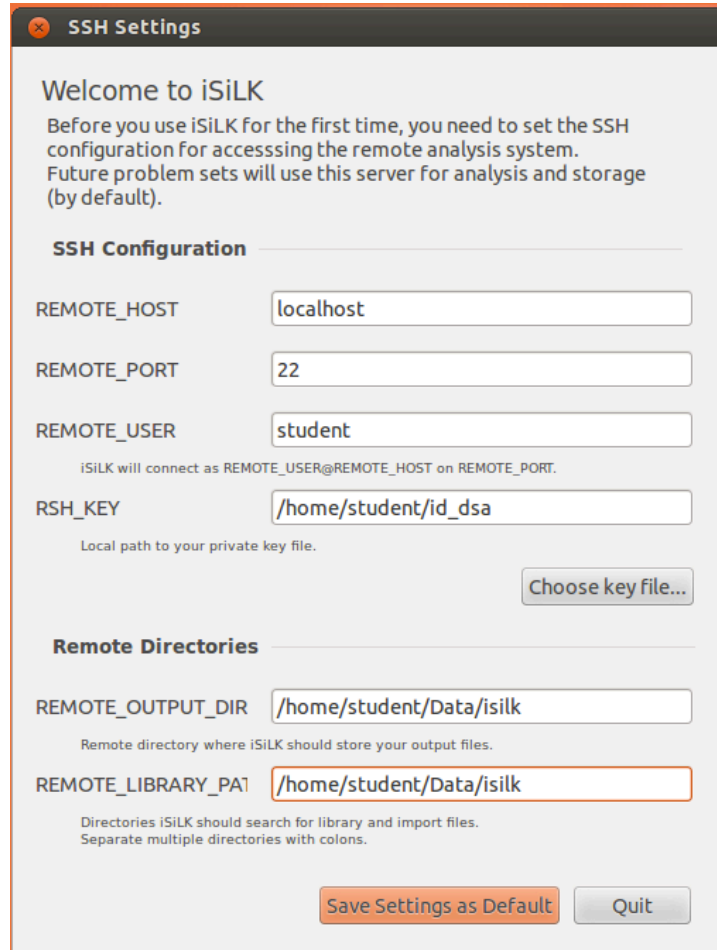
iSiLK

<http://tools.netsa.cert.org/isilk/index.html>

iSilk Overview

- GUI for silk tools
- Provides subset of command line functionality
- Windows installer
- Build instructions
 - <http://tools.netsa.cert.org/isilk/isilk-admin-guide.pdf>
 - <http://tools.netsa.cert.org/isilk/isilk-user-guide.pdf>

iSiLK: Setup



The screenshot shows the 'SSH Settings' dialog box for iSiLK. It contains a welcome message, an 'SSH Configuration' section with fields for REMOTE_HOST (localhost), REMOTE_PORT (22), REMOTE_USER (student), and RSH_KEY (/home/student/id_dsa), and a 'Remote Directories' section with fields for REMOTE_OUTPUT_DIR and REMOTE_LIBRARY_PATH, both set to /home/student/Data/isilk. There are buttons for 'Choose key file...', 'Save Settings as Default', and 'Quit'.

SSH Settings

Welcome to iSiLK

Before you use iSiLK for the first time, you need to set the SSH configuration for accessing the remote analysis system. Future problem sets will use this server for analysis and storage (by default).

SSH Configuration

REMOTE_HOST

REMOTE_PORT

REMOTE_USER

iSiLK will connect as REMOTE_USER@REMOTE_HOST on REMOTE_PORT.

RSH_KEY

Local path to your private key file.

Remote Directories

REMOTE_OUTPUT_DIR

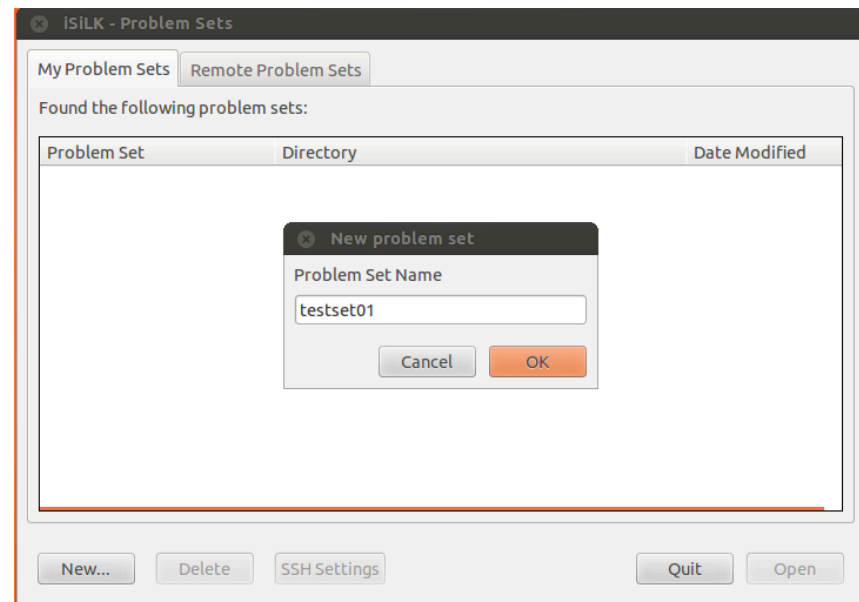
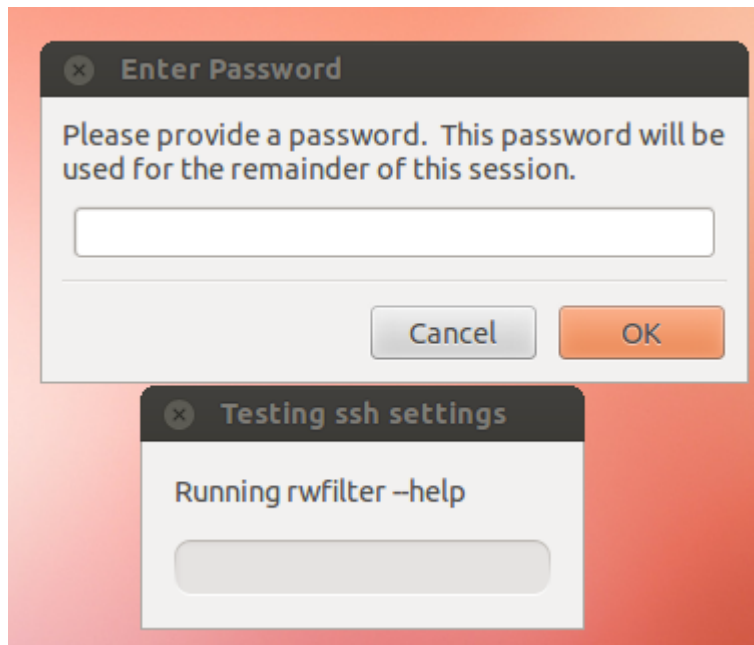
Remote directory where iSiLK should store your output files.

REMOTE_LIBRARY_PATH

Directories iSiLK should search for library and import files. Separate multiple directories with colons.

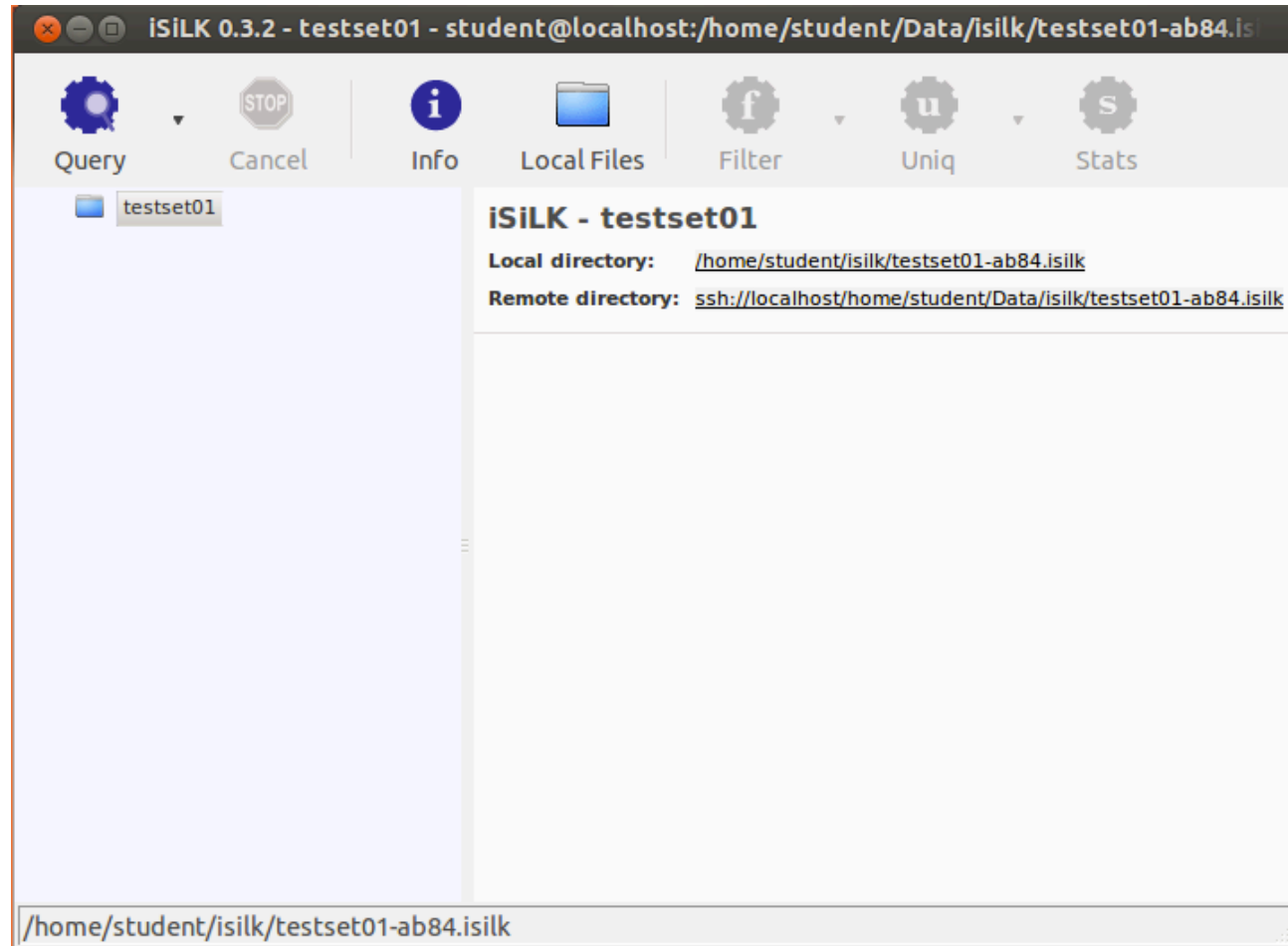
Note: isilk needs country codes <http://tools.netsa.cert.org/silk/rwgeoip2ccmap.html>

iSiLK: Setup

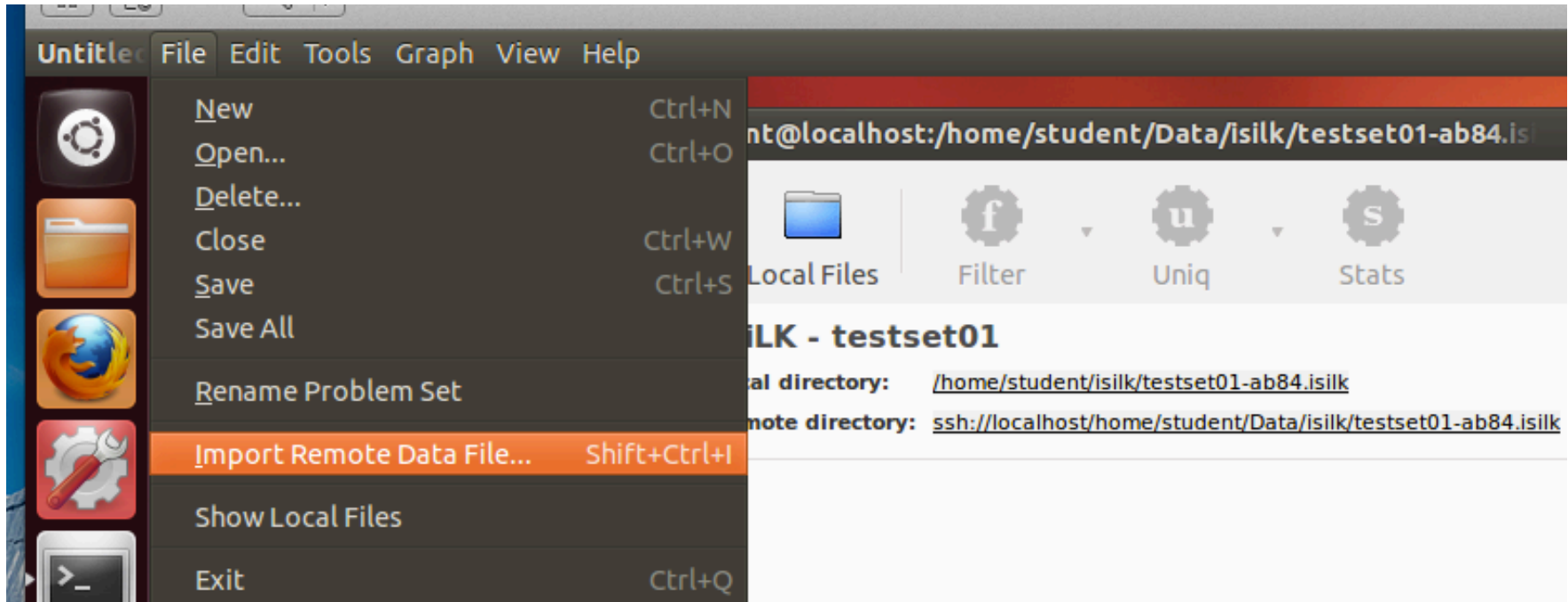


<http://tools.netsa.cert.org/isilk/isilk-admin-guide.pdf>
<http://tools.netsa.cert.org/isilk/isilk-user-guide.pdf>

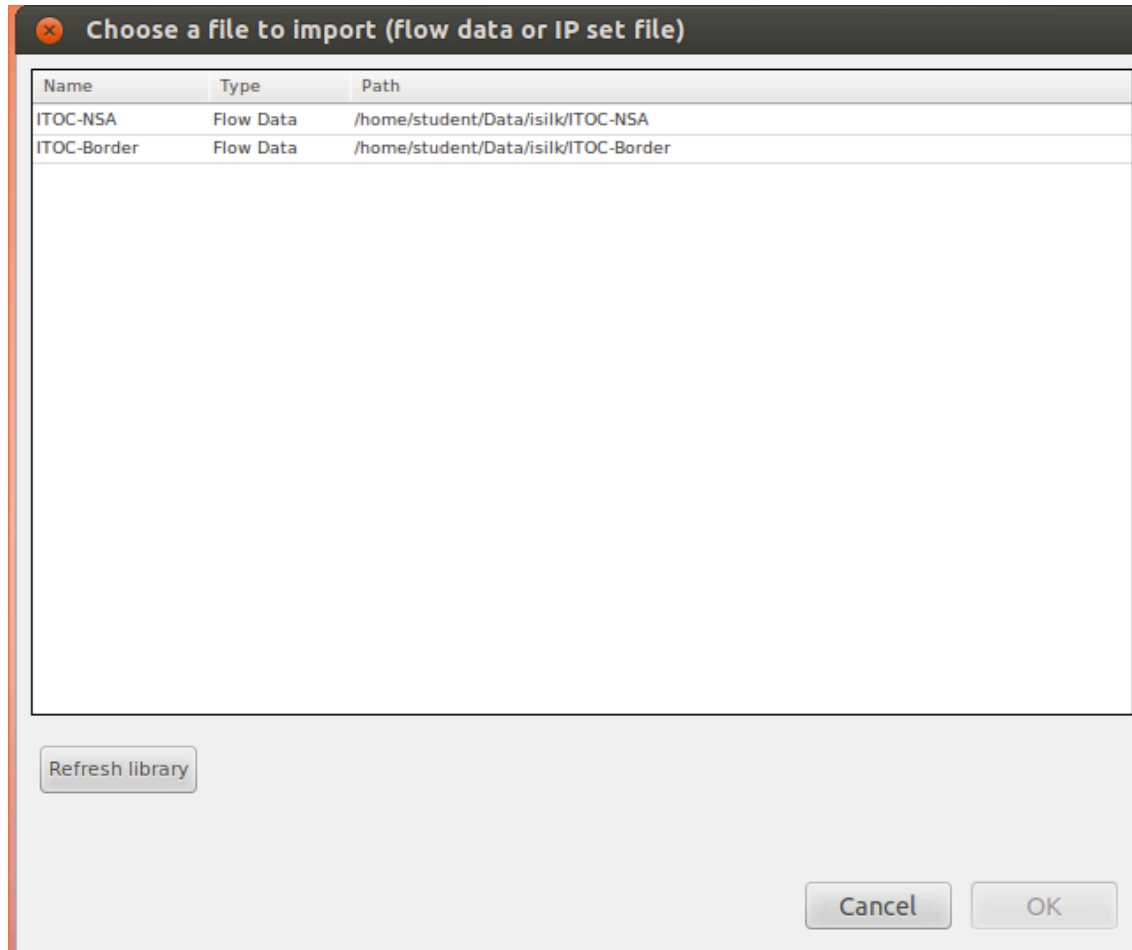
iSilk: Adding Files



iSilk: Adding Files



iSiLK: Adding Files



iSiLK: rwuniq

Running rwuniq

Count by

- Source IP (sip)
- Source Country (scc)
- Source Port (sport)
- Destination IP (dip)
- Destination Country (dcc)
- Destination Port (dport)
- Protocol (proto)
- Sensor (sensor)
- Next Hop IP (nhip)

Apply a prefix map

(Choose a prefix map)

Clear Choose...

Volume fields

- Bytes
- Packets
- Flow Records
- Unique Source IPs
- Unique Destination IPs

Source pmap Value (sval)

Destination pmap Value (dval)

```
rwuniq Imported_Query_Result-hn01.rwf --fields=sip --output-path=$output --bytes
```

Name:

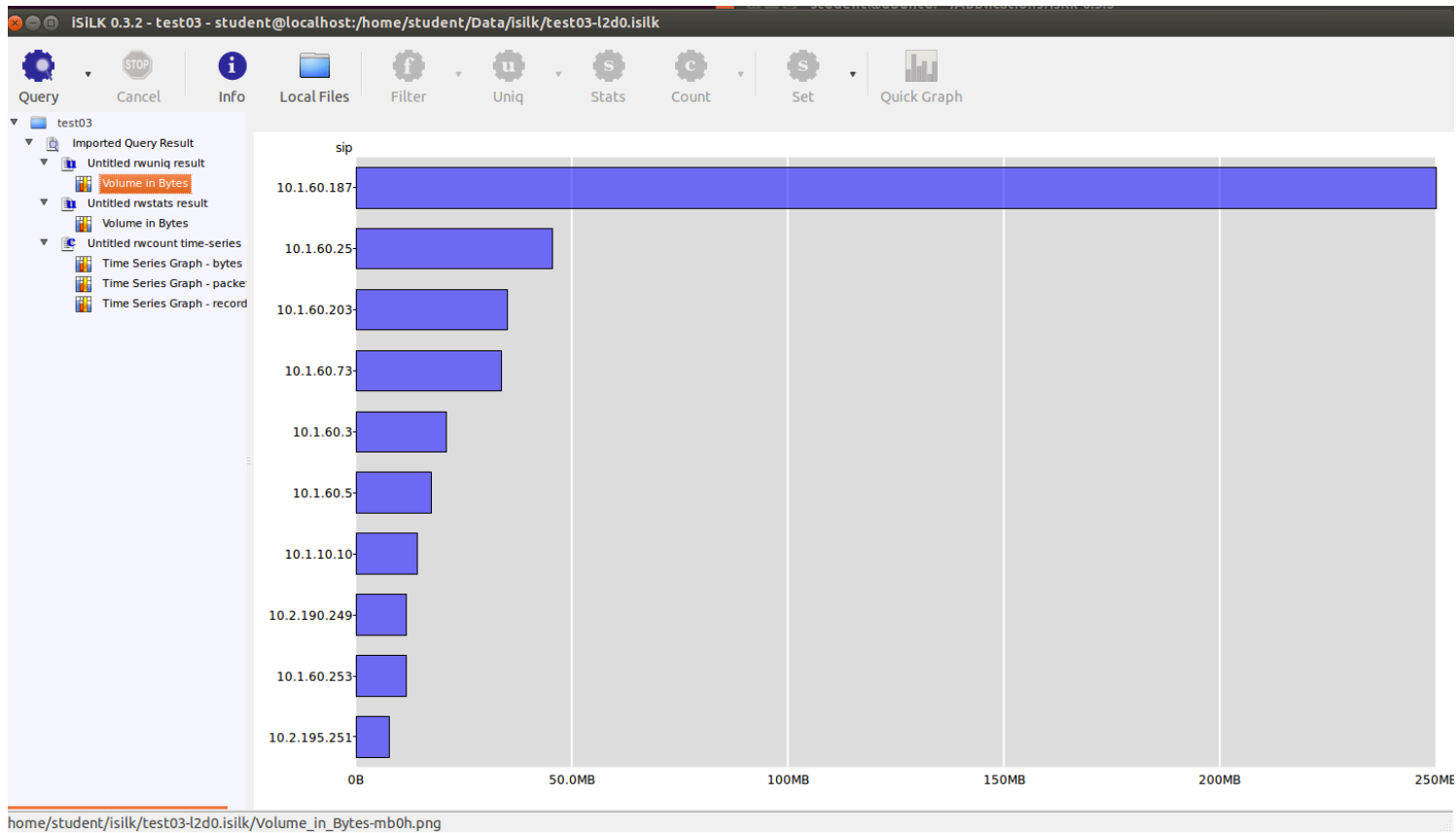
Cancel Run Analysis

iSilk: rwuniq output

54,523 records - /home/student/isilk/test03-l2d0.isilk/Untitled rwuniq result-brch.asc

#	sip	bytes
42246	10.1.60.187	249,966,150
34336	10.1.60.25	45,257,275
26871	10.1.60.203	34,873,480
40610	10.1.60.73	33,581,710
9337	10.1.60.3	20,804,684
22755	10.1.60.5	17,287,955
19482	10.1.10.10	14,003,752
39646	10.2.190.249	11,563,191
19738	10.1.60.253	11,533,035
22635	10.2.195.251	7,671,087
4977	10.1.90.5	4,860,676
47090	10.1.10.5	4,730,009
28920	10.1.60.132	3,023,366
40278	10.2.197.241	2,863,071
1362	10.2.199.236	2,683,244
49472	10.1.60.153	2,629,846
39599	10.1.20.4	2,583,134
25225	10.2.20.60	2,573,620
13135	10.2.198.245	2,479,996
5125	10.1.80.9	2,274,572
19846	10.1.70.200	2,217,843
3591	10.2.197.245	1,915,618
42993	10.1.30.5	1,900,614
42195	10.1.10.20	1,842,297
42779	10.2.190.254	1,825,169
15872	10.2.19.22	1,817,160
18266	10.2.23.195	1,736,453
6192	10.2.199.235	1,651,282

iSilk: rwuniq graph



iSiLK: rwstats

Running rwstats

Threshold
Calculate statistics based on the

top N keys

bottom N keys

by

count

threshold value

percentage (%)

Count by

Source IP (sip)

Source Port (sport)

Destination IP (dip)

Destination Port (dport)

Protocol (proto)

Volume fields

Bytes

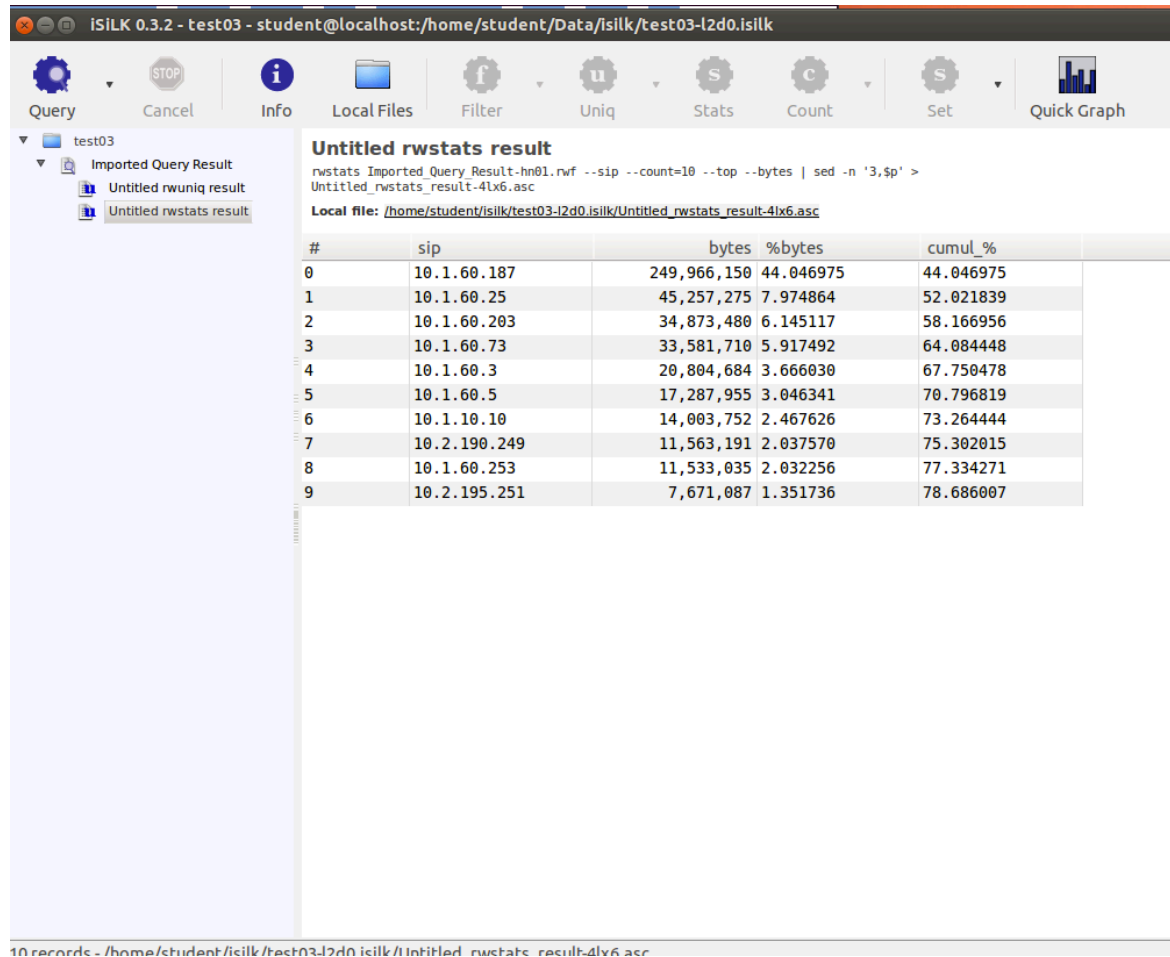
Packets

Flow Records

```
rwstats Imported_Query_Result-hn01.rwf --count=10 --top --bytes | sed -n '3,$p' > $output
```

Name

iSilk: rwstats output



iSilk 0.3.2 - test03 - student@localhost:/home/student/Data/isilk/test03-l2d0.isilk

Query Cancel Info Local Files Filter Uniq Stats Count Set Quick Graph

test03

- Imported Query Result
 - Untitled rwuniq result
 - Untitled rwstats result

Untitled rwstats result

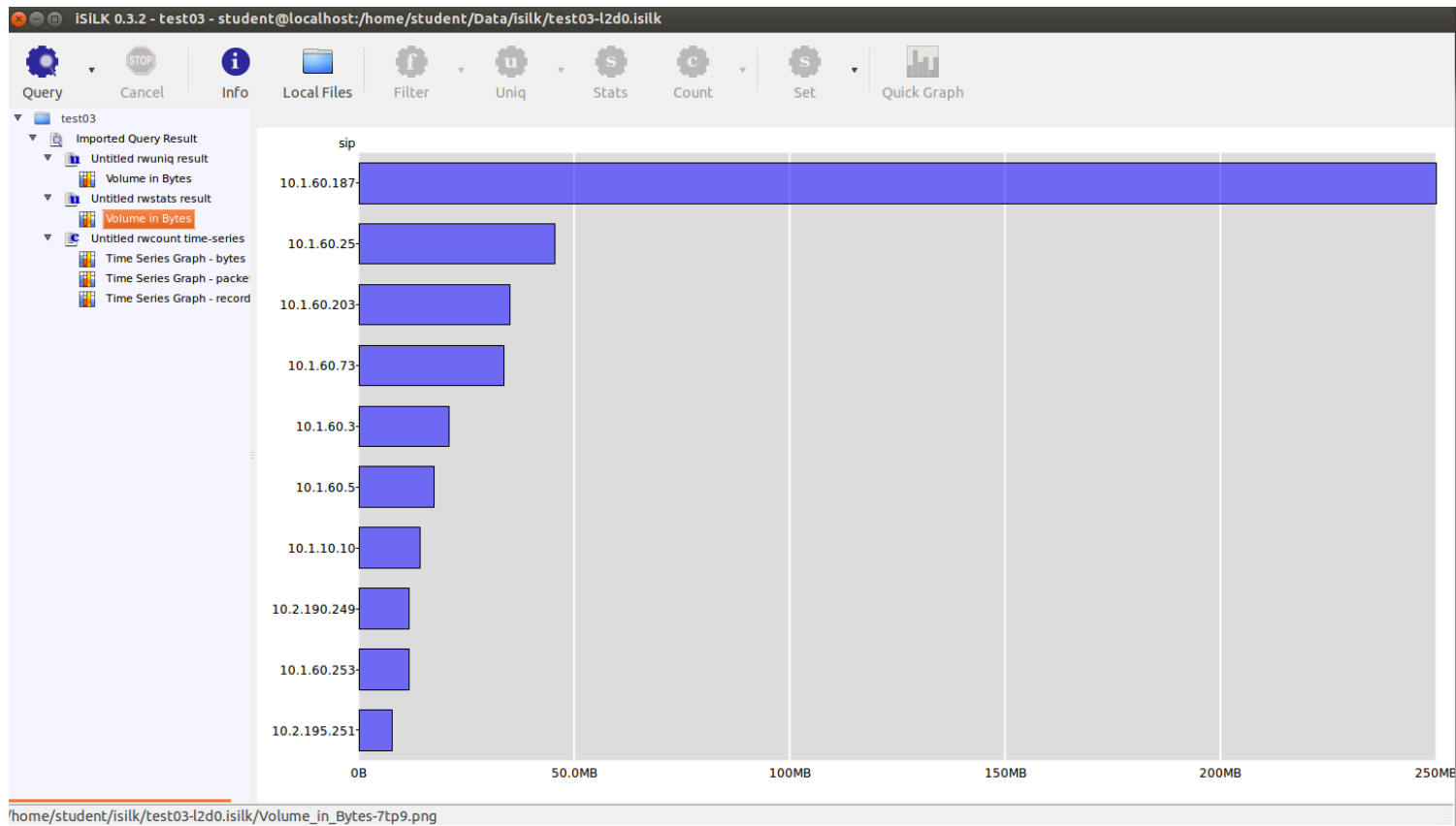
```
rwstats Imported_Query_Result-hn01.rwf --sip --count=10 --top --bytes | sed -n '3,$p' > Untitled_rwstats_result-4lx6.asc
```

Local file: [/home/student/isilk/test03-l2d0.isilk/Untitled_rwstats_result-4lx6.asc](#)

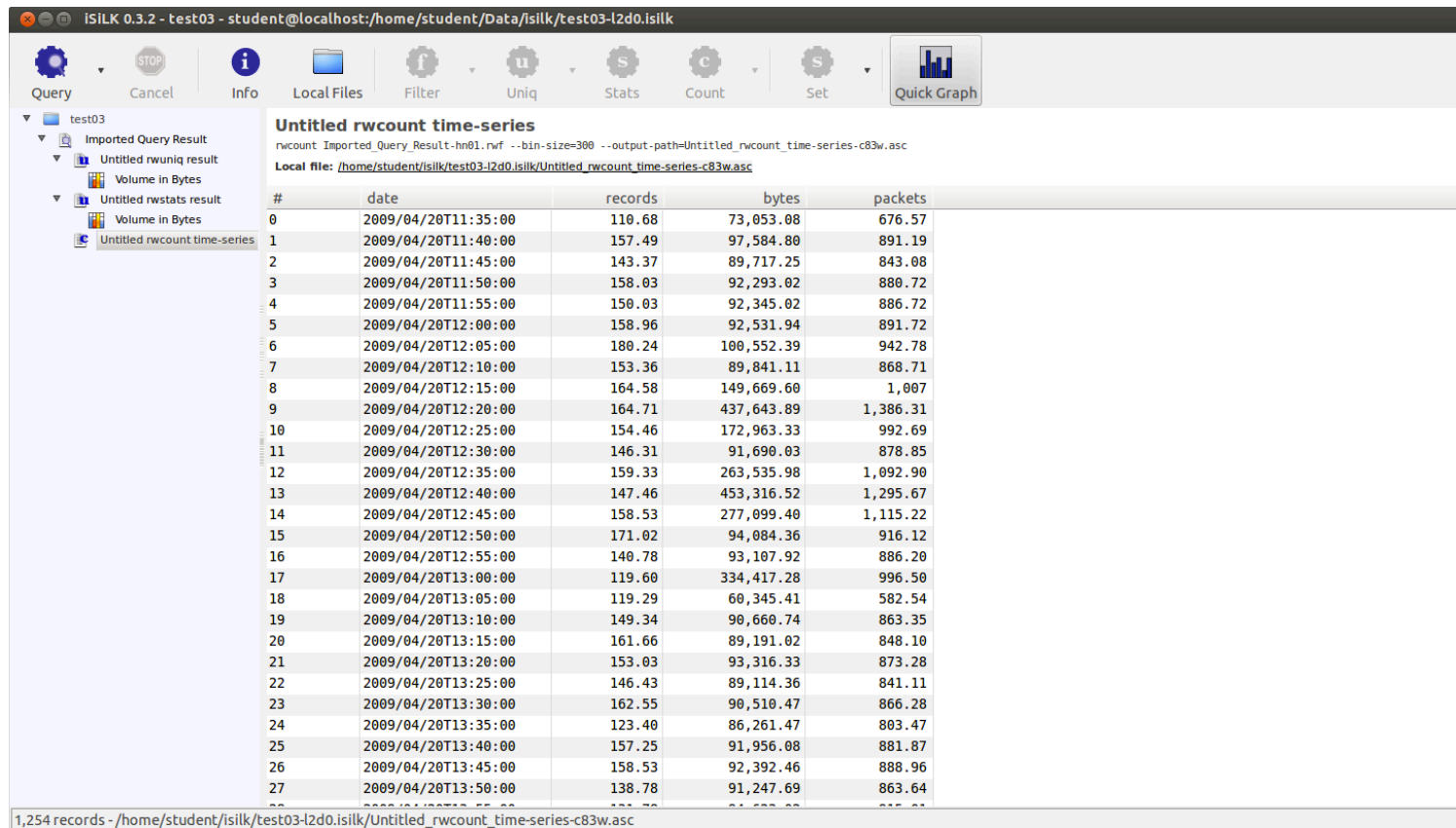
#	sip	bytes	%bytes	cumul_%
0	10.1.60.187	249,966,150	44.046975	44.046975
1	10.1.60.25	45,257,275	7.974864	52.021839
2	10.1.60.203	34,873,480	6.145117	58.166956
3	10.1.60.73	33,581,710	5.917492	64.084448
4	10.1.60.3	20,804,684	3.666030	67.750478
5	10.1.60.5	17,287,955	3.046341	70.796819
6	10.1.10.10	14,003,752	2.467626	73.264444
7	10.2.190.249	11,563,191	2.037570	75.302015
8	10.1.60.253	11,533,035	2.032256	77.334271
9	10.2.195.251	7,671,087	1.351736	78.686007

10 records - /home/student/isilk/test03-l2d0.isilk/Untitled_rwstats_result-4lx6.asc

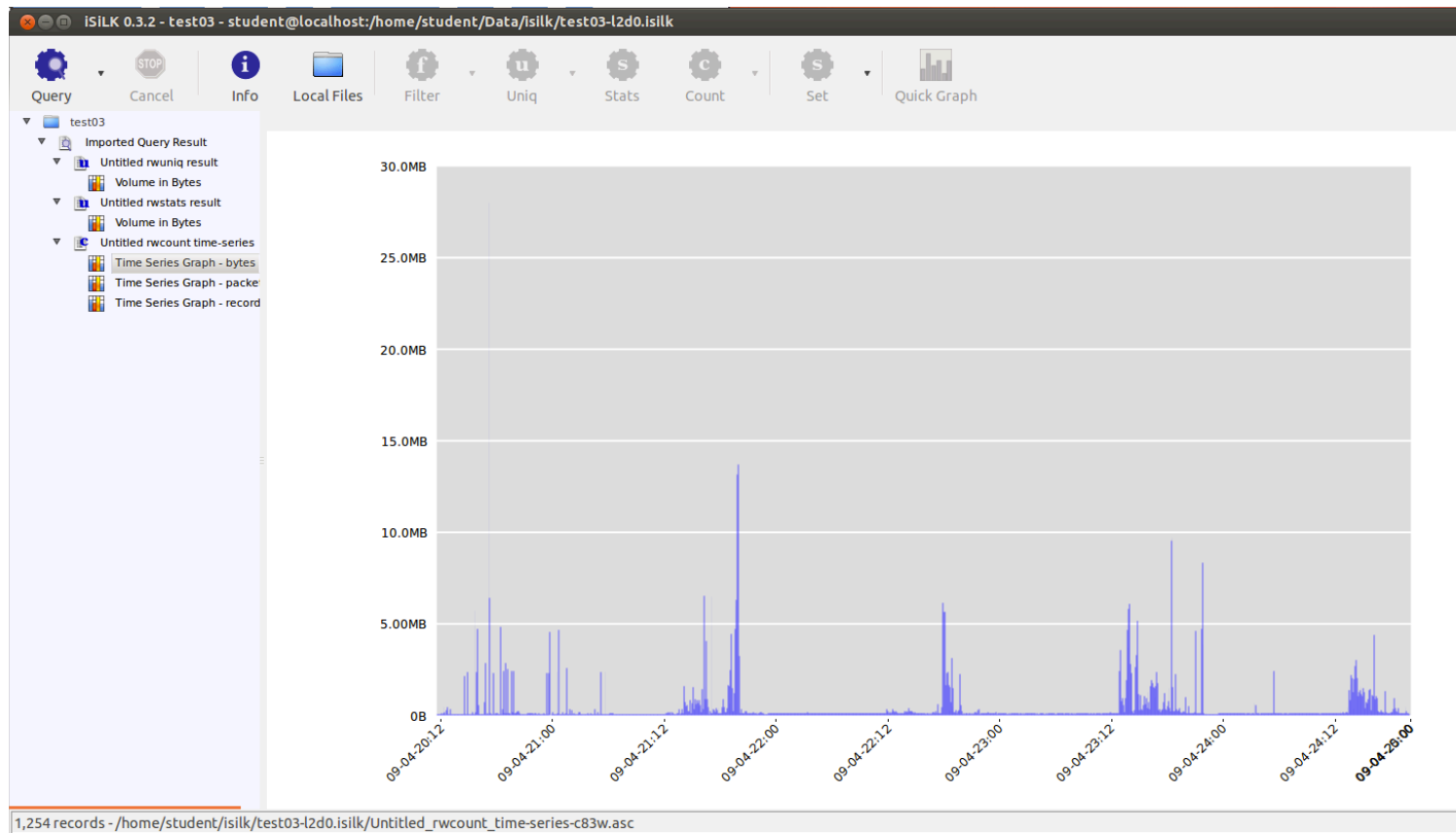
iSilk: rwstats graph



iSiLK: rwcount



iSiLK: rwcount graph



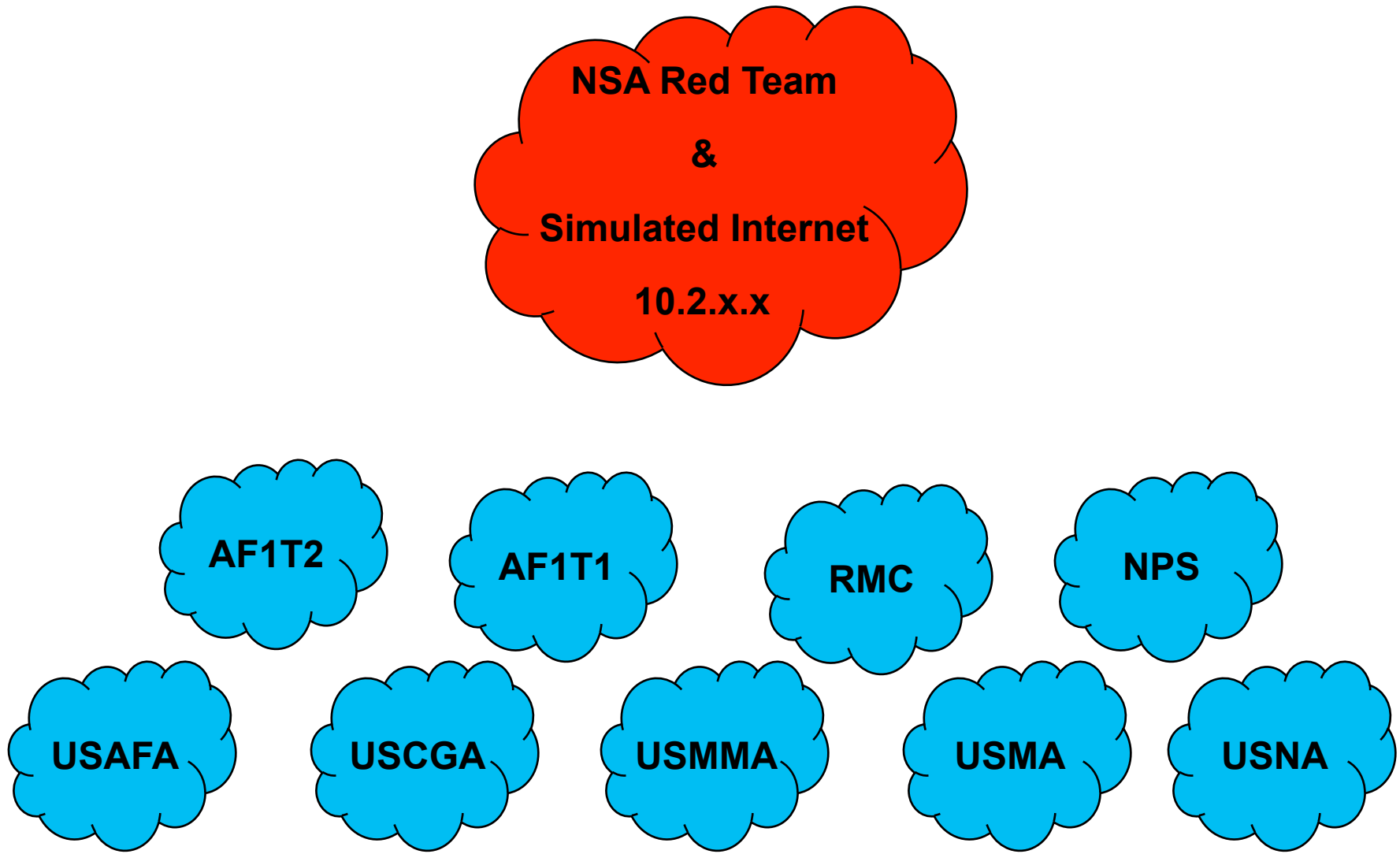
iSiLK: Lab Network Data

- For this lab we will be using data from the ITOC competition from 2009.
- Information ITOC (CRC) competition
 - <http://www.westpoint.edu/crc/SitePages/About.aspx>
 - http://static.usenix.org/event/cset09/tech/full_papers/sangster.pdf
- Download location
 - <http://www.westpoint.edu/crc/SitePages/DataSets.aspx>
- Other data for practicing
 - <http://www.netresec.com/?page=PcapFiles>

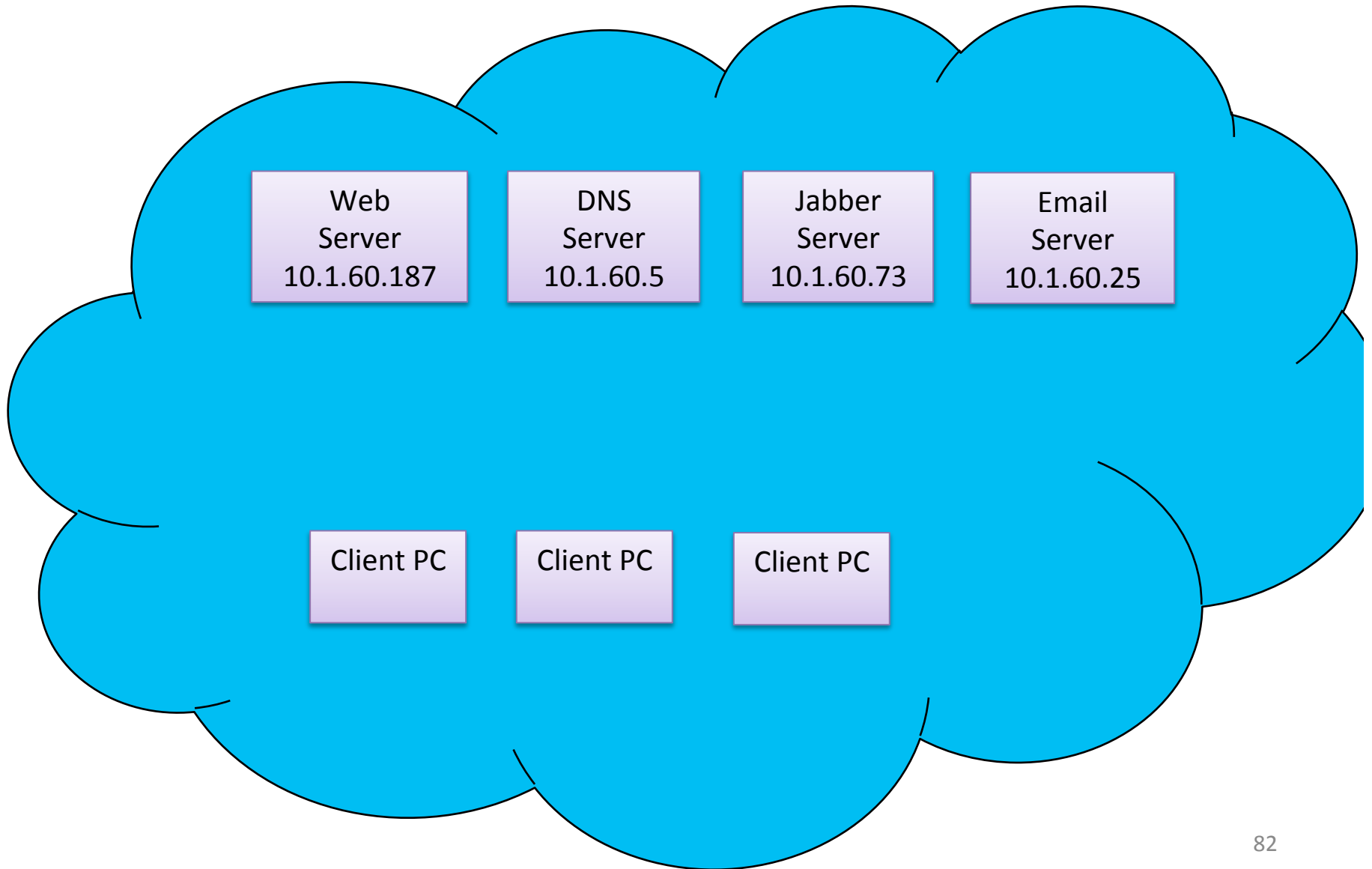
SiLK lab key information

- Bad guys: 10.2.x.x
- Good guys: 10.1.60.x
 - Each team has unique IP range
 - Servers have designated IP addresses

iSiLK: Lab ITOC Competition



iSilk: Lab10.1.60.x Team Network



iSiLK: Lab Port Scan & Netbus

srcip	dstip	sport	dstport	proto	packets	bytes	flags	stime	dur
10.2.190.249	10.1.60.5	36610	46	6	1	44	S	2009/04/22T18:06:08...	0.002
10.2.190.249	10.1.60.5	36610	47	6	1	44	S	2009/04/22T18:05:17...	0.025
10.2.190.249	10.1.60.5	36610	48	6	1	44	S	2009/04/22T18:07:58...	0.004
10.2.190.249	10.1.60.5	36610	49	6	1	44	S	2009/04/22T18:08:06...	0.018
10.2.190.249	10.1.60.5	36610	50	6	1	44	S	2009/04/22T18:07:59...	0.011
10.2.190.249	10.1.60.5	36610	51	6	1	44	S	2009/04/22T18:07:58...	0.004
10.2.190.249	10.1.60.5	36610	52	6	1	44	S	2009/04/22T18:07:37...	0.024
10.2.190.249	10.1.60.5	36610	53	6	1	44	S	2009/04/22T18:05:08...	0.003
10.2.190.249	10.1.60.5	36610	54	6	1	44	S	2009/04/22T18:07:43...	0.014
10.2.190.249	10.1.60.5	36610	55	6	1	44	S	2009/04/22T18:08:12...	0.028
10.2.190.249	10.1.60.5	36610	56	6	1	44	S	2009/04/22T18:06:47...	0.023
10.2.190.249	10.1.60.5	36610	57	6	1	44	S	2009/04/22T18:07:43...	0.012
10.2.190.249	10.1.60.5	36610	58	6	1	44	S	2009/04/22T18:08:09...	0.025
10.2.190.249	10.1.60.5	36610	59	6	1	44	S	2009/04/22T18:07:26...	0.024
10.2.190.249	10.1.60.5	36610	60	6	1	44	S	2009/04/22T18:07:09...	0.025
10.2.190.249	10.1.60.5	36610	61	6	1	44	S	2009/04/22T18:07:07...	0.033
10.2.190.249	10.1.60.5	36610	62	6	1	44	S	2009/04/22T18:07:40...	0.038
10.2.190.249	10.1.60.5	36610	63	6	1	44	S	2009/04/22T18:06:46	0.003

srcip	dstip	sport	dstport	proto	packets	bytes	flags
10.1.70.244	10.2.9.205	4668	12345	6	3	144	S
10.1.50.5	10.2.9.205	4437	12345	6	4	168	FS A
10.1.50.5	10.2.9.205	4436	12345	6	4	181	S PA
10.1.70.244	10.2.9.205	1060	12345	6	4	168	FS A
10.1.70.244	10.2.9.205	1068	12345	6	4	168	FS A
10.1.70.244	10.2.9.205	1072	12345	6	4	168	FS A
10.1.70.244	10.2.9.205	1076	12345	6	4	168	FS A
10.1.70.244	10.2.9.205	1058	12345	6	502	42,437	S PA
10.1.70.244	10.2.9.205	4690	12345	6	94	4,927	S PA
10.1.70.244	10.2.9.205	4683	12345	6	87	4,660	S PA
10.1.50.5	10.2.9.205	4228	12345	6	111	5,826	S PA
10.1.50.5	10.2.9.205	1964	12345	6	5	221	FS PA
10.1.50.5	10.2.9.205	1977	12345	6	4	168	FS A

Argus

Argus: Commands

- Converting Pcap to netflow
Argus -r packet.pcap -w packet.argus
- Reading a netflow file
ra -r netflowfile

Argus: ra Core Clients

Client	Info
ra	Basic argus record reading and printing and storing
rabins	Align argus data to time based bins.
racluster	Argus data aggregation
racount	Tally various aspects of an argus stream.
radium	Argus record collection and distribution.
ranonymize	Anonymization of argus data.
rasort	Argus file or stream sorting.
rasplit	Splits and distributes argus data streams, writes the data to files.

Argus: ra Output Fields

Output field	Info
time	When the argus server is running in default mode, ra reports the transaction starting time. When the server is in DETAIL mode, the transaction ending time is reported.
mac	mac.addr is an optional field, specified using the -m flag. mac.addr represents the first source and destination MAC addresses seen for a particular transaction. These addresses are paired with the host.port fields, so the direction indicator is needed to distinguish between the source and destination MAC addresses.
proto	see next slides. 1 st field is protocol specific, 2 nd is upper protocol used

Argus: ra Protocol Field

Proto	The proto indicator consists of two fields. The first is protocol specific and the designations are:
m	MPLS encapsulated flow
q	802.1Q encapsulated flow
p	PPP over Ethernet encapsulated flow
E	Multiple encapsulations/tags
s	Src TCP packet retransmissions
d	Dst TCP packet retransmissions
*	Both Src and Dst TCP retransmissions
i	Src TCP packets out of order
r	Dst TCP packets out of order
&	Both Src and Dst packet out of order

Argus: ra Protocol Field

Proto	Info
S	Src TCP Window Closure
D	Dst TCP Window Closure
@	Both Src and Dst Window Closure
x	Src TCP Explicit Congestion Notification
t	Dst TCP ECN
E	Both Src and Dst ECN
M	Multiple physical layer paths
I	ICMP event mapped to this flow
S	IP option Strict Source Route
L	IP option Loose Source Route

Argus: ra Protocol Field

Proto	Info
T	IP option Time Stamp
+	IP option Security
R	IP option Record Route
A	IP option Router Alert
O	multiple IP options set
E	unknown IP options set
F	Fragments seen
f	Partial Fragment
V	fragment overlap seen

Argus: ra Direction Field

Field	Info
direction	The dir field will have the direction of the transaction, as can be best determined from the datum, and is used to indicate which hosts are transmitting. For TCP, the dir field indicates the actual source of the TCP connection, and the center character indicating the state of the transaction.
-	Transaction was NORMAL
	Transaction was RESET
o	Transaction TIMED OUT.
?	Direction of transaction is unknown

Argus: ra Output Fields

Output Field	Info
host	The host field is protocol dependent, and for all protocols will contain the IP address/name. For TCP and UDP, the field will also contain the port number/name, separated by a period.
count	An optional field, specified using the -c option. There are 4 fields that are produced. The first 2 are the packet counts and the last 2 are the byte counts for the specific transaction. The fields are paired with the previous host fields, and represent the packets transmitted by the respective host.
status	Indicates the principle status for the transaction report, and is protocol dependent. For all the protocols, except ICMP, this field reports on the basic state of a transaction. See next slide for more info

Argus: ra Status Field

Output field	Info
REQ INT (requested initial)	This indicates that this is the initial status report for a transaction and is seen only when the argus-server is in DETAIL mode. For TCP connections this is REQ, indicating that a connection is being requested. For the connectionless protocols, such as UDP, this is INT.
ACC (accepted)	This indicates that a request/response condition has occurred, and that a transaction has been detected between two hosts. For TCP, this indicates that a connection request has been answered, and the connection will be accepted. This is only seen when the argus-server is in DETAIL mode. For the connectionless protocols, this state indicates that there has been a single packet exchange between two hosts, and could qualify as a request/response transaction.

Argus: ra status field

Output field	Info
EST CON (established connected)	This record type indicates that the reported transaction is active, and has been established or is continuing. This should be interpreted as a status report of a currently active transaction. For TCP, the EST status is only seen in DETAIL mode, and indicates that the three way handshake has been completed for a connection.
CLO (closed)	TCP specific, this record type indicates that the TCP connection has closed normally.
TIM (timeout)	Activity was not seen relating to this transaction, during the argus server's timeout period for this protocol. This status is seen only when there were packets recorded since the last report for this transaction.
ICMP	See ra man file for details on ICMP status

Argus: racount

racount	Sum
records	574583
Total packets	5474657
Source packets	1363063
Destination packets	4111594
Total bytes	3789669692
Source bytes	391088052
Destination bytes	3398581640

- ▶ Data created with `racount -r itoc.argus`
- ▶ `man racount`

Bro

Intrusion Detection System

Bro: Summary

- Behavior and signature-based IDS, framework
- Version 2.0 is much more streamlined than 1.5
- Conn.log is the “netflow” of the Bro outputs
- Bro also supplies detailed data on called scripts/
policy (2.0/1.5)
 - HTTP traffic
 - DNS
 - SSH
 - Strange behavior

Bro: 1.5 vs. 2.0

- Works much better out of the box
 - 64 bit packages
 - Security onion VM
 - Compiling from source very quick
 - No site config file needed for analysis
 - Custom Bro 1.5 code won't port well
- Bro 1.5 scripts were in /policy now in /scripts
- Difference in conn.log (the netflow of Bro)

Bro: Conn.log 1.5

- Start
- Duration
- Local IP
- Remote IP
- Service
- Local Port
- Remote port
- Protocol
- Org bytes sent
- Res byte sent
- State
- Flags
- Tag

Bro: Reading pcap

- **Bro -r filename.pcap**
 - Default analysis
- **Bro -r filename.pcap local**
 - More detection options
- **Bro -r filename.pcap protocols/ssl/validate-certs**
 - Base analysis and SSL cert validation

Bro: Conn.log 2

Bro Fields	Data Sample UDP DNS
Time Stamp	1240301181.402707
Unique Connection Identifier	2KEu8Pvr2Oi
Id.orig_h	10.1.90.5
Id.orig_p	1209
Id.resp_h	10.2.20.52
Id.resp.p	53
proto	UDP
service	DNS
duration	0.000623
Orig bytes	47
Resp bytes	111

Bro: Conn.log 2

Bro Fields	Data Sample UDP DNS
Conn_state	SF
Local_orig	-
Missed_bytes	0
history	Dd
Orig_pkts	1
Orig_ip_bytes	75
resp_pkts	1
Resp_ip_bytes	139
Tunnel_parents	(empty)

Bro: TCP Flags

S0	Connection attempt seen, no reply.
S1	Connection established, not terminated.
SF	Normal establishment and termination. Note that this is the same symbol as for state S1. You can tell the two apart because for S1 there will not be any byte counts in the summary, while for SF there will be.
REJ	Connection attempt rejected.
S2	Connection established and close attempt by originator seen (but no reply from responder).
S3	Connection established and close attempt by responder seen (but no reply from originator).

Bro: TCP Flags

RSTO	Connection established, originator aborted (sent a RST).
RSTR	Established, responder aborted.
RSTOSO	Originator sent a SYN followed by a RST, we never saw a SYN-ACK from the responder.
RSTRH	Responder sent a SYN ACK followed by a RST, we never saw a SYN from the (purported) originator.
SH	Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the responder (hence the connection was "half" open).
SHR	Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the originator.
OTH	No SYN seen, just midstream traffic (a "partial connection" that was not later closed).

Bro: Conn.log Additional Info

Field	explanation
Local_orig	T – connection originated locally F- connection originated remotely Blank - bro::site::local_nets variable is undefined
Missed_bytes	Number of bytes missed in content gaps (if missed_bytes > 0 protocol analysis will fail)
Tunnel_parents	If this connection was over a tunnel, indicate the *uid* values for any encapsulating parent connections used over the lifetime of this inner connection.

Bro: Conn.log History Field

Field value	info
s	a SYN w/o the ACK bit set
h	a SYN+ACK ("handshake")
a	a pure ACK
d	packet with payload ("data")
f	packet with FIN bit set
r	packet with RST bit set
c	packet with a bad checksum
i	inconsistent packet (e.g. SYN+RST bits both set)
lowercase	from the responder,
uppercase	from the originator,

Bro: Additional Logs

- Weird.log
 - Contains unusual/exceptional activity that can indicate malformed connections, traffic that doesn't conform to a particular protocol, malfunctioning/misconfigured hardware, or even an attacker attempting to avoid/confuse a sensor.
 - Without context, it's hard to judge whether this category of activity is interesting and so that is left up to the user to configure.
- Notice.log
 - Identifies specific activity that Bro recognizes as potentially interesting, odd, or bad. In Bro-speak, such activity is called a "notice"

Analytics

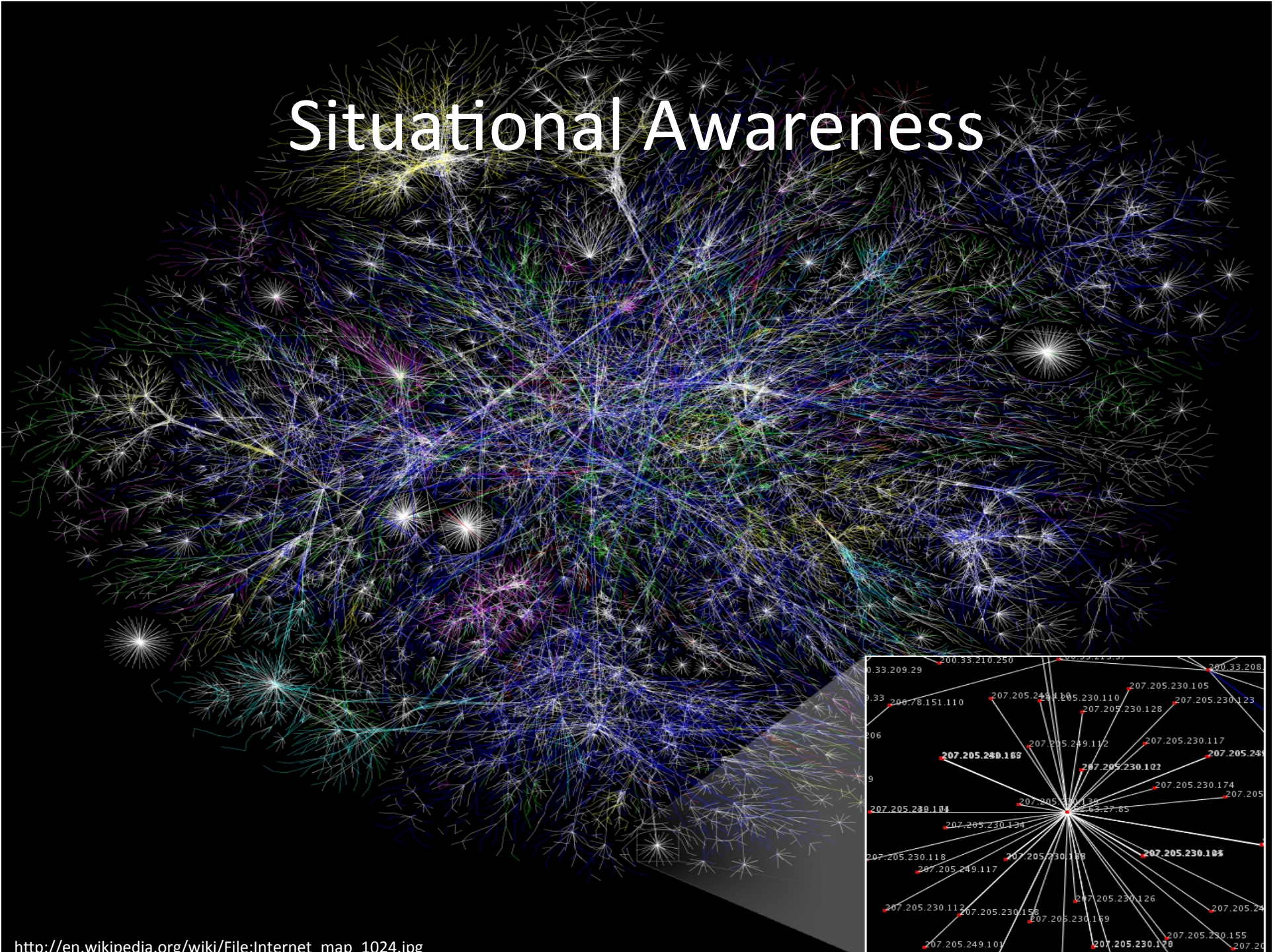
Outline

- Introduction
 - What is Netflow?
 - Sensor Location
 - Sampling
- Tools
 - YAF
 - SiLK
 - iSiLK
 - Argus
 - Bro
- **Analytics**
 - Situational Awareness Analytics
 - Hunting Analytics
 - Data Fusion Analytics
- Wrap Up

Analytics Outline

- Situational awareness analytics
- Hunting analytics
- Fusing netflow with other data
- Sensor visibility differences

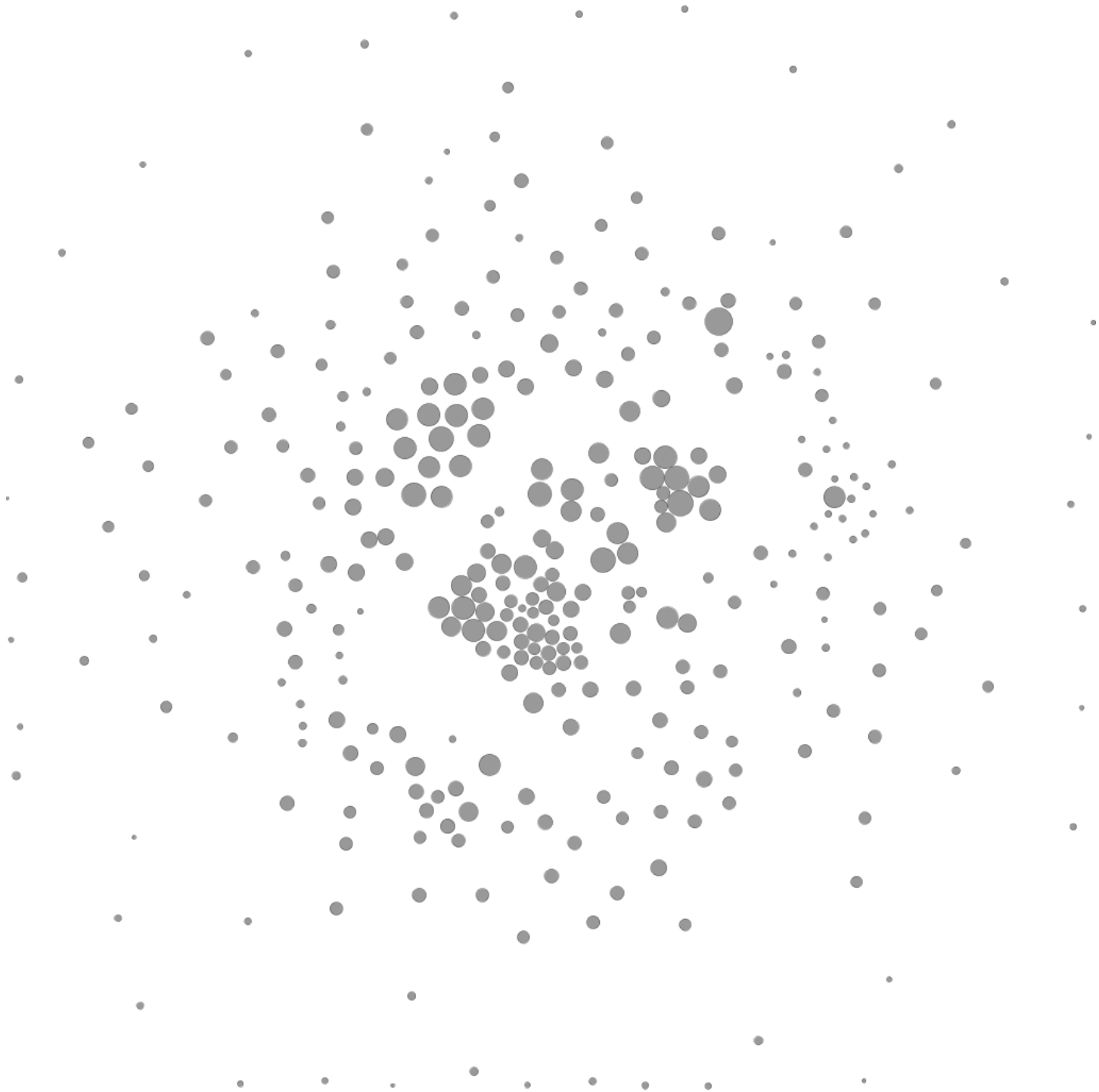
Situational Awareness



Network Profiling

- Useful for inventory server assets
- See the CMU SEI CERT technical report *Network Profiling Using Flow*





Network Profiling: Features

- Look at IP address and hostname
- Top 3 src/dst ports by percentage of traffic and byte count
- Average for number of flows, duration, byte/packet count
- Ratio of sent vs received for above
- Other layer 3 and 4 protocols in use

- Feature space is very rich. Choose wisely.

Host Clustering

- Data overload is a problem for network analysts
- Goal: allow analysts to spend more time on things which require expert human attention
- Inform situational awareness, e.g.,
 - What does “normal” look like on a given network?
 - What do normal host behaviors look like?
 - Which hosts have changed recently?
 - Which hosts are using a particular protocol differently?

Clustering informs questions like

- What do “normal” behaviors look like? What’s abnormal?
 - What web servers also act as web clients?
 - Which hosts have similar users?
 - Which hosts have had significant changes in behavior relative to themselves?
 - Which hosts are acting differently from their peers with respect to a particular protocol?
 - How many peers does a host have?
 - What cliques are present in the network?

Example Network Diagram

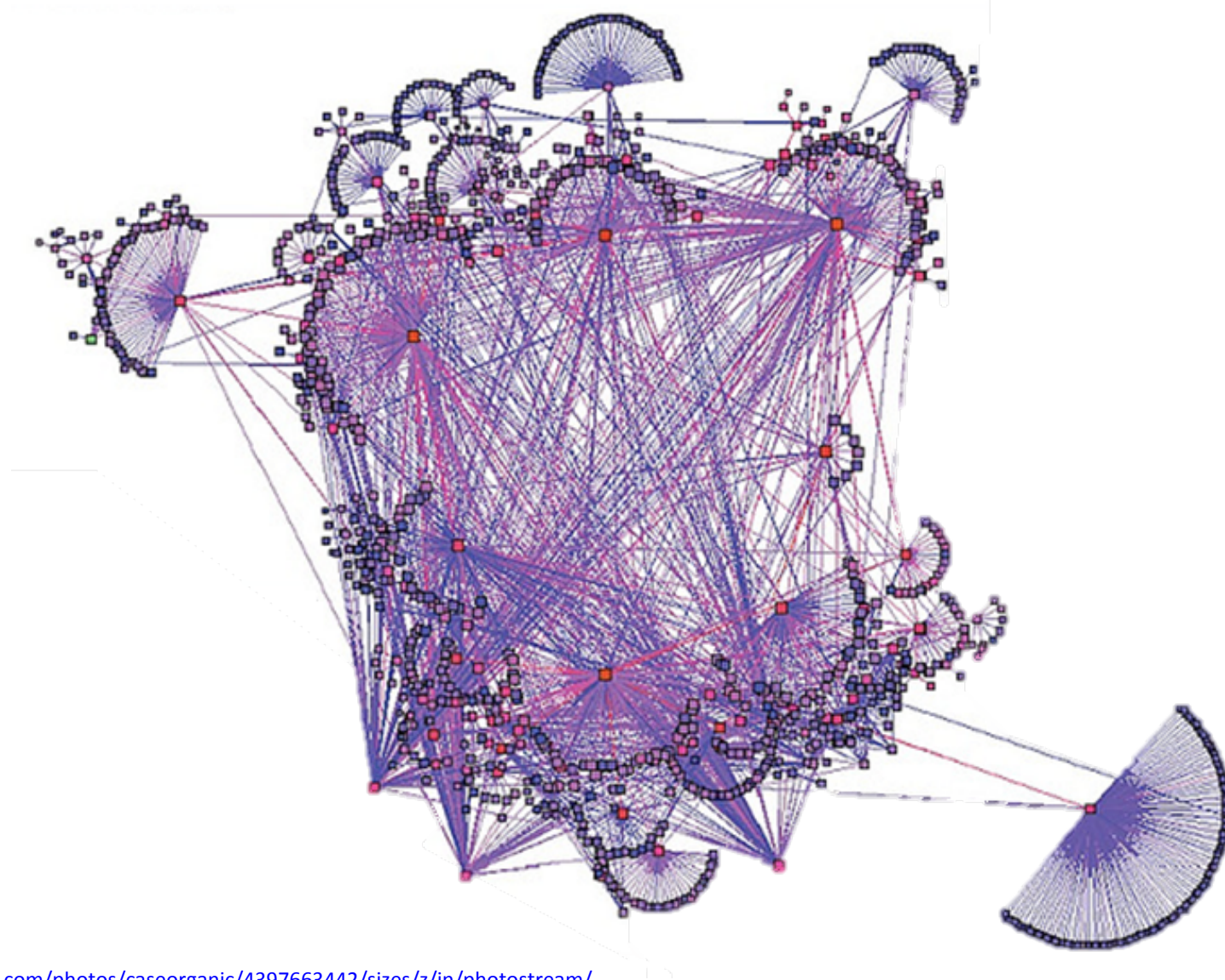
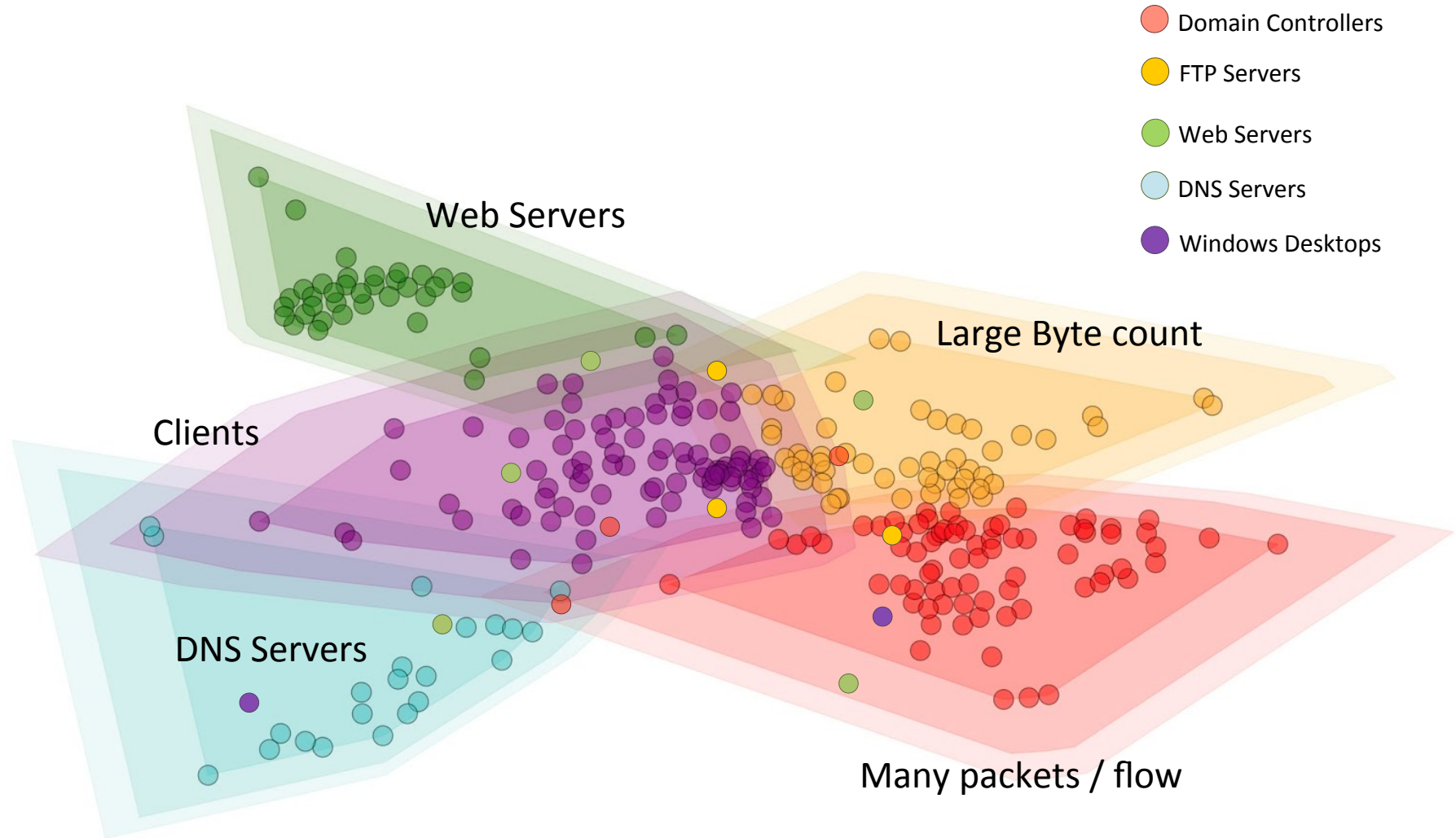


Diagram with Clustering



Operational Issues with Clustering

- Correct selection of features and distances
 - We want to differentiate groups of interest
- Selecting the number of clusters on an arbitrary network
- Need an external frame of reference
 - External labels

Darkspace Monitoring

- Darkspace: routable IP address space with no hosts attached
- By definition, traffic to darkspace is unsolicited
- Darkspace traffic contains
 - Misconfigurations
 - Reconnaissance
 - Backscatter (from scanning or DoS attacks)
 - Automated worm/virus spreading

Darkspace Monitoring

- If you've got darkspace on your network (the DoD has plenty), keep an eye on it
- *Entropy in IP Darkspace Data* by Tanja Zseby, FloCon 2012
 - Work tracks distribution of entropy in {sip, dip, sport, dport} and characterizes common events

Volumetric Analysis

- Top sources and sinks
- Ratio of sent/received data
- Dips contacted per sip
- Examine rates of change
- Consider ways to partition the data
 - Sip
 - Dip
 - Server (individual server or server type)
 - User
 - Protocol (layer: 2, 3, 4, or 7)

Unproxied Connections

- Connections bypassing the proxy are not being inspected or logged
- MITRE's firewall has open ports:
 - 21 FTP
 - 22 SSH
 - 23 Telnet
 - 119 NNTP
 - 389 LDAP
 - 554 RTSP
 - 636 LDAP
 - 2401 CVS
 - 707 Real Audio

Anomalies in Netflow

- Good for finding things that are loud
 - Outages and misconfigurations
 - Denial of Service
 - Port scanning
 - Worms
- Mechanisms for discovering this behavior are standard in many commercial tools

Anomalies in Netflow

- See *Mining Anomalies Using Traffic Feature Distributions* by Lakhina, et al.
- Look for change in the distribution of packet header fields

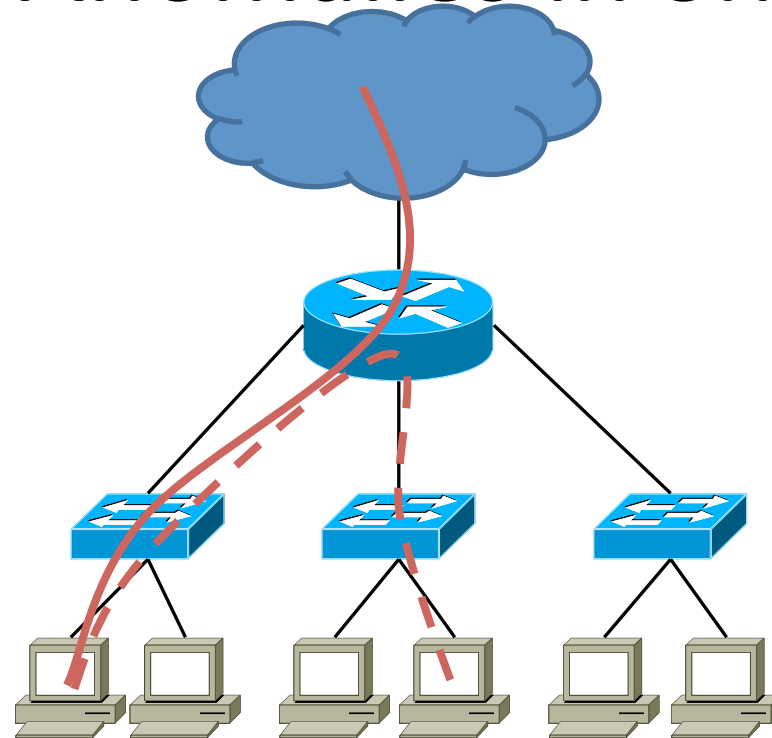
Anomalies in Netflow

- Protocols of interest
 - DNS
 - NTP
 - SMB/CIFS
 - SNMP
 - LDAP
- They tend to be regular
 - As opposed to protocols like HTTP

Anomalous DNS Discovery

- Use case: tunnel detection (IP over DNS, etc.)
- See *Detection of DNS Anomalies using Flow Data Analysis* by Karasaridis, et al.
- Separate DNS packets into requests, responses and unknown
- Calculate histograms over packet size per hour of day
- Look for changes in the histogram distribution
- Other analytics: HTTP flows not preceded by a DNS request

Anomalies in SMB



Anomalies in SMB

- Use case: lateral movement between Windows workstations
- TTP: *net use*, *xcopy*, and *at* commands
- The commands are multiplexed over a single TCP session
 - Result: large duration and byte count
- Caveat: You need sensors placed to see this

Hunting Analytics

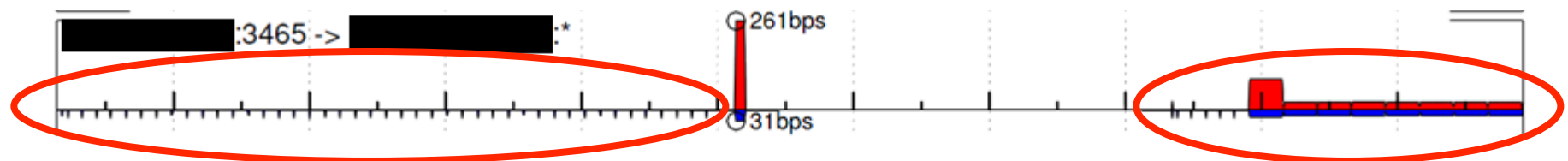


Beacon Detection

- Beaconing
 - Periodic behavior in the network
- Many kinds of malicious software beacon
 - Remote Access Tools (RATs)
 - Bots
- Lots of things beacon
 - We do not care about most of them
- Malicious activity is often beaconing paired with other network activity
- A beacon detector can be married with other signature-based tools to detect specific threats

Beacon Detection: Poison Ivy

- Poison Ivy is characterized by
 - A series of beacons during the “phone home” phase
 - Three packets with SYN flag set
 - One long flow with a roughly symmetric byte count during the actual desktop session
 - Some versions change ports based on system date



Beaconing Detector

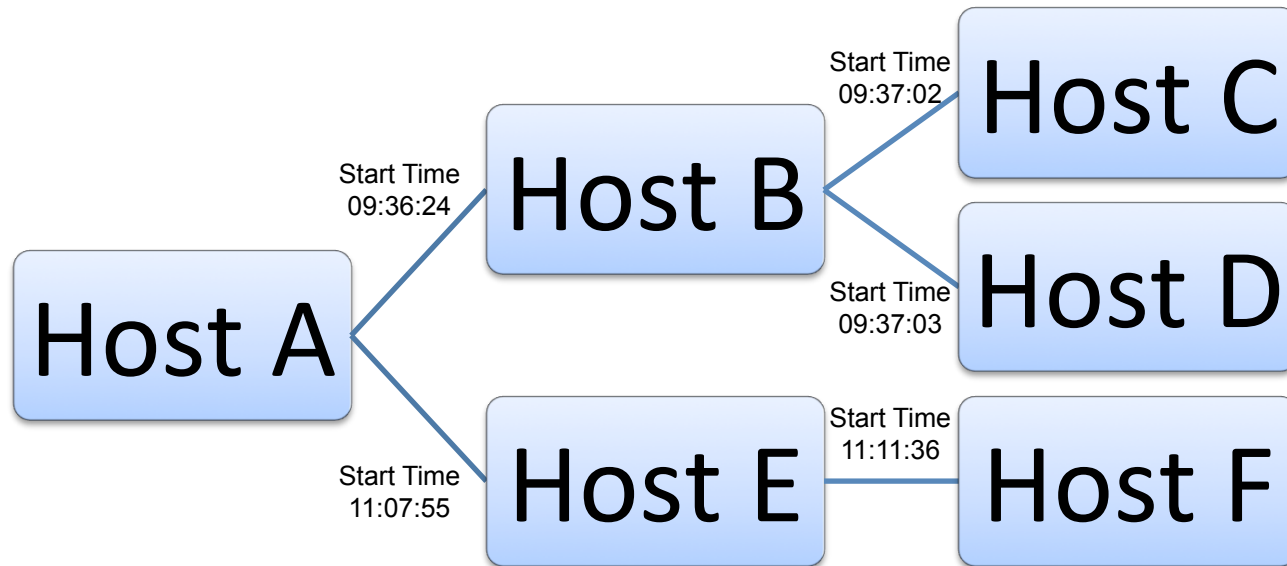
**Interactive
Session Detector**

Beacon Detection: Limitations

- False positives: beacons that are uninteresting
 - Web 2.0 – AJAX
 - Periodic software updates (Windows, browsers, etc.)
 - Network Time Protocol (NTP) updates
- False negatives: interesting beacons that are missed
 - Noise in the channel
 - Intentional evasion (e.g., randomness)
 - Steganography and covert channels
- Finer grained methods (beyond netflow)
 - Individual packet headers
 - Deep packet inspection
 - Examine attributes besides time

Chaining

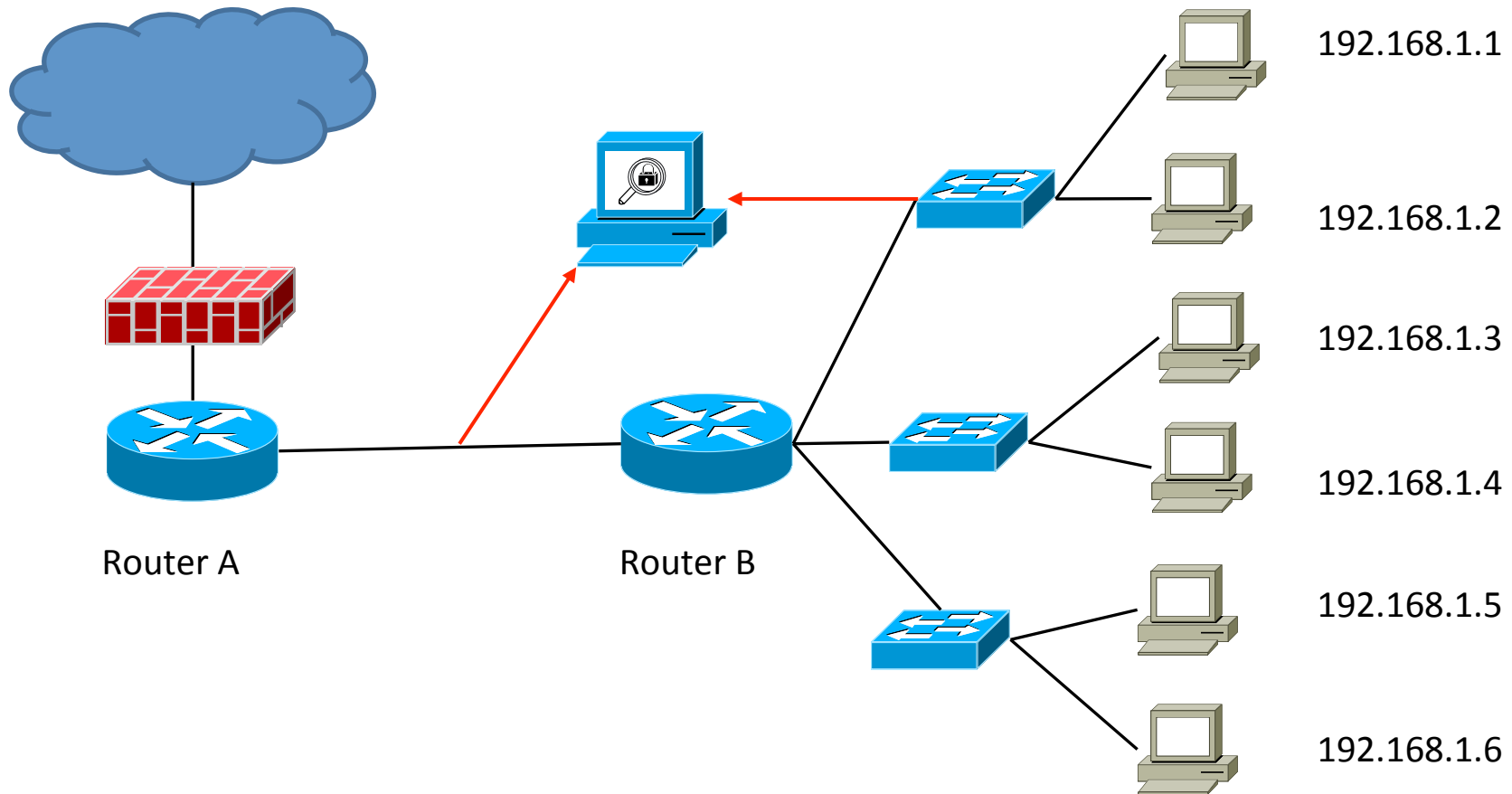
- Temporally correlated netflows that satisfy a common link predicate



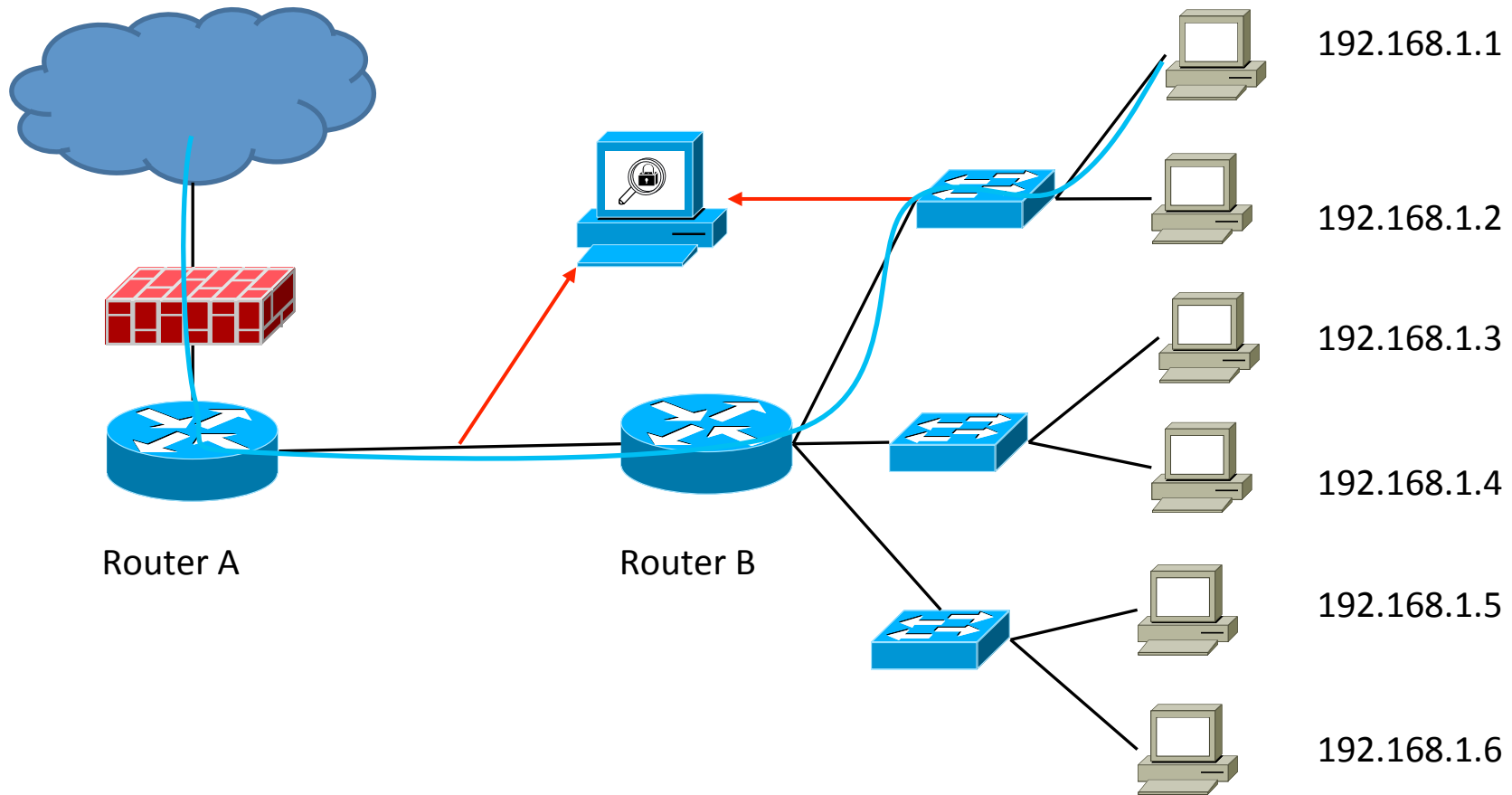
Chaining

- Example predicates
 - dport equals 3389 or 22
 - dport equals 137, 138, 139 or 445
- Use cases
 - Worm detection
 - Lateral propagation detection
 - Network scanner detection
 - Root cause analysis
 - Attribution
 - Proxy/NAT detection
 - Tunnel detection

Ghost In The Machine

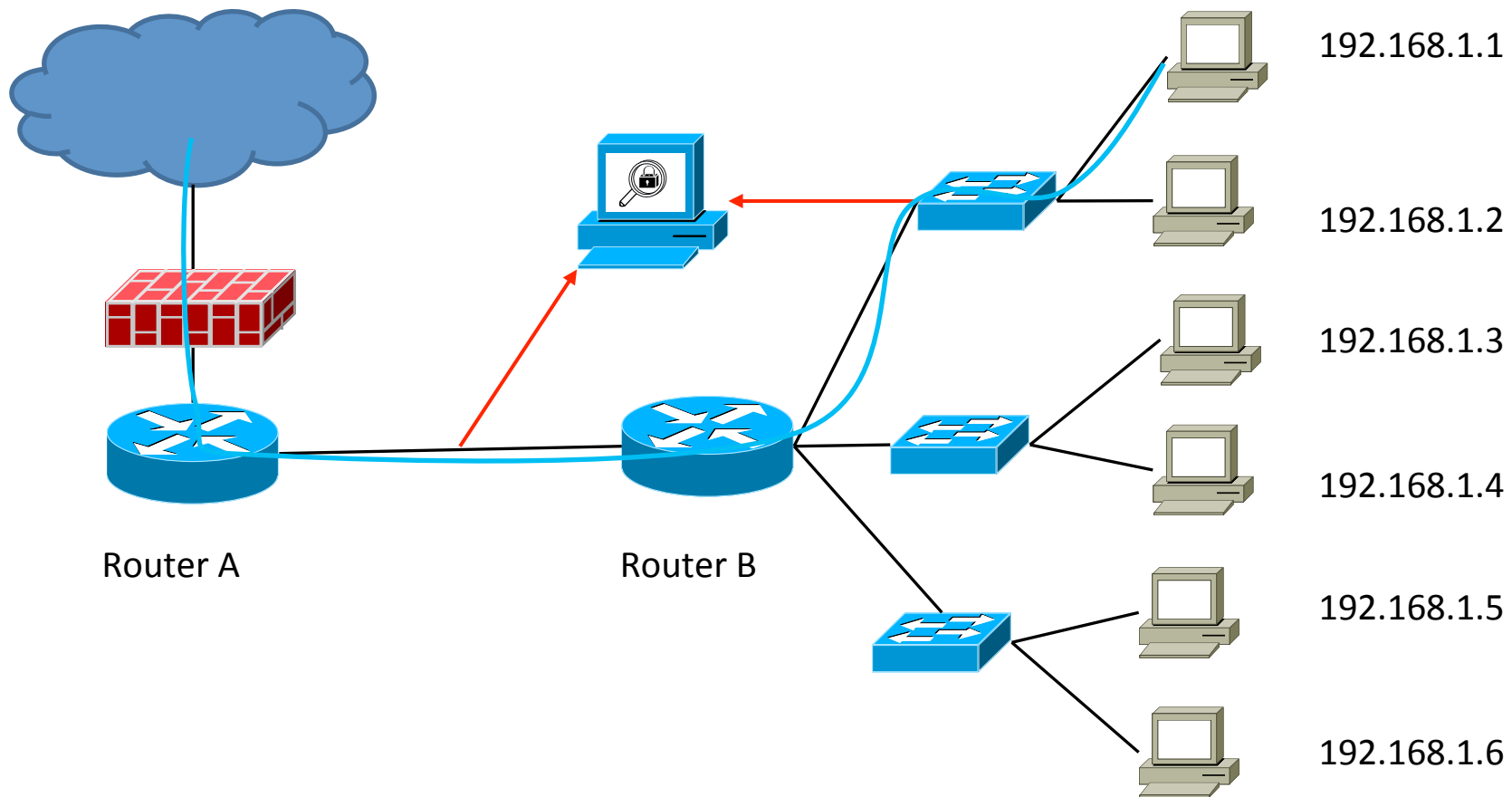


Ghost In The Machine



Ghost In The Machine

- Looking for ~~love~~ flows in all the wrong places



Ghost In The Machine

- What are some reasons you might see this?



Ghost In The Machine

- Maybe you're seeing a C2 or exfil channel
 - *SQL Injection to MIPS Overflows: Rooting SOHO Routers* presented at DefCon 2012
 - SQL injection attack -> initial access
 - Input sanitization vulnerability -> arbitrary file access
 - Buffer overflow -> arbitrary code execution
 - Supply chain risks
 - FBI investigation of counterfeit Cisco routers from PRC (2008)

Ghost In The Machine

InfoWorld Home / Security / News / Hackers reveal critical vulnerabilities in Huawei...

JULY 30, 2012

Hackers reveal critical vulnerabilities in Huawei routers at Defcon

The vulnerabilities -- a session hijack, a heap overflow, and a stack overflow -- were found in the firmware of Huawei AR18 and AR29 series routers and could be exploited to take control of the devices over the Internet

Security researchers disclosed critical vulnerabilities in routers from Chinese networking and telecommunication companies at Defcon 19 on Sunday.

The vulnerabilities were found in the firmware of Huawei AR18 and AR29 series routers. Two researchers

During the Defcon talk, which Lindner gave together with Reccurly Labs security consultant Gregor Kopf, the researchers pointed out that there are over 10,000 calls in the firmware's code to sprintf, a function that's known to be insecure.

Ghost In The Machine

- Detection
 - Diff flows seen at sensors
 - Computationally expensive
 - Hostflow is helpful for visibility

Data Fusion Analytics

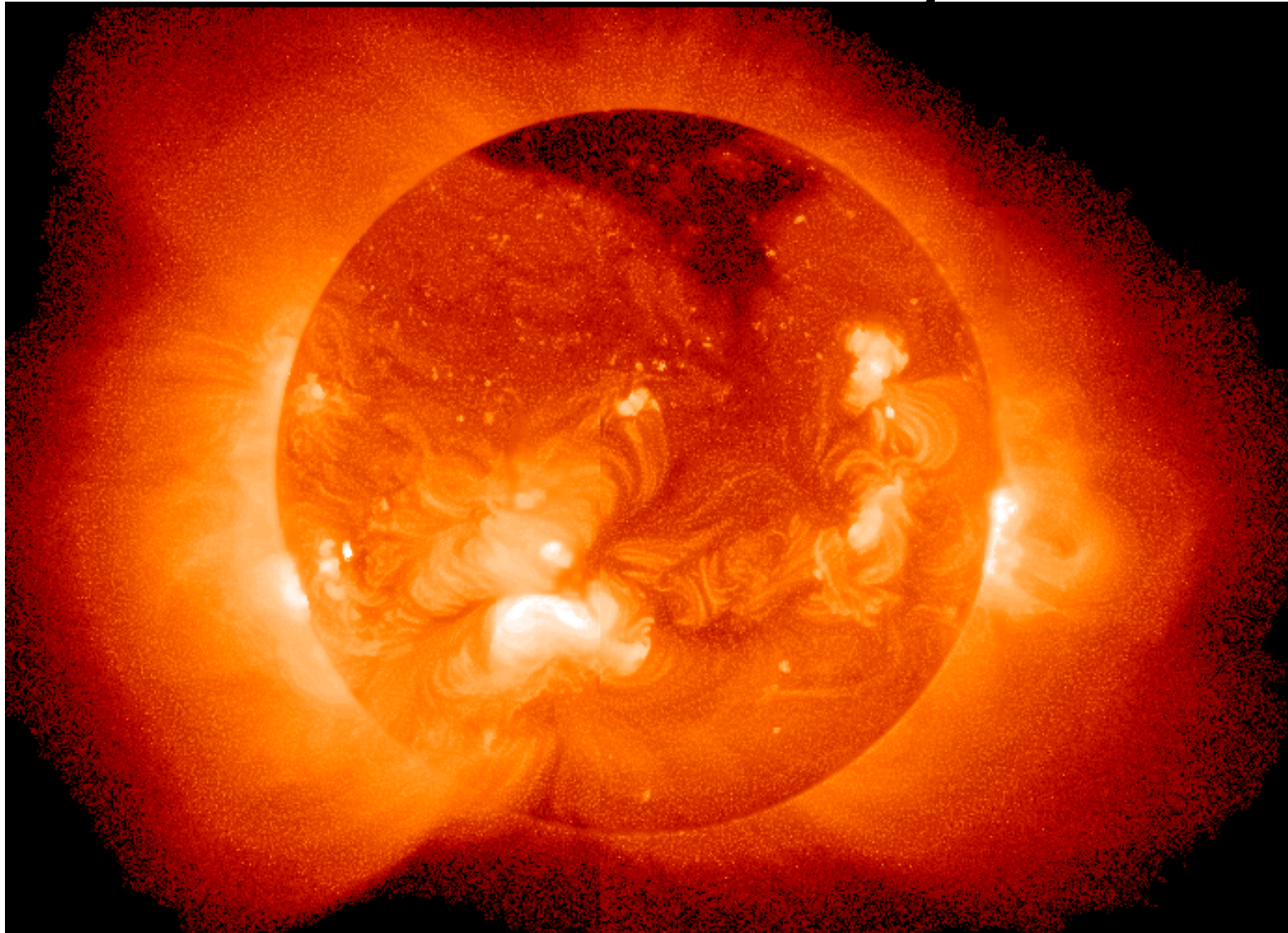


Image courtesy of NASA Goddard Laboratory for Atmospheres

Data Fusion Analytics

- Enrich netflow with other data
 - Additional network-based data
 - Host-based data
 - Non-computer data
- What are some examples of each?

Data Fusion Analytics: Hostflow

- Client-to-client traffic baseline
 - Lateral movement detection
- Blacklisted domain access when off network
- Diff hostflows and netflows
- Link users to netflows, pivot
- Correlate processes to network sessions
 - Discover processes with abnormal port usage

Wrap Up

Outline

- Introduction
 - What is Netflow?
 - Sensor Location
 - Sampling
- Tools
 - YAF
 - SiLK
 - iSiLK
 - Argus
 - Bro
- Analytics
 - Situational Awareness Analytics
 - Hunting Analytics
 - Data Fusion Analytics
- **Wrap Up**

Analytic Platforms

- Tool-specific data stores (e.g., SiLK)
 - Difficult to impossible to fuse with other data
- Traditional SIEMS (e.g., ArcSight)
 - Good for live queries
 - Issues scaling and querying retrospectively
- MapReduce
 - Batch-oriented
 - Cloudera Impala provides real-time query capability
 - Splunk provides real-time analysis on top of MR

Open Problems

- Analytic tradecraft
 - Data fusion
- Deep diving into pcap from flow
- Sensor deployment strategies
- Visualization
- Systems engineering for distributed analysis

Resources

- Flocon
 - Annual conference organized by CMU SEI CERT
 - <http://www.cert.org/flocon/>

Wrap Up

- Recap
 - What is netflow?
 - Sensor architecture
 - Tools
 - Analytic tradecraft
 - Situational awareness analytics
 - Hunting analytics
- What worked well? What didn't?

Further reading

- IDS evasion techniques
 - http://insecure.org/stf/secnet_ids/secnet_ids.html
 - <http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Caswell.pdf>
 - <http://www.securitytube.net/video/5280>