

Monitoring/Sniffing

- Normally, ethernet cards and other broadcast NICs receive all traffic on the subnet.
- Unless the card is in **promiscuous mode**, messages not addressed to the NICs MAC address are filtered out in hardware.
 - on a switched ethernet, with one card per interface to the switch, messages for other cards aren't heard.
- Listening to all messages received by the card in violation of your user agreement is **sniffing**.
- Listening to all messages received by the card as an administrator is called **monitoring**.
- Recording instances of a daemon's execution is called **logging**.

Sniffing Tools

- Most tools are based on TCPdump and the libpcap (packet capture library) extensions to unix.
- Other tools are snort and desniff; for windows there are many, WinDump is a direct port of TCPdump.
- On many versions of unix, the libpcap extension is not installed (or enabled) by default. For example, FreeBSD requires recompiling.
- If you don't need it, keep it off the system. Why load a system up with tools a hacker may exploit later.

Sniffing Attacks

- The simplest is to capture the password after a telnet login.
- One defense is to use an ssh daemon instead of telnet or rsh. There is no reason to run telnet or the rlogin programs on a system.
- With secure shell (ssh), the two computers negotiate a symmetric key for use during the session.
- Key exchange is achieved using public/private key pairs.
- But there is no trusted/certificate authority, so the initial exchange of keys is subject to a man-in-the-middle attack.
- Recently, the designer of the desniff program wrote a plug-in to his program to specifically look for the initial exchange.

Using TCPdump

- to see a list of interfaces on a unix machine:

- `ifconfig -a`

```
eth0 Link encap:Ethernet HWaddr 00:C0:4F:A1:46:0E
      inet addr:128.119.245.66 Bcast:128.119.247.255 Mask:255.255.248.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:344734848 errors:0 dropped:0 overruns:33473 frame:0
      TX packets:251651044 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      Interrupt:19 Base address:0xdc00
```

```
lo Link encap:Local Loopback
    inet addr:127.0.0.1 Mask:255.0.0.0
    UP LOOPBACK RUNNING MTU:3924 Metric:1
    RX packets:3756831 errors:0 dropped:0 overruns:0 frame:0
    TX packets:3756831 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
```

TCPdump

```
tcpdump -i eth0 tcp
```

```
08:23:10.367677 gslacks.cs.umass.edu.1072 >  
    horton.cse.ucsc.edu.22: P 18977  
31:1897751(20) ack 2681372961 win 8168 <nop,nop,timestamp  
    19745 653063287> (DF)  
[tos 0x5]  
08:23:10.532319 horton.cse.ucsc.edu.22 > gslacks.cs.umass.edu.  
    1072: P 1:45(44) ack 20 win 24616 <nop,nop,timestamp  
    653083760 19745> (DF)  
08:23:10.638393 gslacks.cs.umass.edu.1072 >  
    horton.cse.ucsc.edu.22: . ack 4  
5 win 8124 <nop,nop,timestamp 19748 653083760> (DF) [tos 0x5]
```

Other calls

```
tcpdump -i eth0 src biff.cs.umass.edu and tcp
```

```
tcpdump -i eth0 not dest blinky.cs.umass.edu and arp
```

More flags

- e print the link level header.
- n don't convert names using DNS (faster).
- s snarf *bytes* of data from the packet.
- t drop the timestamp from the output line.
- vv interpret as many fields as possible from the packet.
- x print each packet in hex.

On windows, ethereal (wireshark)...

(Untitled) - Ethereal

File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	128.119.1.86	64.236.24.137	TCP	1171 > http [FIN, ACK] Seq=0 Ack=0 win=64260 Len=0
2	0.000190	128.119.1.86	64.236.24.20	TCP	1170 > http [FIN, ACK] Seq=0 Ack=0 win=63478 Len=0
3	0.025697	64.236.24.20	128.119.1.86	TCP	http > 1170 [ACK] Seq=0 Ack=1 win=6520 Len=0
4	0.025755	64.236.24.137	128.119.1.86	TCP	http > 1171 [ACK] Seq=0 Ack=1 win=6432 Len=0
5	0.377641	00:15:62:44:51:Spanning-tree-		STP	Conf. Root = 32768/00:d0:63:4f:45:77 Cost = 23 Port = 0x8004
6	0.704520	128.119.1.33	128.119.1.127	NBNS	Name query NB OIT<1d>

Frame 1 (54 bytes on wire, 54 bytes captured)

- Ethernet II, Src: DellComp_10:e7:26 (00:b0:d0:10:e7:26), Dst: Cisco_4f:44:00 (00:d0:63:4f:44:00)
- Internet Protocol, Src: 128.119.1.86 (128.119.1.86), Dst: 64.236.24.137 (64.236.24.137)
- Transmission Control Protocol, Src Port: 1171 (1171), Dst Port: http (80), Seq: 0, Ack: 0, Len: 0

```
0000  00 d0 63 4f 44 00 00 b0 d0 10 e7 26 08 00 45 00  ..COD... ..&..E.
0010  00 28 0e 84 40 00 80 06 11 0a 80 77 01 56 40 ec  .(.@... ..w.v@.
0020  18 89 04 93 00 50 6a 7c e5 84 b3 33 f9 da 50 11  ....Pj| ...3..P.
0030  fb 04 d7 99 00 00                                     .....

```

File: "C:\DOCUME~1\crispy\LOCALS~1\Temp\Ethernet.pcap" [P: 6 D: 6 M: 0 Drops: 0]

Snort

- What is it?
 - “The *de facto* standard for intrusion detection/prevention”
 - Snort is a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks.
 - Three primary uses
 - Packet sniffer
 - Packet logger
 - Intrusion Detection System
 - <http://www.snort.org>

Snort

- Features
 - Full tcpdump compatibility. Snort can create tcpdump data files
 - Full support for bpf (Berkeley packet filter) rules
 - Simple rules language
 - CVE compatible

Snort

- Useful options
 - -a Display relevant ARP packets
 - -b Log packets to a tcpdump data file
 - -d Dump application layer data
 - -e Display link layer headers
 - -i <if> Listen on interface <if>
 - -r <f> Read tcpdump data file <f>
 - -v Verbose
- Snort uses libpcap-like options
 - `snort -i eth0 -v dst 192.168.0.1 and src 10.1.4.2`
 - `snort -i eth0 -v udp dst 192.168.0.1`

Snort Sample packet capture output

```
Initializing Network Interface...
Decoding Ethernet on interface hme0
-*> Snort! <*-Version 1.6 By Martin Roesch (roesch@clark.net,
www.clark.net/~roesch)

11/09-10:42:23.760220 128.119.175.17:41745 -> 128.119.166.5:21
TCP TTL:255 TOS:0x0 ID:61543 DF
**S***** Seq: 0x36A3E59 Ack: 0x0 Win: 0x2238TCP Options => MSS: 1460

11/09-10:42:23.761102 128.119.166.5:21 -> 128.119.175.17:41745
TCP TTL:59 TOS:0x0 ID:3606 DF
**S***A* Seq: 0xD9DB1A5 Ack: 0x36A3E5A Win: 0x832CTCP Options =>
MSS: 1460 00 00

11/09-10:42:23.761133 128.119.175.17:41745 -> 128.119.166.5:21
TCP TTL:255 TOS:0x0 ID:61544 DF
*****A* Seq: 0x36A3E5A Ack: 0xD9DB1A6 Win: 0x2238

11/09-10:42:23.829616 128.119.166.5:21 -> 128.119.175.17:41745
TCP TTL:59 TOS:0x10 ID:3620 DF
*****PA* Seq: 0xD9DB1A6 Ack: 0x36A3E5A Win: 0x832C
  32 32 30 20 65 6D 69 6C 79 2E 6F 69 74 2E 75 6D 220 emily.oit.um
  61 73 73 2E 65 64 75 20 46 54 50 20 73 65 72 76 ass.edu FTP serv
  65 72 20 28 56 65 72 73 69 6F 6E 20 77 75 2D 32 er (Version wu-2
  2E 34 2E 32 2D 61 63 61 64 65 6D 5B 42 45 54 41 .4.2-academ[BETA
  2D 31 38 5D 28 31 29 20 54 68 75 20 46 65 62 20 -18] (1) Thu Feb
  31 31 20 30 38 3A 31 34 3A 32 38 20 45 53 54 20 11 08:14:28 EST
  31 39 39 39 29 20 72 65 61 64 79 2E 0D 0A 1999) ready...
```

Snort - AddOns

- Snort has been extended both open source and commercial
- <http://www.snort.org/dl/>
 - BASE/ACID
 - Snortalog
 - Pigsentry
 - Snortsnarf
 - Demarc
 - Oinkmaster

Intrusion Detection

- Definitions
- A (very) brief history
- Host-based
- Network-based
- Signature vs. Anomaly Detection
- Snort
 - Rules
- IDS Evasion

Intrusion Detection: Definitions

- The process of monitoring computers and networks to analyze and detect signs of security threats
- “The goal of intrusion detection systems is to identify threats directed against an organization and then ensure the systems are hardened against those threats”
- **IDS**: Intrusion Detection System
- Why?
 - Prevention in the absence of detection and reaction is useless.
 - Basic detection alone is insufficient
 - Localization
 - Identification
 - Assessment
 - IDS is just another layer...

Intrusion Detection: A (very) brief history

- Auditing ~3000 b.c.
- Anderson Report 1972
- Denning, Neumann and IDES 1986
- The Tan Book 1988
- Shadow ~1996
- Snort 1999
- Intrusion detection is a young, immature technology
- SANS Heuristic Analysis system for Defensive Online Warfare (SHADOW).
- “A guide to understanding audit in trusted systems”
<http://downloads.securityfocus.com/library/tan.html>

Auditing

- An official examination and verification of accounts and records.
- Goals:
 - Assign and maintain accountability.
 - Reconstruct past events.
 - Discourage malicious or inappropriate use.
 - Assess level of damage to a compromised system.
 - Allow for faster recovery of compromised systems.
- Just another risk mitigation technique

Host Based Intrusion Detection

- Using system-level data to identify security threats:
 - Regular system audits
 - Syslog/Event Log analysis
 - Application log analysis (e.g. Apache, MySQL)
 - Cryptographic signature of binaries, logfiles, etc. (e.g., Tripwire)
 - Also called “target-based monitoring”
 - Virus scanning

Tripwire

- Written by Eugene Spafford and Gene Kim in 1992. Tripwire attempts to detect changes to critical system binaries and configuration files. Changes are detected by checking file sizes, running cryptographic hashes (signatures) of files and storing these in a database.
- Originally open source (version 1), then commercial (version 2), now the linux port (version 2) is again open source
- From sourceforge.net:
 - “Tripwire is a policy driven file system integrity checking tool that allows system administrators to verify the integrity of their data.”
- There are still some problems, but it helps

Network-Based Intrusion Detection

- Using network-level data to identify security threats
 - Generally using packet captures
 - Real-time versus post-processed
 - BPF/libpcap (Berkeley Packet Filter)
 - Abnormal stimuli, bad flags, spoofing
 - Signature-based (Snort, NFR, Shadow)
 - Router-filter logs
- Evaluation based on
 - false positives: alerts to situations that aren't attacks
 - false negatives: attacks that are missed.

Anomaly vs. Signature

- We can either
 - look for **anomalies** when compare against a baseline
 - or look for known threats in the form of **signatures** (a.k.a., **misuse** detection)

Anomaly Detection

- Anomaly detection is based on models of “normal” behavior.
 - We can build a historical database and perform statistical analysis.
 - Build a profile of users, or of each user:
 - What programs do you normally run when you log in?
 - Where do you normally log in from?
 - What is the normal inter-spacing of your keystrokes?
 - Where do you place your X-windows?

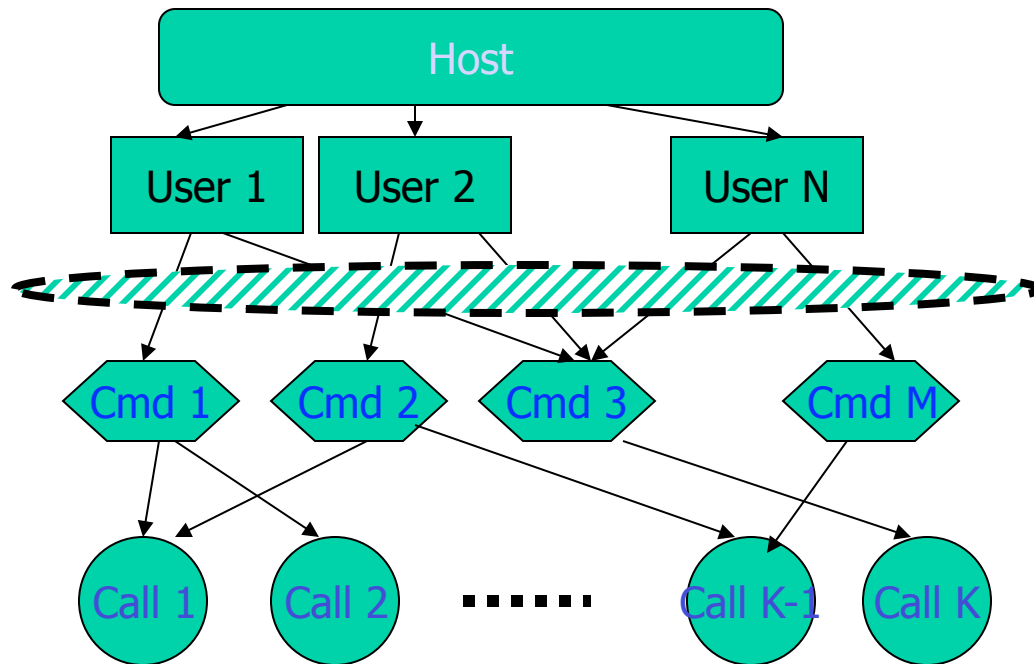
Anomaly Detection IDS

- Few implementations
- One filter detects many attack variants.
- Most new attacks are derivatives of known attacks, and will be caught by existing filters.
- System performance is increased due to inherent design efficiency.
- We already have too many false positives
- According to one vendor:

Anzen's anomaly detection filters model the implementation of a specific protocol and can identify network packets that do not meet a set of well-defined criteria. Any network packets that fall outside of this definition of normal will be flagged and will generate an alert.

Observing behavior

- Hierarchical set of options for establishing a baseline.



- Host level
- User level
- Operation level
- Command level
- System call level

Misuse detection (signatures)

- Signature-based IDS use a large rule-set to match against known attacks. Basically a pattern matching engine.
 - Currently in wide use
 - You just need a lot of **good** signatures
 - You can customize your own signatures
 - CVE helps
 - It takes time to update rule-sets
 - Many false positives

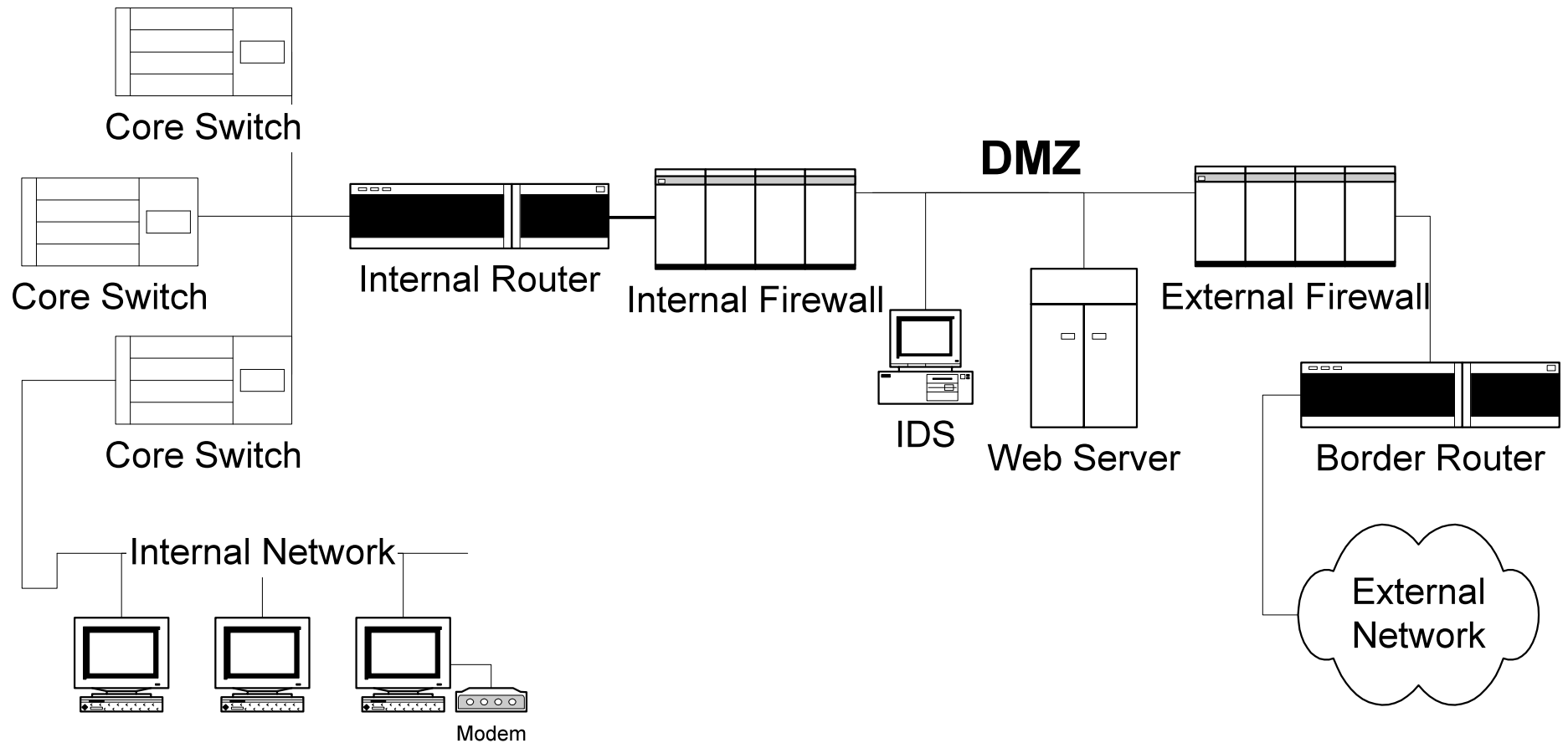
Snort

- A misuse/signature based scheme.
- Three primary uses
 - Packet sniffer
 - Packet logger
 - **Intrusion Detection System**
- We've covered snort before as a packet sniffer, but it is also an excellent freely available IDS...
 - <http://www.snort.org>
 - A good introduction is at:
 - <http://www.snort.org/lisapaper.txt>

Snort IDS

- Snort consists of three subsystems:
 - packet decoder (libpcap-based)
 - detection engine
 - logging and alerting subsystem
- We'll cover the detection engine
 - Rules form signatures
 - Modular detection elements are combined to form these signatures
 - Anomalous activity detection is possible stealth scans, OS fingerprinting, invalid ICMP codes, etc.
 - Rules system is very flexible, and creation of new rules is relatively simple

Intrusion Detection Placement



Snort: Sample IDS output

- Apr 12 01:56:21 ids snort: EXPLOIT sparc setuid 0: 218.19.15.17:544 → xxx.yyy.zzz.41:37987
- Apr 12 01:56:21 ids snort: EXPLOIT x86 NOOP: 23.91.17.7:544 → xxx.yyy.zzz.41:37987
- Apr 12 07:31:03 ids snort: ICMP Nmap2.36BETA or HPING2 Echo : 63.26.255.221 → xxx.yyy.zzz.34
- Apr 12 09:59:38 ids snort: RPC portmap request rstatd: 28.11.67.132:1033 → xxx.yyy.zzz.29:111
- Apr 12 13:20:05 ids snort: ICMP Nmap2.36BETA or HPING2 Echo : 12.13.1.67 → xxx.yyy.zzz.126
- Apr 12 14:13:22 ids snort: RPC portmap request rstatd: 134.1.5.12:3649 → xxx.yyy.zzz.29:111
- Apr 12 20:19:34 ids snort: BACKDOOR back orrifice attempt: 209.255.213.130:1304 → xxx.yyy.zzz.241:31337
- Apr 12 22:53:52 ids snort: DNS named iquery attempt: 209.126.168.231:4410 → xxx.yyy.zzz.23:53

Snort Rules

- Snort rules consist of two parts
 - Rule header
 - Specifies src/dst host and port
 - Alert tcp !128.119.0.0/16 any -> 128.119.166.5 any
 - Rule options
 - Specifies flags, content, output message
 - (flags: SFAPR; msg: "Xmas tree scan")
- Using both parts together gives snort great flexibility
 - Variables are allowed in the ruleset

Writing Snort Rules

- Snort uses a simple rules language
- http://www.snort.org/writing_snort_rules.htm

- Rule header consists of
 - Rule Actions
 - Alert, Log, Pass Dynamic, activate, etc...
 - Protocol
 - Tcp, udp, icmp, etc...
 - IP Addresses
 - Source, dest, CIDR mask
 - Port numbers
 - Source, dest, range
 - Direction
 - Negation

Writing Snort Rules

- Rule options consists of
 - Msg
 - Flags
 - Content
 - Seq
 - Ack
 - Itype/Icode
 - Fragbits
 - ToS
 - TTL
 - Many others

Snort Rules Example

- alert tcp \$EXTERNAL_NET any -> \$HOME_NET 20432
(msg:"DDOS shaft client to handler"; flags: A+;
reference:arachnids,254;)
- alert udp \$EXTERNAL_NET any -> \$HOME_NET 31335
(msg:"DDOS
Trin00:DaemontoMaster(messagedetected)";
content:"l44";reference:arachnids,186;)
- alert udp \$EXTERNAL_NET any -> \$HOME_NET 31335
(msg:"DDOS
Trin00:DaemontoMaster(*HELLO*detected)";
content:"*HELLO*"; reference:arachnids,185;)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET 27665
(msg:"DDOS Trin00:Attacker to Master default startup
password";flags: A+; content:"betaalmostdone";
reference:arachnids,197;)

Snort rules examples

- alert tcp \$EXTERNAL_NET any -> \$HOME_NET
27665 (msg:"DDOS Trin00 Attacker to Master
default password";flags: A+; content:"gOrave";)
- alert tcp \$EXTERNAL_NET any -> \$HOME_NET
27665 (msg:"DDOS Trin00 Attacker to Master
default mdie password";flags: A+; content:"killme";)
- alert udp \$EXTERNAL_NET any -> \$HOME_NET
27444 (msg:"DDOS
Trin00:MastertoDaemon(defaultpassdetected!
)"; content:"l44adsl"; reference:arachnids,
197;)

Intrusion Detection Evasion

- There are two common evasion techniques
 - Ptacek and Newsham
 - http://www.nai.com/media/pdf/nai_labs/ids.pdf
 - There is insufficient information available in packets read off the wire to correctly reconstruct what is occurring inside complex protocol transactions
 - for example, can the IDS predict if the network machine out of memory? Does it drop the packet or not if the checksum doesn't match? Will it detect **unicode**?
 - False positives can degrade the value of a system.
- ID systems are inherently vulnerable to denial of service attacks.
 - “If an attacker can crash the IDS or starve it of resources, she can attack the rest of the network as if the IDS wasn't even there”
 - DoS attacks are difficult to solve

IDS Evasion with Unicode

- Unicode
 - Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.
 - Unicode characters are called code points and can be represented by `U+xxxx` where `x` is a hexadecimal digit.

IDS Evasion with Unicode

- Why is Unicode a problem?
 - It is possible that there could be multiple representations of a single character.
 - There are also code points that may be used to modify the previous code point.
 - Many of these code points have multiple representations themselves. Since IIS is not case-sensitive, this leads to 30 different representations for the letter "A". There are 34 for "E", 36 for "I", 39 for "O", and 58 for "U". The string "AEIOU" can be expressed **83,060,640** different ways
 - “Today it is possible to use UTF-8 encoding to attack an IIS server and evade detection from all vendor's NIDS.”

Unicode problems

- IDS Evasion with Unicode (Eric Hacker)
<http://online.securityfocus.com/infocus/1232>
- Microsoft Internet Information Server (IIS) comes with unicode extensions that are notoriously exploitable:
- <http://192.168.0.1/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\>
- %c0%af is unicode for /
- More details
 - in “Microsoft IIS Unicode Exploit” (Nate Miller, lucent white paper)
http://www.lucent.com/livmlink/197020_Whitepaper.pdf
 - Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability
<http://online.securityfocus.com/bid/1806>

Insertion and Evasion

- `http://192.168.0.1/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\`
- `%c0%af` is unicode for `/`
- Making the end-system and the IDS match up is important.
- When data reaches the IDS but not the endhost, the type of attack is called **insertion**.
 - Fragmentation attacks are a form of insertion because extra data reaches the IDS.
- When data is dropped by the IDS but reaches the end-host, then it is called **evasion**.
 - The unicode example is evasion in a sense because the important data was “dropped”, that is “wasn’t detected” by the IDS.

Data correlation

- An IDS system provides us with another point
 - The strength of the system comes from correlating data among many different sensors...
 - Possibly even across organizations
 - Securityfocus Incidents List/ARIS
 - Unisog
- Example:
 - Name: CPE-144-132-255-160.nsw.bigpond.net.au
 - Address: 144.132.255.160
 - May 7 19:22:43 wkstn2 ipmon[6269]: 19:22:42.899221 hme0 @0:1 b CPE-144-132-255-160.nsw.bigpond.net.au,3249 -> wkstn2.domain.edu,ftp PR tcp len 20 48 -S IN
 - May 7 19:24:26 wkstn1 snort: 144.132.255.160:3284 -> 192.168.160.52:21 SYN *****S*
 - May 7 19:45:11 honeypot snort: External Activity Detected: 144.132.255.160:3930 -> 192.168.1.2:21
- Different formats and naming is a problem
 - CVE helps
 - Parsing engines are helpful

Intrusion Prevention Systems

- What if we let an IDS dynamically modify firewall rules?
 - Are there possibilities for abuse?
 - Why would we not want an IPS
- Operational efficiencies
- Is it better to prevent attacks if they are known to be malicious
- Another device in the forwarding path
- How does this impact diagnostics?

Snot

- Snort requires resources like anything else.
- **Snot** is program that takes a snort ruleset and generates packets that fit the rules.
- By clobbering snort with true hits, you hopefully knock it out of commission or keep it so busy that it can't keep up.
- <http://www.sec33.com/sniph/> (and other places)

Manageability: Diagnostics

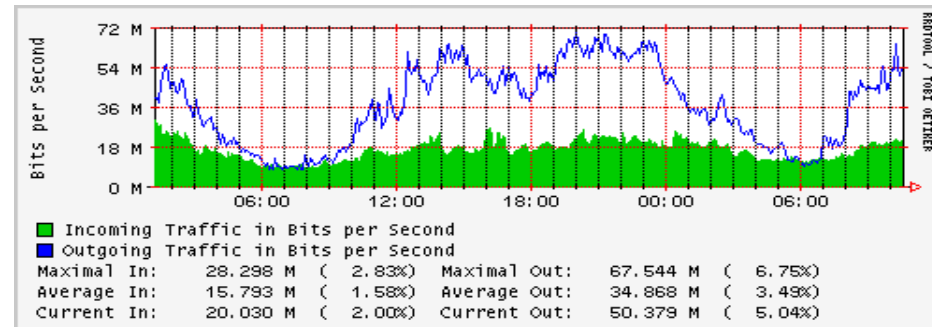
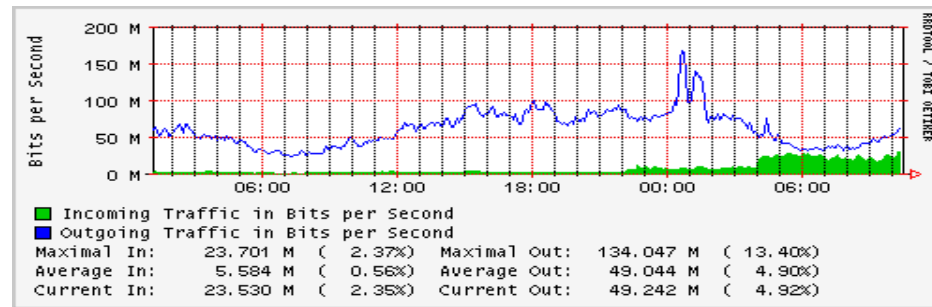
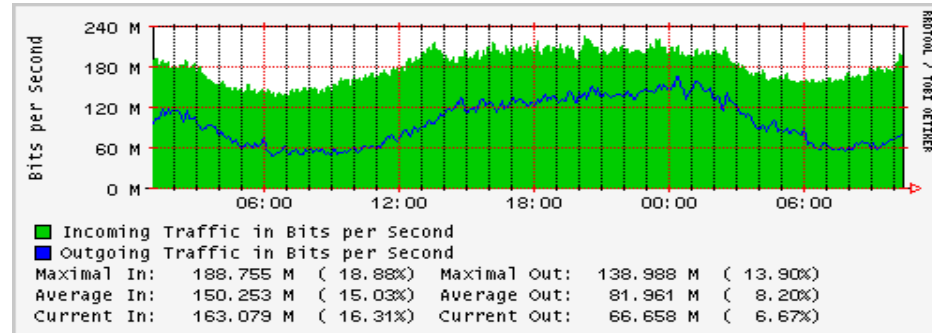
- **Diagnosis** (from the Greek words *dia* = by and *gnosis* = knowledge) is the process of identifying a disease by its signs, symptoms and results of various diagnostic procedures. The conclusion reached through that process is also called a diagnosis.
 - <http://en.wikipedia.org/wiki/Diagnosis>
- **Diagnostic**
 - A symptom or a distinguishing feature serving as supporting evidence in a diagnosis.

Network Diagnostics

- Provide effective exchange, management, and correlation of log and event information
 - between dependent layers
 - among interdependent components
- A data orchestration function
- Enable system managers to pinpoint problems as they occur
- Allow autonomic processes to assist in prediction, management, and maintenance.
 - *<http://www.cmu.edu/computing/eddy/introduction.htm>*

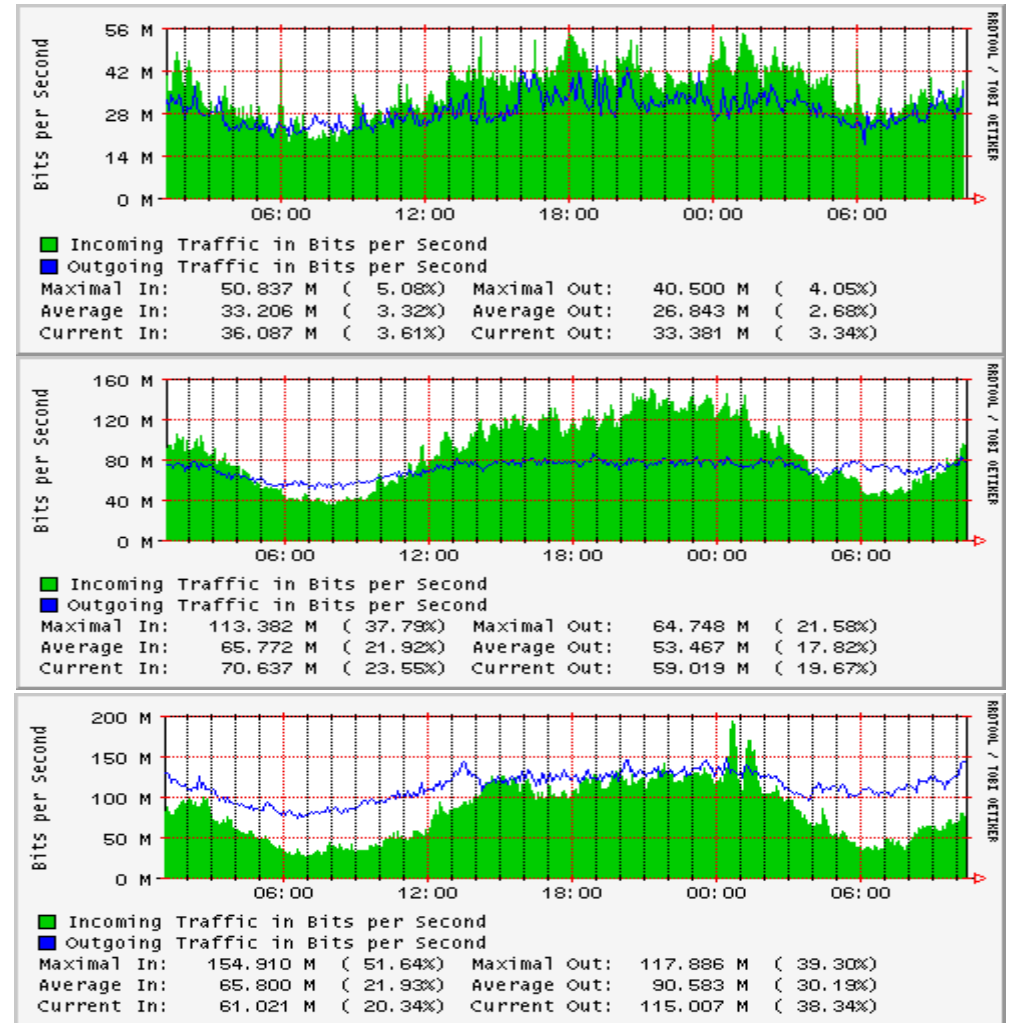
Local Network Bandwidth

- Edge1 <-> Border
 - LGRC
- Edge2 <-> Border
 - LIBR
- Local Peers
 - 5 College
 - 5 Campus



Peering Network Bandwidth

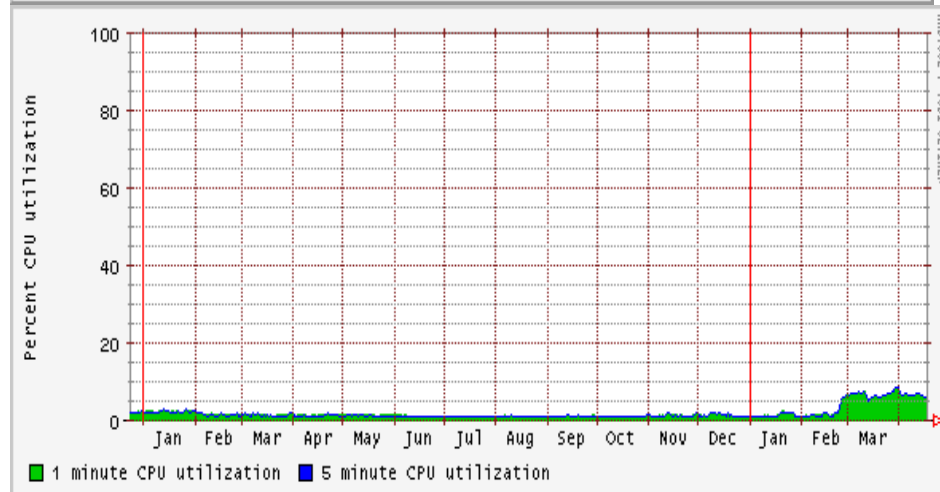
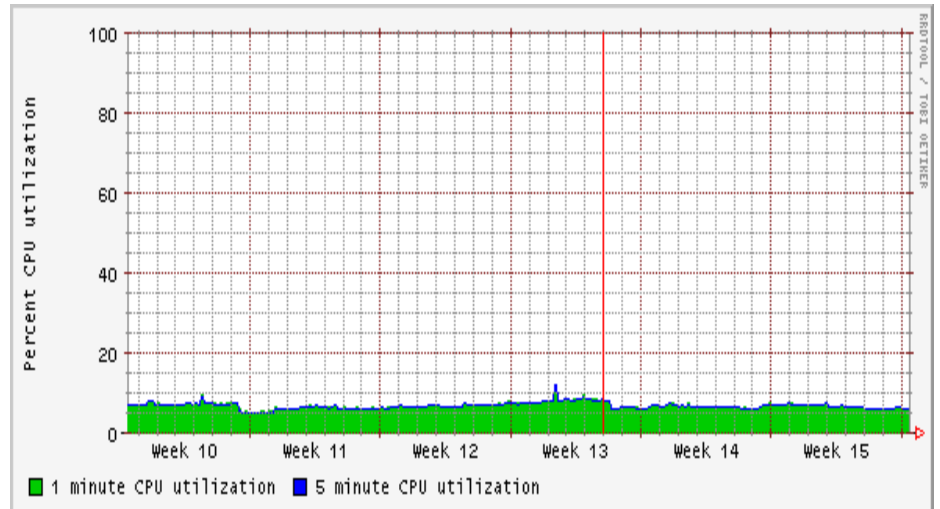
- Internet2
 - Boston/NYC
- Commodity ISP1
 - Boston
- Commodity ISP2
 - Springfield



Router CPU Utilization

- Monthly
 - 1 and 5 minute polling interval

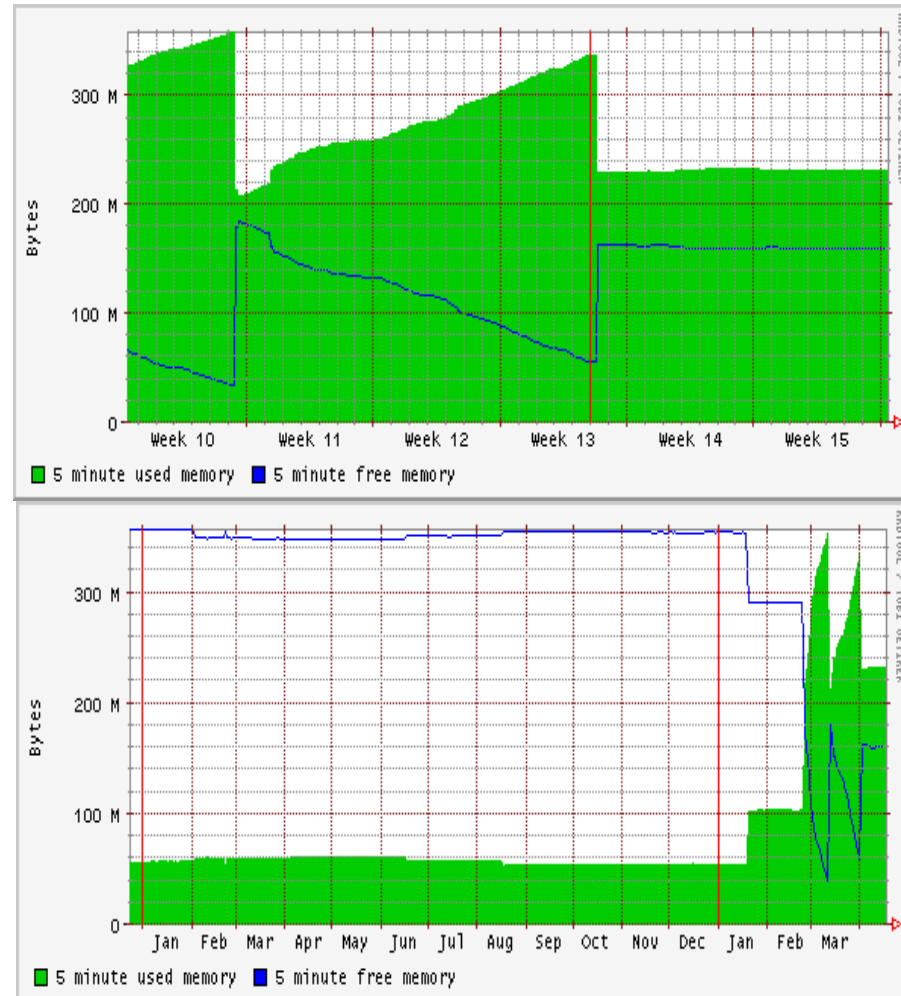
- Yearly
 - 1 and 5 minute polling interval



Router Memory Utilization

- Monthly
 - 5 minute polling
 - Used vs Free

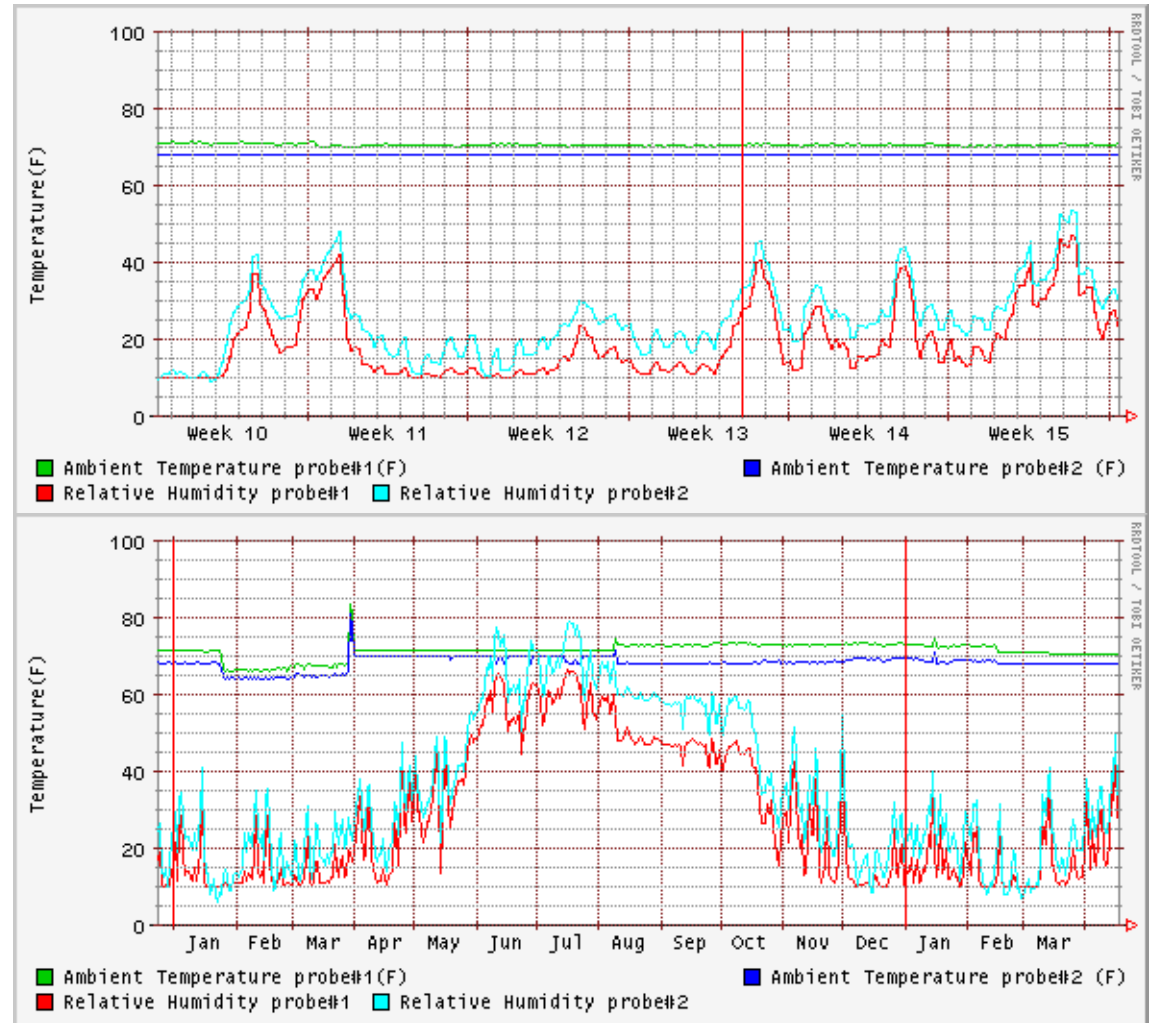
- Yearly
 - 5 minute polling
 - Used vs Free



Nodesite Environmental

- Monthly
 - Temperature
 - Humidity

- Yearly
 - Temperature
 - Humidity



Diagnostics: NetFlow

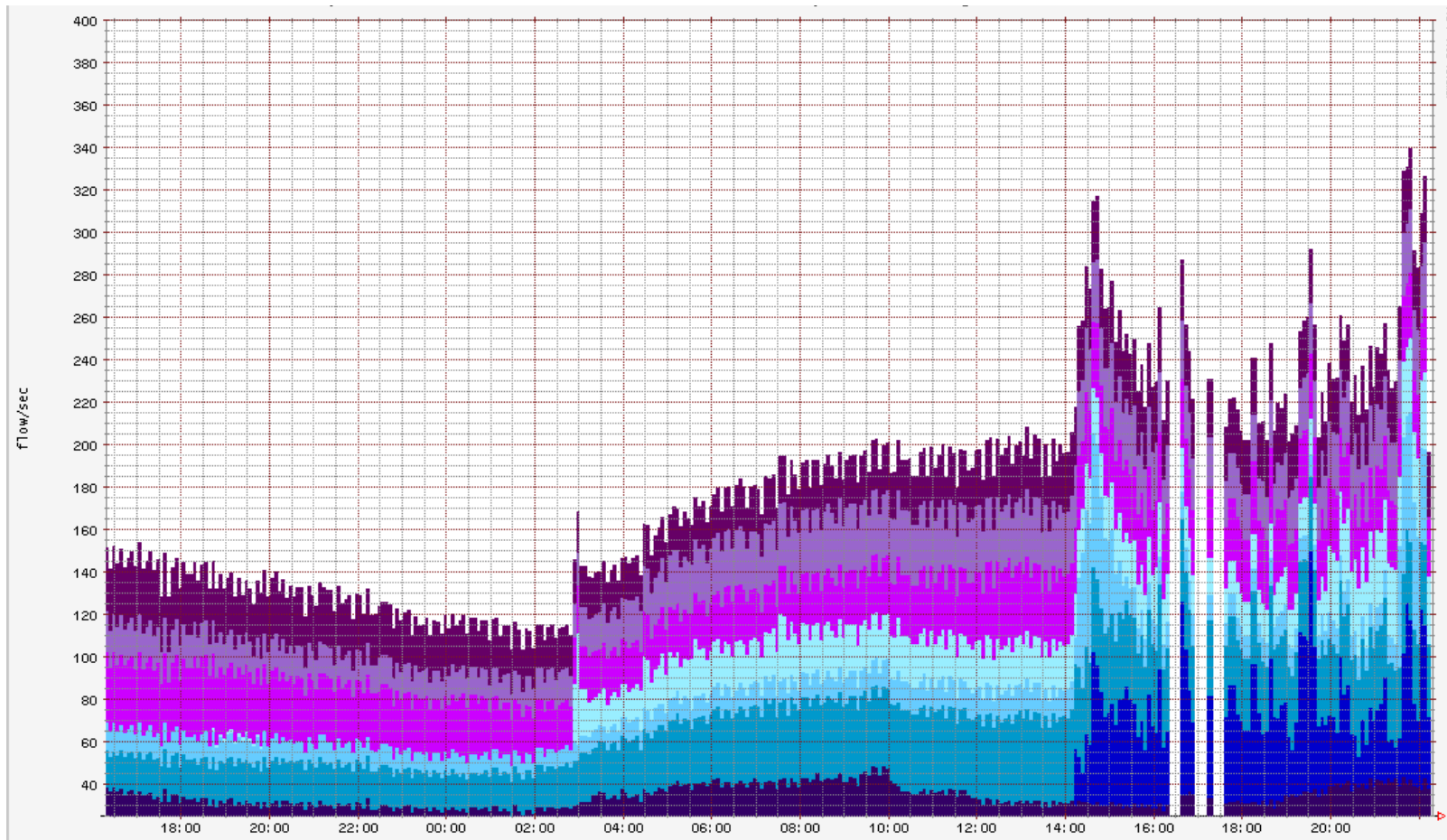
“NetFlow technology efficiently provides the metering base for a key set of applications including network traffic accounting, ...”

- Data export mechanism that records information about router flows.
 - Src/dst IP, port, etc
 - Bytes
 - *No packet content is logged*

NetFlow: Caveats

- Great tool for detecting Denial of Service attacks
 - However, it is prone to data loss under abnormal load
 - Visual analysis is often the most efficient detector
- Great tool for post-incident analysis
 - Provided the data has not been cycled off the system
- As links become faster, many flow exports are sampled
 - You get a statistical representation of data across your network
 - Still useful for Capacity planning and DoS detection, but of limited use for forensics purposes
- Not necessarily the first tool in your toolkit, but an invaluable one to complement all the others

Netflow: Graphic analysis



Netflow: Data

srcIP	dstIP	prot	srcPort	dstPort	octets	pkts
80.116.163.85	xxx.yyy.131.204	17	3111	1434	404	1
81.3.162.10	xxx.yyy.131.182	17	1514	1434	404	1
200.74.27.228	xxx.yyy.131.246	6	447	8080	40	1
200.74.27.228	xxx.yyy.131.246	6	64068	80	40	1
200.74.27.228	xxx.yyy.131.246	6	50265	3128	40	1
142.179.169.213	xxx.yyy.131.178	17	1126	1434	404	1
213.60.21.96	xxx.yyy.131.171	17	1923	1434	404	1
212.180.2.68	xxx.yyy.131.114	6	63559	41544	40	1
200.29.164.162	xxx.yyy.131.233	17	1051	1434	404	1
202.103.13.62	xxx.yyy.131.35	6	9001	30185	40	1
213.119.233.63	xxx.yyy.131.7	17	1246	1434	404	1
216.51.150.219	xxx.yyy.131.7	17	1157	1434	404	1
24.112.24.160	xxx.yyy.131.122	17	1129	1434	404	1

Device/Configuration Management

- Given the large number of infrastructure devices, automated management is required
 - Device availability
 - Scheduled outages
- Configurations need to be centrally stored
 - And retrievable
- Accountability and audit capability
 - To allow efficient restoration of service