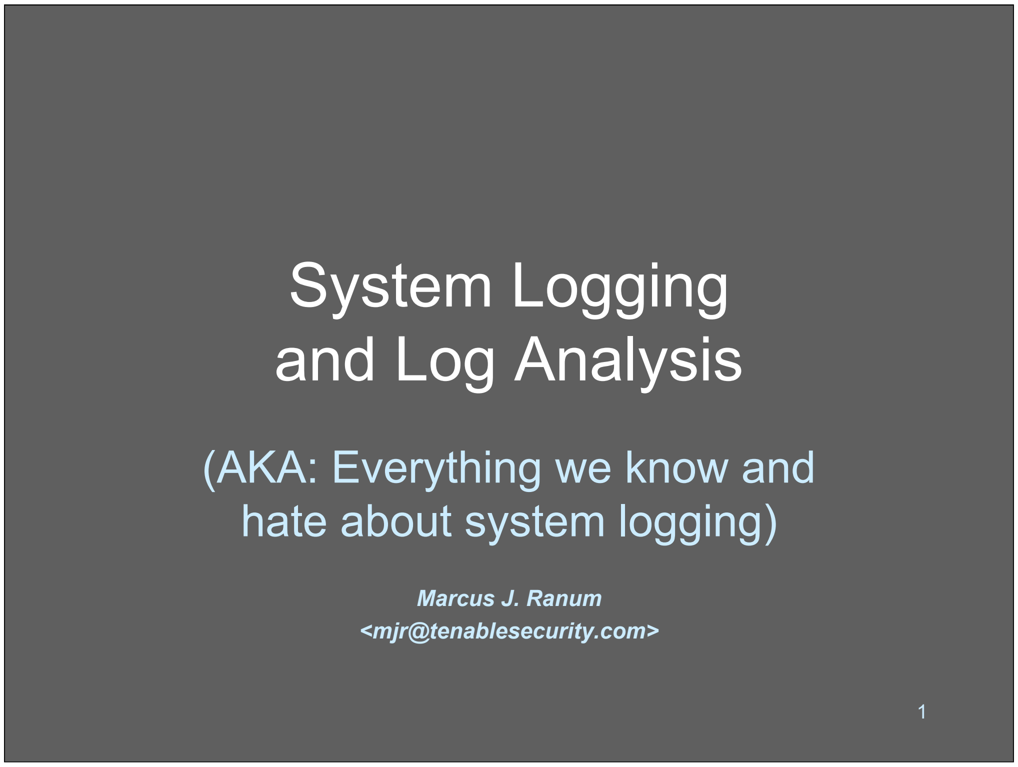


# System Logging and Log Analysis

(AKA: Everything we know and hate about system logging)

*Marcus J. Ranum*
*<mjr@tenablesecurity.com>*

1

Welcome to my system logging and analysis tutorial!!!

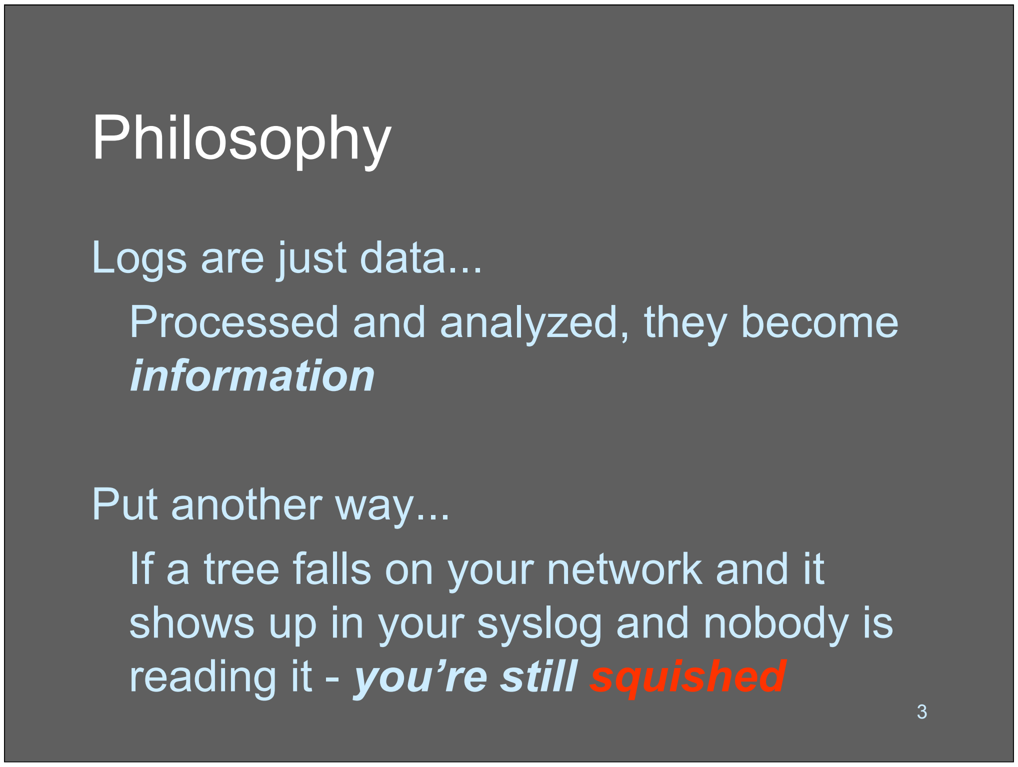
What we're covering is a huge topic that touches on security, system administration, and data management. It's a complex problem and it seems as if everyone has come up with their own approach to dealing with it. In this tutorial, we'll cover a lot of concepts and tools for deploying them.

In order to make this tutorial as useful as possible, don't hesitate to ask questions at any time. Don't hesitate to contact me in the future, if you have questions or need advice; just let me know you were at my tutorial and I'll help you if I can!

This is one of the better “months” from my 2004 Sourcefire Security Calendar. You can see the entire calendar on:

http://www.ranum.com/security/computer_security/calendar/index.html

A lot of organizations are spending a ton of perfectly good money on intrusion detection systems and whatnot, but they don’t ever look at their firewall logs. If you think about it for a second, you’ll realize how absolutely ***backward*** that is!!System logs represent a terrific source of security data straight “from the horse’s mouth” - or from your devices, anyhow.



## Philosophy

Logs are just data...

Processed and analyzed, they become ***information***

Put another way...

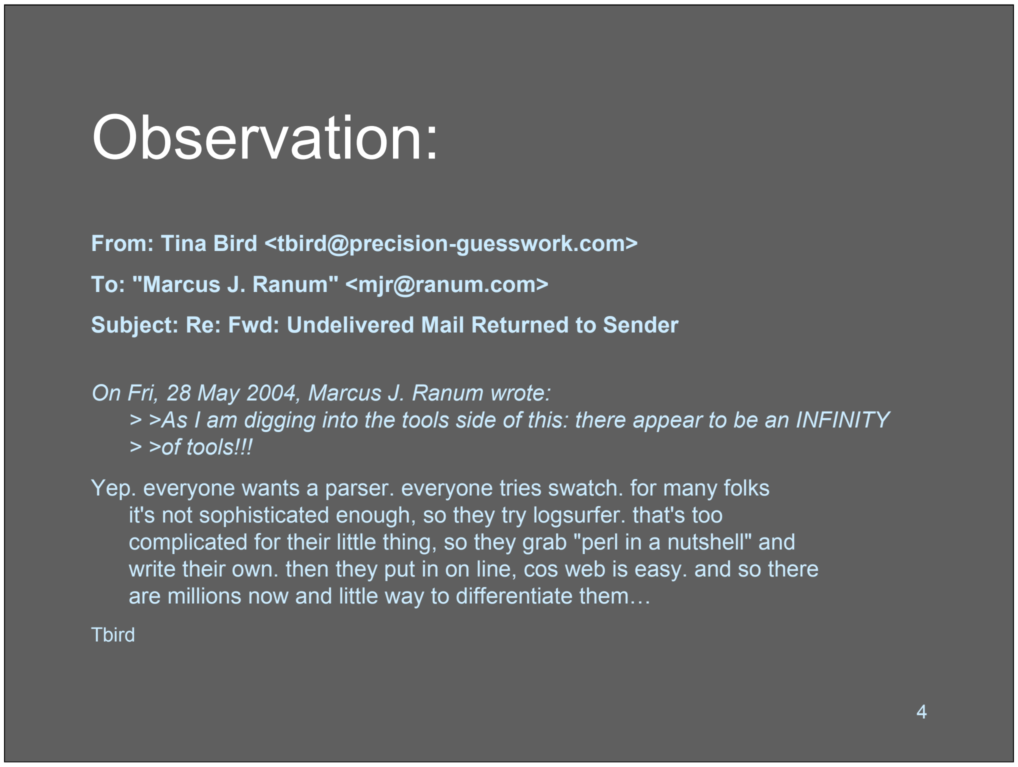
If a tree falls on your network and it shows up in your syslog and nobody is reading it - ***you're still squished***

3

In military circles, the phrase "actionable intelligence" is an important term used to describe intelligence that has immediate practical importance. What we're talking about with system log data is the same thing: we want to boil down this mass of data produced by our firewalls, routers, IDS, and applications, and turn that data into actionable intelligence: useable recommendations for what we should do, and when.

The process of turning data into information is called analysis. That's what "analysts" do. It's an expensive process, because it requires skills and creativity - nobody can replace a good analyst with a perl script - so if you're going to bear the cost of analyzing log data, make sure that the results are actionable. If you're talking to a senior manager who wants to you to put in place all sorts of mechanisms for data capture, reduction, and analysis - make sure you "encourage" them to also put in place processes and personnel to act on any intelligence you gather.

Put differently, it's pretty easy to tell you have a 'botnet when you turn on CNN and they're talking about your site getting hacked to pieces. The trick is telling you have a 'botnet before anyone else (except the hacker) knows - and having the processes and personnel in place to do something about it.



## Observation:

**From: Tina Bird <tbird@precision-guesswork.com>**

**To: "Marcus J. Ranum" <mjr@ranum.com>**

**Subject: Re: Fwd: Undelivered Mail Returned to Sender**

*On Fri, 28 May 2004, Marcus J. Ranum wrote:*
*> >As I am digging into the tools side of this: there appear to be an INFINITY*
*> >of tools!!!*

Yep. everyone wants a parser. everyone tries swatch. for many folks it's not sophisticated enough, so they try logsurfer. that's too complicated for their little thing, so they grab "perl in a nutshell" and write their own. then they put in on line, cos web is easy. and so there are millions now and little way to differentiate them…

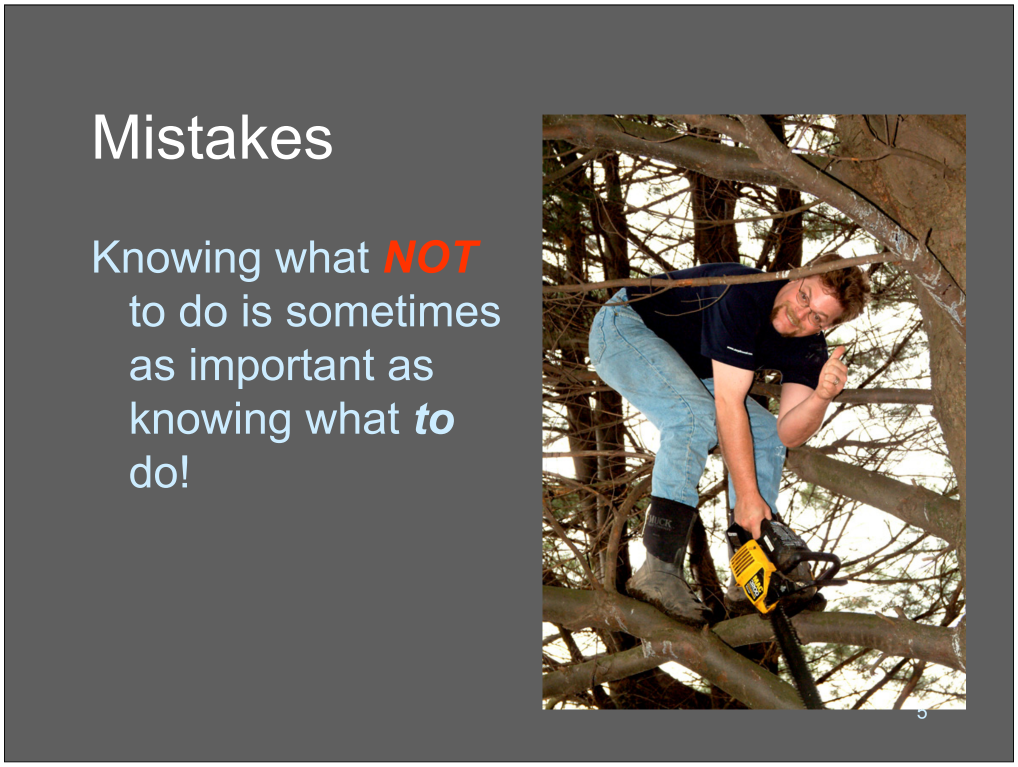
Tbird

4

In preparing my materials for this tutorial I had the exchange above with Tina Bird.

Basically, I was stunned to discover that hundreds of smart people have tried to solve this same problem over and over and over and over again, and many of us follow the same path. There are full-blown systems for processing logs, and they can cost (literally) between a million dollars and $500. Smart people ask themselves, “how hard can it be?” and grab a few tools, get disappointed, and then customize their own. As a result there are hundreds of logging tools out there - so many it’s hard to keep track of even a fraction of them.

The problem - as we’ll see - is that system logging is a ***hard*** problem. There’s no quick solution, short of throwing money at the problem, and the lack of a quick solution drives smart people to build their own tools and solve the problem in their own particular way.



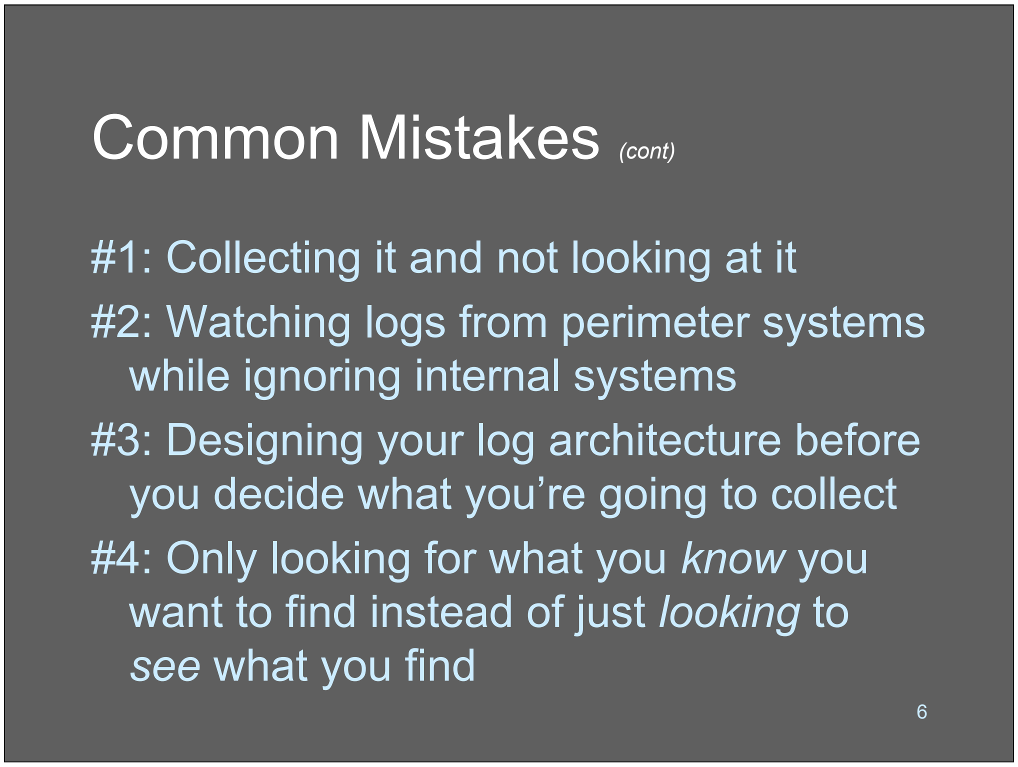
## Mistakes

Knowing what ***NOT*** to do is sometimes as important as knowing what ***to*** do!

5

A lot of instructors like to end a tutorial with a list of “do”s and “don’t”s - but that’s backwards because as you work your way through the materials, you won’t be on the lookout for things that might bite. If you’re in this tutorial it’s because you probably have some kind of system log-related problem to solve. Knowing what doesn’t work might actually be more important for you than knowing what does, if you’re on a tight schedule or don’t have a lot of time to run up blind alleyways.  So we’re going to take a look at some of the ***bad*** things that you can do, so you’ll be able to bear them in mind and avoid them.

(No, the chainsaw was not running; that’s photoshop motion blur. But thanks for caring)



## Common Mistakes *(cont)*

#1: Collecting it and not looking at it

#2: Watching logs from perimeter systems while ignoring internal systems

#3: Designing your log architecture before you decide what you're going to collect

#4: Only looking for what you *know* you want to find instead of just *looking* to *see* what you find

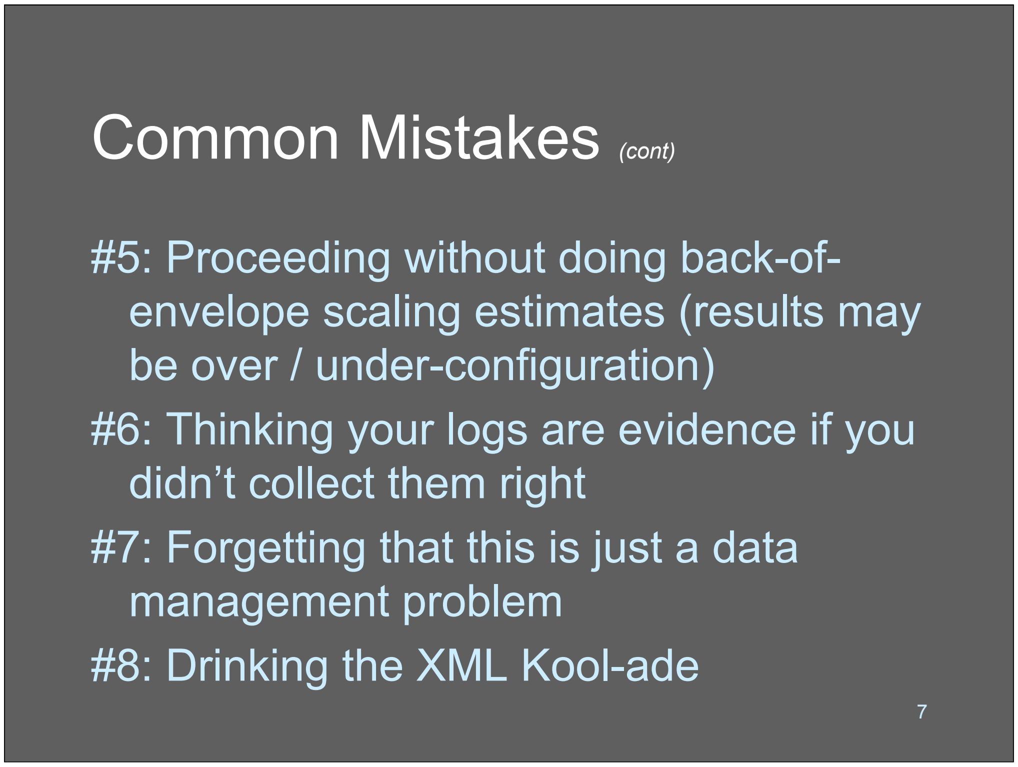
6

**#1:** If you're here because you've been told to aggregate logs and you don't think you'll ever do anything with them, you're going to be frustrated because you're doing something that's basically pointless. Think in terms of doing as much simple automated stuff along with your aggregation so you can get something out of your efforts.

**#2**: Your firewalls and whatnot are incredibly important places to do logging. But so are your crucial servers and even your ordinary workstations. When you're building logging systems think in terms of assessing the likely significance of the data depending on its source.

**#3**: Don't rush to decide what tools you are going to use until you have a good idea what you're going to try to accomplish. My grandfather the carpenter sometimes would tackle a problem that appeared to be huge with just a single screwdriver. Other times he'd bring a whole toolbox. Somehow, he always appeared to have just the right stuff. It wasn't until I was older that I realized he had scoped the problem thoroughly before he even moved a muscle. Don't predispose yourself to a particular tool because it means you'll ignore the others - and they might be better.

**#4:** Looking for well-known stuff is a really straightforward problem - finding the weird stuff in the nooks and crannies is not. Unfortunately, with logs, the stuff you want to find is in the nooks and crannies; your firewall and IDS detected the well-known stuff. Approach log analysis with "the mind of a child" (as the martial artists say) - plan to spend a few days *just looking* at stuff and asking yourself, "hmmm, what can I *do* with *this*?"



## Common Mistakes *(cont)*

#5: Proceeding without doing back-of-envelope scaling estimates (results may be over / under-configuration)

#6: Thinking your logs are evidence if you didn't collect them right

#7: Forgetting that this is just a data management problem

#8: Drinking the XML Kool-ade

7

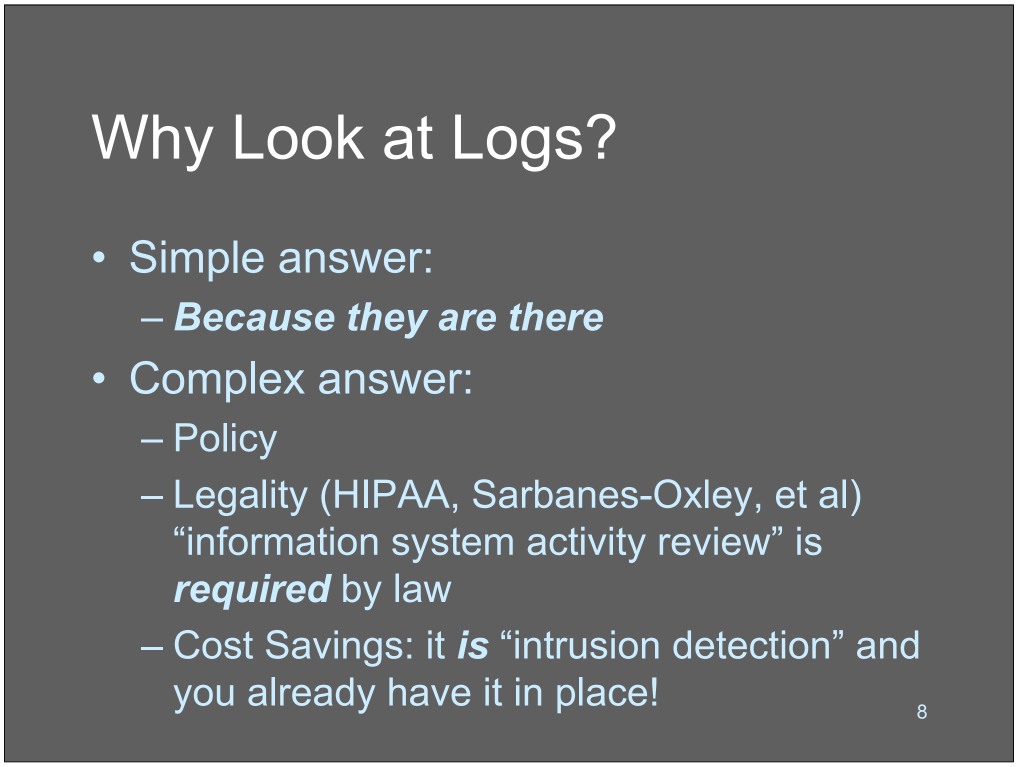
**#5:** Use multiplication technology™ to get a rough idea how many events/second your system will have to handle. If you think in terms of "millions per day" it sounds big but that's only a trickle: 11 events/second. Assume log messages are 100 bytes/line or thereabouts. Measure a few of your machines and do your own estimates. One of my friends' machines generates 16 million logs/day; one of mine generates fewer than a dozen.

**#6:** Lawyers make $400/hr to argue about stuff. Arguing about evidence is fun, if you're making that kind of money! Make sure you understand what your logs are going to be used for, before you pay some lawyers' kid's college tuition with your syslogs.

**#7:** Fundamentally, logging is just data management of free-form data. Don't make the mistake of thinking it's rocket surgery. But don't forget that data management is hard. For example: if you think there's a particular datum you'll need quickly, pre-compute it inline.

**#8:** The hard part about system logging is parsing the data into meaningful information. XML, as a mark-up language, is a good* way of preserving the structure of information once it has been parsed. XML "parsers" parse XML; they don't parse arbitrary free-form text; which is what system logs store. If I had a dollar for everyone who has expected XML to "solve the logging problem" we'd be doing this tutorial on my yacht in Bermuda and the note-book you're holding would be bound in Coach™ leather.

* Actually, it sucks; I'm just being nice



## Why Look at Logs?

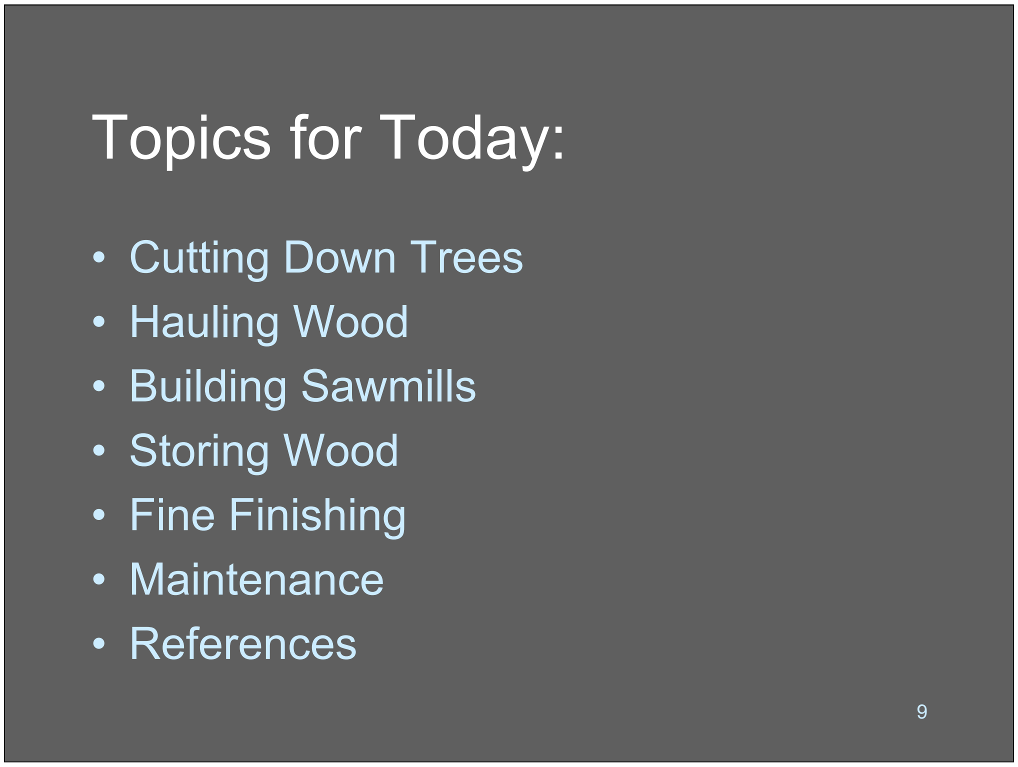
- Simple answer:
  - ***Because they are there***
- Complex answer:
  - Policy
  - Legality (HIPAA, Sarbanes-Oxley, et al) "information system activity review" is ***required*** by law
  - Cost Savings: it ***is*** "intrusion detection" and you already have it in place!

8

Ok, now that I've disappointed you and crushed your aspirations, why would you still want to look at your logs?

***Because they are there*** - most of the best sites, security-wise, are good because they have *smart people* who care about keeping their networks secure. Looking at your logs is an important part of caring. Most of the "great catches" I have seen in 16 years doing computer security are the result of a system or network manager seeing something weird in a log, or have involved figuring out what happened from logs.

But if that's not a good enough reason for you, lawmakers (motto: "We're from the government, and we're here to help - consultants get rich") are beginning to mandate security - and virtually every mandate contains the word "audit" (I.e.: look at logs). The best reason, however, is that if you analyze your systems you'll have a much greater chance of being The Person With a Clue when everyone is running around flapping their hands trying to figure out what went wrong. Being known as The Person With a Clue is sometimes good, sometimes bad, but never boring.



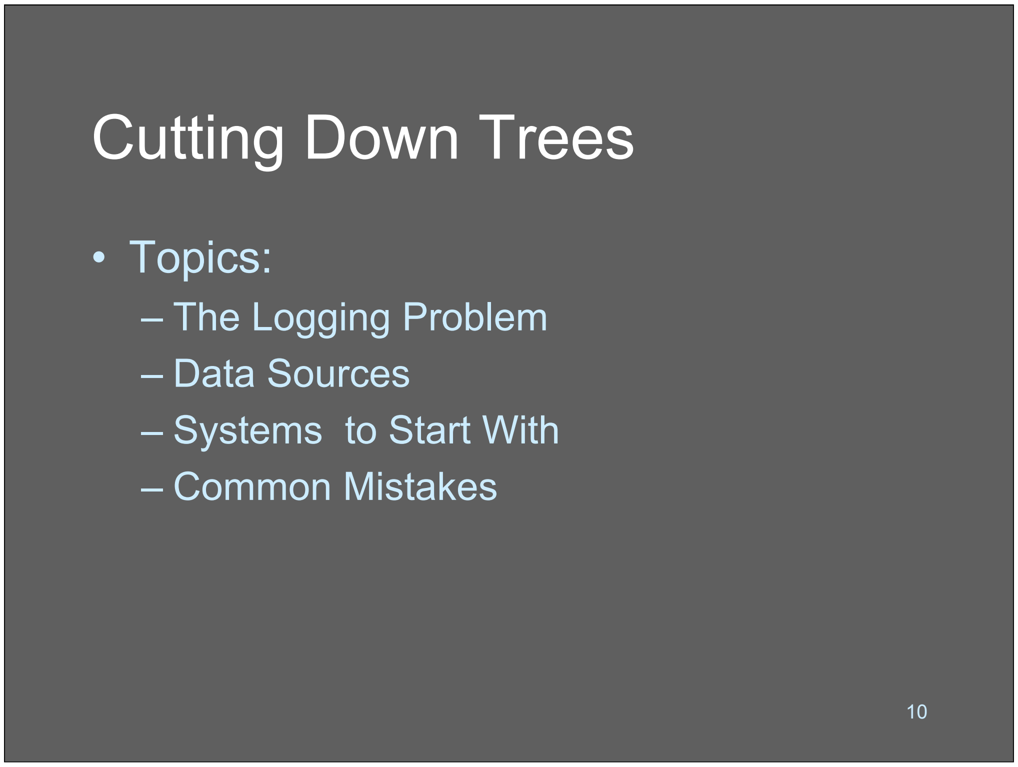
## Topics for Today:

- Cutting Down Trees
- Hauling Wood
- Building Sawmills
- Storing Wood
- Fine Finishing
- Maintenance
- References

9

This is what we're going to talk about today.

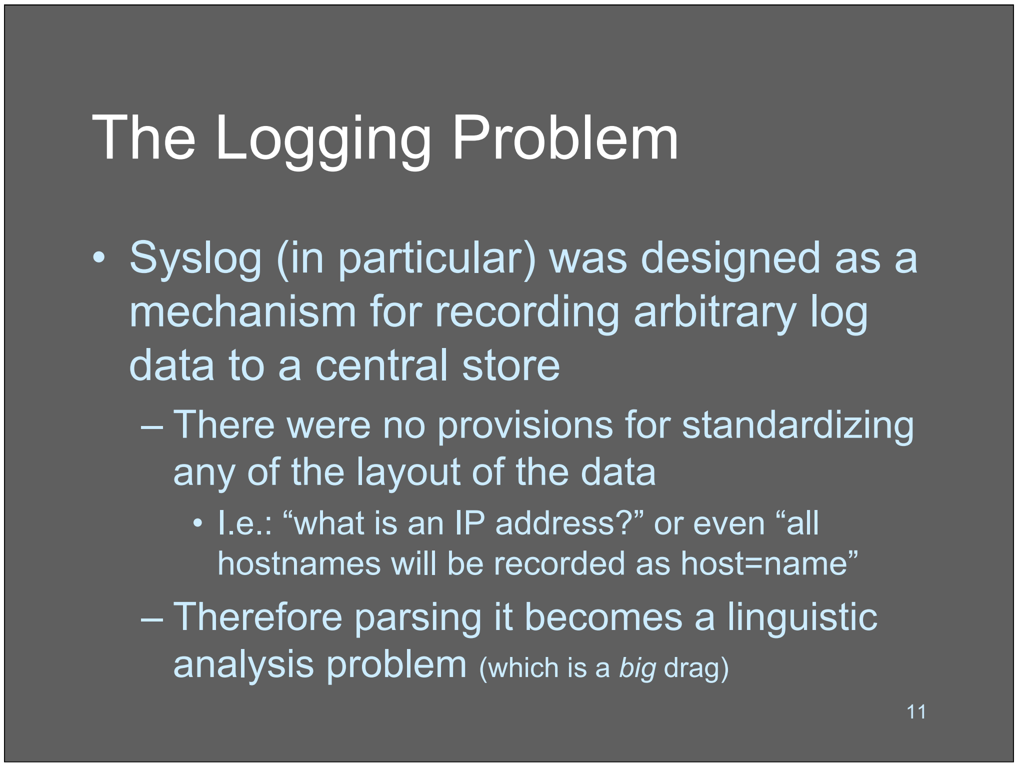
When you're a teacher, they tell you to tie things together, somehow. Well, with a topic like "system logging" using a "logging" metaphor was a pretty easy stretch.



## Cutting Down Trees

- Topics:
  - The Logging Problem
  - Data Sources
  - Systems  to Start With
  - Common Mistakes

10



## The Logging Problem

- Syslog (in particular) was designed as a mechanism for recording arbitrary log data to a central store
  - There were no provisions for standardizing any of the layout of the data
    - I.e.: “what is an IP address?” or even “all hostnames will be recorded as host=name”
  - Therefore parsing it becomes a linguistic analysis problem (which is a *big* drag)

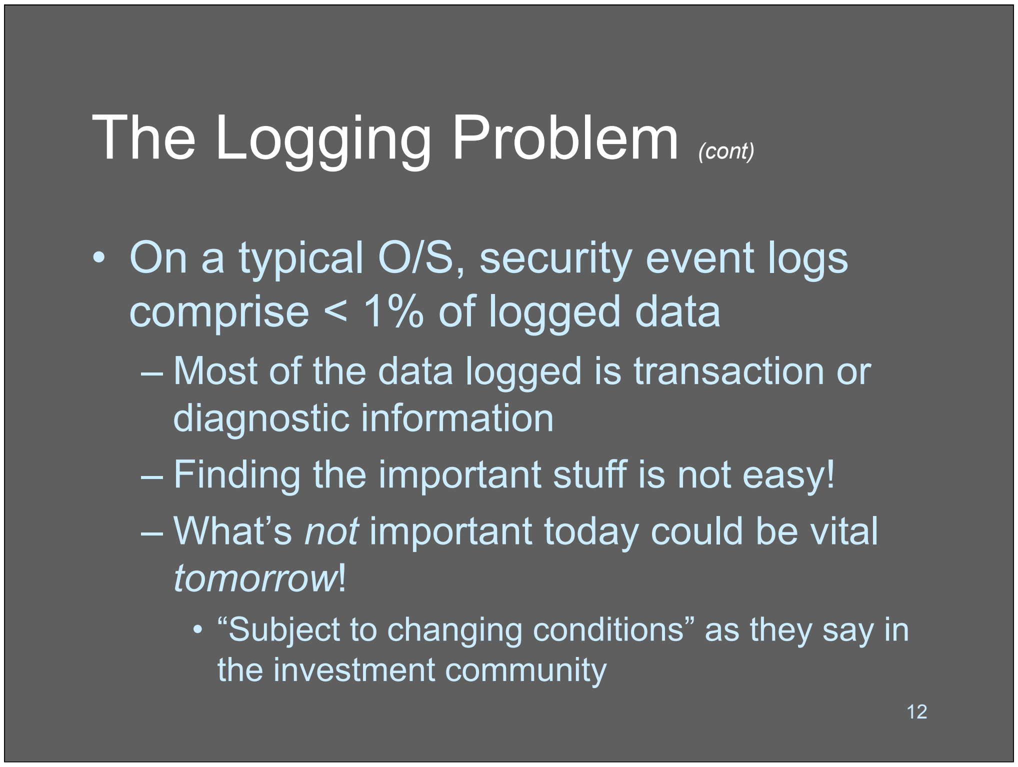
11

I had a chance to talk to Eric Allman, the author of syslog, at a USENIX conference. I asked him, “what the !(&!!&!&! Were you thinking?!” and he explained that syslog evolved because when he was working with the BSD releases, every program kept its own log files in a place of its choice. So there were literally hundreds of log files all over that needed to be cleaned up. Eric designed syslog as a way of internally aggregating a UNIX host’s log messages, and, because most of the programs used:

```
fprintf(logfile,”message\n”,parameters);
```

to do their logging, he designed the syslog interface to match so it would be easy to just search and replace in the source code.

What’s ironic is that Eric did all this work to bring logs together in one place. Now we log analysts are scratching our heads trying to figure out how to separate them again. A little bit more forethought would have been really helpful to us all. Once a large body of code is released, fixing it is hard. Once syslog became widely fielded, it was too late to fix it.



## The Logging Problem *(cont)*

- On a typical O/S, security event logs comprise < 1% of logged data
  - Most of the data logged is transaction or diagnostic information
  - Finding the important stuff is not easy!
  - What's *not* important today could be vital *tomorrow*!
    - "Subject to changing conditions" as they say in the investment community

12

System logs (particularly in the UNIX environment) were originally intended as a tool for diagnosing and debugging code. They evolved into a way of recording transactional information (e.g.: "sendmail stat=sent" messages) to allow statistics collection and summarizing. From a security standpoint, this was another bad decision because it means that, before we can do anything, we have to separate security-specific log data from operational log data.

If you assume that security-critical data is less than 1% of the data collected, here's an interesting question: is it *better* that we're forced to actively search through tons of "noise" as we search for security events? One thing we have learned over time is that some log messages which nobody would consider security event messages may actually be the precursor-indicators of an attack.

Is this a security log message?

```
/scripts/..À¯../winnt/system32/cmd.exe?/c copy c:\winnt\system32\cmd.exe c:\inetpub\scripts\shell.exe
```

It's from an 404-log from a web server. Obviously, 404's aren't a "security event" - except for when they are.

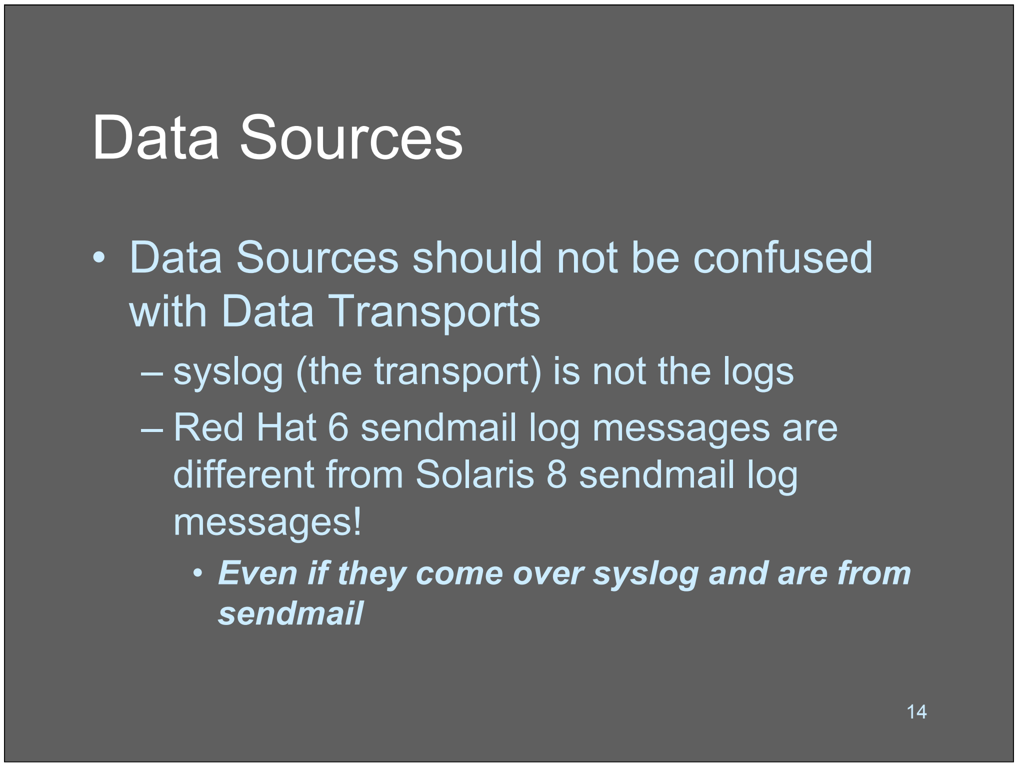
Figuring out what log events are security events or operational events is a moving target, unfortunately. It may be better for us to just accept that we're searching haystacks for needles and plan accordingly!

## Data Sources

13

I actually live right across the street from Greenwood's sawmill, in Morrisdale!

So: I used this tutorial as an excuse to go over and visit and take some pictures. This sign is right across from our yard; the sawmill starts operation at around 5:30AM on weekdays, and we can hear the skid-loaders going "beep beep beep" as they back up and load the logs.



## Data Sources

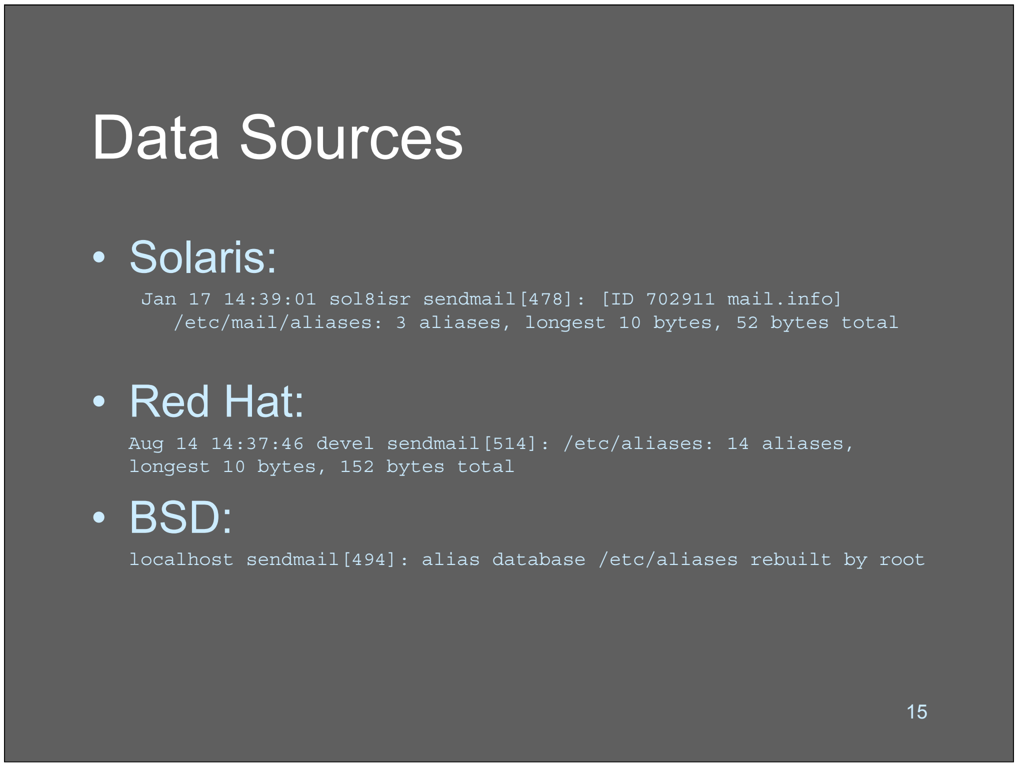
- Data Sources should not be confused with Data Transports
  - syslog (the transport) is not the logs
  - Red Hat 6 sendmail log messages are different from Solaris 8 sendmail log messages!
    - ***Even if they come over syslog and are from sendmail***

14

A lot of people think that because syslog is a “standard” that “syslog” is also a message format. Syslog is just an API and a way of getting free-form text messages back and forth; it’s as much of a logging standard as SMTP would be, if you used Email to carry them around.

It’s important to understand the distinction between the source of the data and how it gets back and forth; some devices use syslog to carry UNIX-style messages, but others simply use syslog to carry proprietary-coded messages (e.g.: “e 344 22”) that make no sense unless you have the correct code-book to decipher them. Even UNIX-style messages are problematic - some versions of UNIX-y operating systems omit date/time stamps for local logs (ow!) or from kernel logs (ugh!) and different releases of the same software package may log using different formats, even on the same system.

When talking about syslog sources, you practically have to specify things down to the version: “I am using *foo 4.3a on OpenBSD 3.2*” I did some research on structural analysis of log messages (research that didn’t go very well!) and discovered that most major releases of sendmail has different and new log message formats. Not just new messages, but whole new ways of structuring them.



## Data Sources

- Solaris:

```
 Jan 17 14:39:01 sol8isr sendmail[478]: [ID 702911 mail.info]
    /etc/mail/aliases: 3 aliases, longest 10 bytes, 52 bytes total
```

- Red Hat:

```
Aug 14 14:37:46 devel sendmail[514]: /etc/aliases: 14 aliases,
longest 10 bytes, 152 bytes total
```

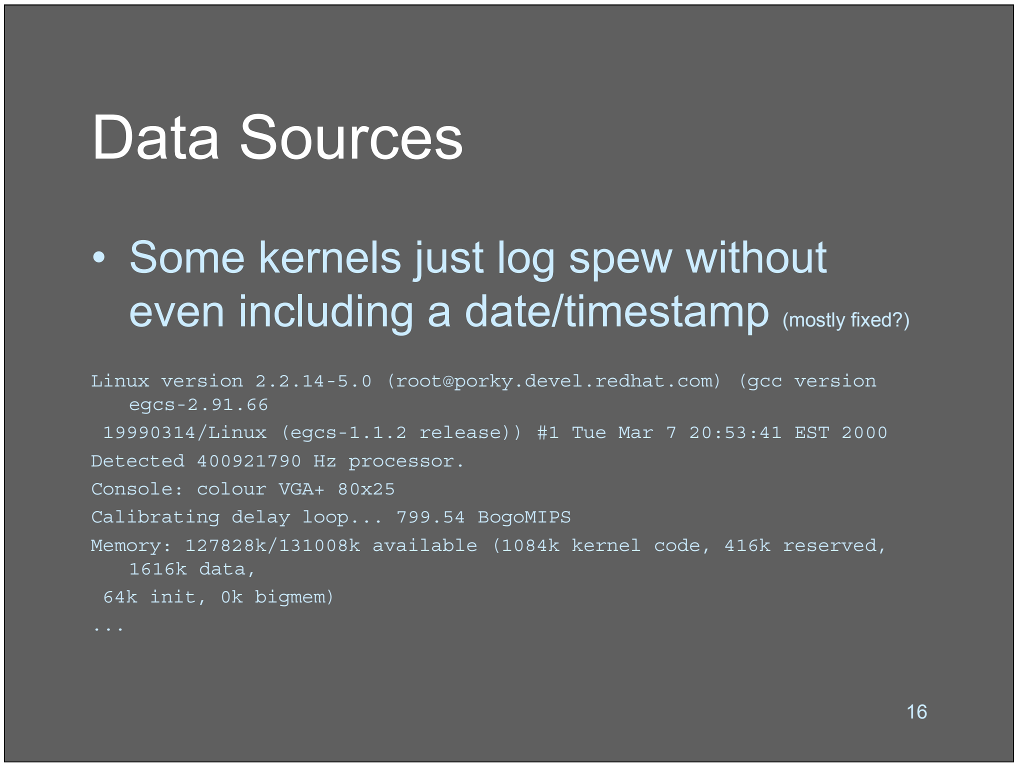
- BSD:

```
localhost sendmail[494]: alias database /etc/aliases rebuilt by root
```

15

This is an example of a couple of different data sources, all carried over syslog transport. As you can see, they're all from sendmail and they're all quite different. Amazingly, they're all from the same *operation* - refreshing the aliases database.

What does this mean for the log analyst? It means that if you want to do something useful with "newaliases" output and you support these 3 platforms, you're going to need 3 different parsing rules and you're still going to ***lose information***. See the BSD log message? It says that the database was rebuilt by root. Do you care? Well, if you *do* care, you haven't got that information in the Solaris or Red Hat logs. If you tried to make a normalization rule that did something useful with all the fields presented in this one message, you'd have a lot of work finding a reasonable superset/subset of fields to parse out and use.



## Data Sources

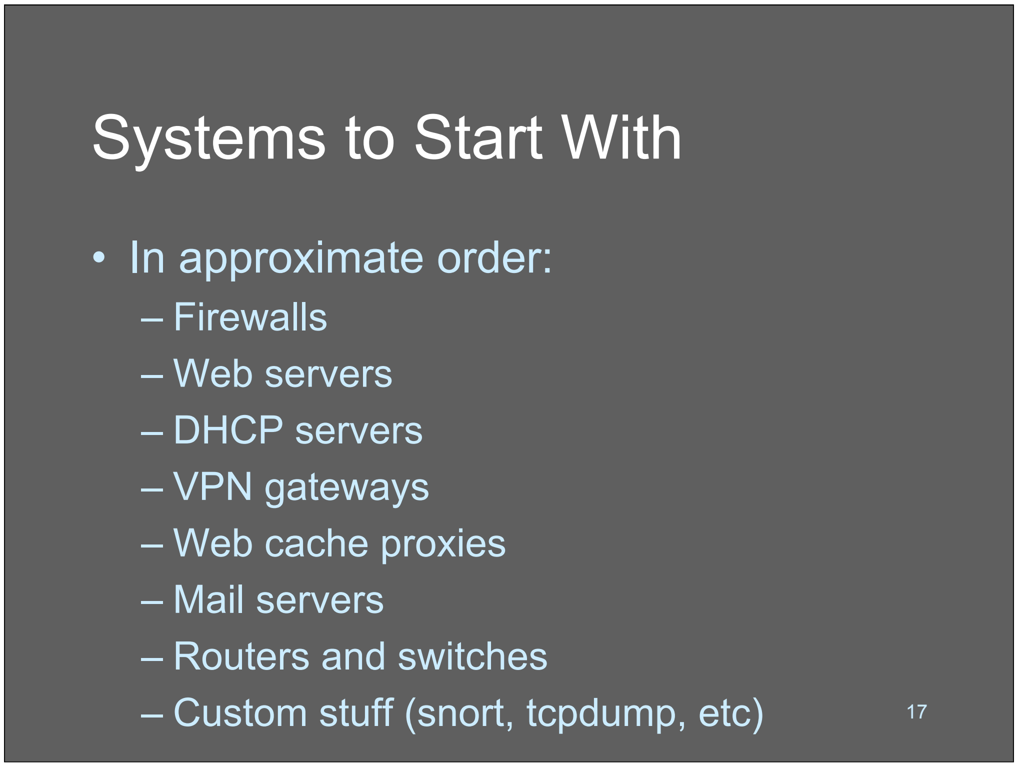
- Some kernels just log spew without even including a date/timestamp (mostly fixed?)

```
Linux version 2.2.14-5.0 (root@porky.devel.redhat.com) (gcc version
   egcs-2.91.66
 19990314/Linux (egcs-1.1.2 release)) #1 Tue Mar 7 20:53:41 EST 2000
Detected 400921790 Hz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 799.54 BogoMIPS
Memory: 127828k/131008k available (1084k kernel code, 416k reserved,
   1616k data,
 64k init, 0k bigmem)
...
```

16

Would you like to try to use your kernel log as evidence in a court case if it didn't have date/time stamps in it? What if you had to explain to a jury that the date and timestamp gets added on by an external process if it's missing? Would that be a comfortable position to be in? Welcome to syslog!

This has been fixed in most UNIXes thanks to innumerable computer security practitioners chasing Linux kernel developers around conferences with pitchforks and burning torches. But you've got to wonder what they were thinking in the first place.



## Systems to Start With

- In approximate order:
  - Firewalls
  - Web servers
  - DHCP servers
  - VPN gateways
  - Web cache proxies
  - Mail servers
  - Routers and switches
  - Custom stuff (snort, tcpdump, etc)

17

There are no hard and fast rules about which systems you should start with, but if you don't already have a good idea of what systems you care about, this list might serve. First and most important are your firewalls. If you've got logging turned off in your firewall for performance reasons ***turn it on*** because it is one of the most valuable sources of incident response data for post-attack cleanup. You should be logging outgoing connections as well as incoming connections; if you need to reduce the logging load slightly, turn off logging on denied traffic. ***Always*** log permits if you can.

Web servers are worth logging data off of simply because they are such a popular target of attack. Once again, many web servers have logging turned off for performance reasons. If you can't get logging enabled on the web servers, use a sniffer next to the web server and run dniff's "urlsnarf" module.

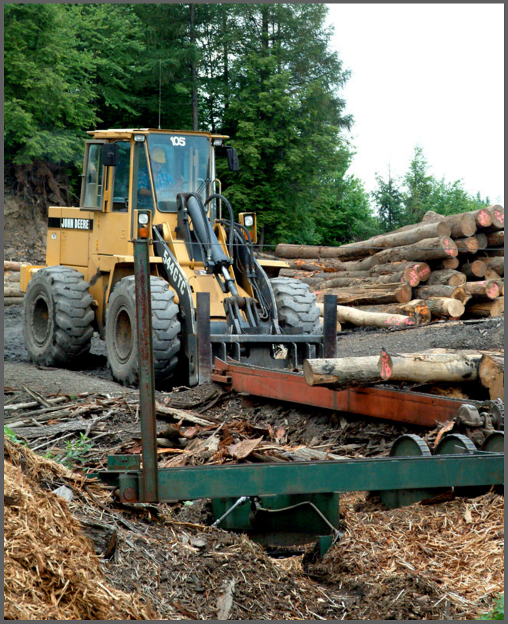
DHCP servers are crucial places for identifying new systems that appear and disappear and are also important for being able to correlate what piece of hardware had what address at what time. If you have one machine on your network with a good clock, it should be your DHCP server!

VPN servers and Web caches are important places to monitor if you can. Watch for weird accesses from your VPN cluster and spikes of traffic from your web caches.

Mail servers are *the* place to look for Email virus tracks (combined with your firewall's deny port 25 logs) - look for new systems originating SMTP.

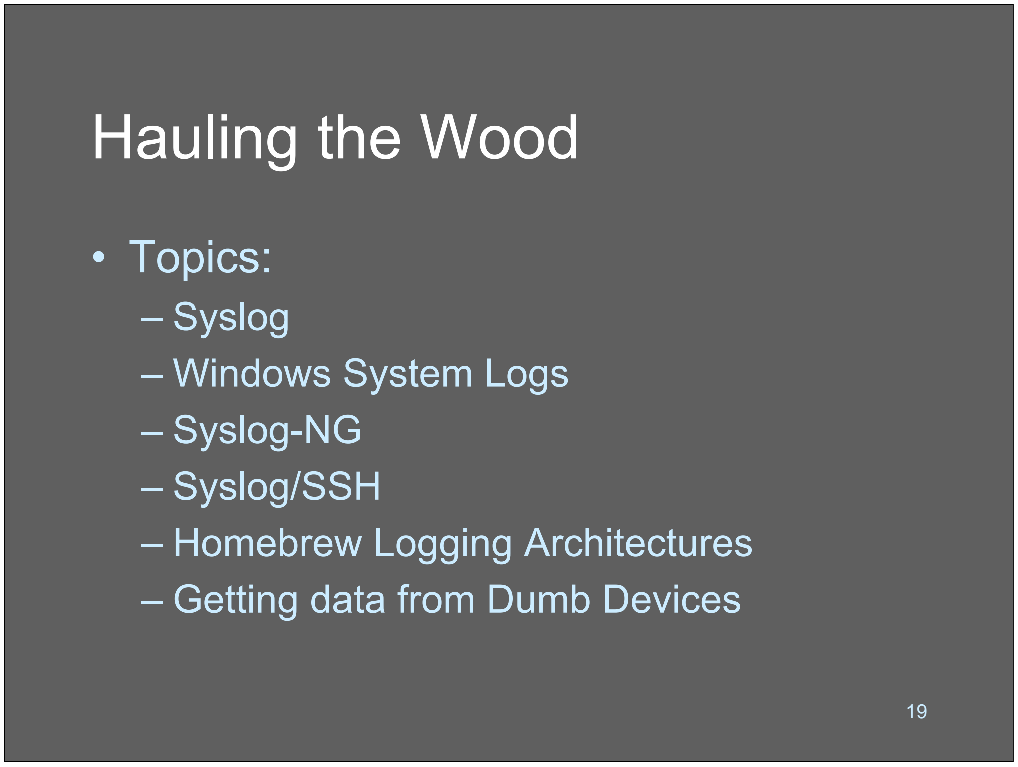
If you can afford to trace netflows at your routers, that's the last step in achieving logging nirvana!!

## Hauling the Wood



At the back of the sawmill is a large ramp made of steel I-beams with a series of toothed, geared wheels that move the logs into the building. The logs come in on trucks (big packets!) with about 50-60 logs apiece and are unloaded onto the ramp with the skid-loader.

Skid-loaders like the one pictured are truly awesome machines - they can lift a tree-trunk like it's a toothpick!

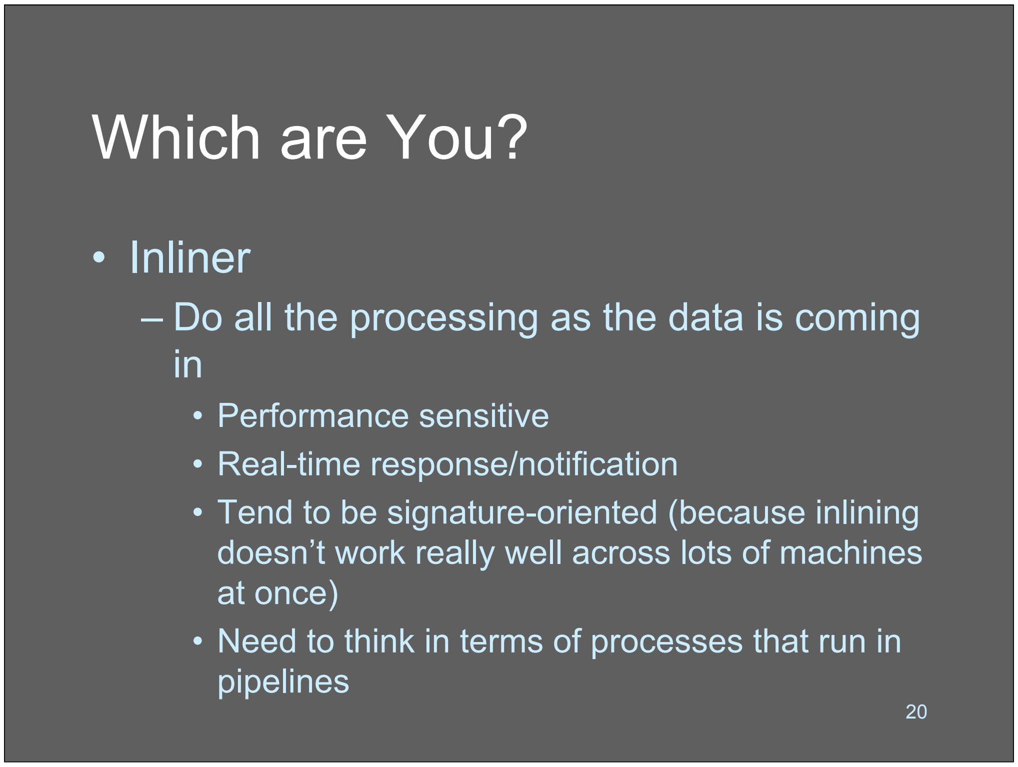


## Hauling the Wood

- Topics:
  - Syslog
  - Windows System Logs
  - Syslog-NG
  - Syslog/SSH
  - Homebrew Logging Architectures
  - Getting data from Dumb Devices

19

Now let's get a bit more technical. In this section we are going to work a few examples of various techniques for hauling log data around. We'll start with the classic syslogd and move up to Syslog-Ng and some useful tricks you can play with that package. Then we'll finish with some notes on how to build your own logging box.



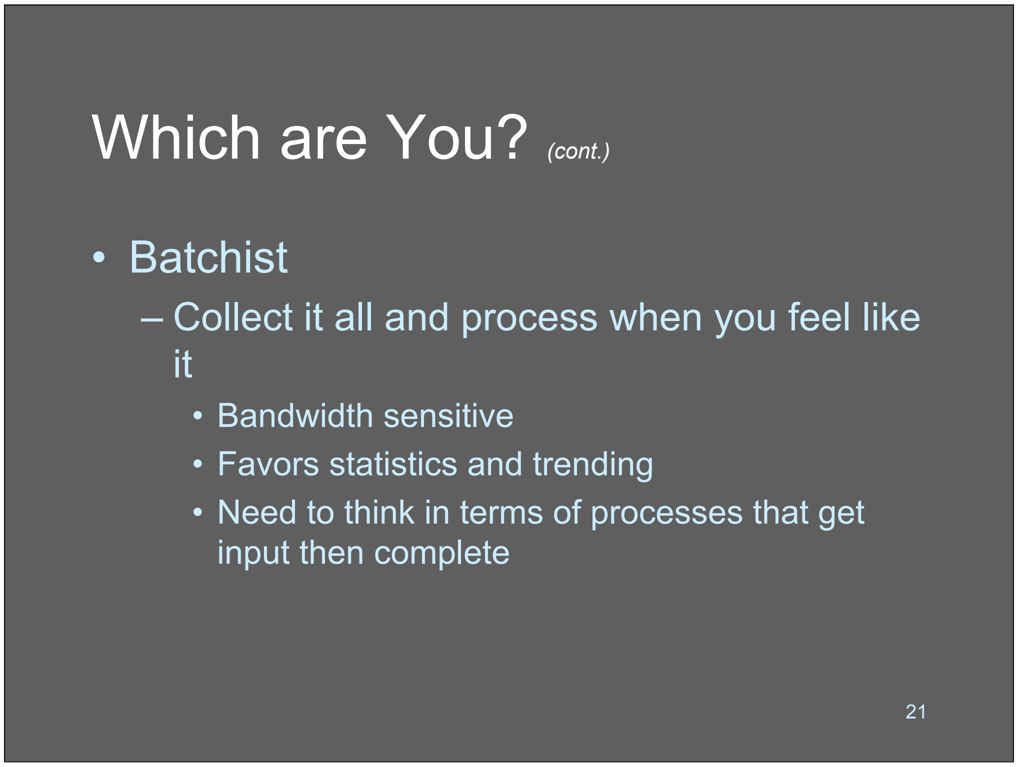
## Which are You?

- Inliner
  - Do all the processing as the data is coming in
    - Performance sensitive
    - Real-time response/notification
    - Tend to be signature-oriented (because inlining doesn't work really well across lots of machines at once)
    - Need to think in terms of processes that run in pipelines

20

There are 2 different approaches to processing log data. I call them "inliners" and "batchists" - whether you're an inliner or a batchist is just a matter of personal preference. It depends on how you want to handle your data.

Inliners want to process their data as it flows through the system. This approach has the advantage that you can find what's going on in "realtime." Inlining tends to be attractive to people who want to run managed security services or operational centers. Inlining involves building processes that work on the data in a pipeline, though usually the data hits the hard disk after being collected, and it's then streamed into the processing software.



## Which are You? *(cont.)*

- Batchist
  - Collect it all and process when you feel like it
    - Bandwidth sensitive
    - Favors statistics and trending
    - Need to think in terms of processes that get input then complete

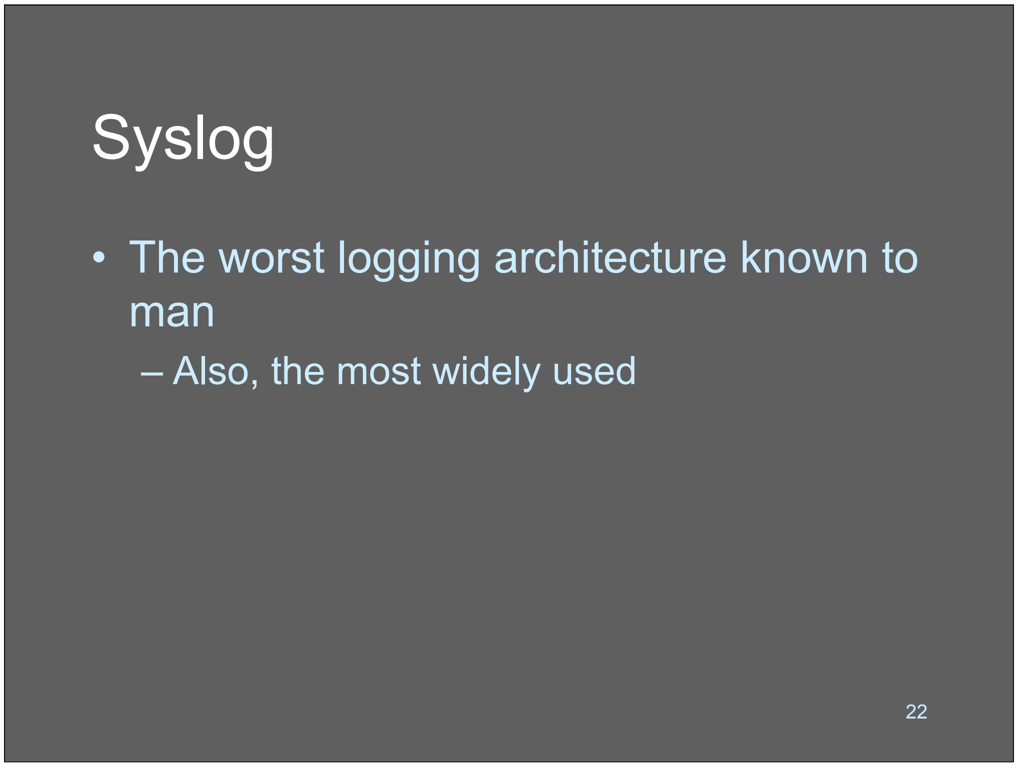
21

Batch processing log data is the most popular approach. The reason is probably because the tools (at least in a UNIX environment) collect their input and act upon it, then output some kind of additional data or analysis. If you're thinking of a typical UNIX pipeline, the input is usually a file consisting of a chunk of data gathered during a specific time interval. The easiest way to do this is for the system logging agent to collect data into a separate file for each hour (or day or whatever) at the completion of which time it's passed to a pipeline for processing.

The

file -> pipeline -> output file -> alert Email

approach is very popular, since it makes it fairly easy to manage data. To get various views of processes within time, you either need to split a bigger file apart (using a tool such as retail) or combine smaller files (e.g.: cat Jun-25-2004-12:* | pipeline)



## Syslog

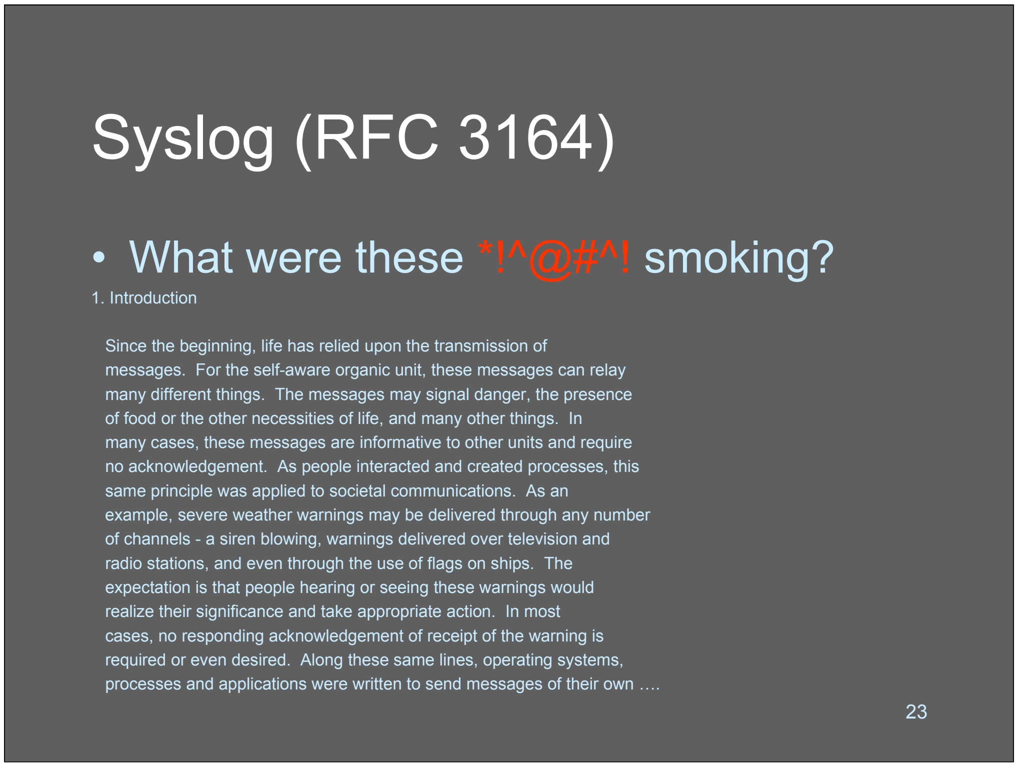
- The worst logging architecture known to man
  - Also, the most widely used

22

Just because lots of people use it, doesn't mean it's good.

On the other hand, it's better than *nothing*.

The interesting question is: how much?



## Syslog (RFC 3164)

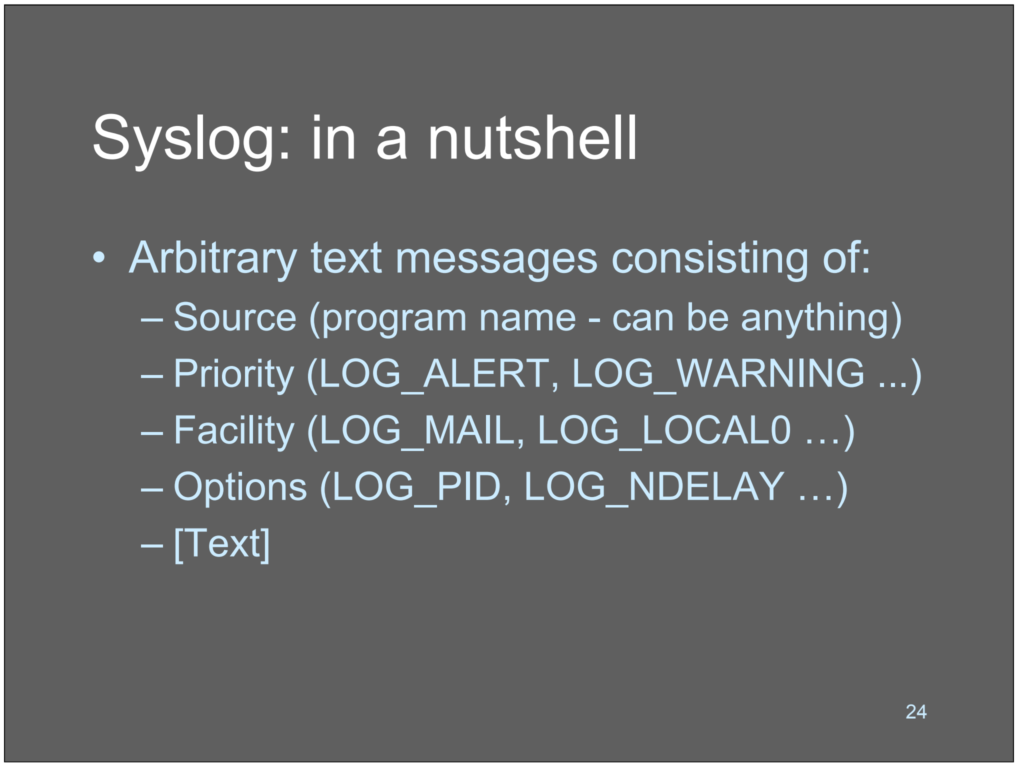
- What were these *!^@#^! smoking?

1. Introduction

Since the beginning, life has relied upon the transmission of
messages. For the self-aware organic unit, these messages can relay
many different things. The messages may signal danger, the presence
of food or the other necessities of life, and many other things. In
many cases, these messages are informative to other units and require
no acknowledgement. As people interacted and created processes, this
same principle was applied to societal communications. As an
example, severe weather warnings may be delivered through any number
of channels - a siren blowing, warnings delivered over television and
radio stations, and even through the use of flags on ships. The
expectation is that people hearing or seeing these warnings would
realize their significance and take appropriate action. In most
cases, no responding acknowledgement of receipt of the warning is
required or even desired. Along these same lines, operating systems,
processes and applications were written to send messages of their own ….

23

A couple of years ago, when I was implementing some code that had to interface with syslog, I decided to read RFC 3164. This was the moment when I stopped believing in the "standards process" and concluded that the IETF were just a bunch of clowns who used the standards process as an excuse to travel to strange places and drink too much diet coke.



## Syslog: in a nutshell

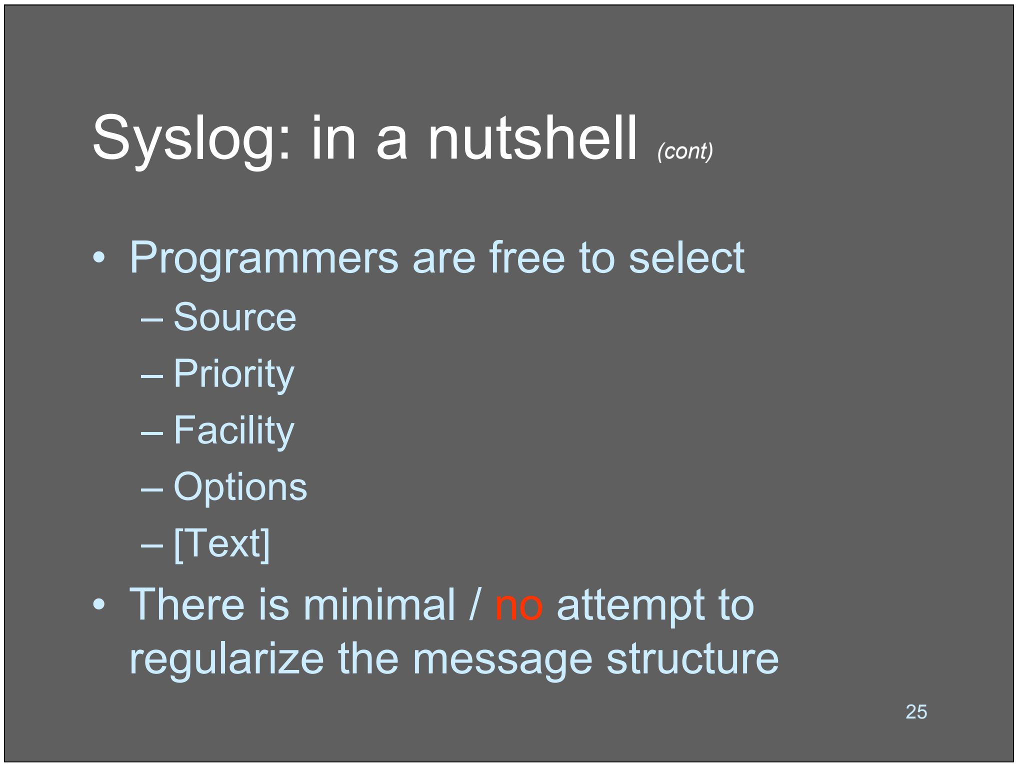
- Arbitrary text messages consisting of:
  - Source (program name - can be anything)
  - Priority (LOG_ALERT, LOG_WARNING ...)
  - Facility (LOG_MAIL, LOG_LOCAL0 …)
  - Options (LOG_PID, LOG_NDELAY …)
  - [Text]

24

Syslog messages are enclosed within a syslog *record* that consists of a binary 32-bit value followed by an arbitrary string of ASCII newline-terminated text. The “bottom” 3 bits of the 32-bit number are the priority and the remainder encode facilities. Facilities were originally intended to be useful for de-multiplexing the message upon reciept but there was no guidance provided as to how to do so, and users of the syslog(3) API pretty much did whatever they wanted.

The Options are not encoded in the message; they are used to control the client-side function of the API. LOG_PID, for example, includes the process-ID in the log message. LOG_NDELAY sends the message immediately, etc.

The beginning of the string (often) consists of a date and time stamp. If the client side API is the “standard” one the date stamp is added by the client side before the record is transmitted. This means, among other things, that it’s really easy to forge dates in syslog records.



## Syslog: in a nutshell *(cont)*

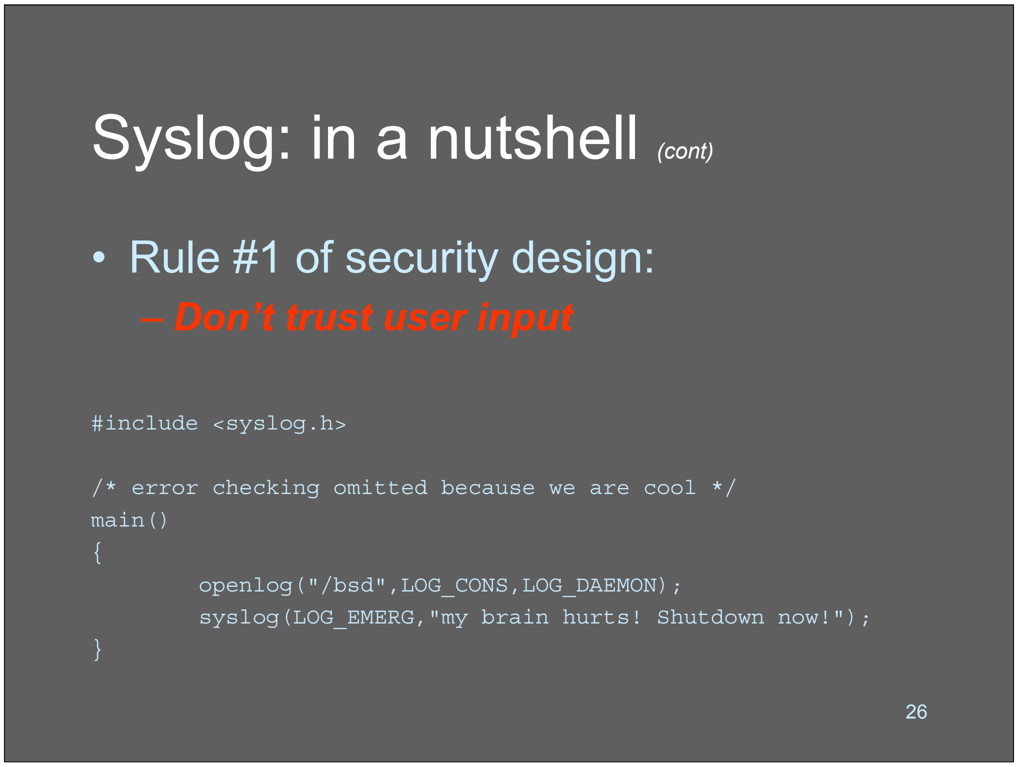
- Programmers are free to select
  - Source
  - Priority
  - Facility
  - Options
  - [Text]
- There is minimal / no attempt to regularize the message structure

25

When you're writing client-side code that wants to syslog a message, virtually everything is left to the programmer's discretion. If you want your message to be logged as coming from the "su(1)" program, that's fine. If you want your message to be ultra-high priority, that's fine, too.

I once saw an accidental denial-of-service attack in which a young programmer, who was debugging some code, decided to use syslog to issue debugging messages (instead of, say, using a debugger). He chose to use a low priority so as not to interfere with the system logs. *Un*fortunately, the syslog server was configured to not do anything with low priority messages. So the programmer decided to use printf() to debug  (instead of, say, using a debugger). The software actually shipped, with *dozens* of syslog calls in it, which slowed down all the systems on which it was installed because syslogd was thrashing as it tried to deal with all the low priority events that it was throwing away. The problem was discovered by one of the product's early users who had a syslog configuration that saved even low priority messages: his hard disk filled up.

Giving programmers the ability to just pick whatever they want, with no guidance, is a *bad design*.


## Syslog: in a nutshell *(cont)*

- Rule #1 of security design:
  - ***Don’t trust user input***

```
#include <syslog.h>

/* error checking omitted because we are cool */
main()
{
        openlog("/bsd",LOG_CONS,LOG_DAEMON);
        syslog(LOG_EMERG,"my brain hurts! Shutdown now!");
}
```

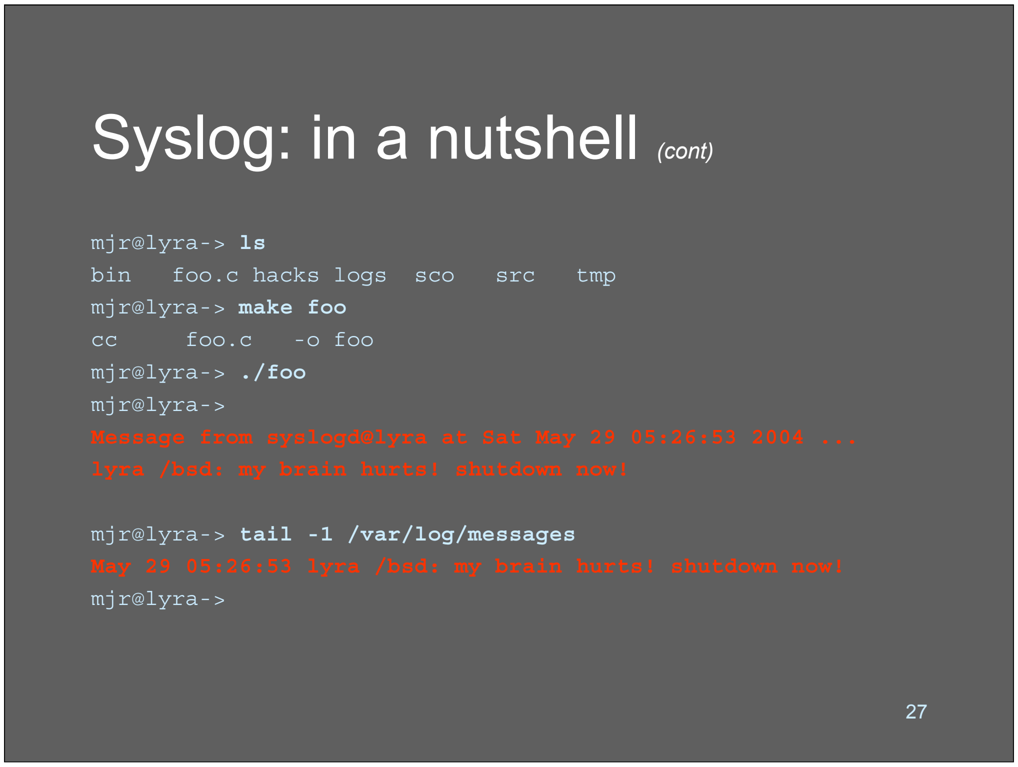
26

One of the fundamental tenets of security design is that user input is not trustworthy.

This code fragment (actually a working program with all the error checking removed to make it 3l33t) illustrates some of the problem with trusting user input from syslog.

Here we open a connection to syslogd and pass the client API a request that the message be logged to the console. We then issue a syslog message with a priority of *emergency* - which, on default syslog configurations, gets widely recorded.

This message could easily be made to look like an IDS alert or a crucial error. Or 10,000 of them, by just adding a `for` loop.



## Syslog: in a nutshell *(cont)*

```
mjr@lyra-> ls
bin   foo.c hacks logs  sco   src   tmp
mjr@lyra-> make foo
cc     foo.c   -o foo
mjr@lyra-> ./foo
mjr@lyra->
Message from syslogd@lyra at Sat May 29 05:26:53 2004 ...
lyra /bsd: my brain hurts! shutdown now!

mjr@lyra-> tail -1 /var/log/messages
May 29 05:26:53 lyra /bsd: my brain hurts! shutdown now!
mjr@lyra->
```

27

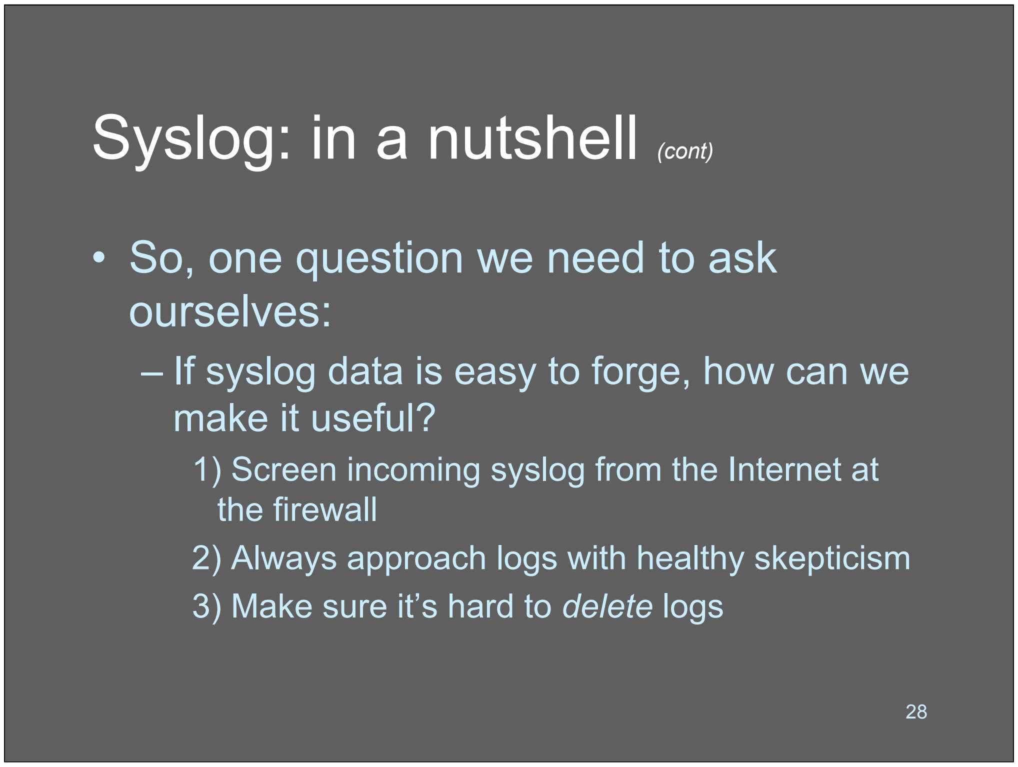
This shows building our silly little application, and what happens when you run it. Most syslog config files have a default rule that looks like:

```
*.err                                 root
*.notice;auth.debug                   root
*.alert                               root
*.emerg                               *
```

This tells syslogd to send the message to any terminals on which root is logged in, or "*" meaning "everyone on the system"

Back in the days of hard-wired terminals with escape codes (like VT100s) syslogd didn't know enough to strip non-printing characters. My first experience with syslog was on an old 4.2BSDsystem, where one of the undergrads wrote a program much like the one above and sent "^[2;9y" to all logged in users. Trivia question: does anyone know what that does to an old DEC VT100 terminal?

Very few UNIX systems are multiuser anymore, comparatively, so this is not as big a deal as it used to be; but you could probably get some sucker to fall for a message saying, "this is the system administrator; due to impending power outage, please issue the command rm -rf * and log out IMMEDIATELY"



## Syslog: in a nutshell *(cont)*

- So, one question we need to ask ourselves:
  - If syslog data is easy to forge, how can we make it useful?
    1) Screen incoming syslog from the Internet at the firewall
    2) Always approach logs with healthy skepticism
    3) Make sure it's hard to *delete* logs

28

Since we've shown that syslog data is not very hard to forge, have we also shown that it's useless?

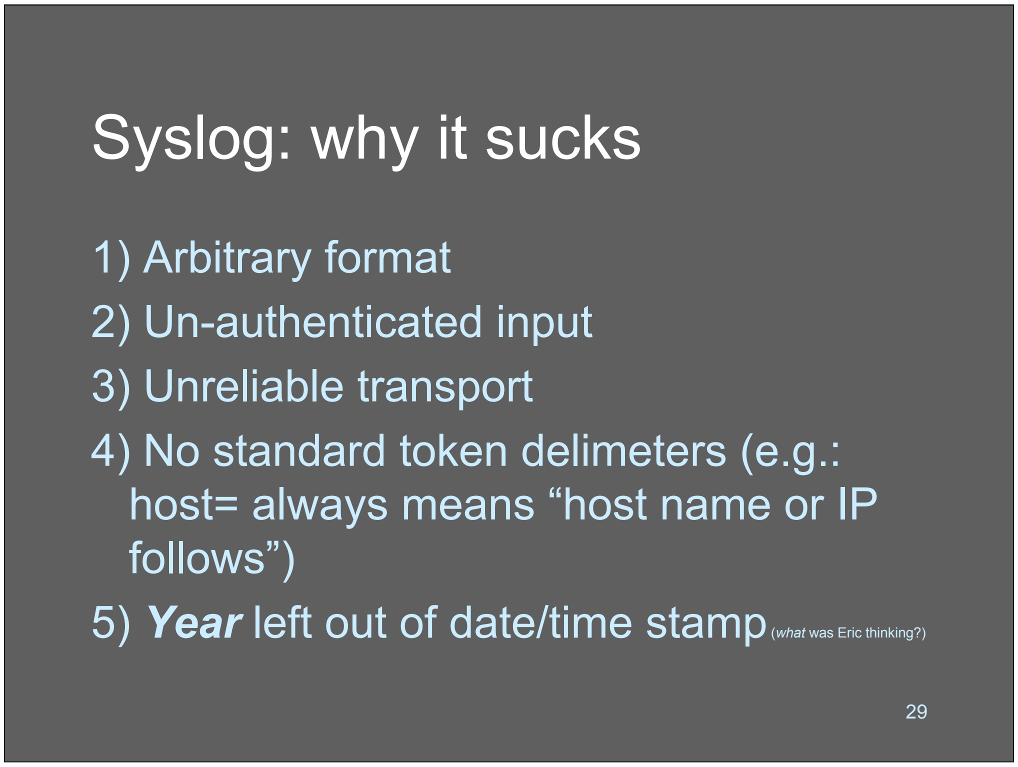
Not really. For most of the things a bad guy is going to want to do, they'd rather have *nothing* in the log than *something*. Accept the fact that things might be injected into your syslog and think of ways to reduce that likelihood. For example, if you have a syslog server that is *not* an aggregator, it should be running with syslogd configured to not accept UDP traffic. A bad guy on the host can still inject syslogs into the UNIX domain socket /dev/log but they'll have to log onto the host and that will leave some traces.

A bit of healthy skepticism is important to have if you're working with log data. If you see a message in syslog that one of your co-workers is trying to "su" your account, don't just run down the hall and start beating them. Make sure that's what's going on, *then* start the beatings. In general, you should treat syslog data as an initial discovery tool and an important pile of clues - never an open and shut case.

Most importantly, make sure it's hard to delete your logs. The most popular way of interfering with logs is to just zap them. If that happens, you can grep(1) your hard disk for them with some difficulty

```
strings /dev/rwd0a | \
grep '^[JFMASOND][aeupco][nbrynlgptyc] [0-9]+ [0-9]+:'
```

… and so forth… your mileage may vary...



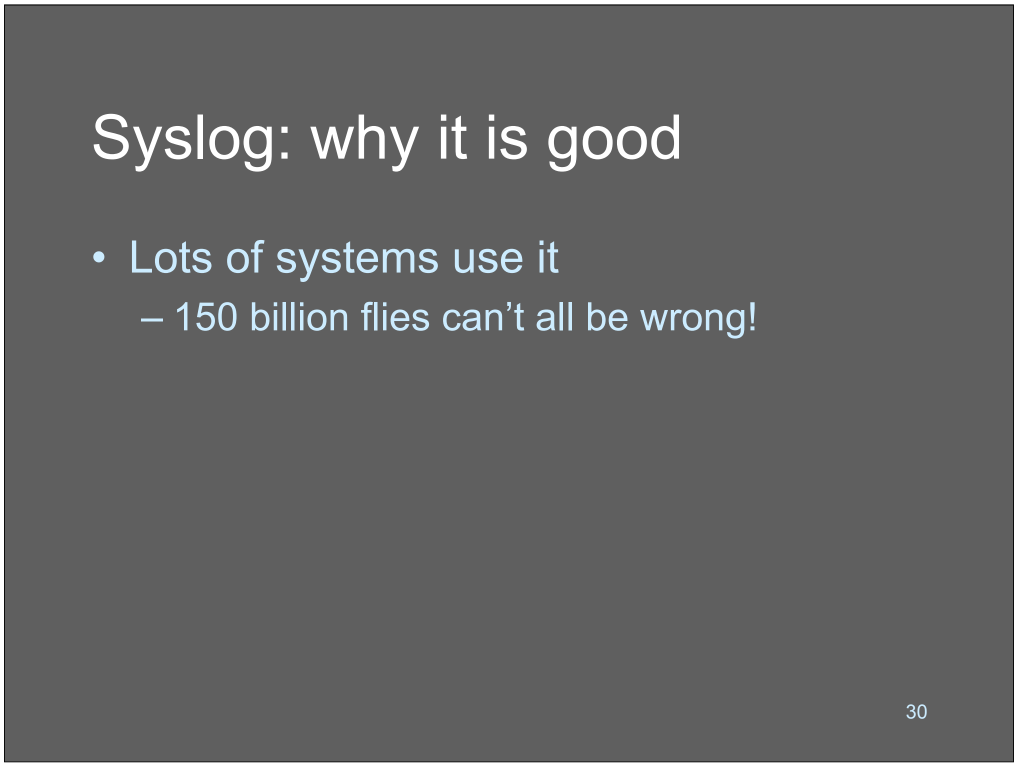
## Syslog: why it sucks

1) Arbitrary format
2) Un-authenticated input
3) Unreliable transport
4) No standard token delimeters (e.g.: host= always means “host name or IP follows”)
5) ***Year*** left out of date/time stamp (*what* was Eric thinking?)

29

By now you may be getting the idea that syslog is not a very good logging system. That’s a pretty accurate view of the situation.

You have my permission to be a little depressed about what you’re getting yourself into.


## Syslog: why it is good

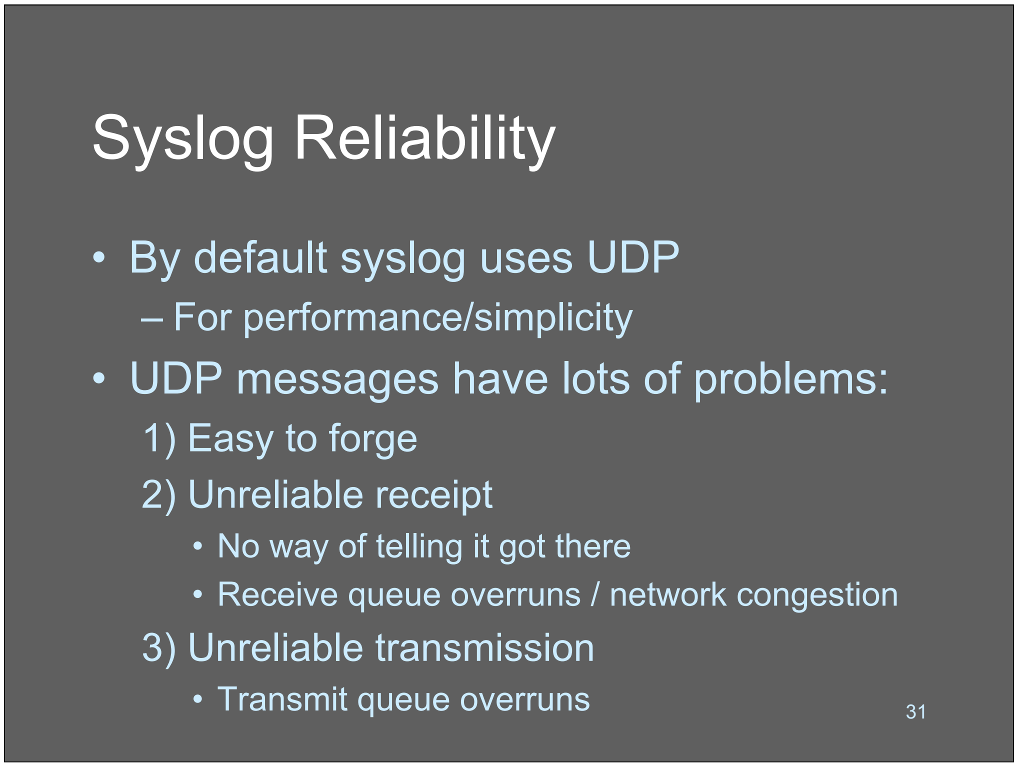
- Lots of systems use it
  - 150 billion flies can't all be wrong!

30

The good news about syslog is that it's widely supported!

In fact, the situation could be much worse! Eric did us all a big favor with syslog (in spite of all the stuff he left out) because if it weren't for syslog, UNIX machines would have 400 different files in /var/log as well as /usr/spool/uucp/log, /var/www/log (oh, wait, they have that…) and so forth.

"syslog may be *garbage* but it's *neatly stacked* garbage!" - anonymous



## Syslog Reliability

- By default syslog uses UDP
  - For performance/simplicity
- UDP messages have lots of problems:
  1) Easy to forge
  2) Unreliable receipt
     - No way of telling it got there
     - Receive queue overruns / network congestion
  3) Unreliable transmission
     - Transmit queue overruns

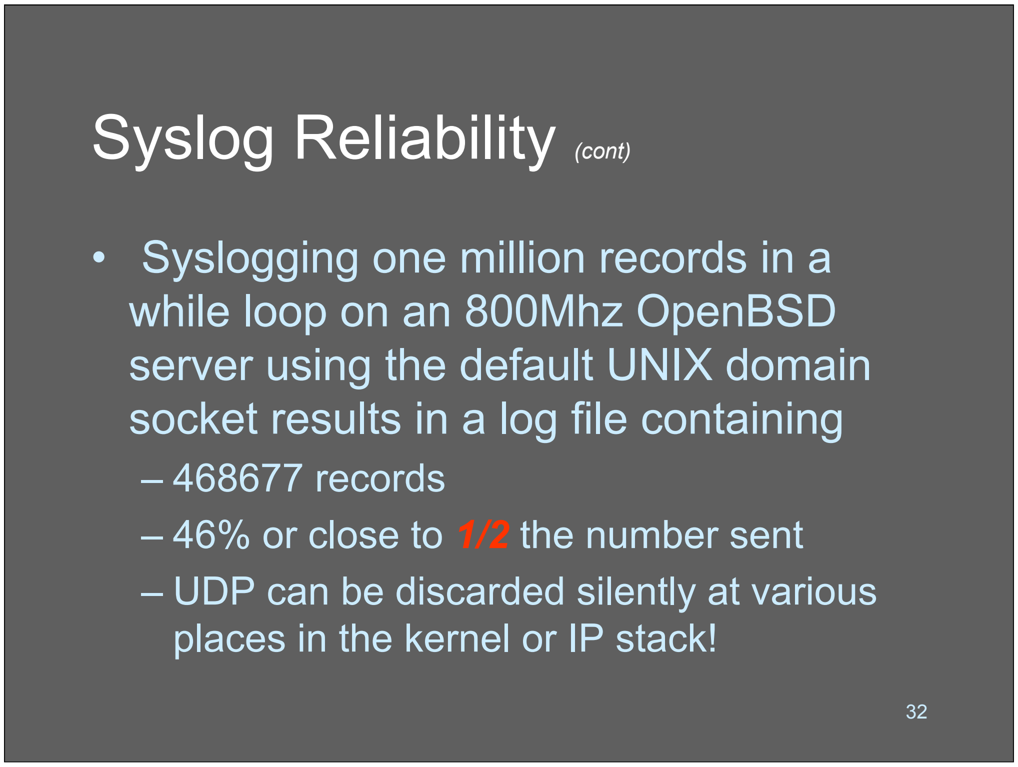
31

Another major problem with syslog is that it uses an unreliable mechanism for delivery.

Normally, when someone sees "UDP" and "unreliable" they think, "well, that's ***bad***" - but that only begins to scratch the surface of how bad UDP syslog truly is. UDP traffic can be discarded at will by any device or layer anyplace in the stack or between the two programs that are communicating. Worse still, there's no way to tell if the traffic did, in fact, get there.

So, usually, when we think of UDP we think of it as a protocol to use for 'quick and dirty' transmission of transient data. In other words, it's the exact wrong thing to use for logging data.

When syslog was written, it was before the days of massive web servers that could handle huge numbers of live TCP connections. If old syslogds had used connected TCP streams, the syslog server would have been much much more complicated, since it would have had to select() across, basically, one TCP stream per app running on the machine. Making it an in-kernel service (with the potential to then use a protected file) would have been too much work, apparently.



## Syslog Reliability *(cont)*

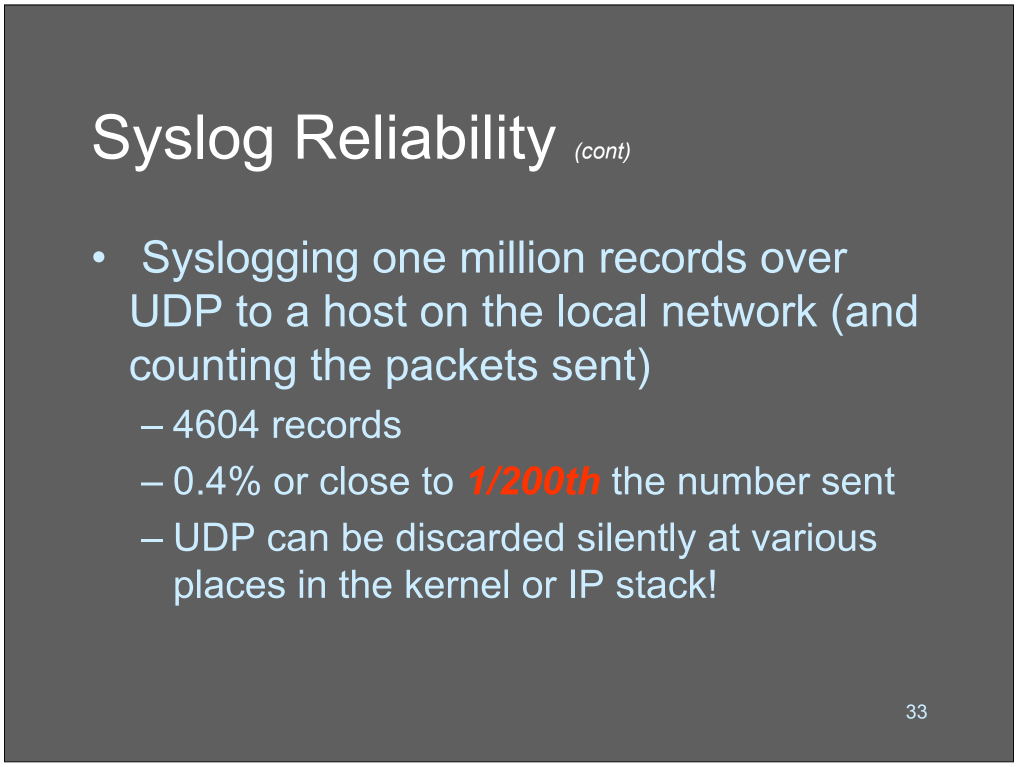
- Syslogging one million records in a while loop on an 800Mhz OpenBSD server using the default UNIX domain socket results in a log file containing
  - 468677 records
  - 46% or close to ***1/2*** the number sent
  - UDP can be discarded silently at various places in the kernel or IP stack!

32

A bunch of us were on a mailing list and were discussing the reliability of syslog. It turned out that nobody had ever actually *measured* it - we were all just guessing. So I did some tests and the results were worse than many of us expected.

Syslogging in a tight loop resulted in 50% losses even using the local UNIX domain socket. This test took approximately 15 seconds to run, which comes to about 67,000 messages/second - about 6,000 ***times*** a "normal" syslog load.

Still, this is not very good.



## Syslog Reliability *(cont)*

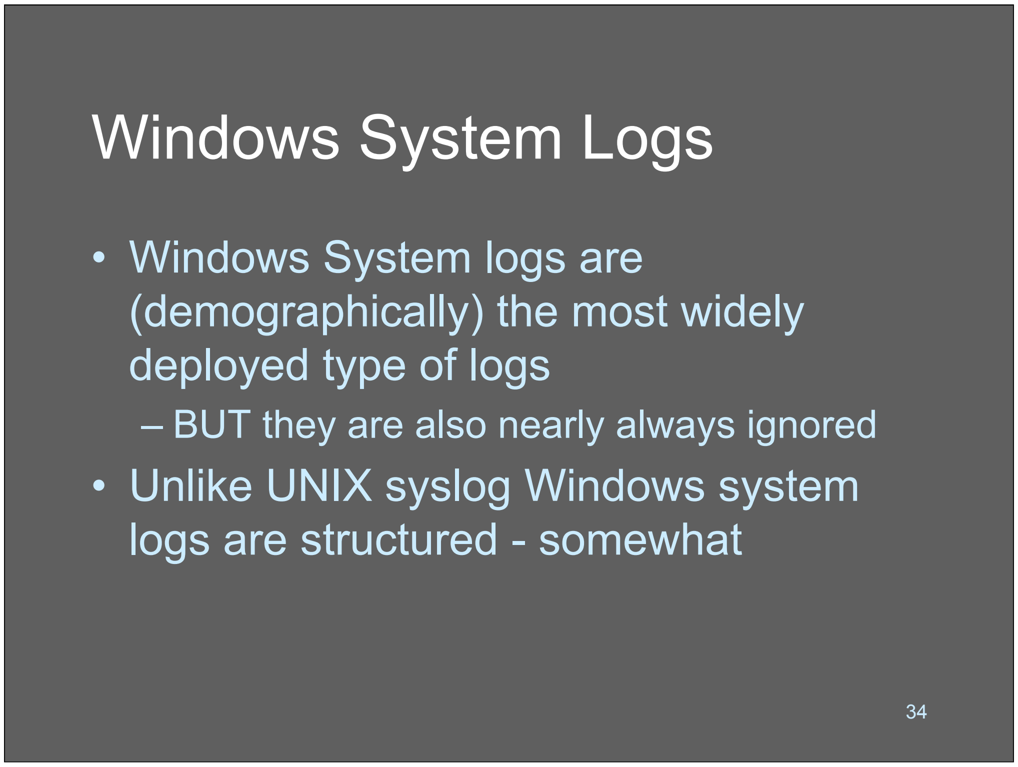
- Syslogging one million records over UDP to a host on the local network (and counting the packets sent)
  - 4604 records
  - 0.4% or close to ***1/200th*** the number sent
  - UDP can be discarded silently at various places in the kernel or IP stack!

33

Using the same test over a UDP datagram to a different machine across a network the results were *dramatically* ***worse***. Of 1,000,000 records, only 4604 arrived, which is 0.4%

4606 messages in 10 seconds is 46 messages/second, which is about 4 times a “normal” syslog load. This raises the interesting question: if you have a syslog server that is being sent traffic by 200 clients, what makes you think that more than a fraction of the traffic is showing up in the log?

This is a really good argument against having a massive central log server, but rather having a lot of smaller aggregators that collect locally and then forward to the central log server over a reliable, encrypted pipe.



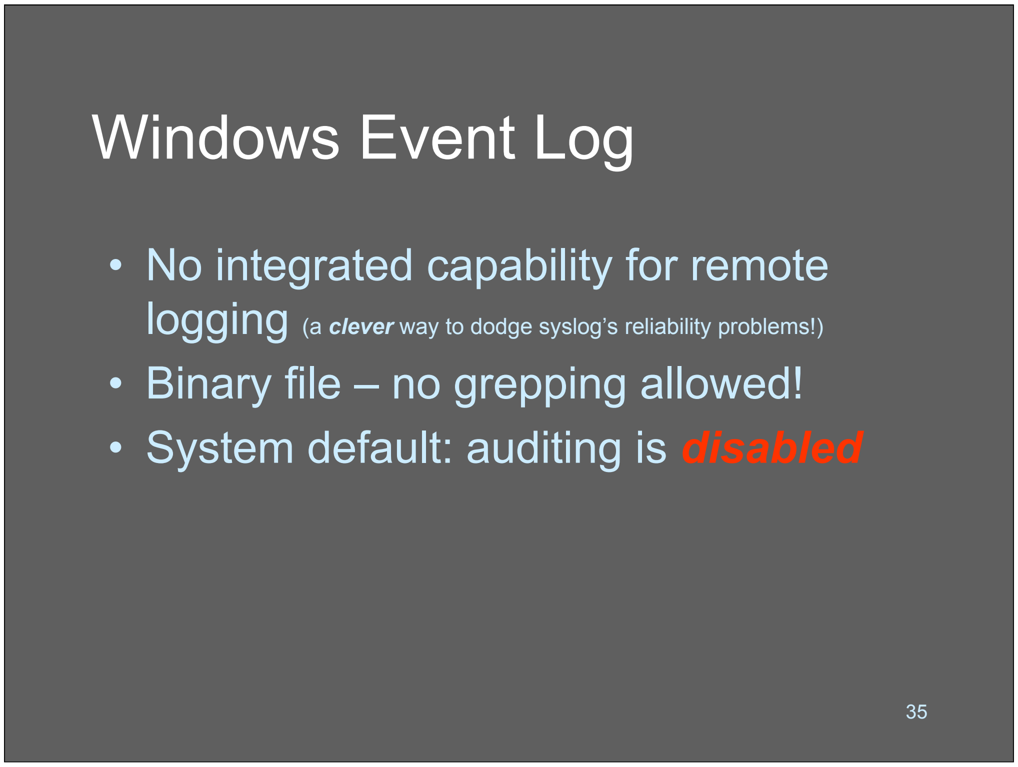
## Windows System Logs

- Windows System logs are (demographically) the most widely deployed type of logs
  - BUT they are also nearly always ignored
- Unlike UNIX syslog Windows system logs are structured - somewhat

34

Windows' system logging architecture is arguably better than UNIX', but it's much less widely used. The words "enterprise computing" don't appear to mean much in Redmond, because Windows logging system completely omits any mechanisms for built-in aggregation.

The good news about Windows logging is that the data is at least somewhat structured. But, the good news is tempered with bad news - the data is still arbitrary and cryptic.



## Windows Event Log

- No integrated capability for remote logging (a ***clever*** way to dodge syslog's reliability problems!)
- Binary file – no grepping allowed!
- System default: auditing is ***disabled***

35

This section of the course is based on Cory Scott's fabulous document, "Dealing with Windows NT Event Logs," at
`http://www.securityfocus.com/focus/microsoft/nt/log1.html`

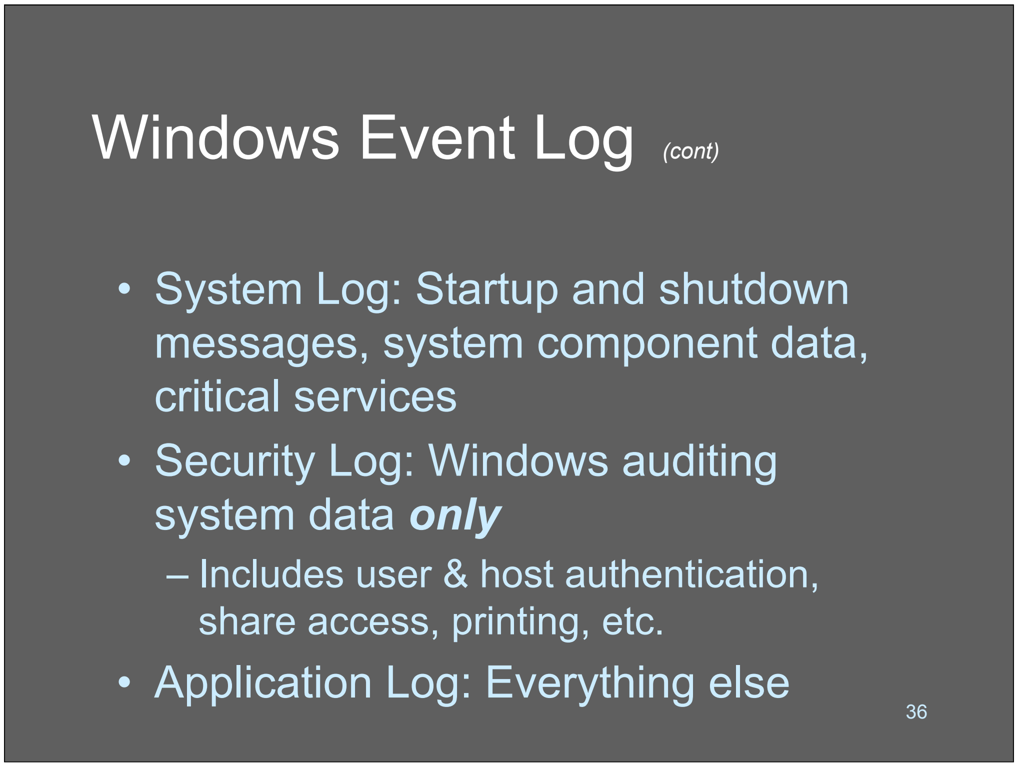
and the course he presented at SANS 2000.

To access Event Log information on WinNT, under the "Start" button, select "Administrative Tools (Common)," then select "Event Viewer."

For further information on configuring NT Event Log, see "Selecting WinNT/4.0 event log settings" at

`http://www.cert.org/security-improvement/implementations/i041.03.html`

The only way to prevent a Windows system from overwriting its log files is to configure it to stop logging when it runs out of disk space, or to shut the machine down entirely. Neither of these is likely to be ideal, since you either start losing audit information or lose the machine's availability altogether. This is one great reason for configuring your Windows systems to send Event Log data to a loghost, even though it takes a bit of work. You'll no longer need to worry about losing data if your Event Log fills up.



## Windows Event Log *(cont)*

- System Log: Startup and shutdown messages, system component data, critical services
- Security Log: Windows auditing system data ***only***
  - Includes user & host authentication, share access, printing, etc.
- Application Log: Everything else

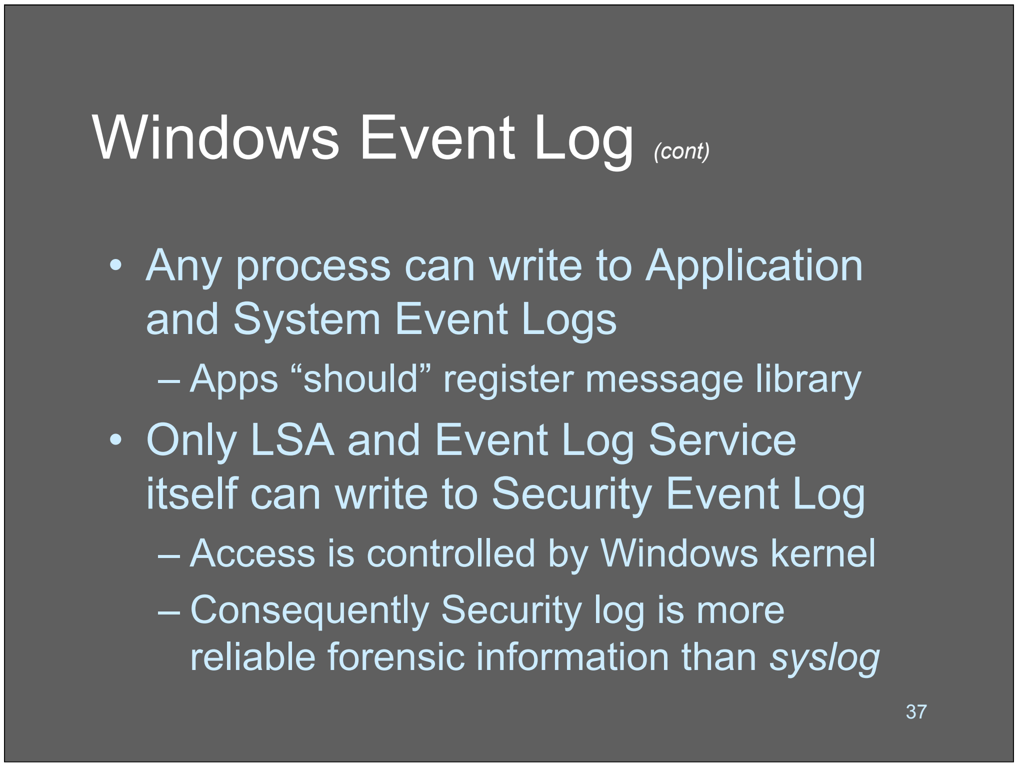
36

My favorite tidbit from Cory's essay on NT Event Logs: "Unfortunately, due to an inexplicable decision by Microsoft, a failed logon to a domain from an NT workstation will only log a security event to the workstation (if auditing for logon events is enabled) attempting to connect, rather than to a domain controller. For that reason alone, it is necessary to audit failed logons on every workstation that is on your domain." Our theory is that since the domain logon failed, the local workstation doesn't have the privilege required to log to the domain controller's Security Log. Ugh. Luckily, this deficiency is fixed in Windows 2000.

The three binary log files are stored in %SystemRoot%\system32\config folder on WinNT, as *SysEvent.Evt, SecEvent.Evt,* and *AppEvent.Evt.*

Any application that registers itself to the Event Log service can write audit data to the Application Log. To see which applications are registered, check this Registry location:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Eventlog\Applica
tion

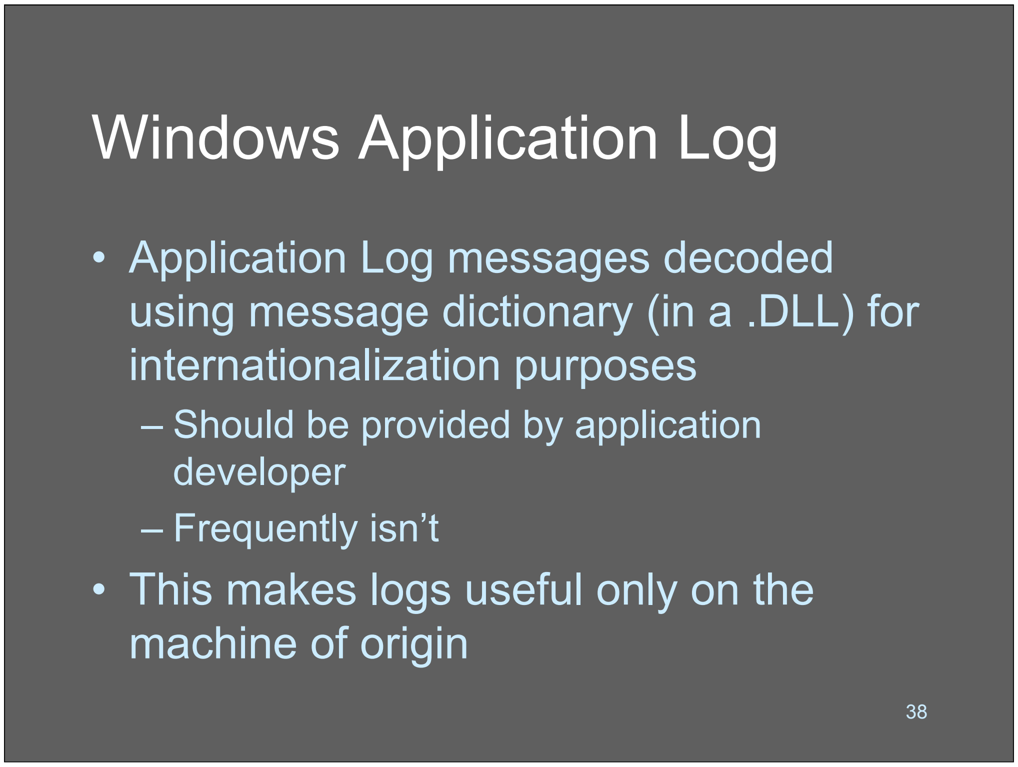


## Windows Event Log *(cont)*

- Any process can write to Application and System Event Logs
  - Apps “should” register message library
- Only LSA and Event Log Service itself can write to Security Event Log
  - Access is controlled by Windows kernel
  - Consequently Security log is more reliable forensic information than *syslog*

37

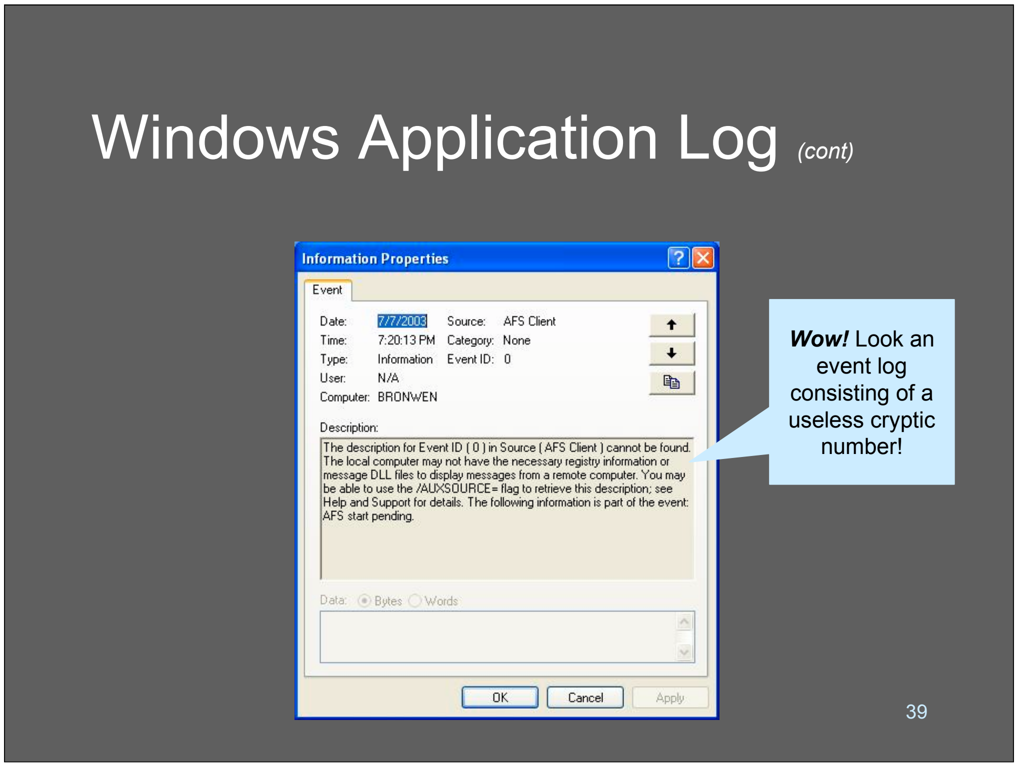
The Event Log API uses an error message library to translate from an event ID to a human-readable message. Well behaved Windows developers create meaningful message libraries, and their applications register themselves with the Event Log service to provide that translation capability. Care to imagine what badly behaved Windows developers do?


## Windows Application Log

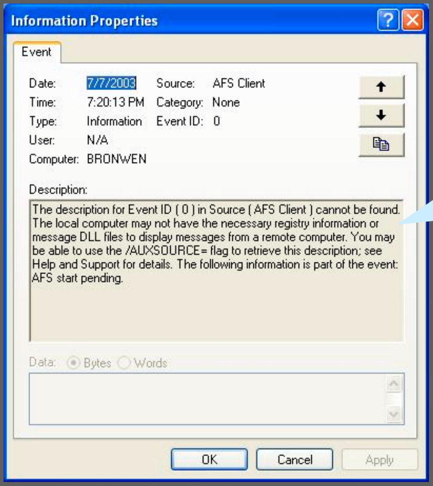
- Application Log messages decoded using message dictionary (in a .DLL) for internationalization purposes
  - Should be provided by application developer
  - Frequently isn't
- This makes logs useful only on the machine of origin

38

If you intend to do anything with Windows application logs, you will need to decode them into text (or some other format) on the machine where they originated. This was intended as a mechanism for internationalization - programmers write a "message dictionary" that maps coded error numbers from the log file into matching messages in a DLL. If you want the error messages to come out in French, you install the French DLL, etc. Do you think there might be a market for error DLLs in Klingon or Latin?



## Windows Application Log *(cont)*



Information Properties

Event

```
Date:      7/7/2003      Source:    AFS Client
Time:      7:20:13 PM    Category:  None
Type:      Information   Event ID:  0
User:      N/A
Computer:  BRONWEN
```

Description:

```
The description for Event ID ( 0 ) in Source ( AFS Client ) cannot be found.
The local computer may not have the necessary registry information or
message DLL files to display messages from a remote computer. You may
be able to use the /AUXSOURCE= flag to retrieve this description; see
Help and Support for details. The following information is part of the event:
AFS start pending.
```

Data: (● Bytes) ○ Words

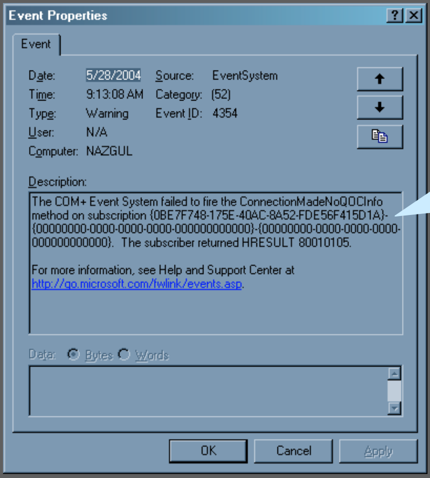
***Wow!*** Look an event log consisting of a useless cryptic number!

39

Anyone who has used software for more than a few years will have at some point or another encountered the dreaded “stupid error message” in which some programmer forgot to remove a debug message from a production application.

This classic example of cryptic uselessness is the result of the logging system trying to decode an error code for which there is no string mapping.

## Windows Application Log *(cont)*



Event Properties

Event

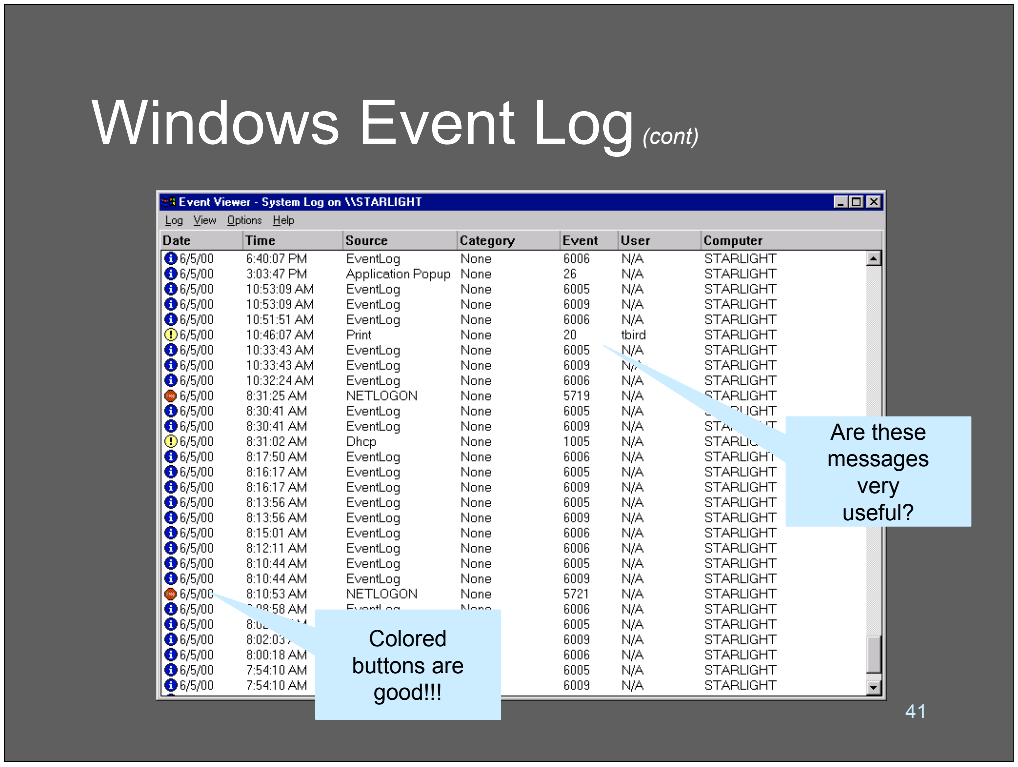
```
Date:      5/28/2004   Source:    EventSystem
Time:      9:13:08 AM  Category:  (52)
Type:      Warning     Event ID:  4354
User:      N/A
Computer:  NAZGUL

Description:
The COM+ Event System failed to fire the ConnectionMadeNoQOCInfo
method on subscription {0BE7F748-175E-40AC-8A52-FDE56F415D1A}-
{00000000-0000-0000-0000-000000000000}-{00000000-0000-0000-0000-
000000000000}.  The subscriber returned HRESULT 80010105.

For more information, see Help and Support Center at
http://go.microsoft.com/fwlink/events.asp.
```

Even when you have the details is it helpful?

40

"Oh, no!! The 80010105 again!"

Do you want to make a bet that the helpdesk can tell us what an 80010105 is?"



## Windows Event Log *(cont)*

Event Viewer - System Log on \\STARLIGHT

| Date | Time | Source | Category | Event | User | Computer |
|---|---|---|---|---|---|---|
| ℹ 6/5/00 | 6:40:07 PM | EventLog | None | 6006 | N/A | STARLIGHT |
| ℹ 6/5/00 | 3:03:47 PM | Application Popup | None | 26 | N/A | STARLIGHT |
| ℹ 6/5/00 | 10:53:09 AM | EventLog | None | 6005 | N/A | STARLIGHT |
| ℹ 6/5/00 | 10:53:09 AM | EventLog | None | 6009 | N/A | STARLIGHT |
| ℹ 6/5/00 | 10:51:51 AM | EventLog | None | 6006 | N/A | STARLIGHT |
| ⚠ 6/5/00 | 10:46:07 AM | Print | None | 20 | tbird | STARLIGHT |
| ℹ 6/5/00 | 10:33:43 AM | EventLog | None | 6005 | N/A | STARLIGHT |
| ℹ 6/5/00 | 10:33:43 AM | EventLog | None | 6009 | N/A | STARLIGHT |
| ℹ 6/5/00 | 10:32:24 AM | EventLog | None | 6006 | N/A | STARLIGHT |
| ⛔ 6/5/00 | 8:31:25 AM | NETLOGON | None | 5719 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:30:41 AM | EventLog | None | 6005 | N/A | [illegible] |
| ℹ 6/5/00 | 8:30:41 AM | EventLog | None | 6009 | N/A | [illegible] |
| ⚠ 6/5/00 | 8:31:02 AM | Dhcp | None | 1005 | N/A | [illegible] |
| ℹ 6/5/00 | 8:17:50 AM | EventLog | None | 6006 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:16:17 AM | EventLog | None | 6005 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:16:17 AM | EventLog | None | 6009 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:13:56 AM | EventLog | None | 6005 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:13:56 AM | EventLog | None | 6009 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:15:01 AM | EventLog | None | 6006 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:12:11 AM | EventLog | None | 6006 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:10:44 AM | EventLog | None | 6005 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:10:44 AM | EventLog | None | 6009 | N/A | STARLIGHT |
| ⛔ 6/5/00 | 8:10:53 AM | NETLOGON | None | 5721 | N/A | STARLIGHT |
| ℹ 6/5/00 | [illegible] | [illegible] | [illegible] | 6006 | N/A | STARLIGHT |
| ℹ 6/5/00 | [illegible] | [illegible] | [illegible] | 6005 | N/A | STARLIGHT |
| ℹ 6/5/00 | [illegible] | [illegible] | [illegible] | 6009 | N/A | STARLIGHT |
| ℹ 6/5/00 | 8:00:18 AM | [illegible] | [illegible] | 6006 | N/A | STARLIGHT |
| ℹ 6/5/00 | 7:54:10 AM | [illegible] | [illegible] | 6005 | N/A | STARLIGHT |
| ℹ 6/5/00 | 7:54:10 AM | [illegible] | [illegible] | 6009 | N/A | STARLIGHT |

Are these messages very useful?

Colored buttons are good!!!

41

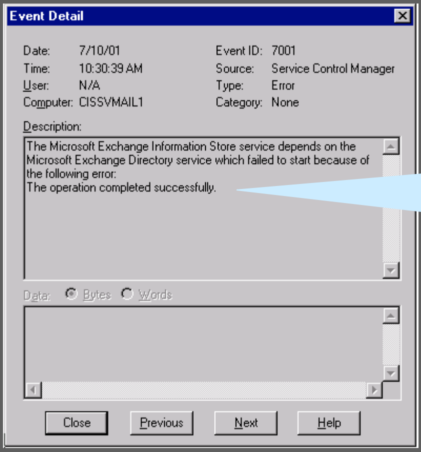
The severity of an event is indicated by the icon on the far left hand side of the window:

•Blue indicates diagnostic information that requires no action

•Yellow indicates a warning that may require attention, but does not severely impact the ability to use the system.

•Red indicates an error that will probably make the system or service unavailable.

## Windows Event Log *(cont)*



Event Detail

Date: 7/10/01
Time: 10:30:39 AM
User: N/A
Computer: CISSVMAIL1

Event ID: 7001
Source: Service Control Manager
Type: Error
Category: None

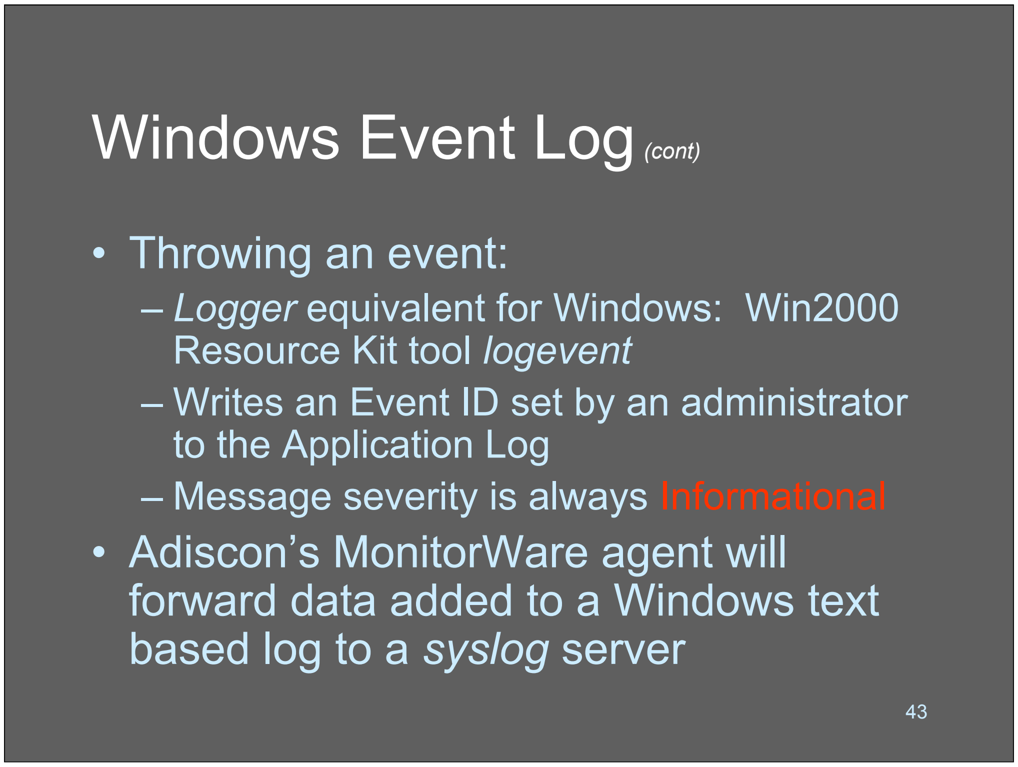
Description:

The Microsoft Exchange Information Store service depends on the Microsoft Exchange Directory service which failed to start because of the following error:
The operation completed successfully.

UNIX geeks apparently aren't the only ones that can log stupid messages!!

42

This is tbird's favorite error message. According to former student Craig Woods (AT&T Professional Services), this message is created due to improper inter-process communication. The Information Store service spawns a shell (the Service Control Manager), whose only goal in life is to start the Microsoft Exchange Directory Service. The Directory Service process has failed for some reason – the ultimate cause of the problems with the Information Store. And the Information Store knows enough to verify that the process upon which it's dependent is up and running. But unfortunately, instead of querying the Directory Service and discovering that all was lost, it queries the Service Control Manager, which has successfully accomplished its mission. So the Information Store continues on its merry way with its own start-up process, only to fall over when it can't get to the Directory Service.



## Windows Event Log *(cont)*

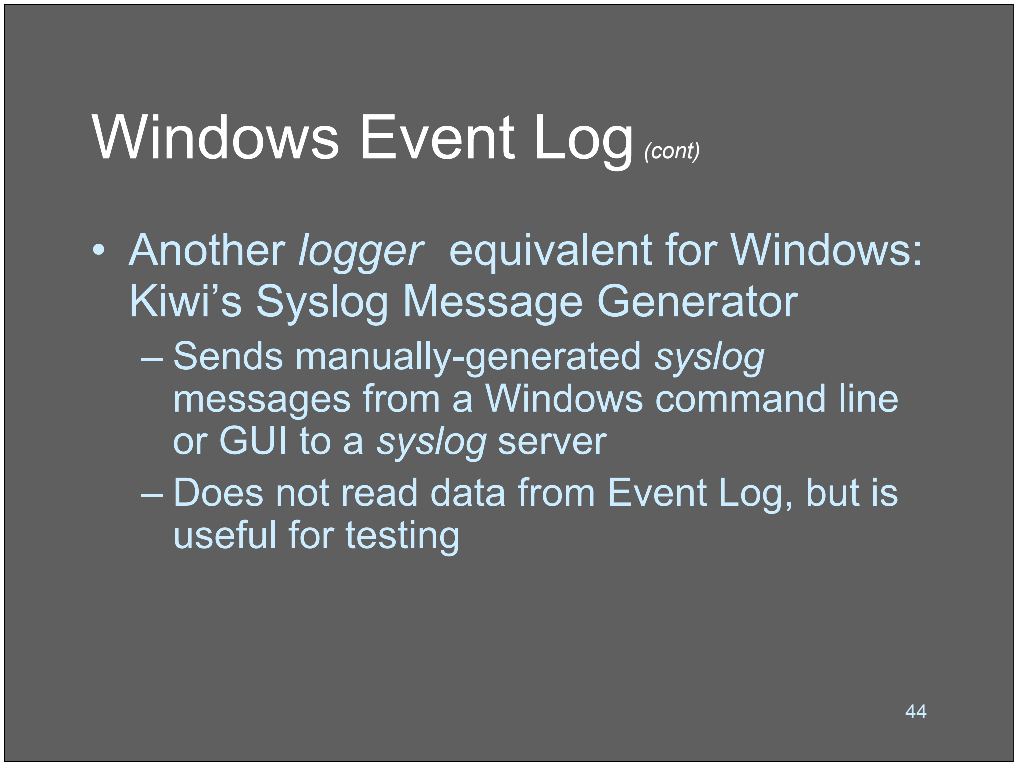
- Throwing an event:
  - *Logger* equivalent for Windows:  Win2000 Resource Kit tool *logevent*
  - Writes an Event ID set by an administrator to the Application Log
  - Message severity is always Informational
- Adiscon's MonitorWare agent will forward data added to a Windows text based log to a *syslog* server

43

There's information on using *logevent* at
`http://support.microsoft.com/support/kb/articles/Q131/0/08.asp.`

For no adequately explained reason, this utility creates Event Log messages with the severity level *informational **only***.  According to Microsoft, "…because these Events are generated by the user, it was felt that it is sufficient to put these in the log as Information Type messages only."  Those silly users would never want to write their own Warnings or Errors.

Adiscon has released a tool that will monitor Windows text files – such as Web logs produced by IIS – and forward them to a central syslog collector:
`http://www.mwagent.com`



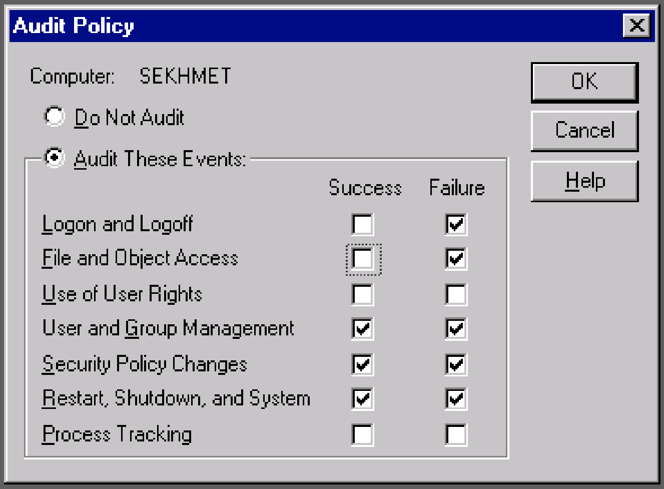
## Windows Event Log *(cont)*

- Another *logger* equivalent for Windows: Kiwi's Syslog Message Generator
  - Sends manually-generated *syslog* messages from a Windows command line or GUI to a *syslog* server
  - Does not read data from Event Log, but is useful for testing

44

For more info: `http://www.kiwi-enterprises.com/info_sysloggen.htm`

## WinNT Audit Configuration


Audit Policy

Computer: SEKHMET

Do Not Audit [✓ Audit These Events:]

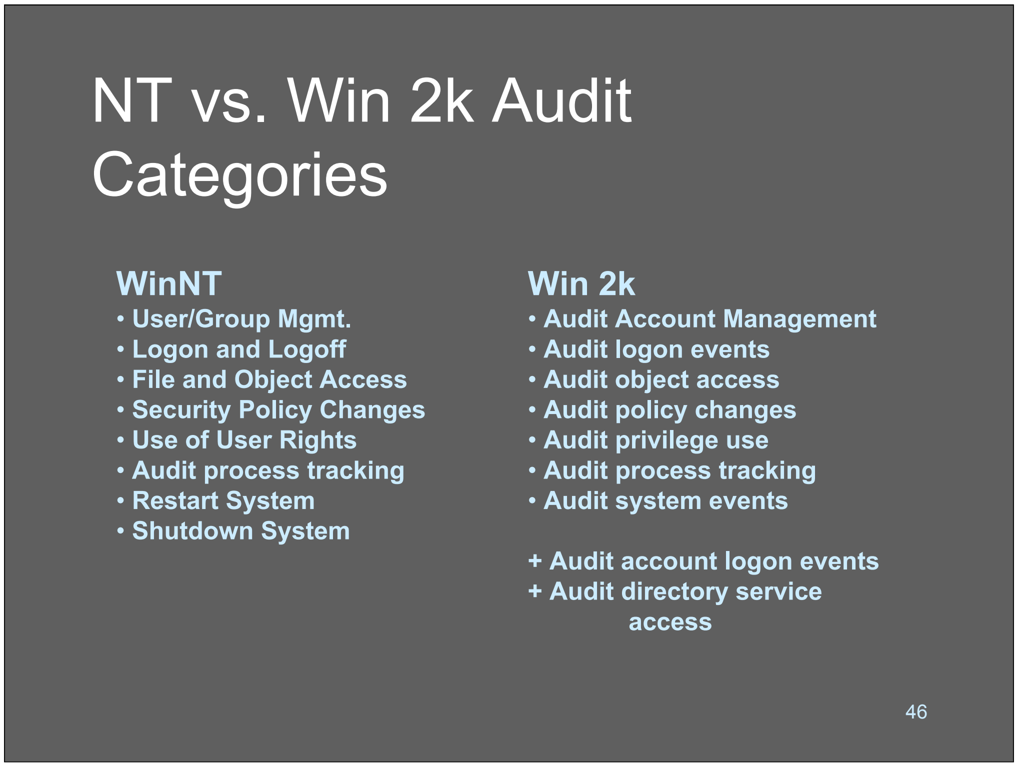
| | Success | Failure |
|---|---|---|
| Logon and Logoff | ☐ | ✓ |
| File and Object Access | ☐ | ✓ |
| Use of User Rights | ☐ | ☐ |
| User and Group Management | ✓ | ✓ |
| Security Policy Changes | ✓ | ✓ |
| Restart, Shutdown, and System | ✓ | ✓ |
| Process Tracking | ☐ | ☐ |

45

To edit your audit policy on Windows NT, go to the Start menu and select "Administrative Tools (Common)." Then select "User Manager," and under the "Policies" menu on the Task Bar, select "Audit." Clear as mud.

On XP Click Start, click Control Panel, click Performance and Maintenance, and then click Administrative Tools. Double-click Local Security Policy. In the left pane, double-click Local Policies to expand it. In the left pane, click Audit Policy to display the individual policy settings in the right pane. Double-click Audit object access. To audit successful access of specified files, folders and printers, select the Success check box. To audit unsuccessful access to these objects, select the Failure check box. To enable auditing of both, select both check boxes. Click OK ***Requires XP Professional!***

No wonder a lot of people never configure windows logging...



## NT vs. Win 2k Audit Categories

**WinNT**
- User/Group Mgmt.
- Logon and Logoff
- File and Object Access
- Security Policy Changes
- Use of User Rights
- Audit process tracking
- Restart System
- Shutdown System

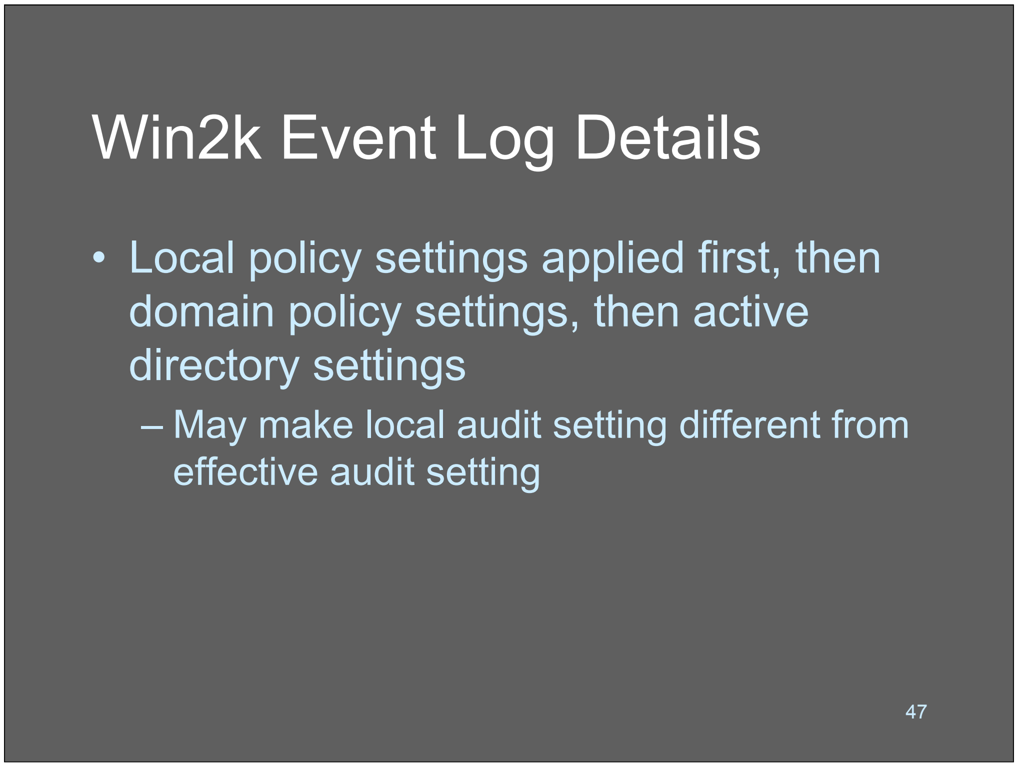
**Win 2k**
- Audit Account Management
- Audit logon events
- Audit object access
- Audit policy changes
- Audit privilege use
- Audit process tracking
- Audit system events

+ Audit account logon events
+ Audit directory service access

46

There are considerable differences between the audit configuration options in NT and Win2K. Similarly, there are big differences between 2K and XP as well as XP Pro.

Maybe syslog isn’t so bad after all!



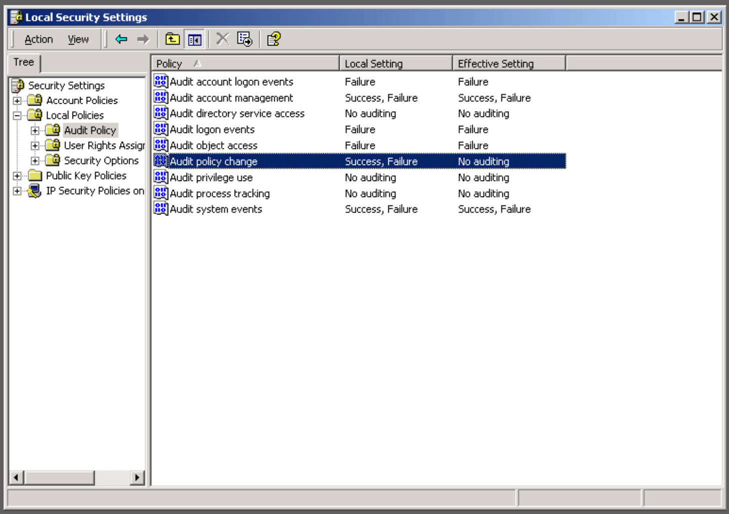
## Win2k Event Log Details

- Local policy settings applied first, then domain policy settings, then active directory settings
  - May make local audit setting different from effective audit setting

47

Best practices guidelines for Windows 2003 server:
`http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/proddocs/entserver/sag_SEconceptsImpAudBP.asp`

## Win2k Audit Configuration

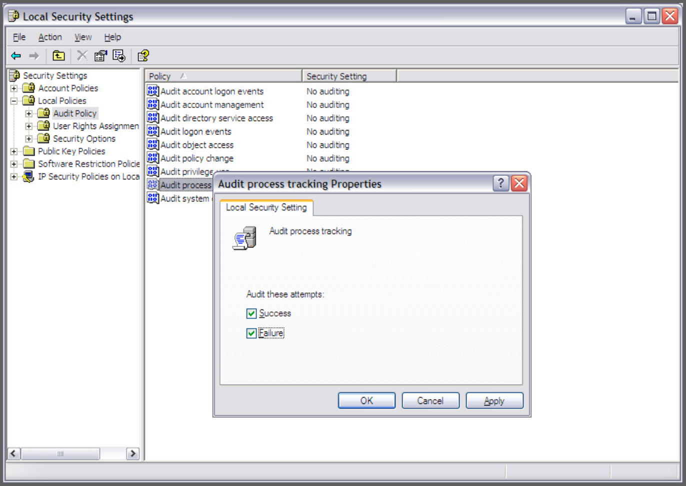


| Policy | Local Setting | Effective Setting |
|---|---|---|
| Audit account logon events | Failure | Failure |
| Audit account management | Success, Failure | Success, Failure |
| Audit directory service access | No auditing | No auditing |
| Audit logon events | Failure | Failure |
| Audit object access | Failure | Failure |
| Audit policy change | Success, Failure | No auditing |
| Audit privilege use | No auditing | No auditing |
| Audit process tracking | No auditing | No auditing |
| Audit system events | Success, Failure | Success, Failure |

48

To edit audit policies on Windows 2000, bring up the Control Panel, and select "Administrative Tools (Common)."  Then click on Security Settings, Local Security Policy, Local Policies, Audit Policy.

## WinXP Audit Configuration


Local Security Settings

File Action View Help

- Security Settings
  - Account Policies
  - Local Policies
    - Audit Policy
    - User Rights Assignmen
    - Security Options
  - Public Key Policies
  - Software Restriction Policie
  - IP Security Policies on Loca

| Policy | Security Setting |
|---|---|
| Audit account logon events | No auditing |
| Audit account management | No auditing |
| Audit directory service access | No auditing |
| Audit logon events | No auditing |
| Audit object access | No auditing |
| Audit policy change | No auditing |
| Audit privilege | |
| Audit process | |
| Audit system | |

Audit process tracking Properties

Local Security Setting

Audit process tracking

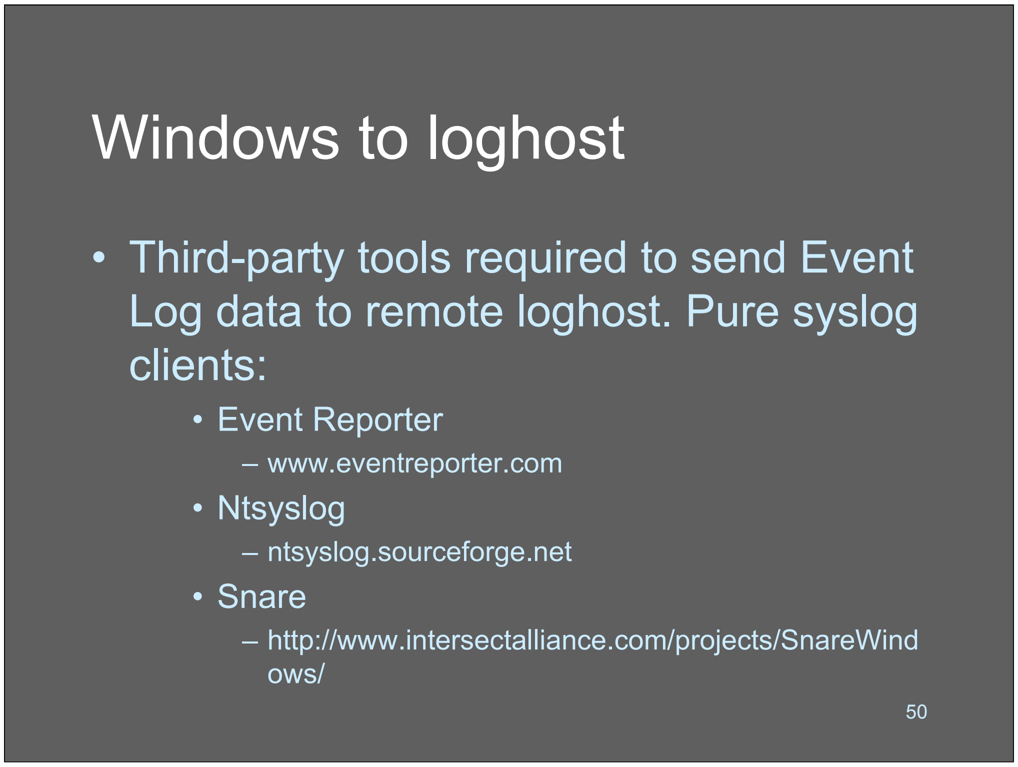
Audit these attempts:

[✓] Success

[✓] Failure

OK Cancel Apply

To edit audit policies on Windows XP, bring up the Control Panel, and select "Administrative Tools" Then choose Security Settings, Local Security Policy, Local Policies, Audit Policy - just like Win2K.

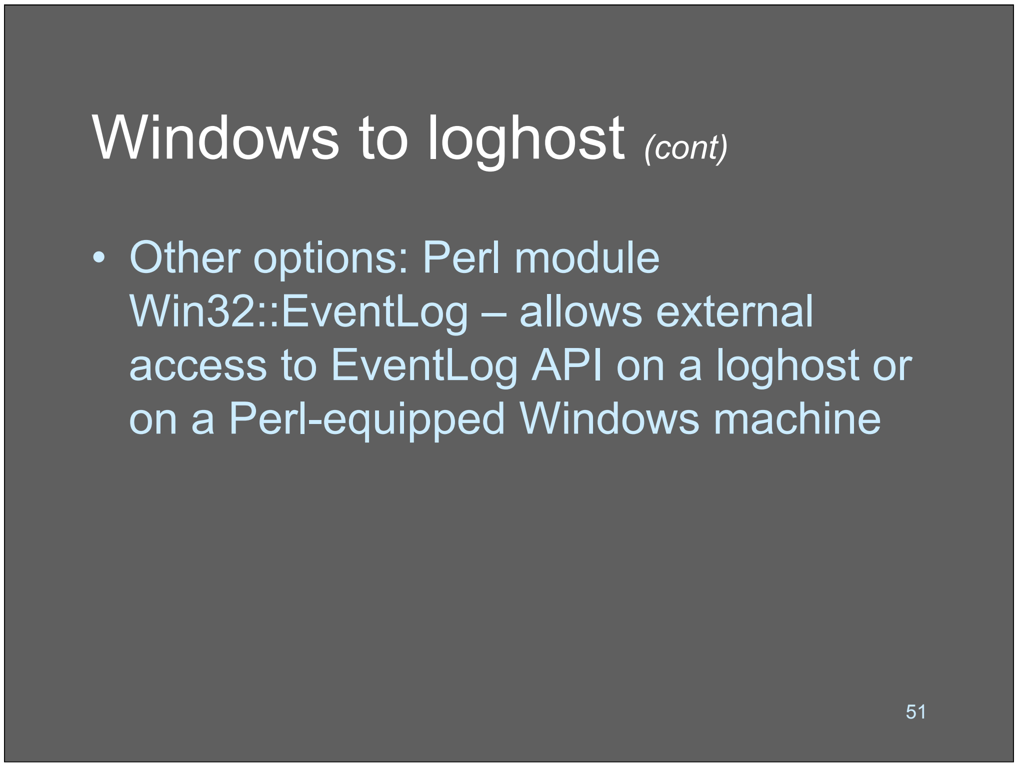


## Windows to loghost

- Third-party tools required to send Event Log data to remote loghost. Pure syslog clients:
  - Event Reporter
    - www.eventreporter.com
  - Ntsyslog
    - ntsyslog.sourceforge.net
  - Snare
    - http://www.intersectalliance.com/projects/SnareWindows/

50

EventReporter and Snare both provide a graphical interface as well as command line or registry edit capability.  NTsyslog is command line only and is not (apparently) being maintained any more. Eventreporter is a commercial product and Snare is available under the Gnu Public License.

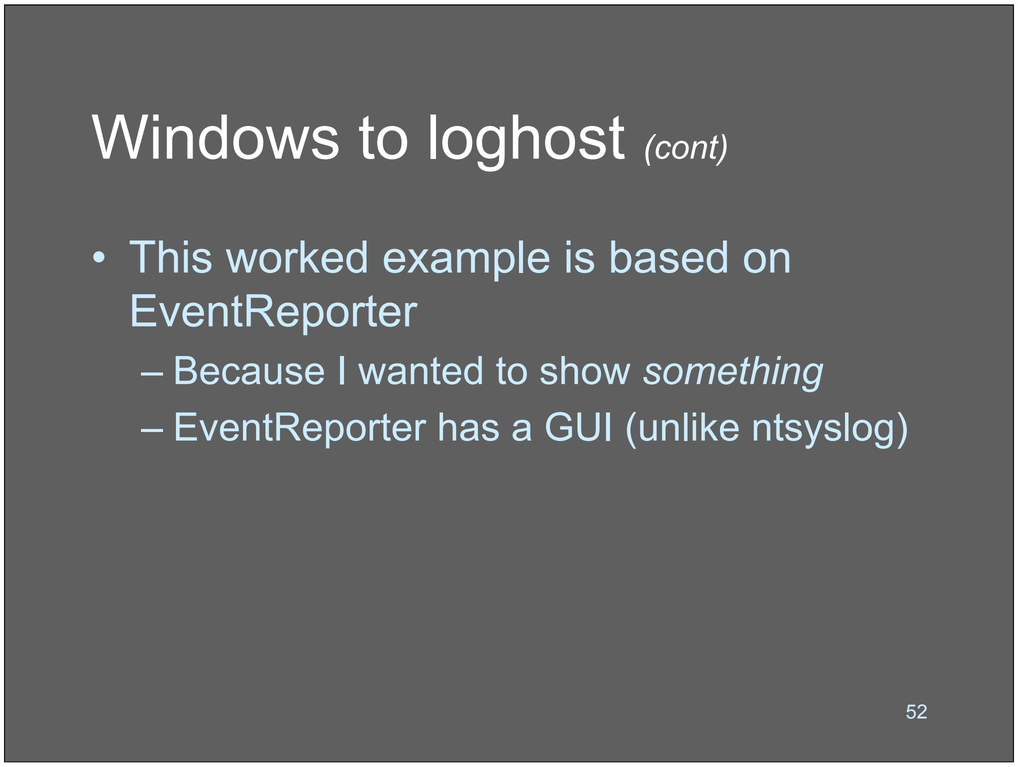


## Windows to loghost *(cont)*

- Other options: Perl module Win32::EventLog – allows external access to EventLog API on a loghost or on a Perl-equipped Windows machine

51

If you're a perl hacker, there is a perl module that handles interpreting Windows Event Log. To use it, you'll need perl on the windows machine as well as the DLLs necessary to decode the Event Log.



## Windows to loghost *(cont)*

- This worked example is based on EventReporter
  - Because I wanted to show *something*
  - EventReporter has a GUI (unlike ntsyslog)

52

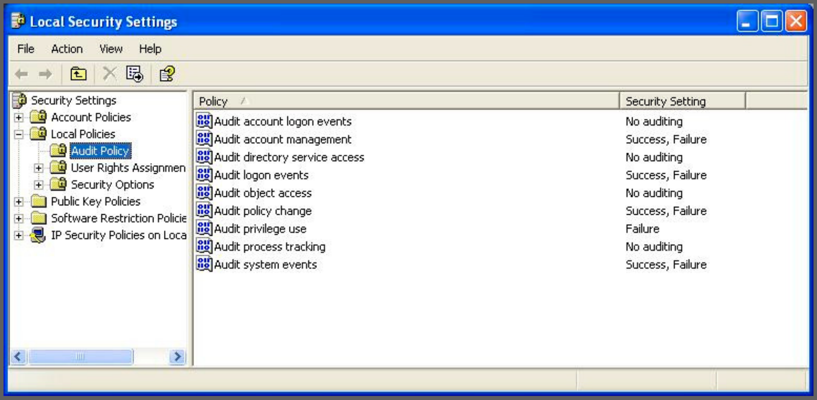
EventReporter and some related tools (such as an NT *syslog* server) are available at `http://www.eventreporter.com`. It's priced starting at $29 per server, with volume discounts. EventReporter includes two components, a configuration client and the engine that translates and forwards Event Log data to the *syslog* server.

Information on Win32::EventLog is available at

`http://search.cpan.org/doc/GSAR/libwin32-0.16/EventLog/EventLog.pm`. The module itself is available at

`http://www.cpan.org/authors/id/GSAR/libwin32-0.16.zip`.

## Windows Audit Policy

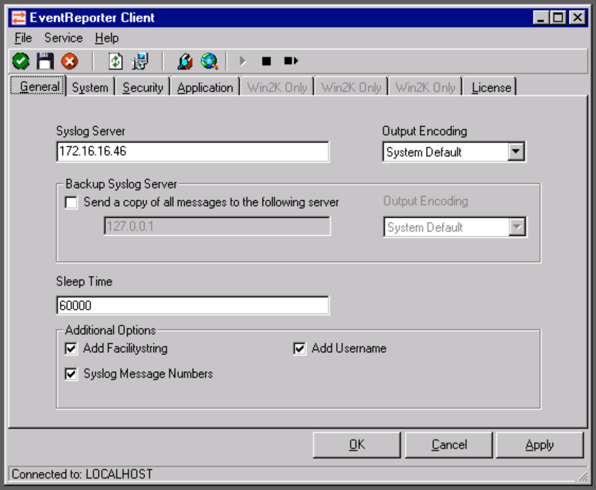


| Policy | Security Setting |
|---|---|
| Audit account logon events | No auditing |
| Audit account management | Success, Failure |
| Audit directory service access | No auditing |
| Audit logon events | Success, Failure |
| Audit object access | No auditing |
| Audit policy change | Success, Failure |
| Audit privilege use | Failure |
| Audit process tracking | No auditing |
| Audit system events | Success, Failure |

53

This is an exam,ple default logging policy for a stand-alone Windows XP Professional system.  It's not part of a domain (so there's no reason to audit the account login category), and it doesn't provide any public services (that I know of).

## Windows to loghost *(cont)*


EventReporter Client
File Service Help
General | System | Security | Application | Win2K Only | Win2K Only | Win2K Only | License
OK Cancel Apply

Syslog Server: 172.16.16.46

Output Encoding: System Default

Backup Syslog Server

☐ Send a copy of all messages to the following server: 127.0.0.1

Output Encoding: System Default

Sleep Time: 60000

Additional Options

☑ Add Facilitystring

☑ Add Username

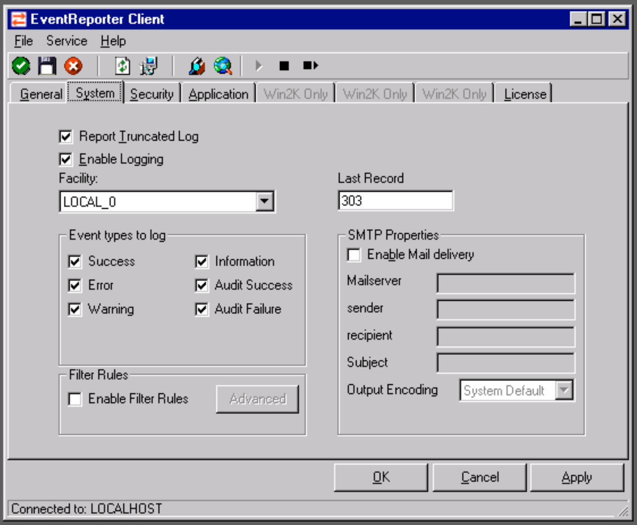
☑ Syslog Message Numbers

Connected to: LOCALHOST

54

After you've installed EventReporter, start the Client application (as Administrator) to configure things to send data to your central loghost. In this example, the loghost IP address is 172.16.16.46. The *sleep interval* configures the time period over which EventReporter processes the Event Log. It's given in milliseconds, and defaults (as shown) to 1 minute. Shorter periods are supported, but they increase the processing requirements for the application (which are otherwise minimal even on a busy NT server).

## Windows to loghost *(cont)*


EventReporter Client
File Service Help
General System Security Application Win2K Only Win2K Only Win2K Only License
Advanced
OK Cancel Apply

☑ Report Truncated Log
☑ Enable Logging
Facility: LOCAL_0
Last Record: 303

Event types to log
☑ Success ☑ Information
☑ Error ☑ Audit Success
☑ Warning ☑ Audit Failure

Filter Rules
☐ Enable Filter Rules

SMTP Properties
☐ Enable Mail delivery
Mailserver
sender
recipient
Subject
Output Encoding: System Default

Connected to: LOCALHOST

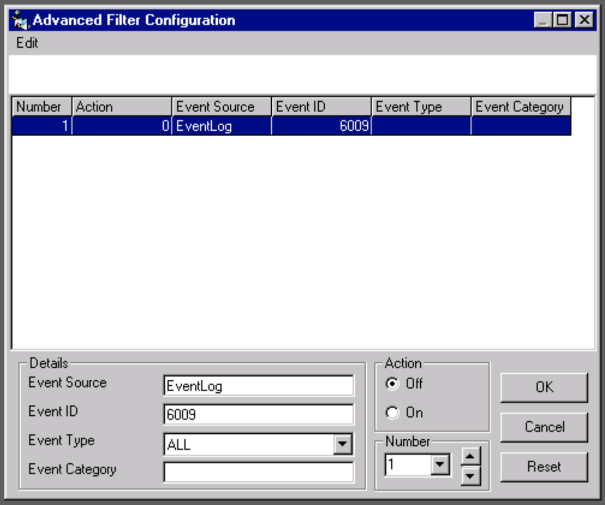
55

I've configured EventReporter to send my System Log events to the remote loghost as facility LOCAL_0 – we'll see that show up in the *syslog* data.

The *Report Truncated Log* option enables EventReporter to send a message when WinNT automatically truncates its Event Log – alerting the administrator that there's a problem.  Leaving this enabled is a really good idea.

The *Filter Rules* enable the administrator to decide specific events to send (or not send) to *syslog.*  This enables you to eliminate messages you know you don't care about from your loghost, or conversely to be extremely granular about which messages you forward.  It's generally safer to eliminate messages you know are unimportant than to specify messages to send.

## Windows to loghost *(cont)*



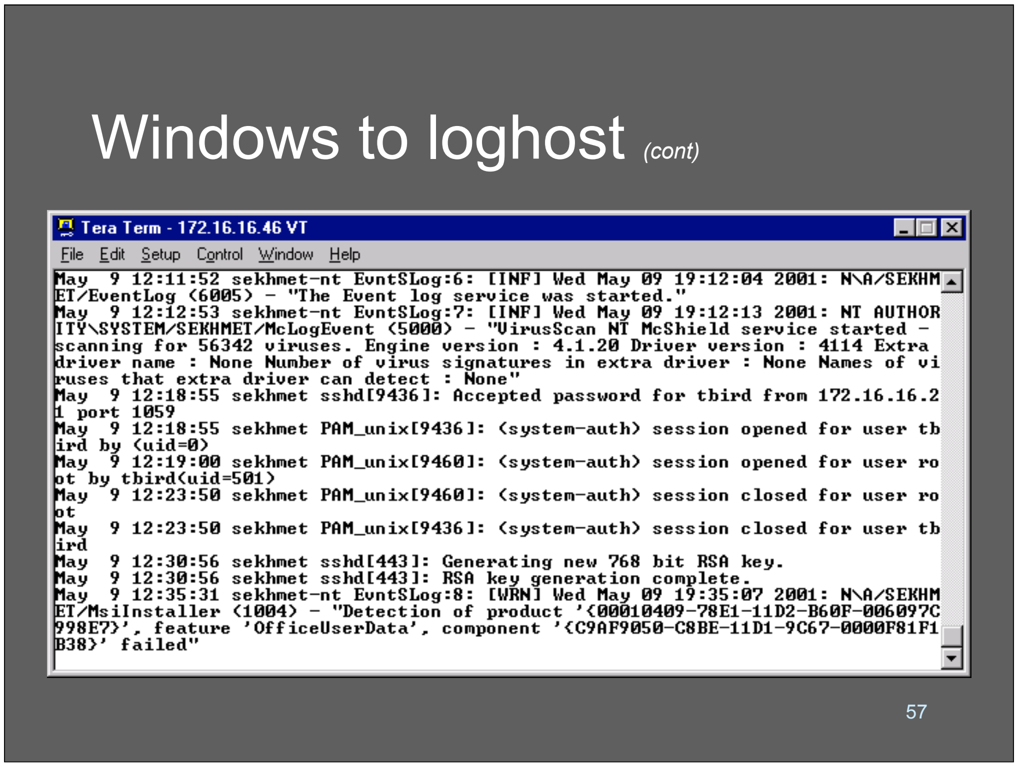
| Number | Action | Event Source | Event ID | Event Type | Event Category |
|---|---|---|---|---|---|
| 1 | 0 | EventLog | 6009 | | |

Details
Event Source: EventLog
Event ID: 6009
Event Type: ALL
Event Category:

Action: [✓ Off] On
Number: 1

The *Filter Rules* enable the administrator to decide specific events to send (or not send) to *syslog*. This enables you to eliminate messages you know you don't care about from your loghost, or conversely to be extremely granular about which messages you forward. It's generally safer to eliminate messages you know are unimportant than to specify messages to send.

Event ID 6009 states "Microsoft (R) Windows NT (R) 4.0 1381 Service Pack 6 Uniprocessor Free." I'm not particularly interested in this as a security event, so I've set up this run to filter it out of my *syslog* stream. This will save my loghost processing time when it comes to processing data in the search for critical security issues.



## Windows to loghost *(cont)*

Tera Term - 172.16.16.46 VT
File Edit Setup Control Window Help

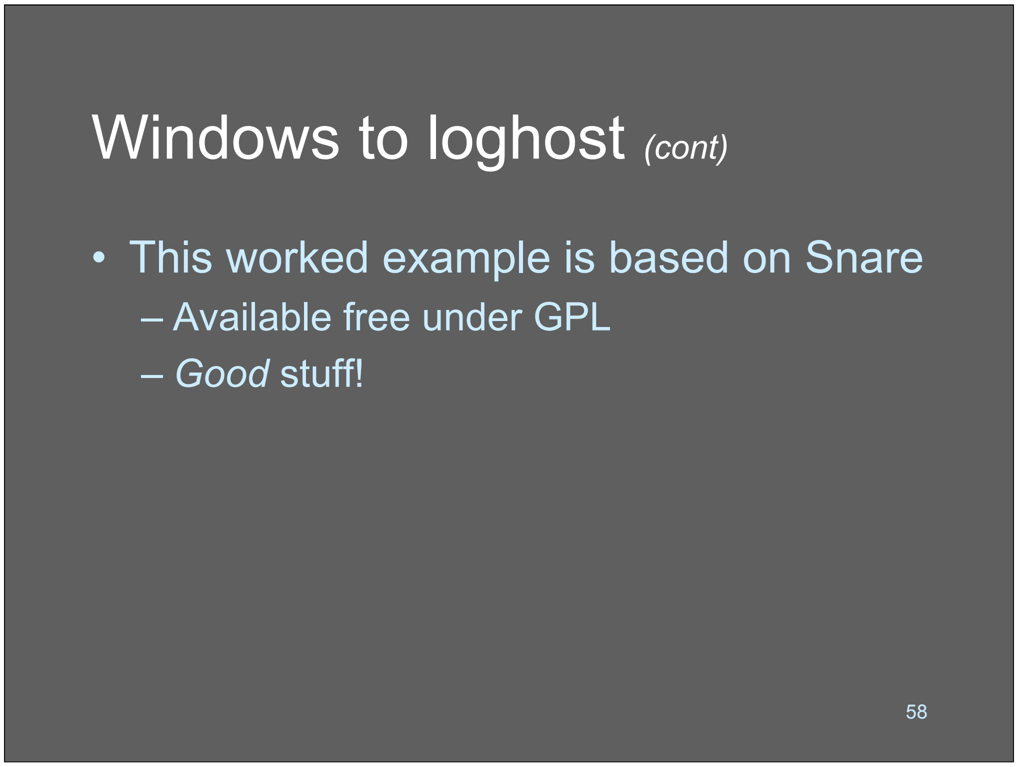
```
May  9 12:11:52 sekhmet-nt EvntSLog:6: [INF] Wed May 09 19:12:04 2001: N\A/SEKHM
ET/EventLog (6005) - "The Event log service was started."
May  9 12:12:53 sekhmet-nt EvntSLog:7: [INF] Wed May 09 19:12:13 2001: NT AUTHOR
ITY\SYSTEM/SEKHMET/McLogEvent (5000) - "VirusScan NT McShield service started -
scanning for 56342 viruses. Engine version : 4.1.20 Driver version : 4114 Extra
driver name : None Number of virus signatures in extra driver : None Names of vi
ruses that extra driver can detect : None"
May  9 12:18:55 sekhmet sshd[9436]: Accepted password for tbird from 172.16.16.2
1 port 1059
May  9 12:18:55 sekhmet PAM_unix[9436]: (system-auth) session opened for user tb
ird by (uid=0)
May  9 12:19:00 sekhmet PAM_unix[9460]: (system-auth) session opened for user ro
ot by tbird(uid=501)
May  9 12:23:50 sekhmet PAM_unix[9460]: (system-auth) session closed for user ro
ot
May  9 12:23:50 sekhmet PAM_unix[9436]: (system-auth) session closed for user tb
ird
May  9 12:30:56 sekhmet sshd[443]: Generating new 768 bit RSA key.
May  9 12:30:56 sekhmet sshd[443]: RSA key generation complete.
May  9 12:35:31 sekhmet-nt EvntSLog:8: [WRN] Wed May 09 19:35:07 2001: N\A/SEKHM
ET/MsiInstaller (1004) - "Detection of product '{00010409-78E1-11D2-B60F-006097C
998E7}', feature 'OfficeUserData', component '{C9AF9050-C8BE-11D1-9C67-0000F81F1
B38}' failed"
```

57

And lo and behold, you get Event Log data cluttering up, whoops, I mean, mingled in with your UNIX *syslog* output!!

EventReporter maps Windows log severities to *syslog* priorities as follows:

| **NT Severity** | **Code** | ***syslog* Priority** |
|---|---|---|
| Audit Success | [AUS] | LOG_NOTICE |
| Audit Failure | [AUF] | LOG_WARNING |
| Information | [INF] | LOG_NOTICE |
| Warning | [WRN] | LOG_WARNING |
| Error | [ERR] | LOG_ERR |
| none | [non] | LOG_NOTICE |



## Windows to loghost *(cont)*

- This worked example is based on Snare
  - Available free under GPL
  - *Good* stuff!

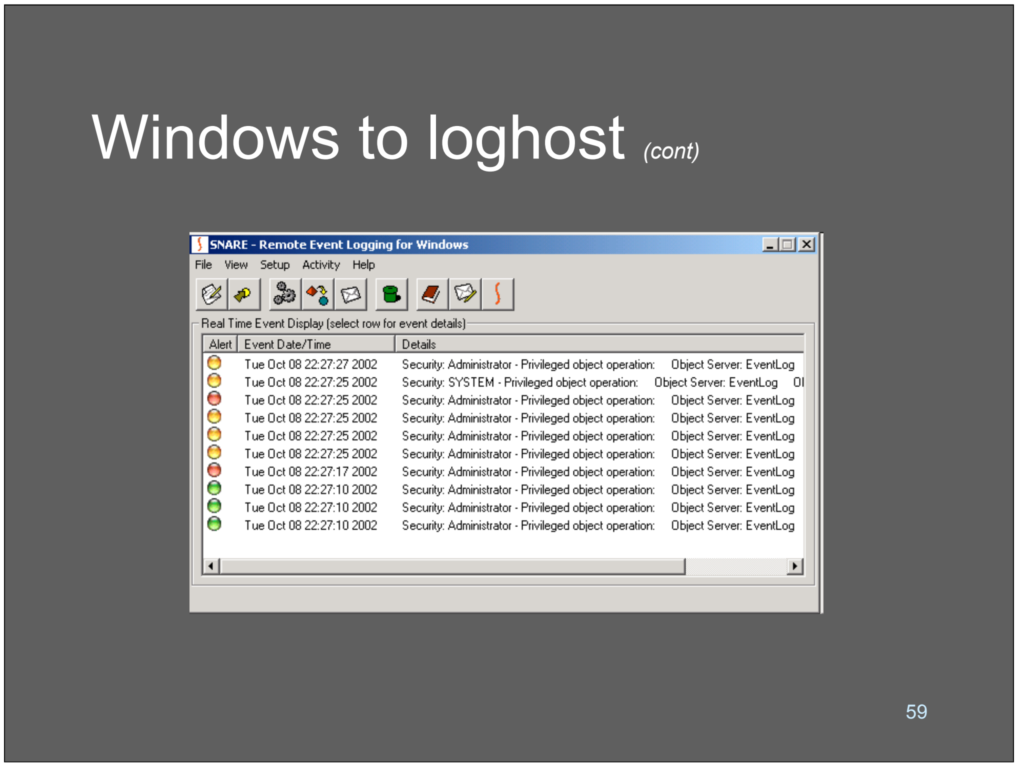
58

The Snare agent for windows syslog is very powerful and is priced affordably.

http://www.intersectalliance.com/projects/SnareWindows/

From the Snare page:
"Snare for Windows is a Windows NT, Windows 2000, Windows XP, and Windows 2003 compatible service that interacts with the underlying Windows Eventlog subsystem to facilitate remote, real-time transfer of event log information.

Event logs from the Security, Application and System logs, as well as the new DNS, File Replication Service, and Active Directory logs are supported. Log data is converted to text format, and delivered to a remote Snare Server, or to a remote Syslog server with configurable and dynamic facility and priority settings."



## Windows to loghost *(cont)*

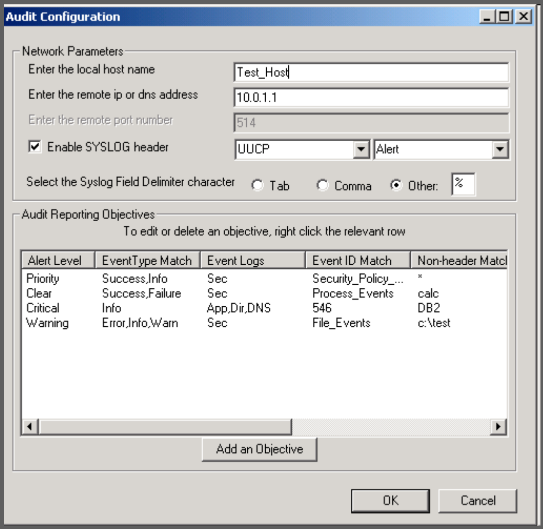
SNARE - Remote Event Logging for Windows
File View Setup Activity Help
Real Time Event Display (select row for event details)

| Alert | Event Date/Time | Details |
|---|---|---|
| 🟡 | Tue Oct 08 22:27:27 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |
| 🟡 | Tue Oct 08 22:27:25 2002 | Security: SYSTEM - Privileged object operation: Object Server: EventLog Ol |
| 🔴 | Tue Oct 08 22:27:25 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |
| 🟡 | Tue Oct 08 22:27:25 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |
| 🟡 | Tue Oct 08 22:27:25 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |
| 🟡 | Tue Oct 08 22:27:25 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |
| 🔴 | Tue Oct 08 22:27:17 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |
| 🟢 | Tue Oct 08 22:27:10 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |
| 🟢 | Tue Oct 08 22:27:10 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |
| 🟢 | Tue Oct 08 22:27:10 2002 | Security: Administrator - Privileged object operation: Object Server: EventLog |

59

This is the snare real-time event viewer; what is seen on the client system. This shows the expanded system log information that is being analyzed to pass to the remote server.

## Windows to loghost *(cont)*



Network Parameters

| | |
|---|---|
| Enter the local host name | Test_Host |
| Enter the remote ip or dns address | 10.0.1.1 |
| Enter the remote port number | 514 |
| [✓] Enable SYSLOG header | UUCP / Alert |

Select the Syslog Field Delimiter character ( ) Tab ( ) Comma [✓ Other:] %

Audit Reporting Objectives

To edit or delete an objective, right click the relevant row

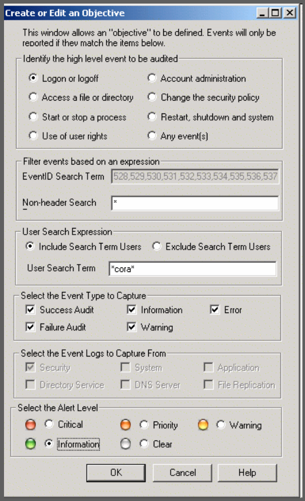
| Alert Level | EventType Match | Event Logs | Event ID Match | Non-header Matc |
|---|---|---|---|---|
| Priority | Success,Info | Sec | Security_Policy_... | * |
| Clear | Success,Failure | Sec | Process_Events | calc |
| Critical | Info | App,Dir,DNS | 546 | DB2 |
| Warning | Error,Info,Warn | Sec | File_Events | c:\test |

60

Here we show configuring Snare. The remote server's address has been specified, and the syslog header fields are set. You'll notice that you can define Snare to transmit with a default  priority and facility. You can overrule some of the defaults on the per-objective rules we'll see in the next slide.

The bottom panel consists of "objectives" which are the matching rules used to decide what events should be syslogged. This is nice because it lets you "tune" the client so that you're not going to overload the server with extra messages you don't need.

## Windows to loghost *(cont)*



Create or Edit an Objective

This window allows an "objective" to be defined. Events will only be reported if they match the items below.

Identify the high level event to be audited
- [✓ Logon or logoff]
- Account administration
- Access a file or directory
- Change the security policy
- Start or stop a process
- Restart, shutdown and system
- Use of user rights
- Any event(s)

Filter events based on an expression
- EventID Search Term: 528,529,530,531,532,533,534,535,536,537
- Non-header Search: *

User Search Expression
- [✓ Include Search Term Users] Exclude Search Term Users
- User Search Term: *cora*

Select the Event Type to Capture
- ✓ Success Audit
- ✓ Information
- ✓ Error
- ✓ Failure Audit
- ✓ Warning

Select the Event Logs to Capture From
- ✓ Security
- System
- Application
- Directory Service
- DNS Server
- File Replication

Select the Alert Level
- Critical
- Priority
- Warning
- [✓ Information]
- Clear

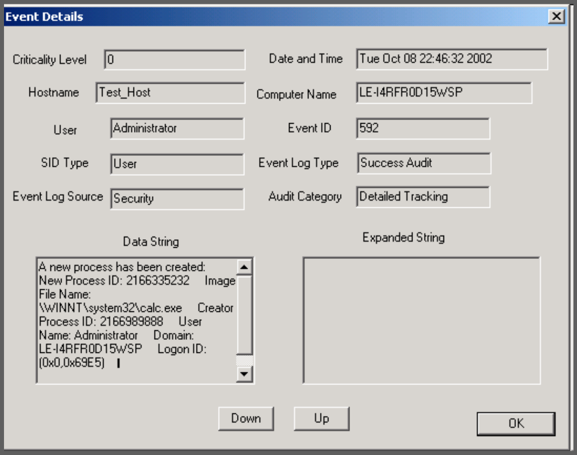
61

This is the editor screen that lets you create a new objective/matching rule.

The top of the screen lets you specify static types of events (like logon/logoff) followed by search terms. Search terms can include wildcards, which is incredibly useful.

The bottom of the screen lets you assign a syslog priority that overrides the default setting.

## Windows to loghost *(cont)*



| | | | |
|---|---|---|---|
| Criticality Level | 0 | Date and Time | Tue Oct 08 22:46:32 2002 |
| Hostname | Test_Host | Computer Name | LE-I4RFR0D15WSP |
| User | Administrator | Event ID | 592 |
| SID Type | User | Event Log Type | Success Audit |
| Event Log Source | Security | Audit Category | Detailed Tracking |

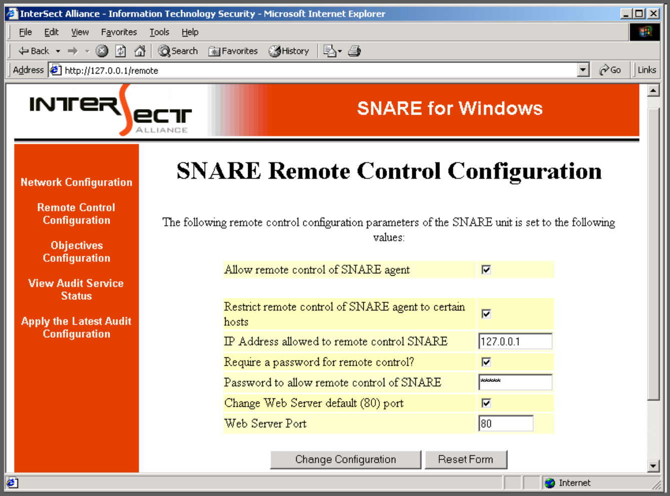
Data String

```
A new process has been created:
New Process ID: 2166335232    Image
File Name:
\WINNT\system32\calc.exe    Creator
Process ID: 2166989888    User
Name: Administrator    Domain:
LE-I4RFR0D15WSP    Logon ID:
(0x0,0x69E5)
```

Expanded String

This shows an expanded event log record in the event viewer.

## Windows to loghost *(cont)*



InterSect Alliance - Information Technology Security - Microsoft Internet Explorer

Address http://127.0.0.1/remote

SNARE for Windows

Network Configuration
Remote Control Configuration
Objectives Configuration
View Audit Service Status
Apply the Latest Audit Configuration

SNARE Remote Control Configuration

The following remote control configuration parameters of the SNARE unit is set to the following values:

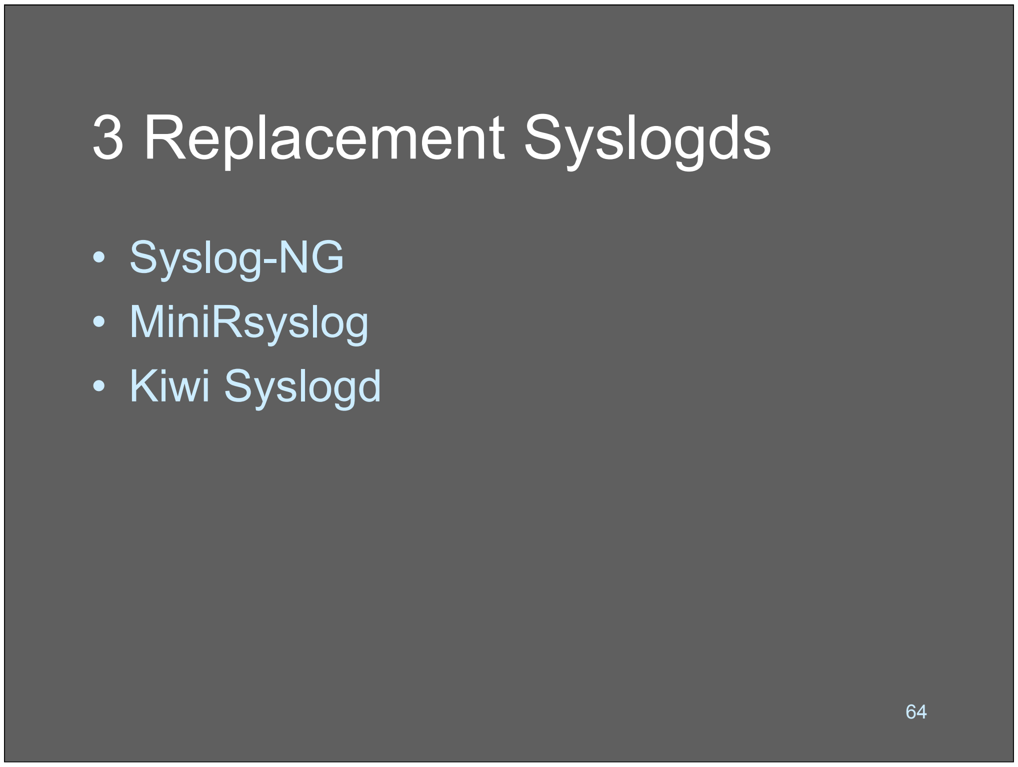
| | |
|---|---|
| Allow remote control of SNARE agent | ✓ |
| Restrict remote control of SNARE agent to certain hosts | ✓ |
| IP Address allowed to remote control SNARE | 127.0.0.1 |
| Require a password for remote control? | ✓ |
| Password to allow remote control of SNARE | ****** |
| Change Web Server default (80) port | ✓ |
| Web Server Port | 80 |

63

One of the interesting things about Snare is that you can configure the client to allow remote management by a specified IP address using a password and a browser.

There are lots of cool potential tricks you can play with this - set up all your systems with a baseline configuration that allows them to be remote-managed by your log server, and then you could conceivably script a lot of your management using perl or http-get, etc.

Extra credit: check and see if Snare logs attempts to talk to the access port using an invalid password. Suppose it does - how might you use this to turn Snare into an agent to detect worms or SMTP probes?



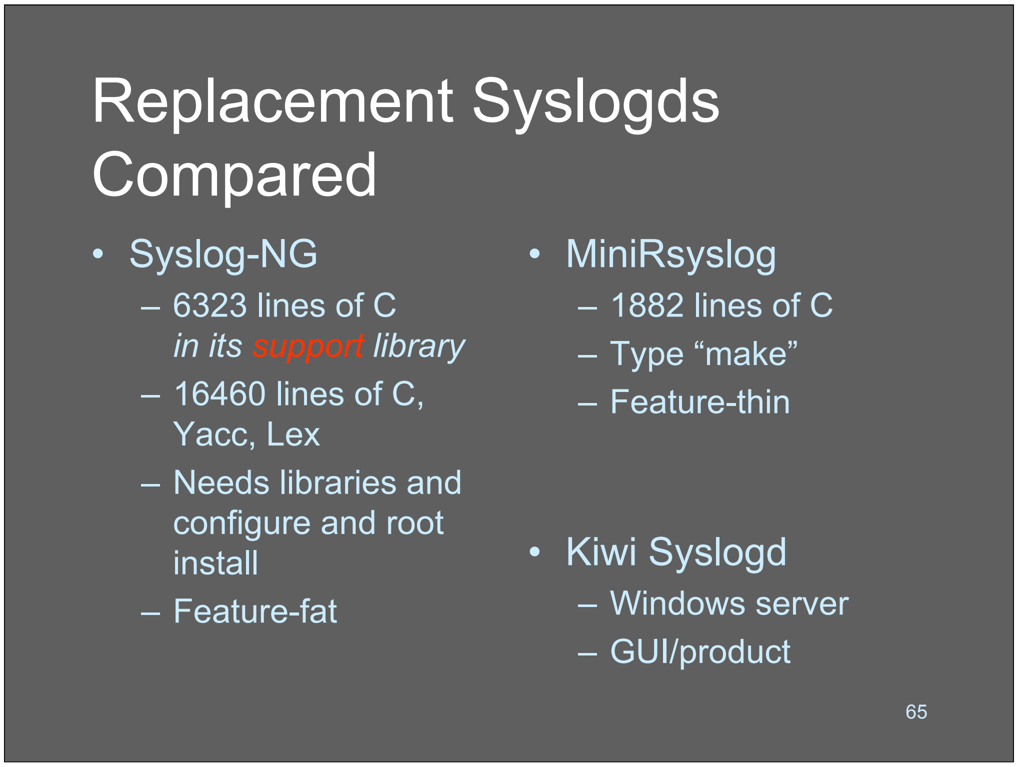
## 3 Replacement Syslogds

- Syslog-NG
- MiniRsyslog
- Kiwi Syslogd

64

If all you want is basic syslog services, the standard syslog program that comes with most UNIXes will probably do an adequate job. If, however, you want to do fancier processing, processing of large loads, or are building an internet-facing log server, you should plan to replace the stock syslogd.

There are a ***large*** number of syslog replacements out there! I can't even come close to covering them all, or even covering 3 of them in detail. So what we're going to do today is look at 3 of the most popular syslogd replacements.



## Replacement Syslogds Compared

- Syslog-NG
  - 6323 lines of C *in its support library*
  - 16460 lines of C, Yacc, Lex
  - Needs libraries and configure and root install
  - Feature-fat
- MiniRsyslog
  - 1882 lines of C
  - Type “make”
  - Feature-thin
- Kiwi Syslogd
  - Windows server
  - GUI/product

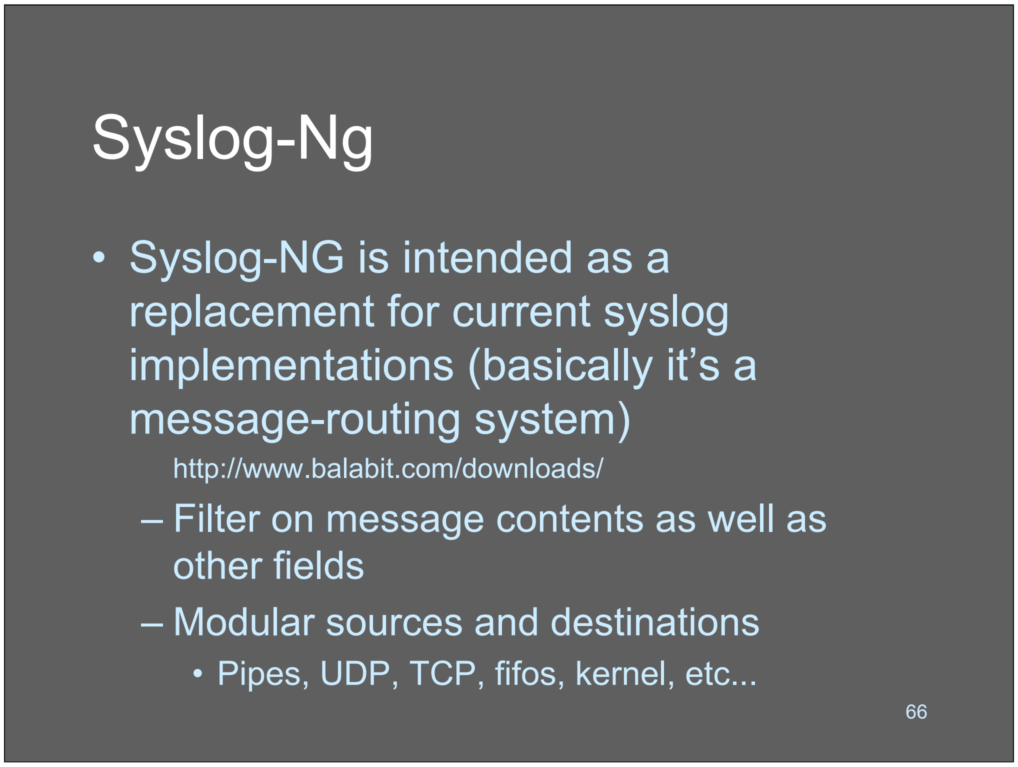
65

It’s instructive to compare our two candidate syslogd replacements. Syslog-ng is “feature-fat” and has a very impressive set of capabilities. Minirsyslog is “feature-thin” and only does one thing well.

When you’re deciding on a tool to use, you need to take into account a lot of things - most importantly, how you plan to use it. I like to think of these two tools as the representatives of a design philosophy debate that has been going on for a very long time. Some people would rather have a complex multi-tool like a LeatherMan™ that is simultaneously pliers, screwdrivers, knife, etc - others prefer to carry a knife, a screwdriver, and a pair of pliers. These is a design philosophy that argues that it is *impossible* to make a combined hammer/screwdriver that would be as good as the separate standalone tools - because the separate tools are optimized to accomplish a single purpose better.

A fundamental of security design is that minimalism is good. Indeed, if you assume (I don’t believe we can prove this) that bugs-per-lines-of-code is a useable metric, then the larger a piece of code grows, the buggier it will be. This has serious implications when talking about security software.

Choose your tools carefully.



## Syslog-Ng

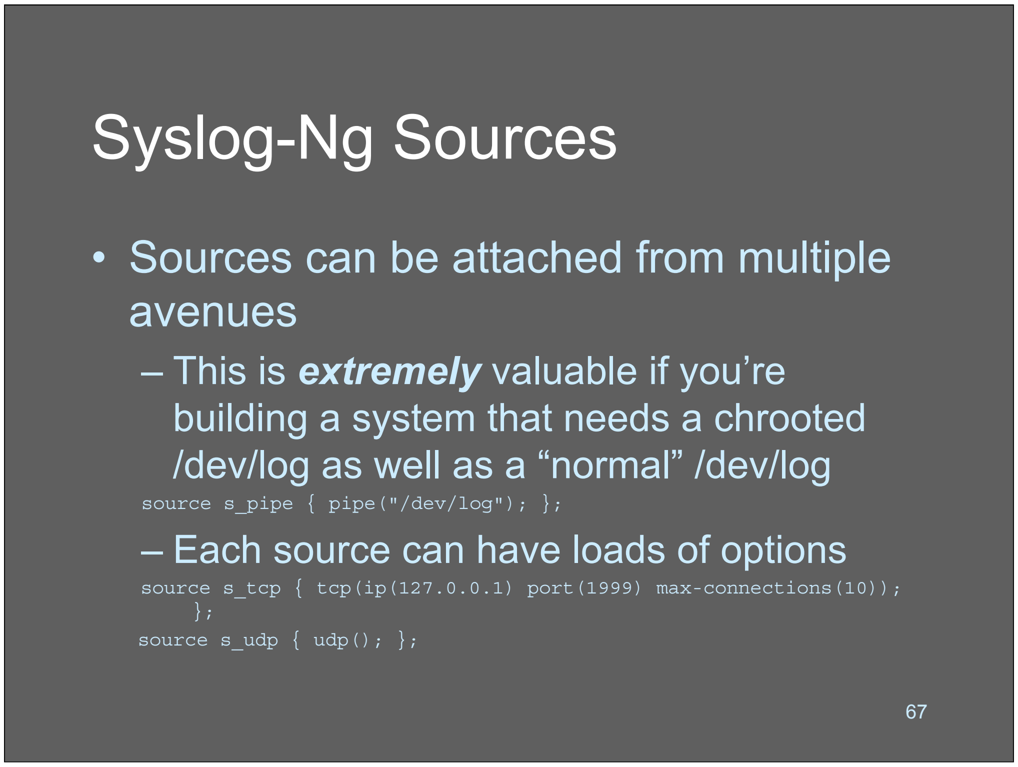
- Syslog-NG is intended as a replacement for current syslog implementations (basically it's a message-routing system)
  http://www.balabit.com/downloads/
  - Filter on message contents as well as other fields
  - Modular sources and destinations
    - Pipes, UDP, TCP, fifos, kernel, etc...

66

Syslog-ng has basically all the features you'd want in a system log daemon. It allows you to set up as many listening service ports as you might want, and supports large numbers (and types) of output channels. It has a powerful regular-expression-based filtering language that lets you sort by message contents, or program, or type.

In the next few slides we'll do a quick walk-through of some features of syslog-ng. These are the features that you'll typically need to get the job done for most purposes.

The design of syslog-ng is based on a data flow in which traffic comes into the system through a source, is matched by a filter, and directed to an output channel. Building a syslog-ng configuration entails defining sources, outputs, and filters, then binding them all together using the log( ) operation.



## Syslog-Ng Sources

- Sources can be attached from multiple avenues
  - This is ***extremely*** valuable if you're building a system that needs a chrooted /dev/log as well as a "normal" /dev/log

```
source s_pipe { pipe("/dev/log"); };
```

  - Each source can have loads of options

```
source s_tcp { tcp(ip(127.0.0.1) port(1999) max-connections(10));
    };
source s_udp { udp(); };
```

67

Syslog-ng servers can collect data from multiple sources simultaneously.

Honestly, this is not that big a deal unless you *need* to handle multiple sources, in which case it's incredibly useful. For example, with "traditional" syslogd, many admins were unable to correctly collect logs in chrooted areas - they needed multiple /dev/log entries (e.g.: /var/www/chroot/dev/log and /dev/log)

Syslog-ng sources are:

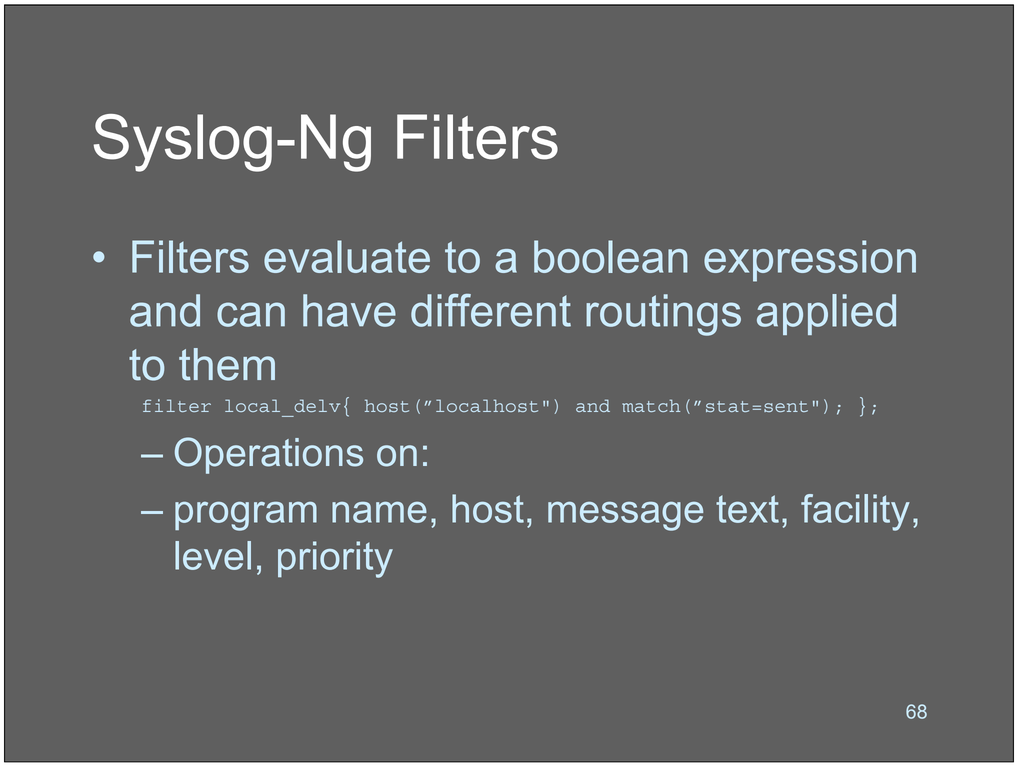
•TCP connection

•UDP datagram ("traditional syslog")

•UDP UNIX dom,ain socket

•kernel log

Each different log source has a multitude of options, such as port, max connections, permitted addresses, etc. In the example above, a TCP listener is created on 127.0.0.1, port 1999 - we might want to create multiple TCP listeners on a variety of ports, or addresses for whatever reason.



## Syslog-Ng Filters

- Filters evaluate to a boolean expression and can have different routings applied to them

```
filter local_delv{ host(”localhost") and match(”stat=sent"); };
```

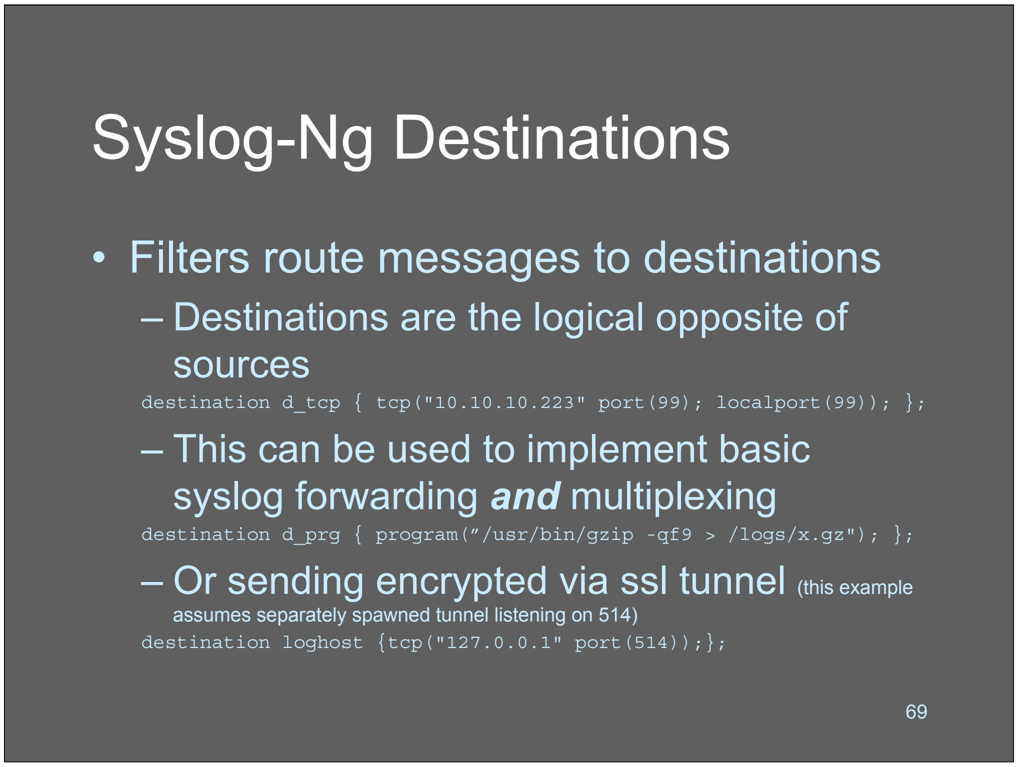
  - Operations on:
  - program name, host, message text, facility, level, priority

68

A filter is a named entity that can combine multiple clauses in an expression to evaluate as a truth-value. All of the fields of a syslog message can be matched in a filter statement.

In this example, the filter will match all messages containing the string “stat=sent” that come from localhost.

There are a couple of different paradigms syslog analysts use for matching and sorting messages with syslog-ng. Either divide the messages up by program, or source, or priority. Each approach has its advantages and disadvantages. Some syslog analysts sort the messages in duplicate (I.e.: sort by host into one set of files and sort by priority into another)



## Syslog-Ng Destinations

- Filters route messages to destinations
  - Destinations are the logical opposite of sources

```
destination d_tcp { tcp("10.10.10.223" port(99); localport(99)); };
```

  - This can be used to implement basic syslog forwarding ***and*** multiplexing

```
destination d_prg { program("/usr/bin/gzip -qf9 > /logs/x.gz"); };
```

  - Or sending encrypted via ssl tunnel (this example assumes separately spawned tunnel listening on 514)

```
destination loghost {tcp("127.0.0.1" port(514));};
```

69

A destination is a syslog-ng output channel; a syslog-ng server can manage a large number of destinations. The most popular destinations used are:

•File

•Udp

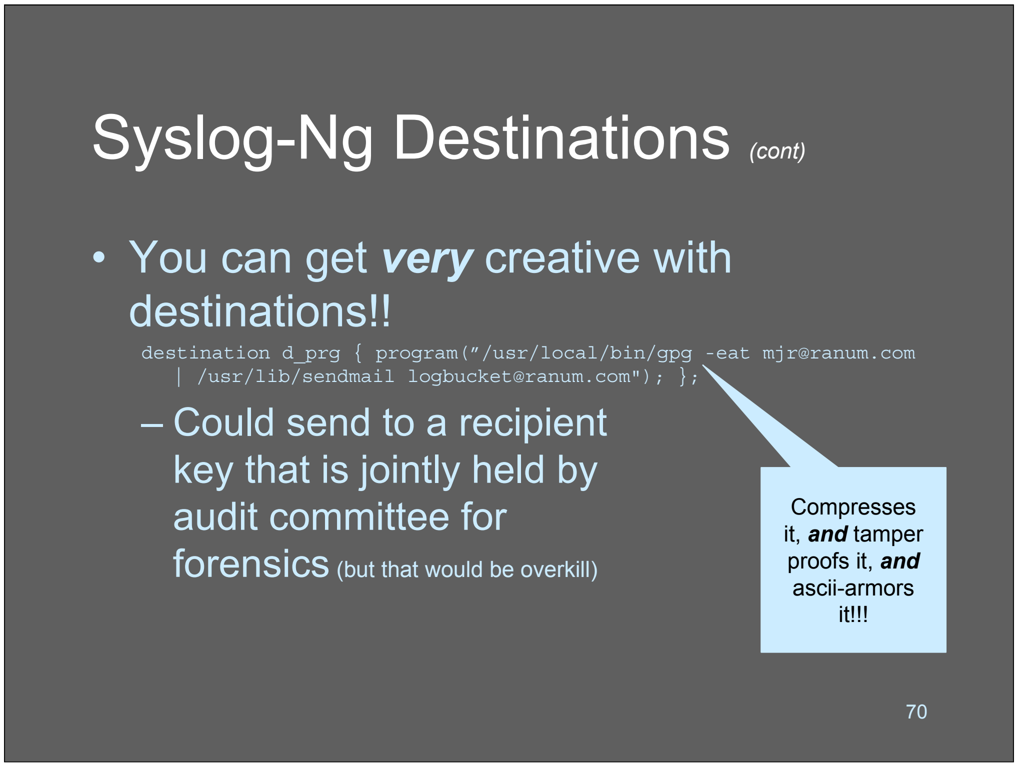
•Tcp

•Program

The file destination file("/var/log/whatever") is the typical means of appending a message to a log file. Each of the destination types has its own configuration options that are specific to how it functions.

The program("whatever") option calls a subshell to receive the messages on its standard input. In the first example above, our logs are being automatically compressed using gzip. One thing to be aware of, if you do this, is that the gzip process may buffer or delay data in transit; if the system crashes while gzip is holding messages in its memory that have not been written to disk, they will be lost. In general, this is not a big deal but it's worth being aware of.

The final example on this slide is a typical example of how many sites tunnel syslog data over an application-level relay like ssh or ssltunnel. The tunnel is brought up using other configuration practices (starting ssltunnel in rc.local, to listen on a local port and connect to a remote system) then syslog-ng is configured to output via TCP to the tunnel. An alternate form of this is to just set up a destination such as sshing a command like "dd -a of=/remote/file"



## Syslog-Ng Destinations *(cont)*

- You can get ***very*** creative with destinations!!

```
destination d_prg { program("/usr/local/bin/gpg -eat mjr@ranum.com
   | /usr/lib/sendmail logbucket@ranum.com"); };
```

  - Could send to a recipient key that is jointly held by audit committee for forensics (but that would be overkill)

Compresses it, ***and*** tamper proofs it, ***and*** ascii-armors it!!!

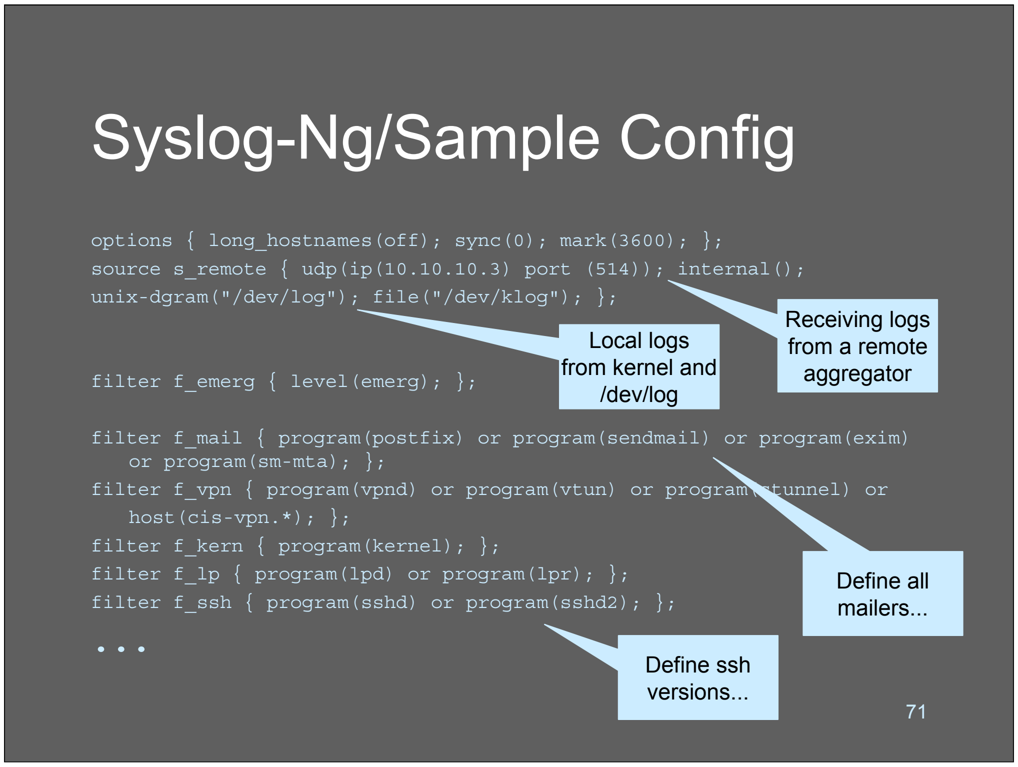
70

The possibilities with program destinations are tremendous.

In this example,  we are outputting all messages to PGP's email encryption, which is then being sent to a remote address via Email. This is a *great* way to get messages to a remote location securely and reliably. After all, the Email system will take care of queuing up and retrying transmission, etc. The PGP message format is portable and includes compression as a "freebie" so not only are these messages tamper-proof, and secure - they're small.

The only problem with this approach is that syslog-ng will only "flush" the data on the program() channel when it exits or gets a SIGHUP. If you use this approach you should have a cron job SIGHUP the daemon at your preferred time interval.

This is a great simple way to keep certain syslogs on a remote location where they are tamper-proof.



## Syslog-Ng/Sample Config

```
options { long_hostnames(off); sync(0); mark(3600); };
source s_remote { udp(ip(10.10.10.3) port (514)); internal();
unix-dgram("/dev/log"); file("/dev/klog"); };


filter f_emerg { level(emerg); };

filter f_mail { program(postfix) or program(sendmail) or program(exim)
   or program(sm-mta); };
filter f_vpn { program(vpnd) or program(vtun) or program(stunnel) or
   host(cis-vpn.*); };
filter f_kern { program(kernel); };
filter f_lp { program(lpd) or program(lpr); };
filter f_ssh { program(sshd) or program(sshd2); };
. . .
```

Receiving logs from a remote aggregator
Local logs from kernel and /dev/log
Define all mailers...
Define ssh versions...
71

Let’s walk through an example syslog-n configuration file, to give you an idea of some of the capabilities of this software.

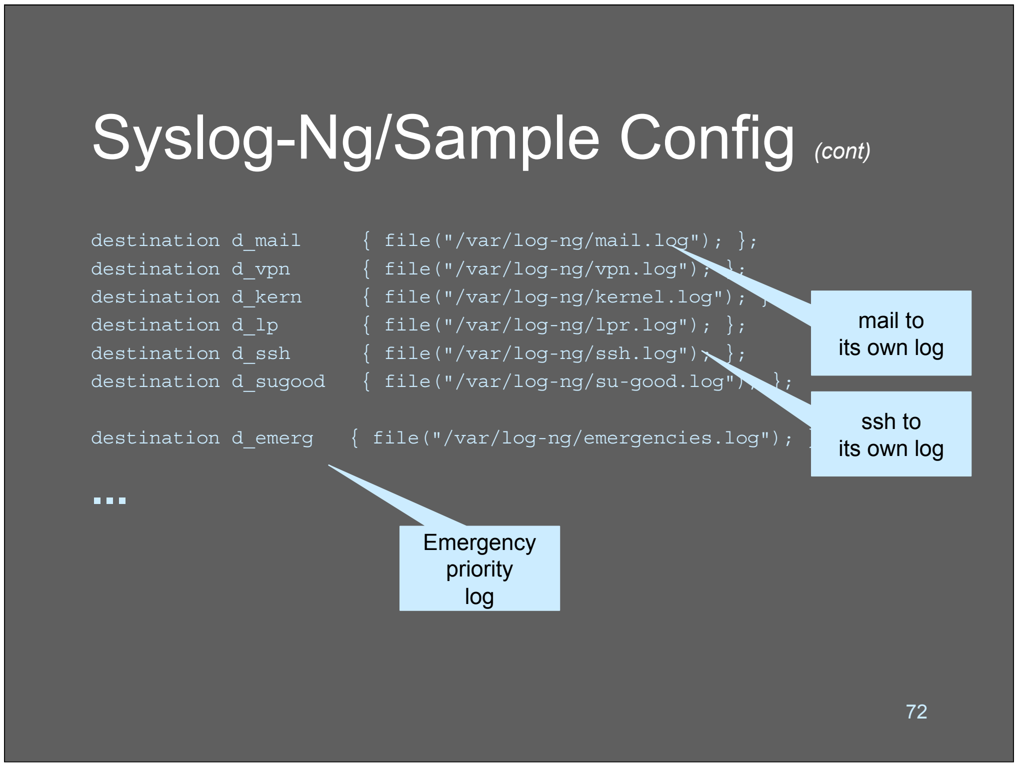
The first line sets up a few global options: we turn long hostnames off (use short forms instead of fully-qualified domain names) sync() is set to zero, which means that logfiles should be forced to disk whenever a new message is added. Mark() is set to 3600 seconds, which means that the log server will output a status heartbeat every hour. The internal() clause is the “magic” syslog-ng-originated source; you can attach it to one of the input sources and have a feed of syslog-ng specific data. In this case we are treating internal() just like everything else.

We then set up a remote source from which we will accept UDP syslog listening on local address 10.10.10.3 on port 514. This is useful for dual-homed machines, in which you may want to have a syslog listener on one interface but not another.

The next set of lines, we initialize listening on the kernel’s logfile, and the UNIX domain datagram socket /dev/log.

The f_hosts filter matches based on “emergency” level syslog priority. So in this example we are doing some matching by priority and some matching by program.

The remainder of lines break each service down into categories based on the program name. So we’ve established a filter for “mail” programs that includes a number of the usual suspects. Here we are setting ourselves up so that we can divide our logs into files by program/service.



## Syslog-Ng/Sample Config *(cont)*

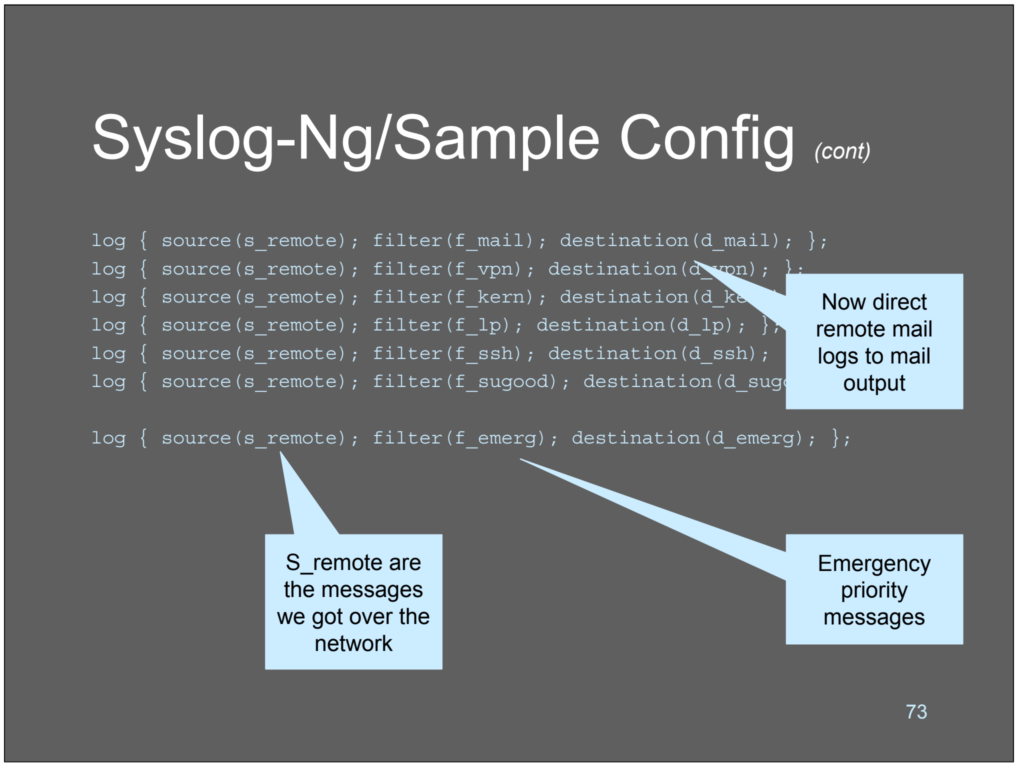
```
destination d_mail     { file("/var/log-ng/mail.log"); };
destination d_vpn      { file("/var/log-ng/vpn.log"); };
destination d_kern     { file("/var/log-ng/kernel.log"); };
destination d_lp       { file("/var/log-ng/lpr.log"); };
destination d_ssh      { file("/var/log-ng/ssh.log"); };
destination d_sugood   { file("/var/log-ng/su-good.log"); };

destination d_emerg   { file("/var/log-ng/emergencies.log"); };

...
```

mail to its own log
ssh to its own log
Emergency priority log

72

The destinations section of the config file is fairly straighforward. For each of the services that we defined a filter for, we create a destination file that is appropriately named.

Our emergency priority level message log is separately defined at the bottom.



## Syslog-Ng/Sample Config *(cont)*

```
log { source(s_remote); filter(f_mail); destination(d_mail); };
log { source(s_remote); filter(f_vpn); destination(d_vpn); };
log { source(s_remote); filter(f_kern); destination(d_ke
log { source(s_remote); filter(f_lp); destination(d_lp); };
log { source(s_remote); filter(f_ssh); destination(d_ssh);
log { source(s_remote); filter(f_sugood); destination(d_sug

log { source(s_remote); filter(f_emerg); destination(d_emerg); };
```

Now direct remote mail logs to mail output

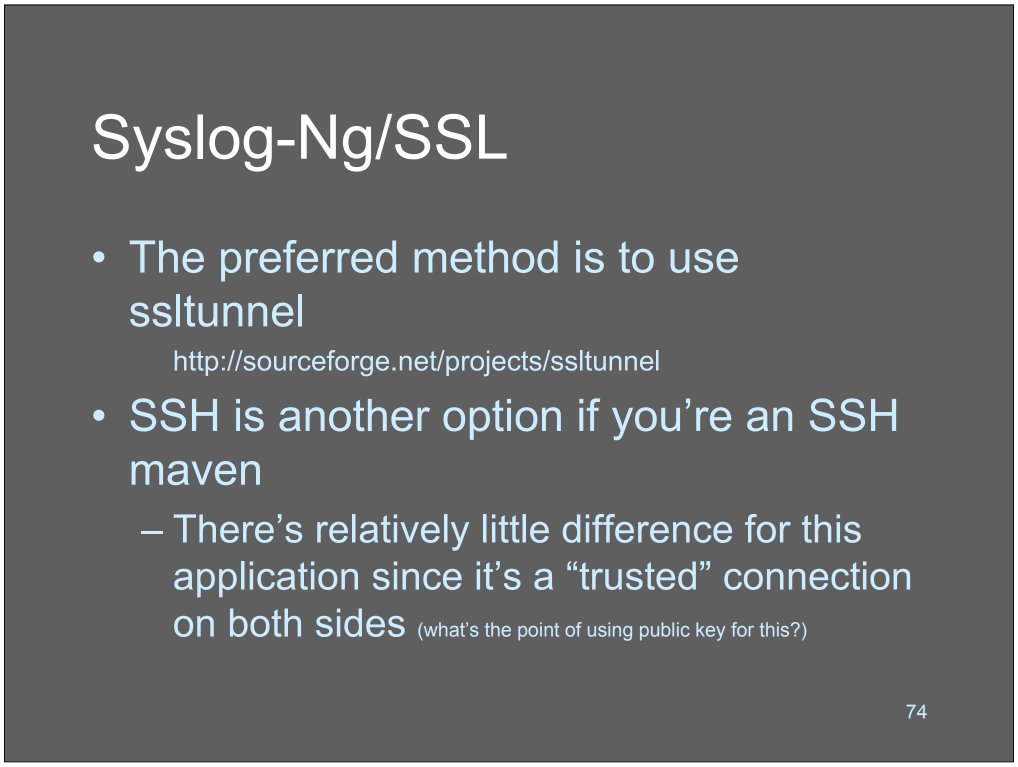
S_remote are the messages we got over the network

Emergency priority messages

73

Here we glue all the pieces together with the log() operation. Messages from specific sources that match specific filters are routed to specific destinations. Here you can apply the same filter multiple times if you want to centralize the definition of a particular matching rule.

At the bottom, you can see we use our priority-based filter to match and send messages to our emergency log destination.



## Syslog-Ng/SSL

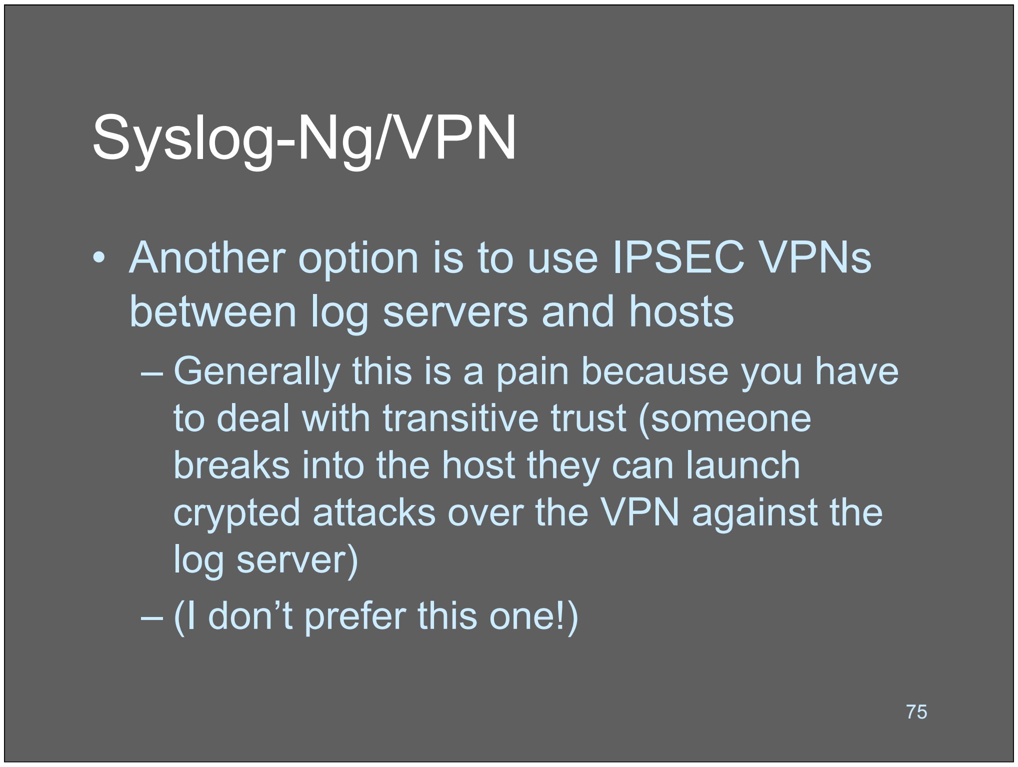
- The preferred method is to use ssltunnel
  http://sourceforge.net/projects/ssltunnel
- SSH is another option if you're an SSH maven
  - There's relatively little difference for this application since it's a "trusted" connection on both sides (what's the point of using public key for this?)

74

There are lots of ways to tunnel syslog-ng traffic over a network using a secure tunnel. The preferred method is probably to use ssltunnel. If you've got SSH set up on your machine, then use SSH instead. There's really no difference between the two approaches; it's a matter of personal preference.

I'm a paranoid, personally. Since SSH and SSL are widely used, there has been a fair amount of effort devoted to hacking the protocols and code for those services. Since public key doesn't really add a lot of security value when you're using preconfigured services between hosts. I use an old tool I wrote a long time ago called "get/put" which does a simple batch-mode encrypted file transfer. It uses fixed DES keys and challenge/response authentication. The bottom line here is that you can use anything you like and as long as it's basically reliable and moderately functional, you'll be fine.

Make sure you protect your log server using firewalling!! Ipchains, iptables, ip_filter - whatever you use; make it as simple and restrictive as possible.



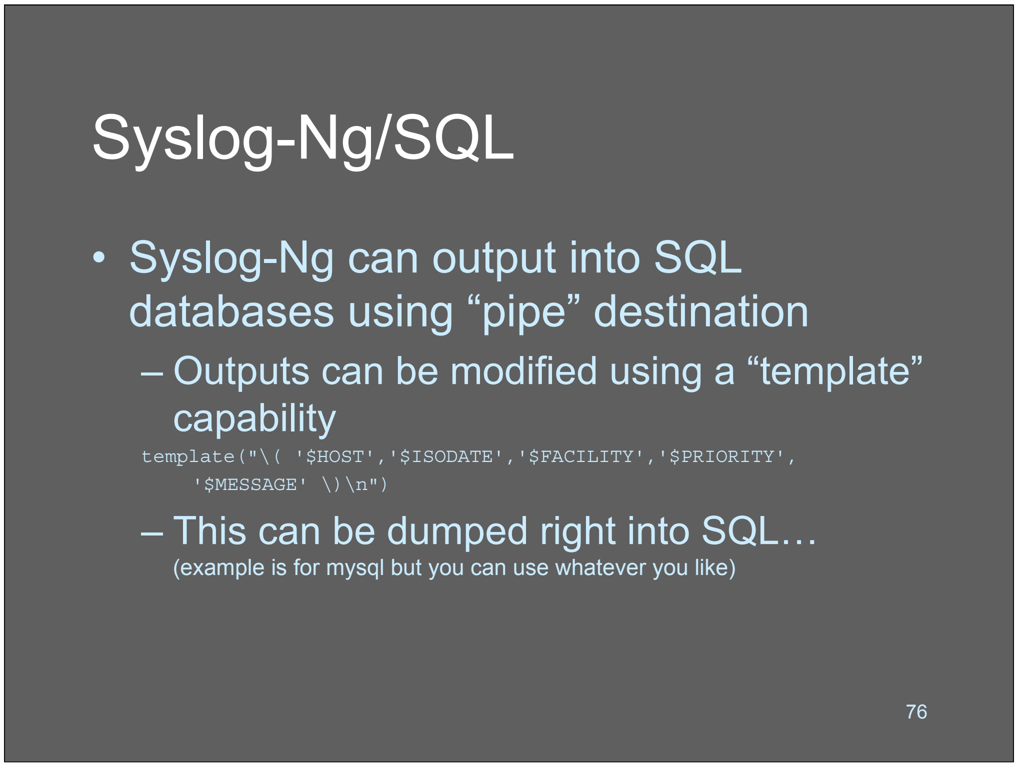
## Syslog-Ng/VPN

- Another option is to use IPSEC VPNs between log servers and hosts
  - Generally this is a pain because you have to deal with transitive trust (someone breaks into the host they can launch crypted attacks over the VPN against the log server)
  - (I don’t prefer this one!)

75

Some log server-builders use IPSEC between their log hosts, so that the security layer is all implemented “below the radar screen” instead of at application level like with a tunnel.

I find this approach to be cumbersome, because of the details involved in setting up IPSEC. Also, I worry that transitive trust attacks may be possible. If someone gets onto one log server then they can attack over the IPSEC VPN!



## Syslog-Ng/SQL

- Syslog-Ng can output into SQL databases using “pipe” destination
  - Outputs can be modified using a “template” capability

```
template("\( '$HOST','$ISODATE','$FACILITY','$PRIORITY',
    '$MESSAGE' \)\n")
```

  - This can be dumped right into SQL…
    (example is for mysql but you can use whatever you like)

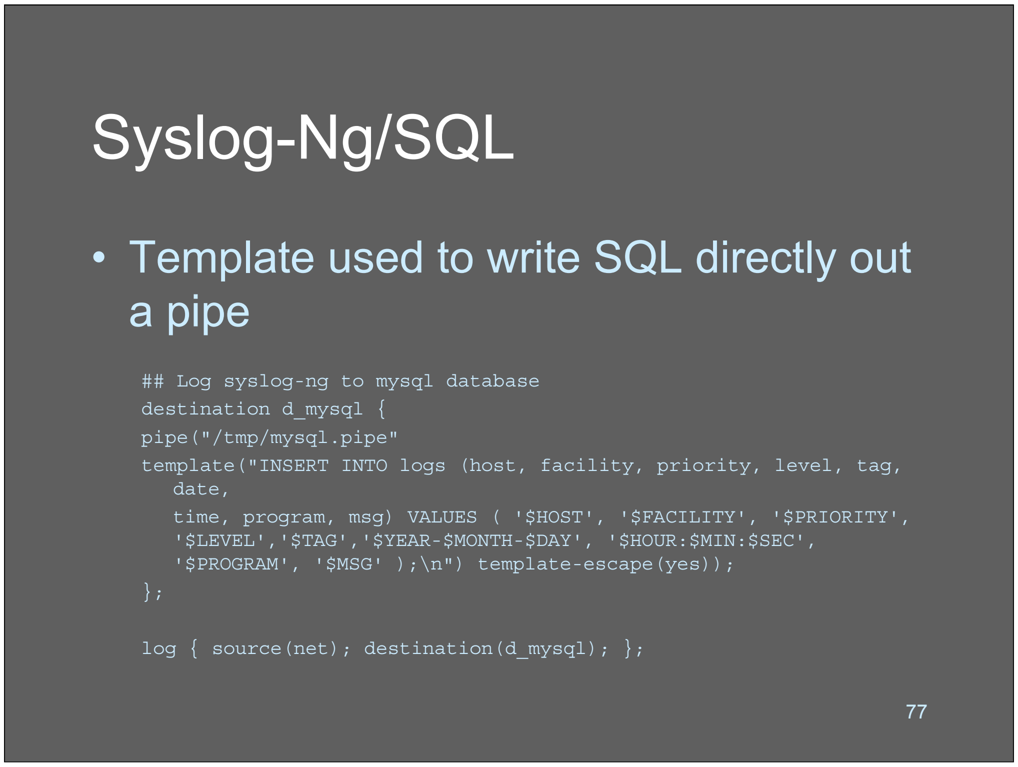
76

Syslog-ng supports re-formatting of output; you can change or omit fields, add line-breaks, etc. This is useful for a wide range of purposes, but is frequently applied to pump data directly from syslog-ng into a SQL database.

This example uses mySQL but you can pretty easily generalize it to any SQL engine you’d like.

Basically, what we do is configure syslog-ng to output SQL insert statements directly out a FIFO, which we then use as input to an interactive SQL session.

Before you approach using a database for your logs, make sure you will actually benefit from doing so. Most of the things you’d want to do with a syslog database will involve linear searching, which eliminates a lot of the value of having the records where they can be queried using SQL. For example, you might wish to search for user-id related information. Unfortunately, since the logs aren’t parsed before they go into the database, that information is not accessible except for via a brute-force search. So if you think you’re going to be doing the SQL equivalent of “grep root database” consider that “grep root file” is often just as fast and is a whole lot easier!



## Syslog-Ng/SQL

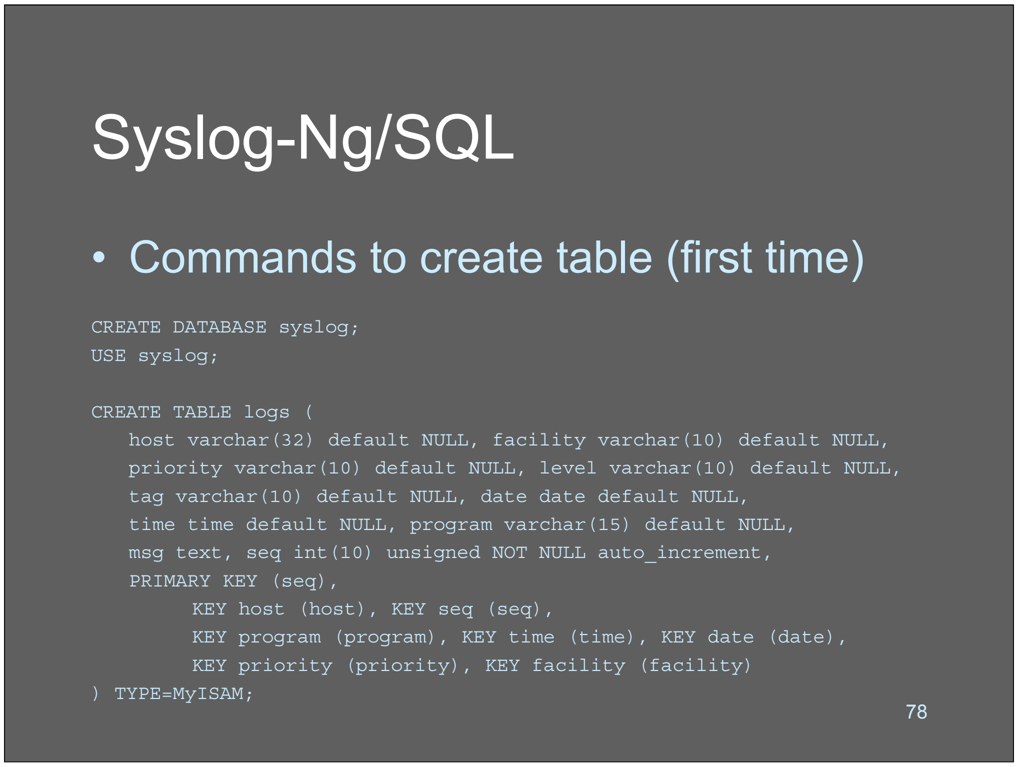
- Template used to write SQL directly out a pipe

```
## Log syslog-ng to mysql database
destination d_mysql {
pipe("/tmp/mysql.pipe"
template("INSERT INTO logs (host, facility, priority, level, tag,
   date,
   time, program, msg) VALUES ( '$HOST', '$FACILITY', '$PRIORITY',
   '$LEVEL','$TAG','$YEAR-$MONTH-$DAY', '$HOUR:$MIN:$SEC',
   '$PROGRAM', '$MSG' );\n") template-escape(yes));
};

log { source(net); destination(d_mysql); };
```

77

Here we define a destination called “mysql” and attach a template to output for that destination. The first part of the template is a stock “insert” SQL command with the name of the fields that we are about to insert into. The next part of the template outputs the values from the syslog record that we have defined in our database.

The pipe(“/tmp/msql.pipe”) is the important part of this trick; we’re cauing syslog-ng to write to a FIFO file which we’re going to use as a vehicle for carrying the SQL insert commands directly into MySQL.



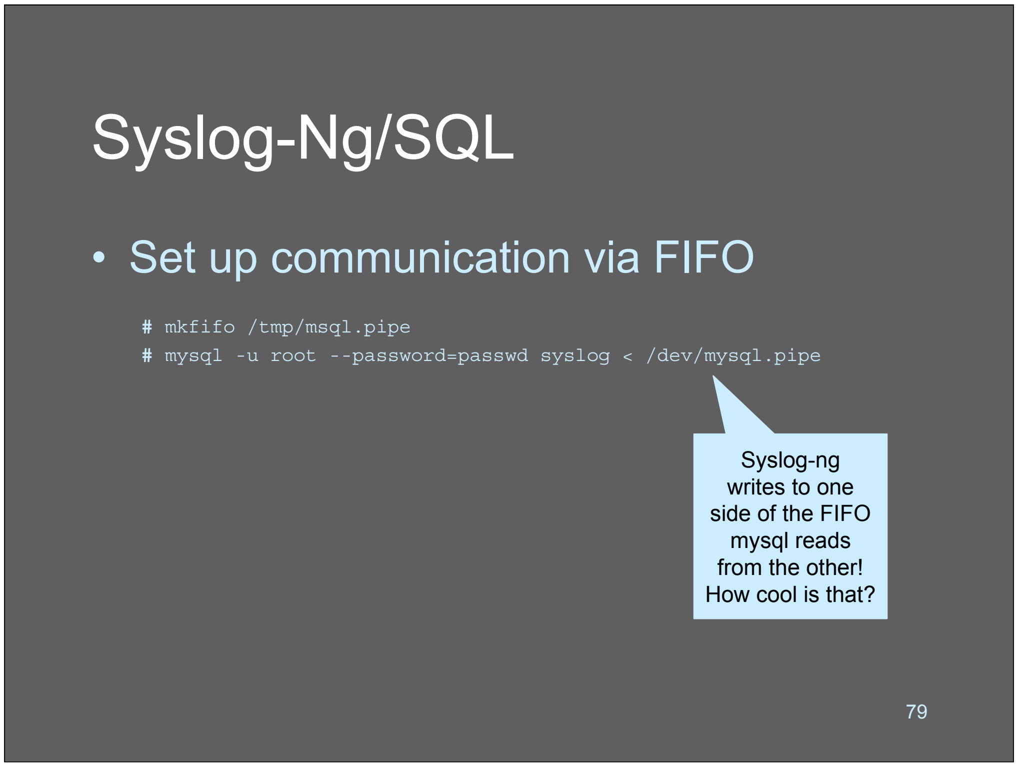
## Syslog-Ng/SQL

- Commands to create table (first time)

```
CREATE DATABASE syslog;
USE syslog;

CREATE TABLE logs (
   host varchar(32) default NULL, facility varchar(10) default NULL,
   priority varchar(10) default NULL, level varchar(10) default NULL,
   tag varchar(10) default NULL, date date default NULL,
   time time default NULL, program varchar(15) default NULL,
   msg text, seq int(10) unsigned NOT NULL auto_increment,
   PRIMARY KEY (seq),
        KEY host (host), KEY seq (seq),
        KEY program (program), KEY time (time), KEY date (date),
        KEY priority (priority), KEY facility (facility)
) TYPE=MyISAM;
```

78

These are the MySQL commands used to create the table that we're populating in this example. We define the fields in the schema and give their types. We also define which fields are primary index keys, so the database will maintain a btree index to speed searching across those attributes. With this particular data schema one could quickly query for things like "all sendmail logs occurring between date1 and date2"



## Syslog-Ng/SQL

- Set up communication via FIFO

```
# mkfifo /tmp/msql.pipe
# mysql -u root --password=passwd syslog < /dev/mysql.pipe
```

Syslog-ng writes to one side of the FIFO mysql reads from the other! How cool is that?

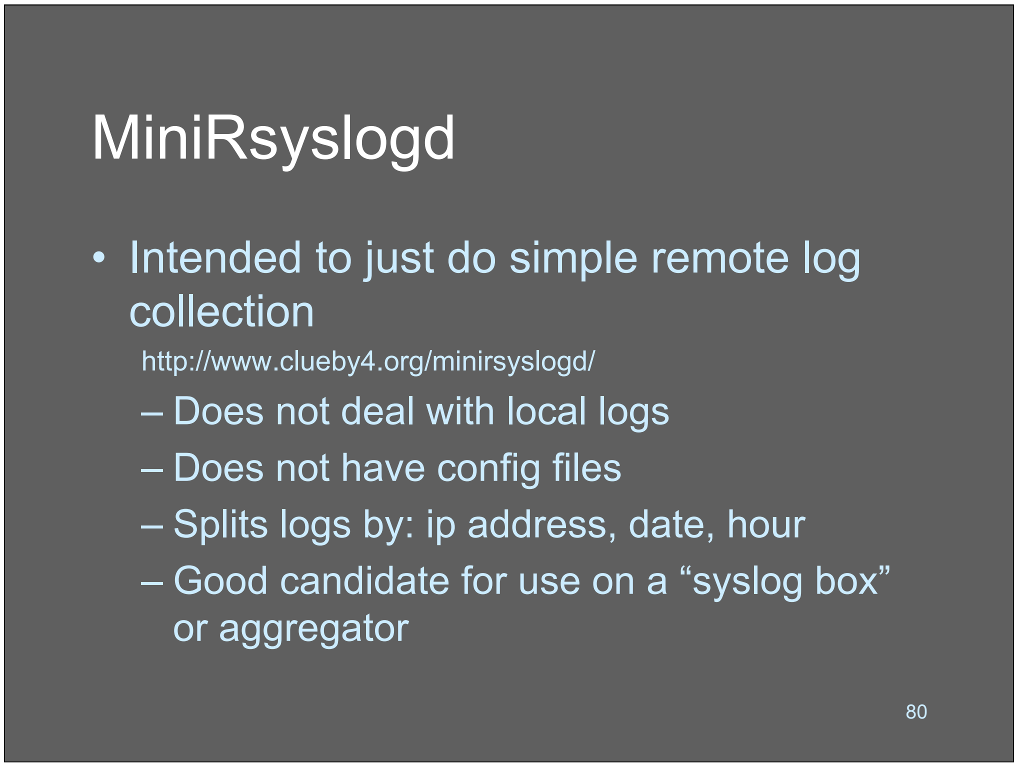
79

Here’s the sneaky part!

We create a FIFO filesystem object using the mkfifo command and give it the same name as the pipe(“/tmo/mysql.pipe”) in our syslog-ng config file. Since the pipe is in /tmp it may  get cleaned up whenever the system reboots - depending on your system. You may wish to keep it someplace else (in a protected directory) where it won’t be tampered with.

Now, we launch an interactive SQL session, as a privileged user, and we tell it to read its input from our FIFO. The SQL engine now sees a stream of insert commands and executes them as long as they keep coming in the pipe!

This setup can be surprisingly efficient and may work fine for large loads, if your hardware platform is powerful enough.



## MiniRsyslogd

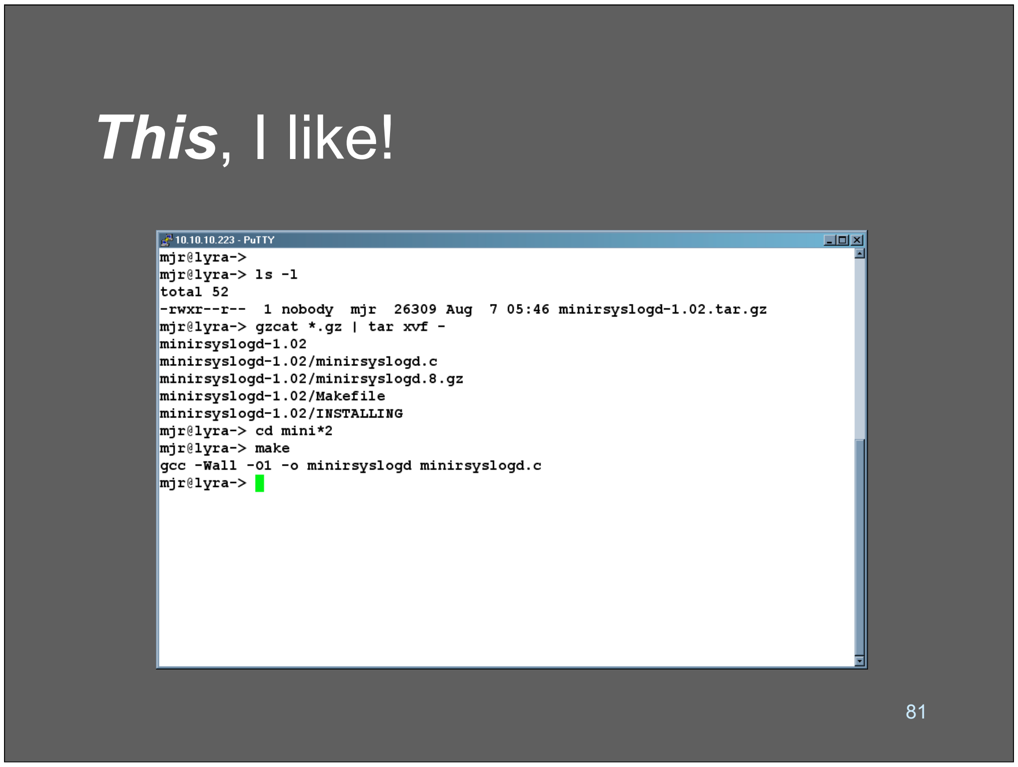
- Intended to just do simple remote log collection
  http://www.clueby4.org/minirsyslogd/
  - Does not deal with local logs
  - Does not have config files
  - Splits logs by: ip address, date, hour
  - Good candidate for use on a “syslog box” or aggregator

80

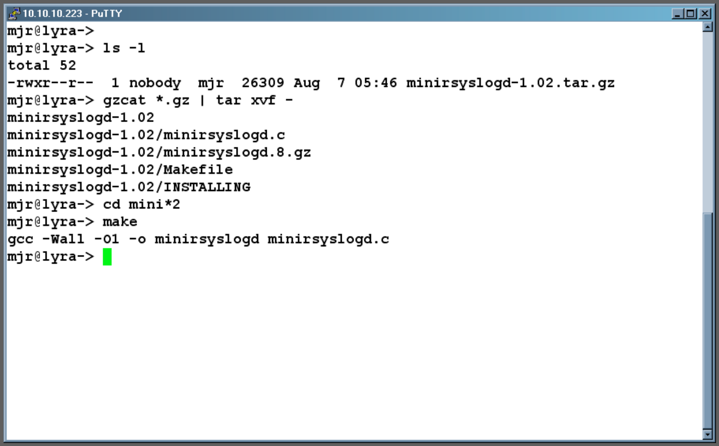
Minirsyslogd is the antithesis of syslog-ng. Where syslog-ng’s options have options, minirsyslogd has no options at all. Where syslog-ng reads its config file with a yacc-generated parser, minirsyslog doesn’t even need a configuration file.

By default, minirsyslog splits its logs by source address, date, and hour. It’s fast, simple, and easy to chroot to someplace safe in your filesystem.

Minirsyslog *only* accepts remote data; the local logs are ignored. If you need local logs, you’ll need to come up with a way of getting them into minirsyslogd on your own.



## *This*, I like!

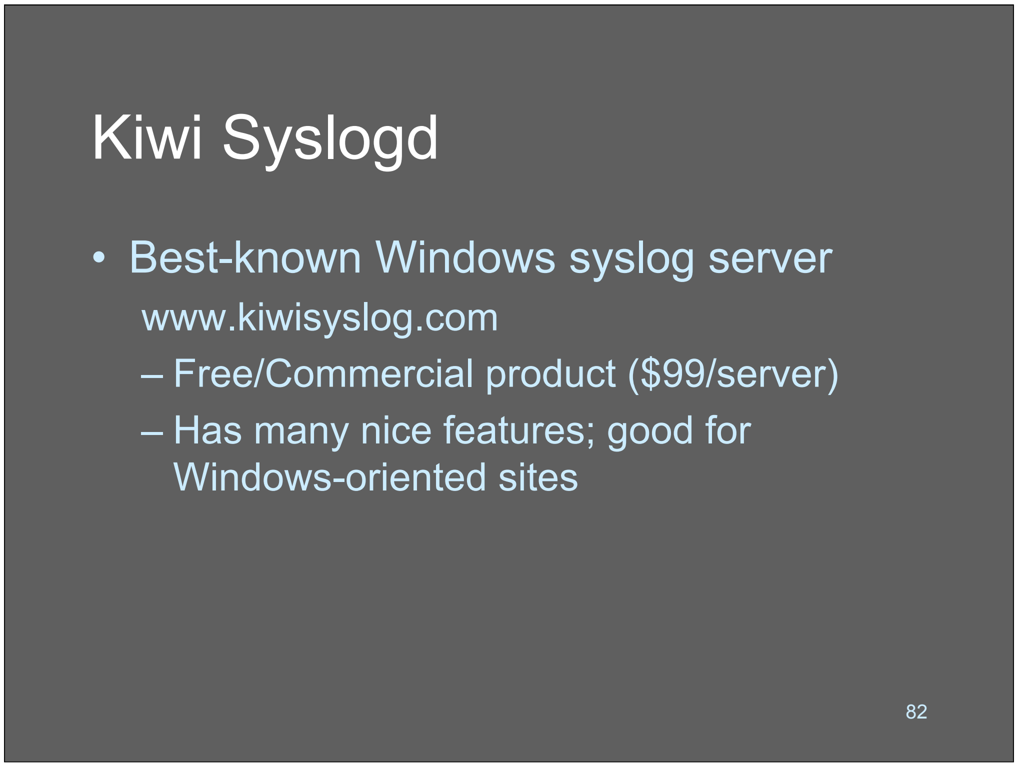


```
mjr@lyra->
mjr@lyra-> ls -l
total 52
-rwxr--r--  1 nobody  mjr  26309 Aug  7 05:46 minirsyslogd-1.02.tar.gz
mjr@lyra-> gzcat *.gz | tar xvf -
minirsyslogd-1.02
minirsyslogd-1.02/minirsyslogd.c
minirsyslogd-1.02/minirsyslogd.8.gz
minirsyslogd-1.02/Makefile
minirsyslogd-1.02/INSTALLING
mjr@lyra-> cd mini*2
mjr@lyra-> make
gcc -Wall -O1 -o minirsyslogd minirsyslogd.c
mjr@lyra->
```

81

(Extra credit if you can tell me the song that was the lead-in for, and the singer)

Building minirsyslogd is trivial. I really like seeing code that is small, portable, and uncomplicated!



## Kiwi Syslogd

- Best-known Windows syslog server
  www.kiwisyslog.com
  - Free/Commercial product ($99/server)
  - Has many nice features; good for Windows-oriented sites

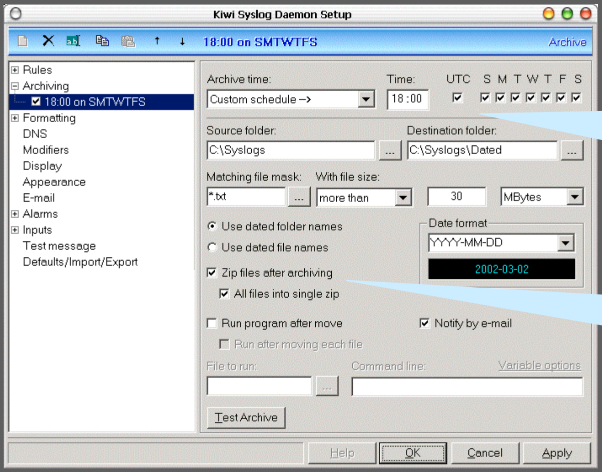
82

Kiwi syslogd is probably the best known of the Windows syslog servers. If you need to do syslogging and you’re in a Windows environment, this is the tool for you!

Kiwi offers the product for free/commercial registration use for $99, which is a low price for an excellent product.

In the next few slides, I’ve assembled a few screenshots from kiwi software’s website - they’ll give a flavor of what the product can do.

## Kiwi Syslogd *(cont)*


Kiwi Syslog Daemon Setup

18:00 on SMTWTFS — Archive

Archive time: Custom schedule -->  Time: 18 :00  UTC [✓]  S M T W T F S [✓] [✓] [✓] [✓] [✓] [✓] [✓]

Source folder: C:\Syslogs  Destination folder: C:\Syslogs\Dated

Matching file mask: *.txt  With file size: more than 30 MBytes

(●) Use dated folder names
( ) Use dated file names

Date format: YYYY-MM-DD  2002-03-02

[✓] Zip files after archiving
[✓] All files into single zip

[ ] Run program after move  [✓] Notify by e-mail
[ ] Run after moving each file

File to run:  Command line:  Variable options

Nice options for archival

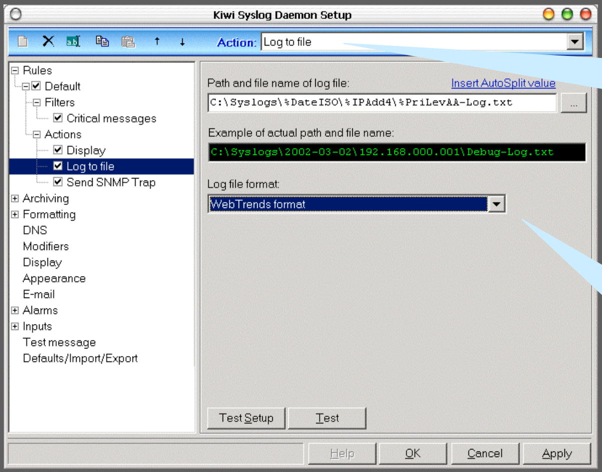
Auto-zip is a cool idea!

83

Here we see the archive configuration screen for kiwisyslogd. You are able to control the archival interval, format of file names, and even compression program to use for long-term storage.

There's even an option for another program to run whenever an archive is moved, and the option to notify an administrator by Email. If you wanted to, you could have the "other program" copy a backup of the archive to a queue for a CDROM burner or tape store.

## Kiwi Syslogd *(cont)*



Action: Log to file

Path and file name of log file:

```
C:\Syslogs\%DateISO\%IPAdd4\%PriLevAA-Log.txt
```

Example of actual path and file name:

```
C:\Syslogs\2002-03-02\192.168.000.001\Debug-Log.txt
```

Log file format: WebTrends format

ODBC is an option, so is ICQ/AIM!

Various formats; this is nice

84

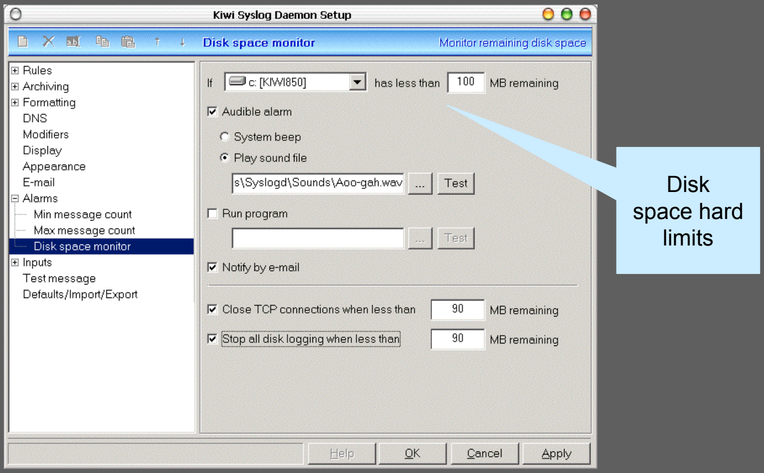
Here we see kiwi syslogd’s option screen for how to log data to a file. We’re specifying the logfile name and are adding macros to the name such as

%IPAdd4

which expands to the sender’s IPV4 address.

Notice the “Action” bar at the top? This is where we set it up to record to a file; we could just as easily have it generate an SNMP trap or send data directly to an ODBC database.

## Kiwi Syslogd *(cont)*



Disk space monitor — Monitor remaining disk space

- Rules
- Archiving
- Formatting
- DNS
- Modifiers
- Display
- Appearance
- E-mail
- Alarms
  - Min message count
  - Max message count
  - Disk space monitor
- Inputs
- Test message
- Defaults/Import/Export

If c: [KIWI850] has less than 100 MB remaining

☑ Audible alarm

○ System beep

● Play sound file

s\Syslogd\Sounds\Aoo-gah.wav ... Test

☐ Run program

☑ Notify by e-mail

☑ Close TCP connections when less than 90 MB remaining

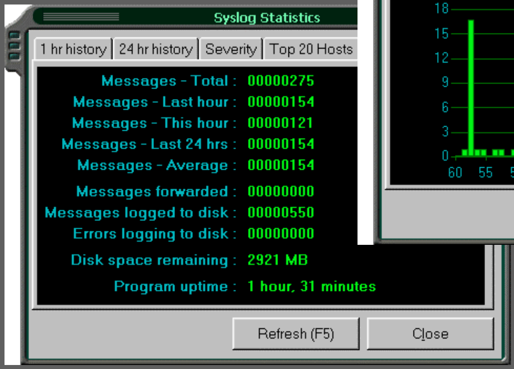
☑ Stop all disk logging when less than 90 MB remaining

85

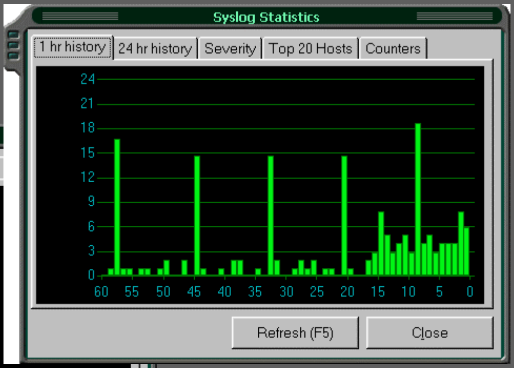
This shows the disk usage monitor panel. In this frame, we are allowed to set alarm levels and actions based on usage rates in the log area.

The audible alarms are particularly cute. “Help Help! I am out of disk space! Help!”

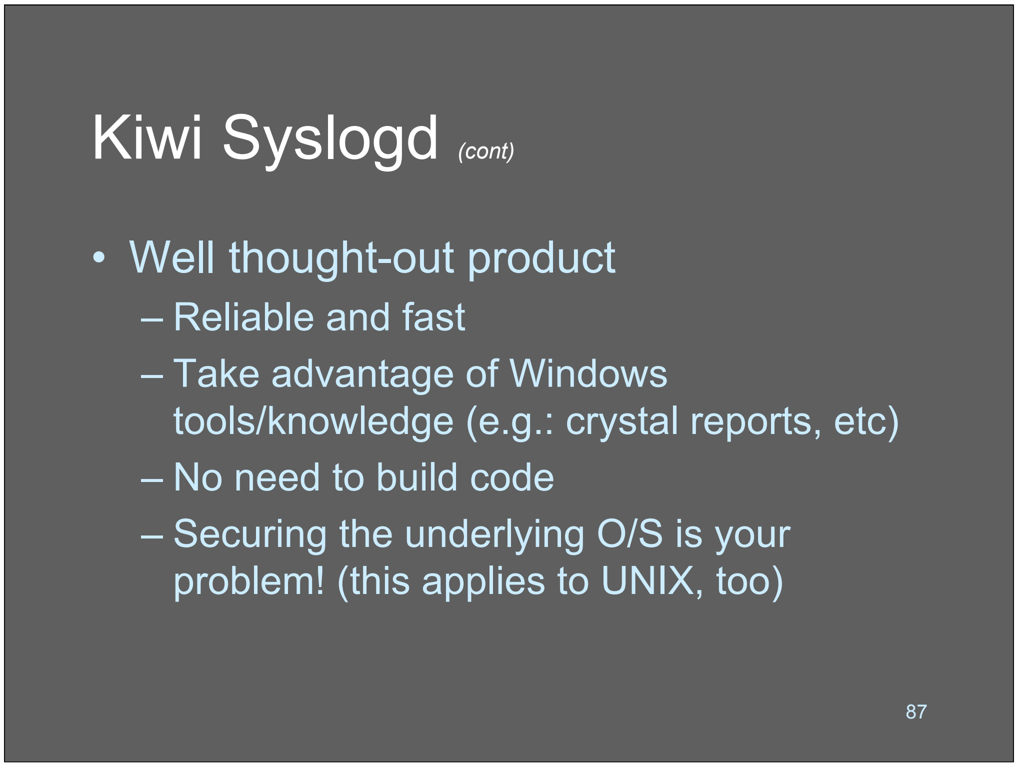
## Kiwi Syslogd *(cont)*



```
       Messages - Total :  00000275
   Messages - Last hour :  00000154
   Messages - This hour :  00000121
  Messages - Last 24 hrs :  00000154
     Messages - Average :  00000154
     Messages forwarded :  00000000
 Messages logged to disk :  00000550
   Errors logging to disk :  00000000
    Disk space remaining :  2921 MB
         Program uptime :  1 hour, 31 minutes
```



Kiwi syslogd also has some nice “eye candy” modules that you can keep running on your screen in the backgroun. Here we are looking at the log message entry rate (what does it mean? Spikes might be an indicator of trouble…)  and some statistics about the running state of the server.

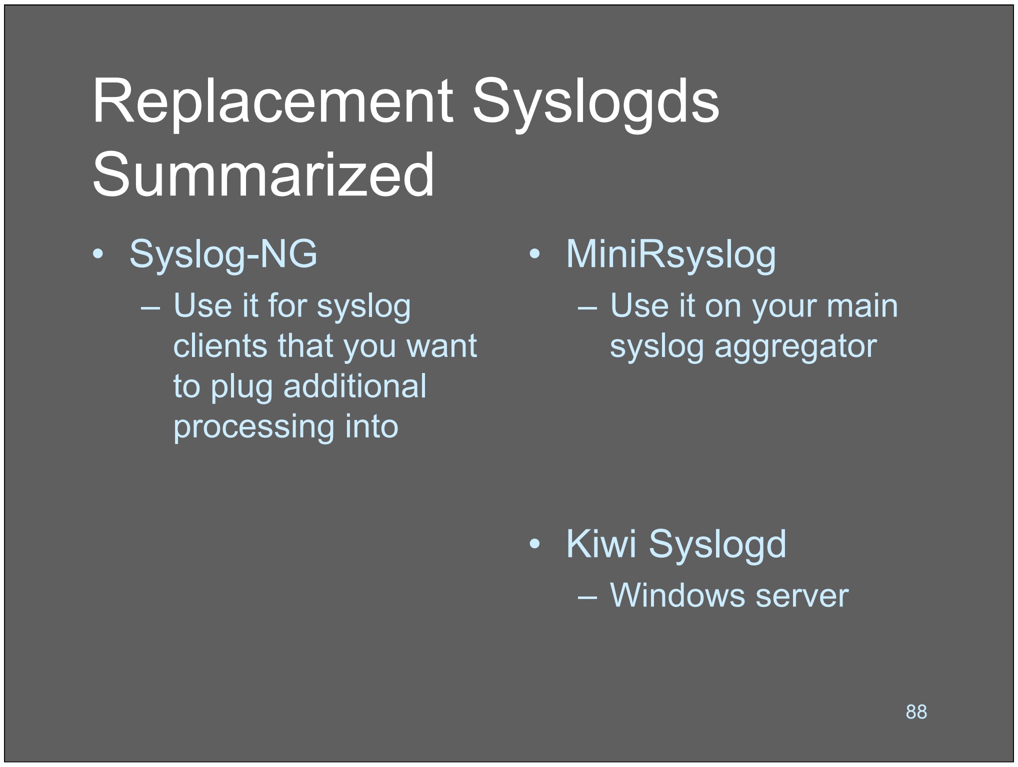


## Kiwi Syslogd *(cont)*

- Well thought-out product
  - Reliable and fast
  - Take advantage of Windows tools/knowledge (e.g.: crystal reports, etc)
  - No need to build code
  - Securing the underlying O/S is your problem! (this applies to UNIX, too)

87

If you're in a Windows environment, it's really worth taking a look at kiwi syslogd. The Windows-oriented capabilities of the product may come in handy; especially if most of your work-flow and tools are one a Windows machine.



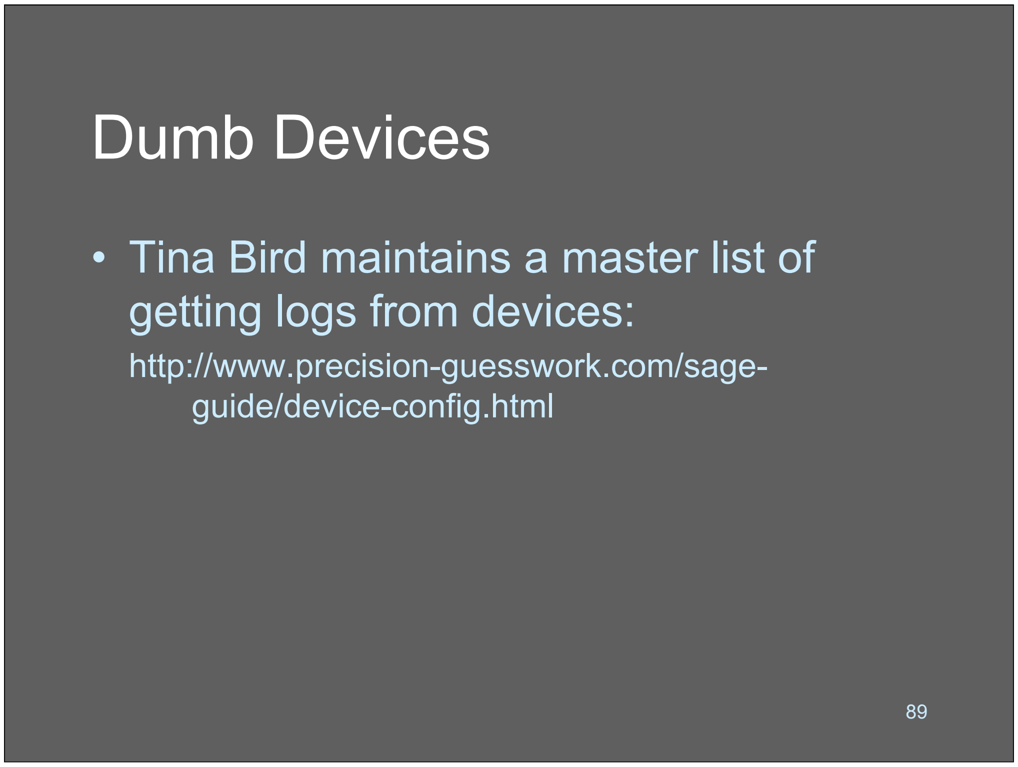
## Replacement Syslogds Summarized

- Syslog-NG
  - Use it for syslog clients that you want to plug additional processing into
- MiniRsyslog
  - Use it on your main syslog aggregator
- Kiwi Syslogd
  - Windows server

88

There is no “right answer” on which syslog server to use! These three are all terrific, depending on your objectives, environment, patience level, and skills.

Plan and consider carefully, before you decide which tool you want to use!

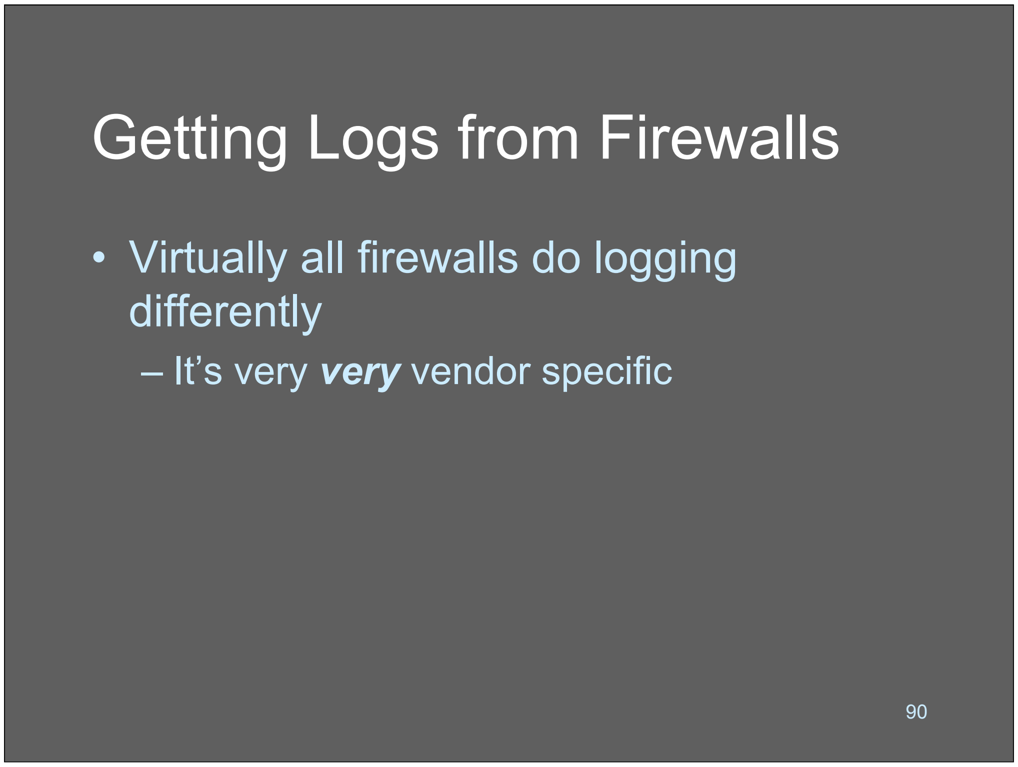


## Dumb Devices

- Tina Bird maintains a master list of getting logs from devices:
  http://www.precision-guesswork.com/sage-guide/device-config.html

89

Getting data from “dumb” (non syslog aware) devices can be a real pain in the ASCII! Virtually every dumb device has its own quirks and there are too many of them to cover in this tutorial. Tina Bird maintains a list of tricks for getting logs from dumb devices on her website and on loganalysis.org.



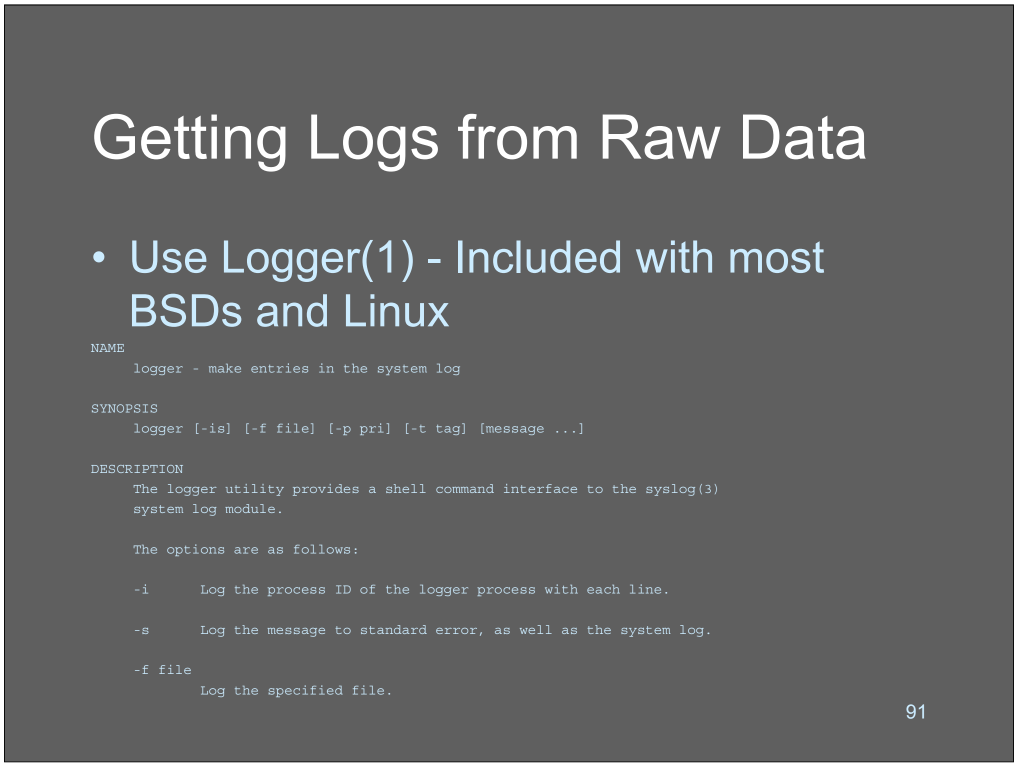
## Getting Logs from Firewalls

- Virtually all firewalls do logging differently
  - It's very ***very*** vendor specific

90

Firewall logs are one of the most important logs you can get! And, unfortunately, the configurations for them are very system-specific. Some firewalls log to proprietary formats (which can often be taken apart with a perl script) while others need to be polled using SNMP (ugh!) - regardless of how you need to get the logs off them, you can probably find a how-to by searching the web for "*product* syslog enable"

Some products don't even do it the same way from one version to the next, so just plan on doing a little searching.



## Getting Logs from Raw Data

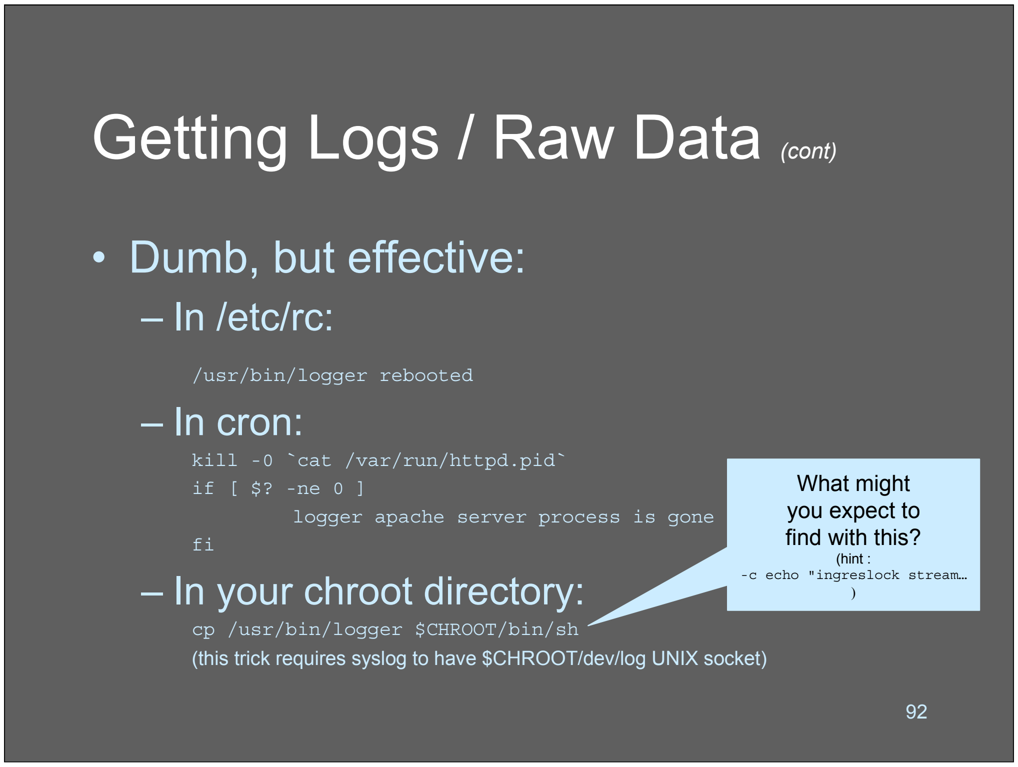
- Use Logger(1) - Included with most BSDs and Linux

```
NAME
     logger - make entries in the system log

SYNOPSIS
     logger [-is] [-f file] [-p pri] [-t tag] [message ...]

DESCRIPTION
     The logger utility provides a shell command interface to the syslog(3)
     system log module.

     The options are as follows:

     -i      Log the process ID of the logger process with each line.

     -s      Log the message to standard error, as well as the system log.

     -f file
             Log the specified file.
```

91

If you're dealing with a dumb device or app that creates its own log or sends its errors someplace else, the popular way of injecting those log messages into a syslog stream is to use the "logger" utility. Logger lets you send a message into syslog manually, and can also submit an entire file of messages.

Typical use of logger is to grep values out of other data sources and to inject them into the logging stream, or to create a log message when a specific event happens. Logger is also very useful within shell scripts if they are performing interesting transactions or need to record errors.



## Getting Logs / Raw Data *(cont)*

- Dumb, but effective:
  - In /etc/rc:

```
/usr/bin/logger rebooted
```

  - In cron:

```
kill -0 `cat /var/run/httpd.pid`
if [ $? -ne 0 ]
        logger apache server process is gone
fi
```

  - In your chroot directory:

```
cp /usr/bin/logger $CHROOT/bin/sh
```

(this trick requires syslog to have $CHROOT/dev/log UNIX socket)

What might you expect to find with this?
(hint :
`-c echo "ingreslock stream…`
)

92

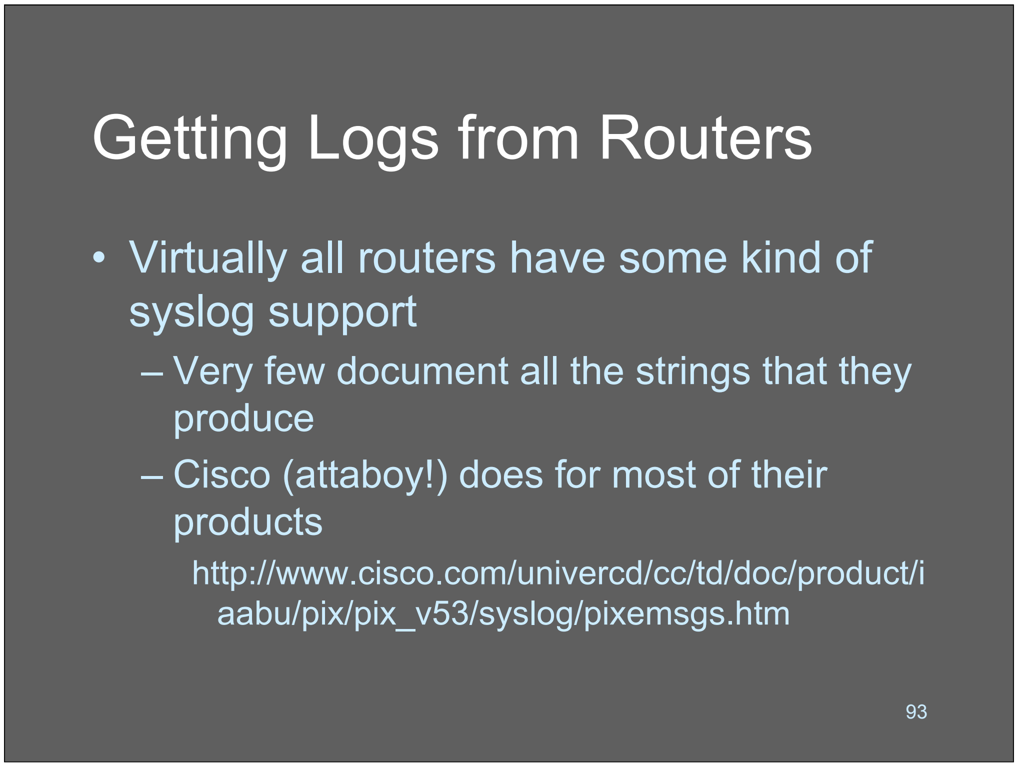
Consider putting calls to logger in interesting places within your system. These are just a few simple examples.

In the first example, logger is called in a system start-up script to record the fact that the system had been restarted. There are lots of cases where this might come in handy.

The second example uses logger and kill as a way of testing to see if a process is still running. Kill -0 on a process (assuming you have permission) will exit with a non-zero value if the process is not running; we can then log the error.

Finally, logger can be used as a handy burglar alarm in chrooted areas. To make this work, you'll need to have an active UNIX domain /dev/log listener (easy to do with syslog-ng)  You can also "wrap" logger to collect command lines for programs that you wish to track:

```
#!/bin/sh
echo "called $0 $*" | logger
exec $*
```



## Getting Logs from Routers

- Virtually all routers have some kind of syslog support
  - Very few document all the strings that they produce
  - Cisco (attaboy!) does for most of their products

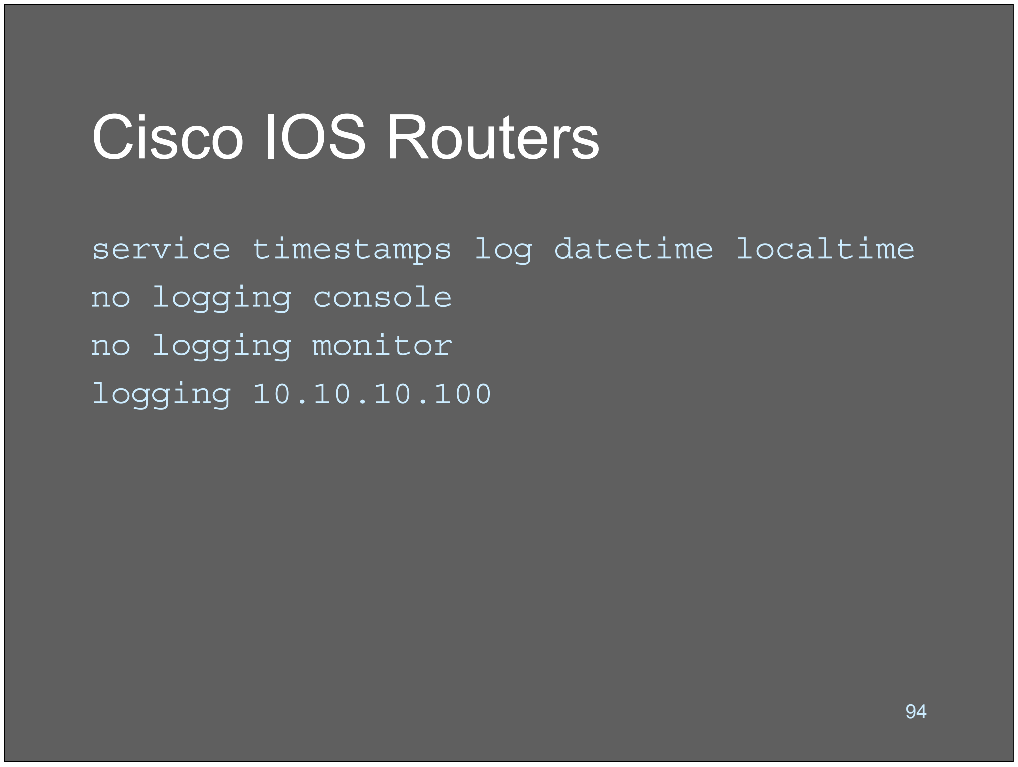
    http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix_v53/syslog/pixemsgs.htm

93

Many devices have syslog support, but not all of them document all their possible syslog messages. In fact, relatively few do. If you want to get logging data from your devices, make sure you actually buy devices that provide it!

Cisco, I must say, does a terrific job of making logging message dictionaries available. If you search Cisco's site for "*productname* log message catalog" or "*productname* log message dictionary"

For other products, I've found that searching the web for those same search terms usually yields results.



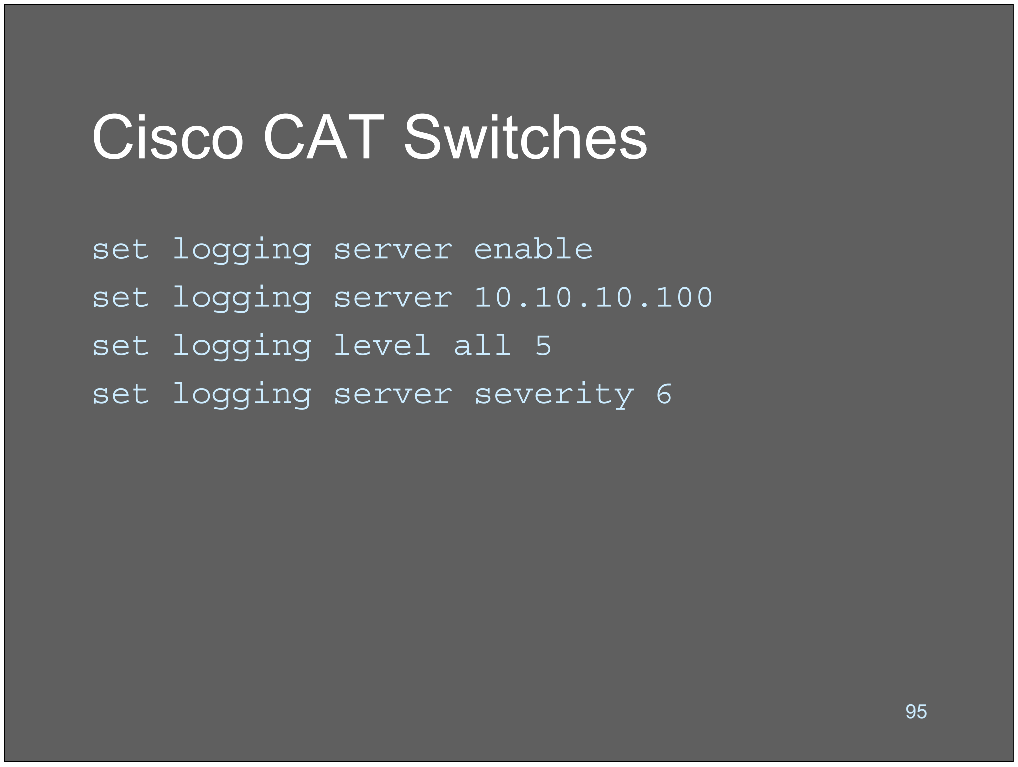
## Cisco IOS Routers

```
service timestamps log datetime localtime
no logging console
no logging monitor
logging 10.10.10.100
```

94

This is an example of how to turn logging on in an IOS router.

We tell the router not to bother logging to the console, and define our log host on the last line.

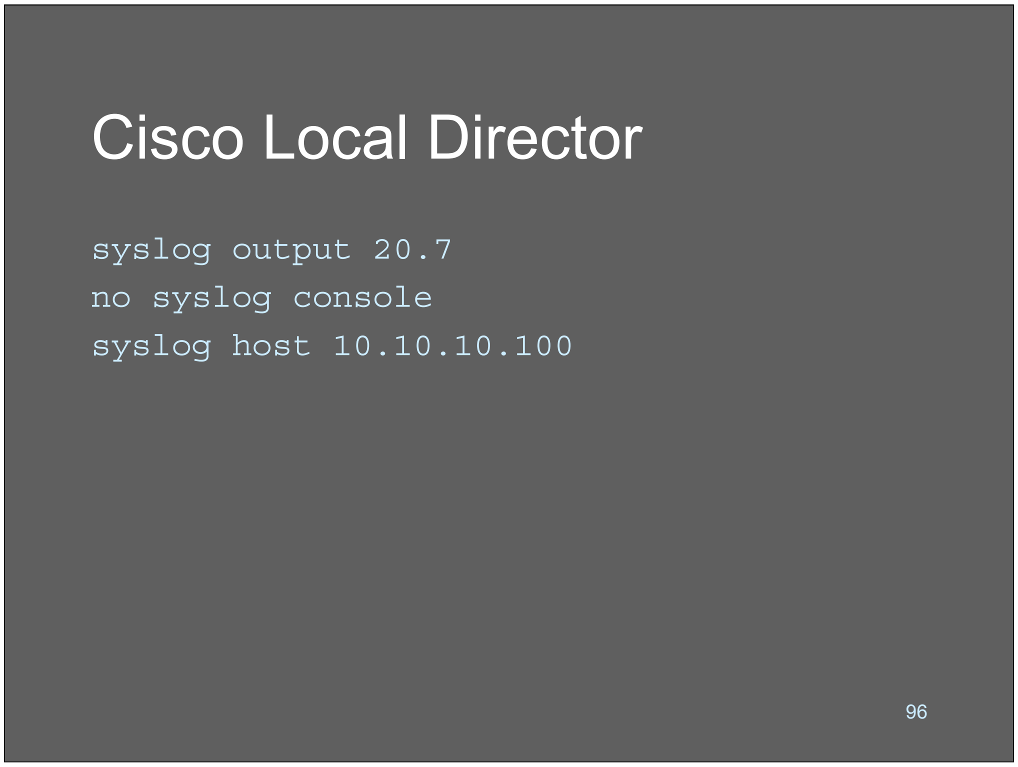


## Cisco CAT Switches

```
set logging server enable
set logging server 10.10.10.100
set logging level all 5
set logging server severity 6
```

95

This is how to enable logging in a Cisco Catalyst switch. We turn logging on and then direct it to the log server using “set logging server *iopaddress*”

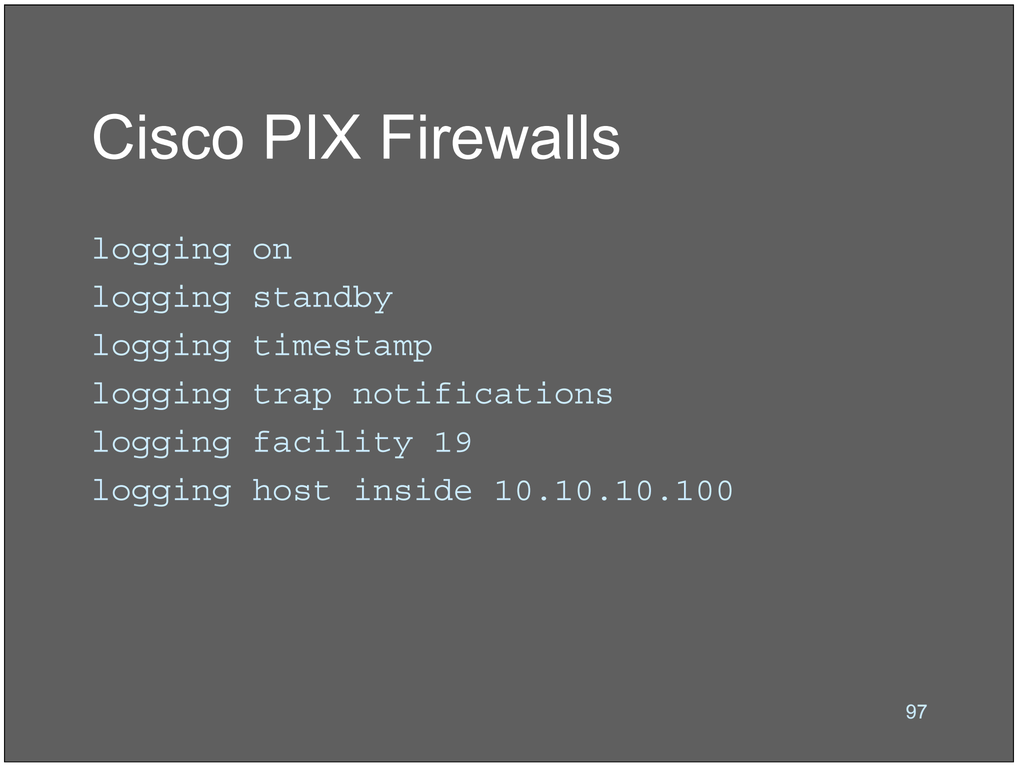


## Cisco Local Director

```
syslog output 20.7
no syslog console
syslog host 10.10.10.100
```

96

Cisco local director logging is enabled with this incantation. What does the “syslog output 20.7” mean?

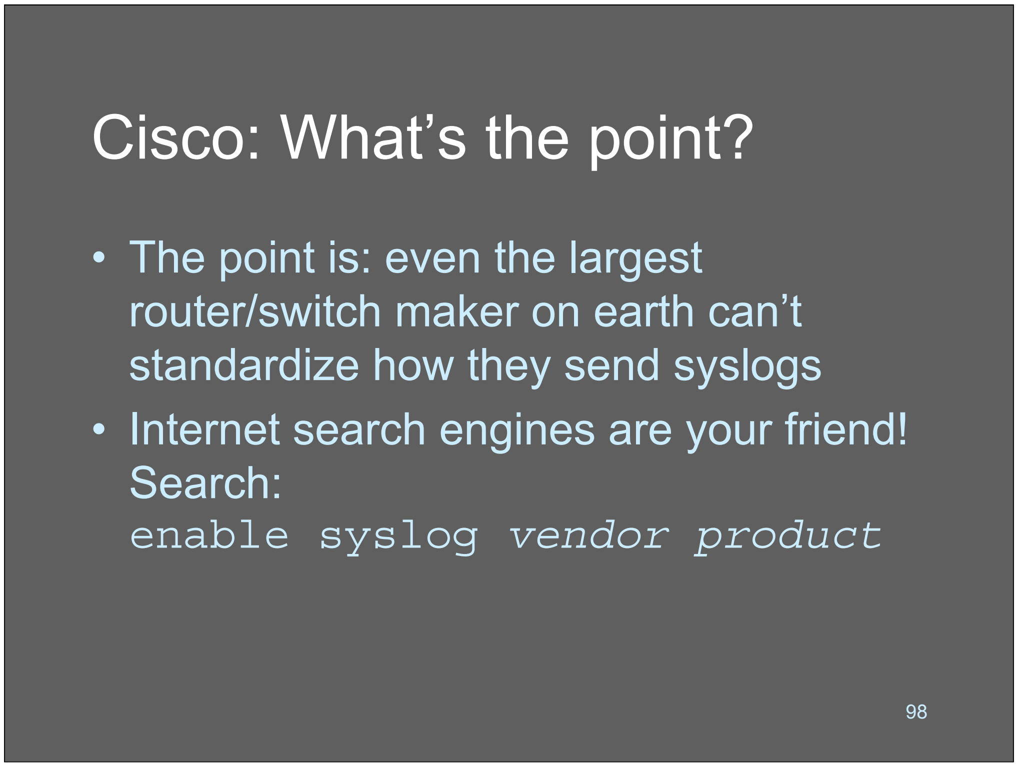


## Cisco PIX Firewalls

```
logging on
logging standby
logging timestamp
logging trap notifications
logging facility 19
logging host inside 10.10.10.100
```

97

Cisco's PIX firewalls use these commands to turn on logging.



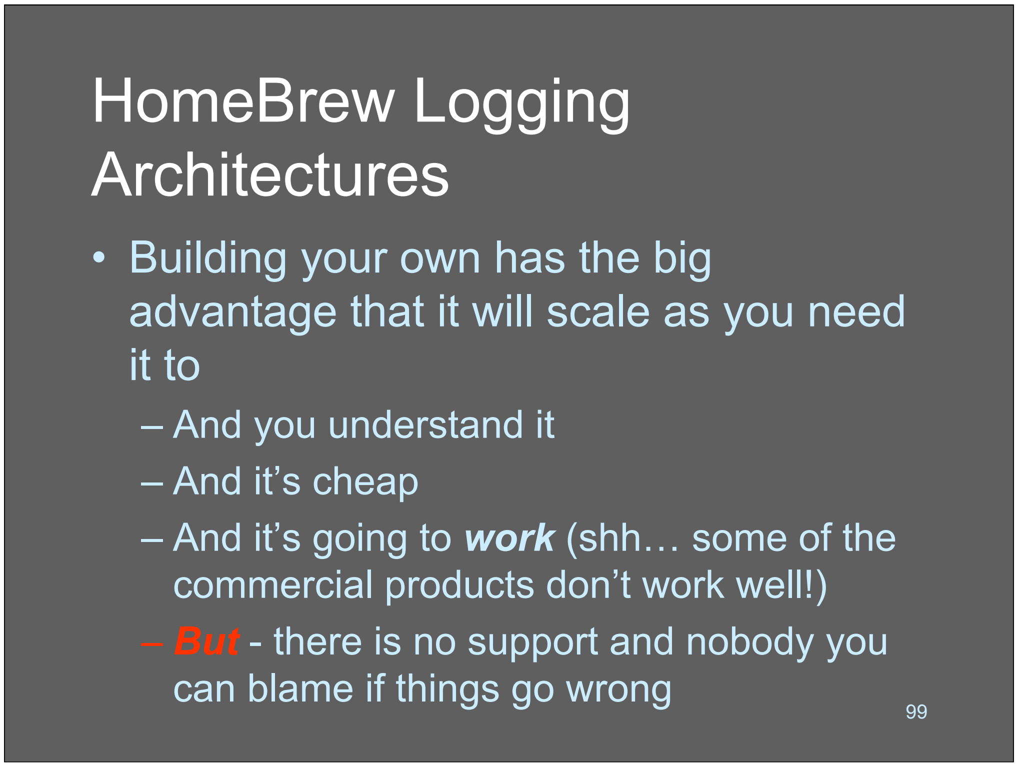
## Cisco: What’s the point?

- The point is: even the largest router/switch maker on earth can’t standardize how they send syslogs
- Internet search engines are your friend! Search:
  `enable syslog` *`vendor product`*

98

OK, what’s the real reason I just walked you through all those examples? Mostly to show you that even a huge vendor can’t standardize on something as simple as an administrative command-set that is consistent. Most of the technologies I just gave examples for were acquired by Cisco from other companies, and they still work pretty much the way they always did. Some of Cisco’s newer acquisitions don’t even support syslog at all (e.g.: Linksys routers) - we have no idea when or if support will be added.

The lesson you should learn from the last few slides is that for each product you want logging from, you’re going to probably need to do some research. The good news is that it’s pretty easy to find the magic incantation for any given product with a couple of quick internet searches.



## HomeBrew Logging Architectures

- Building your own has the big advantage that it will scale as you need it to
  - And you understand it
  - And it's cheap
  - And it's going to ***work*** (shh… some of the commercial products don't work well!)
  - ***But*** - there is no support and nobody you can blame if things go wrong

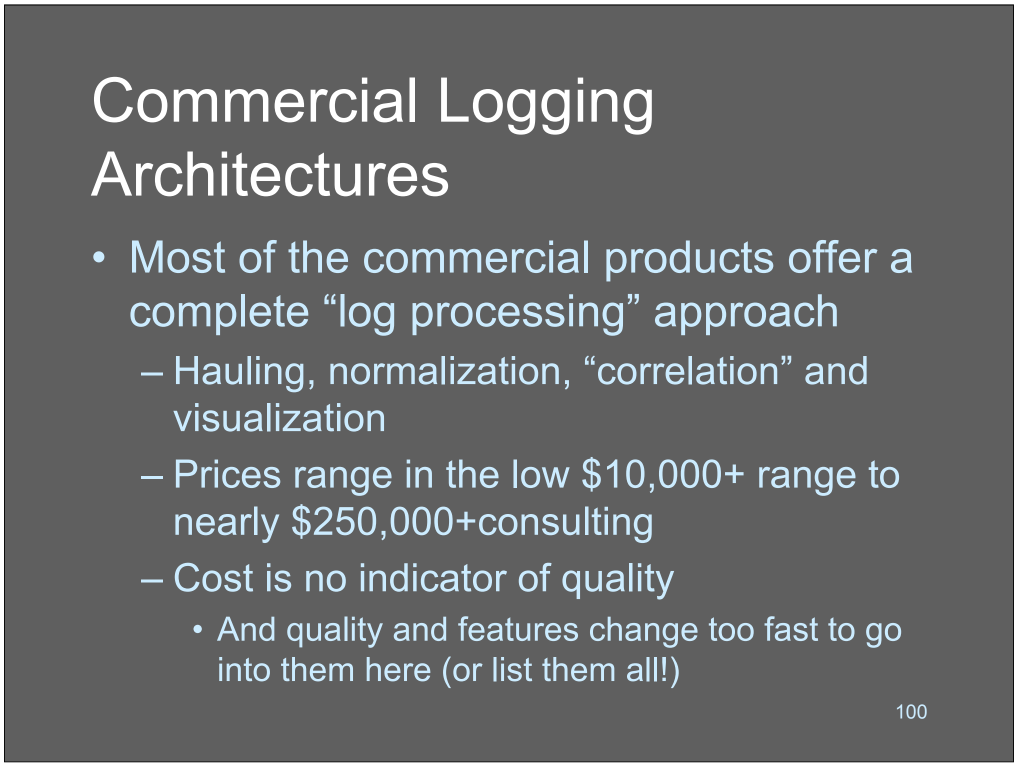
99

Let's start looking at what it takes to build a logging architecture for your organization. On the surface of things, it's just a bit of processing power, some bandwidth, and disk space. Below the surface are a few rocks to avoid (but we've already talked about those!) - so there are big advantages for the "do it yourself-er" if you want to go that route.

First and foremost, if you do it yourself, you'll actually understand how it works, and you'll be able to understand how it'll scale. By the time you've started building your own logging system, you'll have a good idea of message-rates you need to cope with, as well as the kind of reports you want to produce.

Building your own system means you'll have to build your own knowledge-base for tokenizing and parsing your log data. If you have a lot of unusual devices, you will *have to* regardless, since the commercial log aggregators don't support every possible client system. Before you embark on a log analysis architecture, make a realistic assessment of how diverse your systems are, and factor in the work-load that customization will represent.

As far as I am concerned, the only downside of building your own logging architecture is that you've got no support and nobody to blame if it doesn' twork. But the truth is, that's generally how it is, anyhow!



## Commercial Logging Architectures

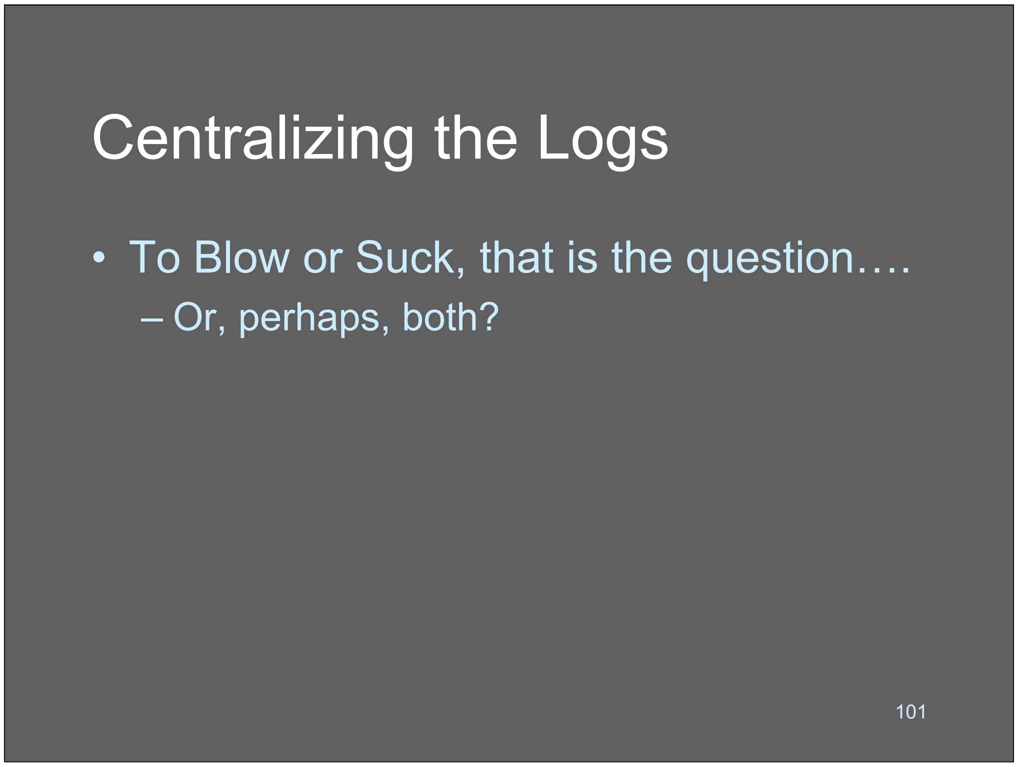
- Most of the commercial products offer a complete “log processing” approach
  - Hauling, normalization, “correlation” and visualization
  - Prices range in the low $10,000+ range to nearly $250,000+consulting
  - Cost is no indicator of quality
    - And quality and features change too fast to go into them here (or list them all!)

100

The commercial log aggregation solutions are fairly expensive stuff, though low-cost entrants are beginning to show up in the market.

The primary value of the commercial systems is that they generally include the knowledge-base of parsing and normalization rules that you’d have to build if you were setting up your own system. Before you proceed with the commercial products, make an inventory of the systems (platform, O/S, release level) that you will want to process log data from, and see how the vendor supports them.

Be aware of the fact that if your vendor is customizing their knowledge-base for your systems, you’re effectively paying them big bucks to make their product more marketable. Some of the commercial vendors started out only supporting a small number of platforms/versions and have let their customers dictate platform support. Know what you are paying for!

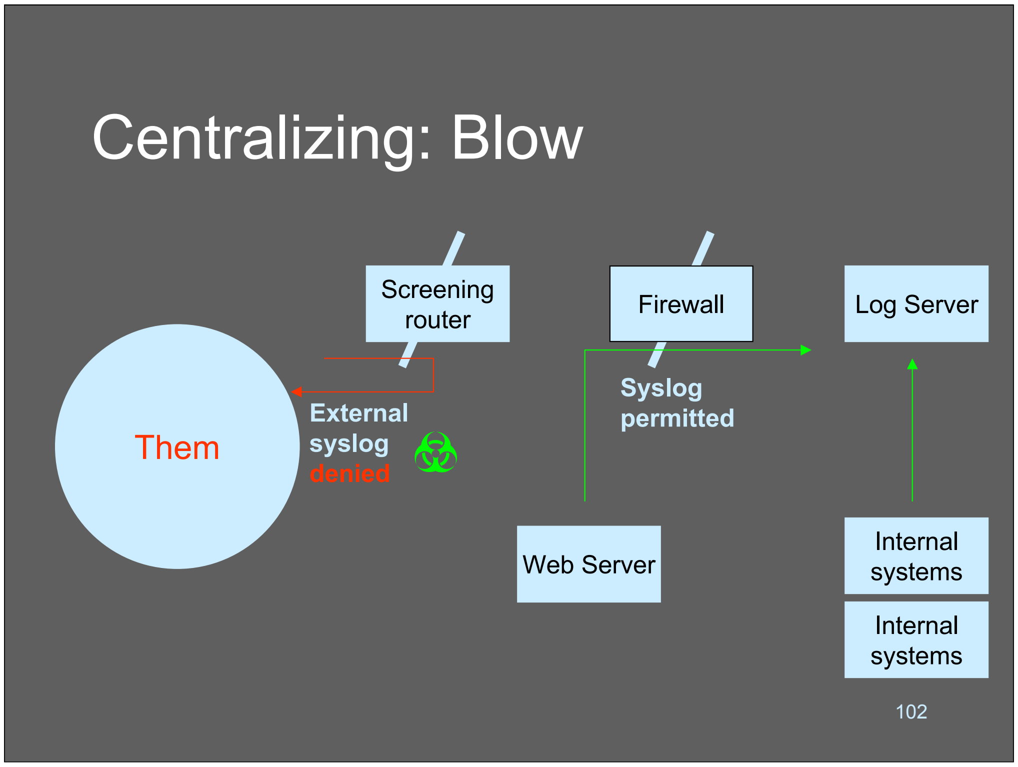


## Centralizing the Logs

- To Blow or Suck, that is the question….
  - Or, perhaps, both?

101

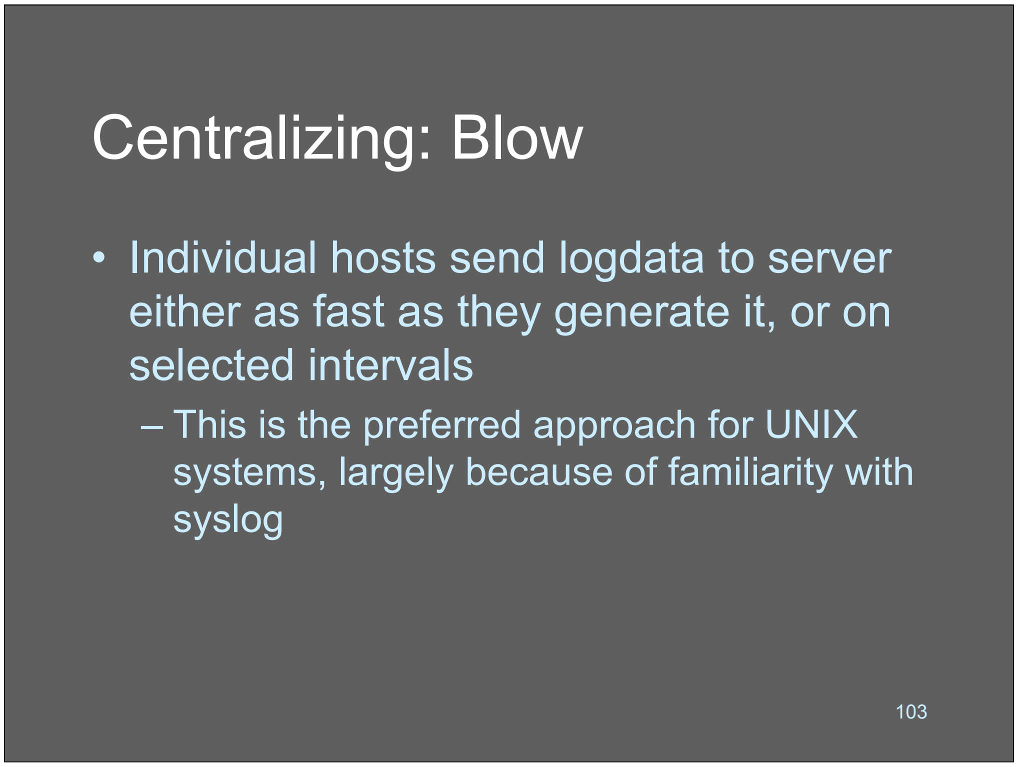
Let's look at some methods of centralizing your log data. There are two basic approaches, and a hybrid approach, and we'll examine them in sequence. Mainly, the question of pushing or pulling revolves around how the log server communicates relative to firewalls.



## Centralizing: Blow

This diagram represents the typical “blow” architecture for centralizing logs. We have a device (a web server) outside of our firewall, that needs to log data to the inside. A “tiny hole” is poked through the firewall’s rules to allow the logging data to reach the central server using some kind of logging protocol (this could be syslog or it could be tunnelled over SSH or whatever). Internal systems are able to forward their data directly to the log server. If you’re paranoid you can put a filtering router in front of the log server(good) or configure the log server with iptables or in-kernel firewalling to reduce the chance of “accidents.”

Note that in order to protect the exterior server, we call for filtering incoming syslog data at the screening boundary router. This prevents outsiders from injecting spurious log messages into the logging stream.



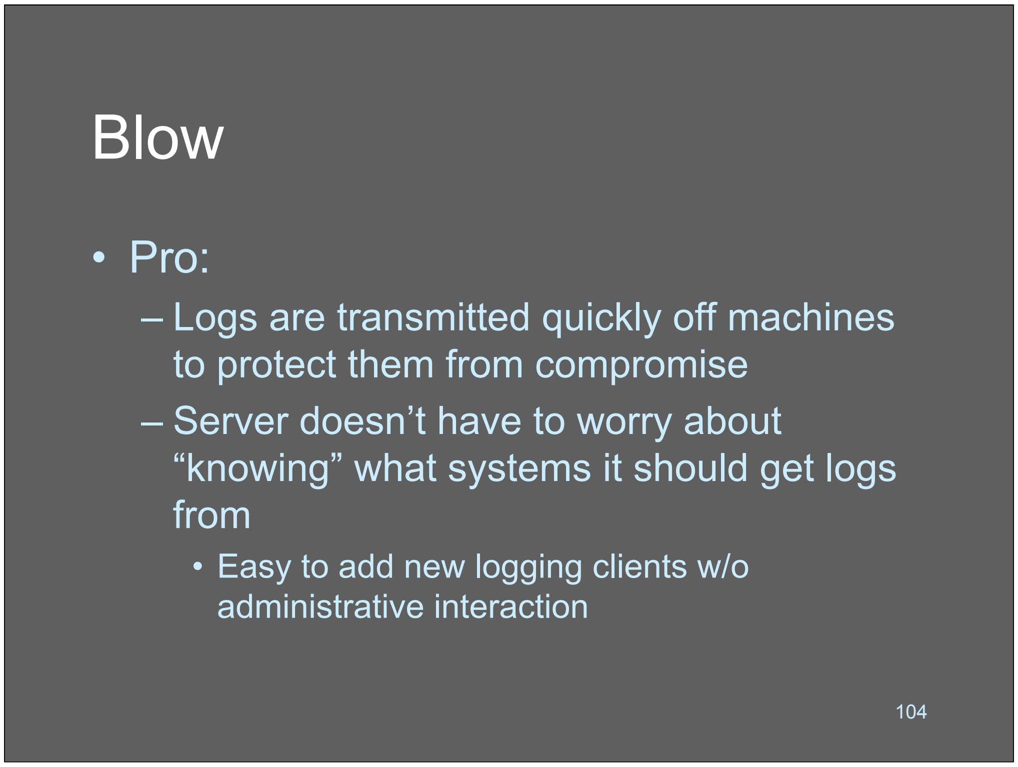
## Centralizing: Blow

- Individual hosts send logdata to server either as fast as they generate it, or on selected intervals
  - This is the preferred approach for UNIX systems, largely because of familiarity with syslog

103

In the blow architecture, logging sources generally send data as soon as they generate it. If you are dealing with low-speed links or bursty traffic, some sites may choose to batch traffic up and send it at intervals, but this approach is rare.

Basically, the blow architecture is UNIX' syslog writ large. You can build this using stock syslogd clients simply by defining a loghost.



## Blow

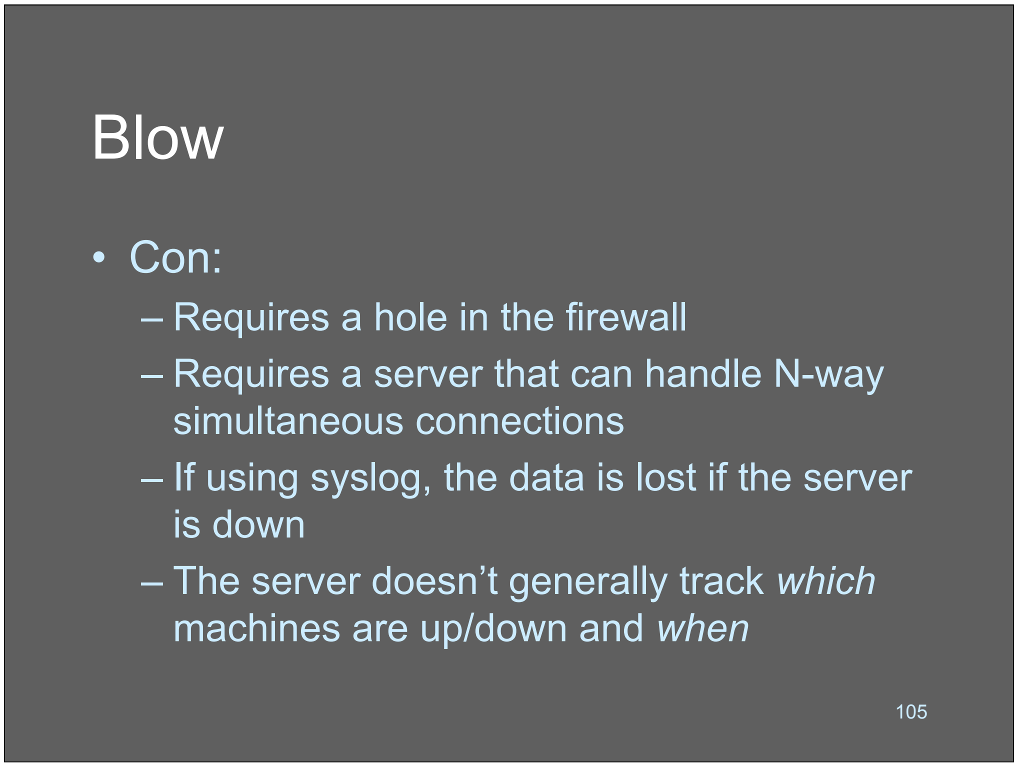
- Pro:
  - Logs are transmitted quickly off machines to protect them from compromise
  - Server doesn't have to worry about "knowing" what systems it should get logs from
    - Easy to add new logging clients w/o administrative interaction

104

The advantages of the blow approach are twofold: management and security.

In practice, if a hacker compromises a system, they will immediately zap its logs. Having the syslogd forwarfing log records off the system in "realtime" greatly reduces the chance that events leading up to the intrusion will be erased.

From an administrative standpoint, the blow architecture tends to favor promiscuous transmission of data. Traditional UNIX syslogd would take messages from anyplace, and it was therefore very easy to add new logging clients: you simply turn them on and let them start transmitting. For administrators who are running the log server as a shared resource, this is attractive since you can just tell people, "send your logs here" and the server can begin to cope with them when they arrive. There's no need to add additional configuration rules in the server whenever a new client starts sending logs. For sites working with Windows logs or DHCP clients this is a very attractive approach.



## Blow

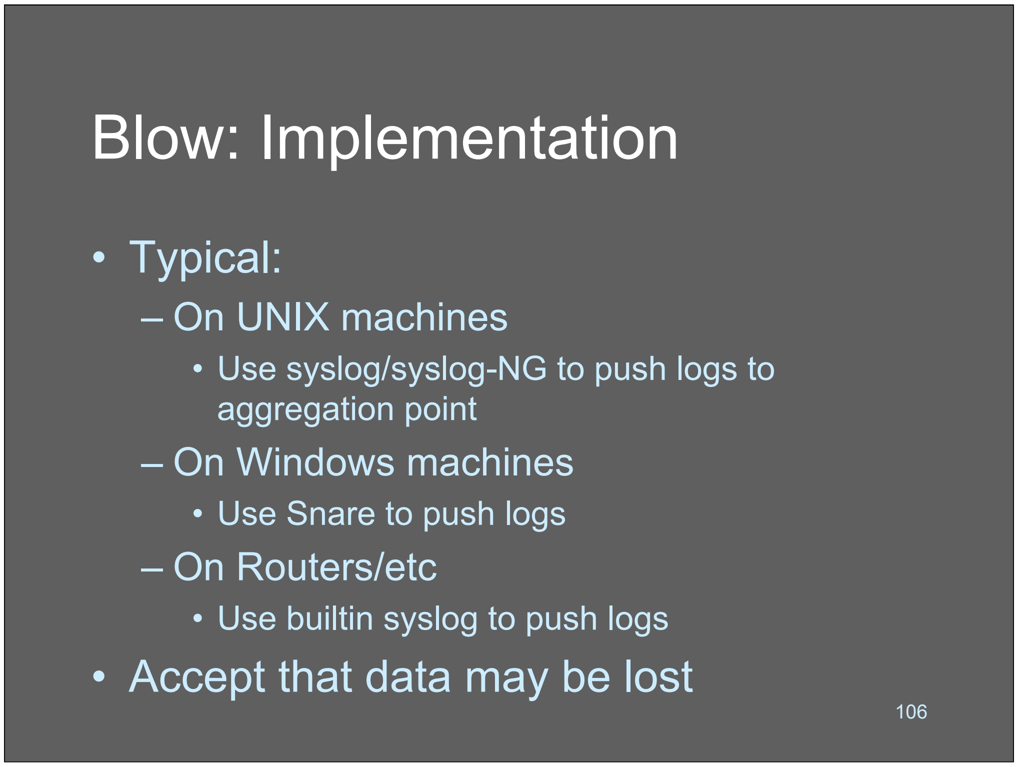
- Con:
  - Requires a hole in the firewall
  - Requires a server that can handle N-way simultaneous connections
  - If using syslog, the data is lost if the server is down
  - The server doesn't generally track *which* machines are up/down and *when*

105

The blow approach has a few disadvantages, to balance its advantages!

First off, since the clients are just spewing log data to the server, uptime at the server becomes an important consideration. In the event that the server is down or is overloaded, log messages will potentially get lost and are irrecoverable. Additionally, since the server doesn't really "know" what its clients are, it might not be able to "know' if they are supposed to be transmitting or receiving anymore. So it's hard to track reliability of the clients.

Lastly, there's the matter of that hole in the firewall. If you're letting the traffic in through the firewall, there is a potential that a vulnerability in the syslog server could be exploited from the outside, through the firewall. This could be disastrous. If you are building a blow architecture system logging architecture, I would strongly recommend using additional host security measures on the system logging server itself (chroot, trojan trapping, etc).



## Blow: Implementation

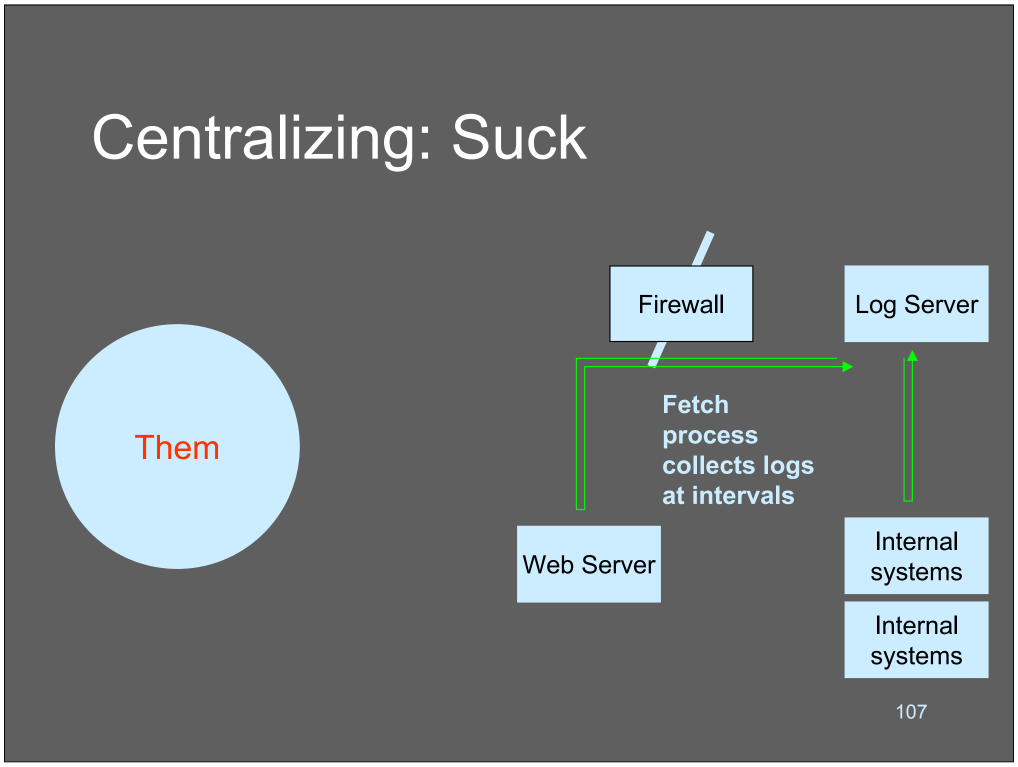
- Typical:
  - On UNIX machines
    - Use syslog/syslog-NG to push logs to aggregation point
  - On Windows machines
    - Use Snare to push logs
  - On Routers/etc
    - Use builtin syslog to push logs
- Accept that data may be lost

106

Building a blow architecture is the easiest form of log aggregation. If you're in a UNIX environment and do not expect large loads, you can just turn syslogd to forward messages to the central loghost. If you want to tunnel the data or send it over TCP, then use syslog-ng.

For windows machines you can use EventReporter or Snare to forward logs to the server. Routers and other devices can use their own internal logging routines.

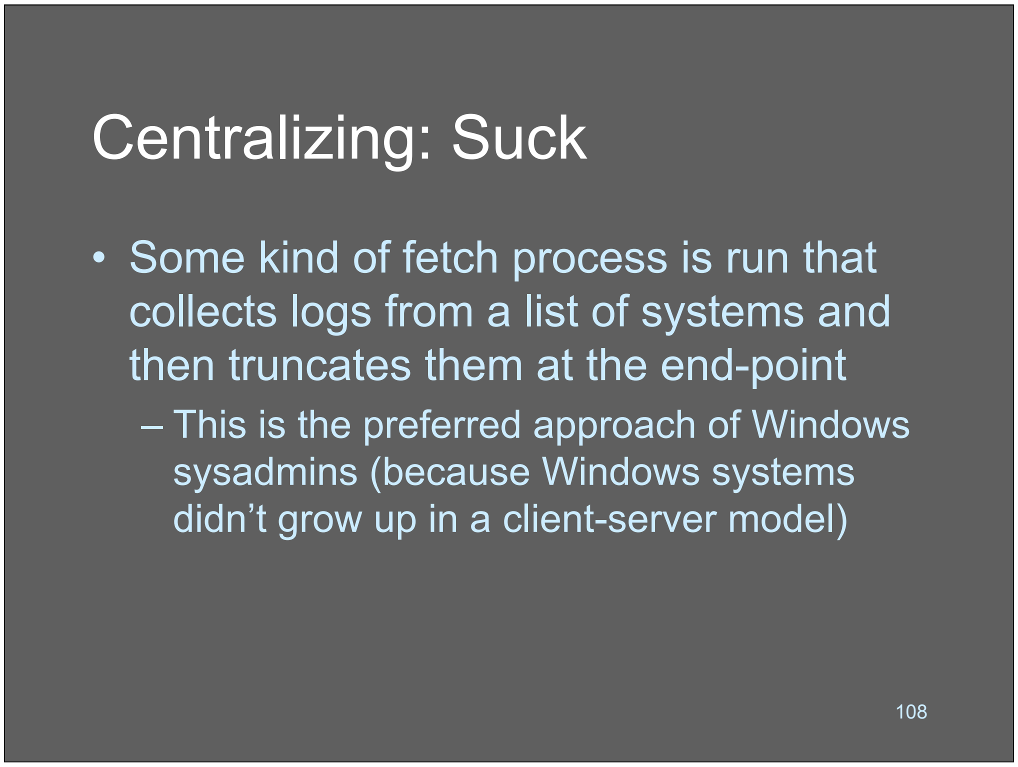
As we said earlier, the blow architecture is syslog writ large. If you use this approach you need to accept that some data may be lost. It's probably not a big deal, really.



## Centralizing: Suck

The suck aggregation approach is diagrammed above. Basically, the log server periodically reaches out through the firewall and collects logs from the clients. The fetch process is implementation dependent, though many facilities use SSH/SCP or S-FTP/FTP or even SNMP to collect the data.

This approach is popular with Windows log aggregators; domain privileges are used to perform the copies and log pruning on the devices.

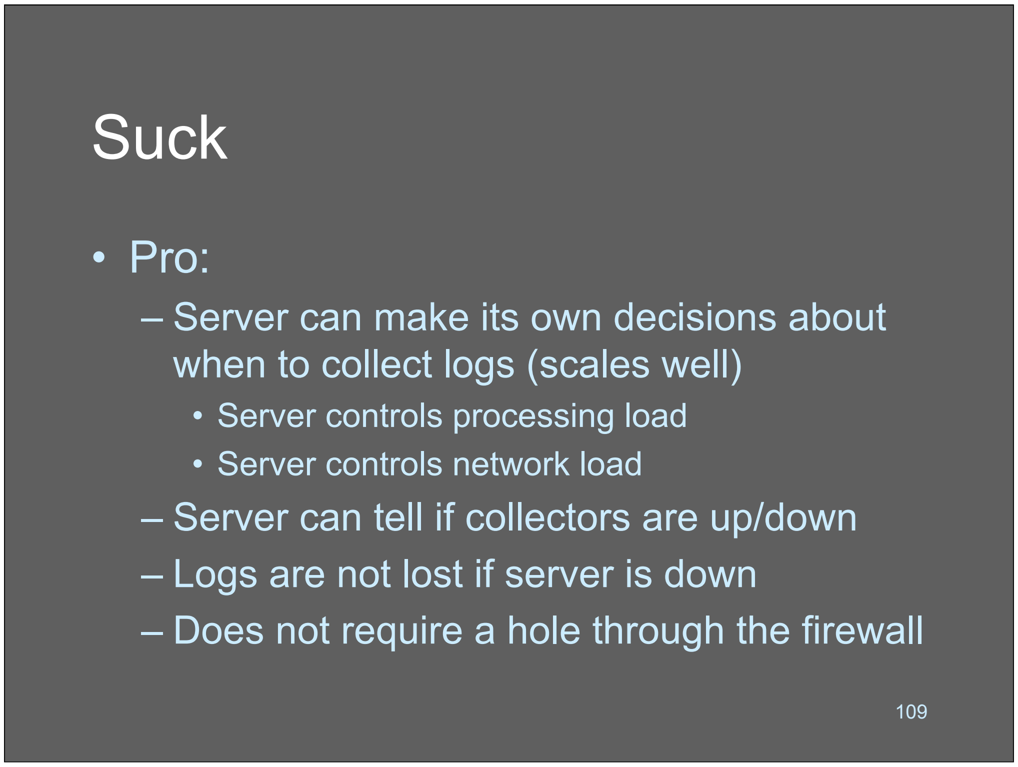


## Centralizing: Suck

- Some kind of fetch process is run that collects logs from a list of systems and then truncates them at the end-point
  - This is the preferred approach of Windows sysadmins (because Windows systems didn't grow up in a client-server model)

108

The suck architecture is primarily popular in the Windows world. I'm not sure why this is; I suspect it's because Windows system administrators are very uncomfortable about adding "one more agent" to desktops, or the administrative overhead of installing new software on many machines. Generally, agents are badly perceived by users and system administrators and it's not uncommon for users to turn them off and/or to blame them for performance and reliability problems.



## Suck

- Pro:
  - Server can make its own decisions about when to collect logs (scales well)
    - Server controls processing load
    - Server controls network load
  - Server can tell if collectors are up/down
  - Logs are not lost if server is down
  - Does not require a hole through the firewall

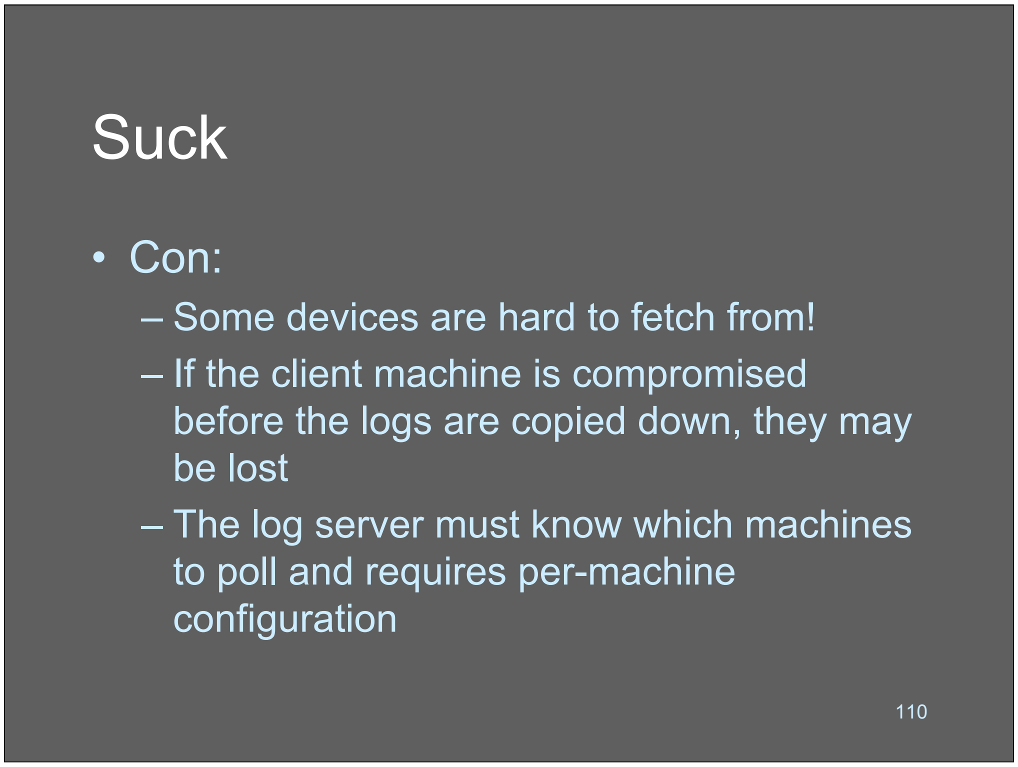
109

The main advantages of the suck model revolve around load-control of the system. The log server can choose its time and place to collect from, and can manage its processing load, the network load, and even the client load accordingly. In most cases, with this architecture, the client is pre-compressing the data before it is collected. This can result in considerable (on the order of 80%) bandwidth savings.

Additionally, the server can queue and retry if one of the clients is down, and can provide useful information to the administratrator regarding uptime or network problems between the central and the client.

Another advantage of the suck approach is that the delivery of data is more reliable; the server can keep asking the client for it until it gets it, and none of the data will be lost if the server is down.

Lastly, the suck architecture works nicely if you have an originate-only firewall. No exposure between the server and the outside world is necessary, and some administrators go the extra step and add in-kernel firewalling to the system log server to make it originate-only.



## Suck

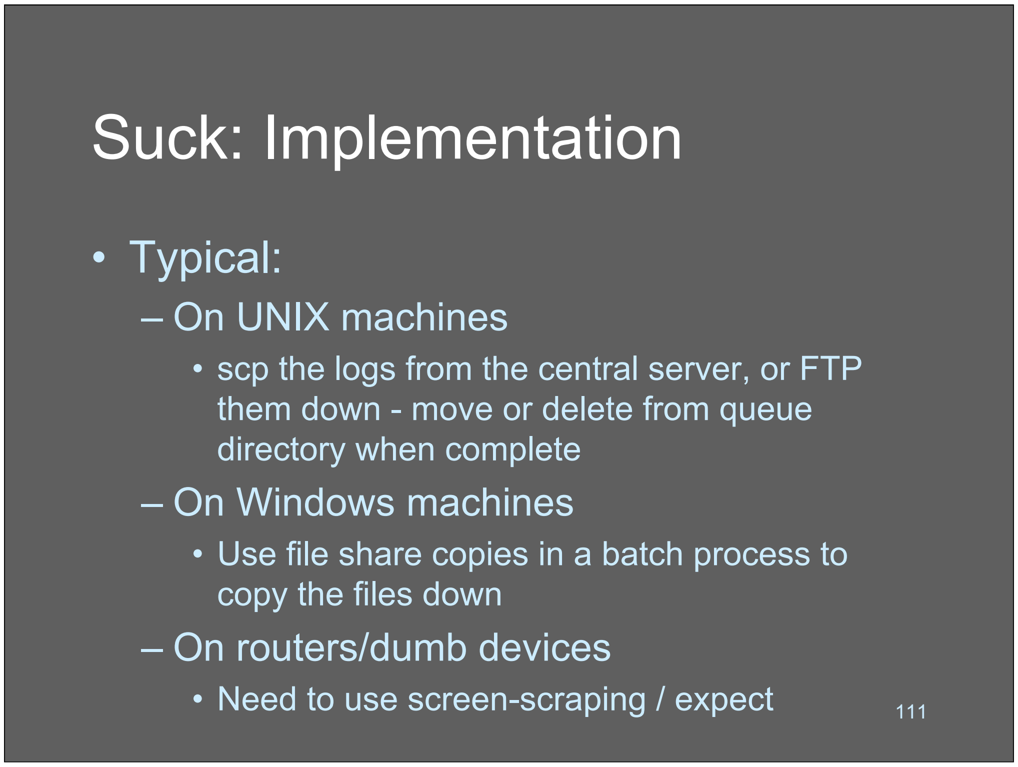
- Con:
  - Some devices are hard to fetch from!
  - If the client machine is compromised before the logs are copied down, they may be lost
  - The log server must know which machines to poll and requires per-machine configuration

110

The biggest problem with the suck architecture is that some devices are very difficult to batch-collect data from. Routers, for example, do not retain logs in memory; if you don't have them handed to you immediately, the are gone. In general suck aggregation architectures work best when you're dealing with relatively "smart" clients that have local hard disks.

From a security standpoint, the suck achitecture is more resistant to attack at the server end, but is more susceptible to hackers truncating logs at the client end. About the only way to address this problem is to ensure files are somehow safe on the clients (a hard problem!) or to generate warnings if the logs are unexpectedly short. Neither of those is particularly attractive!

Lastly, the suck architecture controls the relationship between the client and server more closely. In order to add a new client to a suck architecture, the server needs to have the client added to "the list" and the client needs to be configured to allow the server to collect its data. This may represent a large administrative load in some environments.



## Suck: Implementation

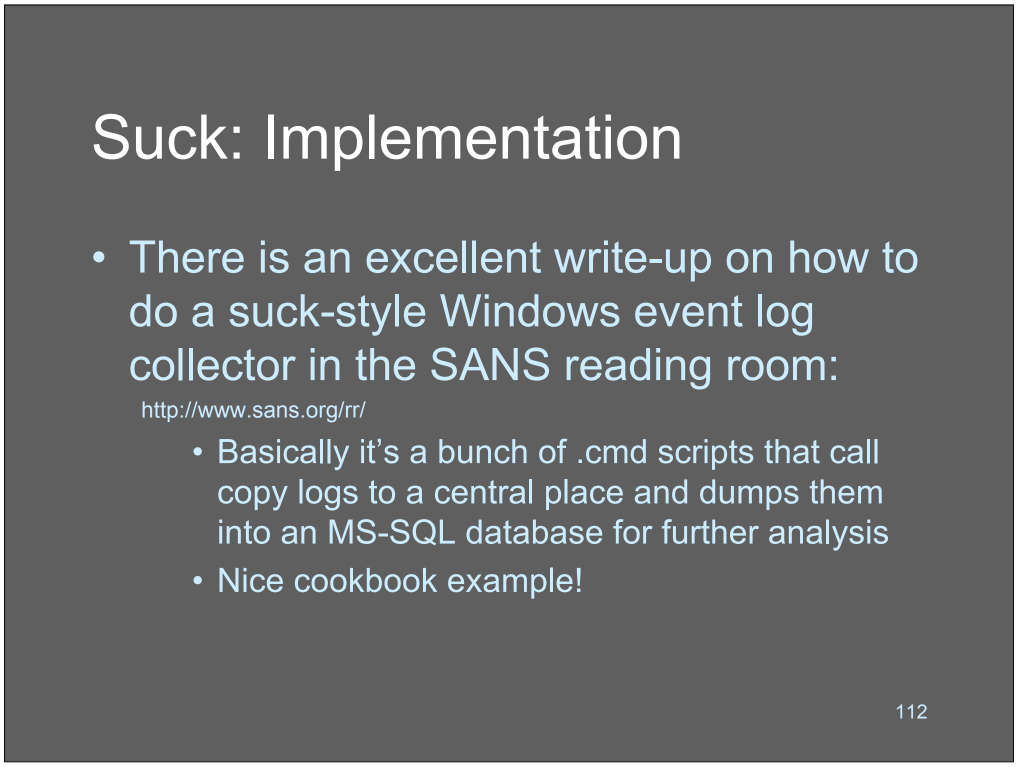
- Typical:
  - On UNIX machines
    - scp the logs from the central server, or FTP them down - move or delete from queue directory when complete
  - On Windows machines
    - Use file share copies in a batch process to copy the files down
  - On routers/dumb devices
    - Need to use screen-scraping / expect

111

The typical implementation of a suck architecture on UNIX systems is to use SSH or rsync over SSH to copy the files down and truncate them or rotate them centrally once they are collected. Probably the most sophisticated way to do this is to periodically rsync the log directory from the client. When the server wants to begin processing a chunk of data, it can move it out of the rsync'd area into a long-term holding area, which will cause it to get deleted from the client.

On Windows machines, suck architectures are usually implemented using windows sharing and copying using windows domain services. There are good write-ups on how to do this on the SANS reading room.

Suck architectures *really* fall down when dealing with dumb devices. Most sites trying to handle dumb devices will resort to SNMP querying to a local aggregator or screen-scraping.



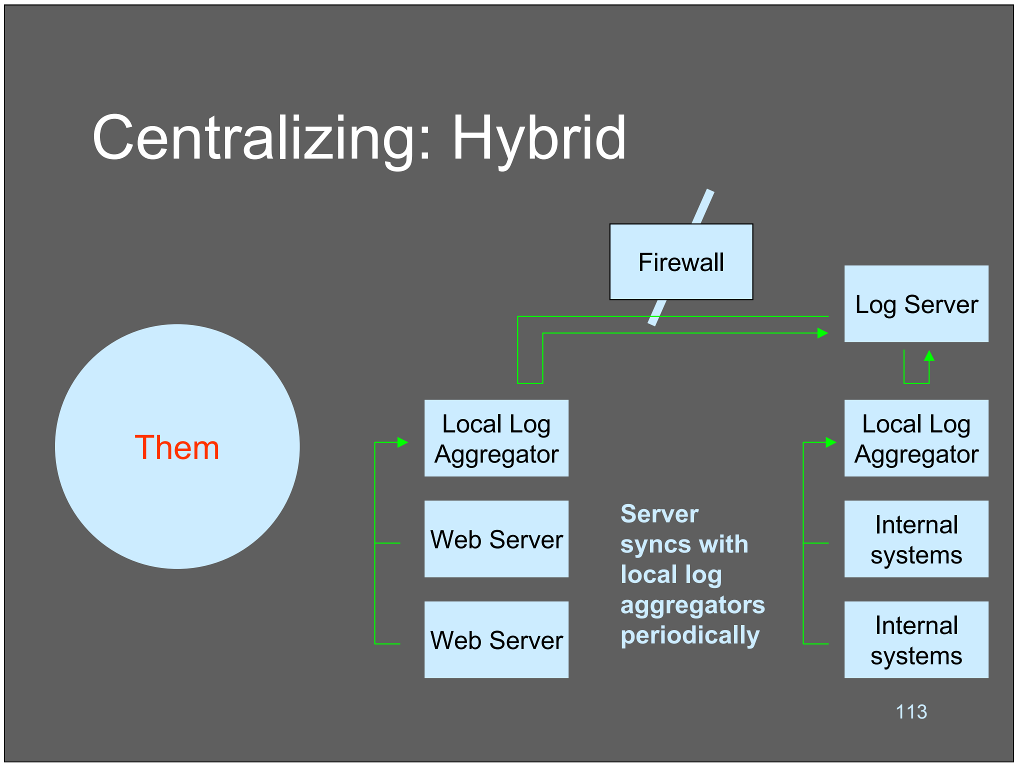
## Suck: Implementation

- There is an excellent write-up on how to do a suck-style Windows event log collector in the SANS reading room:
  http://www.sans.org/rr/
  - Basically it’s a bunch of .cmd scripts that call copy logs to a central place and dumps them into an MS-SQL database for further analysis
  - Nice cookbook example!

112

The SANS reading room has a great cookbook (actually 2) on how to build suck-style log architectures for Windows sysadmins.

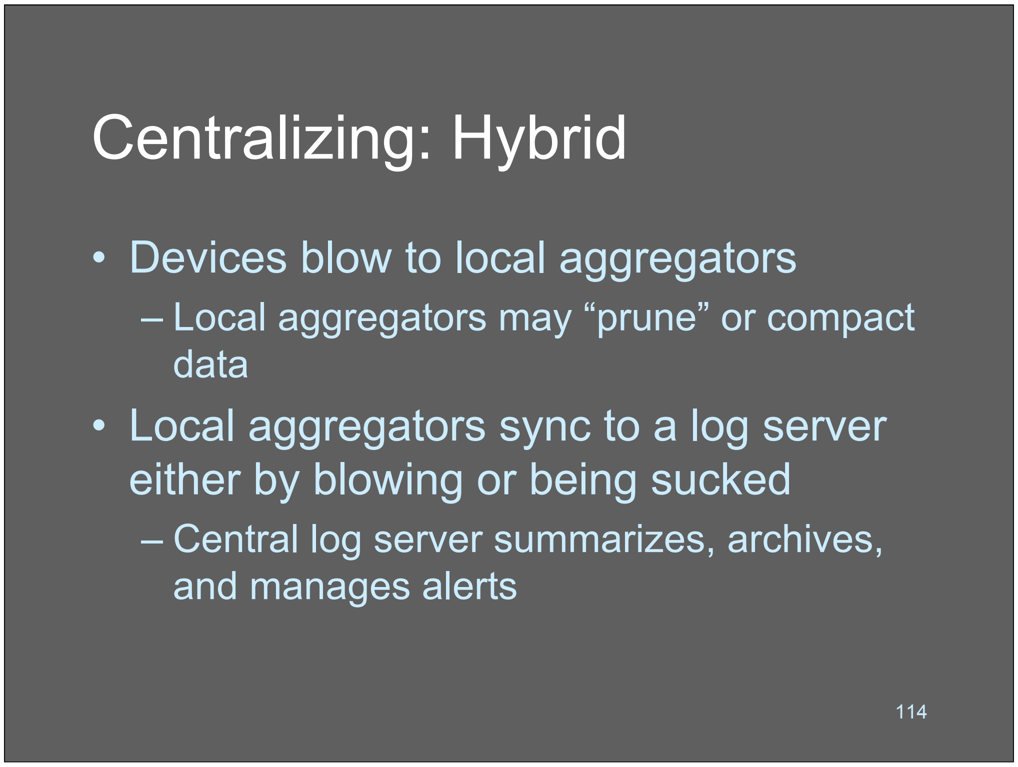
It’s a bit too convoluted to go into here, for space reasons, but it’s worth a read and includes all the command scripts that the author uses.



## Centralizing: Hybrid

The diagram above shows a typical hybrid log centralization architecture. There are local log aggregators that use whatever techniques make sense to collect information. This approach is particularly favored by sites that are also doing “do it yourself” IDS with something like Snort; the local aggregator might be a Snort sensor or other kind of monitoring probe.

Once the data is on the local aggregator a suck process pulls it in through the firewall to the central server.



## Centralizing: Hybrid

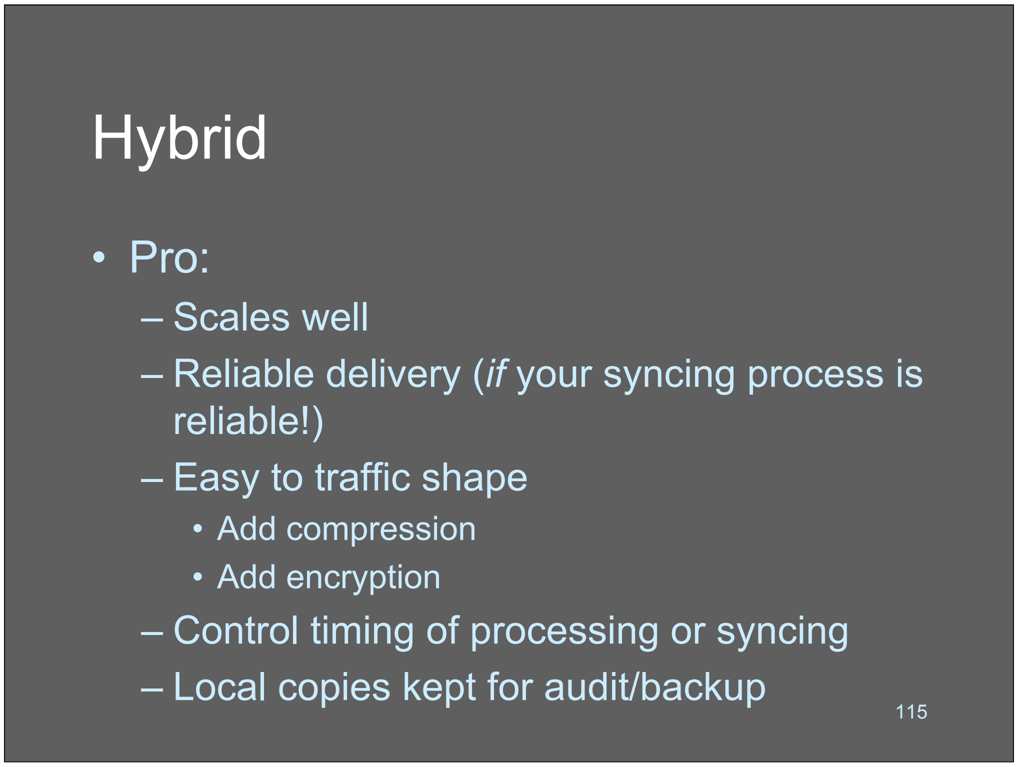
- Devices blow to local aggregators
  - Local aggregators may “prune” or compact data
- Local aggregators sync to a log server either by blowing or being sucked
  - Central log server summarizes, archives, and manages alerts

114

In the hybrid architecture, basically you are mixing blow and suck. Blow is performed locally, while suck is performed to get the data back to the central repository.

One feature of the hybrid approach is that frequently you may inject processing in the loop at the local aggregators. For example, the local aggregator might use syslog-ng to split off data into different buckets and only send up a count instead of the full data (I.e: 50Mb of SMTP messages sent in 3,928 messages) This works nicely since the local aggregators can retain the detailed information for as long as their hard disks can hold it, without needing to send it back to the central server.

Basically, the hybrid approach has the best of both worlds - at the cost of twice the work to build it!



## Hybrid

- Pro:
  - Scales well
  - Reliable delivery (*if* your syncing process is reliable!)
  - Easy to traffic shape
    - Add compression
    - Add encryption
  - Control timing of processing or syncing
  - Local copies kept for audit/backup

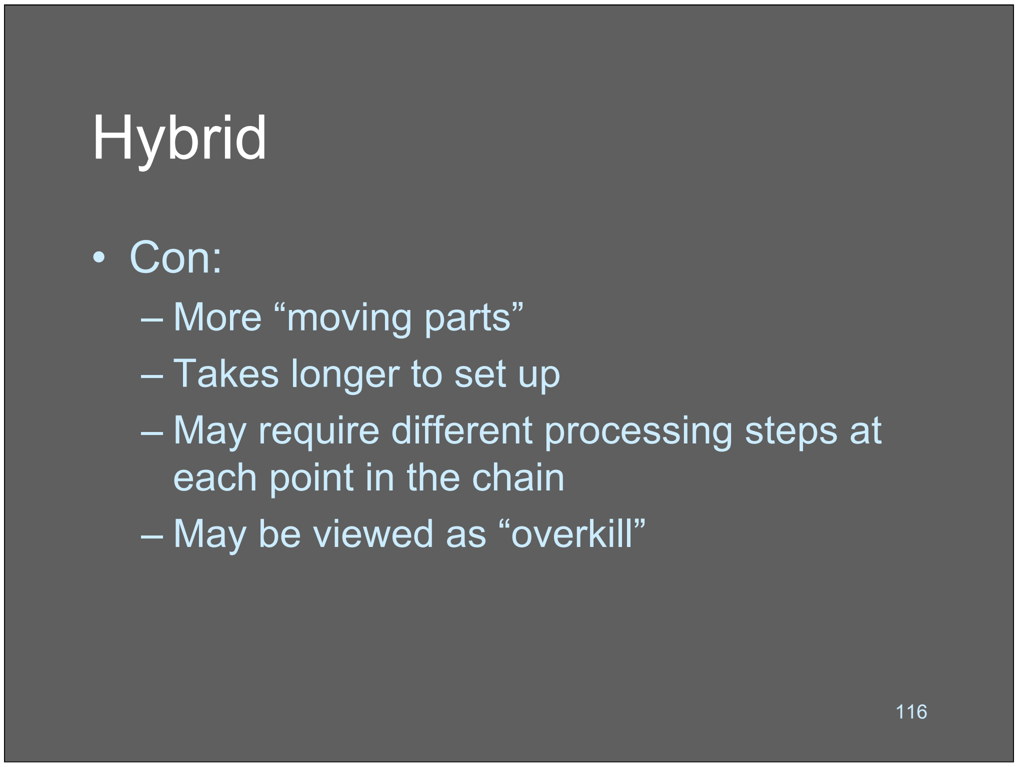
115

The advantages of the hybrid approach are all the advantages of the blow and suck architectures on their own. Scaleability can be terrific, it’s reliable, easy to secure, fault-tolerant, etc, etc.

Probably the best aspect of the hybrid log architecture, in my mind, is that you can also turn the local aggregator into a mini-sensor. It doesn’t need to run a full-blown IDS sensor like snort, but can run a DHCP tracker, TCP flow tracker, and web URL collector without slowing it down too much. That can be incredibly valuable.

Another powerful option to consider with the hybrid approach is running a whitelist/blacklist filter at the local aggregator and only passing up what isn’t knocked out of the stream by the blacklist. The whitelists/blacklists can be pushed out to the local aggregators using the same mechanism (e.g.: SSH or rsync) that is used to pull back the log files that are gathered.

The preceeding paragraphs are basically the design of 99% of the managed security services in existence.



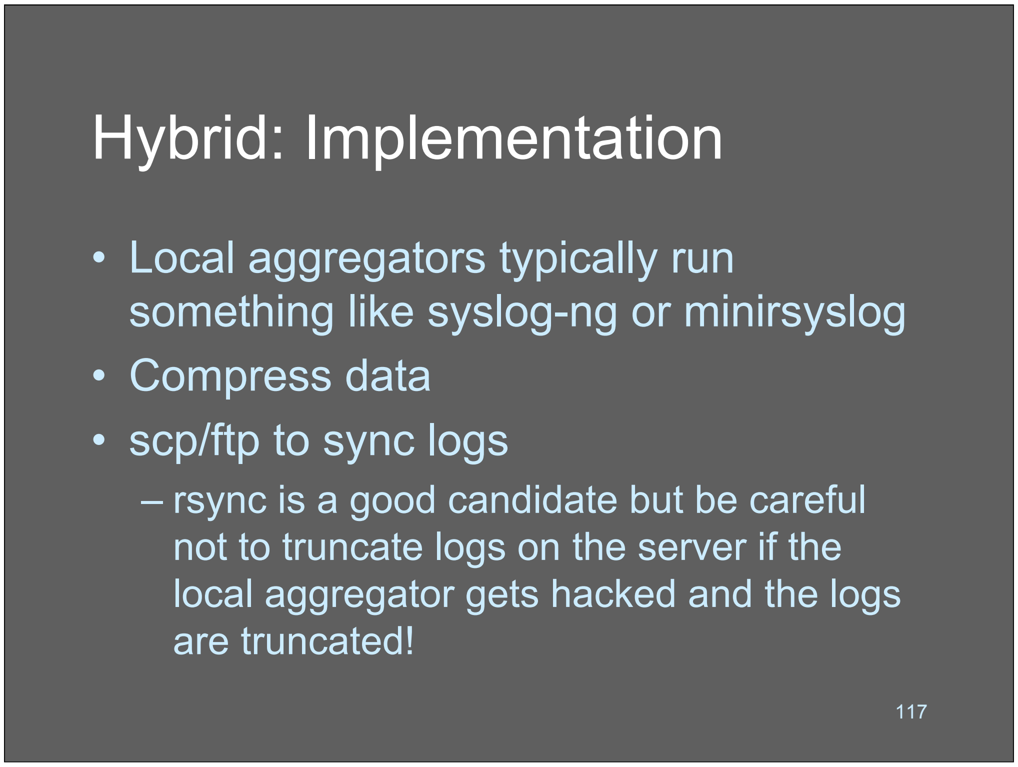
## Hybrid

- Con:
  - More “moving parts”
  - Takes longer to set up
  - May require different processing steps at each point in the chain
  - May be viewed as “overkill”

116

The only downsides of the hybrid architecture is that you have a lot more work to do to get it working.  You also will have a marginally higher hardware cost, though in practice the hybrid approach lets you get away with using cheaper hardware - you make up for using more of it in more places.

I’ve heard that it’s a harder sell to get senior management to “buy off” on a “full scale deployment” of something home-built rather than doing it as a full-blown commercial product. I never could understand pointy-hair boss think: let’s spend $400,000 to save $50,000 in staff time and $6,000 in barebone PCs bought on Ebay… Duh?



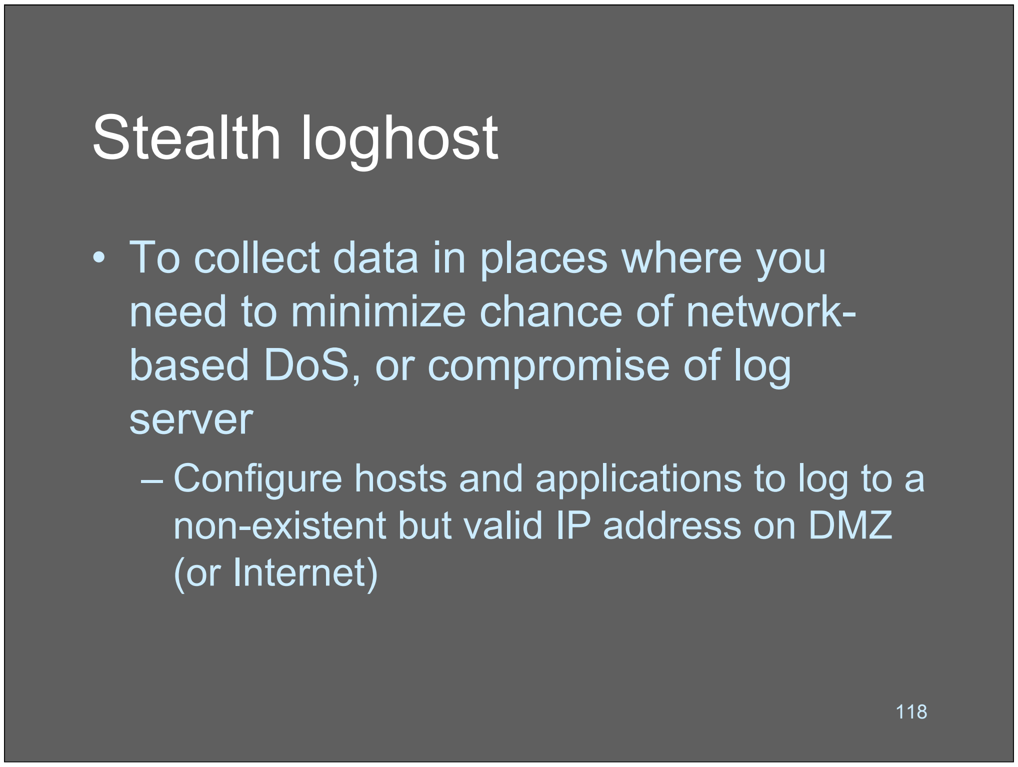
## Hybrid: Implementation

- Local aggregators typically run something like syslog-ng or minirsyslog
- Compress data
- scp/ftp to sync logs
  - rsync is a good candidate but be careful not to truncate logs on the server if the local aggregator gets hacked and the logs are truncated!

117

Typical tools for building a hybrid system are all the tools used to build blow and suck architectures. Basically, we’re talking about a superset of the two architectures.

One thing I cannot emphasize enough: make sure that your processing does not propagate file truncation. That is to say, if the hackers zap the log file on the aggregator, make certain that your process does not propagate the zapped file over top of the unzapped file back at the central aggregator!



## Stealth loghost

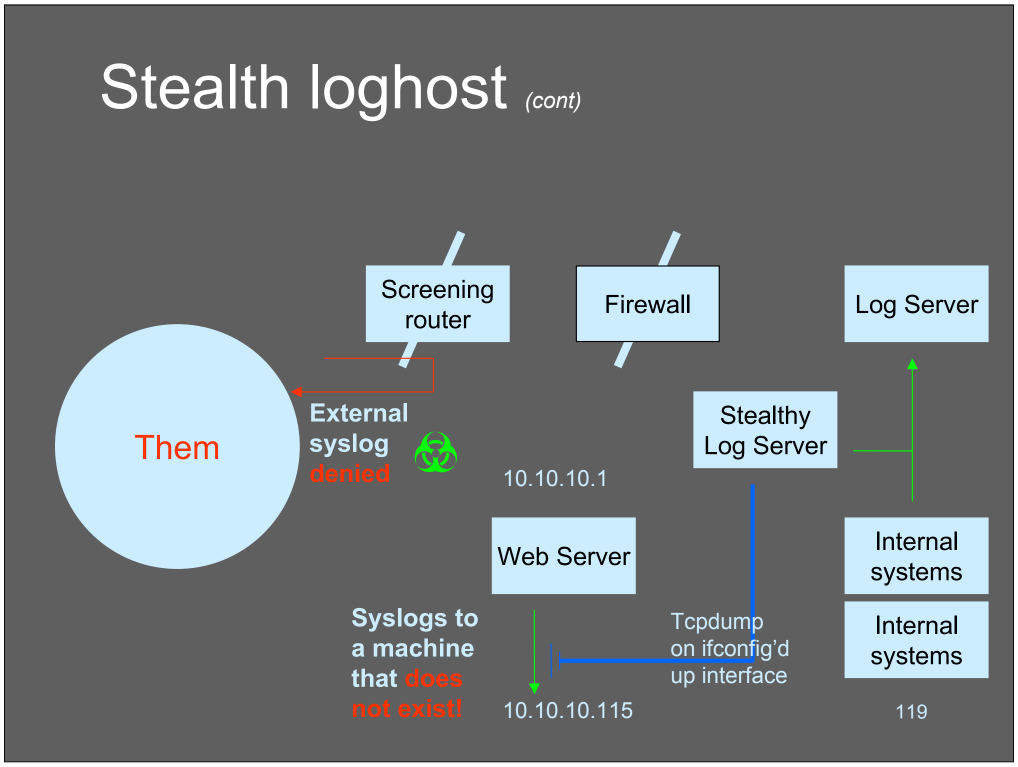
- To collect data in places where you need to minimize chance of network-based DoS, or compromise of log server
  - Configure hosts and applications to log to a non-existent but valid IP address on DMZ (or Internet)

118

One more cool idea from Lance Spitzner and the honeynet gang!

An introduction to the idea is on line at
`http://www.linuxjournal.com/modules.php?op=modload&name=NS-lj-issues/issue92&file=5476s2`

There are several ways to skin this particular cat. The honeynet gang simply vacuum up syslog packets using ethereal or snort and then view them that way.

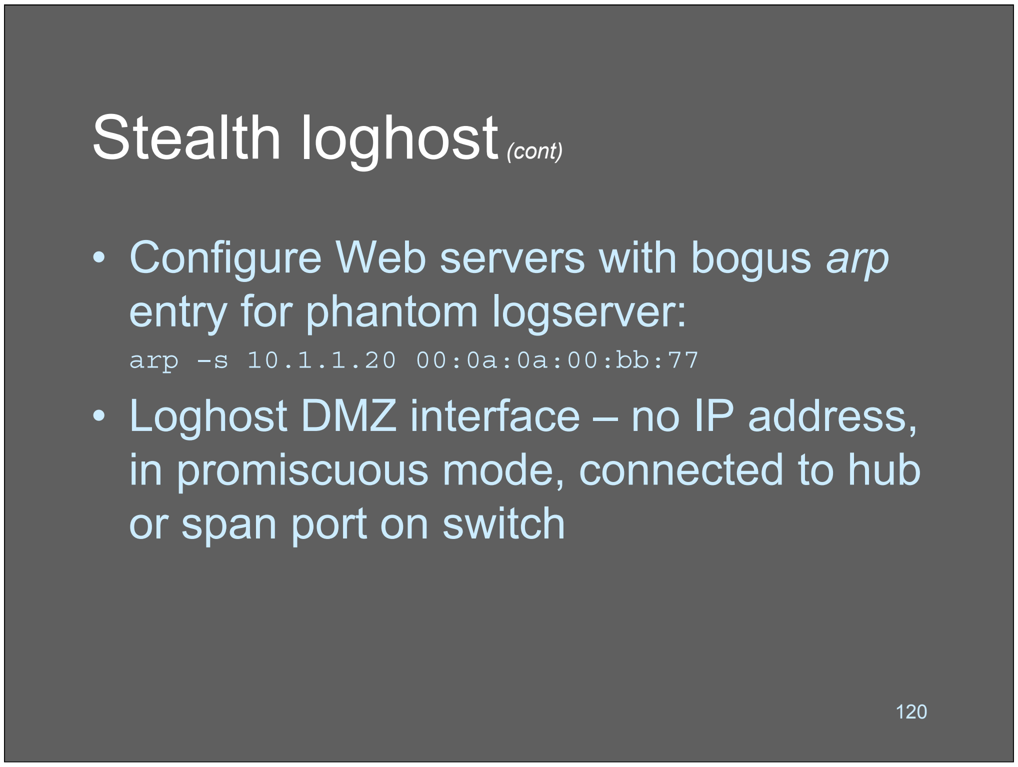


## Stealth loghost *(cont)*

The diagram above shows how a stealthy loghost works; the web server logs its data to 10.10.10.115 - a machine that conveniently does not exist. The stealthy log server is listening with ethereal/tcpdump/whatever to its external interface. The traffic is captured, interpreted as syslog data, and passed to the log server.

Note that some organizations' security policies prohibit this, since the stealth loghost is a "dual homed" system with an interface on each network. However you set this up make sure you have secured that exterior interface!!

The honeynet gang use this trick but actually have a "target" log server on the external network. The hope is that the hackers will see that there is a log server and attempt to go after it with a better class of tools. It's an interesting idea, anyway!



## Stealth loghost *(cont)*

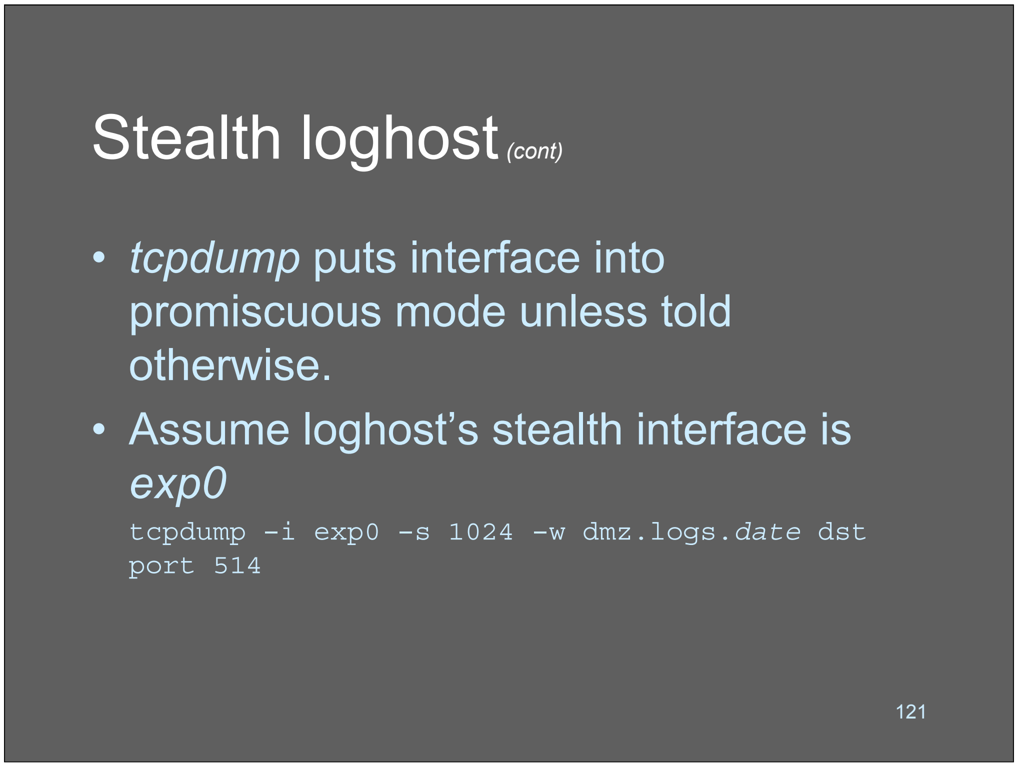
- Configure Web servers with bogus *arp* entry for phantom logserver:

```
arp -s 10.1.1.20 00:0a:0a:00:bb:77
```

- Loghost DMZ interface – no IP address, in promiscuous mode, connected to hub or span port on switch

120

Don’t forget to add your static *arp* entry to the system’s local start-up scripts, so it will continue to log successfully to the nonexistent machine after reboots.



## Stealth loghost *(cont)*

- *tcpdump* puts interface into promiscuous mode unless told otherwise.
- Assume loghost's stealth interface is *exp0*

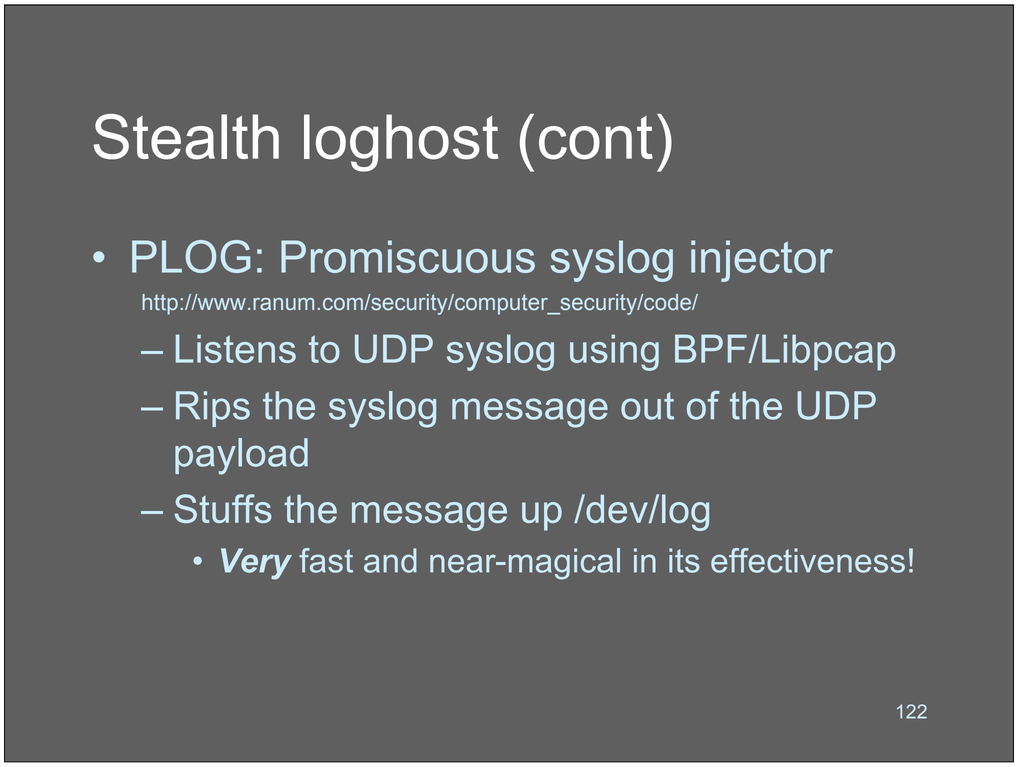
```
tcpdump -i exp0 -s 1024 -w dmz.logs.date dst
port 514
```

121

For loads of information on tcpdump, see:

`http://www.tcpdump.org`

In this example, we are capturing port 514 (syslog) traffic to a capture file (-w dmz.logs) for later analysis. To analyze the traffic, we'll need a tool that can strip syslog payloads out of the packets once they have been collected to the hard disk.



## Stealth loghost (cont)

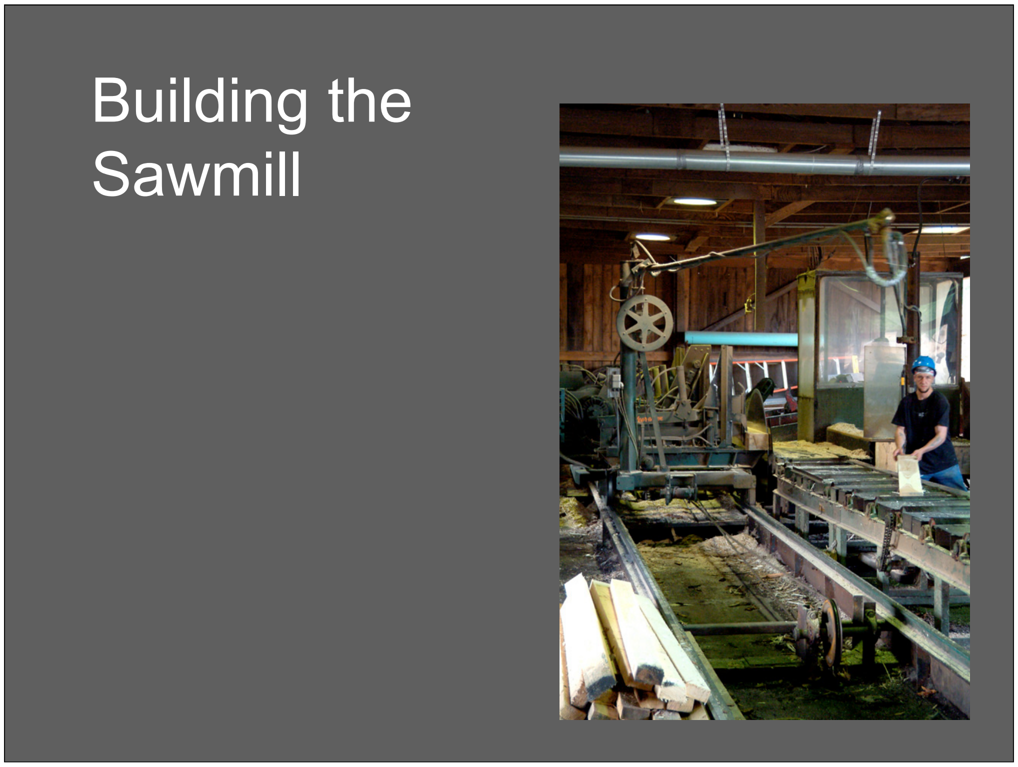
- PLOG: Promiscuous syslog injector
  http://www.ranum.com/security/computer_security/code/
  - Listens to UDP syslog using BPF/Libpcap
  - Rips the syslog message out of the UDP payload
  - Stuffs the message up /dev/log
    - ***Very*** fast and near-magical in its effectiveness!

122

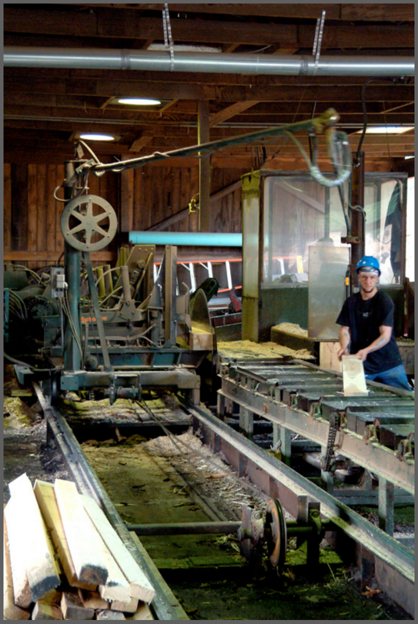
If you want a stealth loghost that has all the properties of a normal syslog aggregator, you can try using PLOG. PLOG is a promiscuous syslog listener that decodes the UDP payload straight out of the syslog packets and injects them into /dev/log as if they had originated locally.

PLOG is extremely fast and a little counter-intuitive in how it works. The log messages simply appear in syslogd's data as they crossed the wire. If you're using syslog-ng you can apply all the usual filtering, etc. This is an extremely powerful tool!

One thing to be careful of - if you are using PLOG you cannot be forwarding syslog data back off your machine using UDP or PLOG will collect it again, and you'll have an input loop that fills your log server's hard disk before you even realize what is going on! You'll notice it  pretty quickly, believe me.



## Building the Sawmill

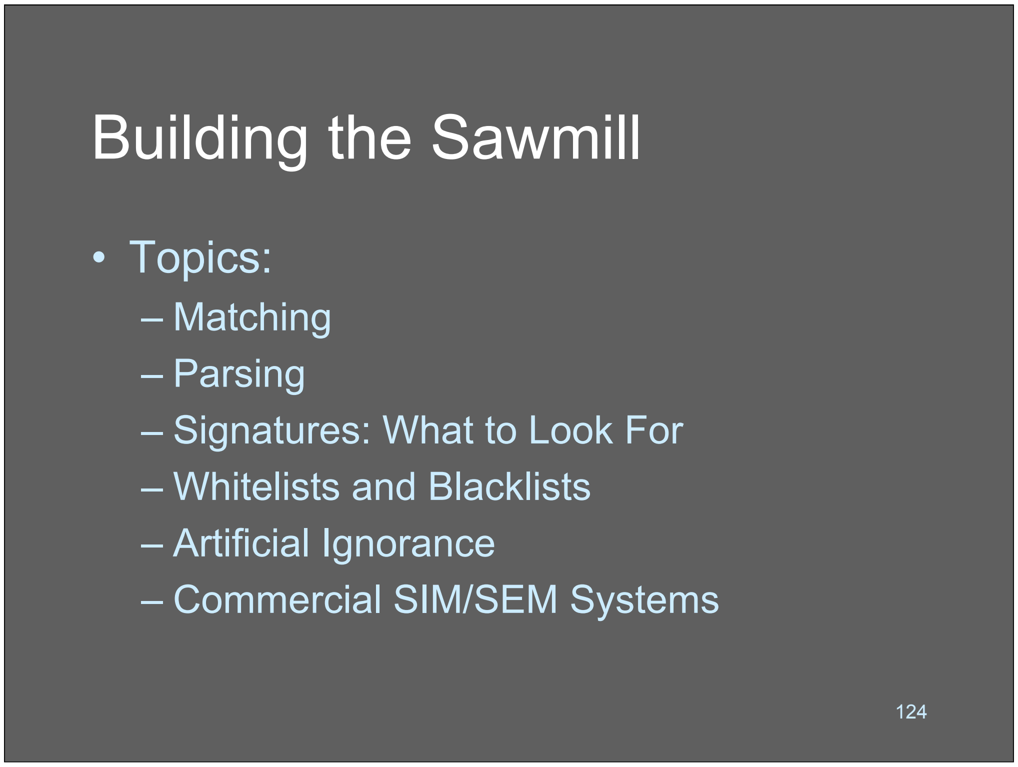


This is the saw at Greenwood's. That blade moves a few thousand RPMs, but if you think about the angular velocity at the edge, it's kind of frightening. While the saw is running the teeth make a hissing sound that definitely helps keep you focussed on staying away from it.

The tracks support an auto-feeder that collects a log off the input chute (to the back on the right) - the feeder grabs the log and then slides forward on the rails to make the cut. It can rotate the logs, so you can very easily get a plank or square-cut piece of wood without having to do anything other than pull the newly cut boards out of the way.

The little plexiglass shield reminds the worker not to get too close to the blade. His job is to take the pieces off the conveyor and throw the scrap onto a scrap belt and the good boards onto a conveyor for stacking.

When you watch the transition area where the blade hits the wood, it's as if the wood is being instantly teleported to an alternate dimension.

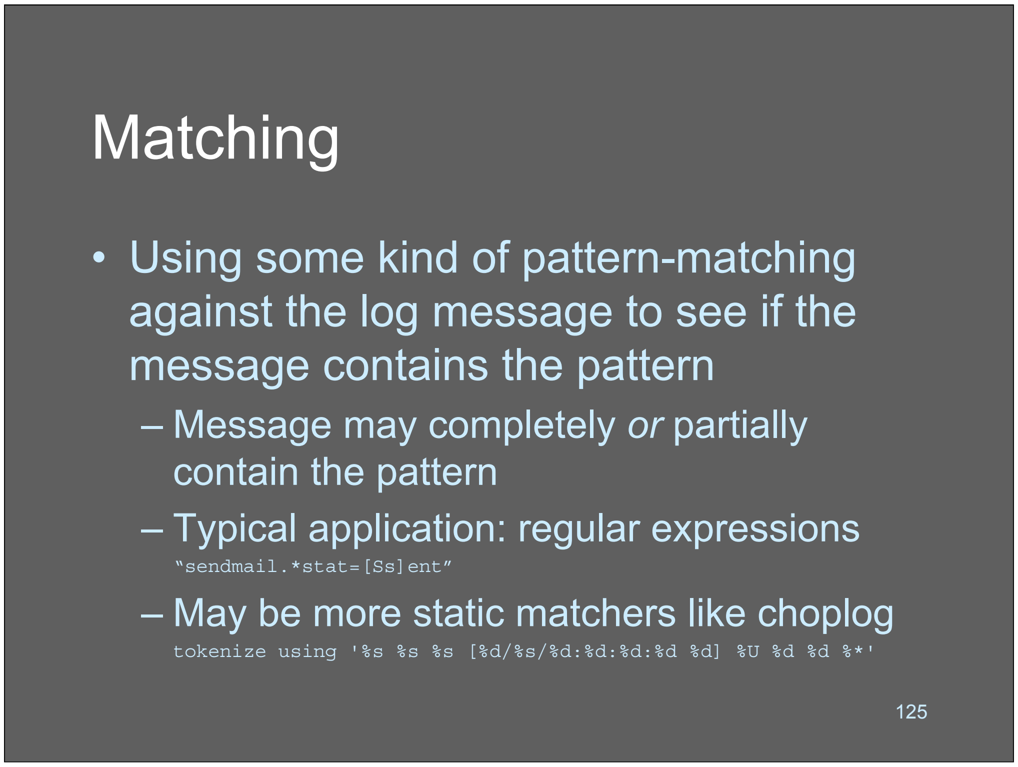


## Building the Sawmill

- Topics:
  - Matching
  - Parsing
  - Signatures: What to Look For
  - Whitelists and Blacklists
  - Artificial Ignorance
  - Commercial SIM/SEM Systems

124

These are the topics we will cover next!! Once we have the data in one place - what do we DO WITH IT?



## Matching

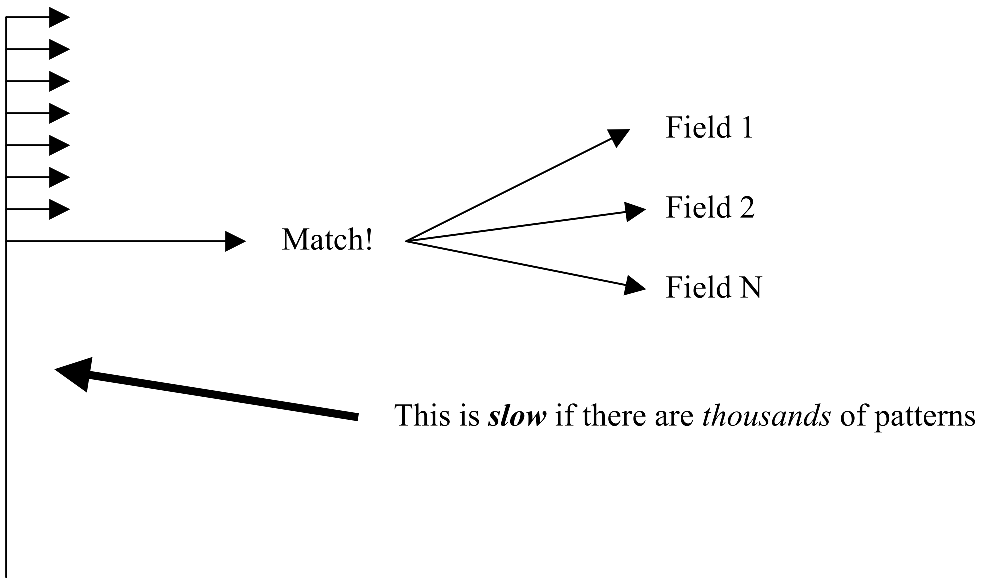
- Using some kind of pattern-matching against the log message to see if the message contains the pattern
  - Message may completely *or* partially contain the pattern
  - Typical application: regular expressions
    `"sendmail.*stat=[Ss]ent"`
  - May be more static matchers like choplog
    `tokenize using '%s %s %s [%d/%s/%d:%d:%d:%d %d] %U %d %d %*'`

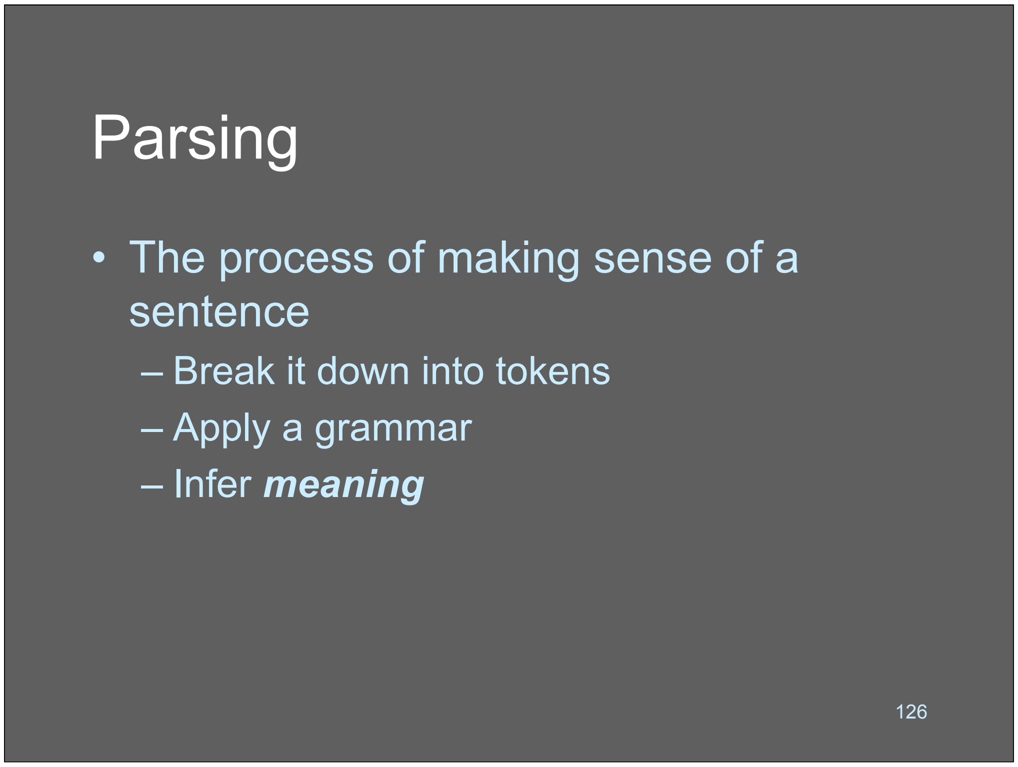
125

Everything to do with processing logs involves the problem of parsing data.

Most people, when they start with syslogs, begin with *matching*. The basic matching approach is to use a Perl regular expression (or something like it) to decide if a syslog message contains a substring. Once the desired message has been identified, then the message is broken apart into useful chunks based on what the user wants.

This approach is inherently slow, because to match, you need to apply all (or many) of the rules, before you decide on the sub-parse fields you want to pick out.





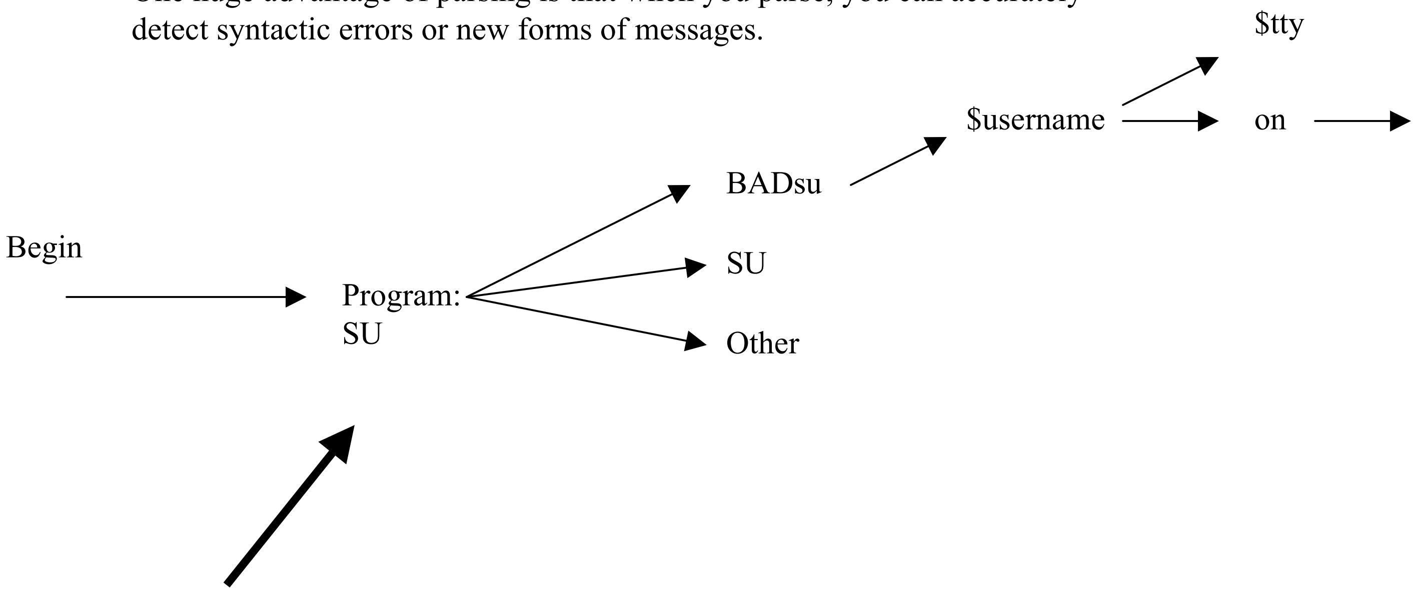
## Parsing

- The process of making sense of a sentence
  - Break it down into tokens
  - Apply a grammar
  - Infer ***meaning***

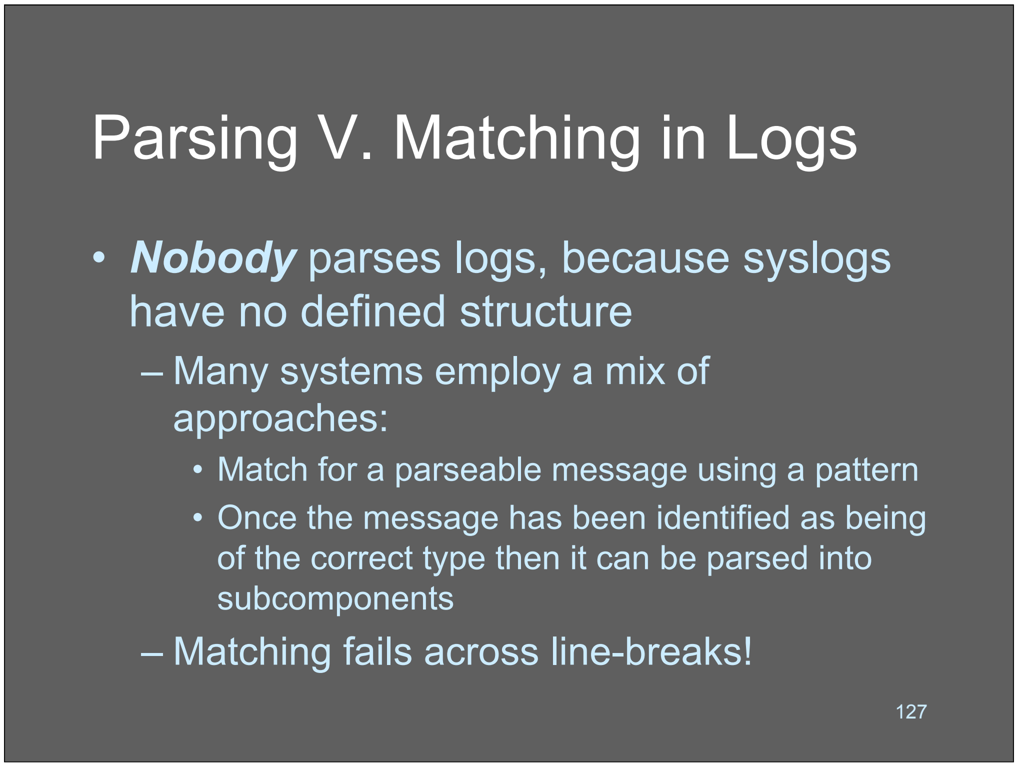
126

Parsing implies building a full-blown parse tree. Parse trees are inherently very efficient because you always limit the number of branches that you need to go down. Building a parse tree for logging is a terribly difficult thing because the structure and contents of log messages are so mutable. But it's important to understand how parsing works, because if you're ever processing truly huge amounts of data, it'll come into play eventually.

One huge advantage of parsing is that when you parse, you can accurately detect syntactic errors or new forms of messages.



Very complex data structures! You need to encompass every possible derivation of every possible message! OW!



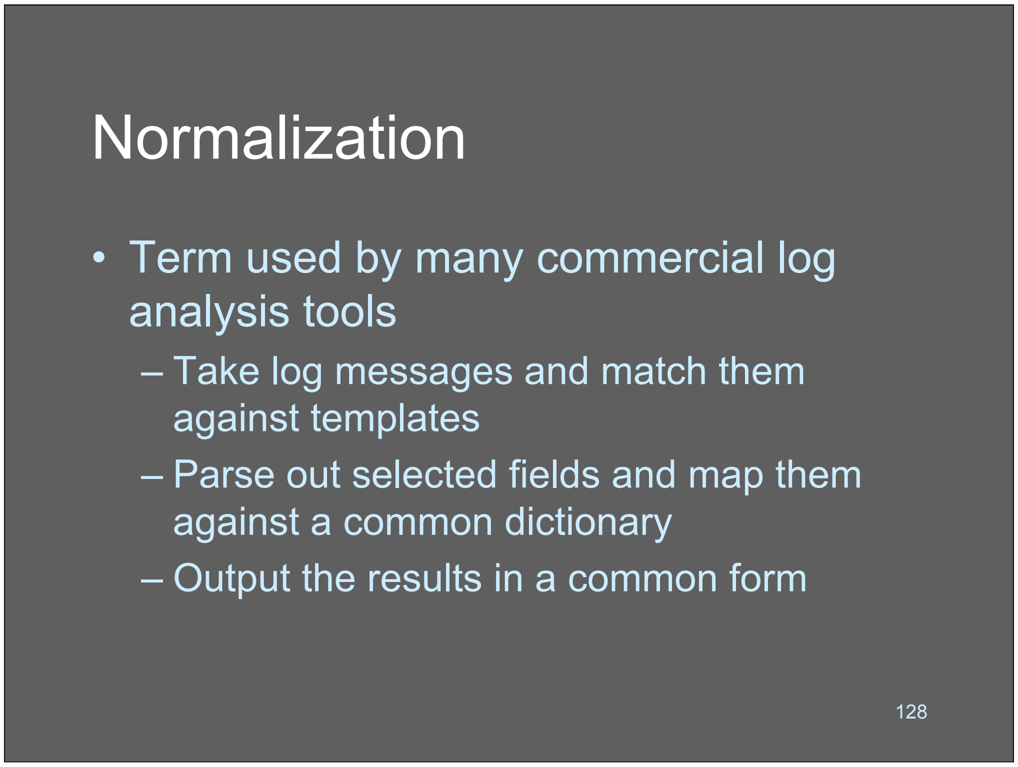
## Parsing V. Matching in Logs

- ***Nobody*** parses logs, because syslogs have no defined structure
  - Many systems employ a mix of approaches:
    - Match for a parseable message using a pattern
    - Once the message has been identified as being of the correct type then it can be parsed into subcomponents
  - Matching fails across line-breaks!

127

At present, nobody that I know of builds a complete log-parser. Most systems today match for a parseable message and then drop it into a message-specific tokenization and parsing process.

This works fine in most cases except it becomes problematic if you're dealing with messages that go across multiple lines. Multiple-line log parsing, at this time, is something of a "we don't go there" kind of process.



## Normalization

- Term used by many commercial log analysis tools
  - Take log messages and match them against templates
  - Parse out selected fields and map them against a common dictionary
  - Output the results in a common form

128

Normalization is the fancy term for matching tokens collected in one log format into another.

The example we had early on:

```
Jan 17 14:39:01 sol8isr sendmail[478]: [ID 702911
mail.info] /etc/mail/aliases: 3 aliases, longest 10
bytes, 52 bytes total
```

```
Aug 14 14:37:46 devel sendmail[514]: /etc/aliases: 14
aliases, longest 10 bytes, 152 bytes total
```

```
localhost sendmail[494]: alias database /etc/aliases
rebuilt by root
```

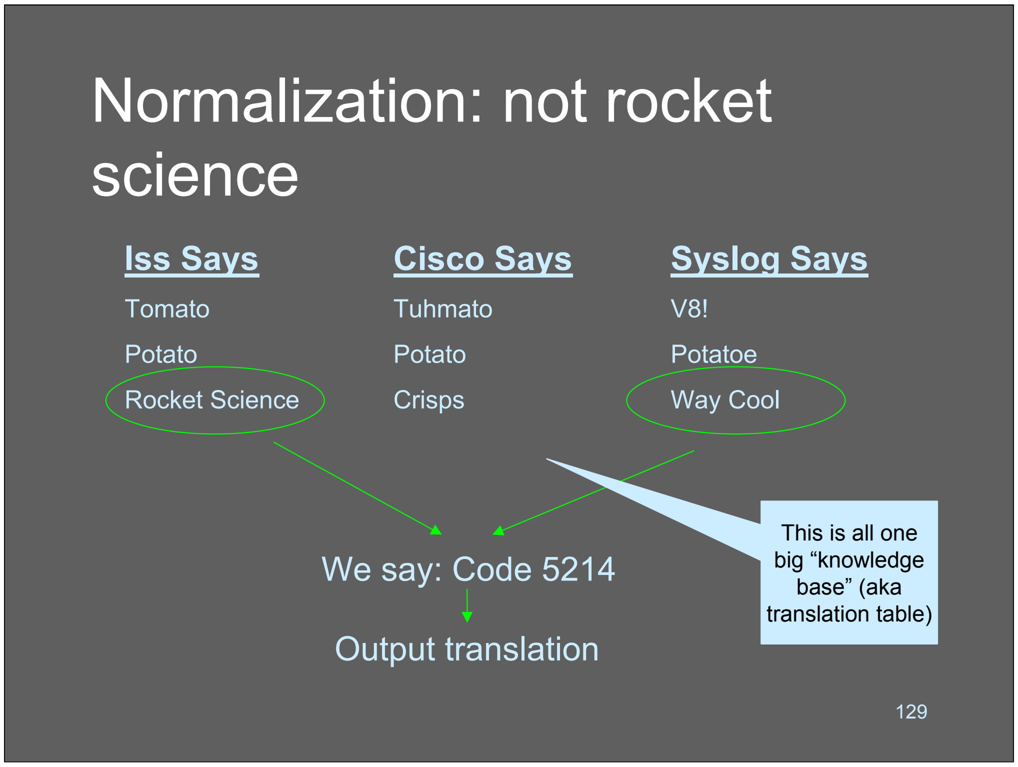
We have 3 completely different formats. We'd normalize them as:

```
Program=mail
```

```
Operation=new aliases
```

```
User=root
```

Note that unless we get very complex data structures, we'll either throw information away or make our new message contain a load of NULL fields. Thus, one important thing to realize about normalization is that you're losing information or gaining NULLs.



## Normalization: not rocket science

| Iss Says | Cisco Says | Syslog Says |
|---|---|---|
| Tomato | Tuhmato | V8! |
| Potato | Potato | Potatoe |
| Rocket Science | Crisps | Way Cool |

We say: Code 5214

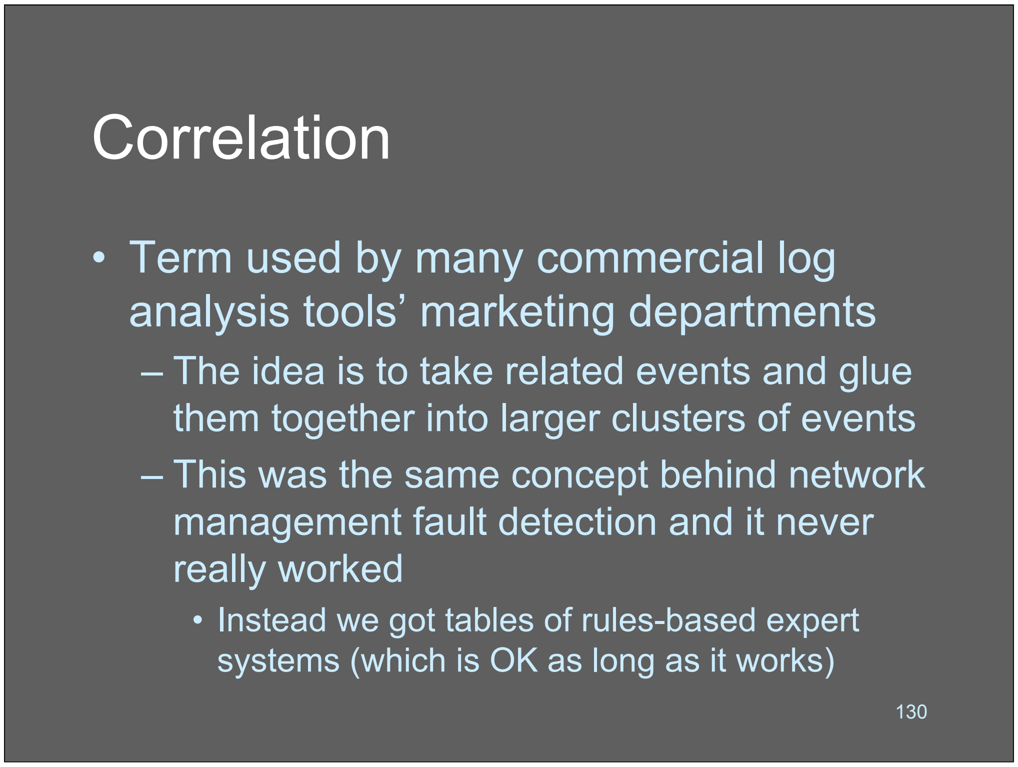
Output translation

This is all one big "knowledge base" (aka translation table)

129

To normalize, you are basically building a big knowledge-base of matching rules that tie to parsing/extraction rules, which map to a common "knowledge representation" that can be output into another form.

This is where XML *might* come in handy. It's a good (ok, mediocre) intermediate representation for structured data. The value proposition of all this stuff is in the structure of the knowledge-base, though, and once you have your data parsed and normalized you're not going to want to apply another layer of normalization or translation. Put another way: if XML is the solution, I'm not sure what the problem is.



## Correlation

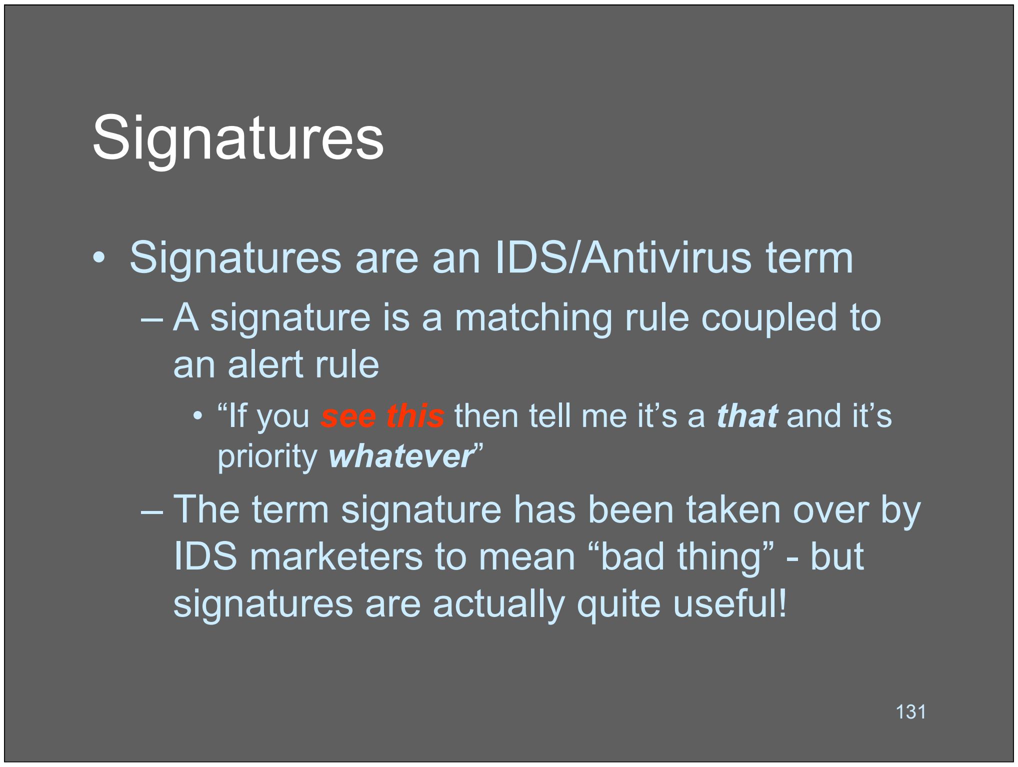
- Term used by many commercial log analysis tools' marketing departments
  - The idea is to take related events and glue them together into larger clusters of events
  - This was the same concept behind network management fault detection and it never really worked
    - Instead we got tables of rules-based expert systems (which is OK as long as it works)

130

Correlation is a "magic word" in security and network management. Kind of like "drill down" - it's something you really want, but nobody actually knows what it is!

To be able to do automated correlation (what most people seem to think of when they say "correlation") you'd need to be able to parse all the messages into normalized tokens, and then begin to match fields looking for common linkages. Before you get excited about the idea, remember that this is what network management tools were supposedly going to do. So far they really don't work very well. Finding relationships between things (the root of the word "co-relation") may be a creative process and may not be something that can be automated well.

One vendor who shall go in the "hall of shame" used to claim they did "screen correlation" What's that? It meant that the events were *on the display at the same time*. Wow! That's pretty fancy, huh? And if you can read 5,000 events/second maybe you can pick out the relationships between them on the fly with your Mark IV Human Eyeball.



## Signatures

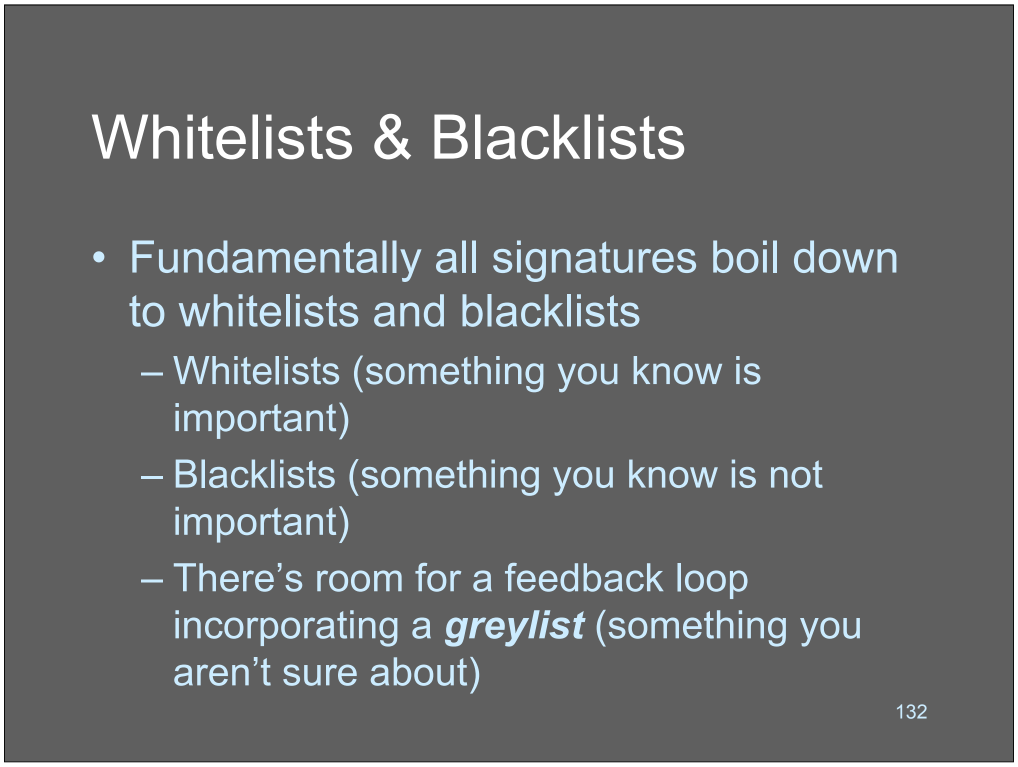
- Signatures are an IDS/Antivirus term
  - A signature is a matching rule coupled to an alert rule
    - "If you ***see this*** then tell me it's a ***that*** and it's priority ***whatever***"
  - The term signature has been taken over by IDS marketers to mean "bad thing" - but signatures are actually quite useful!

131

"Signature" has gotten a bad rap thanks to marketing people deciding to push their (largely signature-based) IDS as "signatureless." In truth, signatures aren't bad at all. Basically, a signature is a:

***A matching rule coupled to an alert rule***

This is valuable because it provides a diagnosis of what was discovered. Fundamentally, diagnosing something is a creative process and thus requires an intelligence. Without a signature matching rule that leads to a diagnosis/alert rule then all you can do is identify "this looks weird - go figure it out" That's also incredibly valuable - we want both!



## Whitelists & Blacklists

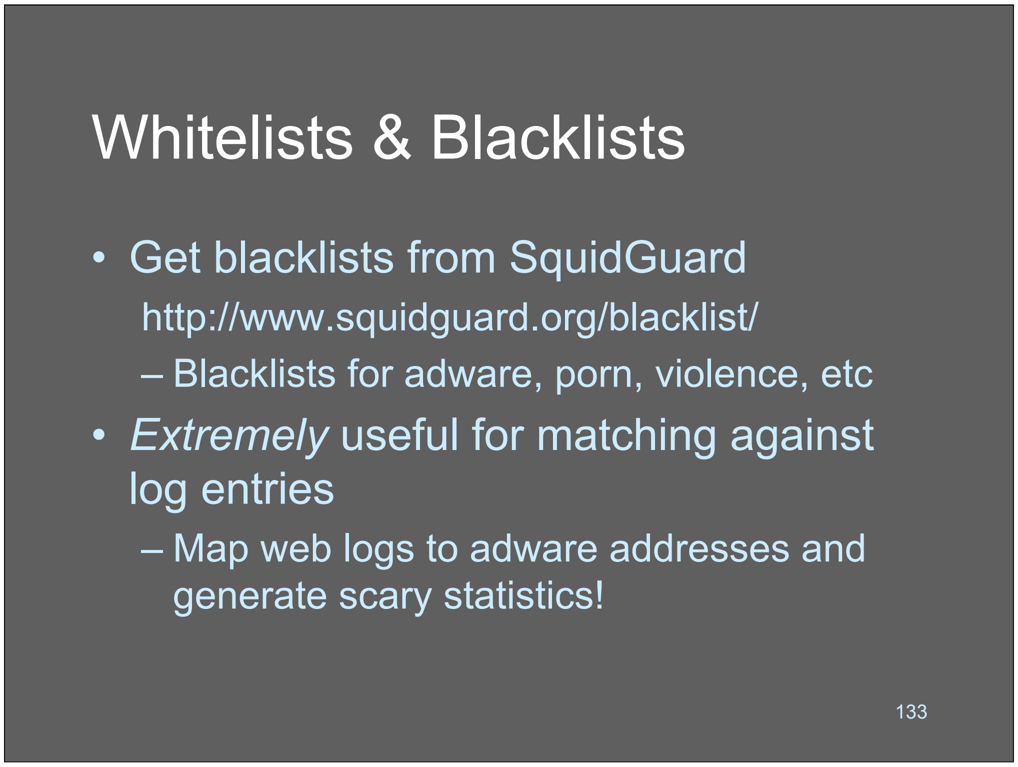
- Fundamentally all signatures boil down to whitelists and blacklists
  - Whitelists (something you know is important)
  - Blacklists (something you know is not important)
  - There's room for a feedback loop incorporating a ***greylist*** (something you aren't sure about)

132

Whitelists and blacklists (and, by extension greylists) are a very powerful technique for sorting logs.

A "whitelist" is something you know is important  (e.g.: you care about) and a "blacklist" is a list of stuff you know you don't care about. A "greylist" is a list of stuff you're not sure about. So basically what you do is mass-match messages against the whitelist and the blacklist and put everything else in the greylist.

Implementing whitelist/greylist processing is pretty simple!



## Whitelists & Blacklists

- Get blacklists from SquidGuard
  http://www.squidguard.org/blacklist/
  - Blacklists for adware, porn, violence, etc
- *Extremely* useful for matching against log entries
  - Map web logs to adware addresses and generate scary statistics!

133

One terrifically scary thing you can do with whitelist and blacklists is to match existing lists against your logs. Best of all, there are people who already provide these lists as a free service!!

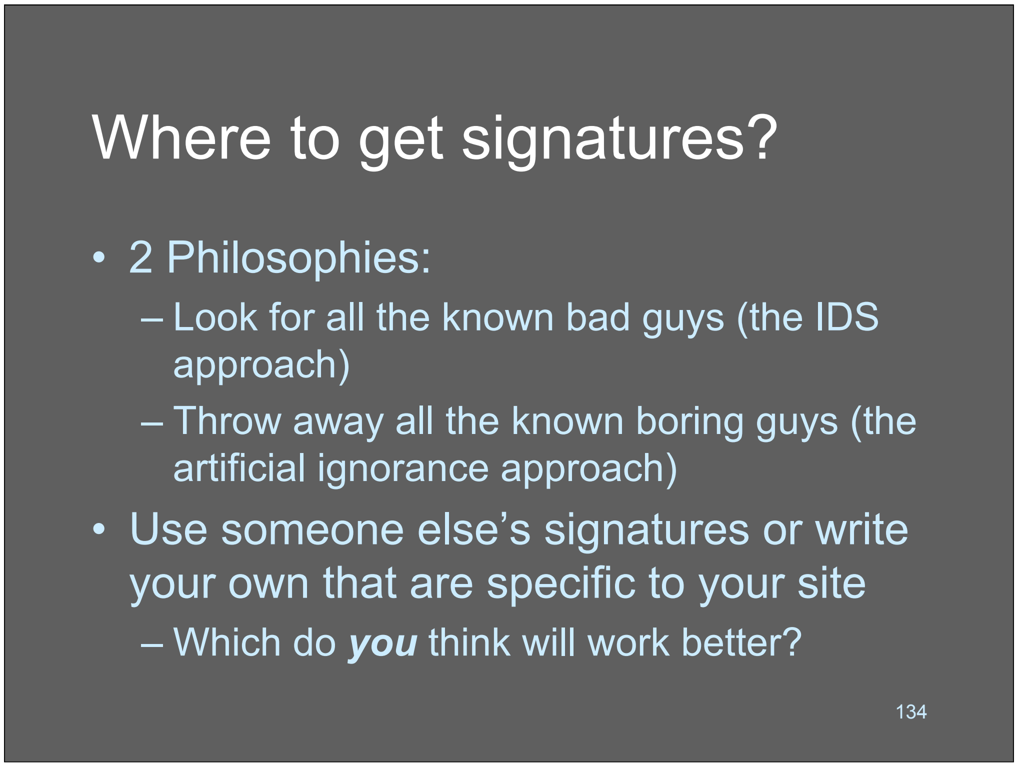
My favorite source for black/white lists is the SquidGuard distribution. They include 300,000 or so IP addresses and names for sites that provide porn, or that are used for adware.

Adware is one of those topics you can use to utterly terrorize your CTO/CIO if you are in a fun-loving mood and work for the DOD, or a hospital. Map your firewall "permit" logs against the list of adware sites and generate a rough statistic of how many machines on your network are infected with Adware!

Whitelists are also useful for creating lists of (approximate) %ages of web surfing that is to porn sites, etc, etc. You can make yourself *amazingly* unpopular with this stuff!!

Approximate  UNIX commands to try:

```
sort whitelist > list.sorted
# system log strip just the destination address
sort destinations > dest.sorted
join dest.sorted list.sorted | uniq -c | sort -r -n
```



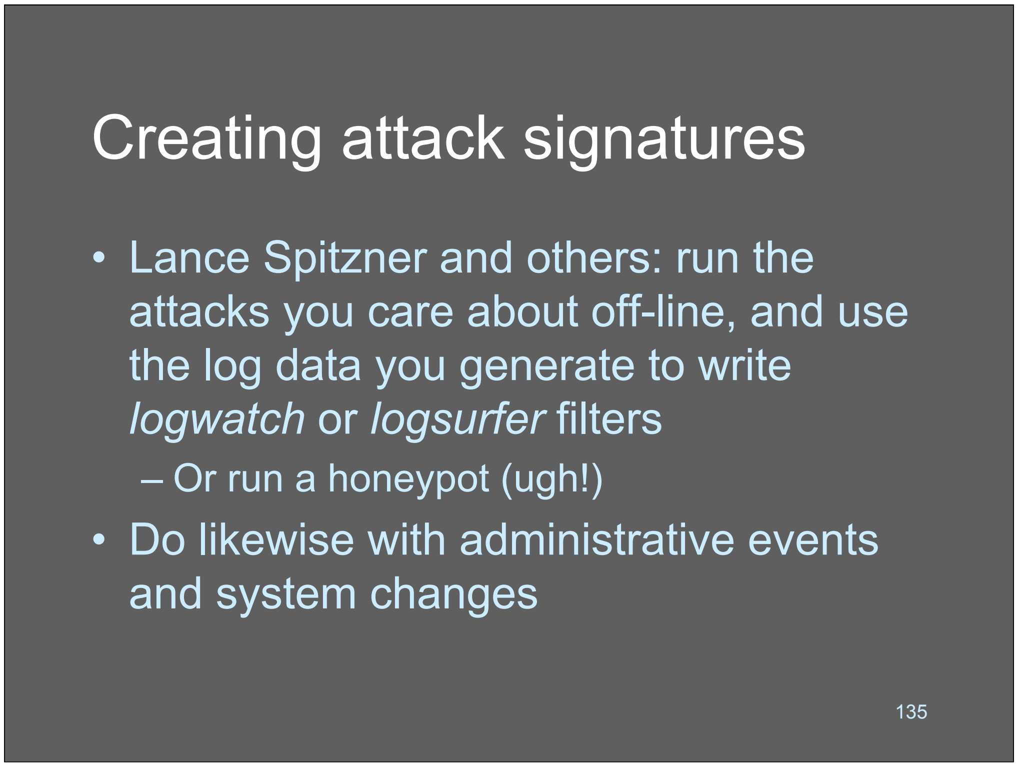
## Where to get signatures?

- 2 Philosophies:
  - Look for all the known bad guys (the IDS approach)
  - Throw away all the known boring guys (the artificial ignorance approach)
- Use someone else's signatures or write your own that are specific to your site
  - Which do ***you*** think will work better?

134

As you've probably figured out by now, I am a big fan of the artificial ignorance approach - know what's not important - over the "look for bad guys" approach. Mostly because it's an expensive process to identify all the bad guys.

*You* need to decide which approach is more likely to work on *your* network. Or do some experimentation. But don't just attack this problem without giving it some thought!


## Creating attack signatures

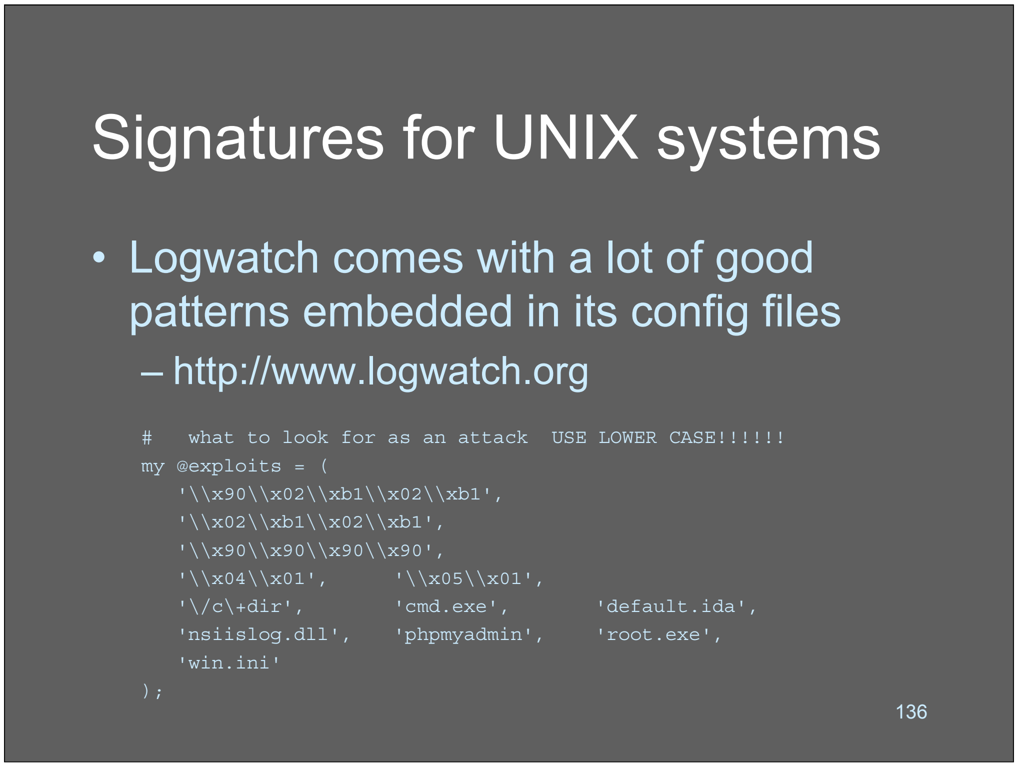
- Lance Spitzner and others: run the attacks you care about off-line, and use the log data you generate to write *logwatch* or *logsurfer* filters
  - Or run a honeypot (ugh!)
- Do likewise with administrative events and system changes

135

Lance’s essay *Know Your Enemy: II Tracking the Blackhat’s Moves* (now maintained by the Honeynet Project) describes the actions taken by an intruder, and the sort of log data they’d generate. It’s a good starting point for understanding how to configure your log monitoring tools.

http://project.honeynet.org/papers/enemy2/

So, one option might be to generate your own signatures by leaving a system exposed to the Internet. After a while, (assuming it survives for a while) you might find useful patterns in logs. As you might expect, different systems react to the worm-of-the-month differently. You might find a good signature, or you might find that your system is completely oblivious to the attack.



## Signatures for UNIX systems

- Logwatch comes with a lot of good patterns embedded in its config files
  - http://www.logwatch.org

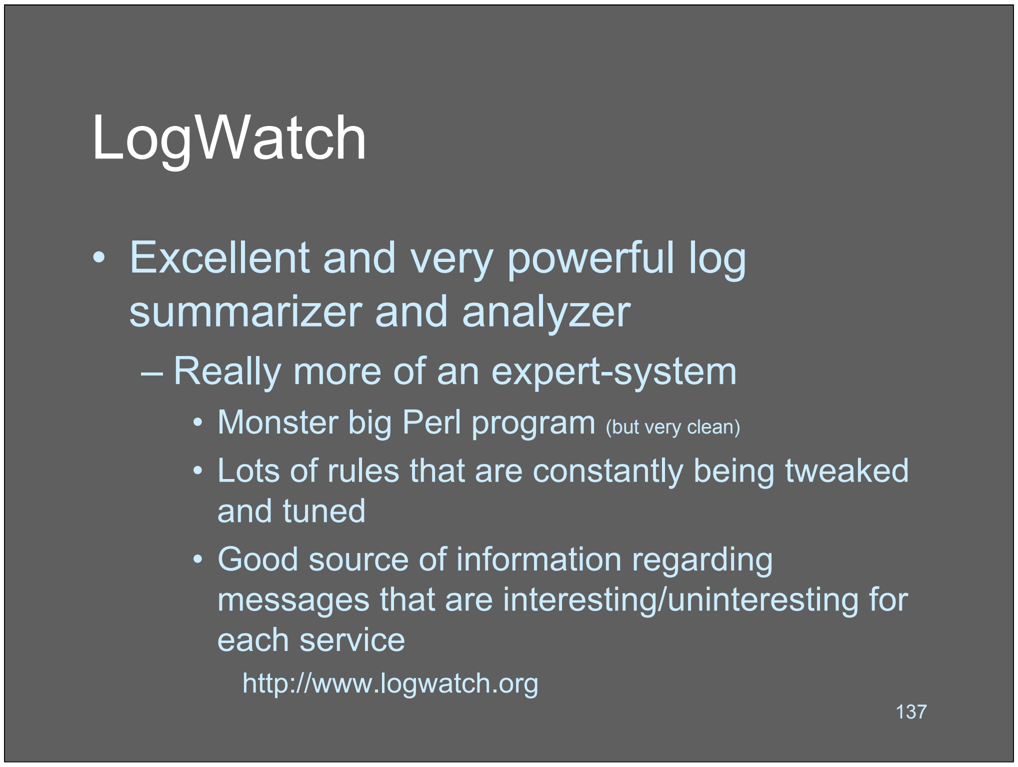
```
#   what to look for as an attack  USE LOWER CASE!!!!!!
my @exploits = (
   '\\x90\\x02\\xb1\\x02\\xb1',
   '\\x02\\xb1\\x02\\xb1',
   '\\x90\\x90\\x90\\x90',
   '\\x04\\x01',      '\\x05\\x01',
   '\/c\+dir',        'cmd.exe',        'default.ida',
   'nsiislog.dll',    'phpmyadmin',     'root.exe',
   'win.ini'
);
```

136

If you want a terrific set of starter signatures, take a look at Logwatch!

It's nicely modularized perl code, and each module includes lists of regular expressions to look for in each different program/protocol. You don't need to be a perl guru to make sense of these; they're very useful.

In the example above, we see logwatch defining a group of strings that might be good indicators of a successful (or near-successful) exploit.



## LogWatch

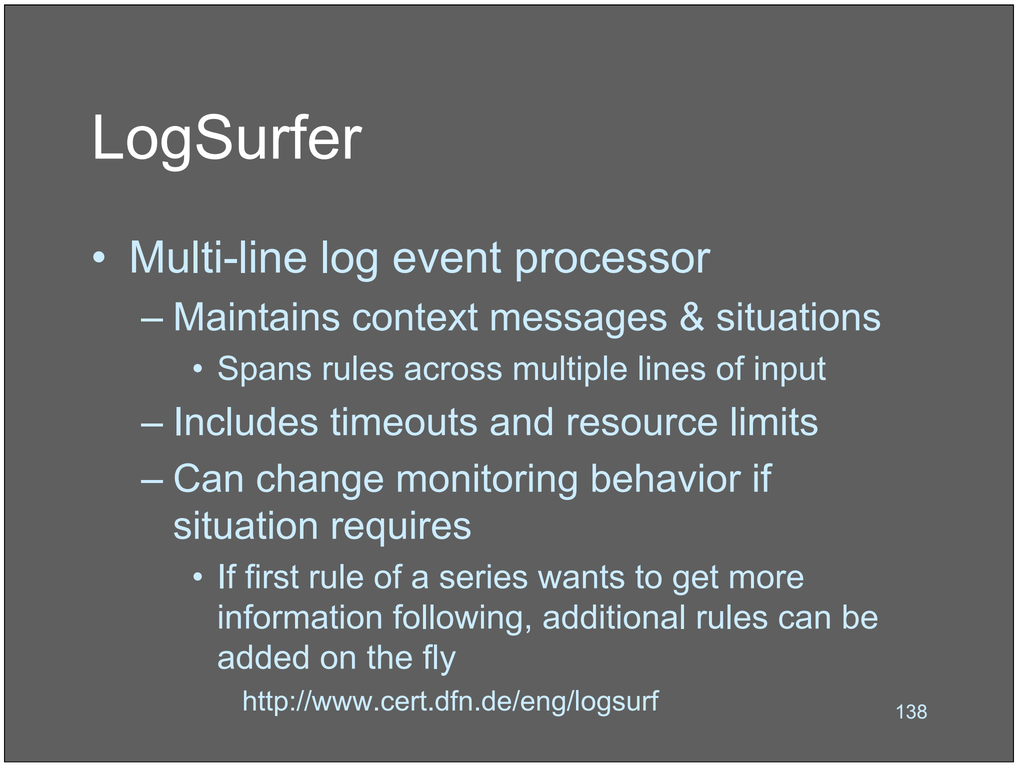
- Excellent and very powerful log summarizer and analyzer
  - Really more of an expert-system
    - Monster big Perl program (but very clean)
    - Lots of rules that are constantly being tweaked and tuned
    - Good source of information regarding messages that are interesting/uninteresting for each service

      http://www.logwatch.org

137

For the entry level basic log watching system, it's hard to beat logwatch, and it's hard to beat logwatch' price.

You can download logwatch from the web, and it takes relatively little time to install it. Tuning it yourself can be a bit tricky, but the good news is that its authors are constantly improving it - so you may not have to.



## LogSurfer

- Multi-line log event processor
  - Maintains context messages & situations
    - Spans rules across multiple lines of input
  - Includes timeouts and resource limits
  - Can change monitoring behavior if situation requires
    - If first rule of a series wants to get more information following, additional rules can be added on the fly

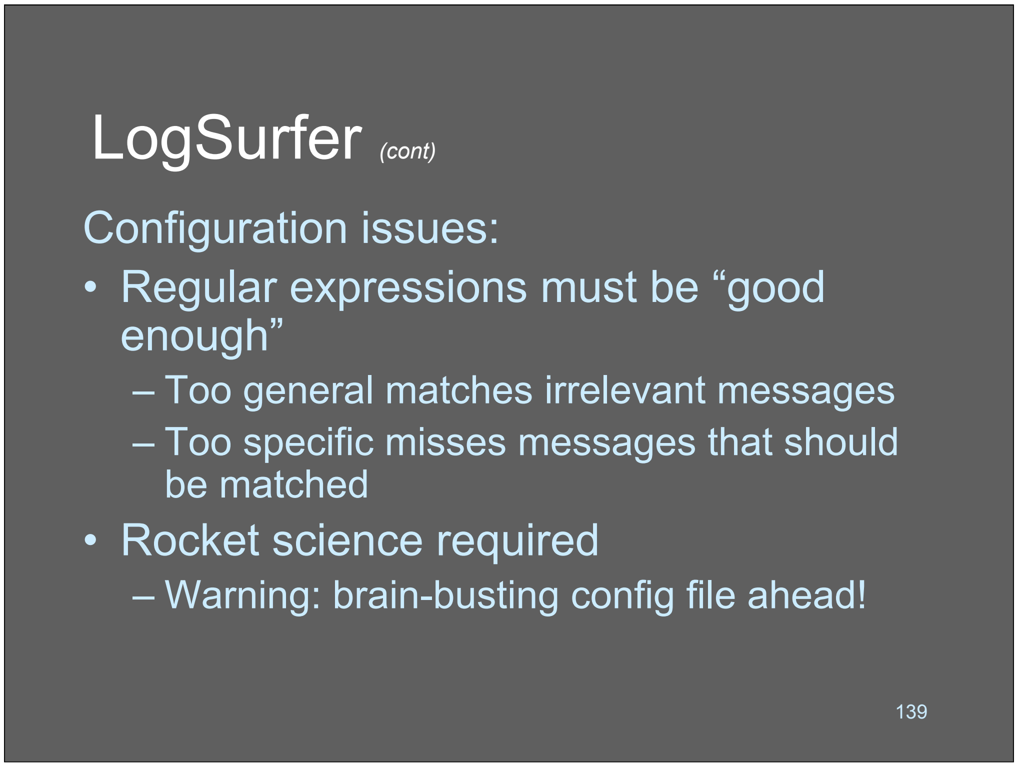
http://www.cert.dfn.de/eng/logsurf

138

If you find logwatch is not powerful enough, you may wish to look at logsurfer. Logsurfer has a lot more options and has some interesting capabilities for combining multiple events in time as well as adjusting the behavior of its monitoring based on what it’s seeing. So, for example, you could begin to collect *everything* from a particular host if you see a certain message go by. That can be quite valuable.

But.

The configuration language logsurfer uses would give an APL programmer nightmares.



## LogSurfer *(cont)*

Configuration issues:

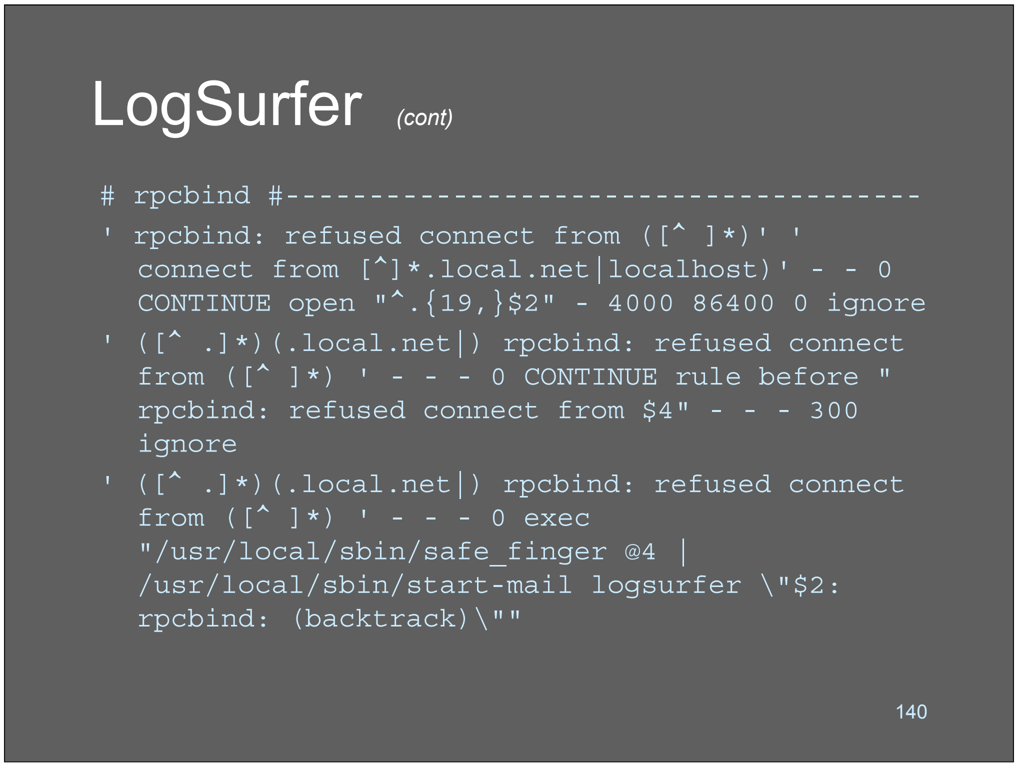
- Regular expressions must be “good enough”
  - Too general matches irrelevant messages
  - Too specific misses messages that should be matched
- Rocket science required
  - Warning: brain-busting config file ahead!

139

*logsurfer* is available at http://www.cert.dfn.de/eng/logsurf. For details on how to deploy it in a Solaris environment, check out

http://www.cert.org/security-improvement/implementations/i042.02.html

Since logsurfer uses regular expressions, you need to be a regular expression guru or you will have a lot of trouble with it.

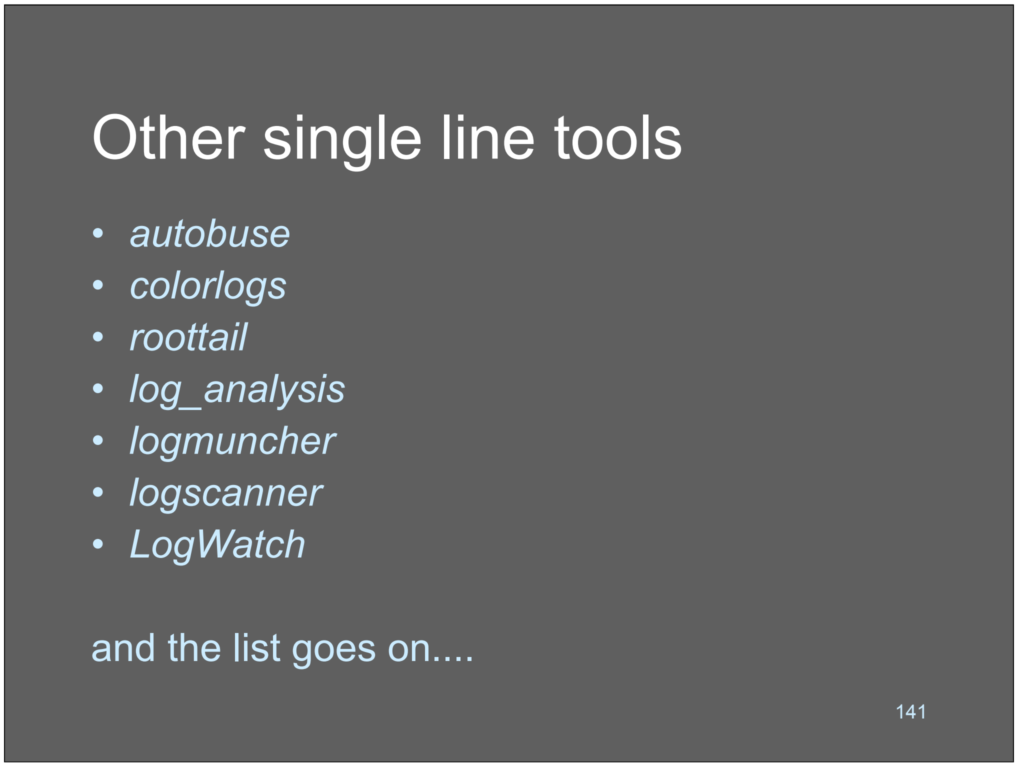


## LogSurfer *(cont)*

```
# rpcbind #-------------------------------------
' rpcbind: refused connect from ([^ ]*)' '
  connect from [^]*.local.net|localhost)' - - 0
  CONTINUE open "^.{19,}$2" - 4000 86400 0 ignore
' ([^ .]*)(.local.net|) rpcbind: refused connect
  from ([^ ]*) ' - - - 0 CONTINUE rule before "
  rpcbind: refused connect from $4" - - - 300
  ignore
' ([^ .]*)(.local.net|) rpcbind: refused connect
  from ([^ ]*) ' - - - 0 exec
  "/usr/local/sbin/safe_finger @4 |
  /usr/local/sbin/start-mail logsurfer \"$2:
  rpcbind: (backtrack)\""
```

140

logsurfer is less popular than swatch or logcheck because of its complexity. Complex? Naaaaaah.



## Other single line tools

- *autobuse*
- *colorlogs*
- *roottail*
- *log_analysis*
- *logmuncher*
- *logscanner*
- *LogWatch*

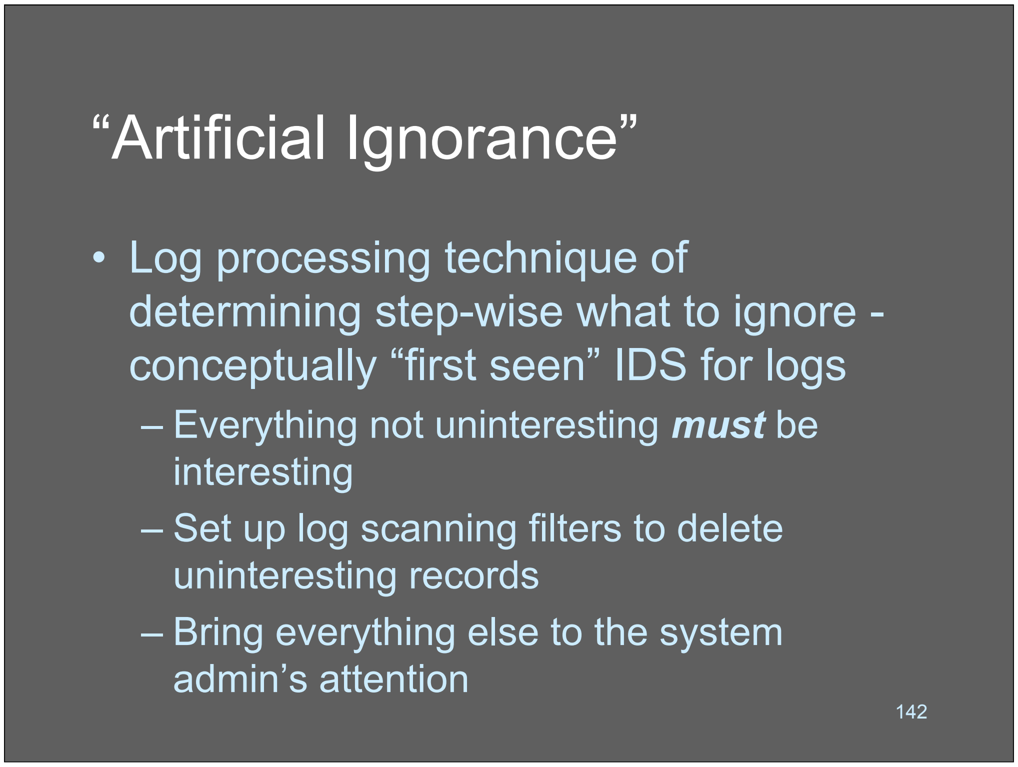
and the list goes on....

141

There are nearly as many programs for parsing log files as there are networks with log files that need to be parsed. This is mostly because everyone wants slightly different information out of their data, and of course every network is different, so a lot of system administrators decide to roll their own rather than to modify someone else's. The applications listed in these slides are provided for information only; I haven't used them, I don't know much about their performance or flexibility. *colorlogs* color codes log files based on a keyword file and a user's configuration file. If you're a visually oriented person, this might be really useful – and great for Web monitoring systems. It's available at http://www.resentment.org/projects/colorlogs/.

The OpenSystems *Network Intelligence Engine* is a commercial appliance that supports a variety of Cisco devices, FireWall-1, and other systems.. It's available at http://www.opensystems.com/

For experienced programmers, the tools available at ftp.sdsc.edu/pub/security/PICS are improved versions of *swatch, syslog* and a couple of other audit applications.



## “Artificial Ignorance”

- Log processing technique of determining step-wise what to ignore - conceptually “first seen” IDS for logs
  - Everything not uninteresting ***must*** be interesting
  - Set up log scanning filters to delete uninteresting records
  - Bring everything else to the system admin’s attention

142

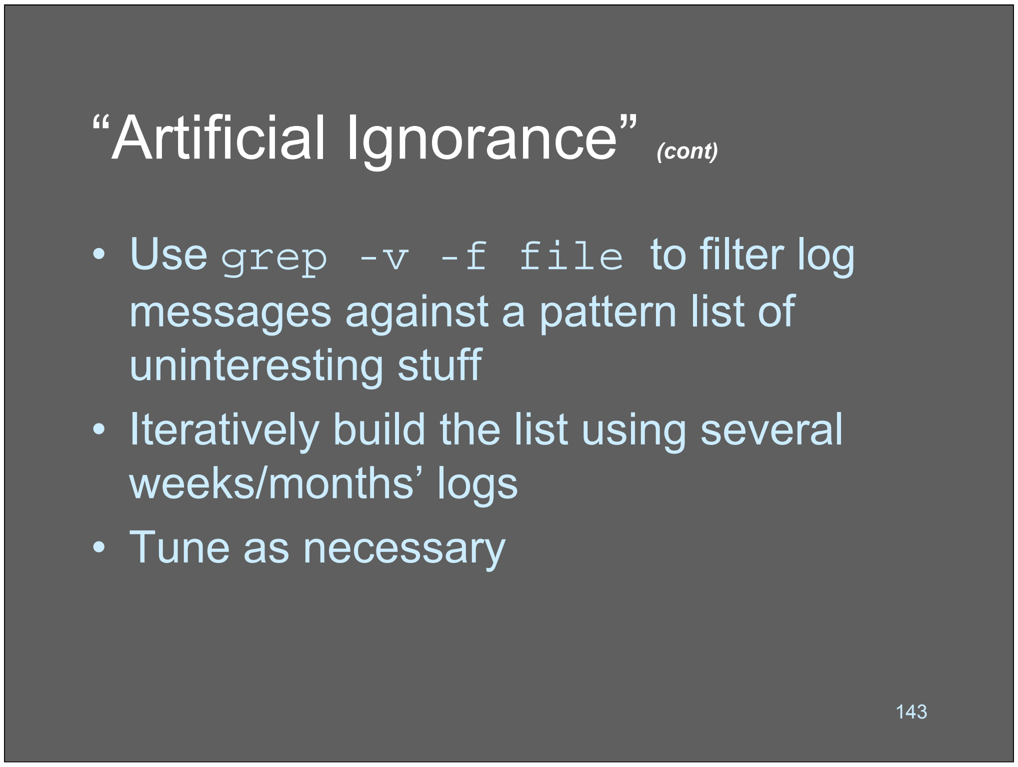
Artificial Ignorance is an incredibly valuable approach to dealing with system logs. I first set it up on a firewall product that I was building in the early 1990’s. After a few firewalls had shipped, I noticed that the log messages were extremely regular and I wrote a log-processing script to collect and summarize data from them. Then I thought about making it detect attacks, and decided to invert the process. Artficial Ignorance was born!

The idea is simple: know what you are *not* interested and throw it away: bring everything else to someone’s attention.

The first system I shipped that was running my artificial ignorance one day kicked back a log message that read something like:

```
wd0: sense retry failed
```

I’ve been a convert ever since. As a security guy I would have only thought to look in the logs for security-related stuff and didn’t even *think* of hard disk errors.



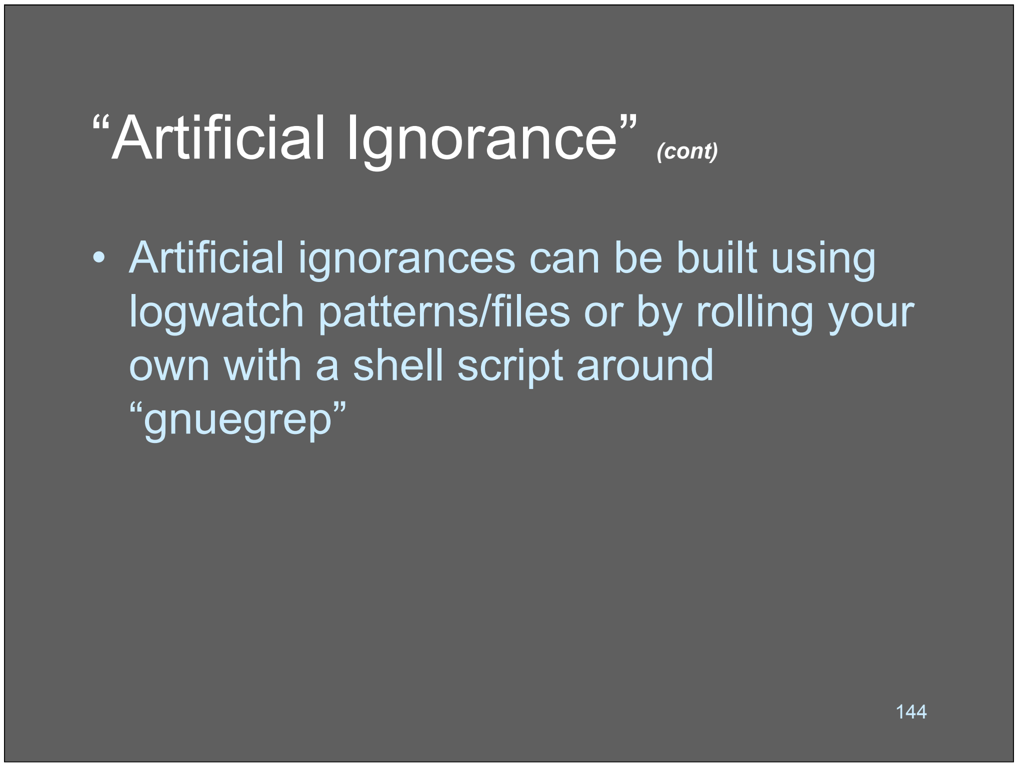
## “Artificial Ignorance” *(cont)*

- Use `grep -v -f file` to filter log messages against a pattern list of uninteresting stuff
- Iteratively build the list using several weeks/months’ logs
- Tune as necessary

143

The other thing that’s great about Artificial Ignorance is how laughably easy it is to set one up. Sure, you can get fancy but -why? Just use plain old UNIX grep (-v -f) to drive the program.

Grep -v says “print only lines that *don’t* match our pattern” and -f gives the filename containing the patterns. Building a pattern file is just an iterative process of adding regular expressions to the file and re-running it through the process. You can manually baseline your Artificial Ignorance by preparing a pattern file, or you can just start adding them as time goes by and eventually your logs will tail off.



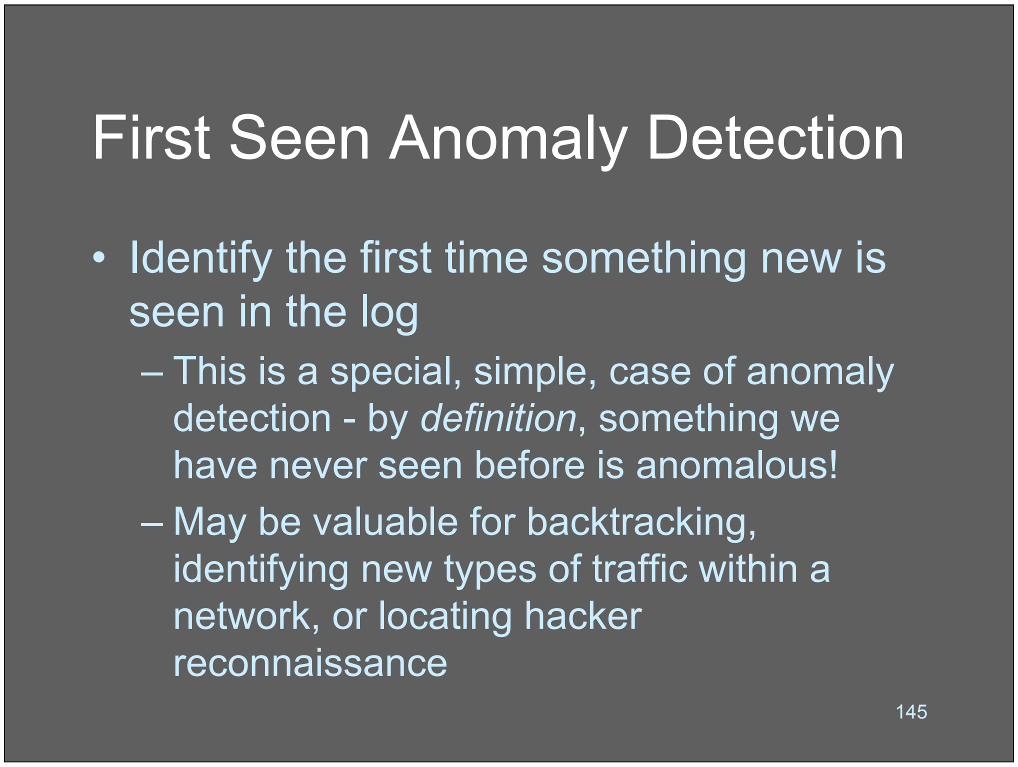
## “Artificial Ignorance” *(cont)*

- Artificial ignorances can be built using logwatch patterns/files or by rolling your own with a shell script around “gnuegrep”

144

See:
http://www.ranum.com/security/computer_security/papers/ai/

for a complete write-up on how to iteratively build an artificial ignorance pattern file.



## First Seen Anomaly Detection

- Identify the first time something new is seen in the log
  - This is a special, simple, case of anomaly detection - by *definition*, something we have never seen before is anomalous!
  - May be valuable for backtracking, identifying new types of traffic within a network, or locating hacker reconnaissance

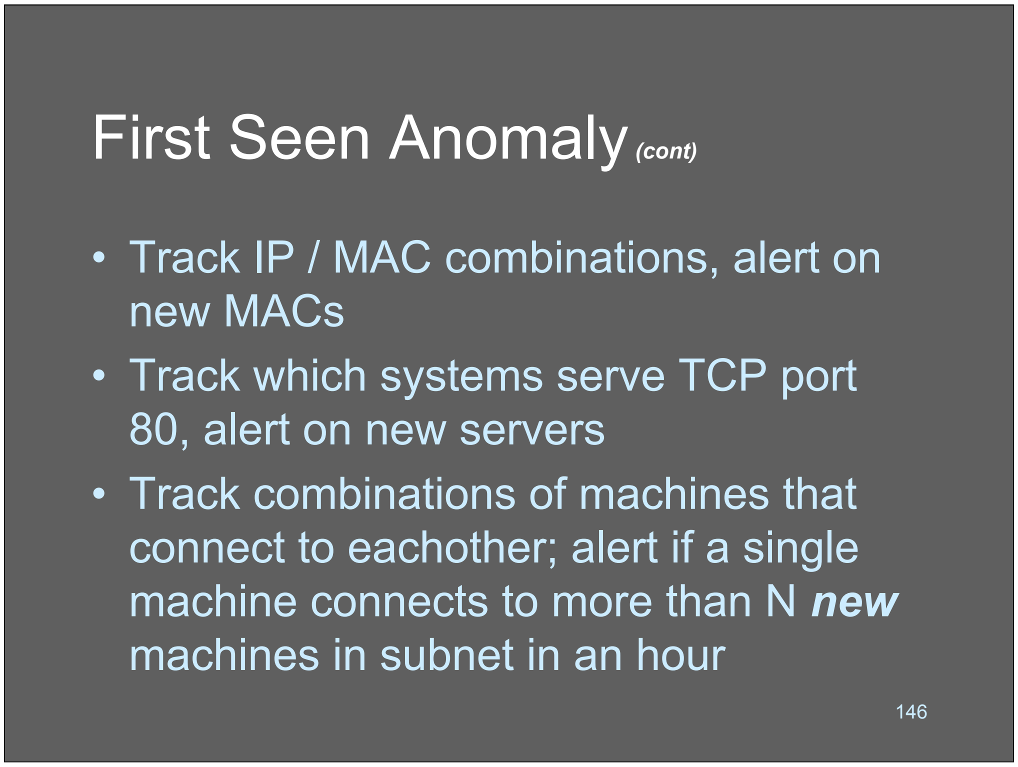
145

First seen anomaly detection is another of those really simple, dumb, effective ideas like artificial ignorance. In fact, they are almost but not quite the same thing.

The term “anomaly detection” has been abused by marketing people for IDS companies until it has lost most of its meaning. The idea, though, is to detect that which is unusual or anomalous.

*What’s more unusual than something that has never happened before?*

Like Artificial Ignorance, this technique works wonderfully and is extremely low-tech.

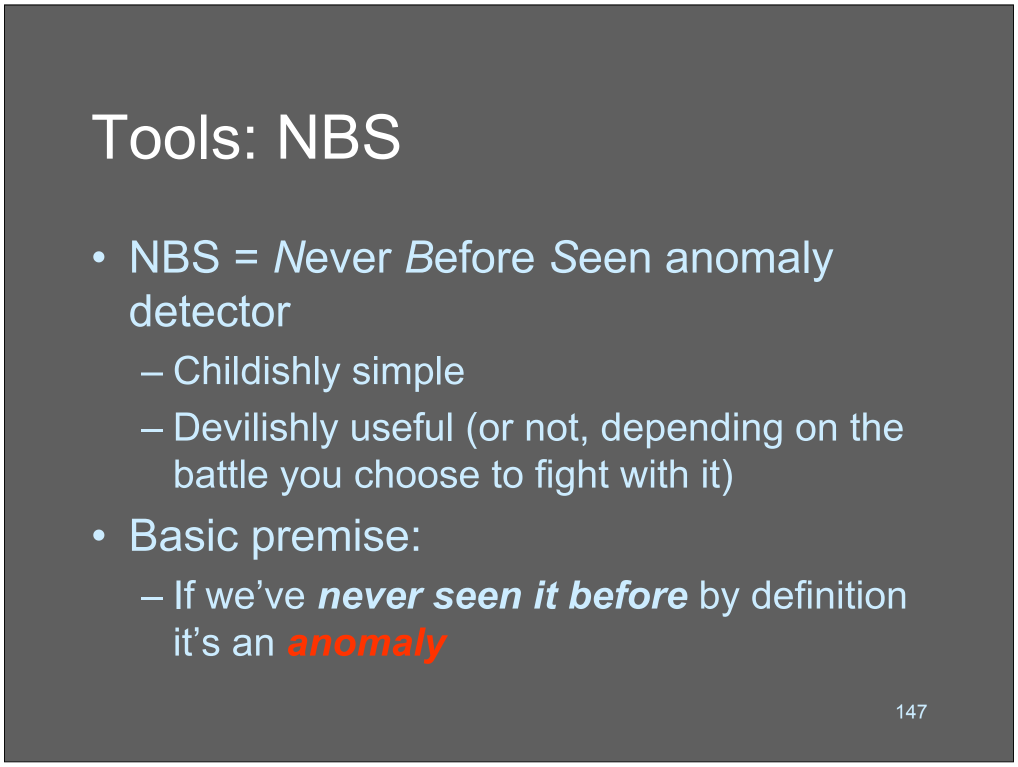


## First Seen Anomaly *(cont)*

- Track IP / MAC combinations, alert on new MACs
- Track which systems serve TCP port 80, alert on new servers
- Track combinations of machines that connect to eachother; alert if a single machine connects to more than N ***new*** machines in subnet in an hour

146

Virtually anything on your network and systems that achieves  a steady-state is a good candidate for NBS. Mail sender addresses? Mail recipient addresses? Machines returning ACK packets to SYN packets on port 80? New IP to IP mappings? New ARP mappings? New users executing programs, by program? Etc. You can imagine all kinds of insanely fun applications for this concept!



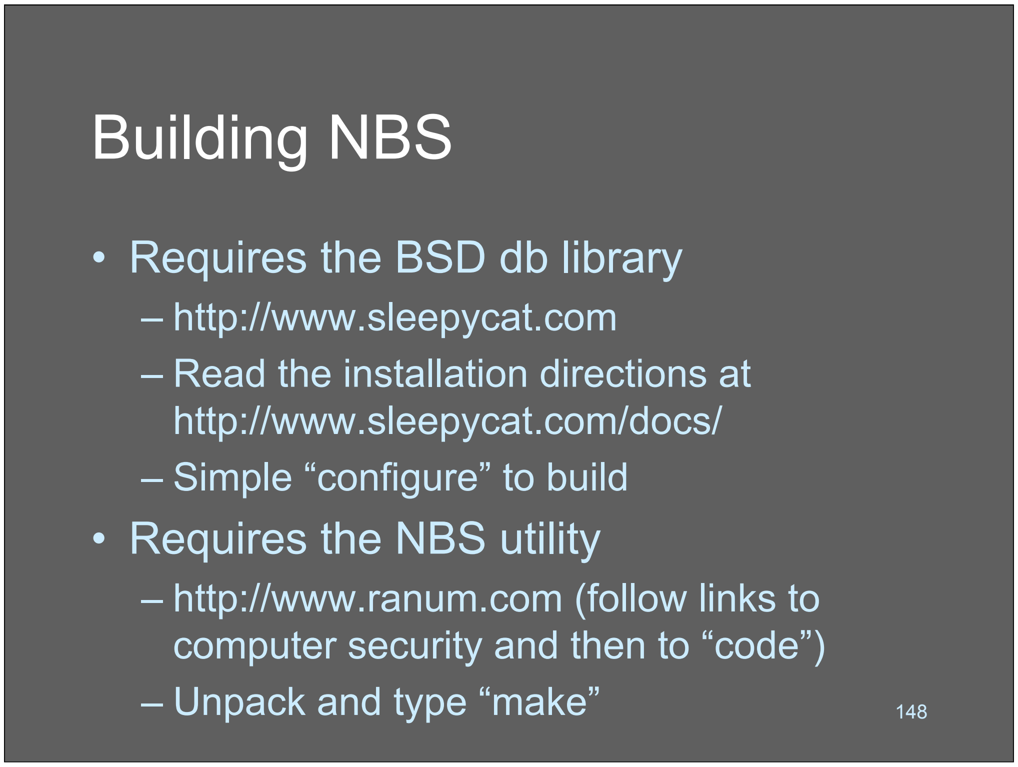
## Tools: NBS

- NBS = *N*ever *B*efore *S*een anomaly detector
  - Childishly simple
  - Devilishly useful (or not, depending on the battle you choose to fight with it)
- Basic premise:
  - If we've ***never seen it before*** by definition it's an ***anomaly***

147

Typically, the reason people don't do NBS is because it might (potentially) have to manage a lot of data entries and the processing could get intense. Imagine mapping connectivity between all your 3,000 hosts! There are 9,000,000 possible combinations. The database would get huge!

Fortunately, I've solved that. The next few slides provide a walkthrough of the NBS anomaly detection driver.



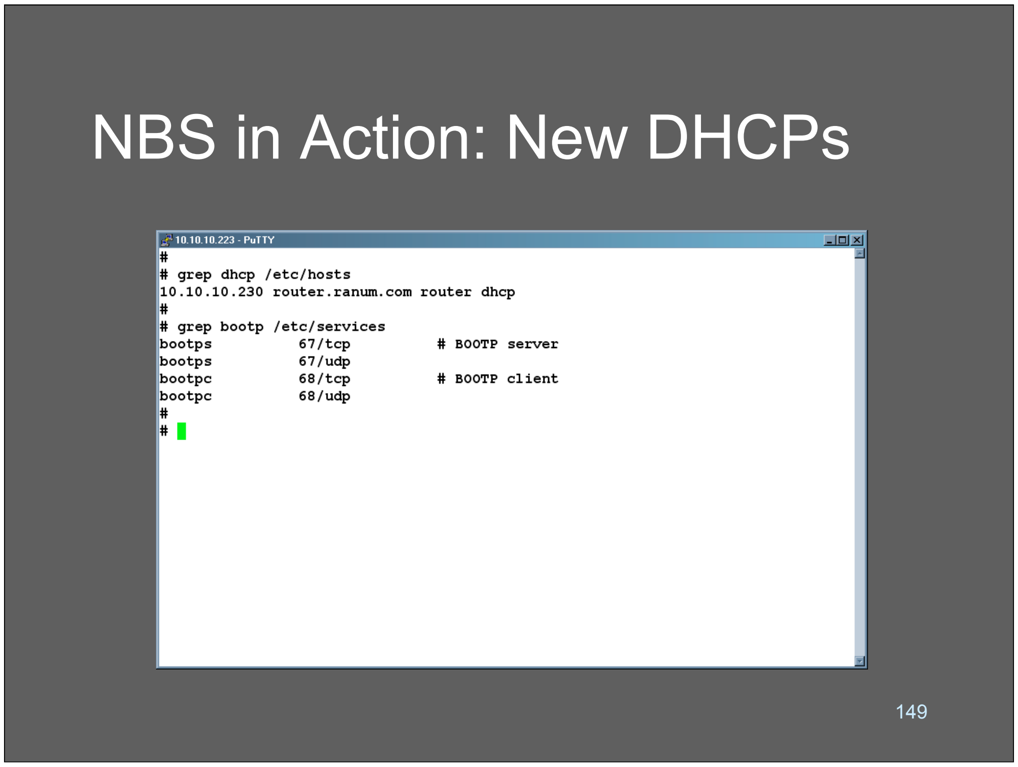
## Building NBS

- Requires the BSD db library
  - http://www.sleepycat.com
  - Read the installation directions at http://www.sleepycat.com/docs/
  - Simple “configure” to build
- Requires the NBS utility
  - http://www.ranum.com (follow links to computer security and then to “code”)
  - Unpack and type “make”

148

To build the NBS driver you’ll need to BSD database library; NBS is basically nothing more than a couple of B-tree indexes, and the BSD database library has a very high-performance and reliable implementation of B-trees.

Once you’ve build BSD-DB (read the directions! It’s not just “configure” “make”) and installed it, then get the NBS sources off of Marcus Ranum’s web page and unpack and compile them. You may need to change the LIBS= parameter in the Makefile and may need to add -I/usr/local/BerkelyDB*blahblah* to the CFLAGS= parameter so that the #include file for <db.h> is found. It shouldn’t be hard to build!



## NBS in Action: New DHCPs

```
#
# grep dhcp /etc/hosts
10.10.10.230 router.ranum.com router dhcp
#
# grep bootp /etc/services
bootps          67/tcp          # BOOTP server
bootps          67/udp
bootpc          68/tcp          # BOOTP client
bootpc          68/udp
#
#
```

149

This is a walkthrough example of how easy it is to use NBS and the kind of results you can get with it. Setup time for something like this is a few *minutes* and it'll give you results that might make you a local hero someday.

Here what we are doing is set-up stuff. We're going to monitor our network for Never Before Seen MAC/IP combinations being leased by DHCP servers. If any of those 3 values changes, we're going to know about it!

So, with one simple operation we'll:

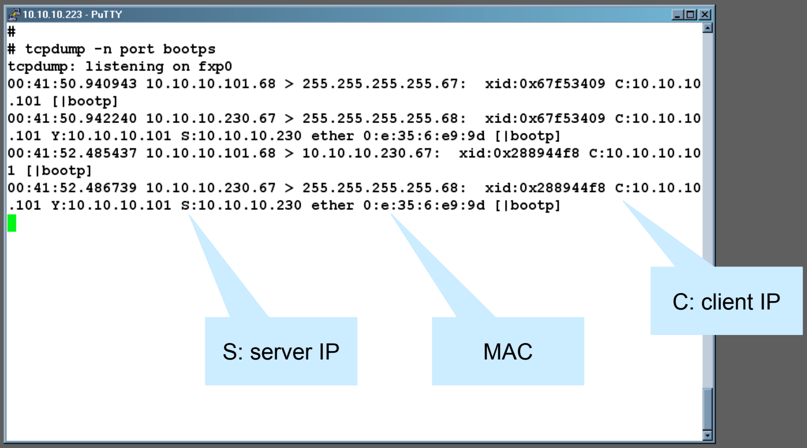
•Learn about new MACs

•Learn about new DHCP servers (that should be rare!)

•Learn about new IP addresses being leased that we don't normally lease

First subtlety: DHCP is kludged onto bootp. So we need to look at bootp traffic. Bootp has a server and a client side. Let's watch the server, because it's less likely to lie. If we were looking for NBSsing malformed records and weird client queries we could look at the client side, but that's a project for another day.

## Tcpdump as log data source



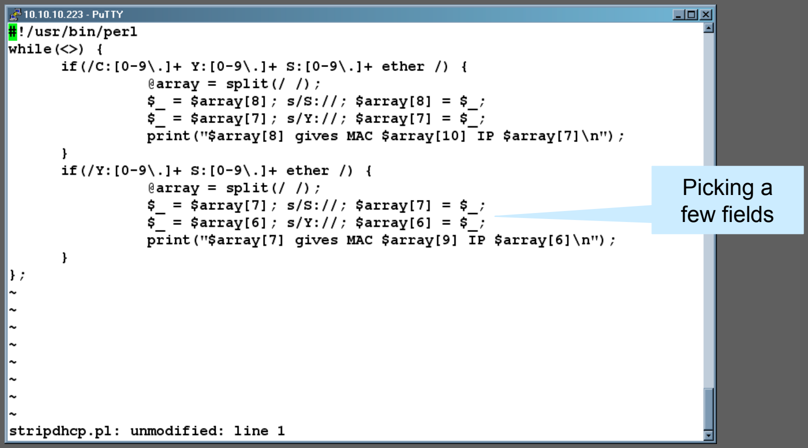
```
#
# tcpdump -n port bootps
tcpdump: listening on fxp0
00:41:50.940943 10.10.10.101.68 > 255.255.255.255.67:  xid:0x67f53409 C:10.10.10
.101 [|bootp]
00:41:50.942240 10.10.10.230.67 > 255.255.255.255.68:  xid:0x67f53409 C:10.10.10
.101 Y:10.10.10.101 S:10.10.10.230 ether 0:e:35:6:e9:9d [|bootp]
00:41:52.485437 10.10.10.101.68 > 10.10.10.230.67:  xid:0x288944f8 C:10.10.10.10
1 [|bootp]
00:41:52.486739 10.10.10.230.67 > 255.255.255.255.68:  xid:0x288944f8 C:10.10.10
.101 Y:10.10.10.101 S:10.10.10.230 ether 0:e:35:6:e9:9d [|bootp]
```

150

NBS works on text strings. *Any* strings. So, we want to turn our bootp service traffic into strings. What better tool to do that than good old Tcpdump?

Here we run tcpdump on bootp traffic and observe some of how it works. The main thing we observe is the layout of the traffic that we're looking for. Aha! There's the client, server, and MAC address. We have everything we need - now let's just strip those fields out with a little script!

## Marcus is a Perl Newbie



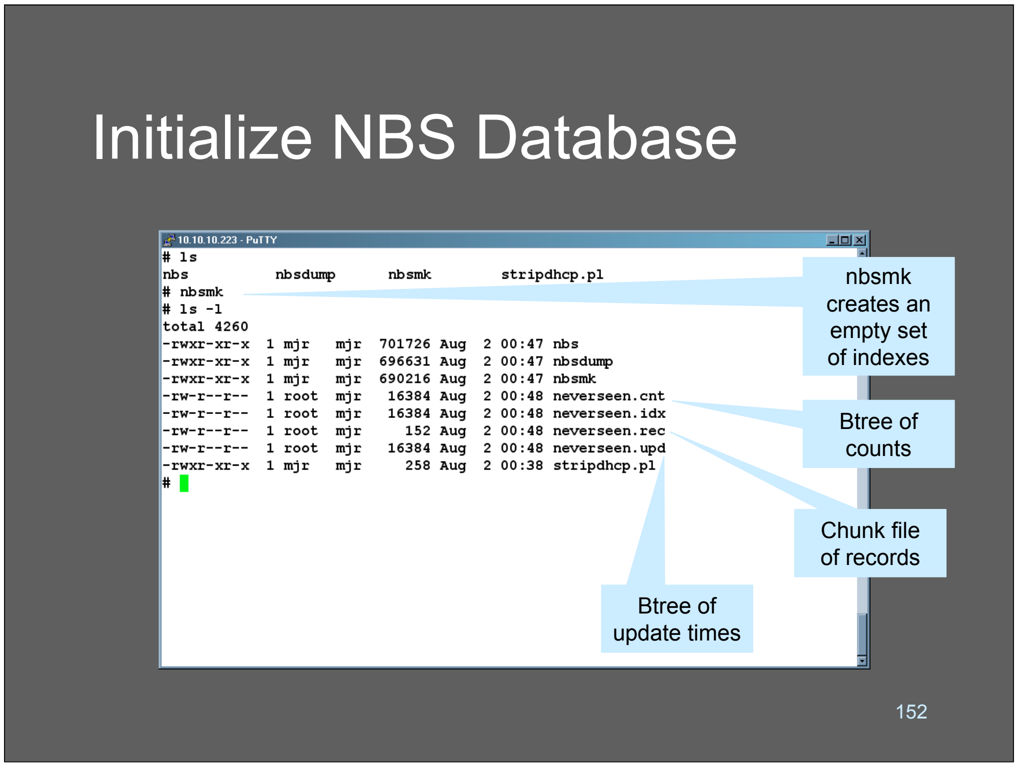
```
#!/usr/bin/perl
while(<>) {
      if(/C:[0-9\.]+ Y:[0-9\.]+ S:[0-9\.]+ ether /) {
                @array = split(/ /);
                $_ = $array[8]; s/S://; $array[8] = $_;
                $_ = $array[7]; s/Y://; $array[7] = $_;
                print("$array[8] gives MAC $array[10] IP $array[7]\n");
      }
      if(/Y:[0-9\.]+ S:[0-9\.]+ ether /) {
                @array = split(/ /);
                $_ = $array[7]; s/S://; $array[7] = $_;
                $_ = $array[6]; s/Y://; $array[6] = $_;
                print("$array[7] gives MAC $array[9] IP $array[6]\n");
      }
};
~
~
~
~
~
~
~
~
stripdhcp.pl: unmodified: line 1
```

151

I suck at perl. In fact, you’re looking at my very first (and so far: only) perl program. Normally, for something like this I’d use awk but perl seems to be everyone’s choice these days… *sigh* Dan Klein has since explained to me how to write this same script much better. But you get the idea.

Anyhow, what this script does is matches on the two different forms in which *the version of tcpdump on my machine* outputs the DHCP lease-giving packet. One downside with using a program as input to another program is that different versions may vary their outputs and break everything.

Just make this work for your machine and don’t mess with it.



## Initialize NBS Database

```
# ls
nbs          nbsdump      nbsmk        stripdhcp.pl
# nbsmk
# ls -l
total 4260
-rwxr-xr-x  1 mjr   mjr   701726 Aug  2 00:47 nbs
-rwxr-xr-x  1 mjr   mjr   696631 Aug  2 00:47 nbsdump
-rwxr-xr-x  1 mjr   mjr   690216 Aug  2 00:47 nbsmk
-rw-r--r--  1 root  mjr    16384 Aug  2 00:48 neverseen.cnt
-rw-r--r--  1 root  mjr    16384 Aug  2 00:48 neverseen.idx
-rw-r--r--  1 root  mjr      152 Aug  2 00:48 neverseen.rec
-rw-r--r--  1 root  mjr    16384 Aug  2 00:48 neverseen.upd
-rwxr-xr-x  1 mjr   mjr      258 Aug  2 00:38 stripdhcp.pl
#
```

152

The program nbsmk is used to create a new NBS database. The way NBS works, it reads its configuration from the database itself. So once you've created the database you can just call nbs on it and it'll figure everything out.

The default nbsmk parameters create a Btree of counts, a Btree of update times, a Btree of nbs strings, and a record file of update/count information.

There are a bunch of options to nbsmk that can cause it to omit some of the indexes for additional performance. In fact, if you want no information other than just that a record was seen, you can omit updating the data records and save a lot of space and time. Generally, nbs is fast enough that you don't need to worry about performance even for very large datasets.

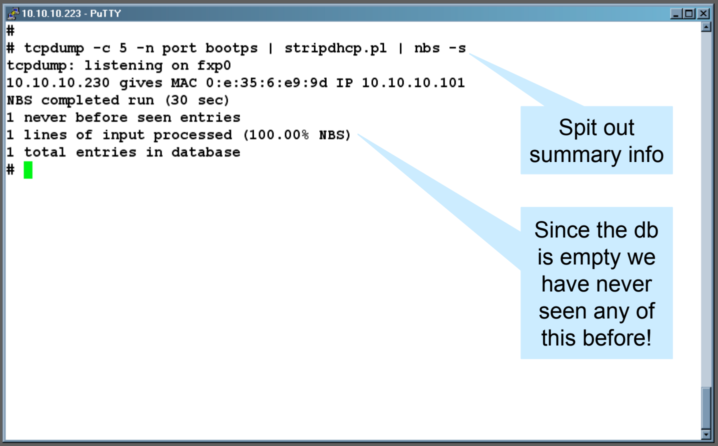
If you call nbsmk with the -? Flag or a bad flag it'll print out its current set of options. The most popular alternative option is the:

```
-d database
```

option. This allows you to set the database name to something other than the default (neverseen). Full pathnames also work, e.g.:

```
nbsmk -d /var/nbs/dhcp
```

## Tcpdump | stripdhcp | nbs



```
10.10.10.223 - PuTTY
#
# tcpdump -c 5 -n port bootps | stripdhcp.pl | nbs -s
tcpdump: listening on fxp0
10.10.10.230 gives MAC 0:e:35:6:e9:9d IP 10.10.10.101
NBS completed run (30 sec)
1 never before seen entries
1 lines of input processed (100.00% NBS)
1 total entries in database
#
```

Spit out summary info
Since the db is empty we have never seen any of this before!

153

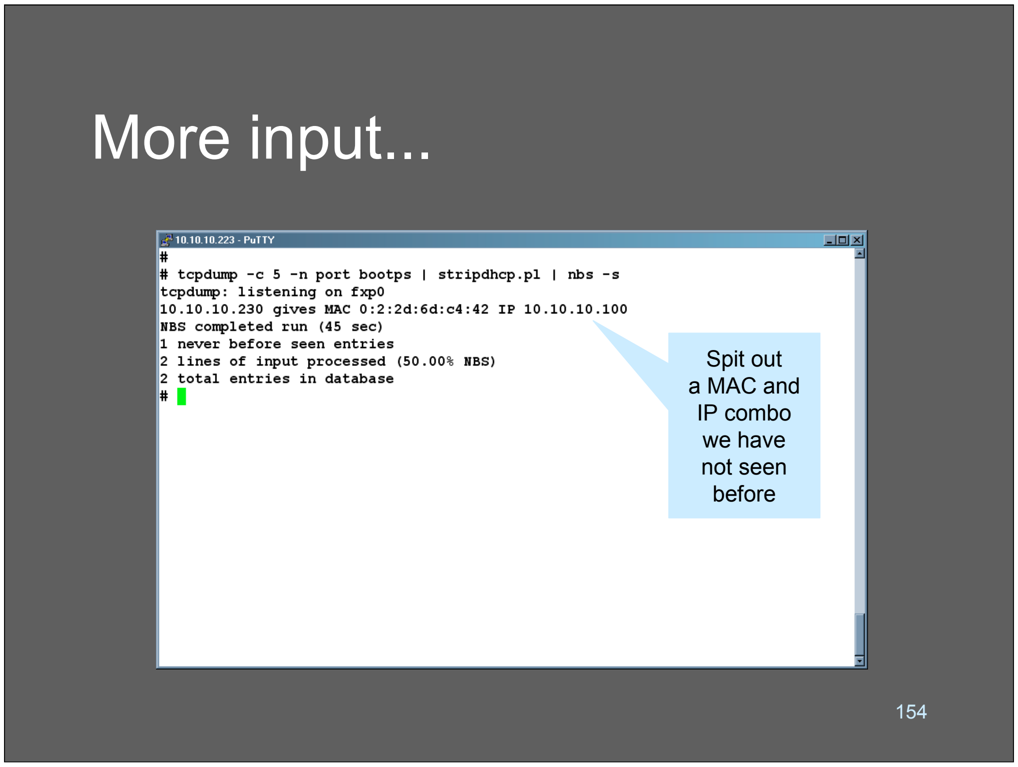
Here we run the initial "training" round of nbs. Since it's never seen *anything* before, everything it sees now is an anomaly. So nbs prints it out on the standard output. The:

-s

flag tells nbs to output a summary when it is completed its run. The "100% NBS" score indicates that every line of input we sent through had never been seen before. That's not unexpected.

In this particular example, our use of NBS is a bit contrived. Tcpdump is exiting after collecting 5 packets, in order to cause it to actually terminate. In a production setup, you might run this in a loop, in which tcpdump exits after every 100,000 packets, so nbs flushes and the script is able to process the output. An alternative approach is to have a script that starts and restarts tcpdump:

```
OLDPID=`cat /var/log/tcpdump.pid`
mv newdump.out olddump.out
nohup tcpdump -n -w newdump.out &
echo $! > /var/log/tcpdump.pid
kill -HUP $OLDPID
tcpdump -n -r olddump.out | nbs -s > /tmp/nbs.out
if [ -s /tmp/nbs.out ]; then
        cat /tmp/nbs.out | mail -s "NBS report" mjr
fi
rm olddump.out
```



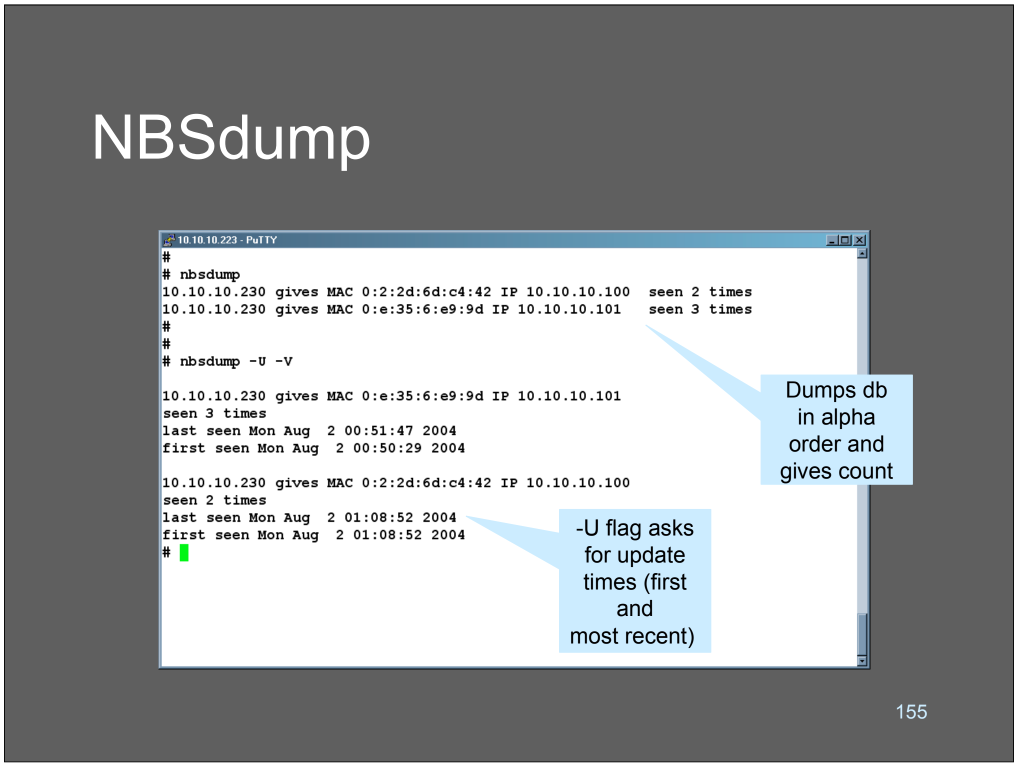
## More input...

```
#
# tcpdump -c 5 -n port bootps | stripdhcp.pl | nbs -s
tcpdump: listening on fxp0
10.10.10.230 gives MAC 0:2:2d:6d:c4:42 IP 10.10.10.100
NBS completed run (45 sec)
1 never before seen entries
2 lines of input processed (50.00% NBS)
2 total entries in database
#
```

154

When we run a second pass with new data (I turned on a new wireless machine in the home network) - surprise! We see a never before seen value! From now on, we’ll never get notified about that particular value, unless we age it from the database.

That’s enough of a simple example! Let’s look at a more complex and possibly more powerful one!



## NBSdump

```
#
# nbsdump
10.10.10.230 gives MAC 0:2:2d:6d:c4:42 IP 10.10.10.100  seen 2 times
10.10.10.230 gives MAC 0:e:35:6:e9:9d IP 10.10.10.101   seen 3 times
#
#
# nbsdump -U -V

10.10.10.230 gives MAC 0:e:35:6:e9:9d IP 10.10.10.101
seen 3 times
last seen Mon Aug  2 00:51:47 2004
first seen Mon Aug  2 00:50:29 2004

10.10.10.230 gives MAC 0:2:2d:6d:c4:42 IP 10.10.10.100
seen 2 times
last seen Mon Aug  2 01:08:52 2004
first seen Mon Aug  2 01:08:52 2004
#
```

10.10.10.223 - PuTTY
Dumps db in alpha order and gives count
-U flag asks for update times (first and most recent)
155

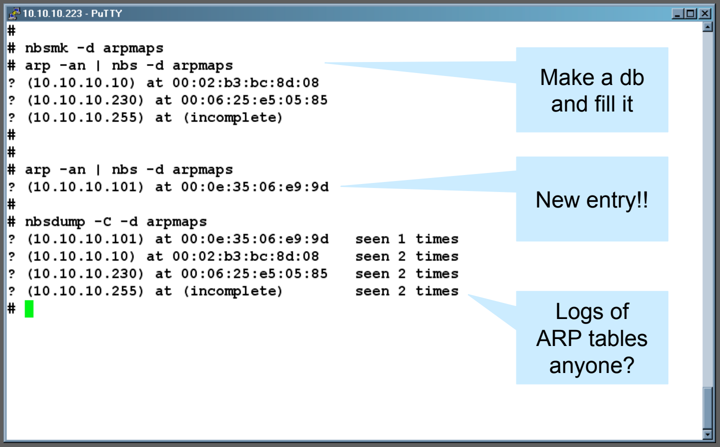
Here we are using nbsdump to output the contents of an NBS database. There are a variety of different flags you can pass to the dumper to control the format of the output as well as the order and count of what is dumped.

One of the neat things about Btrees is that they inherently sort the data that is stored in them. That's *why* nbs uses Btrees - when you dump the records, you get them sorted based on whatever value you asked for. If you ask for counts, you get the results sorted from least frequently seen to most frequently, etc. This can be incredibly useful and - most importantly - incredibly fast. A lot of the time log analysis is not done because of the processing constraints inherent in searching and sorting through lots of records. Nbs keeps the databases the way it does so that things like counts are pre-computed for you.

In the first example, we dump the values, and it displays the number of times they are seen.

In the second example, we asked for the last update time (-U) and full output (-V) which prints the time that the record was first installed in the database as well as the last update time. When we requested the last update times, the output is sorted in terms of most recent to least recent.

## NBSsing ARP maps

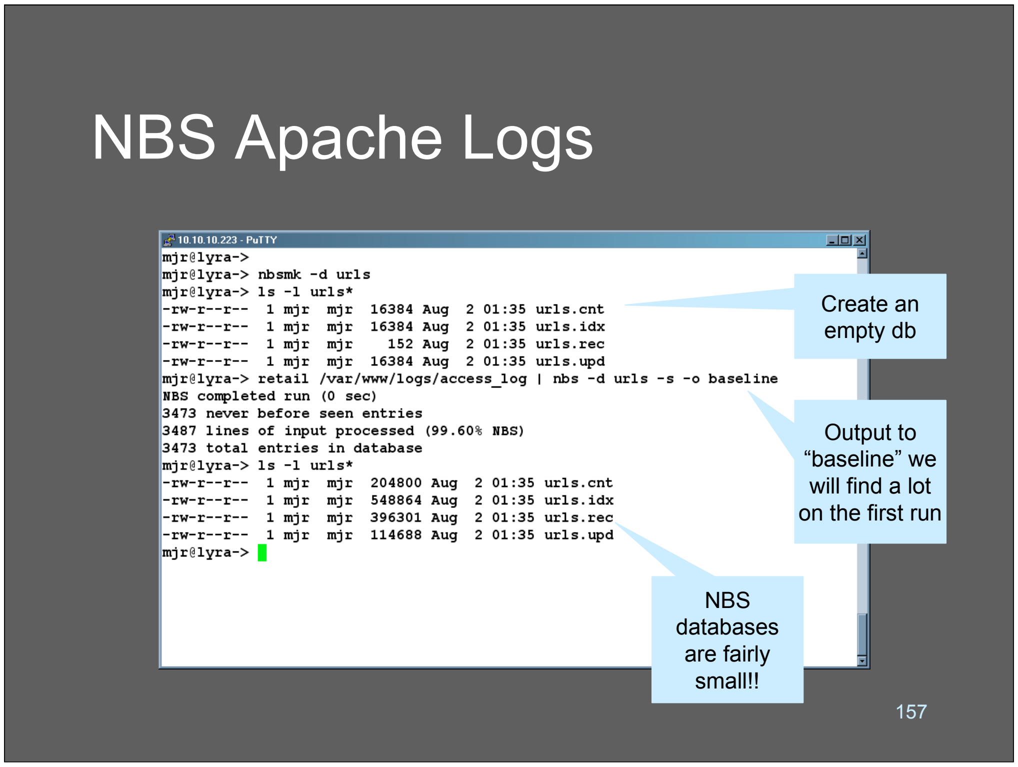


```
#
# nbsmk -d arpmaps
# arp -an | nbs -d arpmaps
? (10.10.10.10) at 00:02:b3:bc:8d:08
? (10.10.10.230) at 00:06:25:e5:05:85
? (10.10.10.255) at (incomplete)
#
#
# arp -an | nbs -d arpmaps
? (10.10.10.101) at 00:0e:35:06:e9:9d
#
# nbsdump -C -d arpmaps
? (10.10.10.101) at 00:0e:35:06:e9:9d   seen 1 times
? (10.10.10.10) at 00:02:b3:bc:8d:08    seen 2 times
? (10.10.10.230) at 00:06:25:e5:05:85   seen 2 times
? (10.10.10.255) at (incomplete)        seen 2 times
#
```

156

In this example, we're tracking ARP/IP mappings on a network. Remember, we won't get notified whenever there's a *change* we'll only get notified when something happens that we've never seen before.

This would be a fairly simple cron job. Operationally, I'd expect to have a cron job run every 20 minutes that collected various bits of never-before seen stuff into a directory of output files. I'd then put a master script together that wrapped them all up into an Email message to the administrator.



## NBS Apache Logs

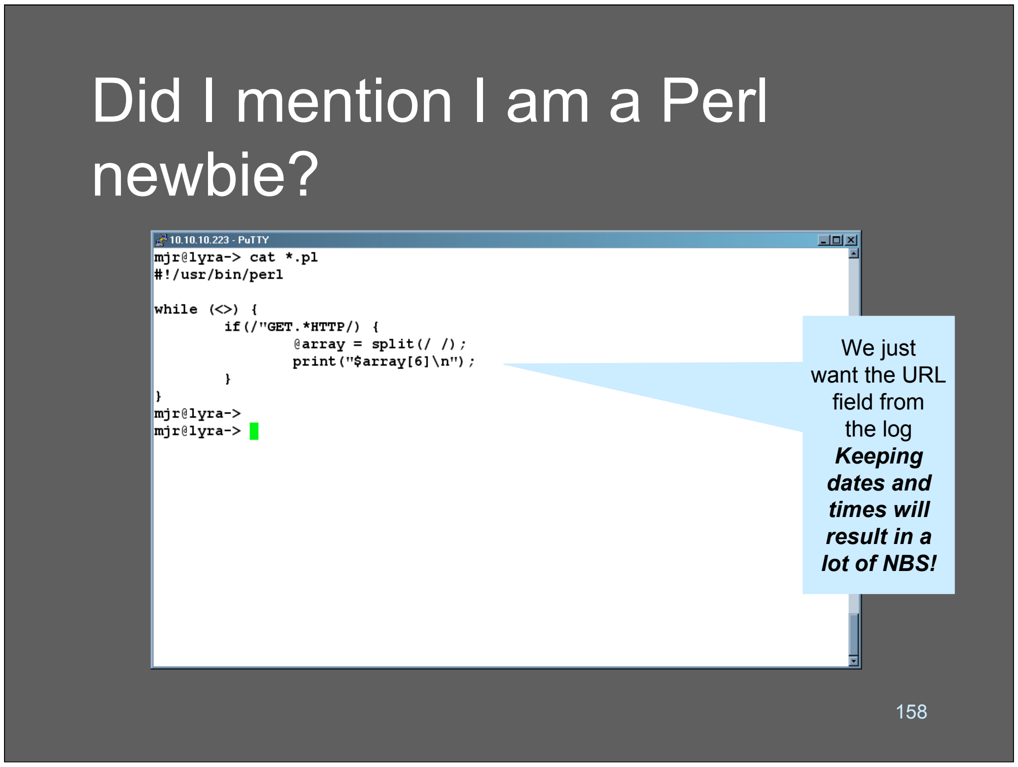
```
mjr@lyra->
mjr@lyra-> nbsmk -d urls
mjr@lyra-> ls -l urls*
-rw-r--r--  1 mjr  mjr  16384 Aug  2 01:35 urls.cnt
-rw-r--r--  1 mjr  mjr  16384 Aug  2 01:35 urls.idx
-rw-r--r--  1 mjr  mjr    152 Aug  2 01:35 urls.rec
-rw-r--r--  1 mjr  mjr  16384 Aug  2 01:35 urls.upd
mjr@lyra-> retail /var/www/logs/access_log | nbs -d urls -s -o baseline
NBS completed run (0 sec)
3473 never before seen entries
3487 lines of input processed (99.60% NBS)
3473 total entries in database
mjr@lyra-> ls -l urls*
-rw-r--r--  1 mjr  mjr  204800 Aug  2 01:35 urls.cnt
-rw-r--r--  1 mjr  mjr  548864 Aug  2 01:35 urls.idx
-rw-r--r--  1 mjr  mjr  396301 Aug  2 01:35 urls.rec
-rw-r--r--  1 mjr  mjr  114688 Aug  2 01:35 urls.upd
mjr@lyra->
```

10.10.10.223 - PuTTY
Create an empty db
Output to "baseline" we will find a lot on the first run
NBS databases are fairly small!!
157

Here we're dumping web server access logs into an NBS database.

You'll notice that most of the URLs going into the database are ones that it has seen before, even during the training run. This works well on my website because there are not a lot of files or dynamic content.

NBS does not work very well on sites with a lot of dynamic content; since everything is "never before seen" if it changes all the time.



## Did I mention I am a Perl newbie?

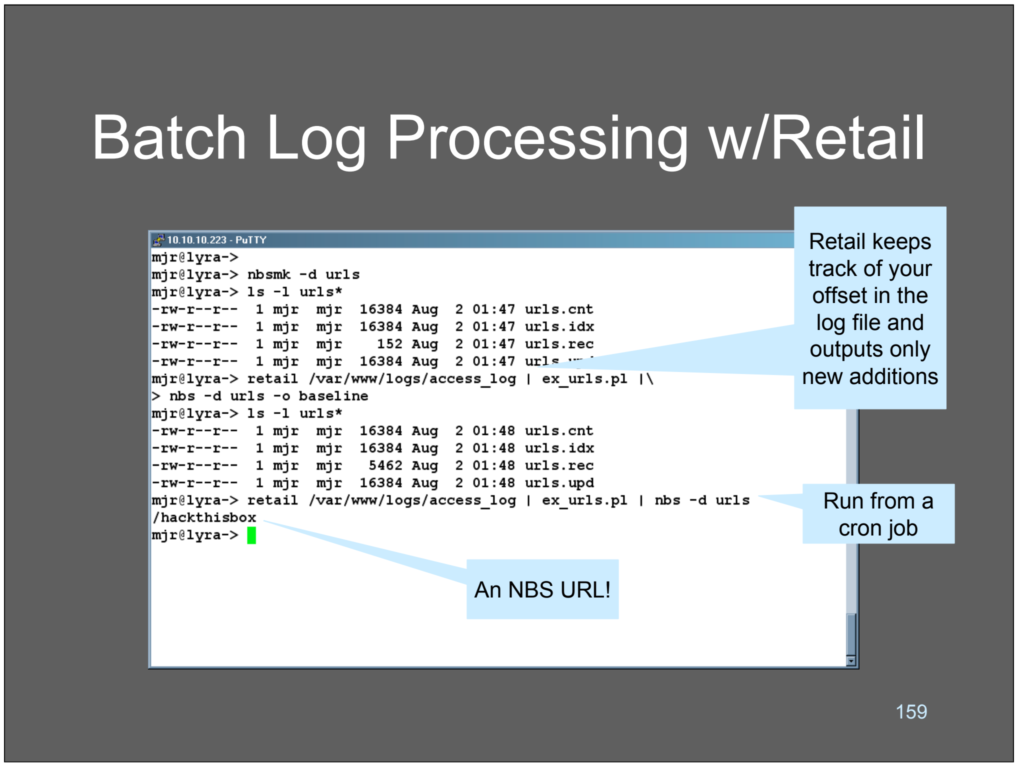
```
mjr@lyra-> cat *.pl
#!/usr/bin/perl

while (<>) {
        if(/"GET.*HTTP/) {
                @array = split(/ /);
                print("$array[6]\n");
        }
}
mjr@lyra->
mjr@lyra->
```

We just want the URL field from the log ***Keeping dates and times will result in a lot of NBS!***

158

This is my second-ever perl program, so please be forgiving.

All that this script does is pulls the URL out of “get” methods in the log file. So we’re just going to build a database of URLs that have been requested from our system. Very simple.



## Batch Log Processing w/Retail

10.10.10.223 - PuTTY

```
mjr@lyra->
mjr@lyra-> nbsmk -d urls
mjr@lyra-> ls -l urls*
-rw-r--r--   1 mjr  mjr  16384 Aug  2 01:47 urls.cnt
-rw-r--r--   1 mjr  mjr  16384 Aug  2 01:47 urls.idx
-rw-r--r--   1 mjr  mjr    152 Aug  2 01:47 urls.rec
-rw-r--r--   1 mjr  mjr  16384 Aug  2 01:47 urls
mjr@lyra-> retail /var/www/logs/access_log | ex_urls.pl |\
> nbs -d urls -o baseline
mjr@lyra-> ls -l urls*
-rw-r--r--   1 mjr  mjr  16384 Aug  2 01:48 urls.cnt
-rw-r--r--   1 mjr  mjr  16384 Aug  2 01:48 urls.idx
-rw-r--r--   1 mjr  mjr   5462 Aug  2 01:48 urls.rec
-rw-r--r--   1 mjr  mjr  16384 Aug  2 01:48 urls.upd
mjr@lyra-> retail /var/www/logs/access_log | ex_urls.pl | nbs -d urls
/hackthisbox
mjr@lyra->
```

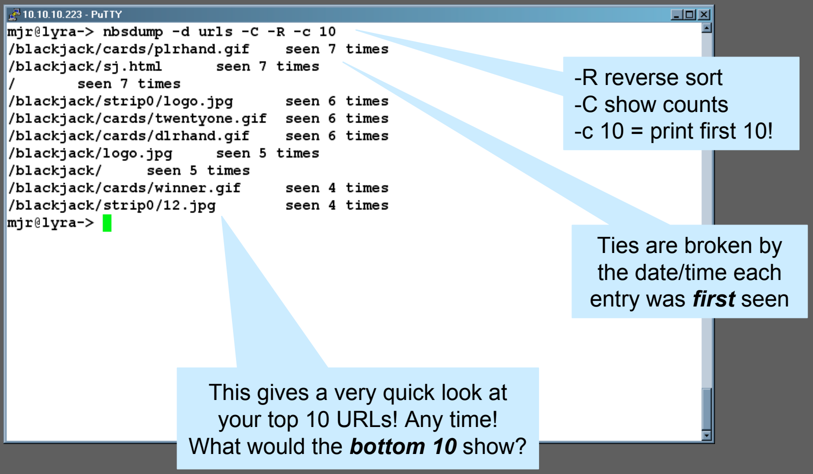
Retail keeps track of your offset in the log file and outputs only new additions
Run from a cron job
An NBS URL!
159

The retail program is extremely useful for passing data into an nbs database. Basically, it is a stateful "tail" that dumps whatever has been added to a log file since the last time it was called. This is useful since it gets you out of having to keep the log monitoring process constantly running. In this example you might put the command line as used right into a cron job to run every 15 minutes.

In the example above, I "trained" the nbs database by running it once with my log records. I sent the output to "baseline" with the "-o baseline" option, and then threw the baseline file away once the database was constructed. After the baseline is built, any new requests that come in will appear as NBS!

For the sake of making this an interesting viewgraph, I went in another window and opened a browser to my web site and entered a URL for a document that does not exist. You can see that it came out the back of NBS on the next run, just like it was supposed to!

## Statistics from NBS



```
mjr@lyra-> nbsdump -d urls -C -R -c 10
/blackjack/cards/plrhand.gif    seen 7 times
/blackjack/sj.html      seen 7 times
/       seen 7 times
/blackjack/strip0/logo.jpg      seen 6 times
/blackjack/cards/twentyone.gif  seen 6 times
/blackjack/cards/dlrhand.gif    seen 6 times
/blackjack/logo.jpg     seen 5 times
/blackjack/     seen 5 times
/blackjack/cards/winner.gif     seen 4 times
/blackjack/strip0/12.jpg        seen 4 times
mjr@lyra->
```

-R reverse sort
-C show counts
-c 10 = print first 10!

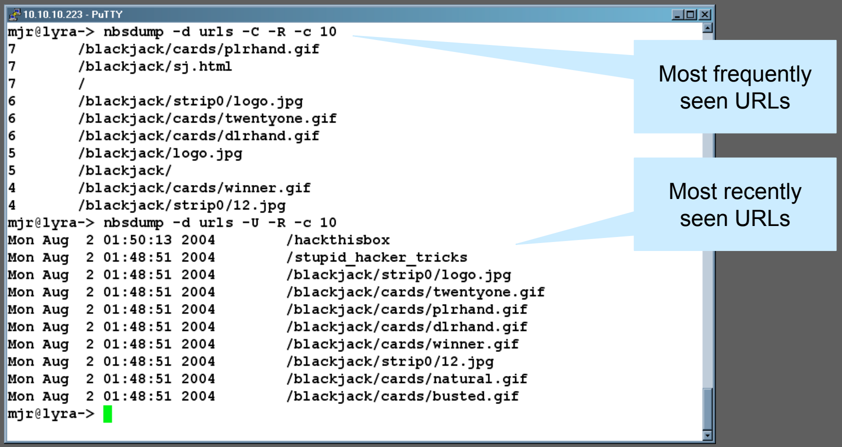
Ties are broken by the date/time each entry was ***first*** seen

This gives a very quick look at your top 10 URLs! Any time! What would the ***bottom 10*** show?

160

Nbsdump is also very useful for collecting statistics about arbitrary things. Because the data is all stored sorted in a Btree it's extremely quick if you want to retrieve "top 20" or "bottom whatever" values. In this example, we use nbsdump to retrieve counts (-C) in reverse order (-R) highest-to-lowest and to print the first 10 (-c 10) values.

## Statistics from NBS

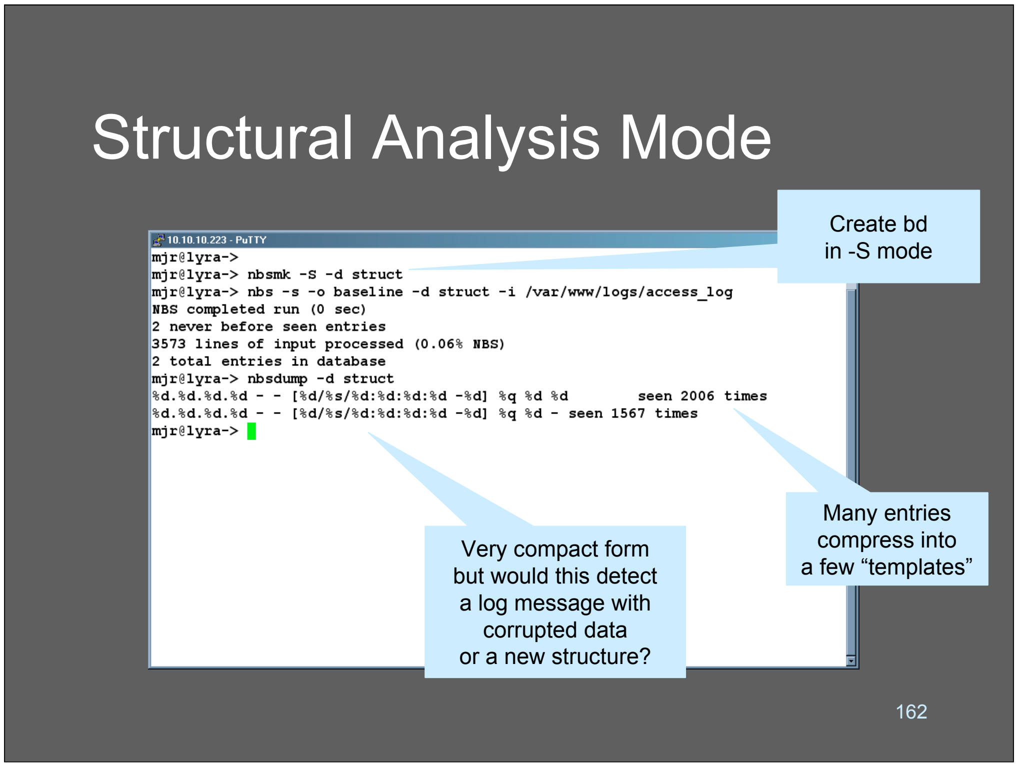


```
mjr@lyra-> nbsdump -d urls -C -R -c 10
7       /blackjack/cards/plrhand.gif
7       /blackjack/sj.html
7       /
6       /blackjack/strip0/logo.jpg
6       /blackjack/cards/twentyone.gif
6       /blackjack/cards/dlrhand.gif
5       /blackjack/logo.jpg
5       /blackjack/
4       /blackjack/cards/winner.gif
4       /blackjack/strip0/12.jpg
mjr@lyra-> nbsdump -d urls -U -R -c 10
Mon Aug  2 01:50:13 2004        /hackthisbox
Mon Aug  2 01:48:51 2004        /stupid_hacker_tricks
Mon Aug  2 01:48:51 2004        /blackjack/strip0/logo.jpg
Mon Aug  2 01:48:51 2004        /blackjack/cards/twentyone.gif
Mon Aug  2 01:48:51 2004        /blackjack/cards/plrhand.gif
Mon Aug  2 01:48:51 2004        /blackjack/cards/dlrhand.gif
Mon Aug  2 01:48:51 2004        /blackjack/cards/winner.gif
Mon Aug  2 01:48:51 2004        /blackjack/strip0/12.jpg
Mon Aug  2 01:48:51 2004        /blackjack/cards/natural.gif
Mon Aug  2 01:48:51 2004        /blackjack/cards/busted.gif
mjr@lyra->
```

161

Depending on the options you give nbsdump, you can get a variety of forms of output. Generally, the output is sorted by whichever key you asked it to be retrieved with. In the first example, we asked it to dump counts (-C) in reverse order (-R), so we got a sorted frequency chart. In the second example we asked it to dump in order of date and time, so we got the data back as most recently seen URLs.

Whenever nbs is sorting data if there are duplicate entries, it sorts based on which ones entered the database *first*.



## Structural Analysis Mode

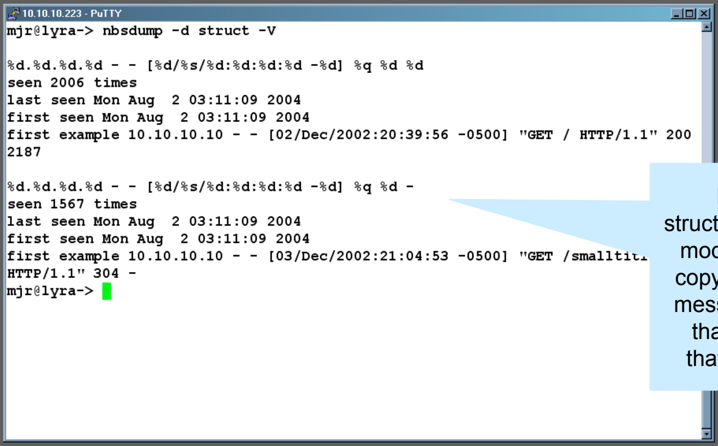
```
10.10.10.223 - PuTTY
mjr@lyra->
mjr@lyra-> nbsmk -S -d struct
mjr@lyra-> nbs -s -o baseline -d struct -i /var/www/logs/access_log
NBS completed run (0 sec)
2 never before seen entries
3573 lines of input processed (0.06% NBS)
2 total entries in database
mjr@lyra-> nbsdump -d struct
%d.%d.%d.%d - - [%d/%s/%d:%d:%d:%d -%d] %q %d %d        seen 2006 times
%d.%d.%d.%d - - [%d/%s/%d:%d:%d:%d -%d] %q %d - seen 1567 times
mjr@lyra->
```

Nbs also includes an experimental capability called *structural analysis mode*. In this mode, what nbs does is creates a database of "templates" that represent the structure of a message with all the potentially variable fields knocked out into %s-style parameter strings. The variable fields are then thrown away ad al that is stored is the structural templates.

The call to nbsmk -S tells it to create the database in structural analysis mode. Then we send my web server access logs into it. Here you see that my entire web server logs reduce to 2 templates!

In this case, this particular rule would not be super interesting because the templates compress down to mostly a quoted string (see the "%q" in there?) If we wanted to do more effective watching of just templates, we'd look only at the URLs as non-quoted strings. The potential transforms are limited only by your imagination!

## Show Examples of Templates

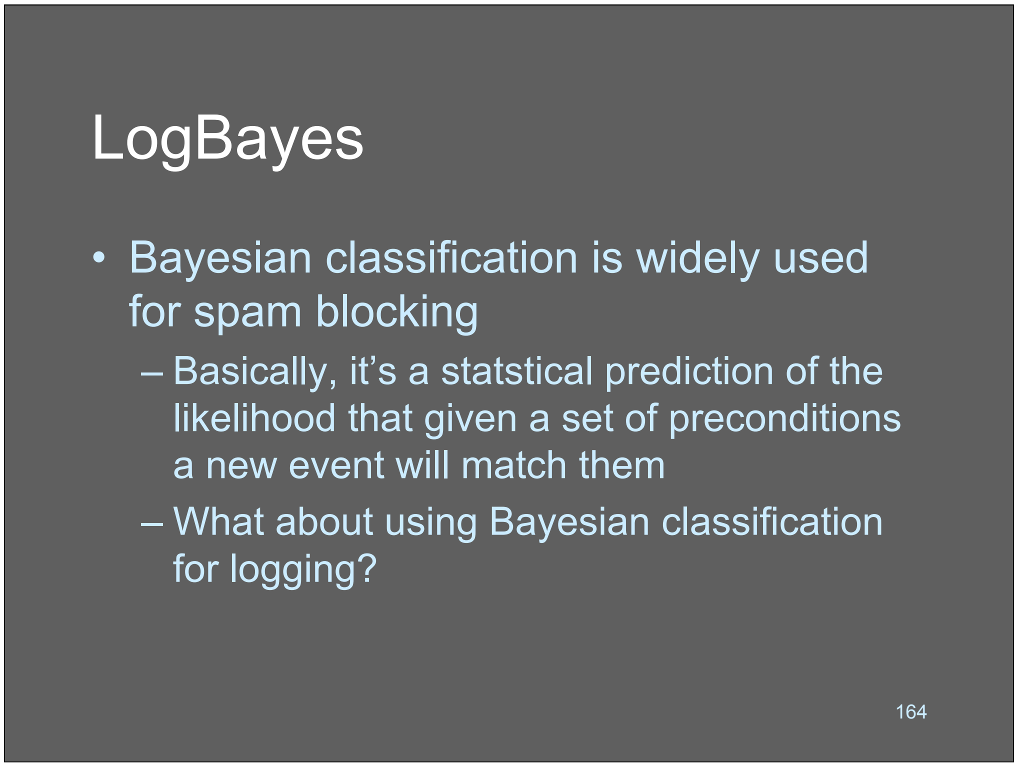


```
mjr@lyra-> nbsdump -d struct -V

%d.%d.%d.%d - - [%d/%s/%d:%d:%d:%d -%d] %q %d %d
seen 2006 times
last seen Mon Aug  2 03:11:09 2004
first seen Mon Aug  2 03:11:09 2004
first example 10.10.10.10 - - [02/Dec/2002:20:39:56 -0500] "GET / HTTP/1.1" 200
2187

%d.%d.%d.%d - - [%d/%s/%d:%d:%d:%d -%d] %q %d -
seen 1567 times
last seen Mon Aug  2 03:11:09 2004
first seen Mon Aug  2 03:11:09 2004
first example 10.10.10.10 - - [03/Dec/2002:21:04:53 -0500] "GET /smalltit
HTTP/1.1" 304 -
mjr@lyra->
```

NBS in structure analysis mode stores a copy of the first message it saw that created that template

163

If you dump a structural analysis mode database in verbose mode, it will print out the first example it saw of an entry that had that structure. It’s potentially deceptive, though, since simple structures (e.g.: “%s.%s”) may match a lot of things!



## LogBayes

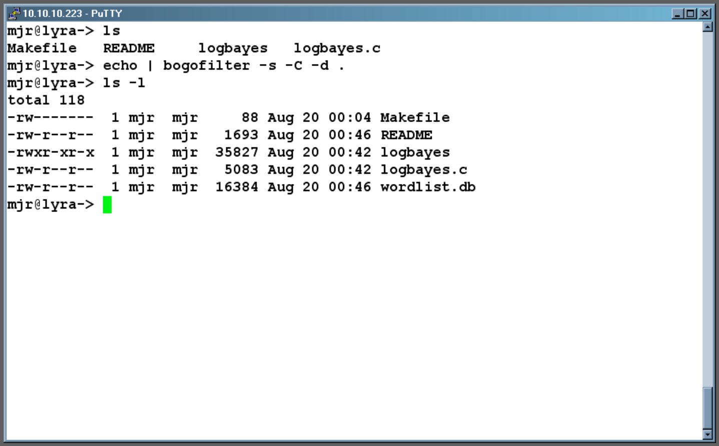
- Bayesian classification is widely used for spam blocking
  - Basically, it’s a statstical prediction of the likelihood that given a set of preconditions a new event will match them
  - What about using Bayesian classification for logging?

164

Word-weighting statistics and Bayesian classifiers are yielding spectacular results in the spam-blocking arena. The spammers are finding clever ways around them, but -system log messages don’t even try.

Logbayes is a poorly-implemented proof-of-concept hack that harnesses a bayesian classifier into a logging stream. It also experiments with an idea that is near and dear to my heart: managing the workflow of assessing messages. Logbayes has a rapid training mode and can be run permanently in training mode if you want it to be. Basically it learns what tokens you don’t care about in your logs. The results can be surprising.

## LogBayes in action

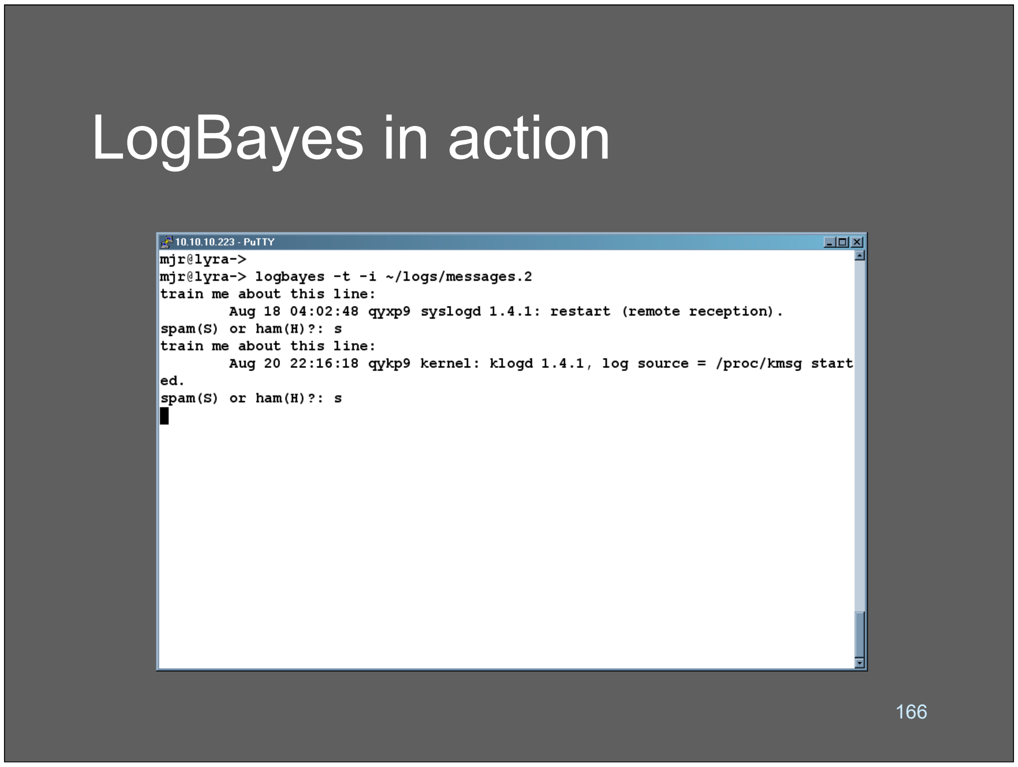


```
mjr@lyra-> ls
Makefile   README     logbayes   logbayes.c
mjr@lyra-> echo | bogofilter -s -C -d .
mjr@lyra-> ls -l
total 118
-rw-------  1 mjr  mjr     88 Aug 20 00:04 Makefile
-rw-r--r--  1 mjr  mjr   1693 Aug 20 00:46 README
-rwxr-xr-x  1 mjr  mjr  35827 Aug 20 00:42 logbayes
-rw-r--r--  1 mjr  mjr   5083 Aug 20 00:42 logbayes.c
-rw-r--r--  1 mjr  mjr  16384 Aug 20 00:46 wordlist.db
mjr@lyra->
```

165

In this slide we see logbayes going into a training run. The program is given an input file, which it processes line-by-line. When it encounters a message that falls below 90% likelihood of being garbage it asks the user, then continues.

Processing interactively like this is kind of strange. At first the messages come frequently but then the rate at which questions are asked begins to tail off rapidly.



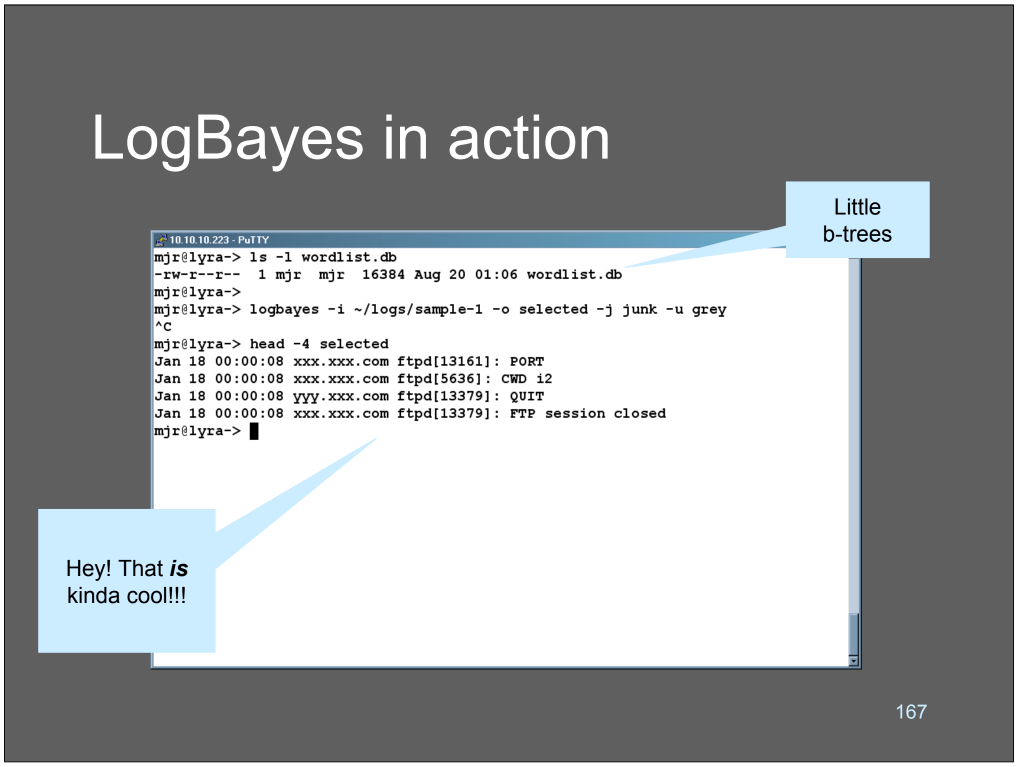
## LogBayes in action

```
mjr@lyra->
mjr@lyra-> logbayes -t -i ~/logs/messages.2
train me about this line:
        Aug 18 04:02:48 qyxp9 syslogd 1.4.1: restart (remote reception).
spam(S) or ham(H)?: s
train me about this line:
        Aug 20 22:16:18 qykp9 kernel: klogd 1.4.1, log source = /proc/kmsg start
ed.
spam(S) or ham(H)?: s
```

166

In this slide we see logbayes going into a training run. The program is given an input file, which it processes line-by-line. When it encounters a message that falls below 90% likelihood of being garbage it asks the user, then continues.

Processing interactively like this is kind of strange. At first the messages come frequently but then the rate at which questions are asked begins to tail off rapidly.



## LogBayes in action

```
mjr@lyra-> ls -l wordlist.db
-rw-r--r--  1 mjr  mjr  16384 Aug 20 01:06 wordlist.db
mjr@lyra->
mjr@lyra-> logbayes -i ~/logs/sample-1 -o selected -j junk -u grey
^C
mjr@lyra-> head -4 selected
Jan 18 00:00:08 xxx.xxx.com ftpd[13161]: PORT
Jan 18 00:00:08 xxx.xxx.com ftpd[5636]: CWD i2
Jan 18 00:00:08 yyy.xxx.com ftpd[13379]: QUIT
Jan 18 00:00:08 xxx.xxx.com ftpd[13379]: FTP session closed
mjr@lyra->
```

Here we see the resulting keyword databases that are built by bogofilter. One thing that kind of leaps out is that they are fairly small. I don’t know what difference that makes except that it’s kind of cool.

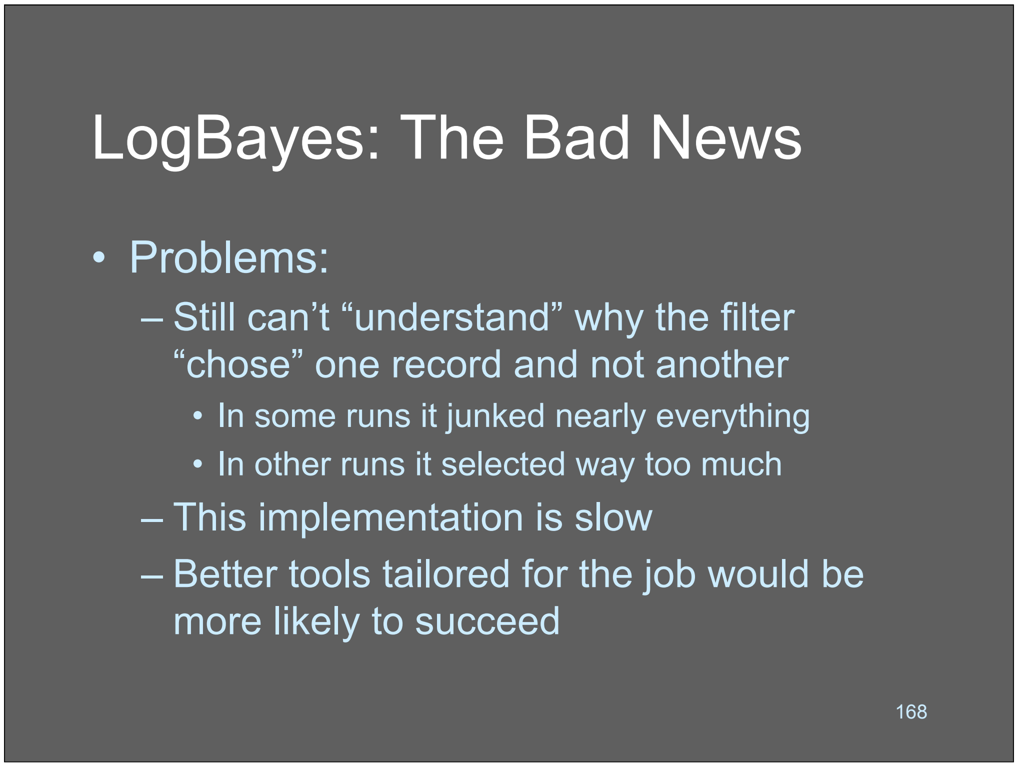
Next we run the data through logbayes to separate it into 3 files

-o  stuff we have selected as interesting

-j stuff we have selected as junk

-g stuff we are unsure of

There are cutoffs in the source code that allow you to tune the sensitivity of logbayes to the results provided from bogofilter.



## LogBayes: The Bad News

- Problems:
  - Still can’t “understand” why the filter “chose” one record and not another
    - In some runs it junked nearly everything
    - In other runs it selected way too much
  - This implementation is slow
  - Better tools tailored for the job would be more likely to succeed

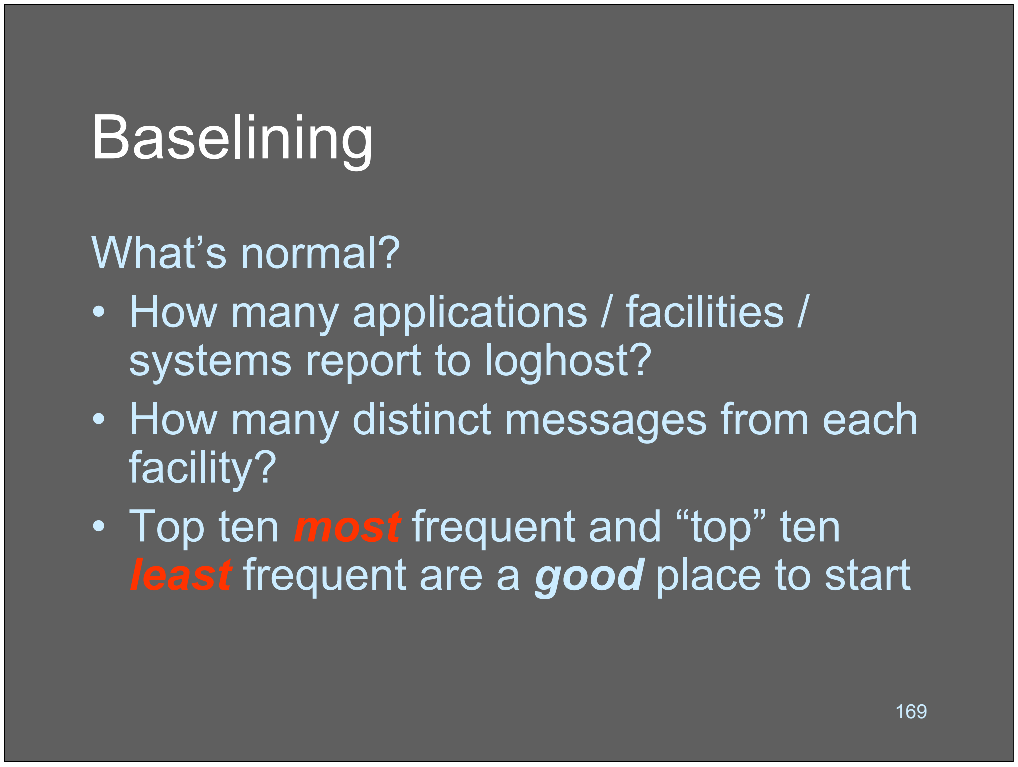
168

One problem with this approach is that - even if it works - how can you be *sure* it is working? You need to go back and look at the logs or run extensive tests.

This implementation is extremely ugly and somewhat unreliable, since it depends deeply on side-effects of normal operation for bogofilter. If bogofilter’s author changes anything about how it outputs its messages, logbayes will break.

The way logbayes operates is slow because bogofilter is very Email-oriented. It wants to operate on messages, not lines of text. To deal with that, logbayes writes *each line* of the log to a file, calls bogofilter on it, then calls it again in training mode. It’ll really light your hard disk up - this is experimental code!

I think the idea has merit, though! It might make a good research topic!



## Baselining

What's normal?

- How many applications / facilities / systems report to loghost?
- How many distinct messages from each facility?
- Top ten ***most*** frequent and "top" ten ***least*** frequent are a ***good*** place to start

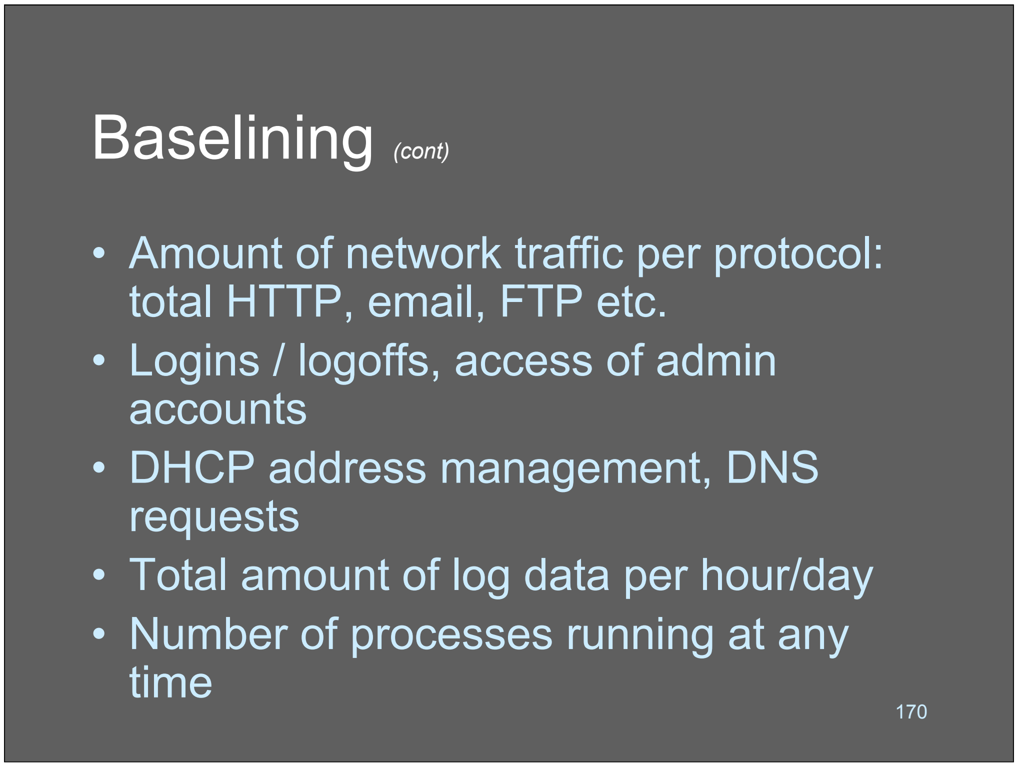
169

Baselining is the process of figuring out what's normal - so that you can look for what does not fit the baseline.

The problem with baselining is that it's a creative process and is site-dependent. The differences between the baselines for a hospital network and a university network (should be) extreme.

So, how do you build a baseline? By hand. Some of the things to look at are simply the reporting rates of various systems. Looking at "top ten" lists or "bottom ten" lists are two good places to start. But don't stop there - look at not only the rates at which things happen; look for the rate at which those rates *change*. If you're seeing 10 machines reporting 2,000 events/day, it's interesting if that number jumps 20% in the course of a week - regardless of which systems in particular are doing any given thing.

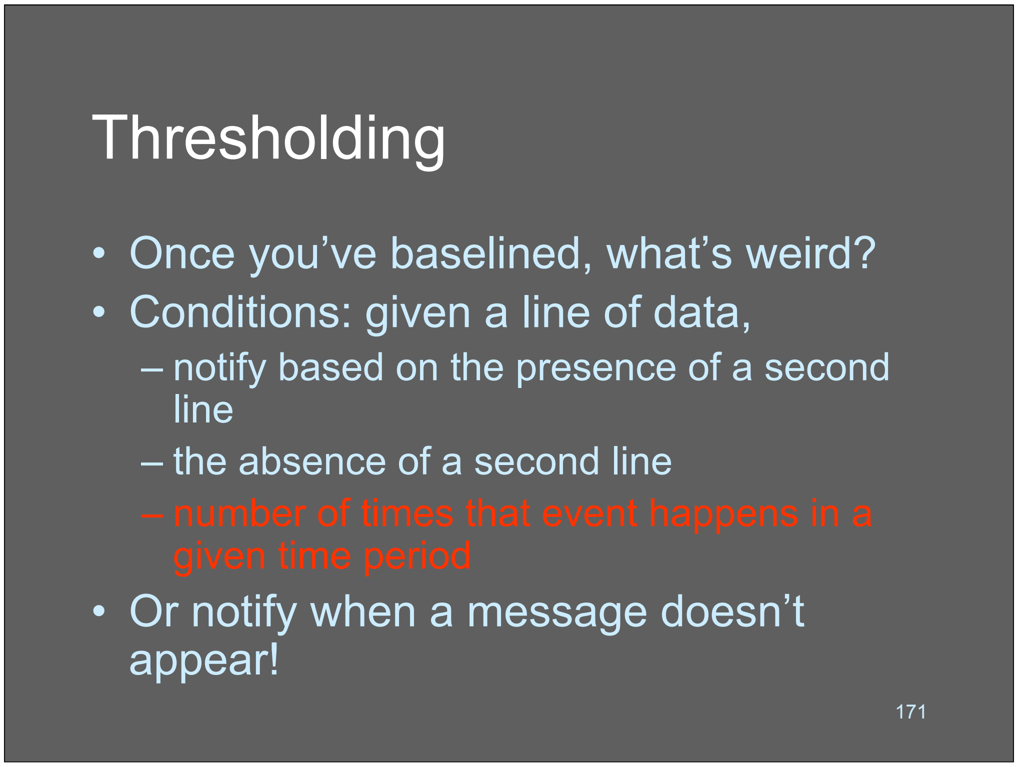
To generate a baseline, start collecting "speeds and feeds" for anything you can think of on your network. It's all potentially valuable.


## Baselining *(cont)*

- Amount of network traffic per protocol: total HTTP, email, FTP etc.
- Logins / logoffs, access of admin accounts
- DHCP address management, DNS requests
- Total amount of log data per hour/day
- Number of processes running at any time

170

These are a few possibly interesting values you might want to baseline. The possibilities are endless.



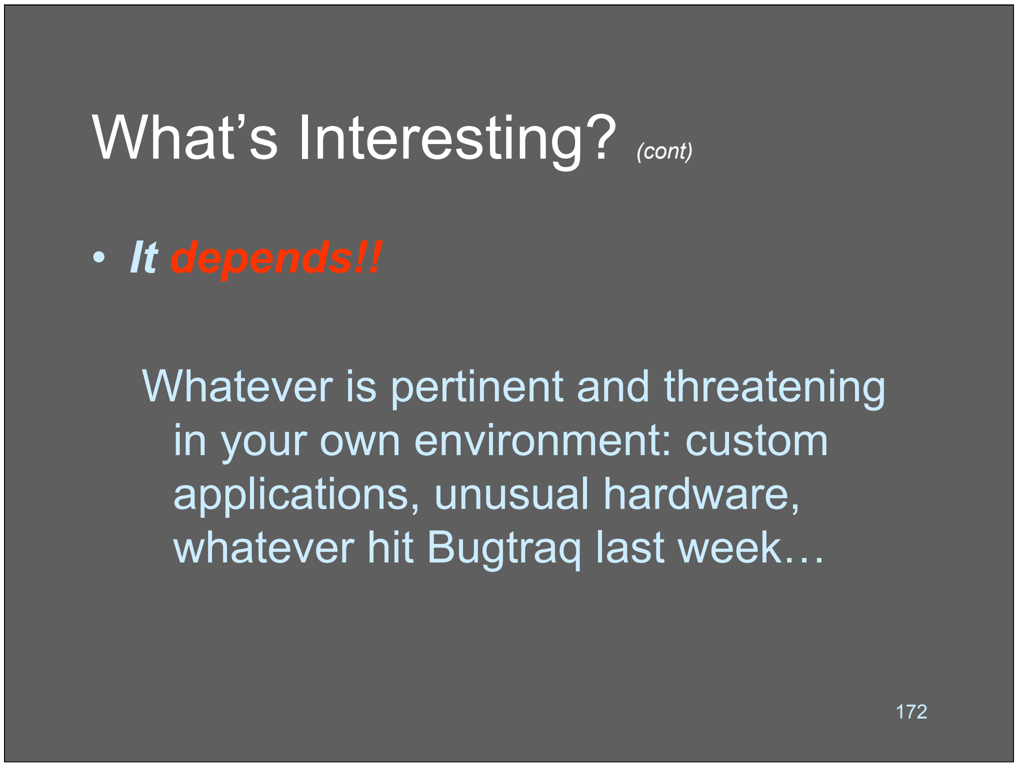
## Thresholding

- Once you've baselined, what's weird?
- Conditions: given a line of data,
  - notify based on the presence of a second line
  - the absence of a second line
  - number of times that event happens in a given time period
- Or notify when a message doesn't appear!

171

Once you have established baselines, you can start looking for variations around that baseline. Generally, you will want to look for variations *above* the baseline but sometimes variations below the baseline are also interesting.Almost always, thresholds will be dealing with the number of times an event happens.

Consider thresholding and reporting to be aspects of the same problem. Most log analysts combine the two functions into the same processing routines.



## What's Interesting? *(cont)*

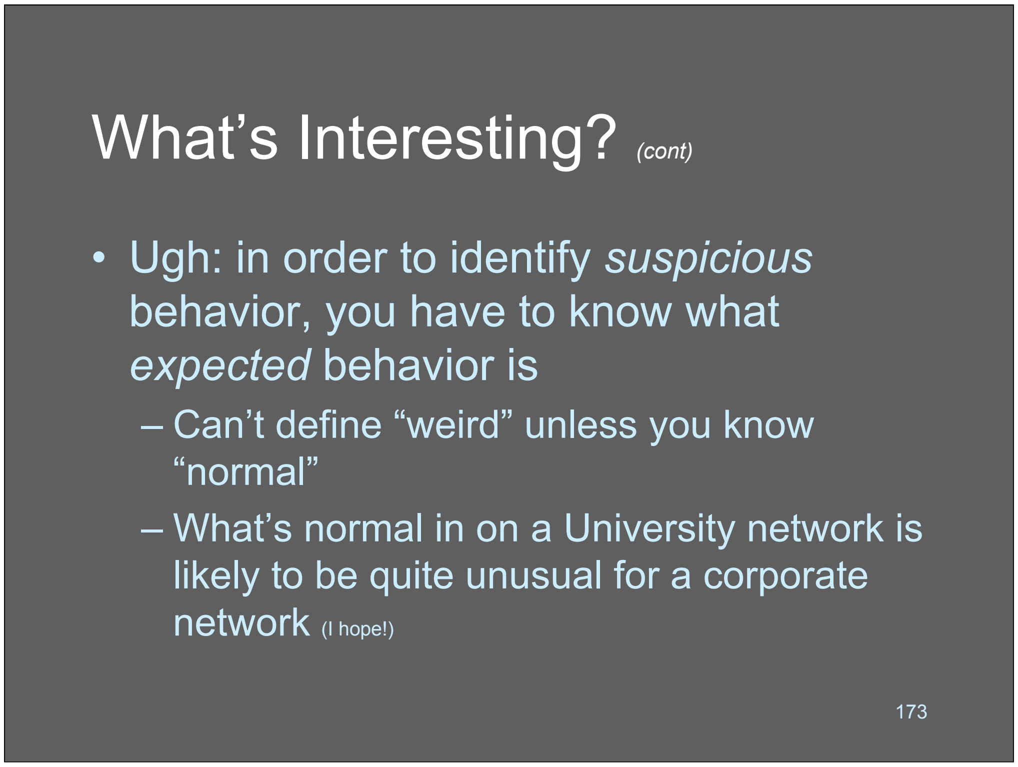
- ***It depends!!***

  Whatever is pertinent and threatening in your own environment: custom applications, unusual hardware, whatever hit Bugtraq last week…

172

Useful reference: *Identify data that characterize systems and aid in detecting signs of suspicious behavior*

http://www.cert.org/security-improvements/practices/p091.html

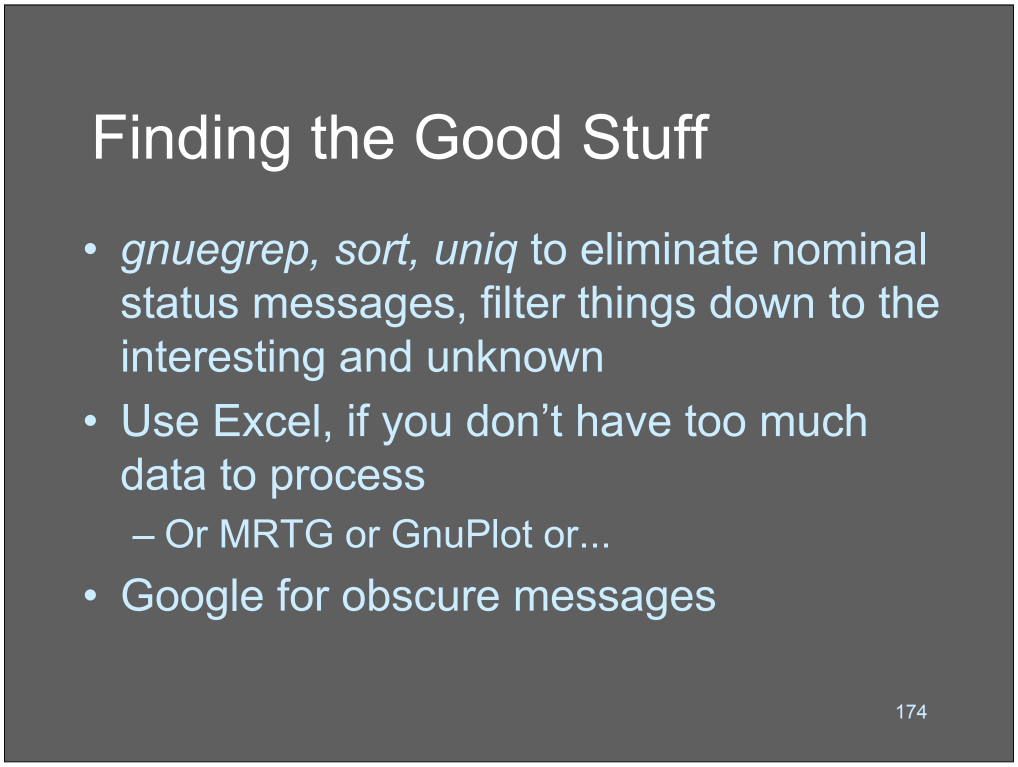


## What's Interesting? *(cont)*

- Ugh: in order to identify *suspicious* behavior, you have to know what *expected* behavior is
  - Can't define "weird" unless you know "normal"
  - What's normal in on a University network is likely to be quite unusual for a corporate network (I hope!)

173

Have you ever noticed how virtually every security problem comes (eventually) down to policy? This is another of those cases!! 99% of your ability to identify suspicious log entries will depend on you're a priori knowledge of what is normal for your systems. If you don't know that, you'll wind up having to learn - it's as simple as that.



## Finding the Good Stuff

- *gnuegrep, sort, uniq* to eliminate nominal status messages, filter things down to the interesting and unknown
- Use Excel, if you don't have too much data to process
  - Or MRTG or GnuPlot or...
- Google for obscure messages

174

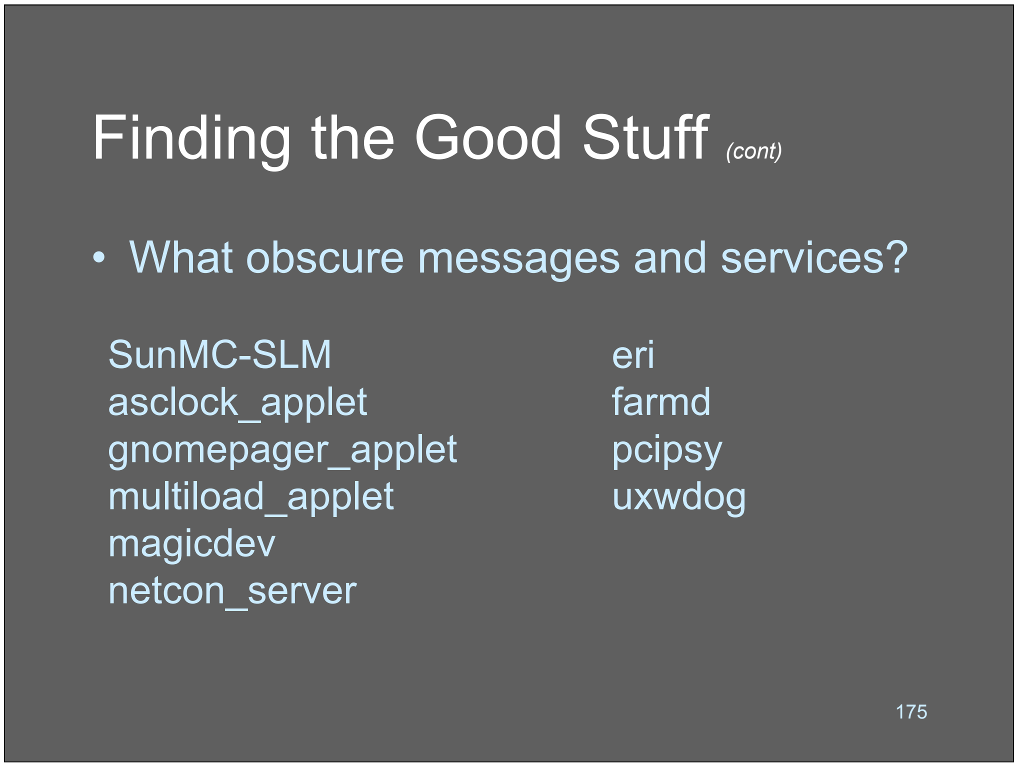
Detecting outlying data is also a process of mapping what is seen against what is known to be OK. Log analuysts use whatever tools they are most comfortable with. The high-level view of the process is one of:

•Sort

•Evaluate

•Resarch

•Respond



## Finding the Good Stuff *(cont)*

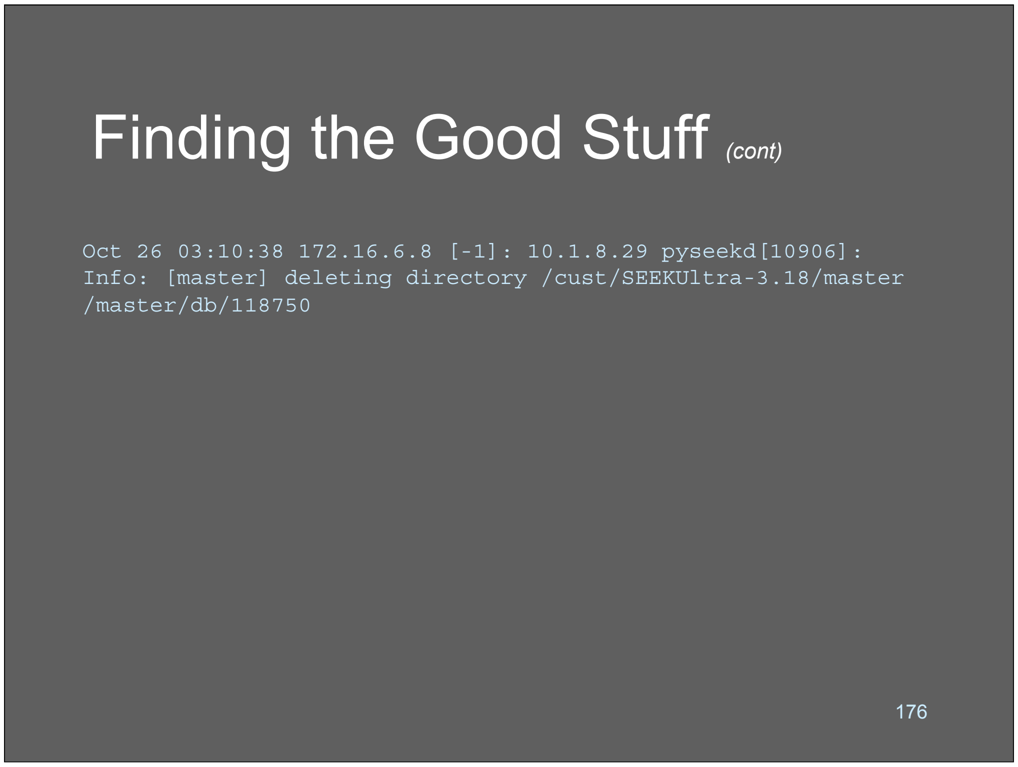
- What obscure messages and services?

| | |
|---|---|
| SunMC-SLM | eri |
| asclock_applet | farmd |
| gnomepager_applet | pcipsy |
| multiload_applet | uxwdog |
| magicdev | |
| netcon_server | |

175

We need all the magicdev's we can get.

What are these various doo-dads? I don't know all of them myself but you're running some of them.



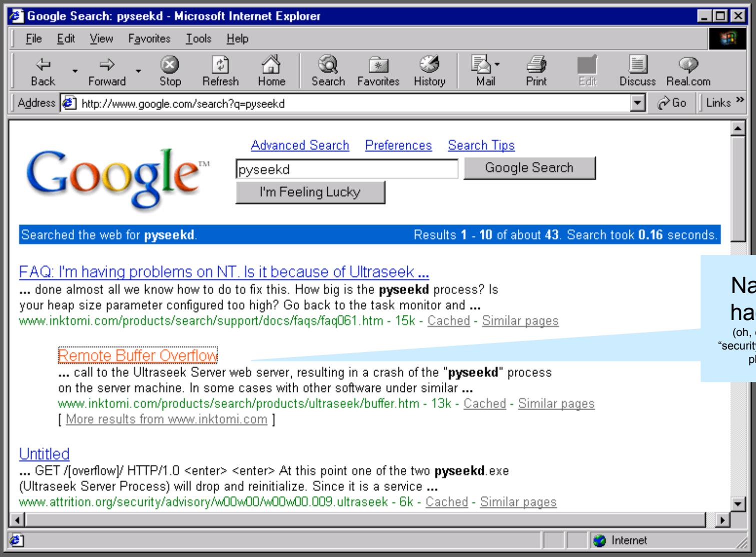
## Finding the Good Stuff *(cont)*

```
Oct 26 03:10:38 172.16.6.8 [-1]: 10.1.8.29 pyseekd[10906]:
Info: [master] deleting directory /cust/SEEKUltra-3.18/master
/master/db/118750
```

176

Deleting directories seems like a good item to investigate.

## Finding the Good Stuff cont.


Google Search: pyseekd - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Real.com
Address http://www.google.com/search?q=pyseekd
Go Links
Advanced Search Preferences Search Tips
pyseekd
Google Search
I'm Feeling Lucky
Internet

Searched the web for **pyseekd**. Results **1** - **10** of about **43**. Search took **0.16** seconds.

FAQ: I'm having problems on NT. Is it because of Ultraseek ...
**...** done almost all we know how to do to fix this. How big is the **pyseekd** process? Is your heap size parameter configured too high? Go back to the task monitor and **...**
www.inktomi.com/products/search/support/docs/faqs/faq061.htm - 15k - Cached - Similar pages

Remote Buffer Overflow
**...** call to the Ultraseek Server web server, resulting in a crash of the "**pyseekd**" process on the server machine. In some cases with other software under similar **...**
www.inktomi.com/products/search/products/ultraseek/buffer.htm - 13k - Cached - Similar pages
[ More results from www.inktomi.com ]

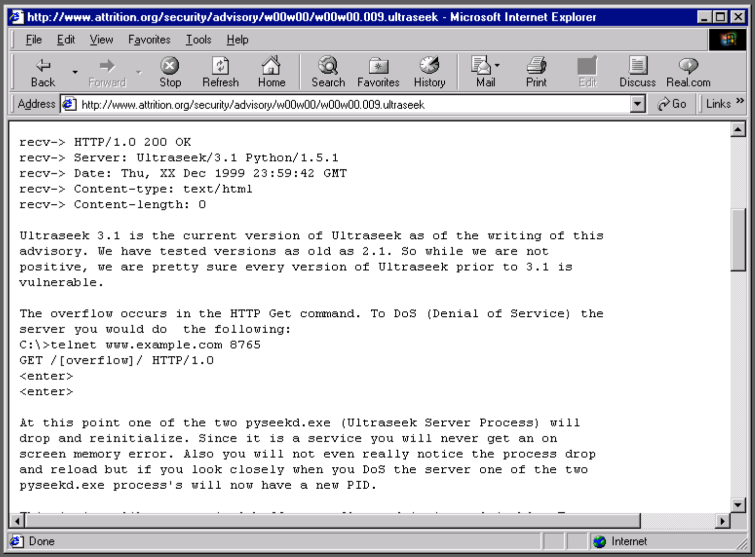
Untitled
**...** GET /[overflow]/ HTTP/1.0 <enter> <enter> At this point one of the two **pyseekd**.exe (Ultraseek Server Process) will drop and reinitialize. Since it is a service **...**
www.attrition.org/security/advisory/w00w00/w00w00.009.ultraseek - 6k - Cached - Similar pages

Naughty hackers!
(oh, excuse me, "security researchers" please…)

177

A bit of hard-core investigation on the Internet reveals that *pyseekd* is the primary binary use by the Inktomi suite of search engines and e-commerce support applications. Although we haven't tracked down the particular meaning of the message, we have some context now. It's probably reasonable for a search engine to create and delete dynamic directories.

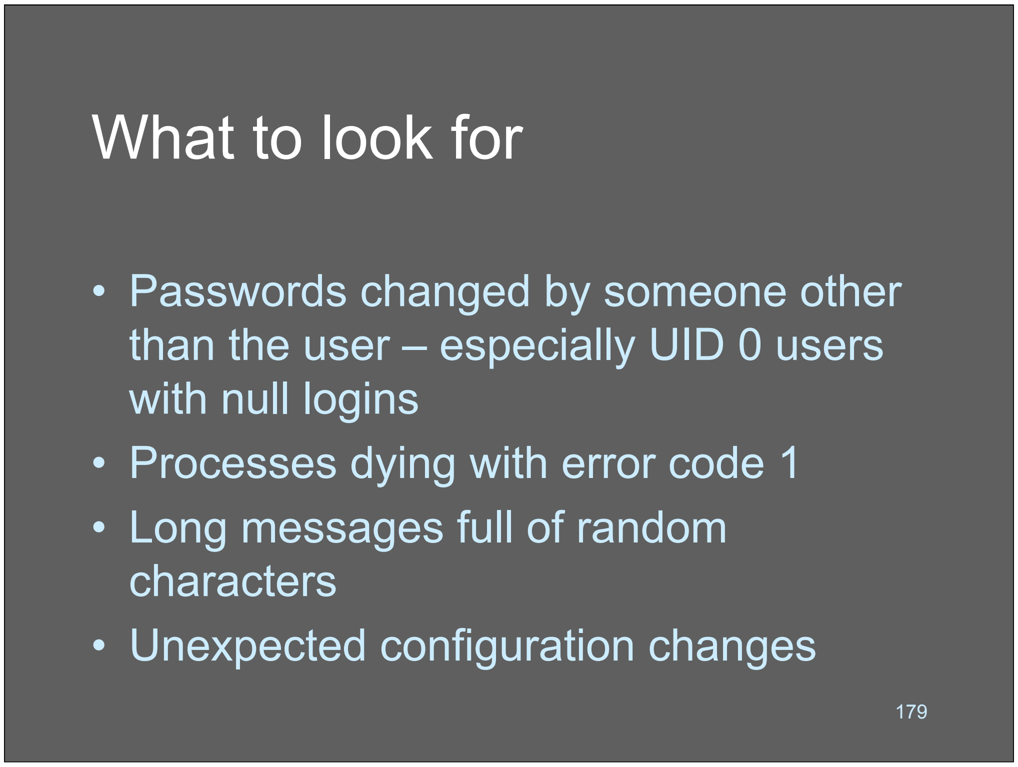
## Finding the Good Stuff *(cont)*


http://www.attrition.org/security/advisory/w00w00/w00w00.009.ultraseek - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Real.com
Address http://www.attrition.org/security/advisory/w00w00/w00w00.009.ultraseek
Go Links
Done
Internet

```
recv-> HTTP/1.0 200 OK
recv-> Server: Ultraseek/3.1 Python/1.5.1
recv-> Date: Thu, XX Dec 1999 23:59:42 GMT
recv-> Content-type: text/html
recv-> Content-length: 0

Ultraseek 3.1 is the current version of Ultraseek as of the writing of this
advisory. We have tested versions as old as 2.1. So while we are not
positive, we are pretty sure every version of Ultraseek prior to 3.1 is
vulnerable.

The overflow occurs in the HTTP Get command. To DoS (Denial of Service) the
server you would do  the following:
C:\>telnet www.example.com 8765
GET /[overflow]/ HTTP/1.0
<enter>
<enter>

At this point one of the two pyseekd.exe (Ultraseek Server Process) will
drop and reinitialize. Since it is a service you will never get an on
screen memory error. Also you will not even really notice the process drop
and reload but if you look closely when you DoS the server one of the two
pyseekd.exe process's will now have a new PID.
```

178

Internet serendipity also reveals that *pyseekd* is vulnerable to a potential buffer overflow.  That’s an extremely useful bit of intelligence – and a guide for further log reconnaissance.



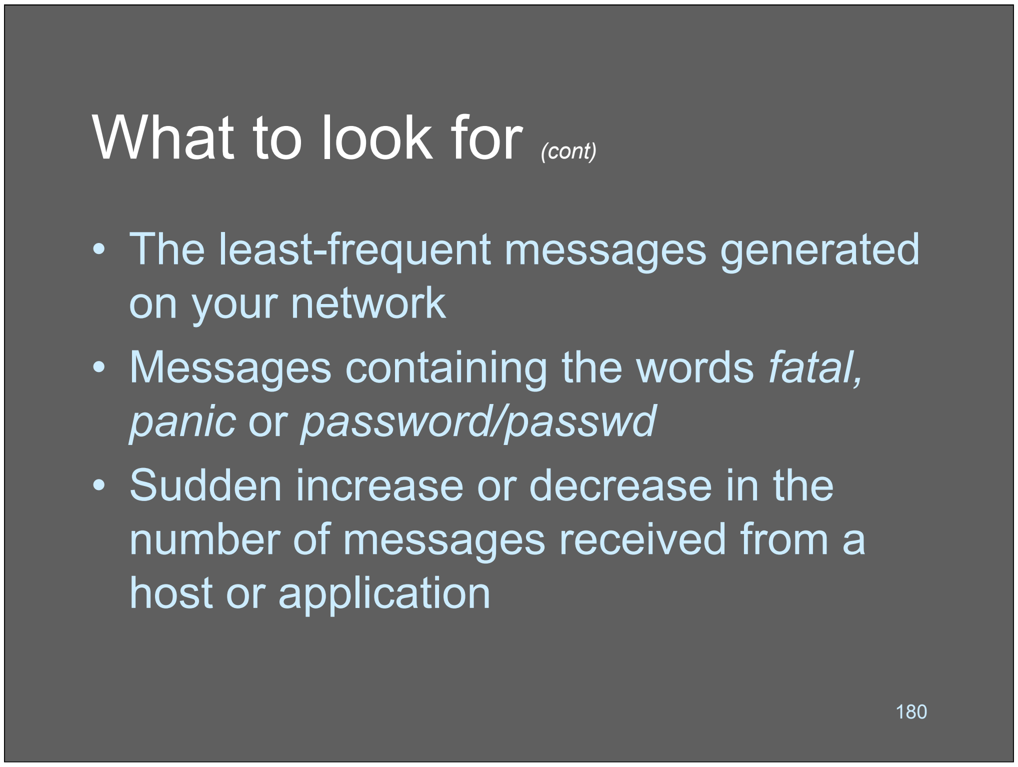
## What to look for

- Passwords changed by someone other than the user – especially UID 0 users with null logins
- Processes dying with error code 1
- Long messages full of random characters
- Unexpected configuration changes

179

If you have a regularly scheduled maintenance window and someone changes your router configuration at a different time, that probably merits some investigation!

When your log messages contain mixtures of weird characters that they normally wouldn't contain it's a decent chance you're looking at a new worm's tracks, as it's trying to exploit a buffer overrun or some kind of data-stuffing attack. Your results are going to vary - systems that were susceptible to the attack aren't going to log anything, because the attack succeeded - but a lot of Solaris/Apache log analysts catch on fairly quickly whenever there's a new worm mass-exploiting a Windows/IIS hole.

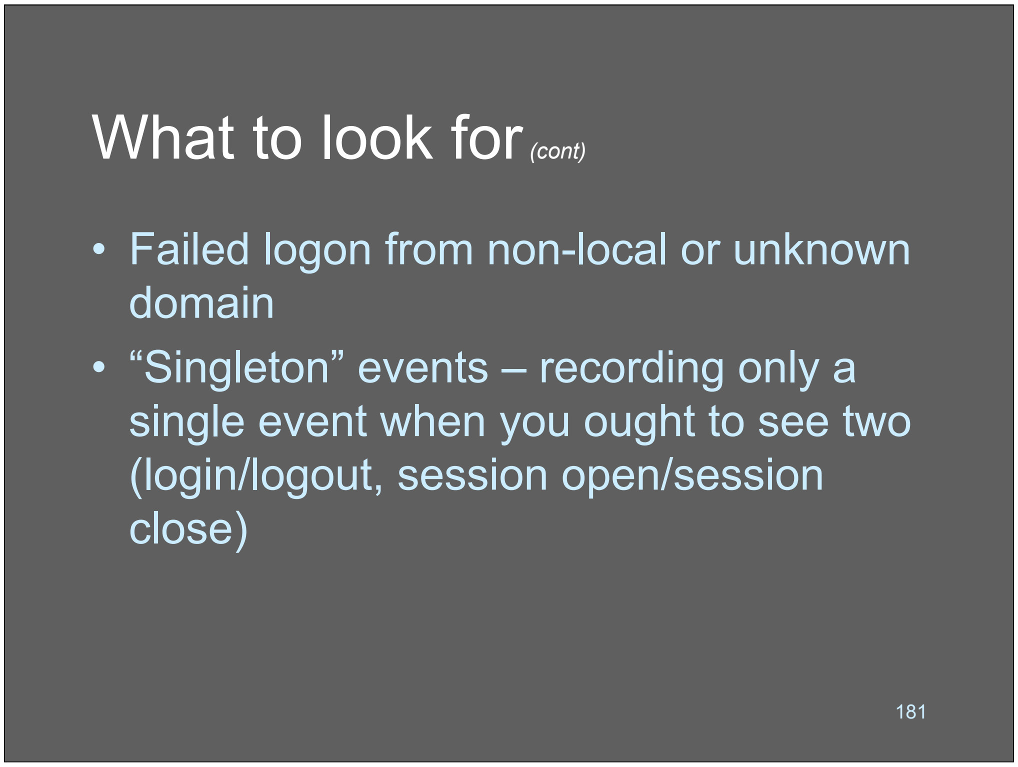


## What to look for *(cont)*

- The least-frequent messages generated on your network
- Messages containing the words *fatal, panic* or *password/passwd*
- Sudden increase or decrease in the number of messages received from a host or application

180

These are a few other things to look for - look for rates of change at the high end of your traffic, and look for introduction of sparse events at the low end of your traffic. Remember Ranum’s law: the number of times an uninteresting thing happens is an interesting thing!

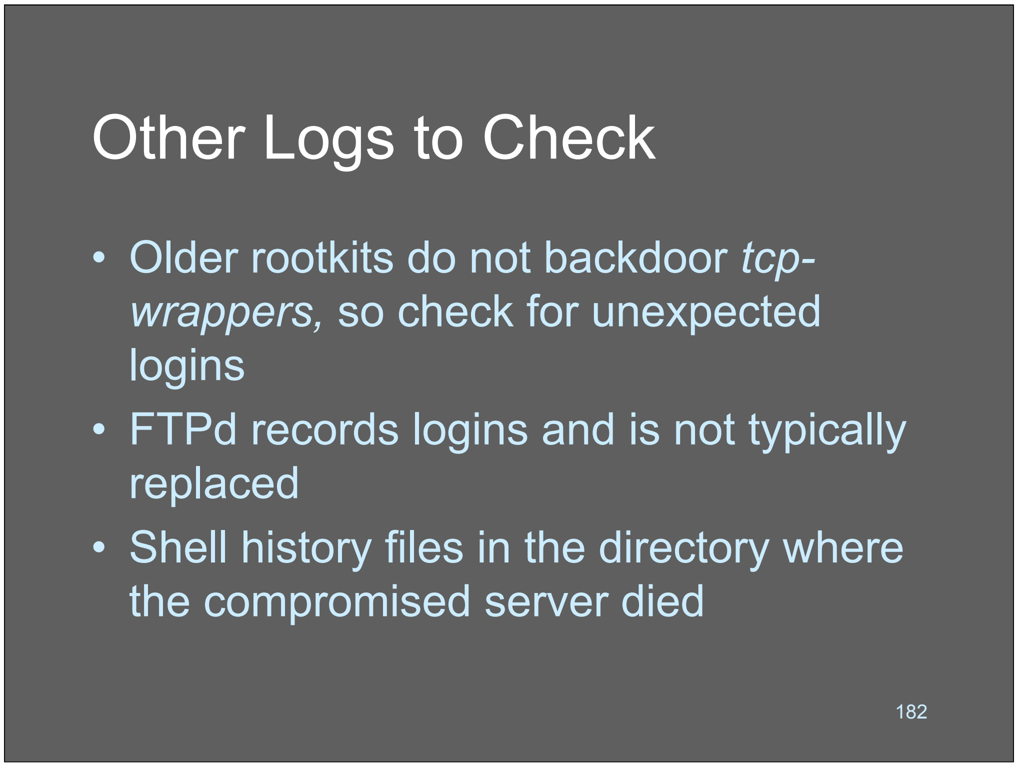


## What to look for *(cont)*

- Failed logon from non-local or unknown domain
- “Singleton” events – recording only a single event when you ought to see two (login/logout, session open/session close)

181

Matching singleton events is a very powerful technique but is not used by many log analysts because there aren’t good tools for doing it. Ben Laurie has a MOD_AUDIT module for Apache which starts a log entry when a request comes in, and completes it when the request has been processed and dispatched. This, way, if you see a start event without a finish, you can be pretty sure something (like a buffer overrun) prevented the process from completing the request. This technique is also called “tomstoning”

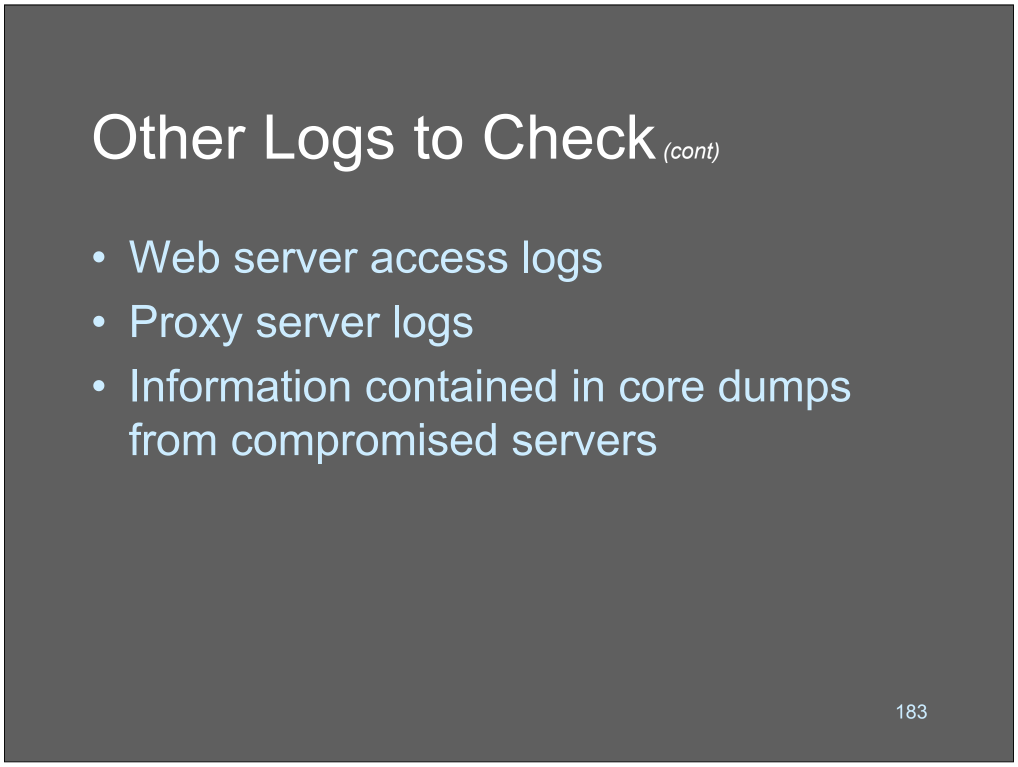

## Other Logs to Check

- Older rootkits do not backdoor *tcp-wrappers,* so check for unexpected logins
- FTPd records logins and is not typically replaced
- Shell history files in the directory where the compromised server died

182

The essay at http://mixter.warrior2k.com/logs.txt is about audit trails that hackers typically forget to remove or modify. So if your preliminary *syslog* data suggests that a system's been compromised, these are good places to check.

If you're running a crucial server, searching for core files is also a good idea.



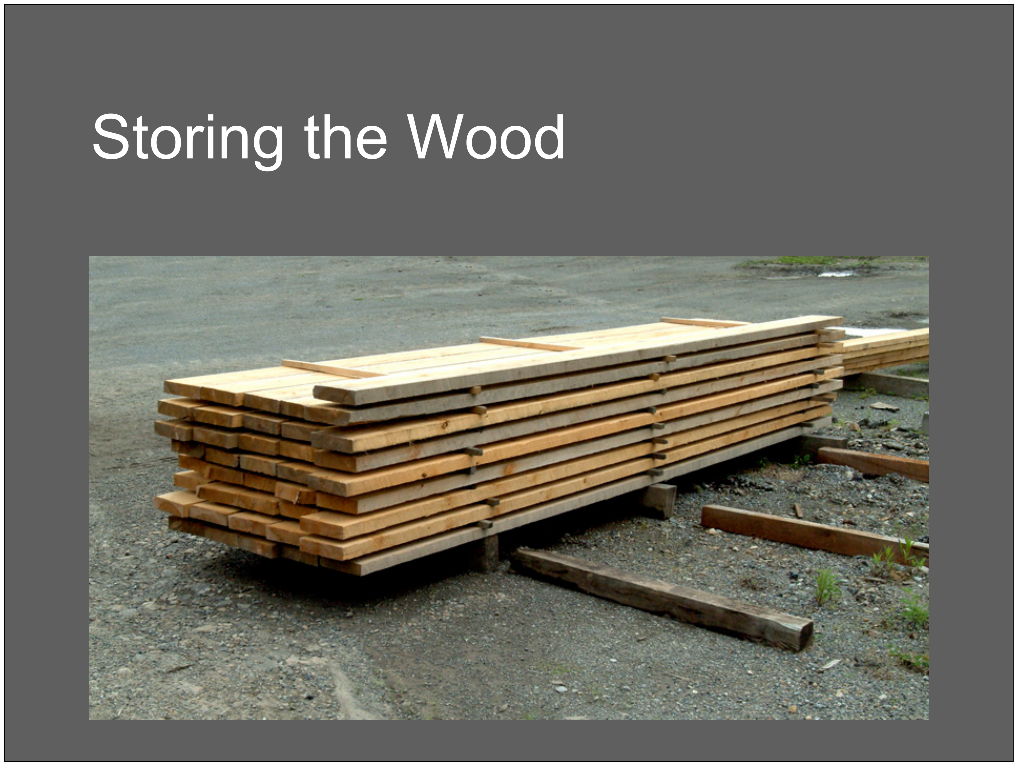
## Other Logs to Check *(cont)*

- Web server access logs
- Proxy server logs
- Information contained in core dumps from compromised servers

183

A lot of sites turn off web server access logs “for performance reasons.” This is kind of like poking your own eyes out “for performance reasons” Don’t do it!

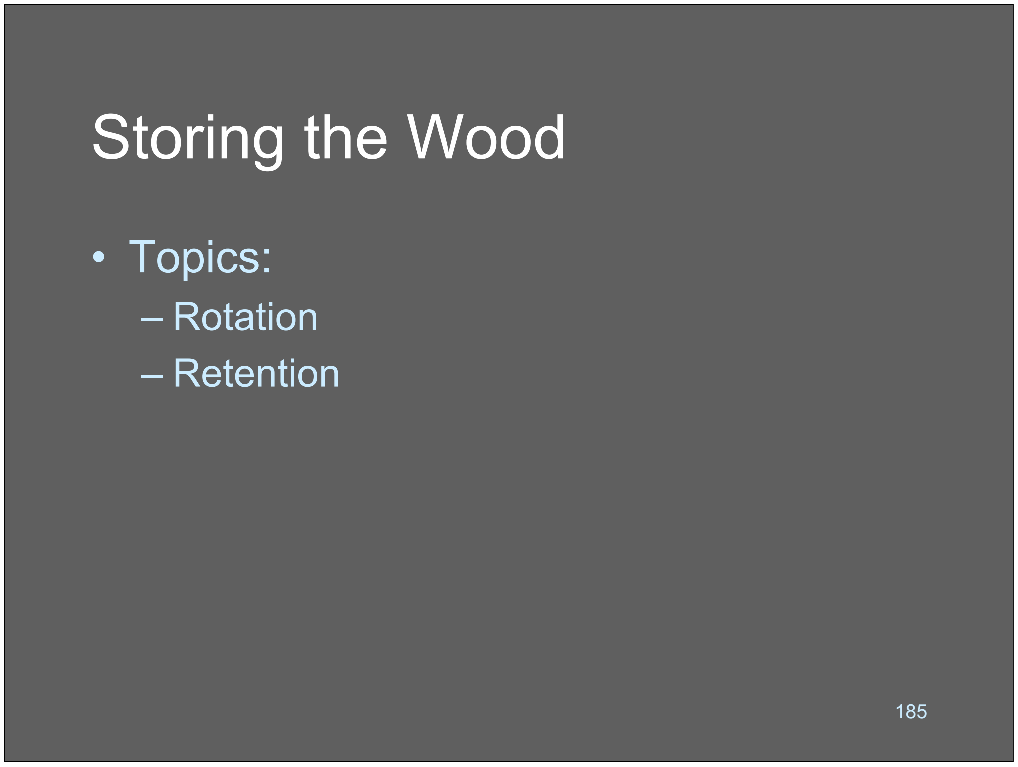
Proxy server logs are also extremely interesting, and may reveal attempts to bounce traffic or to transfer spam. Unusual errors in proxy logs, or unusual usage changes in proxy logs are key indicators of problems. You could probably generate a pretty useful security report simply by plotting the number of accesses/hour over a rolling period (e.g.: wc -l proxylog ) the rate at which your proxy server is used should tend to remain approximately constant.



## Storing the Wood

Sawmills dry their wood the good old-fashioned way: sunlight and time. This is a typical board stack of fence boards, freshly cut. They smell *great* if you like the smell of fresh-cut wood.

When you buy lumber like this at the home improvement store, it has been planed and sometimes sanded, so the surfaces that are exposed to the air have all been cleaned up.

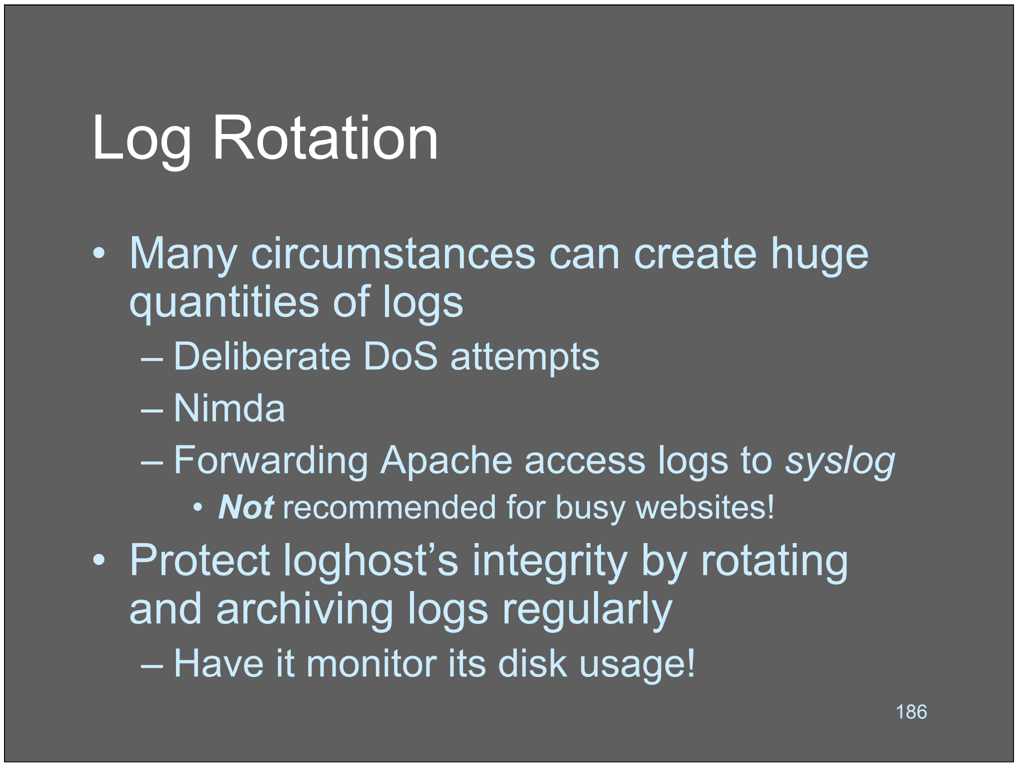


## Storing the Wood

- Topics:
  - Rotation
  - Retention

185

This space intentionally left bereft.



## Log Rotation

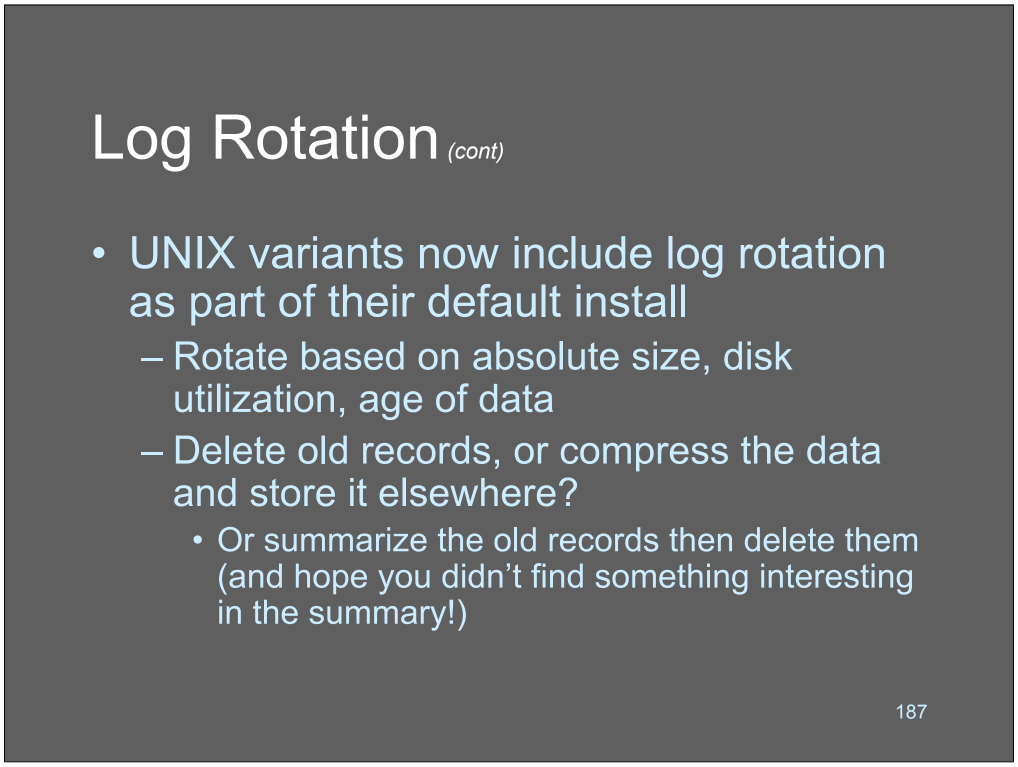
- Many circumstances can create huge quantities of logs
  - Deliberate DoS attempts
  - Nimda
  - Forwarding Apache access logs to *syslog*
    - ***Not*** recommended for busy websites!
- Protect loghost's integrity by rotating and archiving logs regularly
  - Have it monitor its disk usage!

186

Log rotation is something that is usually performed on a daily basis.

The problem with the stock "newsyslog" routines that come with many O/S is that they don't handle the case when the logs get unusually large very fast. For high-volume servers, consider rewriting newsyslog so that it runs hourly instead, and include checks for (df -h /var/log) disk space problems or unusual file sizes.

One thing we don't recommend is forwarding web server logs from busy sites over syslog. The traffic bursts that web servers generate can easily swamp UDP input/output queues and cause loss of data. If you want your apache logs, use SSH or rsync or some other reliable way of collecting them in batch mode. Hint: apache logs compress really well - compress them first and you'll save a lot of bandwidth. Compression also helps minimize the impact of log-overflows. A log file with millions of duplicates of a particular message will compress the millions of duplicates down to a single instance, automatically.

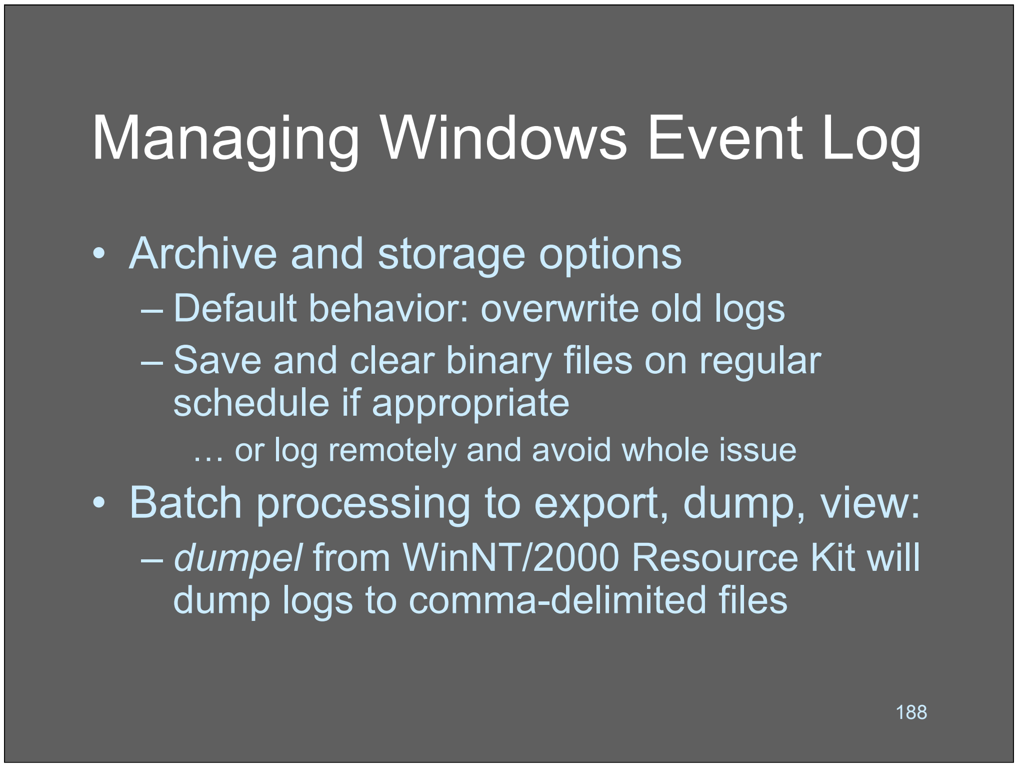


## Log Rotation *(cont)*

- UNIX variants now include log rotation as part of their default install
  - Rotate based on absolute size, disk utilization, age of data
  - Delete old records, or compress the data and store it elsewhere?
    - Or summarize the old records then delete them (and hope you didn’t find something interesting in the summary!)

187

Virtually every UNIX-like system includes a different way of handling log rotation. Don’t be afraid to replace them, if they don’t do the job for you!


## Managing Windows Event Log

- Archive and storage options
  - Default behavior: overwrite old logs
  - Save and clear binary files on regular schedule if appropriate
    - … or log remotely and avoid whole issue
- Batch processing to export, dump, view:
  - *dumpel* from WinNT/2000 Resource Kit will dump logs to comma-delimited files

188

http://packetstorm.securify.com/NT lists a variety of useful freeware for NT administration and security.  One of them, NTOtools2.zip, contains a command line tool for backing up the three Event Log binary files and clearing the active logs.  This action can be scheduled through the NT "at" command or using the Task Scheduler, and protects you from the danger of your log files being unexpectedly overwritten.

The tools are also available at http://www.securityfocus.com/data/tools/ntotoolsSD.zip.

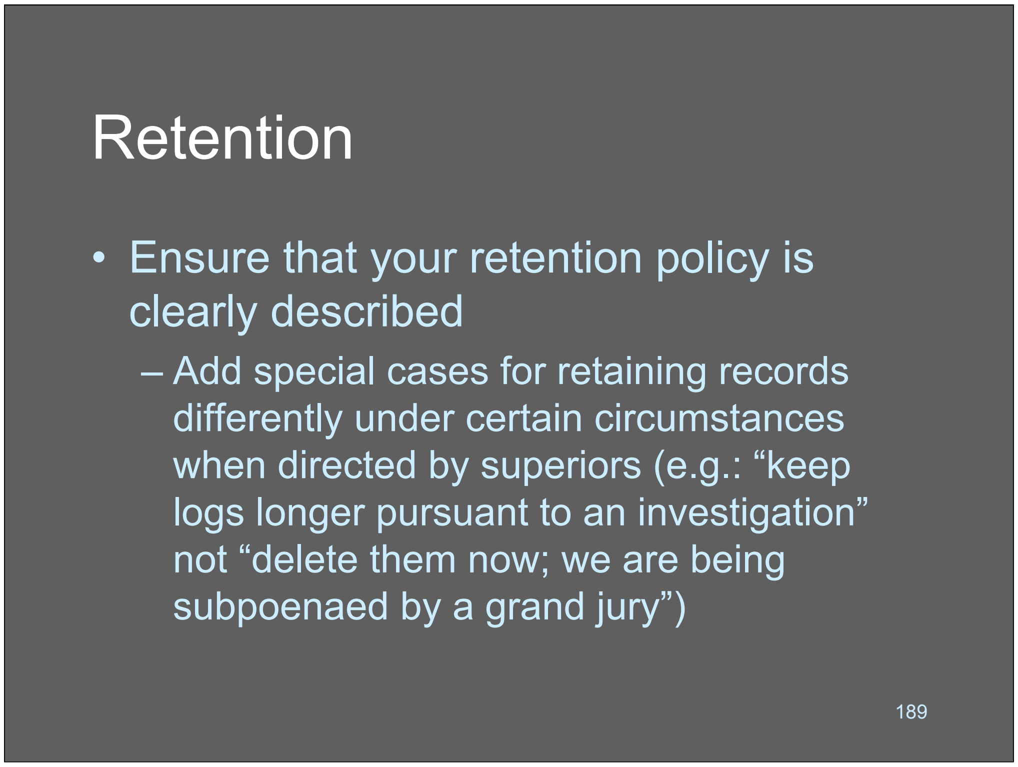
From the NT command line,

```
ntolog \\SERVER /b /c /sec /f 05092001.evt
```

will back up your current Security Event log to a file called *05092001.evt* and clear the active log.

```
dumpel -f 05092001.txt -l Security -d 1
```

will dump the last day's worth of entries from the Security Events log to a file called *05092001.txt.*



## Retention

- Ensure that your retention policy is clearly described
  - Add special cases for retaining records differently under certain circumstances when directed by superiors (e.g.: “keep logs longer pursuant to an investigation” not “delete them now; we are being subpoenaed by a grand jury”)

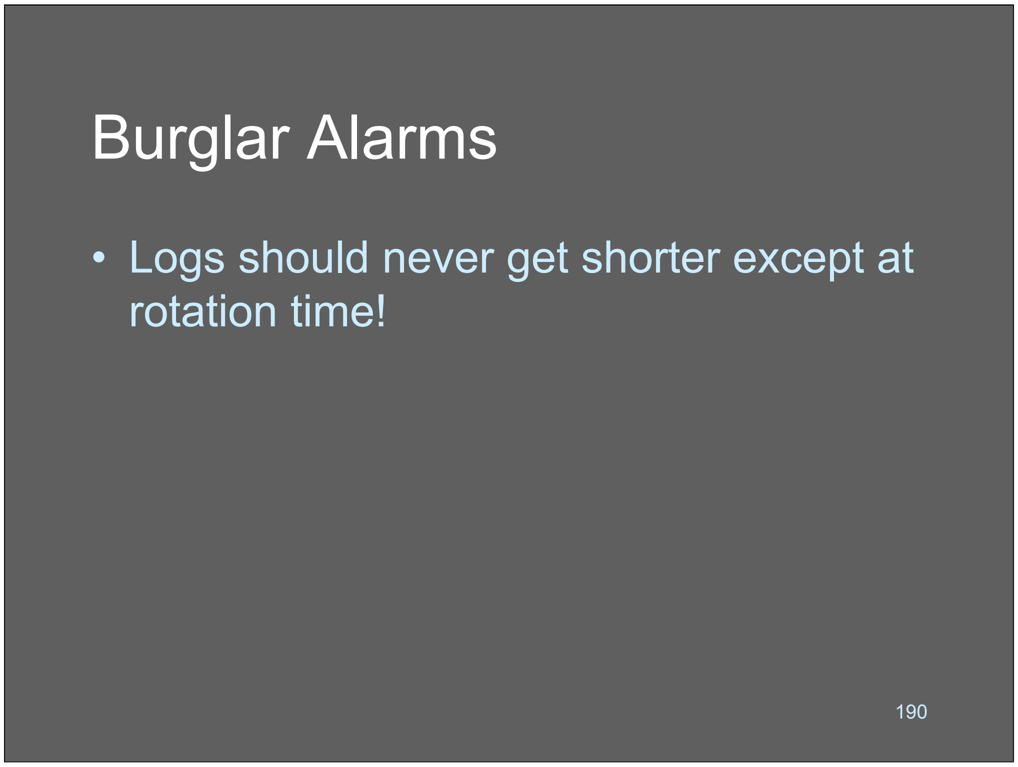
189

Make sure that you address a retention policy in *case* you ever want to do something with your logs for legal purposes. The importance of logs is that they are a business record. You should keep them *exactly* as described in your policy - no more, and no less.

Consider adding special clauses in your policy that direct variations of log-processing based on circumstances. E.g.:

*System logs are to be retained on fast media for a week, and slow media (tape) for 1 year. After one year, they are to be purged. In the event that security staff, human resources, or other authorized managers of bigfoo, Inc, request it pursuant to an investigation, selected logs will be preserved on fast media for the duration of the investigation.*

Presto! Now you’re covered. If your boss tells you to investigate what’s going on for a particular day’s logs, you’re covered if you keep them past the mandated expiry date.


## Burglar Alarms

- Logs should never get shorter except at rotation time!

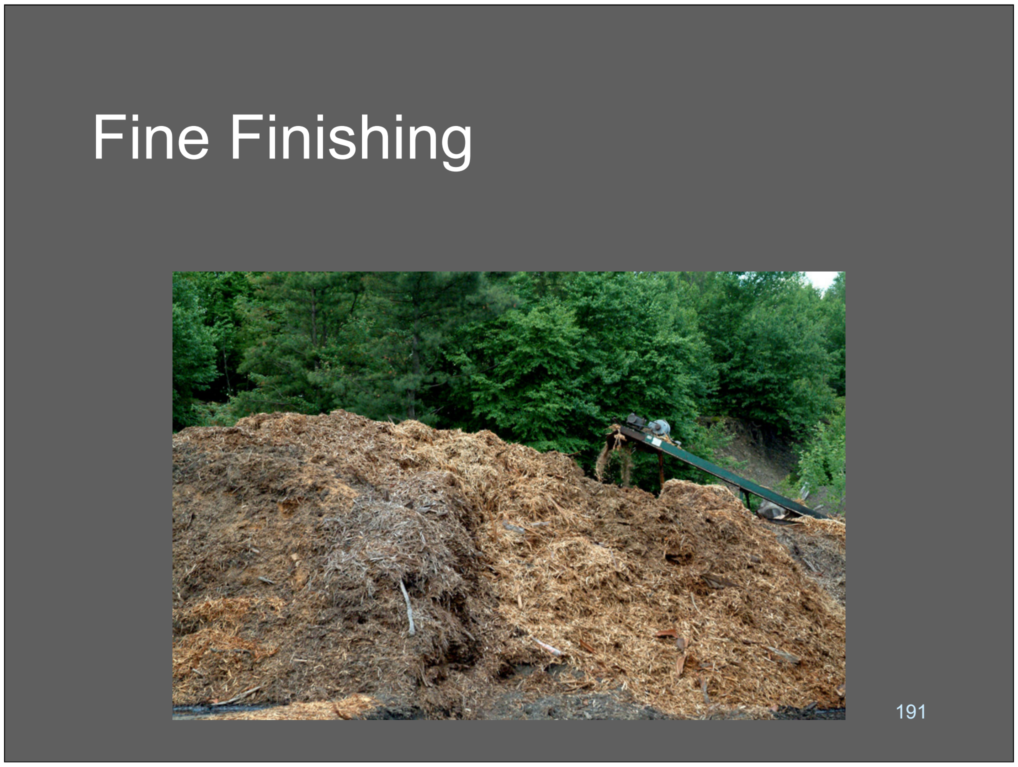
190

Consider putting hooks directly into your log processing loop that will notify you if your logs ever get shorter all of a sudden! On one high value target, a friend of mine used to have a process that lurked in the background and did not a whole lot more than watch for /var/log/messages getting shorter except for around midnight.

Extra credit for the UNIX gurus in the room: if the process opens /var/log/messages and does an fstat() on the fd to see if its inode has changed at an unusual time:

•What has happened?

•What would happen if the program fseek()s the fd to 0L and read/writes the file to another location?

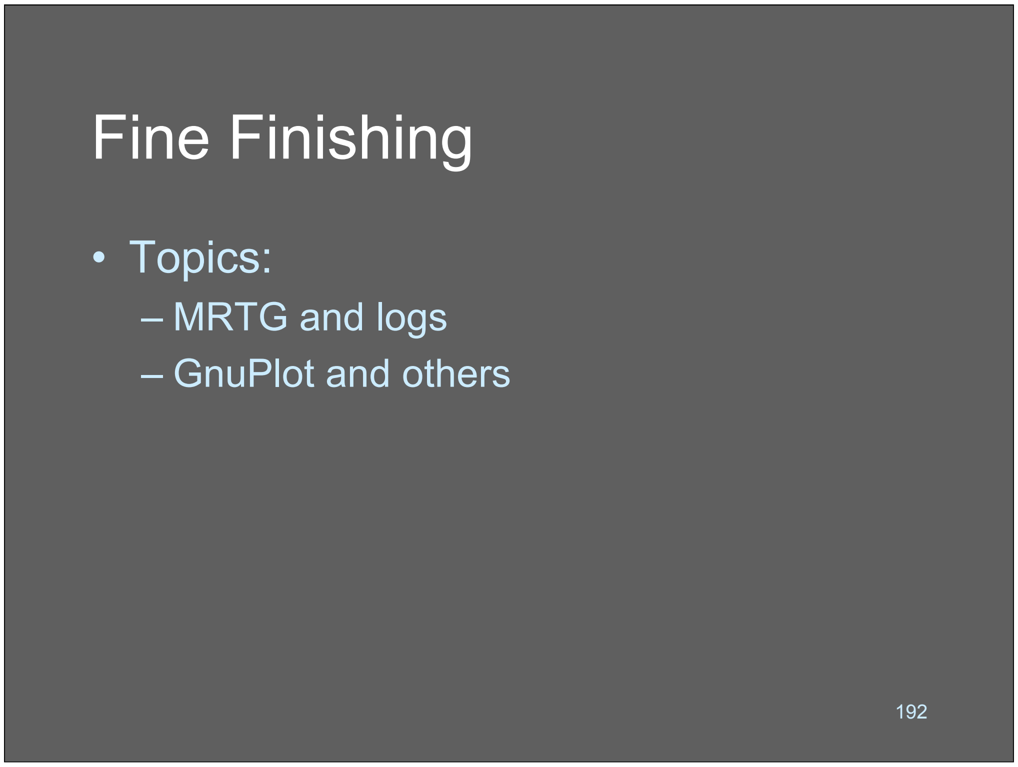


## Fine Finishing

191

Ever wondered what comes out the back of a sawmill? Mulch, of course. Lots of mulch. Any time my wife wants fresh oak mulch she can take our pickup truck across the street and have fun with a shovel. Actually, the guys are usually so nice they won't let a lady do all the work, and they dump a scoop from the backhoe.

Periodically, the mulch is loaded into a semitrailer and hauled to a paper mill or garden supply store.

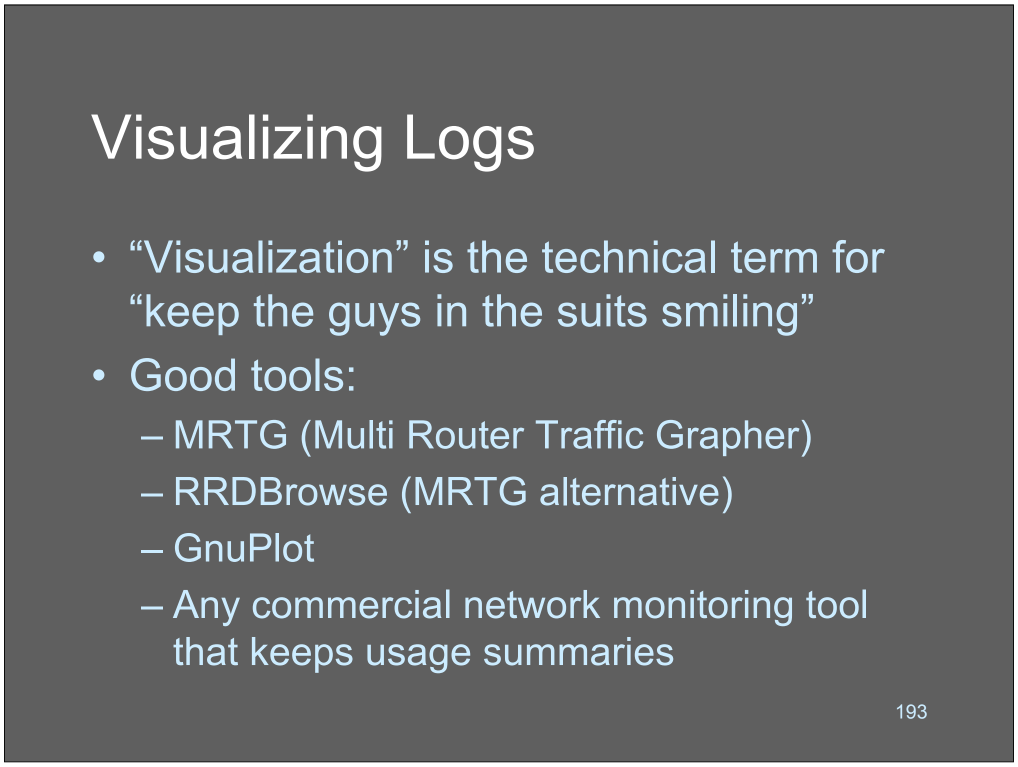


## Fine Finishing

- Topics:
  - MRTG and logs
  - GnuPlot and others

192

One of the most under-thought-of aspects of log analysis is reporting. Technical experts really don't care much about that stuff, but managers do. So if you want to show the value of your log analysis, you're going to have to just accept that there are going to be 3-d pie charts in your future. We're going to talk a little bit about 2 approaches for plotting and graphing your output.



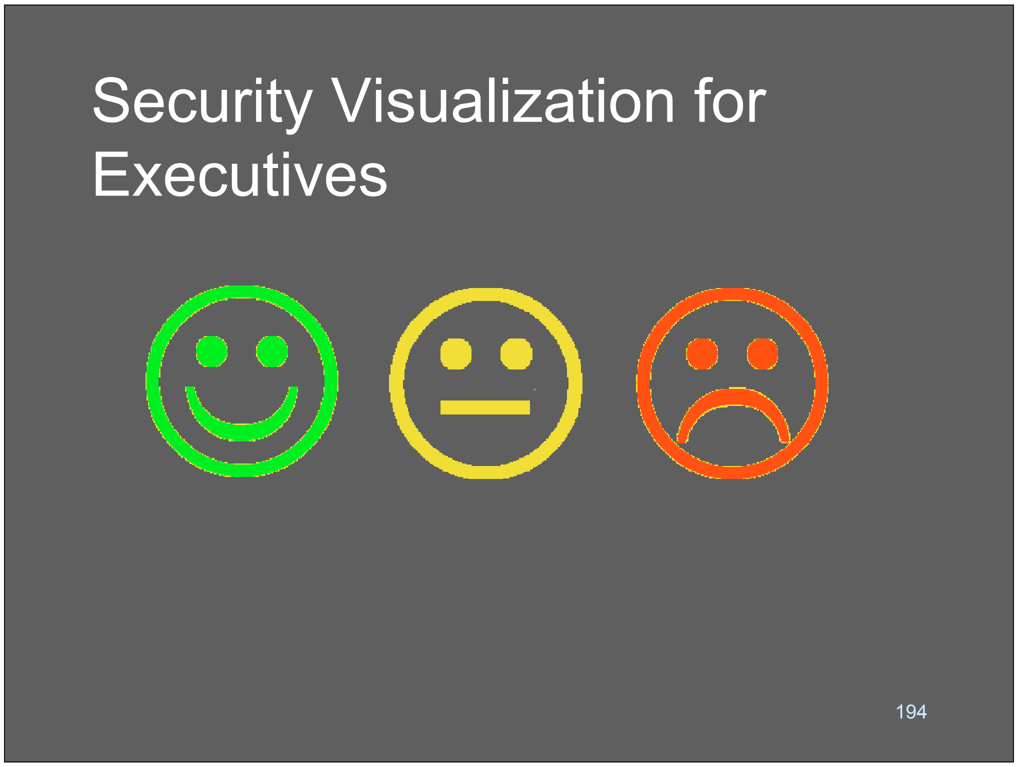
## Visualizing Logs

- "Visualization" is the technical term for "keep the guys in the suits smiling"
- Good tools:
  - MRTG (Multi Router Traffic Grapher)
  - RRDBrowse (MRTG alternative)
  - GnuPlot
  - Any commercial network monitoring tool that keeps usage summaries

193

Visualization is a good exploratory tool. It's what you do when you've got a lot of data and you're not sure what you're looking for. So you compress the data visually onto a screen, somehow, so your eyeball can make sense of it. Sometimes the visualizations may be complex and involve things that make your 3-d graphic card ache in pain. Other times they are simple.

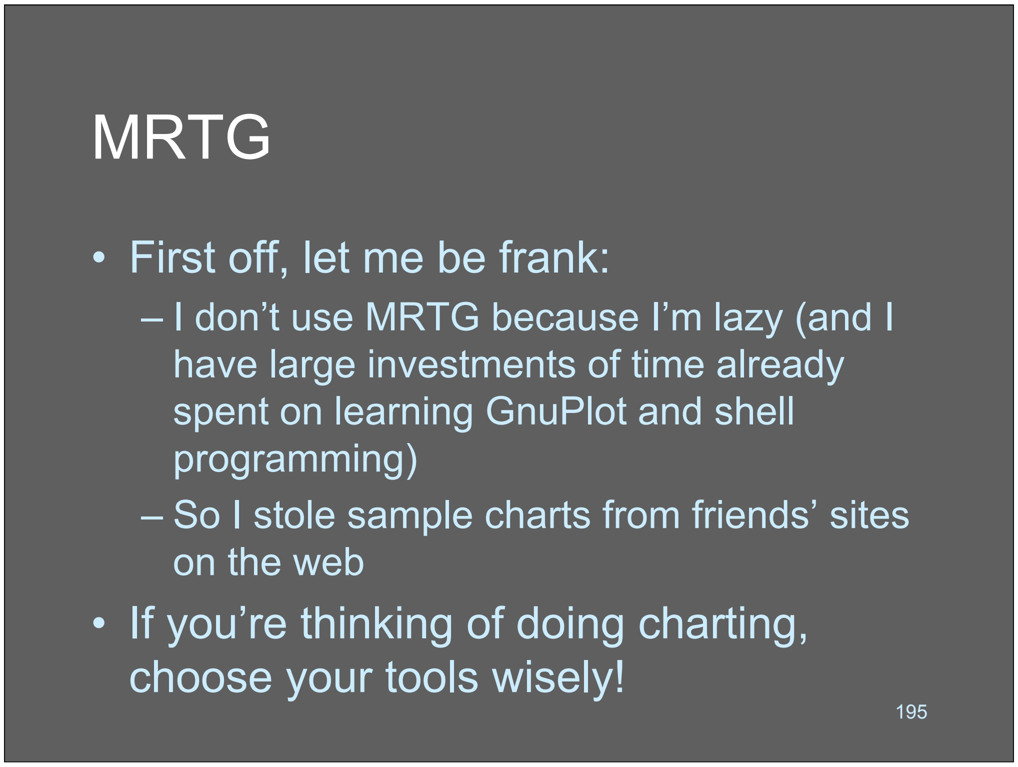
My preference is for histograms of time-related data, ideally with a moving average, if you can compute it. Most network monitoring tools and usage tools support histograms. You can build your own very easily with tools like GNUplot, too.



## Security Visualization for Executives

194

The red guy over there means you just made the cover of the Wall St Journal!



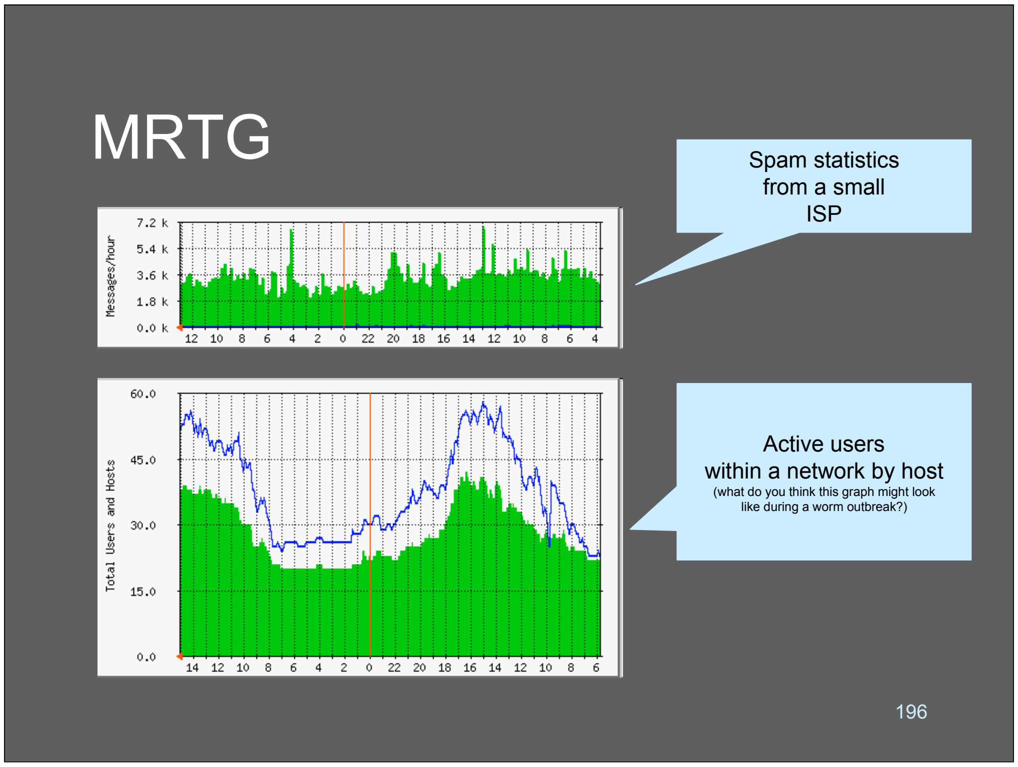
## MRTG

- First off, let me be frank:
  - I don’t use MRTG because I’m lazy (and I have large investments of time already spent on learning GnuPlot and shell programming)
  - So I stole sample charts from friends’ sites on the web
- If you’re thinking of doing charting, choose your tools wisely!

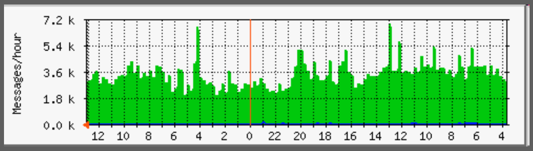
195

When you’re doing this stuff you’ll find that it’s best to get familiar with one tool and stick with it. Spend a week or so researching the options and then pick one and forget that the others ever existed.

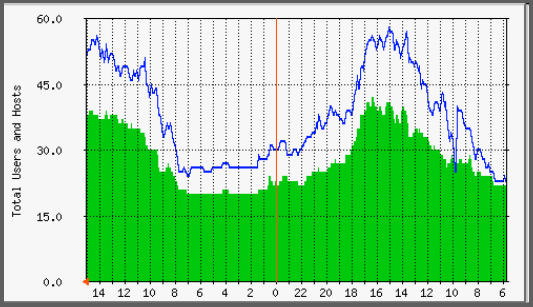
I’ve been writing my own charts using plot and later GNUplot for about 10 years, now - it’s too late to teach this particular old dog new tricks. So I don’t actually use MRTG myself but I know a lot of people who love it and swear by it. It’s worth a look, especially if you are in an environment where SNMP is used to manage systems.



## MRTG



Spam statistics from a small ISP

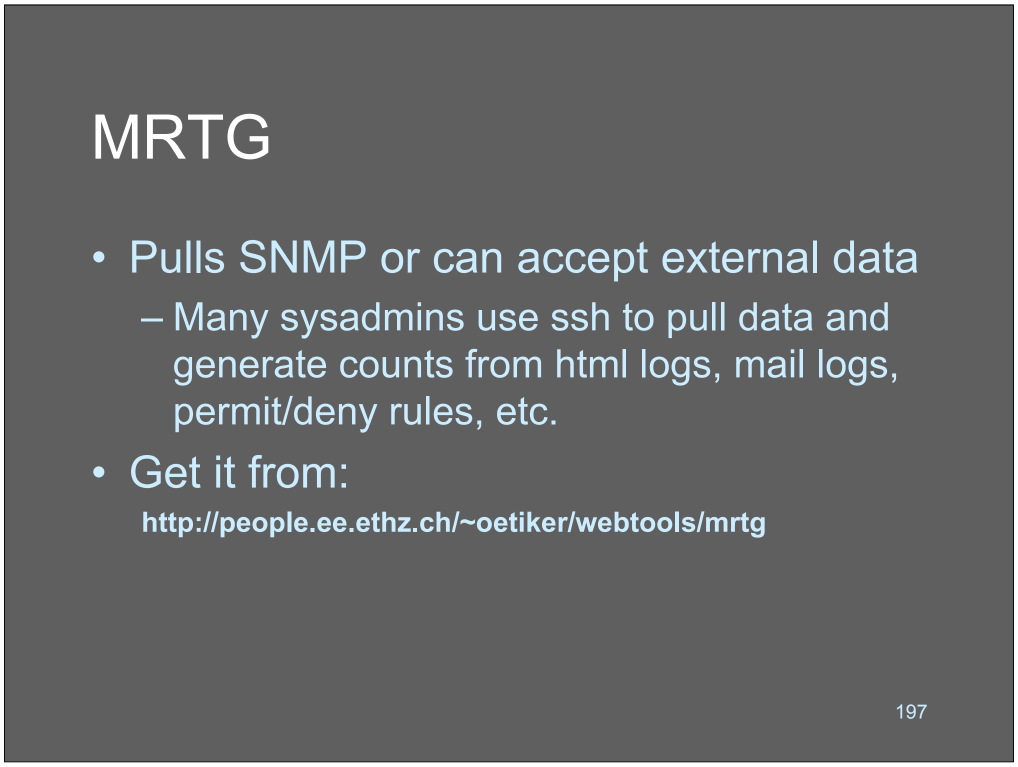


Active users within a network by host
(what do you think this graph might look like during a worm outbreak?)

196

This is some MRTG chart output from a couple of sites showing the kind of data you can collect and what it looks like. MRTG graphs have a sameness to their look, which is not a bad thing. Basically, you set it up to present a couple of different items on a graph, throw data into it, and you get a graph. Easy. Then you link those graphs to a web page that you wrap with HTML that makes it reload every minute. Presto! You have a $300,000 network management console! (Except that yours works and you can customize it)

Both of these charts represent things we'd want to know about if they were changing dramatically all of a sudden. Showing it visually is a very good way to keep tabs on it.



## MRTG

- Pulls SNMP or can accept external data
  - Many sysadmins use ssh to pull data and generate counts from html logs, mail logs, permit/deny rules, etc.
- Get it from:

  **http://people.ee.ethz.ch/~oetiker/webtools/mrtg**

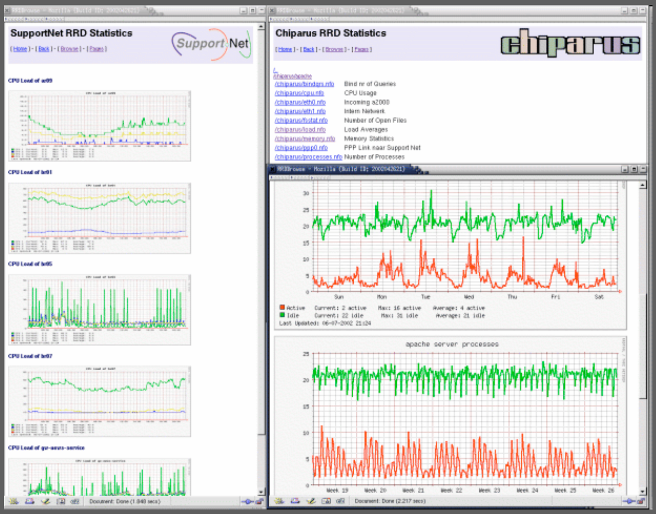
197

MRTG is designed to pull its data via SNMP so if you're an SNMP site or an SNMP guru this is a good tool for you. If you're more of a batch-oriented guy you can pull data down using other mechanisms and feed it into MRTG directly.

Would you like a nice constantly updated picture of firewall permits versus firewall denies? Of course you would!

MRTG is free; there's no excuse not to look at using it.

## RRDBrowse



SupportNet RRD Statistics

Chiparus RRD Statistics

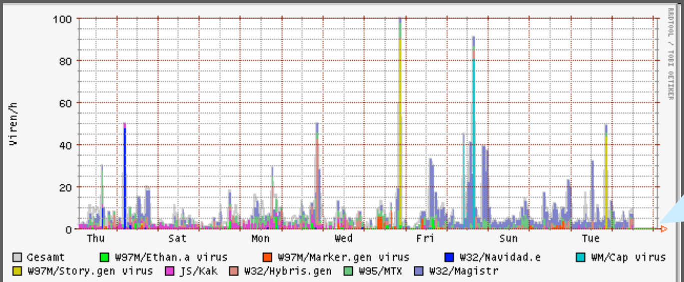
[ Home ] - [ Back ] - [ Browse ] - [ Pages ]

| | |
|---|---|
| /chiparus/bindqrs.nfo | Bind nr of Queries |
| /chiparus/cpu.nfo | CPU Usage |
| /chiparus/eth0.nfo | Incoming a2000 |
| /chiparus/eth1.nfo | Intern Netwerk |
| /chiparus/fsstat.nfo | Number of Open Files |
| /chiparus/load.nfo | Load Averages |
| /chiparus/memory.nfo | Memory Statistics |
| /chiparus/ppp0.nfo | PPP Link naar Support Net |
| /chiparus/processes.nfo | Number of Processes |

Lots of statistics on one grouped screen

RRDBrowse is an MRTG-like plotter for network traffic. It’s intended to be an “MRTG light” clustering console and puts a lot of different things onto a single central panel with grouped screens. Once again, you can customize it a great deal by deciding what you want to stuff into it.

## RRDBrowse



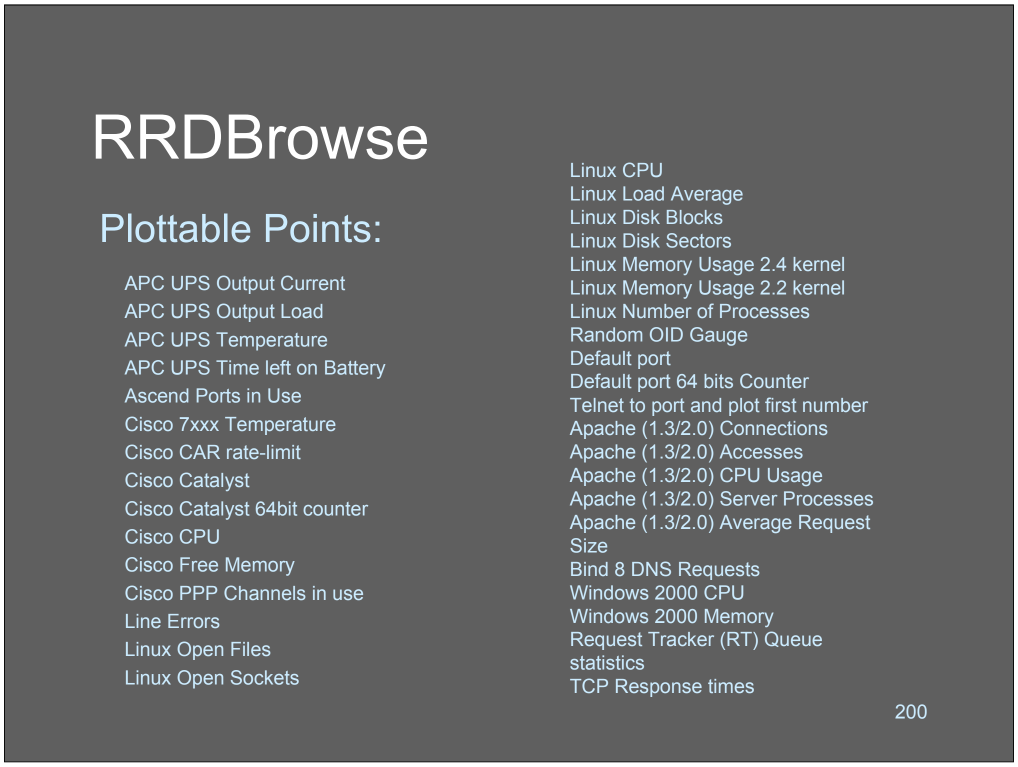
```
Abgewehrte eMail-Viren pro Stunde,   Thu Apr 19 02:00:00 2001 bis Thu May  3 02:00:00 2001

Virus/Wurm                 Mittelw.  Maximum
--------------------------------------------
W32/Magistr ............     2.79    38.00
W95/MTX ................     1.13    16.00
W32/Hybris.gen .........     0.85    41.00
JS/Kak .................     0.73     6.00
W97M/Story.gen virus ....    0.40    88.00
WM/Cap virus ...........     0.40    81.00
W32/Navidad.e ..........     0.19    48.00
W97M/Marker.gen virus ...    0.18     9.00
W97M/Ethan.a virus ......    0.11     6.00
Gesamt .................     7.21   100.00

                                         http://rs6000.univie.ac.at/virstats/
```

Plot of type of E-mail virus collected by firewall A/V

199

This is an example I found on the web of a network administrator who configured his copy of RRDBrowse to color-code and plot different viruses and the rate at which they impinged upon his network.

I think this is really kind of cool; if I were a suit I would be very excited by this chart, even though it doesn't provide actionable intelligence.



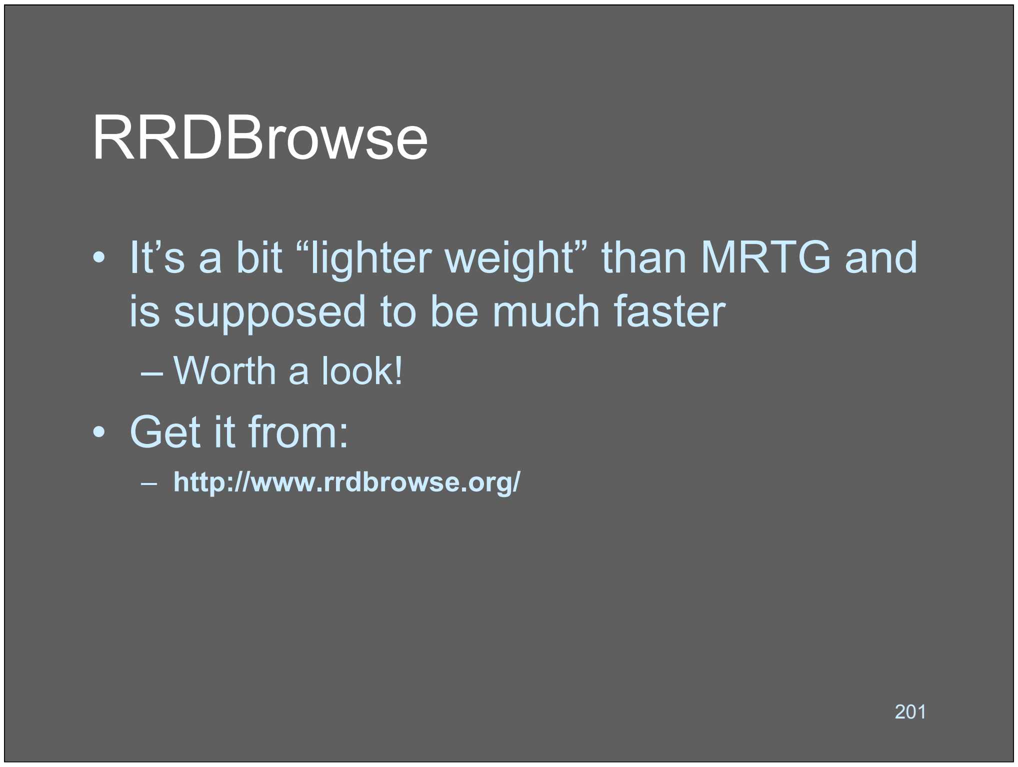
## RRDBrowse

### Plottable Points:

APC UPS Output Current
APC UPS Output Load
APC UPS Temperature
APC UPS Time left on Battery
Ascend Ports in Use
Cisco 7xxx Temperature
Cisco CAR rate-limit
Cisco Catalyst
Cisco Catalyst 64bit counter
Cisco CPU
Cisco Free Memory
Cisco PPP Channels in use
Line Errors
Linux Open Files
Linux Open Sockets
Linux CPU
Linux Load Average
Linux Disk Blocks
Linux Disk Sectors
Linux Memory Usage 2.4 kernel
Linux Memory Usage 2.2 kernel
Linux Number of Processes
Random OID Gauge
Default port
Default port 64 bits Counter
Telnet to port and plot first number
Apache (1.3/2.0) Connections
Apache (1.3/2.0) Accesses
Apache (1.3/2.0) CPU Usage
Apache (1.3/2.0) Server Processes
Apache (1.3/2.0) Average Request Size
Bind 8 DNS Requests
Windows 2000 CPU
Windows 2000 Memory
Request Tracker (RT) Queue statistics
TCP Response times

200

This is a *partial* list of the SNMP-pollable counters that RRDBrowse knows how to collect and do something with right out of the box. As you can see, some of them might be immediately useful to a security analyst. There are mystical incantations in RRDBrowse that will let you inject your own information into the system, as well, if you want to. So you could easily plot firewall permit/denies, for example.

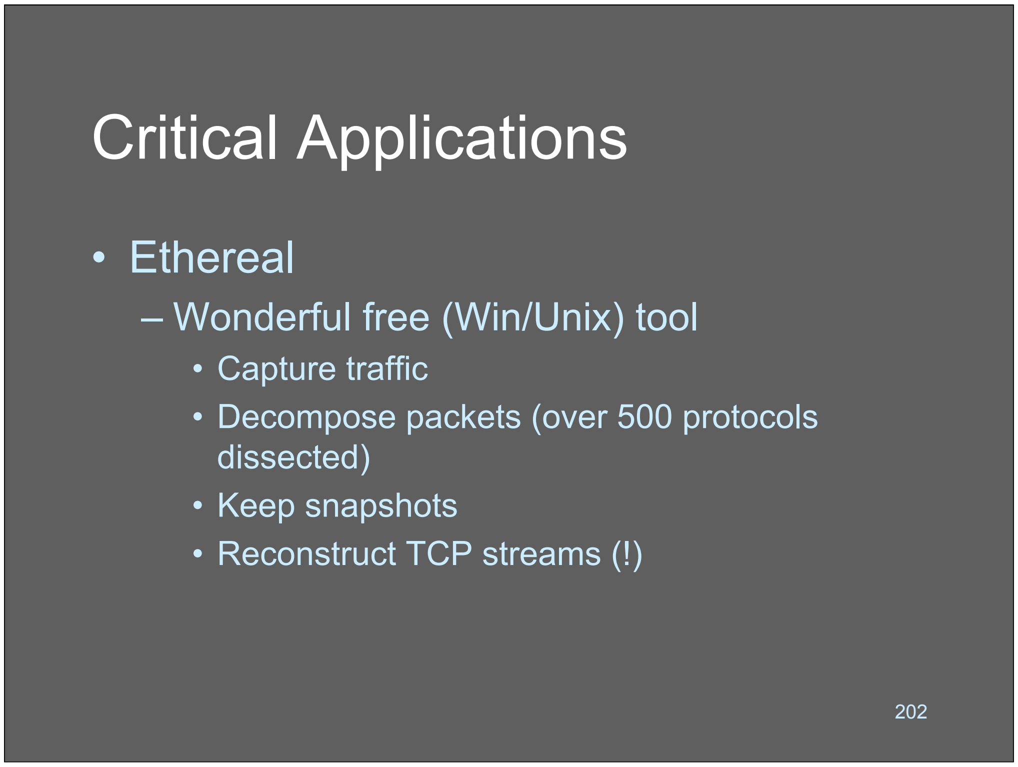


## RRDBrowse

- It's a bit "lighter weight" than MRTG and is supposed to be much faster
  - Worth a look!
- Get it from:
  - **http://www.rrdbrowse.org/**

201

What are the relative advantages/disadvantages between RRDBrowse and MRTG? MRTG is supposedly more powerful and flexible, RRDBrowse is supposedly faster and easier to set up. My guess is that either is a fine tool and they're both better than nothing!

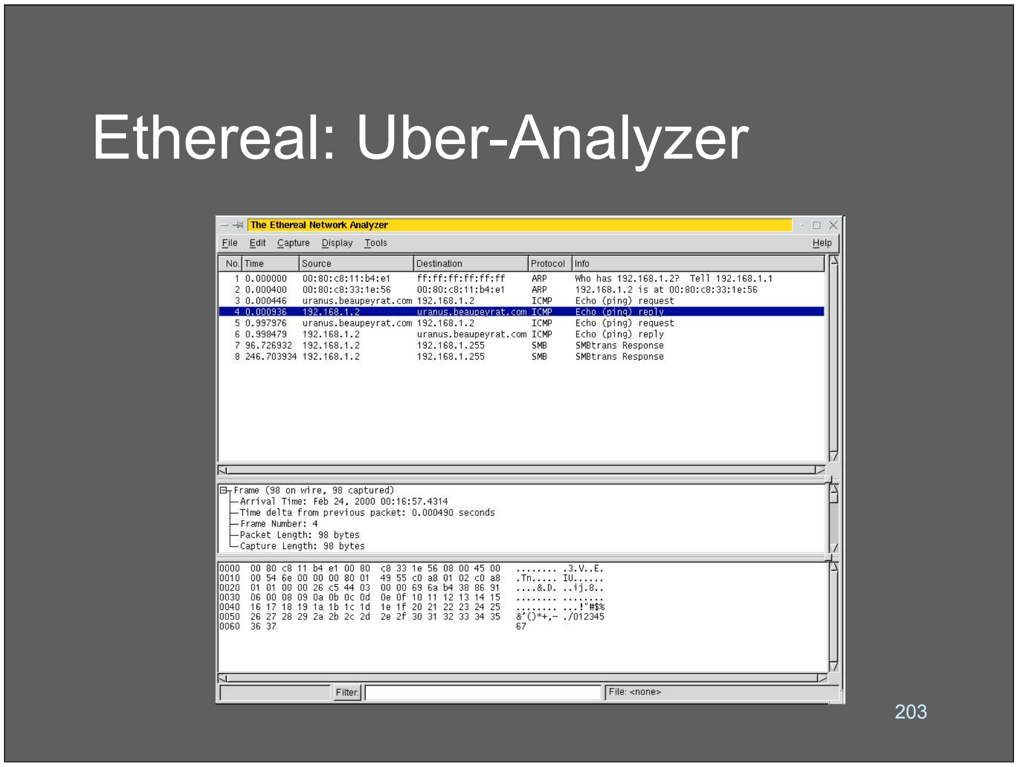


## Critical Applications

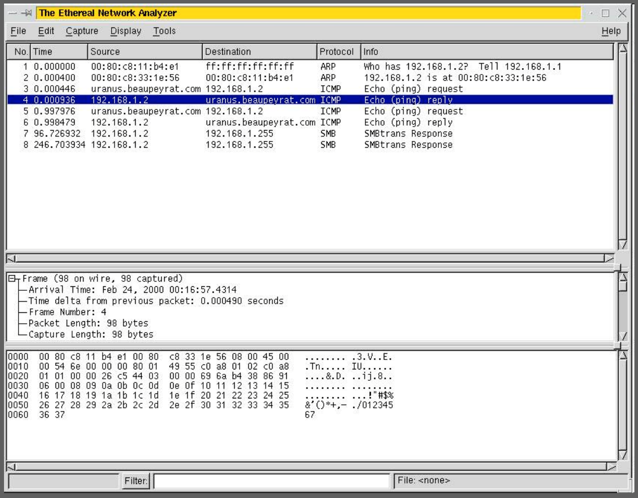
- Ethereal
  - Wonderful free (Win/Unix) tool
    - Capture traffic
    - Decompose packets (over 500 protocols dissected)
    - Keep snapshots
    - Reconstruct TCP streams (!)

202

For log analysts, Ethereal is a key tool, especially if you are collecting data from tcpdump to inject into your log stream. I'm a big fan of using tcpdump to collect specific types of packets to a file, then periodically killing off tcpdump and restarting it, while passing the file to tcpdump -r to a batch process that does further analysis on it. One thing that's nice about this approach is that if you find something you've never before seen, you have the raw packets sitting right there to look at. And, if you ever have raw packets to look at, Ethereal is *the* tool for the job.



## Ethereal: Uber-Analyzer


The Ethereal Network Analyzer
File Edit Capture Display Tools Help
Filter:
File: <none>

```
No. Time       Source                Destination           Protocol Info
  1 0.000000   00:80:c8:11:b4:e1     ff:ff:ff:ff:ff:ff     ARP      Who has 192.168.1.2?  Tell 192.168.1.1
  2 0.000400   00:80:c8:33:1e:56     00:80:c8:11:b4:e1     ARP      192.168.1.2 is at 00:80:c8:33:1e:56
  3 0.000446   uranus.beaupeyrat.com 192.168.1.2           ICMP     Echo (ping) request
  4 0.000936   192.168.1.2           uranus.beaupeyrat.com ICMP     Echo (ping) reply
  5 0.997976   uranus.beaupeyrat.com 192.168.1.2           ICMP     Echo (ping) request
  6 0.998479   192.168.1.2           uranus.beaupeyrat.com ICMP     Echo (ping) reply
  7 96.726932  192.168.1.2           192.168.1.255         SMB      SMBtrans Response
  8 246.703934 192.168.1.2           192.168.1.255         SMB      SMBtrans Response
```

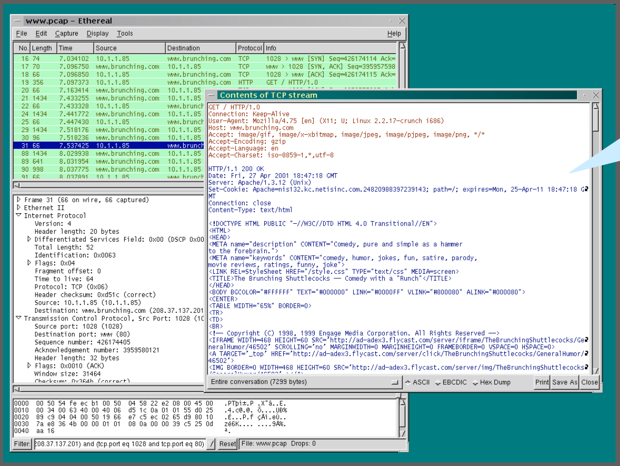
```
⊟┬Frame (98 on wire, 98 captured)
 ├─Arrival Time: Feb 24, 2000 00:16:57.4314
 ├─Time delta from previous packet: 0.000490 seconds
 ├─Frame Number: 4
 ├─Packet Length: 98 bytes
 └─Capture Length: 98 bytes
```

```
0000  00 80 c8 11 b4 e1 00 80  c8 33 1e 56 08 00 45 00   ........ .3.V..E.
0010  00 54 6e 00 00 00 80 01  49 55 c0 a8 01 02 c0 a8   .Tn..... IU......
0020  01 01 00 00 26 c5 44 03  00 00 69 6a b4 38 86 91   ....&.D. ..ij.8..
0030  06 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ ...!"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

203

This is just a simple example of Ethereal displaying a packet. You can see the nice way it explodes the header fields out for you in the middle panel.

# Ethereal: Uber-Analyzer



```
No. Length Time      Source            Destination       Protocol Info
16  74     7.034102  10.1.1.85         www.brunching.com TCP      1028 > www [SYN] Seq=426174114 Ack=
17  70     7.096750  www.brunching.com 10.1.1.85         TCP      www > 1028 [SYN, ACK] Seq=395957598
18  66     7.096850  10.1.1.85         www.brunching.com TCP      1028 > www [ACK] Seq=426174115 Ack=
19  356    7.097373  10.1.1.85         www.brunching.com HTTP     GET / HTTP/1.0
20  66     7.163414  www.brunching.com 10.1.1.85
21  1434   7.433255  www.brunching.com 10.1.1.85
22  66     7.433328  10.1.1.85         www.brunch
24  1434   7.441772  www.brunching.com 10.1.1.85
25  66     7.447430  10.1.1.85         www.brunch
29  1434   7.518176  www.brunching.com 10.1.1.85
30  96     7.518236  www.brunching.com 10.1.1.85
31  66     7.537425  10.1.1.85         www.brunch
88  1434   8.029938  www.brunching.com 10.1.1.85
89  641    8.031954  www.brunching.com 10.1.1.85
90  998    8.037775  www.brunching.com 10.1.1.85
91  66     8.037891  10.1.1.85         www.brunch
```

```
▷ Frame 31 (66 on wire, 66 captured)
▷ Ethernet II
▽ Internet Protocol
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00
    Total Length: 52
    Identification: 0x0063
  ▷ Flags: 0x04
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0xd51c (correct)
    Source: 10.1.1.85 (10.1.1.85)
    Destination: www.brunching.com (208.37.137.201
▽ Transmission Control Protocol, Src Port: 1028 (10
    Source port: 1028 (1028)
    Destination port: www (80)
    Sequence number: 426174405
    Acknowledgement number: 3959580121
    Header length: 32 bytes
  ▷ Flags: 0x0010 (ACK)
    Window size: 31464
    Checksum: 0x364b (correct)
```

```
0000  00 50 54 fe ec b1 00 50  04 58 22 e2 08 00 45 00   .PTþì±.P .X"â..E.
0010  00 34 00 63 40 00 40 06  d5 1c 0a 01 01 55 d0 25   .4.c@.@. Õ....UÐ%
0020  89 c9 04 04 00 50 19 66  e7 c5 ec 02 65 d9 80 10   .É...P.f çÅì.eÙ..
0030  7a e8 36 4b 00 00 01 01  08 0a 00 00 39 c5 25 0d   zè6K.... ....9Å%.
0040  aa 16                                              ª.
```

Filter: 208.37.137.201) and (tcp.port eq 1028 and tcp.port eq 80)   Reset   File: www.pcap  Drops: 0

Contents of TCP stream

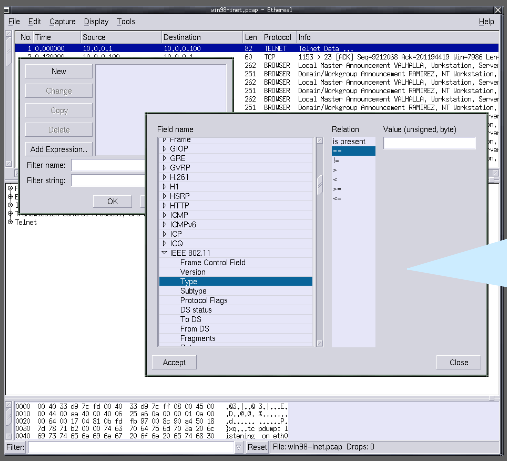
```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.75 [en] (X11; U; Linux 2.2.17-crunch i686)
Host: www.brunching.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8

HTTP/1.1 200 OK
Date: Fri, 27 Apr 2001 18:47:18 GMT
Server: Apache/1.3.12 (Unix)
Set-Cookie: Apache=nis132.kc.netisinc.com.24820988397239143; path=/; expires=Mon, 25-Apr-11 18:47:18 G
MT
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<META name="description" CONTENT="Comedy, pure and simple as a hammer
to the forebrain.">
<META name="keywords" CONTENT="comedy, humor, jokes, fun, satire, parody,
movie reviews, ratings, funny, joke">
<LINK REL=StyleSheet HREF="/style.css" TYPE="text/css" MEDIA=screen>
<TITLE>The Brunching Shuttlecocks — Comedy with a "Runch"</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#0000FF" VLINK="#800080" ALINK="#000080">
<CENTER>
<TABLE WIDTH="65%" BORDER=0>
<TR>
<TD>
<BR>
<!-- Copyright (C) 1998, 1999 Engage Media Corporation. All Rights Reserved -->
<IFRAME WIDTH=468 HEIGHT=60 SRC='http://ad-adex3.flycast.com/server/iframe/TheBrunchingShuttlecocks/Ge
neralHumor/46502' SCROLLING='no' MARGINWIDTH=0 MARGINHEIGHT=0 FRAMEBORDER=0 VSPACE=0 HSPACE=0>
<A TARGET='_top' HREF='http://ad-adex3.flycast.com/server/click/TheBrunchingShuttlecocks/GeneralHumor/
46502'>
<IMG BORDER=0 WIDTH=468 HEIGHT=60 SRC='http://ad-adex3.flycast.com/server/img/TheBrunchingShuttlecocks
```

Live! Stream decoding

204

This screenshot shows Ethereal stream-decoding a TCP stream. This is terrific for figuring out what happened in a packet capture set. Back in the old days we used to decode our TCP traffic by hand. Not anymore!

## Ethereal: Uber-Analyzer



```
No. Time       Source      Destination  Len Protocol Info
  1 0.000000   10.0.0.1    10.0.0.100   82  TELNET   Telnet Data ...
                                        60  TCP      1153 > 23 [ACK] Seq=9212068 Ack=201194419 Win=7986 Len
                                        262 BROWSER  Local Master Announcement VALHALLA, Workstation, Server
                                        251 BROWSER  Domain/Workgroup Announcement RAMIREZ, NT Workstation,
                                        262 BROWSER  Local Master Announcement VALHALLA, Workstation, Server
                                        251 BROWSER  Domain/Workgroup Announcement RAMIREZ, NT Workstation,
                                        262 BROWSER  Local Master Announcement VALHALLA, Workstation, Server
                                        251 BROWSER  Domain/Workgroup Announcement RAMIREZ, NT Workstation,
```

```
0000  00 40 33 d9 7c fd 00 40  33 d9 7c ff 08 00 45 00   .@3.|..@ 3.|...E.
0010  00 44 00 aa 40 00 40 06  25 a6 0a 00 00 01 0a 00   .D..@.@. %.......
0020  00 64 00 17 04 81 0b fd  fb 97 00 8c 90 a4 50 18   .d...... ......P.
0030  7d 78 71 b2 00 00 74 63  70 64 75 6d 70 3a 20 6c   }xq...tc pdump: l
0040  69 73 74 65 6e 69 6e 67  20 6f 6e 20 65 74 68 30   istening  on eth0
```

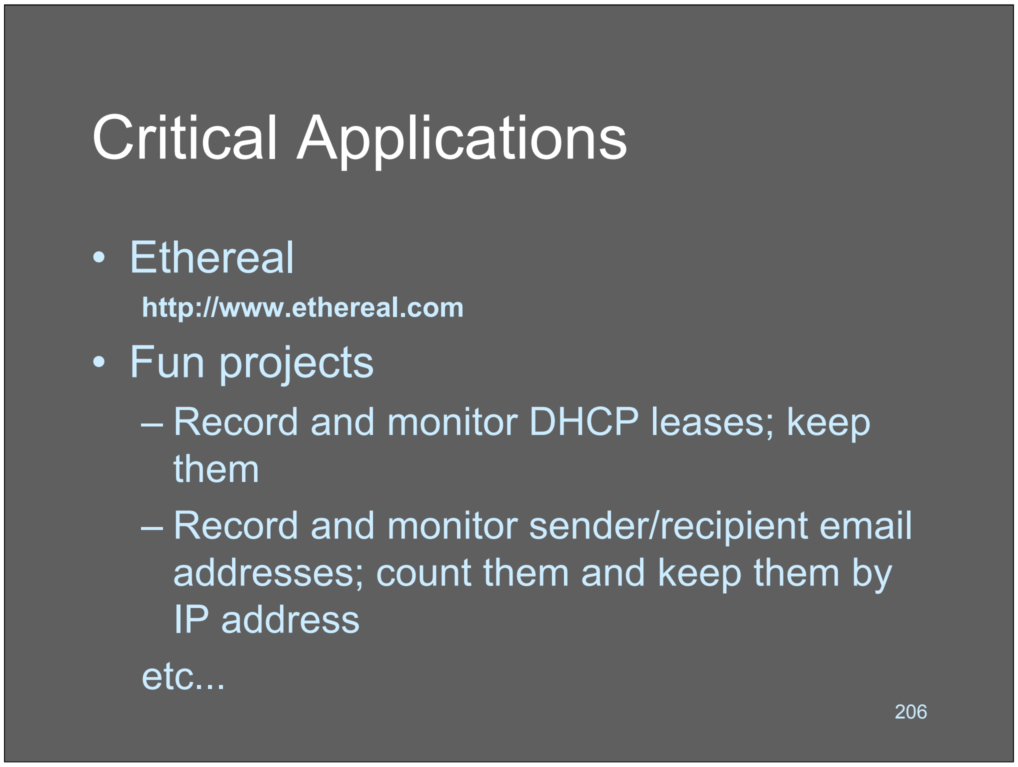
File: win98-inet.pcap Drops: 0

Incredibly powerful filtering lets you specify fields to collect

Save 'em and MRTG 'em?

205

I used to write complicated tcpdump rules by hand but now I have completely forgotten how to. Another wonderful feature of Ethereal is that it has menus that let you build queries based on full knowledge of what fields can be decoded and how. So you can write your tcpdump rule in Ethereal, and then use tcpdump to collect it on a command line for later analysis in Ethereal.



## Critical Applications

- Ethereal

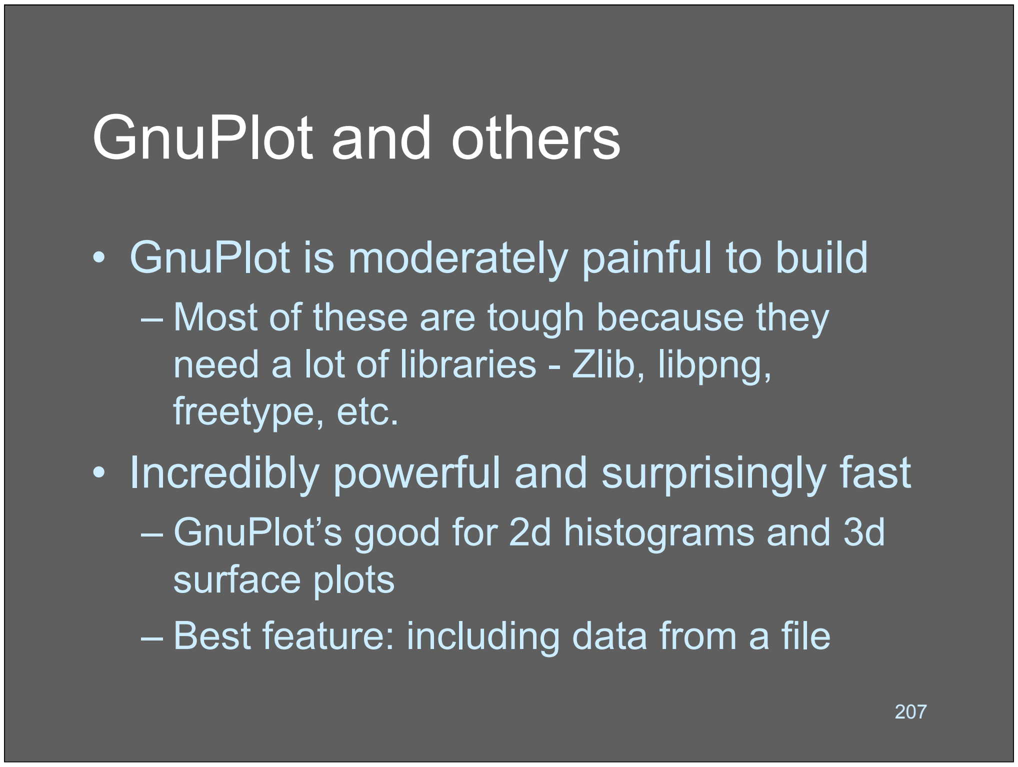
  **http://www.ethereal.com**

- Fun projects
  - Record and monitor DHCP leases; keep them
  - Record and monitor sender/recipient email addresses; count them and keep them by IP address

  etc...

206

Www.ethereal.com is the place to get this wonderful tool. Frankly, I think they're insane for giving away something so great - I'd gladly pay $100 for it (in fact, I did) since I like it better than a lot of the older commercial tools I used to use for the purpose.



## GnuPlot and others

- GnuPlot is moderately painful to build
  - Most of these are tough because they need a lot of libraries - Zlib, libpng, freetype, etc.
- Incredibly powerful and surprisingly fast
  - GnuPlot's good for 2d histograms and 3d surface plots
  - Best feature: including data from a file

207

Gnuplot is another wonderful tool, but it's a bit hard to build. In order to get a functioning Gnuplot build you'll need several libraries built and installed on your system. Install (in this order!)

•Zlib

•Libpng

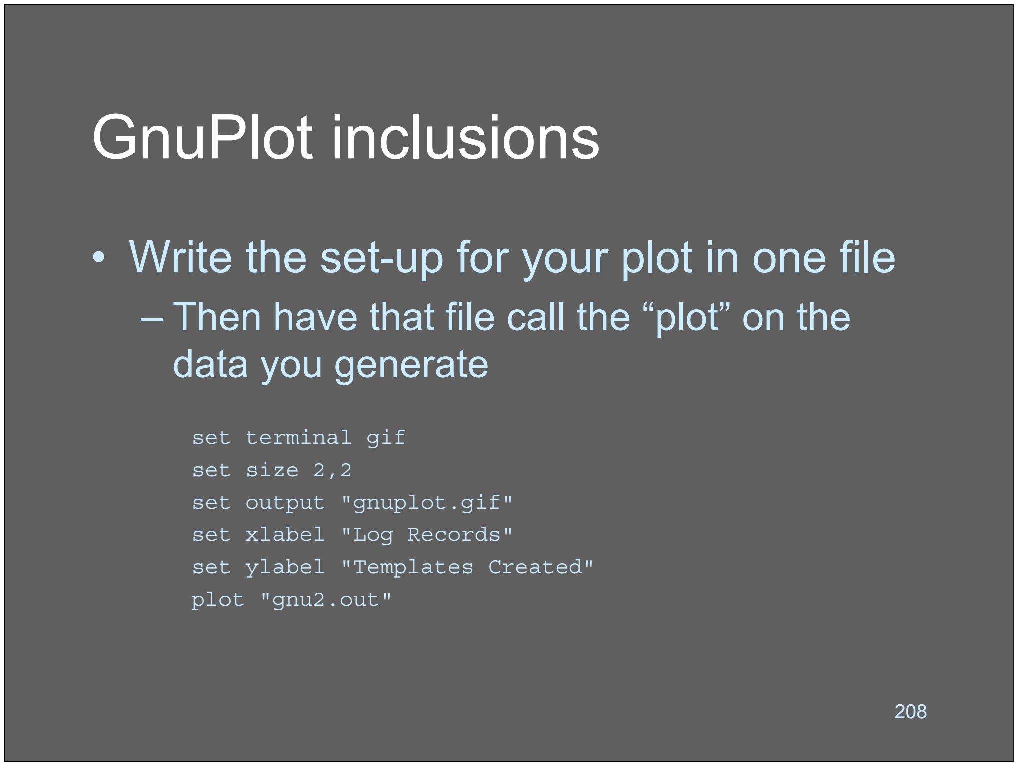
•Libjpeg

•freetype

then you can proceed with the configure/build process. Make sure you read the directions for Gnuplot before you just start with "configure" because "configure" will adjust your build based on what is available on the system. I produce almost all my plots as Jpegs because the Jpeg driver in Gnuplot lets you set an arbitrary pixel output. Do you want that chart wall-sized? No problem!

Gnuplot has zillions and zillions (really) of plotting options, so you can control the rendering of your data to a high degree. It's also *very* good about selecting sensible defaults. A lot of the time I just use default plot layouts; there's not a lot of effort in making a simple plot.

My favorite feature of Gnuplot is that it's very good about plotting data from a file. So you can build a gnuplot script that includes data from other files, and have multiple scripts that attack the same data from different perspectives - *or* you can point to a file where the data is constantly being updated.



## GnuPlot inclusions

- Write the set-up for your plot in one file
  - Then have that file call the “plot” on the data you generate

```
set terminal gif
set size 2,2
set output "gnuplot.gif"
set xlabel "Log Records"
set ylabel "Templates Created"
plot "gnu2.out"
```

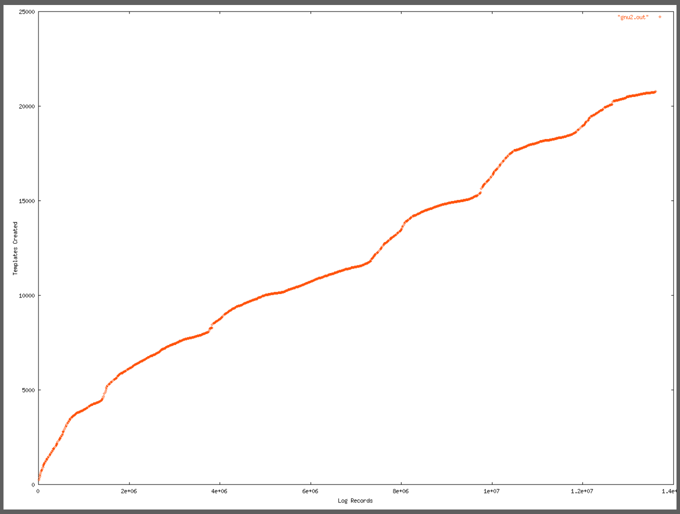
208

This is a simple example of a plot file for a fairly large and complex chart. “Set terminal” controls the output format or device; in this case to a .GIF file. I set the output size to a specific value that means something to the .GIF file output driver. Some output drivers take parameters like “large” while others (like the Jpeg driver) take parameters like “400x8909” - it’s device dependent. The “set output gnuplot.gif” tells gnuplot to write its output to the specified gif file/device.

Then we do some simple labeling. The X axis is “Log records” and the Y axis is “Templates Created”

The last line is the interesting one. It reads the file “gnu2.out” (which was created by another program) and generates a plot. The data format of the input file is just space-delimited lines of numbers - very easy to construct with a simple script.

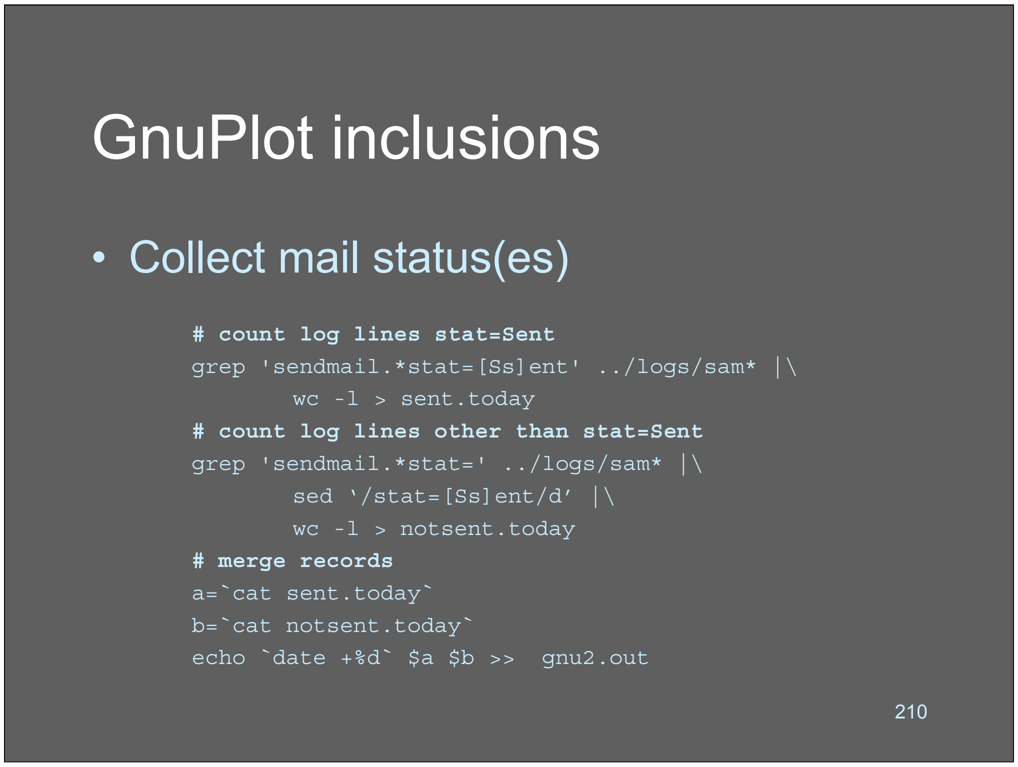
## GnuPlot



209

This is the output from a run with the preceeding script. You’ll notice that gnuplot just picked its own default values for the tick marks on the X and Y axis, and set its own scale, etc. There are lots of options that let you have an incredible degree of control over tick marks and scales, etc. One nice thing you can do is map a value to a particular tick mark, if you like, so you could make the left side named “Monday, Tuesday, Wednesday…” etc if you wanted and only had 7 values. You can change the colors of the points, put lines between them, make them bars, or literally dozens of other options.

This chart is an interesting one, by the way. It’s a plot of the rate at which NBS identified never-before-seen structures in 4 years of San Diego Supercomputer Center’s log data. What we can see is that there is apparently a constant level of introduction of new log templates in a large system. This indicates that parsing logs is going to be an ongoing process.



## GnuPlot inclusions

- Collect mail status(es)

```
# count log lines stat=Sent
grep 'sendmail.*stat=[Ss]ent' ../logs/sam* |\
       wc -l > sent.today
# count log lines other than stat=Sent
grep 'sendmail.*stat=' ../logs/sam* |\
       sed '/stat=[Ss]ent/d' |\
       wc -l > notsent.today
# merge records
a=`cat sent.today`
b=`cat notsent.today`
echo `date +%d` $a $b >>  gnu2.out
```

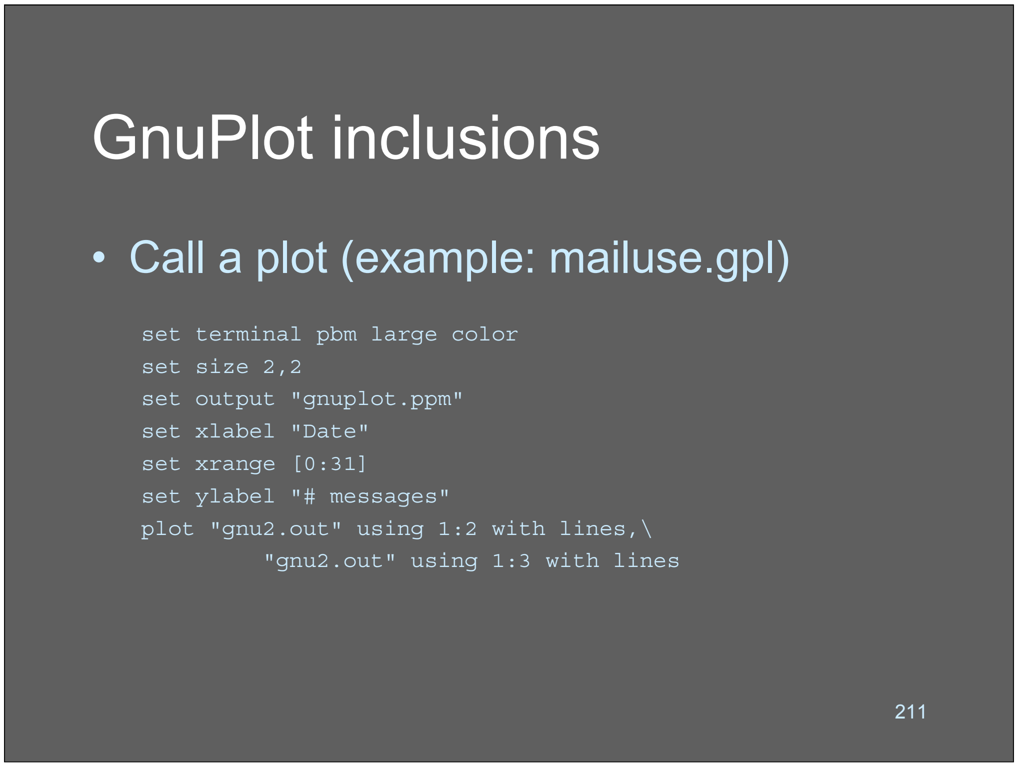
210

Here's a simple example of a silly shell script I whipped up to produce a Gnuplot output of a system value.

What we're doing is grepping a pattern out of sendmail and generating a count of the number of times it's seen using "wc -l" (word count, print lines). We're generating a count of messages *matching* "stat=sent" as well as a count of messages *not* matching stat=sent. (This is not an optimal processing approach, of course, if I really wanted this value I would pre-sort the data with syslog-ng and have it ready and waiting in a file…)  Once we've generated the data we echo it to the end of our "gnu2.out" file, which is our plot data.

A separate process is calling Gnuplot to plot "gnu2.out" periodically, so it just plots the most recent version of the data-set. What's nice is that I can use the same header layout for all my gnuplot scripts and just vary the input file to plot whatever dataset I am collecting. It's very fast and easy to do this. Mess with it for a day and you'll be producing management reports that will make everyone smile.

Who needs MRTG, huh?



## GnuPlot inclusions

- Call a plot (example: mailuse.gpl)

```
set terminal pbm large color
set size 2,2
set output "gnuplot.ppm"
set xlabel "Date"
set xrange [0:31]
set ylabel "# messages"
plot "gnu2.out" using 1:2 with lines,\
         "gnu2.out" using 1:3 with lines
```

211

This plot script should look pretty familiar; it’s basically the same as the other one, though I decided to plot to a pbm file (because my version of Gnuplot didn’t get built with the jpeg driver in it) and I am specifying a particular X range (days of the month)

The last line contains two “plot” commands, which will cause Gnuplot to produce two different wiggle-lines one using the first data item in the file and the other using the second.

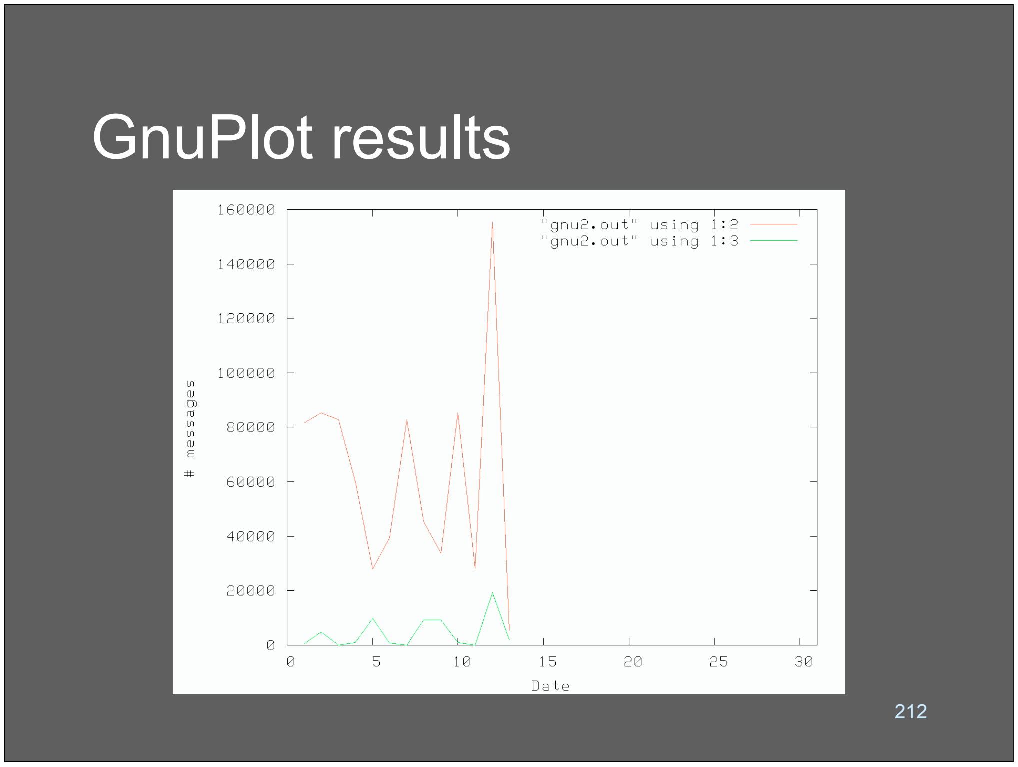
Our input file for this looks like:

1 473 203

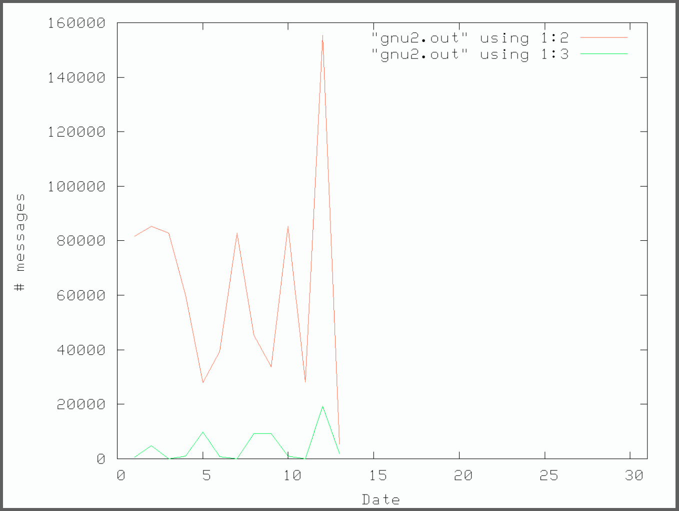
2 842 832

3 227 278

(I just made those up) the first field is the day of the month and the second and 3rd the values we want to plot against the day of the month.



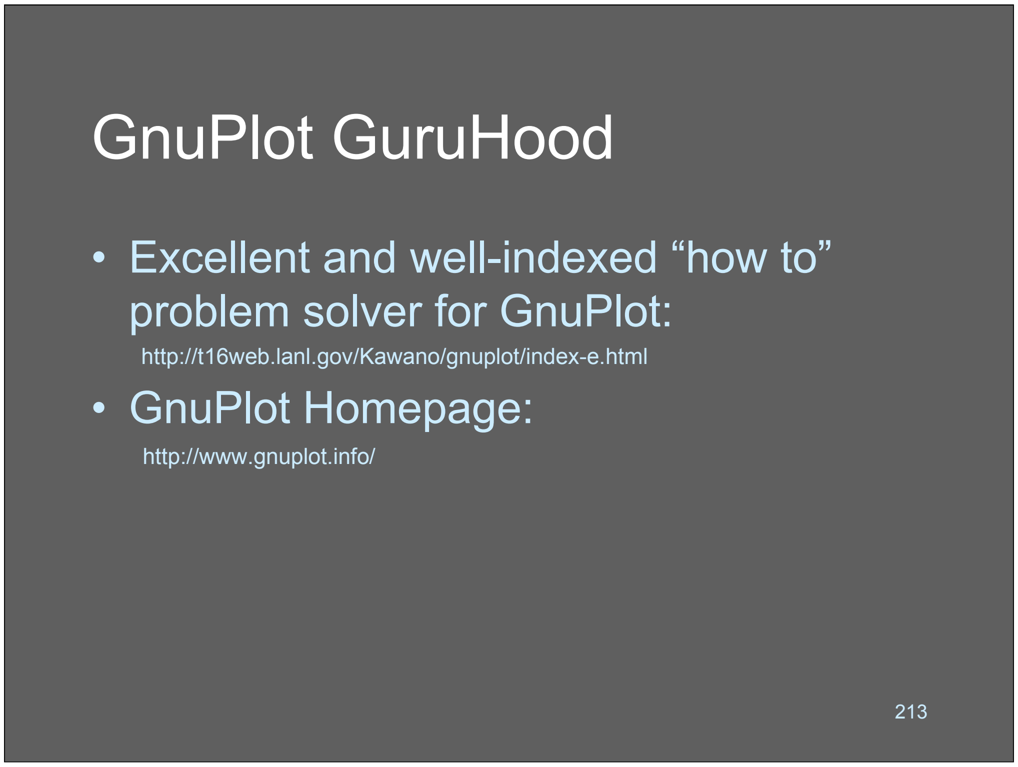
## GnuPlot results



212

And there’s the output from our previous example!

Looks professional, huh? That’s because it is! If you want to get fancy and add moving averages, deviation bars, multiple colors, etc, etc, you can do it. But the point here is that you can be up and Gnuplotting in a couple of hours without having to do a lot of rocket science.



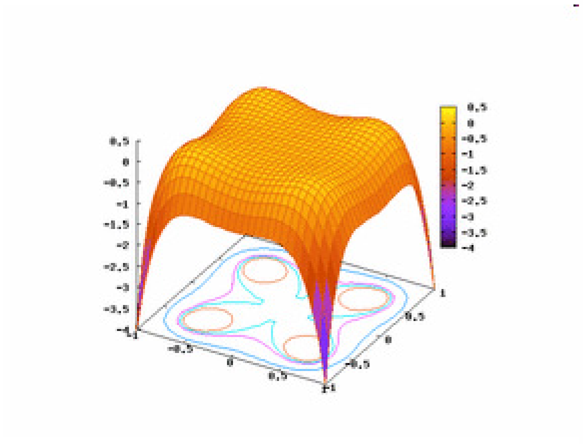
## GnuPlot GuruHood

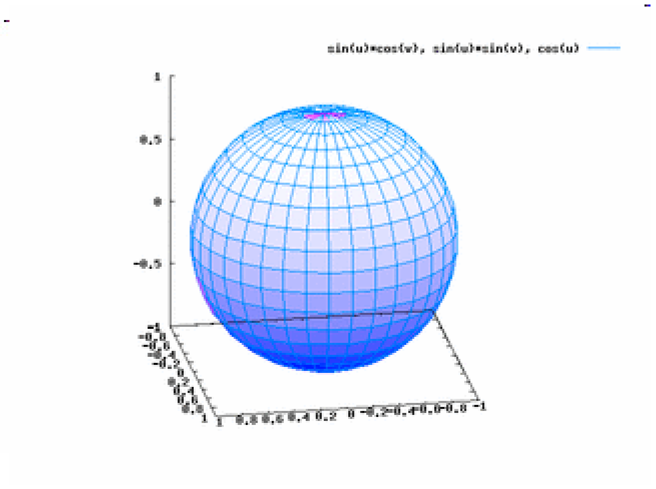
- Excellent and well-indexed "how to" problem solver for GnuPlot:
  http://t16web.lanl.gov/Kawano/gnuplot/index-e.html
- GnuPlot Homepage:
  http://www.gnuplot.info/

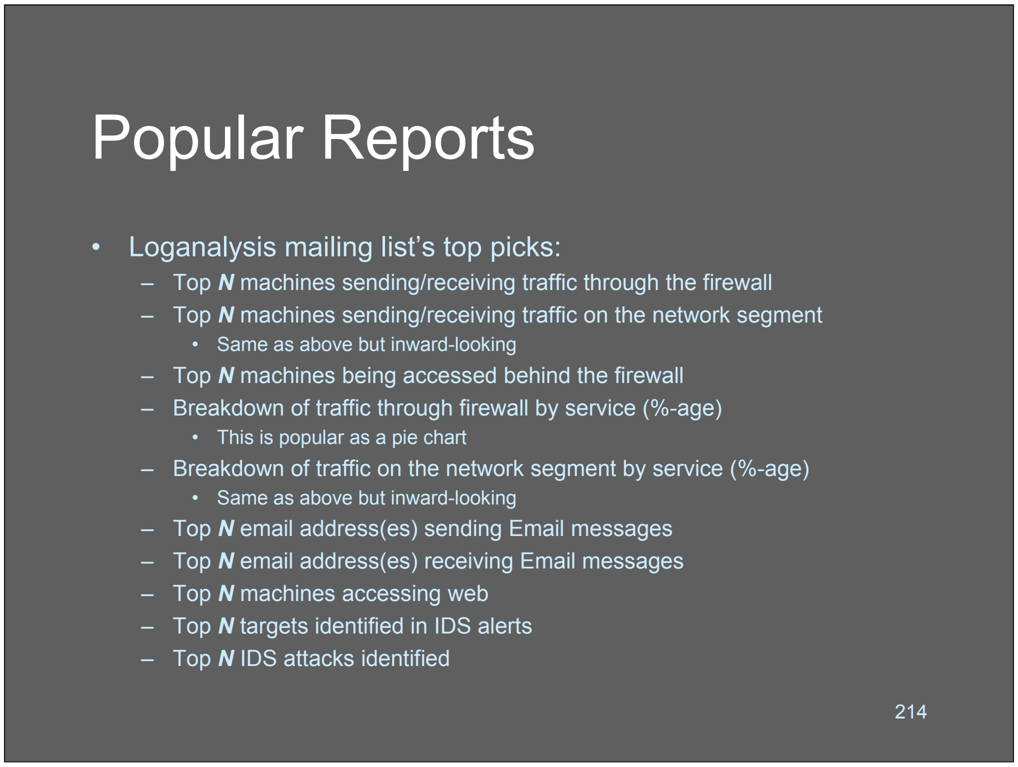
213

If you want to become a Gnuplot master, you need to sit at the feet of the mighty Kawano. He maintains a well-indexed and incredibly detailed Gnuplot problem solver page. It covers how to do pretty much anything you'd want to do and has clear well-written examples, too!

The GnuPlot homepage has links to a few other sites that are doing ultra-cool Gnuplot plots.







## Popular Reports

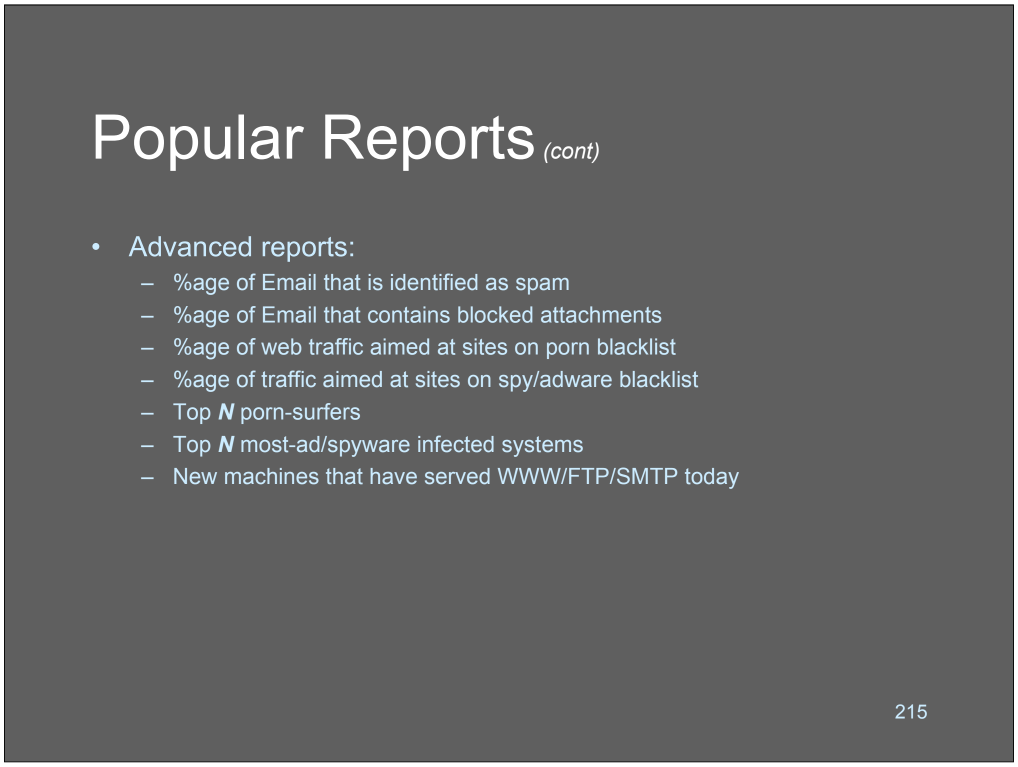
- Loganalysis mailing list's top picks:
  - Top ***N*** machines sending/receiving traffic through the firewall
  - Top ***N*** machines sending/receiving traffic on the network segment
    - Same as above but inward-looking
  - Top ***N*** machines being accessed behind the firewall
  - Breakdown of traffic through firewall by service (%-age)
    - This is popular as a pie chart
  - Breakdown of traffic on the network segment by service (%-age)
    - Same as above but inward-looking
  - Top ***N*** email address(es) sending Email messages
  - Top ***N*** email address(es) receiving Email messages
  - Top ***N*** machines accessing web
  - Top ***N*** targets identified in IDS alerts
  - Top ***N*** IDS attacks identified

214

If you're looking into plotting stuff, here's an idea-list of good things to consider.

I polled the loganalysis mailing list and asked for "favorite things to report." These are all great suggestions. You'll notice that there's an intellectual separation between inward-facing reports and "across a boundary" reports - that's a good idea to maintain. Comparing inward-facing traffic to outgoing traffic is sometimes really interesting!

Most of the data to build these suggested reports can be collected from your firewall logs, or from routers (via SNMP) or system logs.



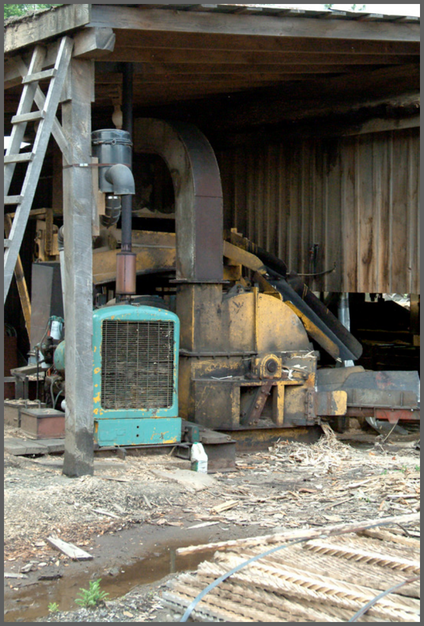
## Popular Reports *(cont)*

- Advanced reports:
  - %age of Email that is identified as spam
  - %age of Email that contains blocked attachments
  - %age of web traffic aimed at sites on porn blacklist
  - %age of traffic aimed at sites on spy/adware blacklist
  - Top ***N*** porn-surfers
  - Top ***N*** most-ad/spyware infected systems
  - New machines that have served WWW/FTP/SMTP today

215

These are more advanced reports that are a bit harder to implement. To build some of these you need to match knowledge-bases against your log data (e.g.: a porn blacklist against your firewall/proxy log)  or you will need to potentially engage in multi-event matching (e.g: match a SYN packet on port 80 with an ACK packet instead of an RST) - you might want to dust off a tool like argus or tcpwatch or collect router netflows if you want to implement these.

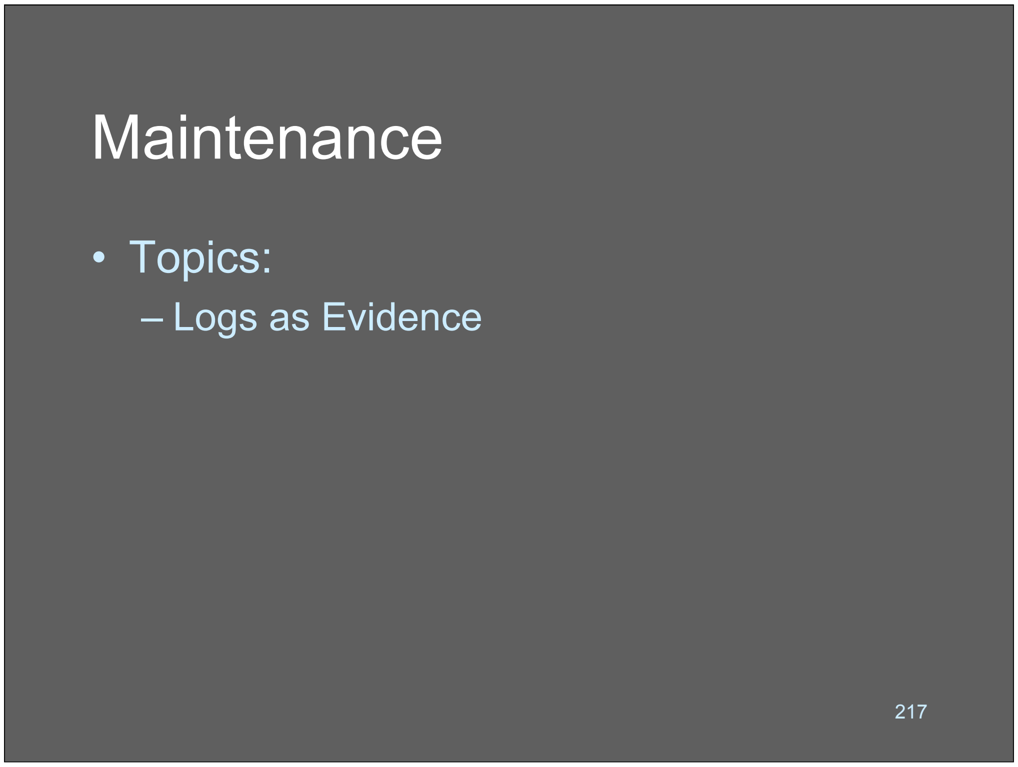
## Maintenance



This is the wood-chipper at Greenwood’s. It’s the source of the mulch pile in our previous sawmill photo. At the far right are the rails of the saw, and the saw table - the saw is at the far end of the shed. When a piece of scrap is discarded by the crew they throw it on a conveyor at the lower right that carries it into the yellow-colored chipper housing. Inside there are a bunch of gears with some teeth that will spit out a piece of oak in the form of little shreds. You do NOT want your hand in this device, believe me!

The shredder is powered by its own diesel engine (on the left) which came from an old tractor; it just lunks away all day spinning the shredder. When wood has been shredded, it goes up that pipe to a sorter housing that either sends it to the conveyor to the mulch-pile, or back into the shredder (via the chute) for another pass through the rotating knives.

If you saw the Coen Brothers’ movie “Fargo” - this is what they *should* have used.

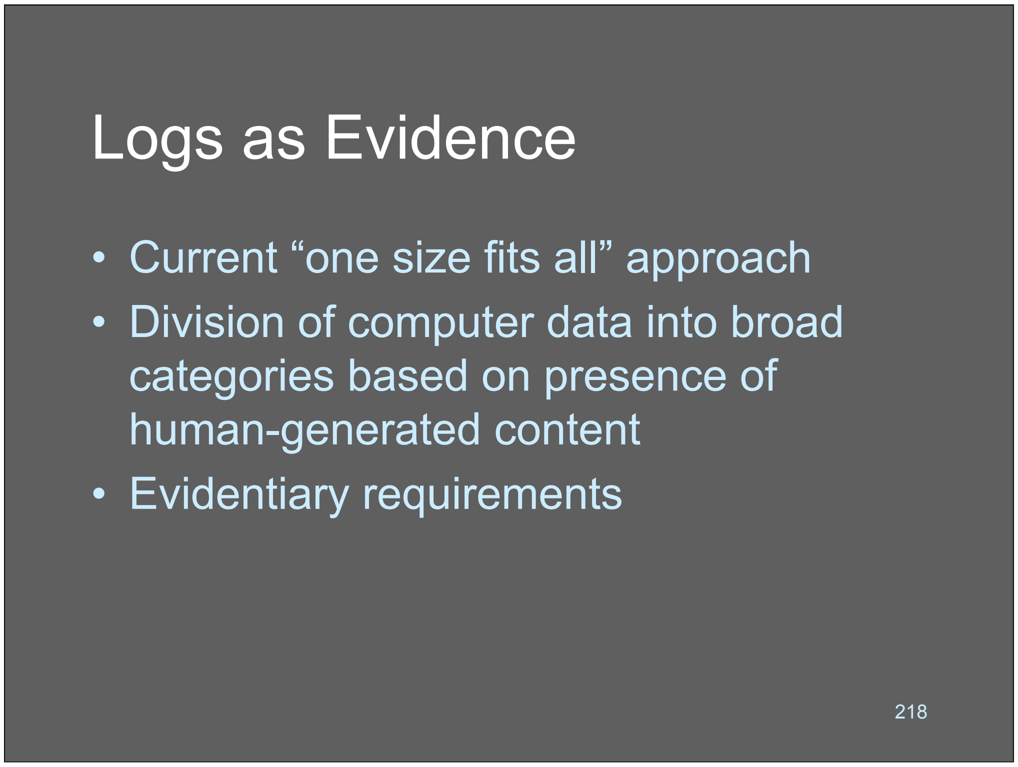


## Maintenance

- Topics:
  - Logs as Evidence

217

Now let’s look at some boring legal stuff. Namely, the problem of dealing with logs as evidence. There’s a lot of misunderstanding on this topic, and it’s important to know what you’re likely to encounter.

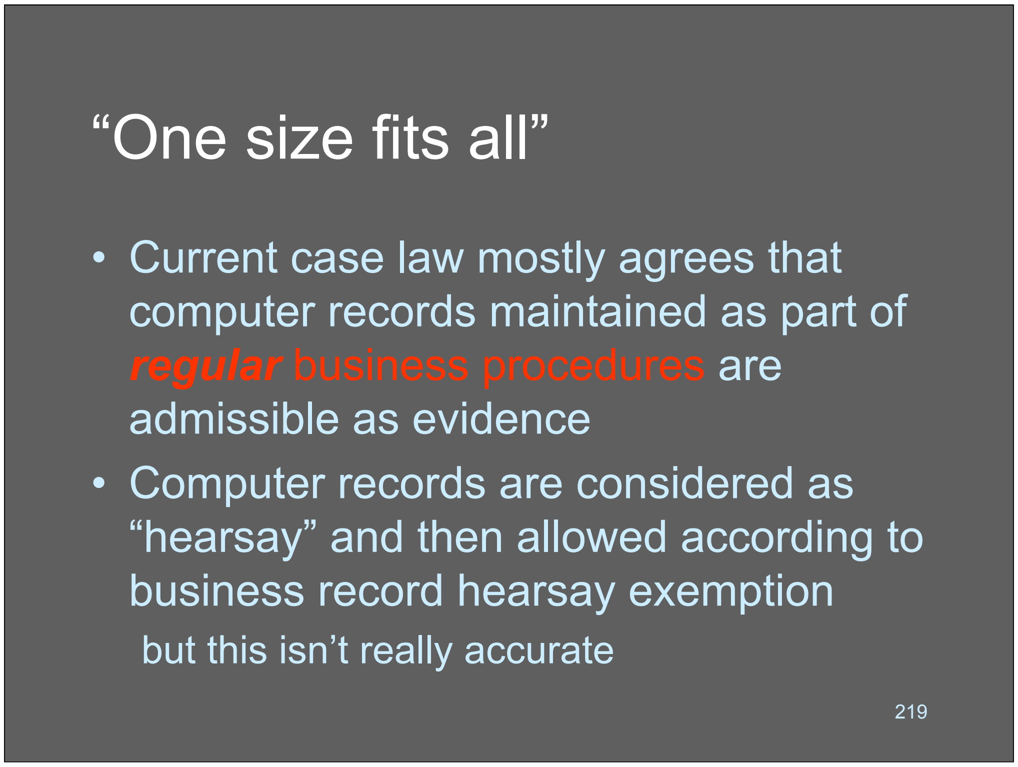


## Logs as Evidence

- Current “one size fits all” approach
- Division of computer data into broad categories based on presence of human-generated content
- Evidentiary requirements

218

These are the topics we’ll discuss regarding the use of logs as evidence!



## "One size fits all"

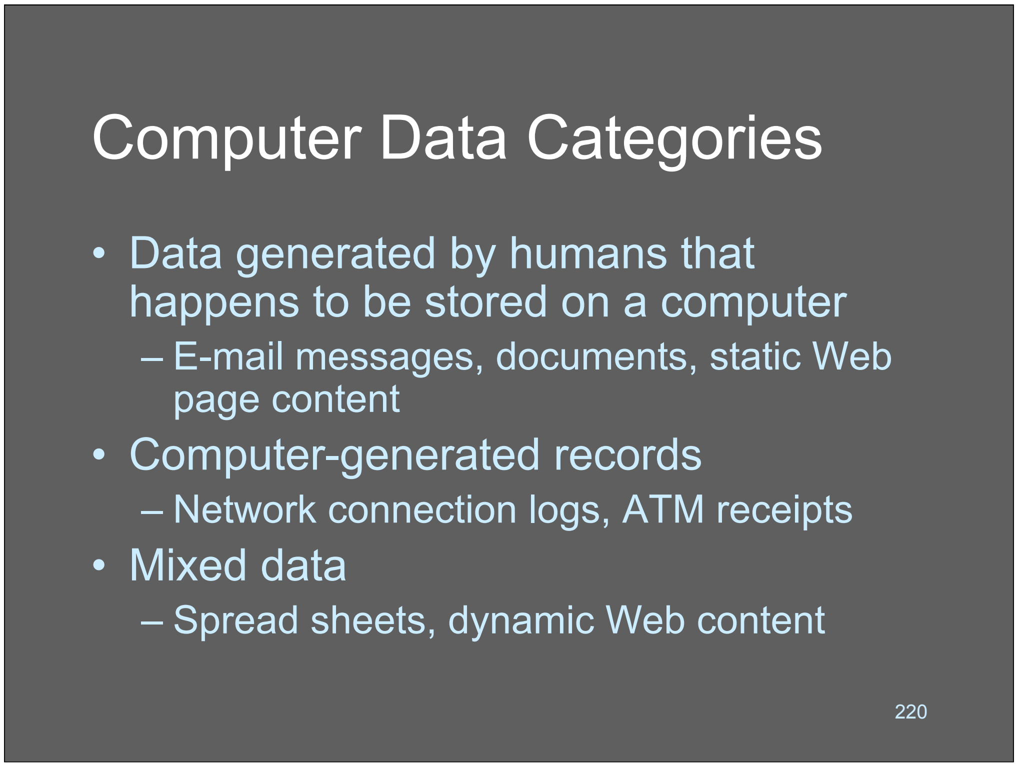
- Current case law mostly agrees that computer records maintained as part of ***regular*** business procedures are admissible as evidence
- Computer records are considered as "hearsay" and then allowed according to business record hearsay exemption

  but this isn't really accurate

219

Many people believe that logs are not "evidence" but are only "heresay" (which is legal-ese for "something a witness said") it turns out that logs can be used as evidence.

One of the many things *I* learned from reading Kerr's report is that hearsay rules do not apply to computer records without human-generated content.  According to the Federal Rules of Evidence, hearsay is defined as information contributed by a human, which must be substantiated before it is allowed to influence the outcome of a court proceeding.  To quote Kerr, "As several courts and commentators have noted, this limitation on the hearsay rules necessarily means that computer-generated records untouched by humans hands cannot contain hearsay."



## Computer Data Categories

- Data generated by humans that happens to be stored on a computer
  - E-mail messages, documents, static Web page content
- Computer-generated records
  - Network connection logs, ATM receipts
- Mixed data
  - Spread sheets, dynamic Web content

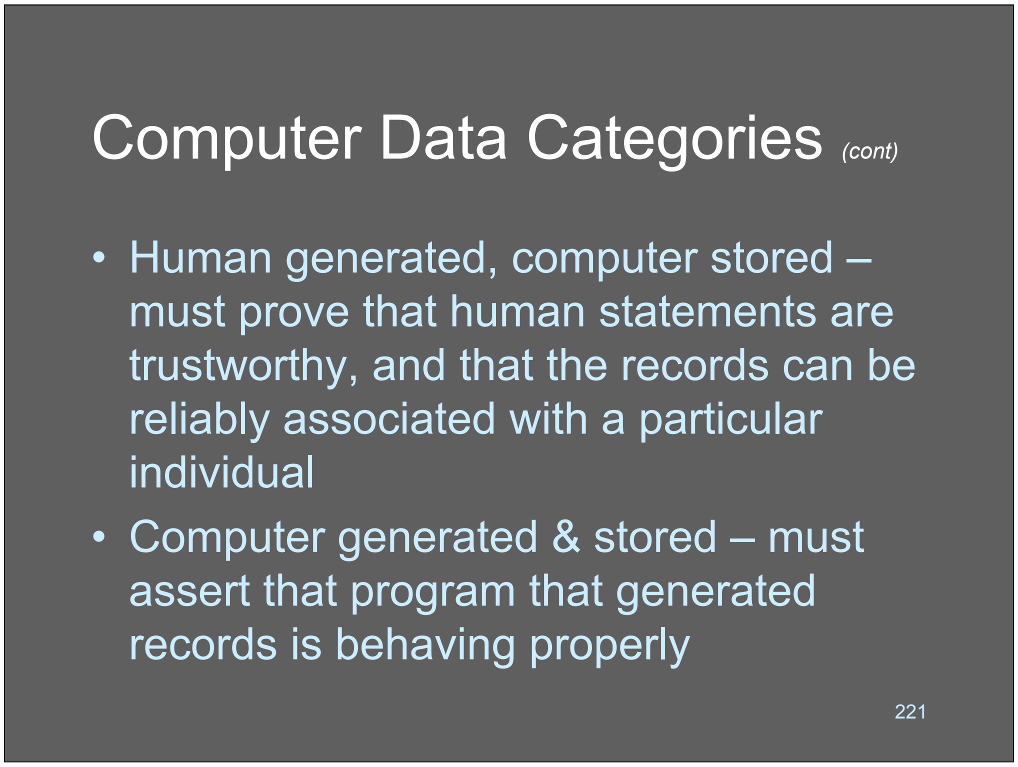
220

Computer data comes in various formats - or at lawyers and judges think so.

Data generated by humans is different from data generated by computers as a matter of their normal function. This is very important when dealing with evidence because with human-generated data the question is: “did the human generate it?” and “is it accurate?” while with computer-generated data the question is “did the process that created this data function correctly?”

Mixed data is something in which human input is acted upon by a computer. So, for example, a spreadsheet might compute a value based on what a corrupt accountant typed into it. The questions, then, are “did the accountant enter the value that caused the result?” and “did the result follow from the value predictably?” (I.e.: was the spreadsheet working right?)

If you demystify this legal gobbledeygook it’s not rocket science.



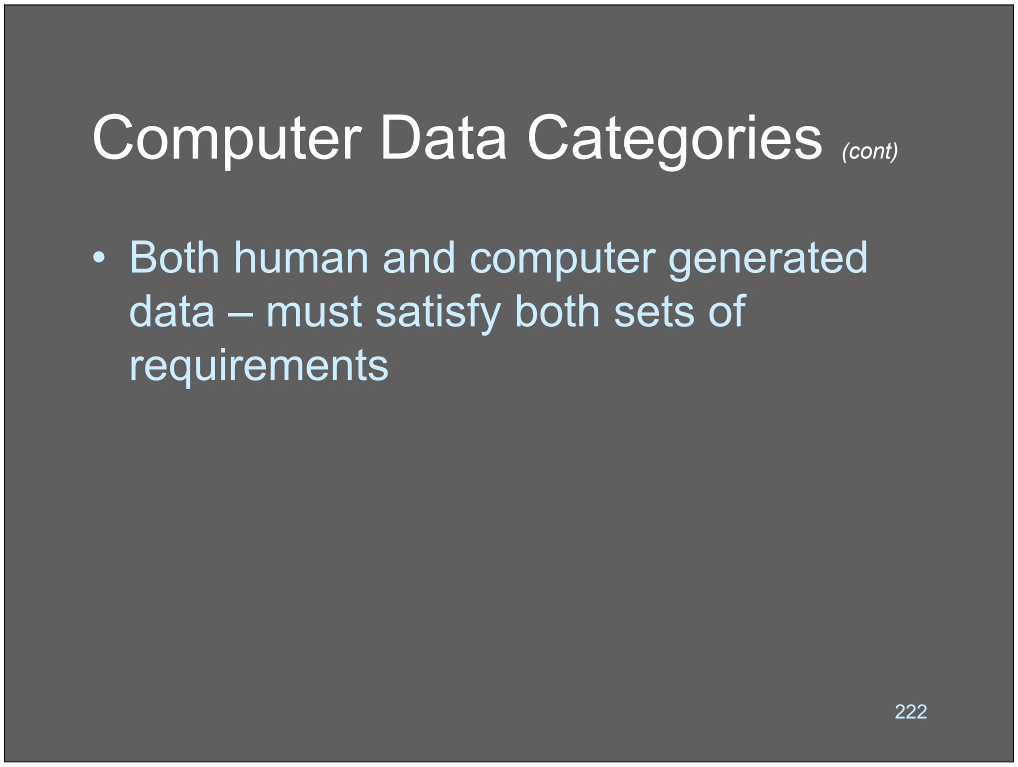
## Computer Data Categories *(cont)*

- Human generated, computer stored – must prove that human statements are trustworthy, and that the records can be reliably associated with a particular individual
- Computer generated & stored – must assert that program that generated records is behaving properly

221

There's also the question of how the data was stored. If you're dealing with a record that stores something a person created, then the main issue is arguing that the storage of the data is reliable and predictable. If you're dealing with computer-generated data (e.g.: a syslog) then the same question applies. You want to be able to argue, "Well, the hacker entered this URL, and it was duly recorded by our web server. Our web server records *all* URLs and it does so reliably and repeatably and has been doing so for 2 years. Once we have the logs from the web server, we copy it for further processing, and don't delete or alter it. This process is also predictable and has been very reliable."

Basically, the defense is going to be trying to challenge the reliability and predictability of your logs. That's a lot easier if you only set them up yesterday. You want your logging architecture to have been in place years ago. But if it were, would you be in this tutorial? ;)

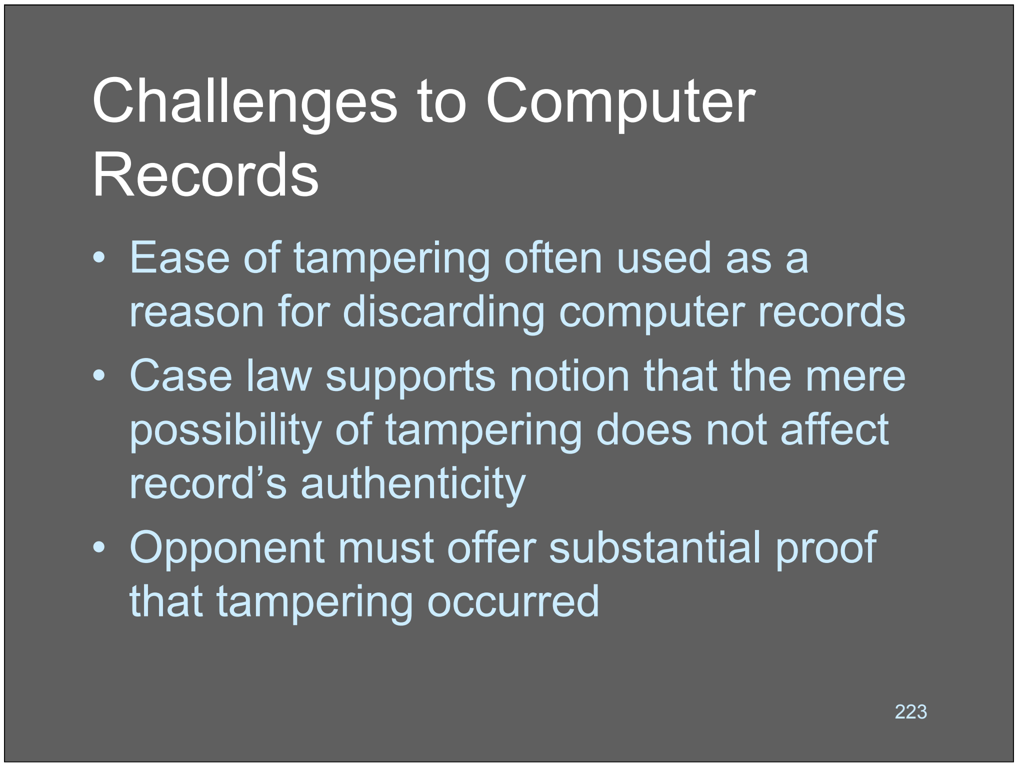


## Computer Data Categories *(cont)*

- Both human and computer generated data – must satisfy both sets of requirements

222

Human and computer generated data - basically - you have the union of the previous requirements. It must be predictable, and reliable.



## Challenges to Computer Records

- Ease of tampering often used as a reason for discarding computer records
- Case law supports notion that the mere possibility of tampering does not affect record's authenticity
- Opponent must offer substantial proof that tampering occurred

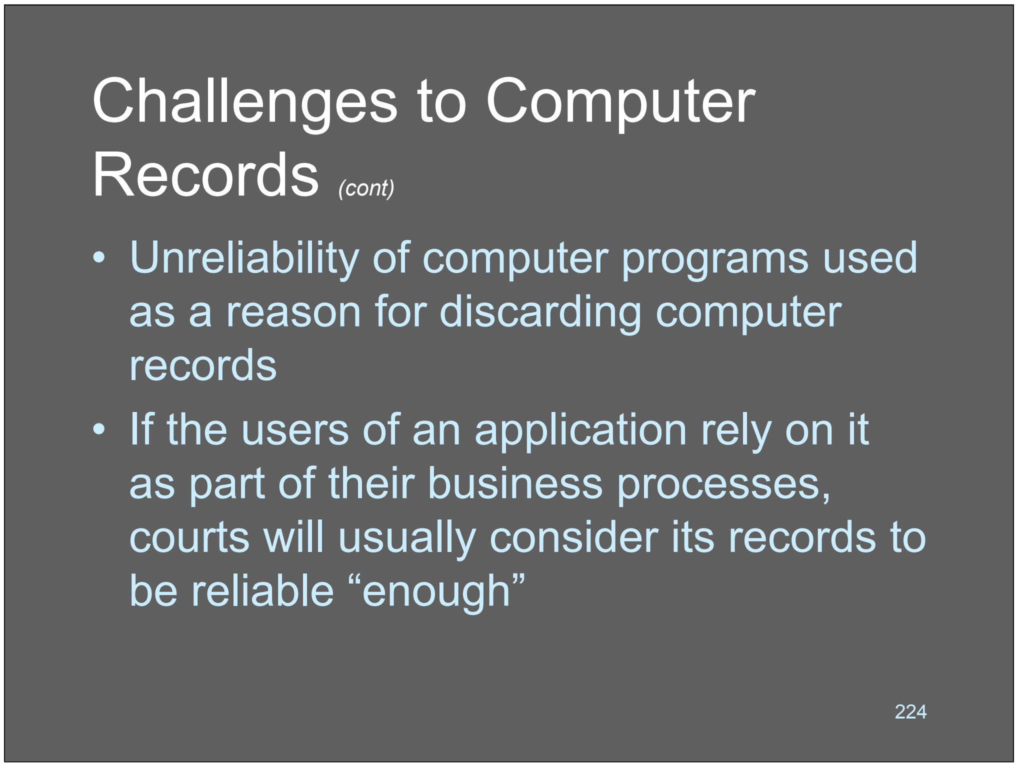
223

There are a lot of urban legends about log data. Some hackers have told me that they figure they could beat a logging rap because they would argue that syslogs are unreliable and can be edited easily with "vi" - unfortunately for them (if they try it) they have watched too much courtroom drama television.

*United States v. Bonallo,* 858 F.2d 1427, 1436 (9th Cir. 1988): "The fact that it is possible to alter data contained in a computer is plainly insufficient to establish untrustworthiness."

Again, what you need to argue is "look, this stuff *works* and has *always worked* and it's *working now* and you're *busted*." Juries and judges are actually pretty sympathetic to the line of argument! If your line of argument is something like, "Our logs (which have always worked fine - see above) led us to believe we needed to see what bob was doing, so we collected every packet out of his system using this industry-standard tool 'tcpdump'" - 'lock and load', hacker-boy is going down.

The converse is true. Don't say, "We set this logging architecture up the day we began to suspect that bob was up to something… We think it works OK."



## Challenges to Computer Records *(cont)*

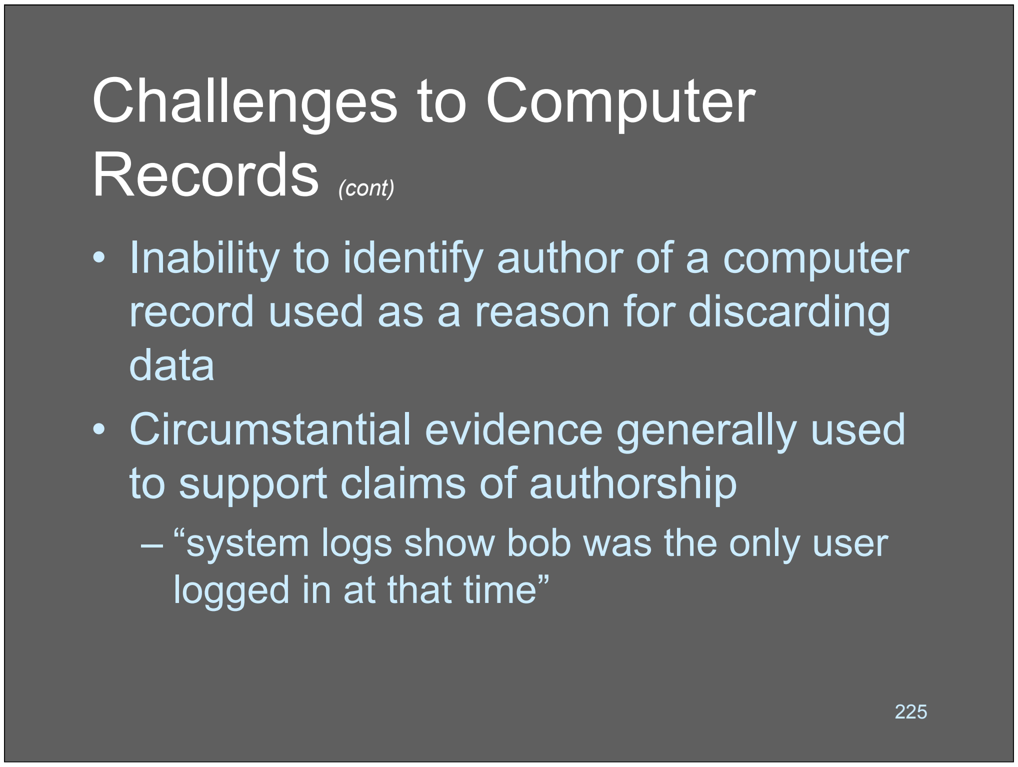
- Unreliability of computer programs used as a reason for discarding computer records
- If the users of an application rely on it as part of their business processes, courts will usually consider its records to be reliable “enough”

224

Again, reliability challenges are made much more difficult if the tool the defense is challenging is part of normal business processes. I.e:

> Defense attorney: “Sir, did you ever consider the possibility that your DHCP server logs were inaccurate?
>
> Sysadmin: “Uh, no, because if it was screwing up the leases it said it was giving, our whole network would have crashed ages ago. Duh!”

*United States v. Moore,* 923 F.2d 910, 915 (1st Cir. 1991): “The ordinary business circumstances described suggest trustworthiness…at least where absolutely nothing in the record in any way implies the lack thereof.”

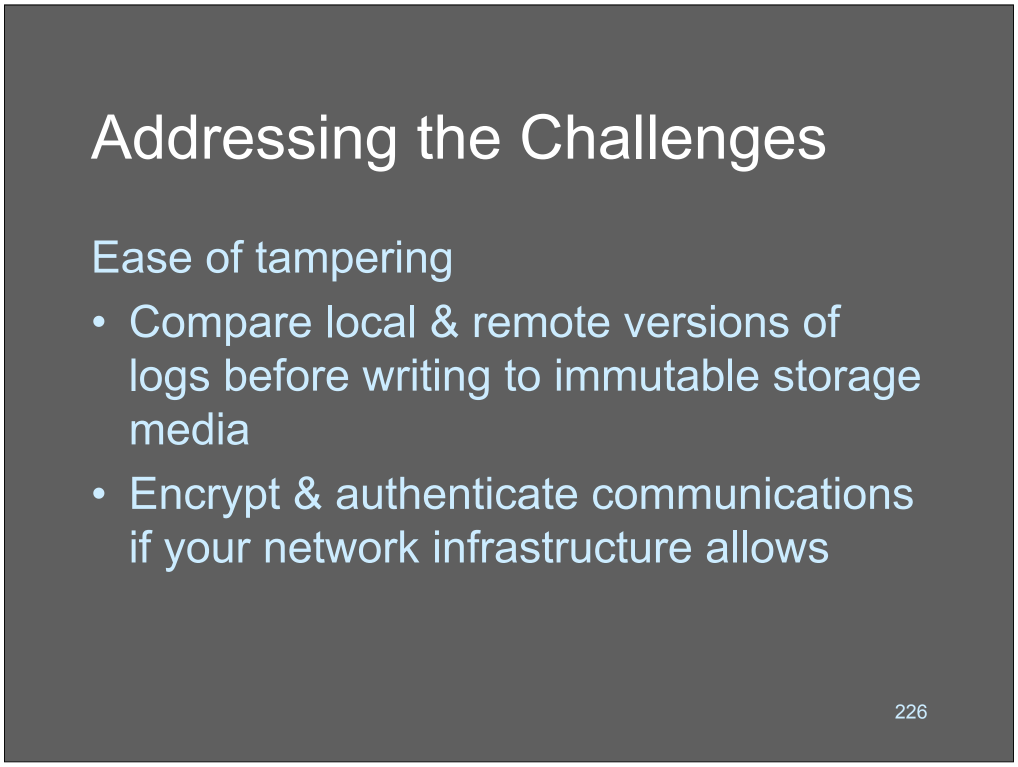

## Challenges to Computer Records *(cont)*

- Inability to identify author of a computer record used as a reason for discarding data
- Circumstantial evidence generally used to support claims of authorship
  - "system logs show bob was the only user logged in at that time"

225

Many people believe that inability to associate a user with a log record is cause to dismiss the log as evidence. Not so! Circumstantial evidence has been sufficient to gain convictions in a number of cases, based on a log entry and a date/time.

I was involved peripherally with a case where a student was accused of breaking into another student's account and dropping them from a class as a malicious stunt. The evidence in the case consisted entirely of a web server log showing that the drops occurred at a specific time and from a specific IP address, coupled with a DHCP leases database dump taken 2 *weeks later* indicating that the accused's MAC address was usually granted that particular IP address. The student was convicted of a felony based on this evidence. I was surprised that the jury found her guilty, and felt that the defense didn't do a good job - they could have asked why the DHCP server logs weren't being kept, whether the DHCP server had been restarted in that time interval (the university's sysadmins didn't *know* if it had!!!) etc. But… Justice might have been done. We'll never know for sure.



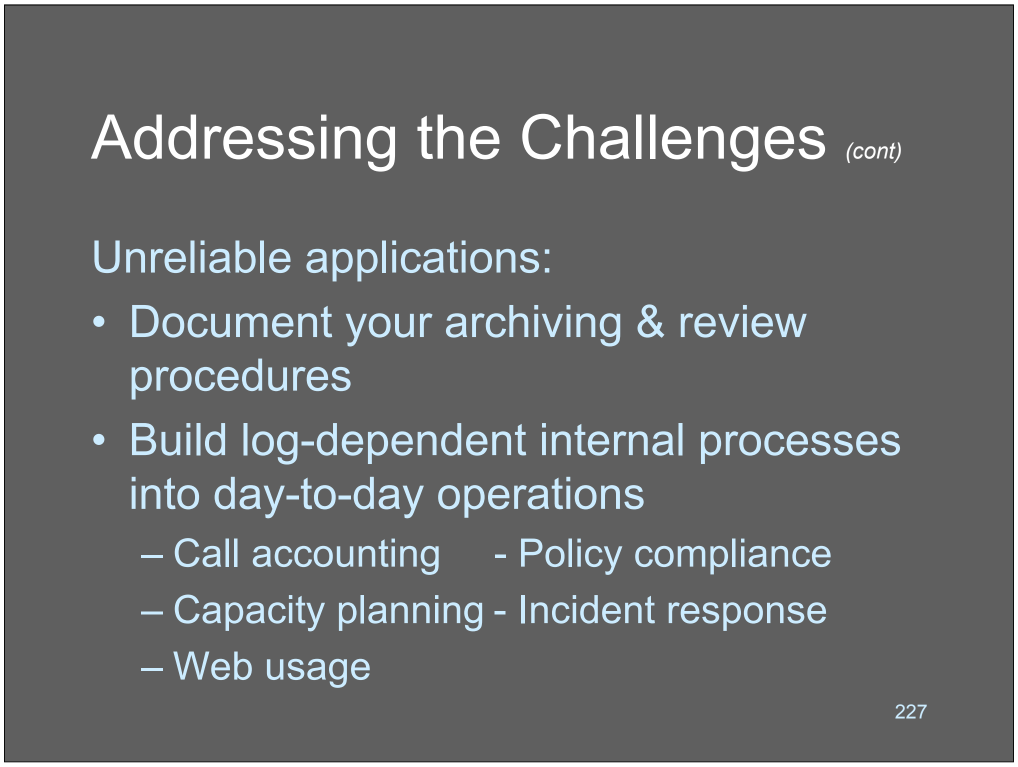
## Addressing the Challenges

Ease of tampering

- Compare local & remote versions of logs before writing to immutable storage media
- Encrypt & authenticate communications if your network infrastructure allows

226

If you wish to address the question of log-tampering, the simple answer is to keep duplicate copies, and checksum them or digitally sign them before committing them to immutable storage media. For most intents and purposes, this is overkill. But it might be fun, anyhow!



## Addressing the Challenges *(cont)*

Unreliable applications:

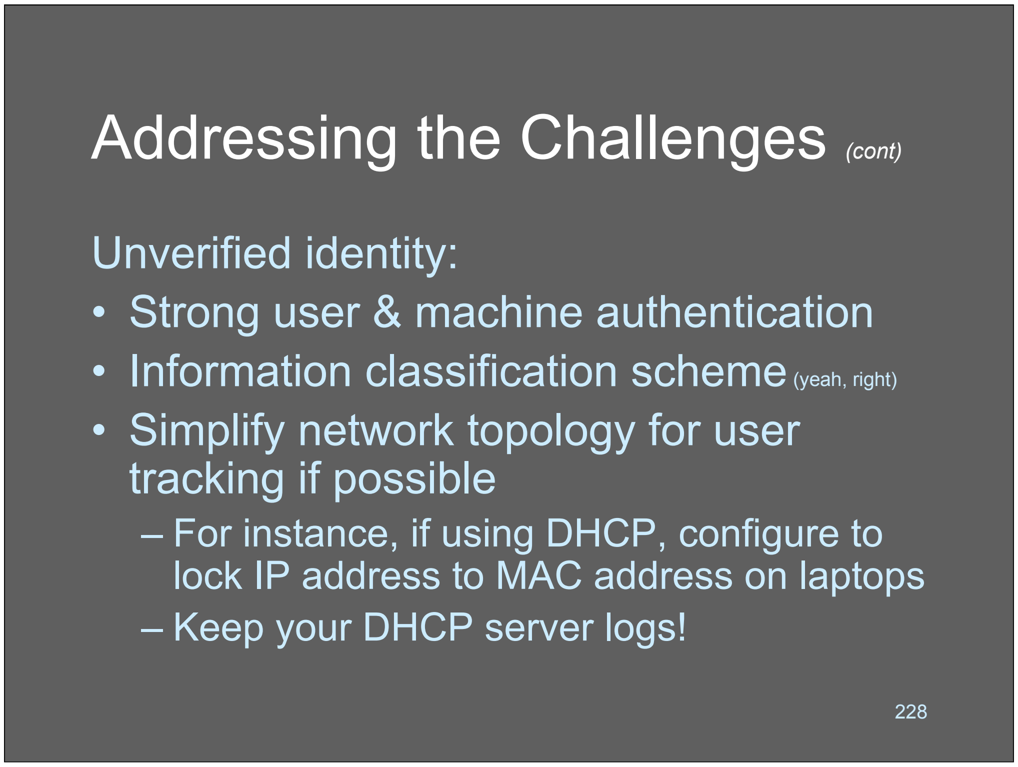
- Document your archiving & review procedures
- Build log-dependent internal processes into day-to-day operations
  - Call accounting - Policy compliance
  - Capacity planning - Incident response
  - Web usage

227

To address the challenge that your log processing may be unreliable, build it into your day-to-day business processes. If you can argue that your NOC team has received an accurate chart of message use every day for a year, it makes it very hard for a defense attorney to generate uncertainty that it was wrong just at the convenient moment.

If you're a paper-trail oriented person, get a directive from your manager ordering you something like this:

`“In order to ensure the security, audit, and function of our network, it is our policy that you aggregate all system logs, store them for 6 months on fast media, and indefinitely on slow media. Further, prepare reports for me daily that report` *`stuff`* `automatically, as an operational capability. In the event you are directed to do so, have in place a procedure whereby you can retain logs that would be scheduled for purging, if they are required for an ongoing investigation. Due to the possible sensitivity of log data, only {` *`short list`* `} authorized security administrators are to have access to the log aggregation system, and all their administrative actions that affect that system should be logged and retained within the system.”`



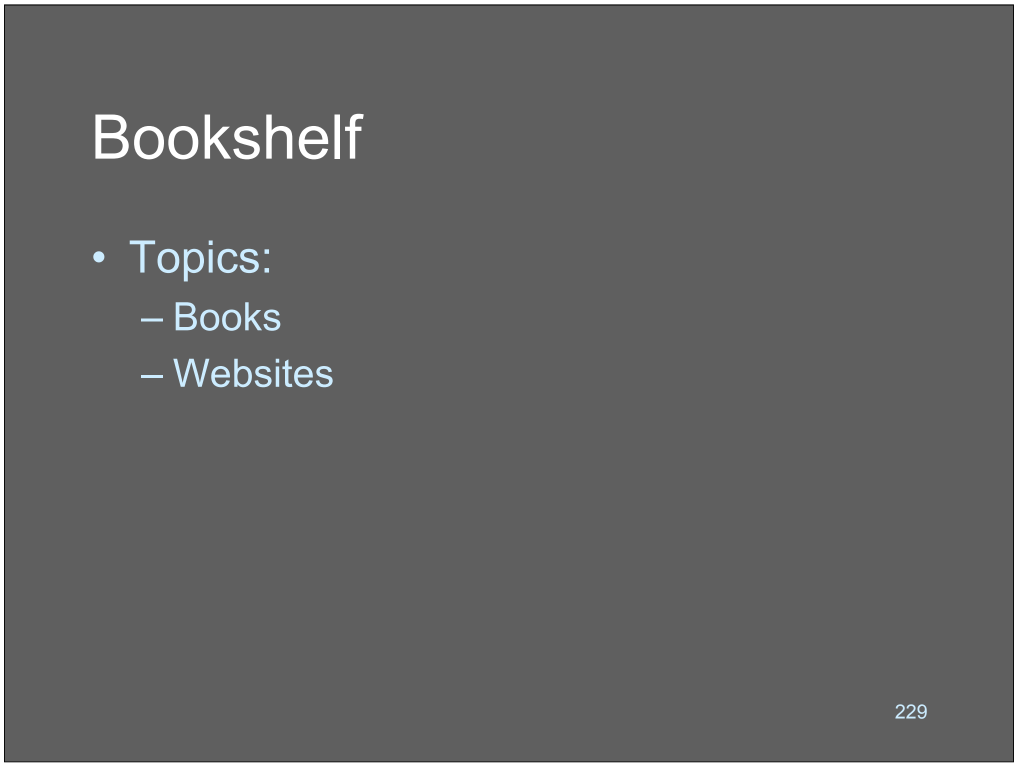
## Addressing the Challenges *(cont)*

Unverified identity:

- Strong user & machine authentication
- Information classification scheme (yeah, right)
- Simplify network topology for user tracking if possible
  - For instance, if using DHCP, configure to lock IP address to MAC address on laptops
  - Keep your DHCP server logs!

228

Unverified identity is the biggest challenge to log data. Make sure that you keep any data you can about who is associated with what. DHCP logs are a *very* important piece of data to keep.

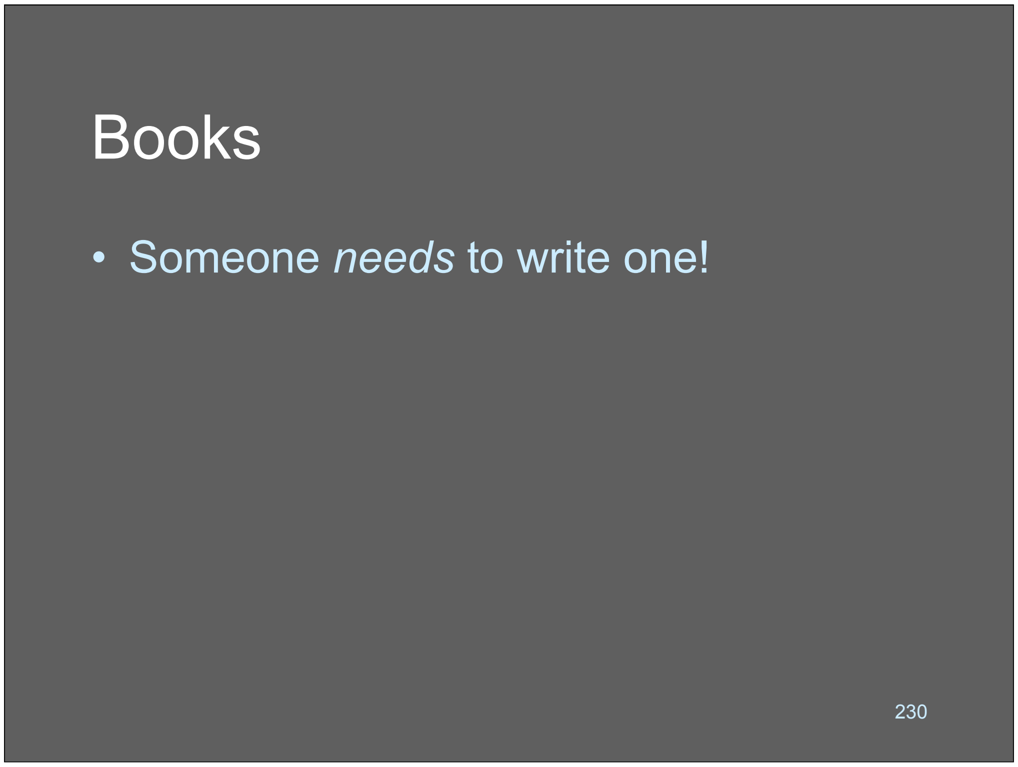


## Bookshelf

- Topics:
  - Books
  - Websites

229

Let’s wrap up with a couple of references and then we’re done!

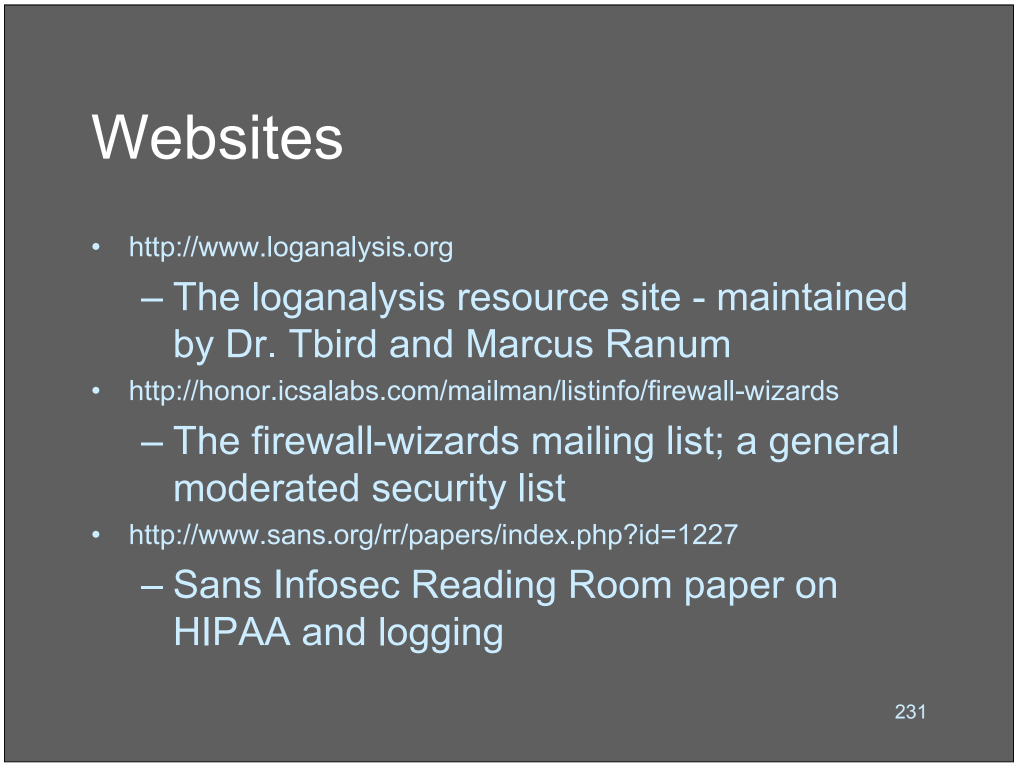


## Books

- Someone *needs* to write one!

230

There aren't any good books on system log analysis at this time.
Someone needs to write one!!



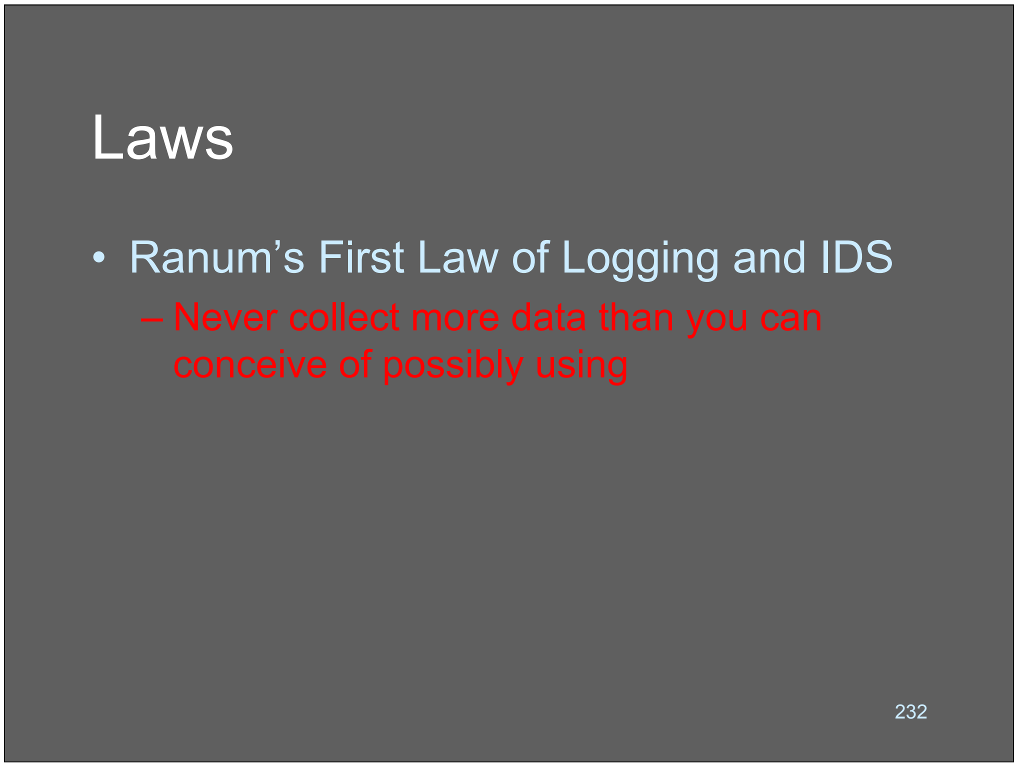
## Websites

- http://www.loganalysis.org
  - The loganalysis resource site - maintained by Dr. Tbird and Marcus Ranum
- http://honor.icsalabs.com/mailman/listinfo/firewall-wizards
  - The firewall-wizards mailing list; a general moderated security list
- http://www.sans.org/rr/papers/index.php?id=1227
  - Sans Infosec Reading Room paper on HIPAA and logging

231

There are a couple of websites around with good information on system logging. Loganalysis.org (OK, I am biassed!) has a lot of links to log-related data. The firewall-wizards mailing list is another deep resource for information about security in general.

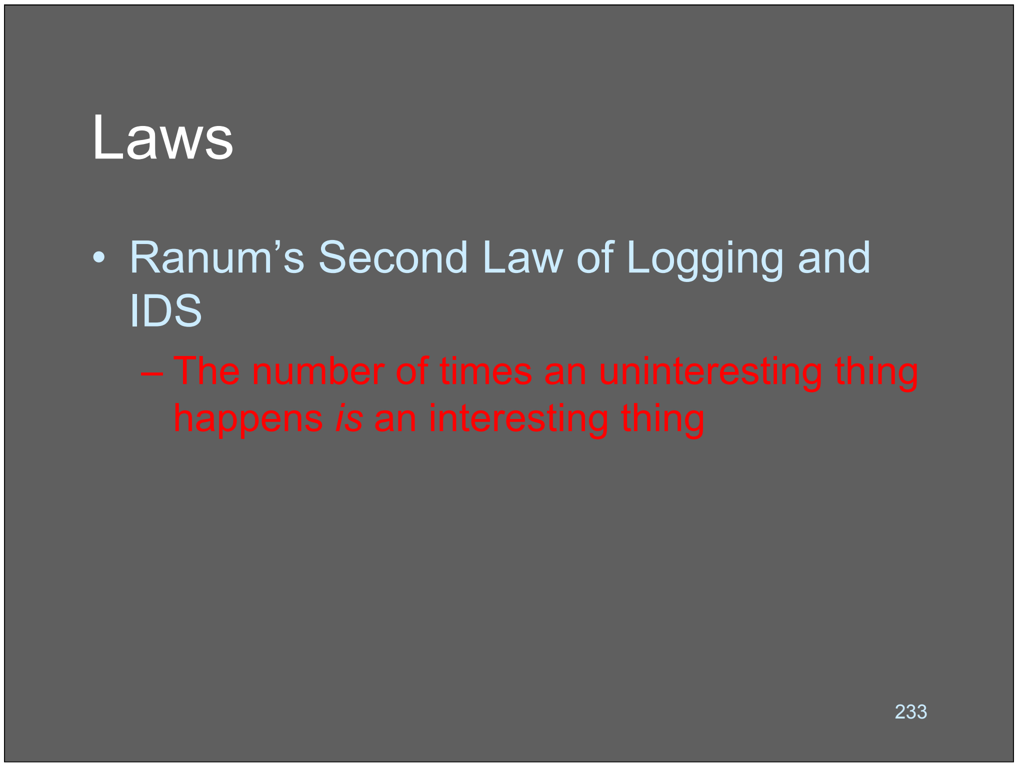
SANS' reading room has some good articles on logging that are worth reading!



## Laws

- Ranum’s First Law of Logging and IDS
  - Never collect more data than you can conceive of possibly using

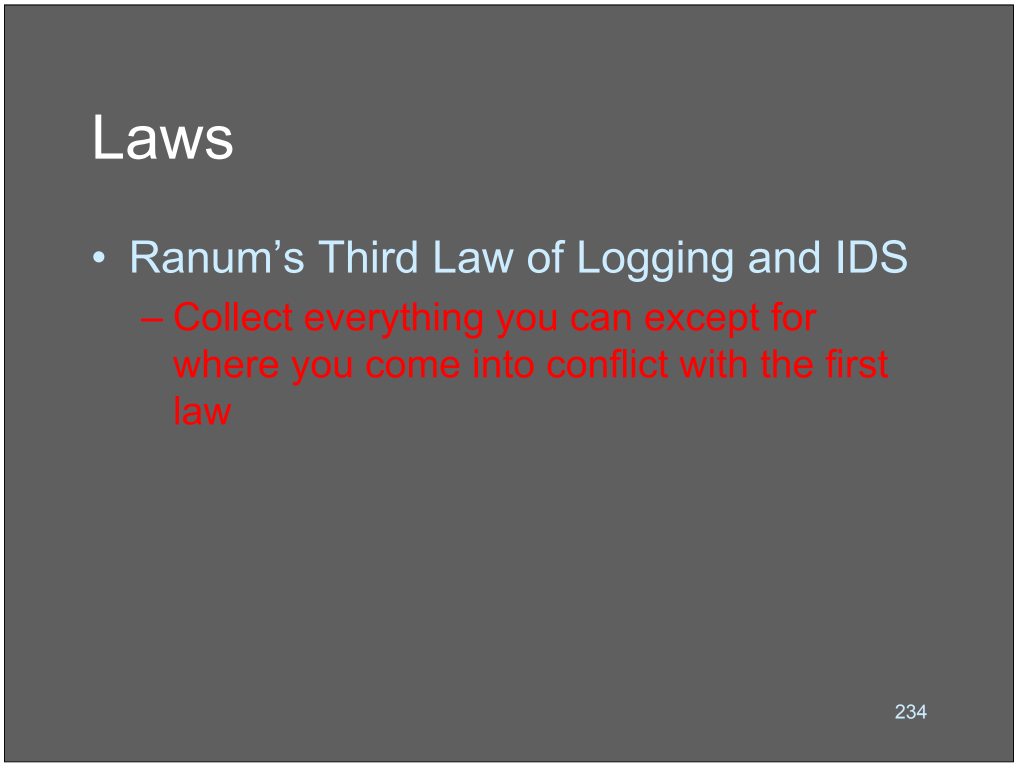
232



## Laws

- Ranum's Second Law of Logging and IDS
  - The number of times an uninteresting thing happens *is* an interesting thing

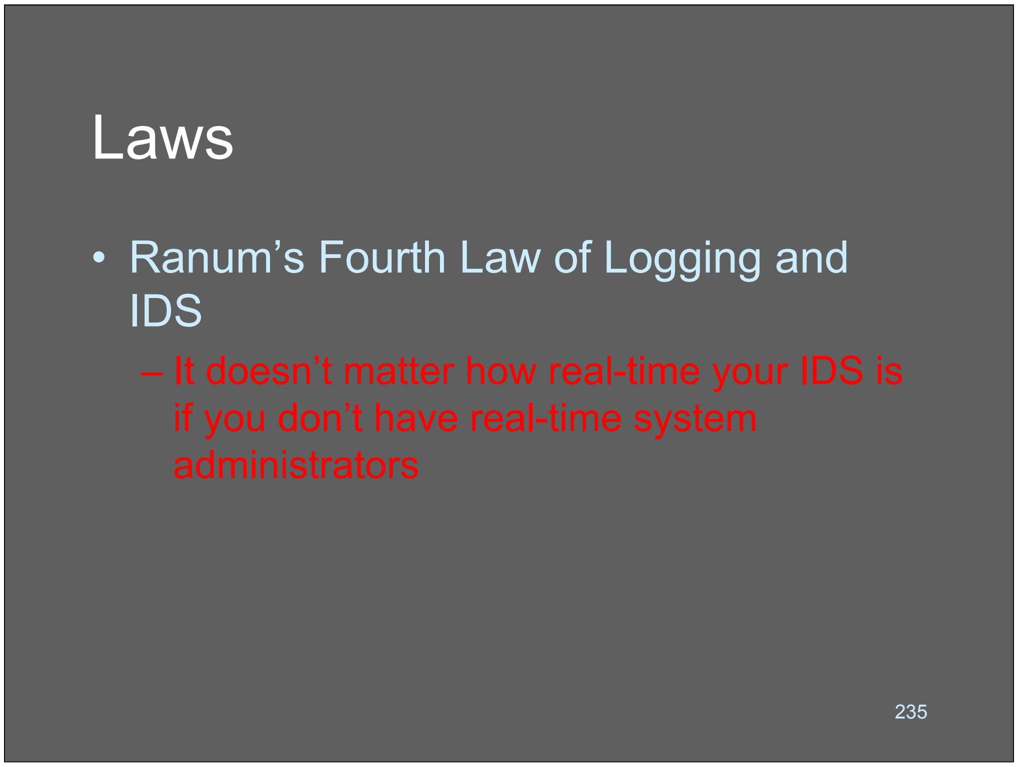
233



## Laws

- Ranum’s Third Law of Logging and IDS
  - Collect everything you can except for where you come into conflict with the first law

234



## Laws

- Ranum’s Fourth Law of Logging and IDS
  - It doesn’t matter how real-time your IDS is if you don’t have real-time system administrators

235