Centralized Logging with syslog-ng and SEC

Distilling Information from Data

Leon Towns-von Stauber
Cascadia IT, March 2012
http://www.occam.com/

Contents

Introduction	3
Example Issues	14
syslog-ng	26
Ye Olden Days	54
Simple Event Correlator	58
Support Tools	120
Review & Future Activities	139

Introduction

→This talk describes an infrastructure that provides:

- →Aggregation of system logs from many UNIX hosts and other network devices
- →Automated analysis of logged events

Introduction

→The benefits of centralized log aggregation and analysis include:

- →Log reduction and correlation reduce the workload associated with viewing logs, making regular review feasible
- →Regular review of logs gives sysadmins a better feel for the computing environment, allows them to spot anomalies more readily
- →Automated analysis and reporting provides early warning of unusual and possibly problematic events
- →Relaying log messages to a secure loghost makes them immune to tampering by a local intruder, permits later forensic analysis

Legal Notices

→This presentation Copyright © 2007-2012 Leon Towns-von Stauber. All rights reserved.

→Trademark notices

→syslog-ng™ is a trademark of BalaBit IT Security. See http://www.balabit.com/trademarks/.

→Solaris™ is a trademark of Oracle. See http://www.oracle.com/us/legal/third-party-trademarks/.

→Other trademarks are the property of their respective owners.

Introduction - Logging Environment

→Loghost

→HP ProLiant DL360 G5

- Two quad-core 2.33-GHz 64-bit Intel Xeon CPUs
- 16 GB RAM
- Two Gigabit Ethernet interfaces (1 used)
- Two 146-GB disks, RAID 1 => 136-GB boot volume
- Fifteen 146-GB disks, RAID 5 => 1.9 TB for log data

→Red Hat Enterprise Linux 4.6

→syslog-ng 2.0.9, SEC 2.4.2

- This host placed in service May 2008, previous server in November 2007

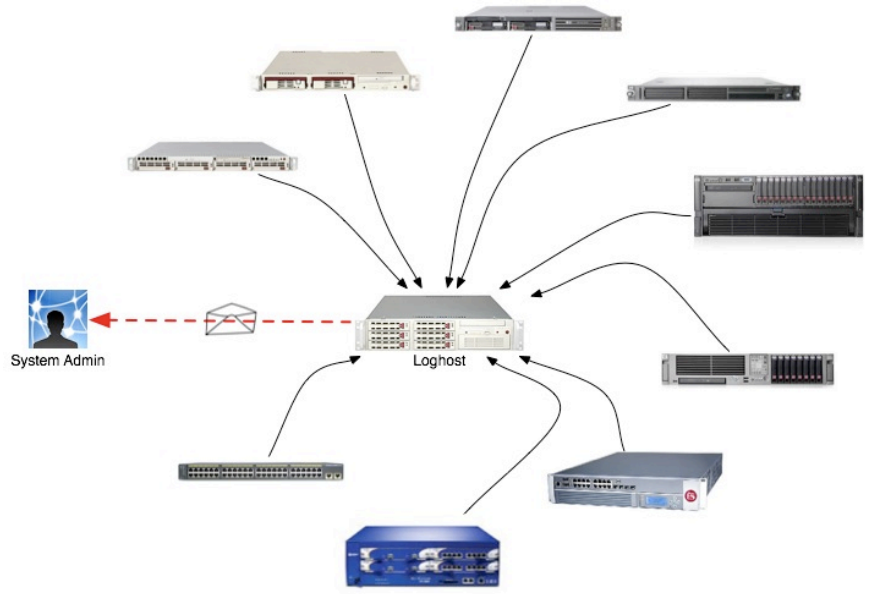
Introduction - Logging Environment

→Clients

→About 400 Red Hat Enterprise Linux hosts

→Over 80 networking devices: F5 BIG-IP load balancers, Juniper NetScreen firewalls and SSL-VPN concentrators, Cisco, Juniper, and Nortel switches, Cisco wireless controllers

Introduction - Logging Environment



Centralized Logging Environment

Introduction

→General approach

→"Artificial ignorance" (Marcus Ranum, 1997)

- http://www.ranum.com/security/computer_security/papers/ai/
- We can't know about everything we'll find interesting in advance
  - Insufficient understanding of environment, new services, rarely seen events, etc.
  - Selective attention to expected events can blind you to the unexpected
- We have a better idea of what's normal and uninteresting
- Analyze everything, toss the boring stuff, look at what's left

Introduction

→General approach

→Process

- Send **all** logs from clients to loghost
- Run **all** logs through filters
  - Suppress routine or unimportant things
  - Use correlation to simplify complex logging events
- Send whatever makes it through to admins on a regular basis
  - Plus realtime alerts for specific known events

Introduction

→A quick detour into systems theory

→Hierarchies of knowledge

→Many variations, but the most common is DIKW, for Data, Information, Knowledge, Wisdom

- Commonly ascribed to Russell Ackoff (1988), but discussed by Milan Zeleny in 1987, and T.S. Eliot wrote this

Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?

T.S. Eliot, “The Rock” (1934)

Introduction

→DIKW

→In the realm of system administration, the hierarchy might break down this way (with explanatory terms by Zeleny)

- Data (“know-nothing”)
  - Example: Most log messages (essentially unusable as-is)
- Information (“know-what”)
  - Examples: Distilled logs, alerts, tickets
- Knowledge (“know-how”)
  - Example: System administration experience
- Wisdom (“know-why”)
  - Example: Senior SA judgment

→Log analysis distills information from log data

Introduction

Raw logs go in...

System administrator

...reduced logs come out

Distilling Information from Data

Example Issues

Example Issues

→To help motivate the discussion, following are descriptions of issues, some trivial and others more serious, discovered through the use of automated log analysis, and the actions taken to resolve them

→The focus is on issues that might have been difficult or impossible to detect with other tools you have in use

Example Issue - Hardware problems

→Loose fan

```
Mar 4 09:37:26 host1.example.com hpasmlited: WARNING: System Fans Not Redundant (Location Power Supply)
Mar 4 09:37:36 host1.example.com hpasmlited: NOTICE: System Fans Not Redundant (Location Power Supply) has
been repaired
Mar 4 09:55:50 host1.example.com hpasmlited: WARNING: System Fans Not Redundant (Location Power Supply)
Mar 4 09:56:00 host1.example.com hpasmlited: NOTICE: System Fans Not Redundant (Location Power Supply) has
been repaired
```

→Broken fan

```
Apr 2 10:00:11 host2.example.com hpasmlited: CRITICAL: Fan Failure (Fan 2, Location CPU)
Apr 2 10:00:11 host2.example.com hpasmlited: WARNING: System Fans Not Redundant (Location CPU)
Apr 2 10:00:21 host2.example.com hpasmlited: NOTICE: Fan Failure (Fan 2, Location CPU) has been repaired
Apr 2 10:00:21 host2.example.com hpasmlited: NOTICE: System Fans Not Redundant (Location CPU) has been
repaired
Apr 2 10:39:13 host2.example.com hpasmlited: CRITICAL: Fan Failure (Fan 2, Location CPU)
Apr 2 10:39:13 host2.example.com hpasmlited: WARNING: System Fans Not Redundant (Location CPU)
```

→Fan reseated or replaced

Example Issue - Orphaned crontabs

→`crond` complaining about `root.cfsaved`

```
Jan 21 16:31:01 host5.example.com crond: (root.cfsaved) ORPHAN (no passwd entry)
Jan 21 16:31:01 host7.example.com crond: (root.cfsaved) ORPHAN (no passwd entry)
Jan 21 16:31:01 host3.example.com crond: (root.cfsaved) ORPHAN (no passwd entry)
Jan 21 16:31:01 host4.example.com crond: (root.cfsaved) ORPHAN (no passwd entry)
Jan 21 16:31:01 host1.example.com crond: (root.cfsaved) ORPHAN (no passwd entry)
Jan 21 16:31:01 host2.example.com crond: (root.cfsaved) ORPHAN (no passwd entry)
```

→When Cfengine updated the `root crontab`, it saved a backup as `root.cfsaved`

→`crond` complained since no user named root.cfsaved exists

→Set `backup=false` in Cfengine config that copies `crontab`

Example Issue - `xinetd` won't start

→Recurring messages

```
Jan 21 17:00:08 host4.example.com cfengine:host4: Executing shell command: /etc/
init.d/xinetd start;/sbin/chkconfig xinetd on
Jan 21 17:00:08 host4.example.com cfengine:host4: (Done with /etc/init.d/xinetd
start;/sbin/chkconfig xinetd on)
```

→Problem in `/etc/sysconfig/network`

→Changed

- `NETWORKING=YES`

→to

- `NETWORKING=yes`

→Who knew that was case-sensitive?

Example Issue - DHCP misconfiguration

→Errors from `dhcpd`

```
Apr 7 12:56:34 host1.example.com dhcpd: /etc/dhcpd/172.27.4.conf line 153: expecting numeric value.
Apr 7 12:56:34 host1.example.com dhcpd: hardware ethernet 00:b0:c7:82:3u:
Apr 7 12:56:34 host1.example.com dhcpd: ^
Apr 7 12:56:34 host1.example.com dhcpd: /etc/dhcpd/172.27.4.conf line 158: expecting numeric value.
Apr 7 12:56:34 host1.example.com dhcpd: /etc/dhcpd.conf line 17: /etc/dhcpd/172.27.4.conf: bad parse.
Apr 7 12:56:34 host1.example.com dhcpd: include "/etc/dhcpd/172.27.4.conf"
Apr 7 12:56:34 host1.example.com dhcpd: ^
Apr 7 12:56:34 host1.example.com dhcpd: Configuration file errors encountered -- exiting
```

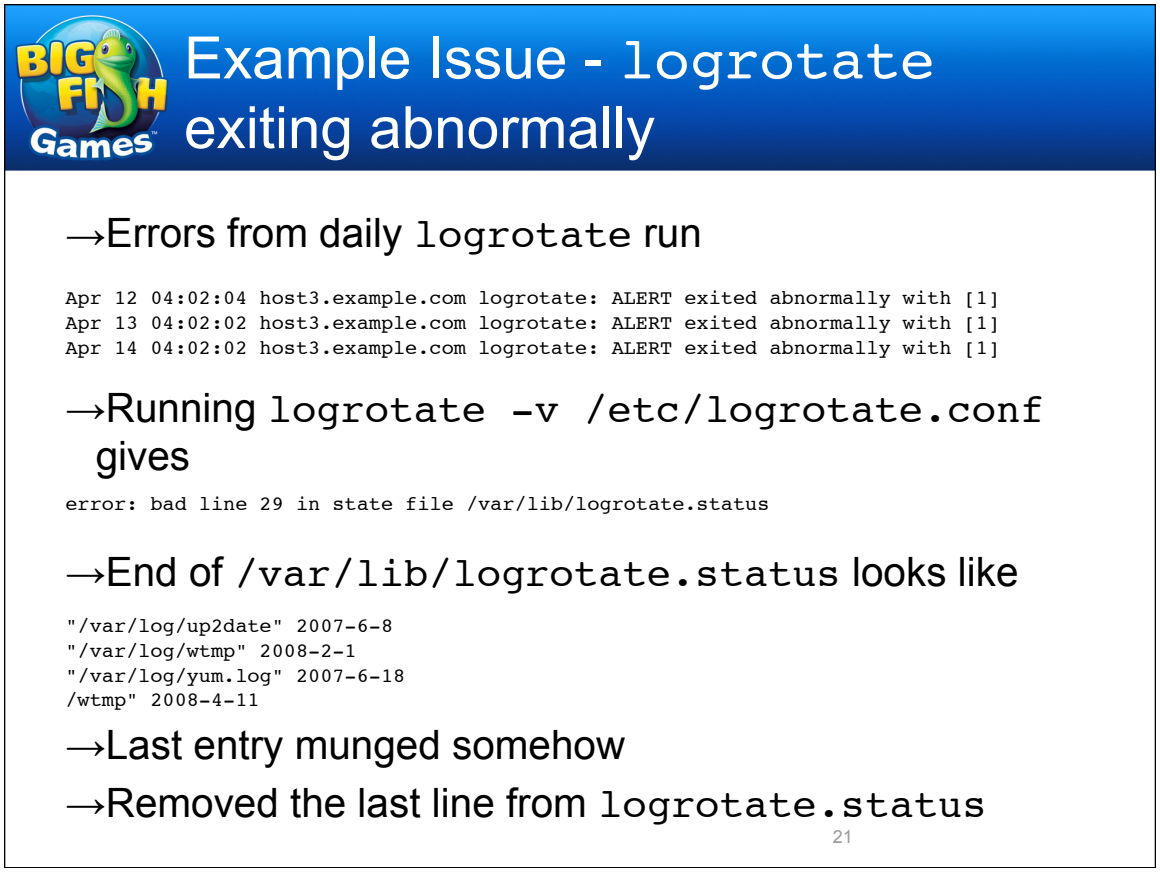
→`3e` mistakenly entered as non-hexadecimal `3u`

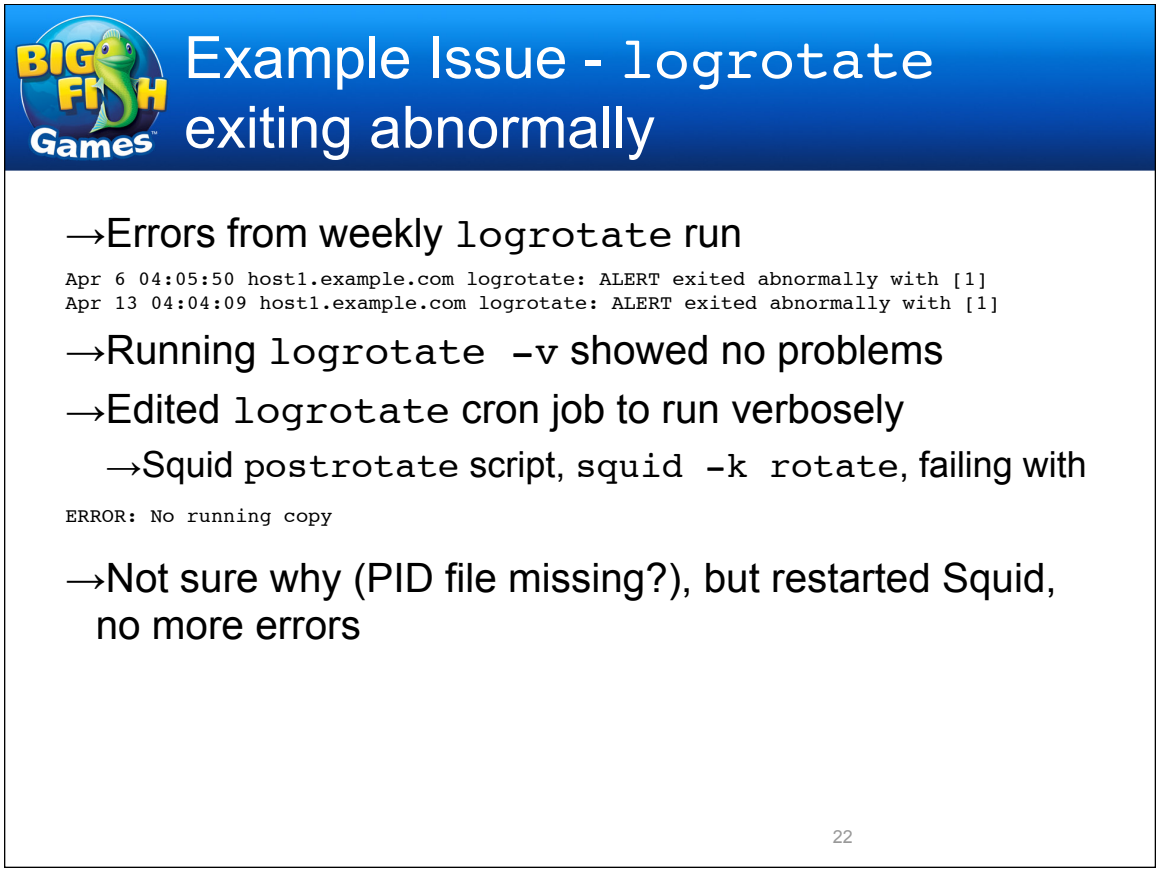
Example Issue - Defunct Realtime Blackhole List

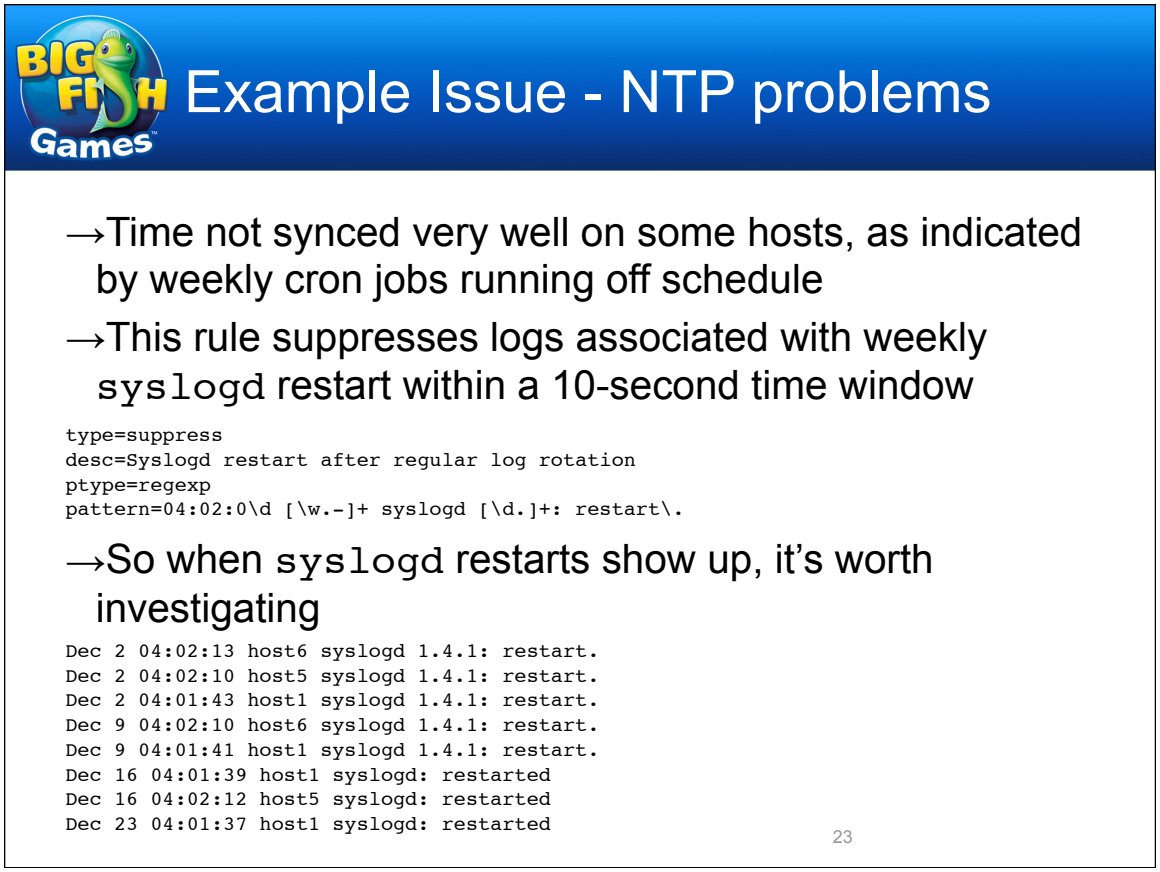
→Postfix errors saying relays.ordb.org had been shut down some time ago

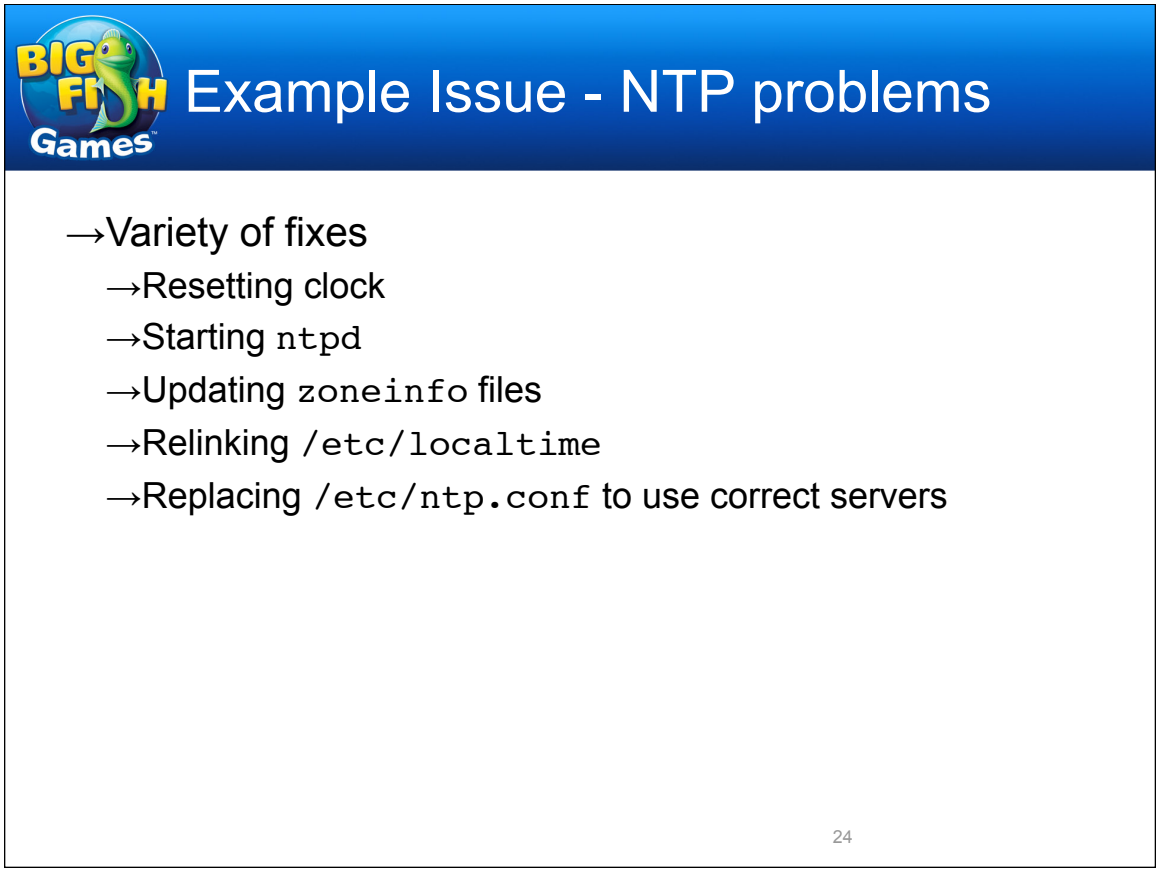
```
Mar 26 23:50:08 host2.example.com postfix/smtpd[31301]: AF8AC159EBA: reject: RCPT
from 201-42-186-68.dsl.telesp.net.br[201.42.186.68]: 554 Service unavailable; Client
host [201.42.186.68] blocked using relays.ordb.org; ordb.org was shut down on
December 18, 2006. Please remove from your mailserver.; from=<Lutz-ataiccar@964-
rock.com> to=<fboskeifanni@example.com> proto=ESMTP
helo=<201-42-186-68.dsl.telesp.net.br>
Mar 26 23:50:10 host2.example.com postfix/smtpd[32311]: 4C00F15A131: reject: RCPT
from server206-35.live-servers.net[213.171.206.35]: 554 Service unavailable; Client
host [213.171.206.35] blocked using relays.ordb.org; ordb.org was shut down on
December 18, 2006. Please remove from your mailserver.;
from=<MatrixMail@NoReturnAddress.uk> to=<ootherworld@addresses.com> proto=SMTP
helo=<[213.171.206.35]>
```

→Removed references to relays.ordb.org from `/etc/postfix/main.cf`, reloaded Postfix


Example Issue - logrotate exiting abnormally
→Errors from daily logrotate run
Apr 12 04:02:04 host3.example.com logrotate: ALERT exited abnormally with [1]
Apr 13 04:02:02 host3.example.com logrotate: ALERT exited abnormally with [1]
Apr 14 04:02:02 host3.example.com logrotate: ALERT exited abnormally with [1]
→Running logrotate -v /etc/logrotate.conf gives
error: bad line 29 in state file /var/lib/logrotate.status
→End of /var/lib/logrotate.status looks like
"/var/log/up2date" 2007-6-8
"/var/log/wtmp" 2008-2-1
"/var/log/yum.log" 2007-6-18
/wtmp" 2008-4-11
→Last entry munged somehow
→Removed the last line from logrotate.status
21


Example Issue - logrotate exiting abnormally
→Errors from weekly logrotate run
Apr 6 04:05:50 host1.example.com logrotate: ALERT exited abnormally with [1]
Apr 13 04:04:09 host1.example.com logrotate: ALERT exited abnormally with [1]
→Running logrotate -v showed no problems
→Edited logrotate cron job to run verbosely
→Squid postrotate script, squid -k rotate, failing with
ERROR: No running copy
→Not sure why (PID file missing?), but restarted Squid, no more errors
22


Example Issue - NTP problems
→Time not synced very well on some hosts, as indicated by weekly cron jobs running off schedule
→This rule suppresses logs associated with weekly syslogd restart within a 10-second time window
type=suppress
desc=Syslogd restart after regular log rotation
ptype=regexp
pattern=04:02:0\d [\w.-]+ syslogd [\d.]+: restart\.
→So when syslogd restarts show up, it's worth investigating
Dec 2 04:02:13 host6 syslogd 1.4.1: restart.
Dec 2 04:02:10 host5 syslogd 1.4.1: restart.
Dec 2 04:01:43 host1 syslogd 1.4.1: restart.
Dec 9 04:02:10 host6 syslogd 1.4.1: restart.
Dec 9 04:01:41 host1 syslogd 1.4.1: restart.
Dec 16 04:01:39 host1 syslogd: restarted
Dec 16 04:02:12 host5 syslogd: restarted
Dec 23 04:01:37 host1 syslogd: restarted
23


Example Issue - NTP problems
→Variety of fixes
→Resetting clock
→Starting ntpd
→Updating zoneinfo files
→Relinking /etc/localtime
→Replacing /etc/ntp.conf to use correct servers
24

Example Issue - DNS probes

→Lots of DNS zone transfer attempts on our external nameservers from a variety of sources

```
Dec 4 17:08:50 MULTIPLE-HOSTS named: PROBE from 12.108.127.137: zone transfer '125.94.64.in-addr.arpa' denied
Dec 4 17:08:52 MULTIPLE-HOSTS named: PROBE from 208.117.131.116: zone transfer 'example.com' denied
Dec 4 17:08:52 MULTIPLE-HOSTS named: PROBE from 129.24.211.26: zone transfer 'example.com' denied
Dec 4 17:08:52 MULTIPLE-HOSTS named: PROBE from 142.150.238.13: zone transfer 'example.com' denied
Dec 4 17:08:53 MULTIPLE-HOSTS named: PROBE from 131.246.191.41: zone transfer 'example.com' denied
```

→Traced to a PlanetLab project described here:

→http://wwwse.inf.tu-dresden.de/SEDNS/SEDNS_home.htm

→Contacted researchers, added our nameservers to exclusion list

syslog-ng

syslog-ng - Intro

- →syslog-ng is a replacement for UNIX `syslogd`, started by Balázs Scheider in 1998
  - →Now also offered in a commercial version by BalaBit
  - →http://www.balabit.com/network-security/syslog-ng/
  - →Central Logging for Unix
    - http://sial.org/talks/central-logging/
- →This talk is based on version 2.0.9
  - →Current open source versions are 3.2.5 and 3.3.4

syslog-ng - Client Setup

- →Clients continue to use stock `syslogd`
  - →They require only one configuration change
- →`/etc/syslog.conf`
  - →Send all logs to loghost
    - `*.debug @loghost`
  - →Here's the full config file used on our Linux hosts:

```
*.info;mail.none;authpriv.none;cron.none   /var/log/messages
authpriv.*                                 /var/log/secure
local7.*                                   /var/log/boot.log

*.emerg                                    *

*.debug                                    @loghost
```

syslog-ng - Client Setup

→/etc/syslog.conf

→Send all logs to loghost

- *.debug @loghost

→Here's what I've used on Solaris hosts:

```
*.err;kern.notice;auth.notice                  /dev/sysmsg
*.err;kern.debug;daemon.notice;mail.crit       /var/adm/messages

*.alert;kern.err;daemon.err                    operator
*.alert                                        root

*.emerg                                        *

*.debug                                        @loghost
```

→Remember to remove the Solaris-default loghost alias to the host itself (in /etc/hosts)

syslog-ng - Server Setup

→Dedicated account for use by syslog-ng

→syslog:x:514:514::/mnt0/syslog:/bin/false

→Locked password

→Group is used by those who need to view logs

→After compiling, installed under /usr/local/

→Created init script

→On Solaris 10, created an SMF service

- /lib/svc/method/syslog-ng
  - Startup script on next slide
- /var/svc/manifest/system/syslog-ng.xml
  - Can provide SMF manifest upon request

→Disabled syslogd

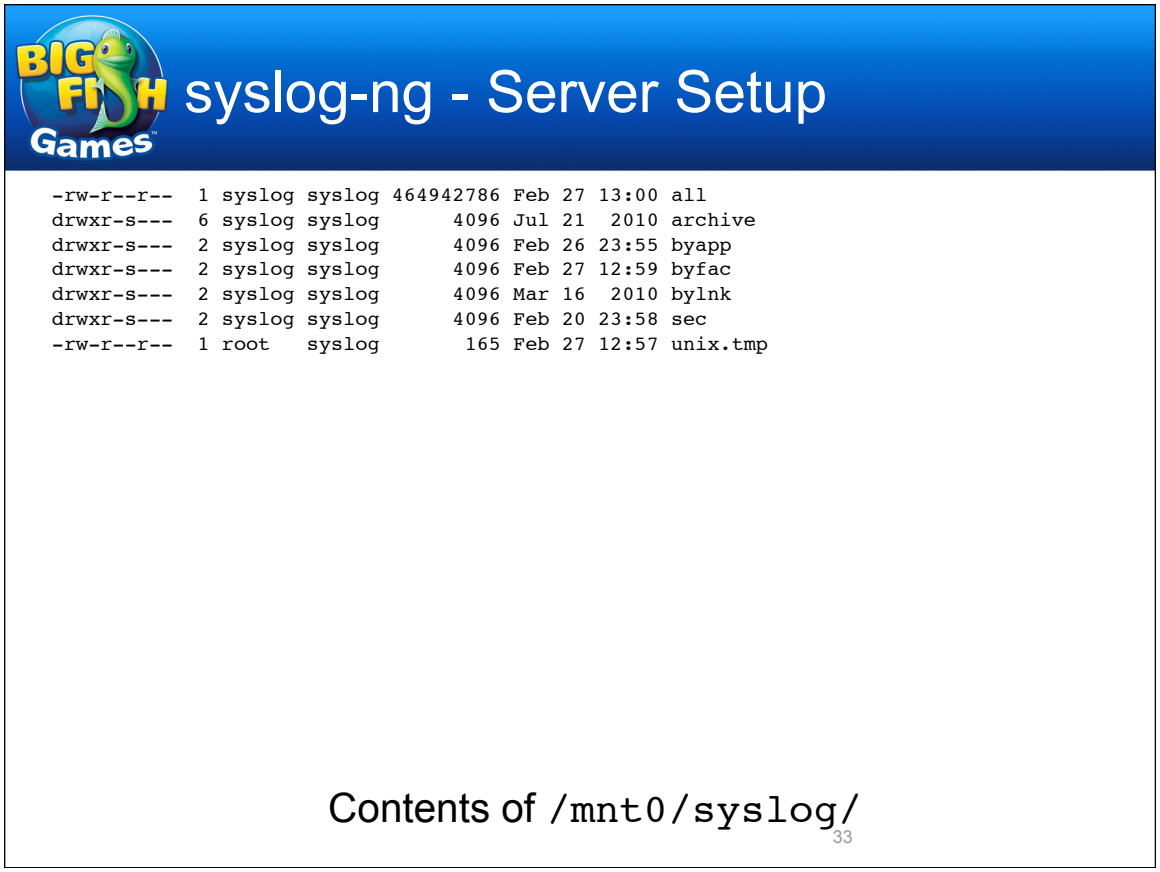
syslog-ng - Server Setup

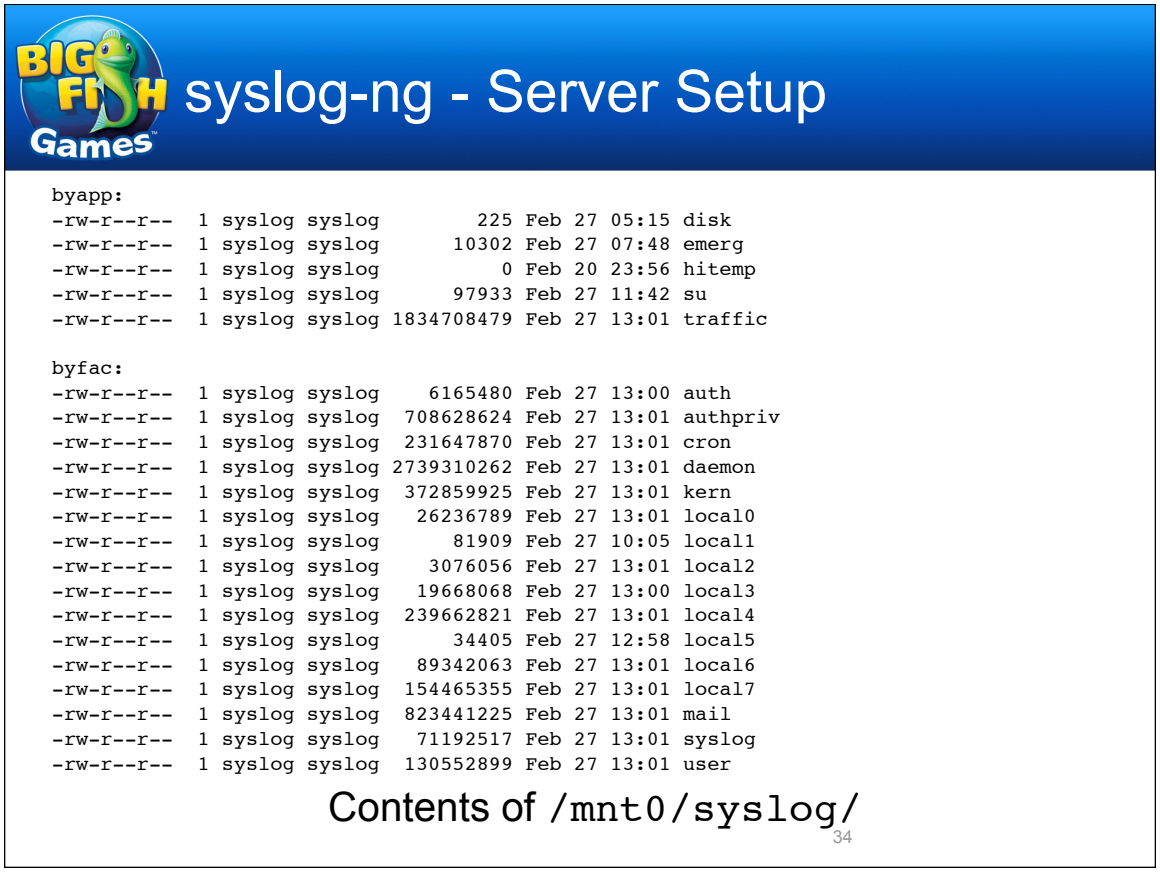
```
#!/sbin/sh

DAEMON=/usr/local/sbin/syslog-ng
USER=syslog
CONFFILE=/usr/local/etc/syslog-ng.conf
PIDFILE=/var/run/syslog-ng.pid

echo 'syslog-ng service starting.'

# Before syslogd starts, save any messages from previous crash dumps so that
# messages appear in chronological order.
/usr/bin/savecore -m
if [ -r /etc/dumpadm.conf ]; then
   . /etc/dumpadm.conf
   [ -n "$DUMPADM_DEVICE" -a "x$DUMPADM_DEVICE" != xswap ] && \
      /usr/bin/savecore -m -f $DUMPADM_DEVICE
fi

$DAEMON -u $USER -f $CONFFILE -p $PIDFILE
```

`/lib/svc/method/syslog-ng`

syslog-ng - Server Setup

- →All the log files are under `/mnt0/syslog/`
  - →The complete record for the day is `all`
  - →The working files used by SEC for regular updates are `net.tmp` and `unix.tmp`
    - These files go away when a regular update is sent out
  - →syslog-ng-filtered logs are in `byfac/` and `byapp/`
    - Some handy symlinks are in `bylnk/`, to help remember what the various `local` facilities (`local1`, `local2`, etc.) are used for
  - →SEC-filtered logs are in `sec/`
  - →Rotated log files are in `archive/`


syslog-ng - Server Setup
-rw-r--r-- 1 syslog syslog 464942786 Feb 27 13:00 all
drwxr-s--- 6 syslog syslog 4096 Jul 21 2010 archive
drwxr-s--- 2 syslog syslog 4096 Feb 26 23:55 byapp
drwxr-s--- 2 syslog syslog 4096 Feb 27 12:59 byfac
drwxr-s--- 2 syslog syslog 4096 Mar 16 2010 bylnk
drwxr-s--- 2 syslog syslog 4096 Feb 20 23:58 sec
-rw-r--r-- 1 root syslog 165 Feb 27 12:57 unix.tmp
Contents of /mnt0/syslog/
33


syslog-ng - Server Setup
byapp:
-rw-r--r-- 1 syslog syslog 225 Feb 27 05:15 disk
-rw-r--r-- 1 syslog syslog 10302 Feb 27 07:48 emerg
-rw-r--r-- 1 syslog syslog 0 Feb 20 23:56 hitemp
-rw-r--r-- 1 syslog syslog 97933 Feb 27 11:42 su
-rw-r--r-- 1 syslog syslog 1834708479 Feb 27 13:01 traffic
byfac:
-rw-r--r-- 1 syslog syslog 6165480 Feb 27 13:00 auth
-rw-r--r-- 1 syslog syslog 708628624 Feb 27 13:01 authpriv
-rw-r--r-- 1 syslog syslog 231647870 Feb 27 13:01 cron
-rw-r--r-- 1 syslog syslog 2739310262 Feb 27 13:01 daemon
-rw-r--r-- 1 syslog syslog 372859925 Feb 27 13:01 kern
-rw-r--r-- 1 syslog syslog 26236789 Feb 27 13:01 local0
-rw-r--r-- 1 syslog syslog 81909 Feb 27 10:05 local1
-rw-r--r-- 1 syslog syslog 3076056 Feb 27 13:01 local2
-rw-r--r-- 1 syslog syslog 19668068 Feb 27 13:00 local3
-rw-r--r-- 1 syslog syslog 239662821 Feb 27 13:01 local4
-rw-r--r-- 1 syslog syslog 34405 Feb 27 12:58 local5
-rw-r--r-- 1 syslog syslog 89342063 Feb 27 13:01 local6
-rw-r--r-- 1 syslog syslog 154465355 Feb 27 13:01 local7
-rw-r--r-- 1 syslog syslog 823441225 Feb 27 13:01 mail
-rw-r--r-- 1 syslog syslog 71192517 Feb 27 13:01 syslog
-rw-r--r-- 1 syslog syslog 130552899 Feb 27 13:01 user
Contents of /mnt0/syslog/
34

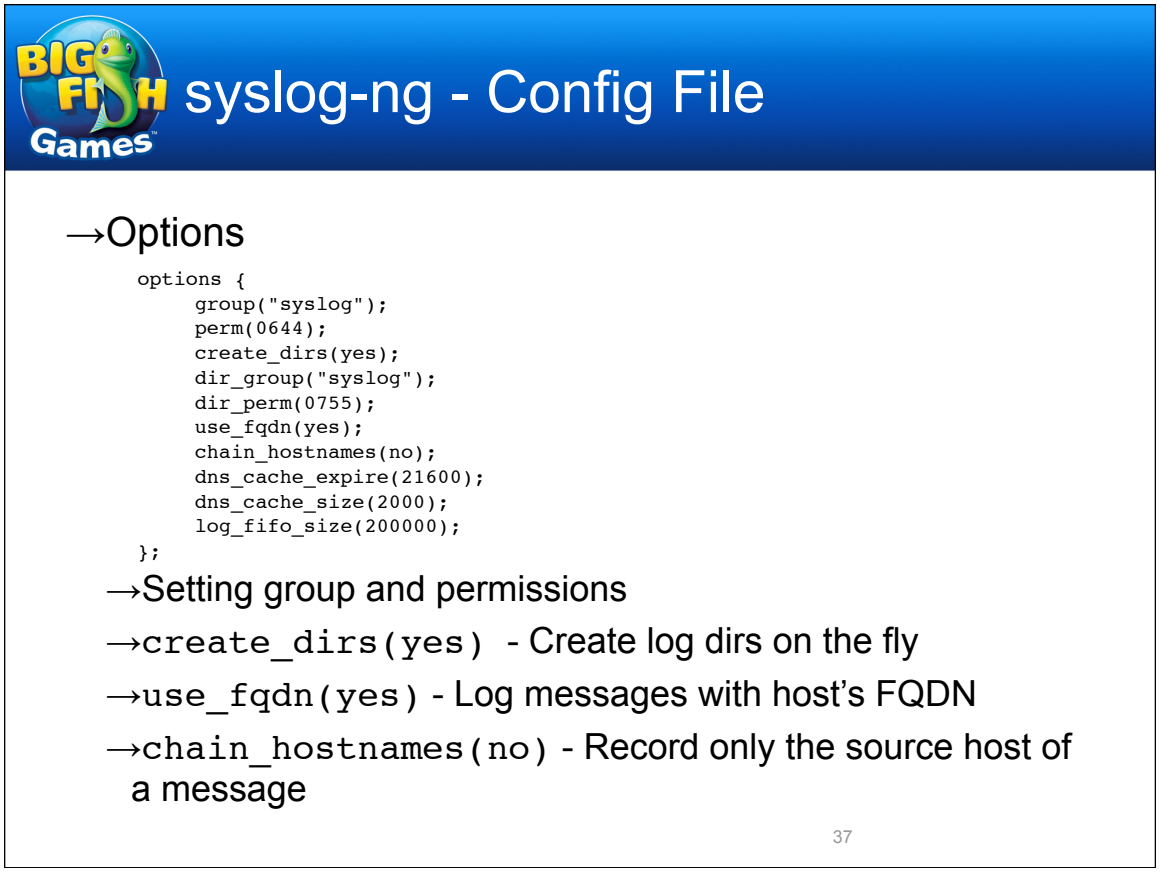
syslog-ng - Server Setup

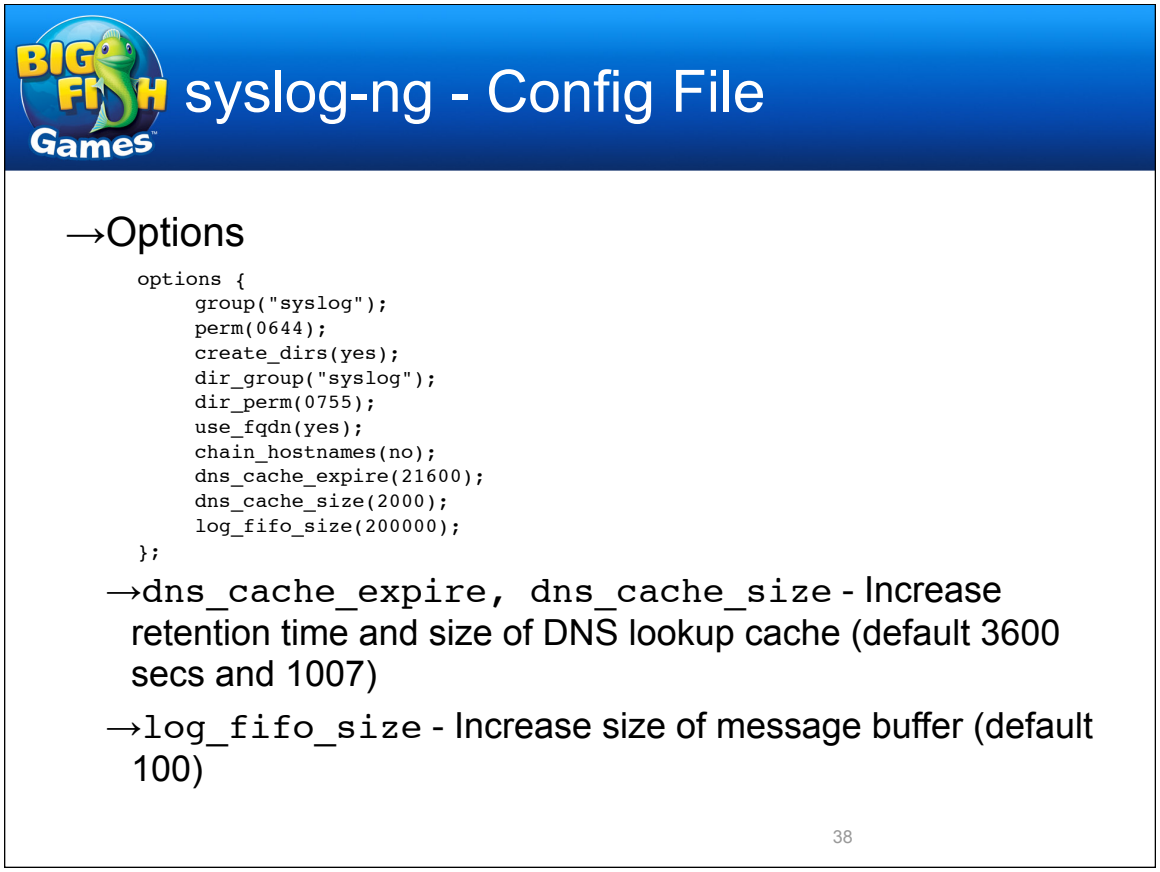
```
bylnk:
lrwxrwxrwx  1 syslog syslog 15 May  1  2008 boot -> ../byfac/local7
lrwxrwxrwx  1 syslog syslog 15 May  1  2008 cisco -> ../byfac/local7
lrwxrwxrwx  1 syslog syslog 15 Jul 29  2008 clamd -> ../byfac/local6
lrwxrwxrwx  1 syslog syslog 15 Feb 12  2009 enviromux -> ../byfac/local0
lrwxrwxrwx  1 syslog syslog 15 Aug 26  2008 juniper -> ../byfac/local7
lrwxrwxrwx  1 root   syslog 15 Mar 16  2010 mysqld -> ../byfac/daemon
lrwxrwxrwx  1 syslog syslog 13 Aug 13  2008 nagios -> ../byfac/user
lrwxrwxrwx  1 syslog syslog 15 May  5  2008 named -> ../byfac/local4
lrwxrwxrwx  1 syslog syslog 13 May  5  2008 netbackup -> ../byfac/user
lrwxrwxrwx  1 syslog syslog 15 May  5  2008 rsyncd -> ../byfac/local3
lrwxrwxrwx  1 syslog syslog 15 May  1  2008 sec -> ../byfac/local1
lrwxrwxrwx  1 syslog syslog 15 May  5  2008 slapd -> ../byfac/local4
lrwxrwxrwx  1 syslog syslog 17 May  1  2008 sudo -> ../byfac/authpriv
lrwxrwxrwx  1 syslog syslog 15 Apr 27  2009 ups -> ../byfac/local2
```

Contents of `/mnt0/syslog/`

syslog-ng - Config File

- →Config file is `/usr/local/etc/syslog-ng.conf`
- →The config file has 5 kinds of statements
  - →General options
  - →Sources and destinations
  - →Filters
  - →Log statements, where you direct messages from sources to destinations through filters
- →I use this configuration for rough filtering and message routing, and to launch SEC processes for further, finer-grained parsing
  - →Also, SEC can't filter based on facility or severity unless they're included in the message text, so syslog-ng is useful for that


syslog-ng - Config File
→Options
options {
group("syslog");
perm(0644);
create_dirs(yes);
dir_group("syslog");
dir_perm(0755);
use_fqdn(yes);
chain_hostnames(no);
dns_cache_expire(21600);
dns_cache_size(2000);
log_fifo_size(200000);
};
→Setting group and permissions
→create_dirs(yes) - Create log dirs on the fly
→use_fqdn(yes) - Log messages with host's FQDN
→chain_hostnames(no) - Record only the source host of a message
37


syslog-ng - Config File
→Options
options {
group("syslog");
perm(0644);
create_dirs(yes);
dir_group("syslog");
dir_perm(0755);
use_fqdn(yes);
chain_hostnames(no);
dns_cache_expire(21600);
dns_cache_size(2000);
log_fifo_size(200000);
};
→dns_cache_expire, dns_cache_size - Increase retention time and size of DNS lookup cache (default 3600 secs and 1007)
→log_fifo_size - Increase size of message buffer (default 100)
38

syslog-ng - Config File

→Here's the first `log` statement

```
log { source(s_all); destination(d_all); };
```

→Every log message received by syslog-ng goes to the `d_all` destination (no filtering)

→Here's the source definition

```
source s_all {
     internal();
     unix-stream("/dev/log");
     udp();
};
```

- Messages are generated internally by syslog-ng, from loghost itself, or from remote clients (via UDP)
- On Solaris, replace `unix-stream` line with:

```
–sun-streams("/dev/log" door("/var/run/syslog_door"));
```


syslog-ng - Config File

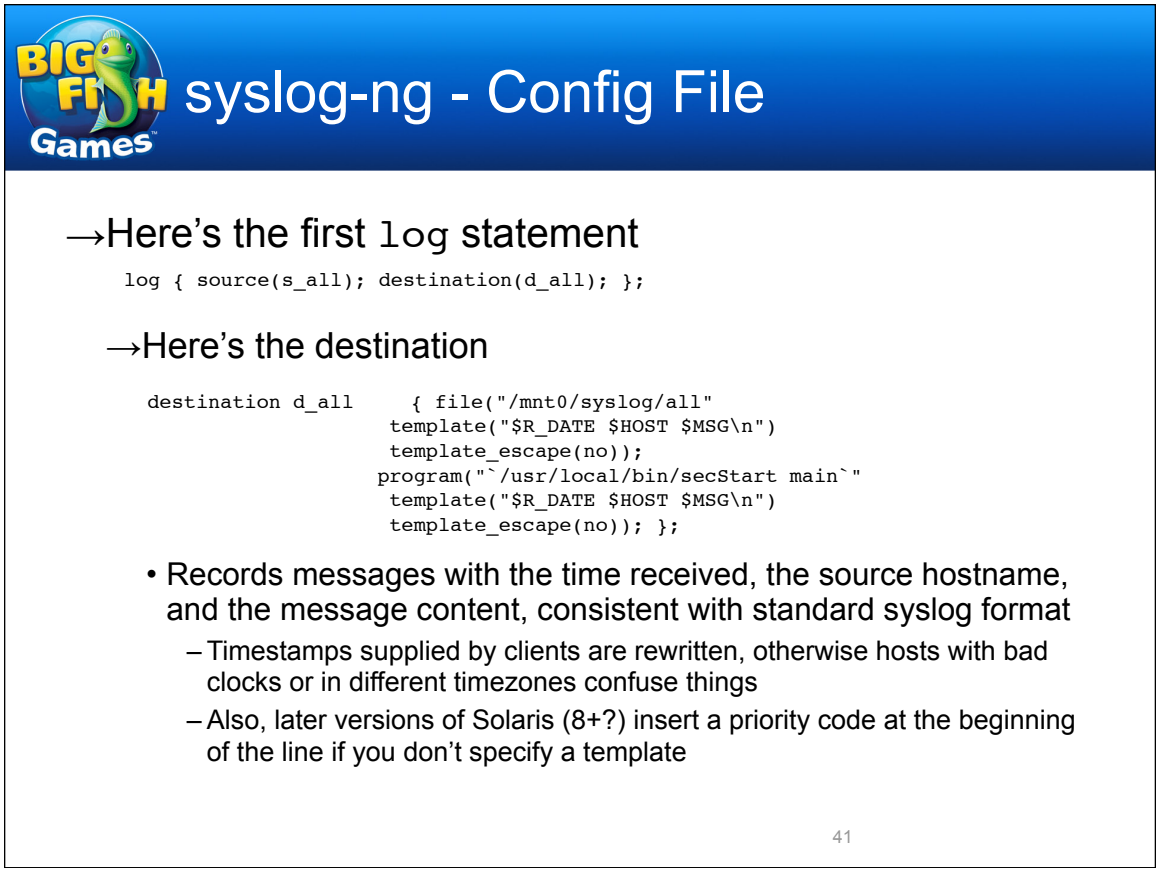
→Here's the first `log` statement

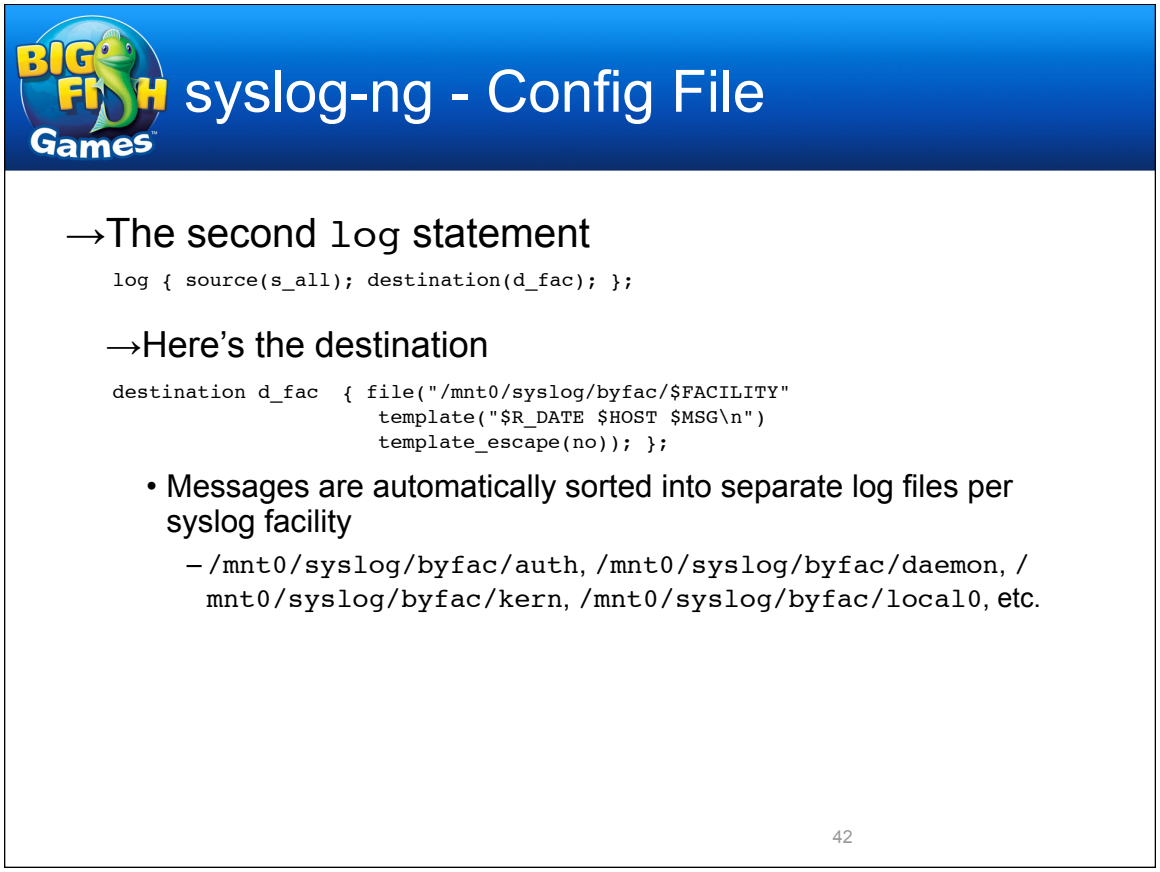
```
log { source(s_all); destination(d_all); };
```

→Here's the destination

```
destination d_all     { file("/mnt0/syslog/all"
                      template("$R_DATE $HOST $MSG\n")
                      template_escape(no));
                     program("`/usr/local/bin/secStart main`"
                      template("$R_DATE $HOST $MSG\n")
                      template_escape(no)); };
```

- All messages are recorded in `/mnt0/syslog/all`
- In addition, when this destination is set up the `secStart` script runs, used to spawn an SEC process to handle the same set of messages
  - More on `secStart` later


syslog-ng - Config File
→Here's the first log statement
log { source(s_all); destination(d_all); };
→Here's the destination
destination d_all { file("/mnt0/syslog/all"
template("$R_DATE $HOST $MSG\n")
template_escape(no));
program("`/usr/local/bin/secStart main`"
template("$R_DATE $HOST $MSG\n")
template_escape(no)); };
• Records messages with the time received, the source hostname, and the message content, consistent with standard syslog format
– Timestamps supplied by clients are rewritten, otherwise hosts with bad clocks or in different timezones confuse things
– Also, later versions of Solaris (8+?) insert a priority code at the beginning of the line if you don't specify a template
41


syslog-ng - Config File
→The second log statement
log { source(s_all); destination(d_fac); };
→Here's the destination
destination d_fac { file("/mnt0/syslog/byfac/$FACILITY"
template("$R_DATE $HOST $MSG\n")
template_escape(no)); };
• Messages are automatically sorted into separate log files per syslog facility
– /mnt0/syslog/byfac/auth, /mnt0/syslog/byfac/daemon, /mnt0/syslog/byfac/kern, /mnt0/syslog/byfac/local0, etc.
42

syslog-ng - Config File

→The third `log` statement introduces the `final` flag, meant to stop further processing if a message makes it through this filter, as the remaining `log` statements are for specific, non-overlapping purposes

```
log { source(s_int); destination(d_int); flags(final); };
```

→Source

```
source s_int { internal(); };
```

→Destination

```
destination d_int  { file("/mnt0/syslog/byfac/syslog"
                template("$R_DATE $HOST $MSG\n")
                template_escape(no)); };
```

- Log internally-generated messages to a separate file
  - Not otherwise logged to a facility-specific file by the previous statement, even though `internal` source included in `s_all`

syslog-ng - Config File

→The fourth `log` statement has a simple filter attached

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

→Destination

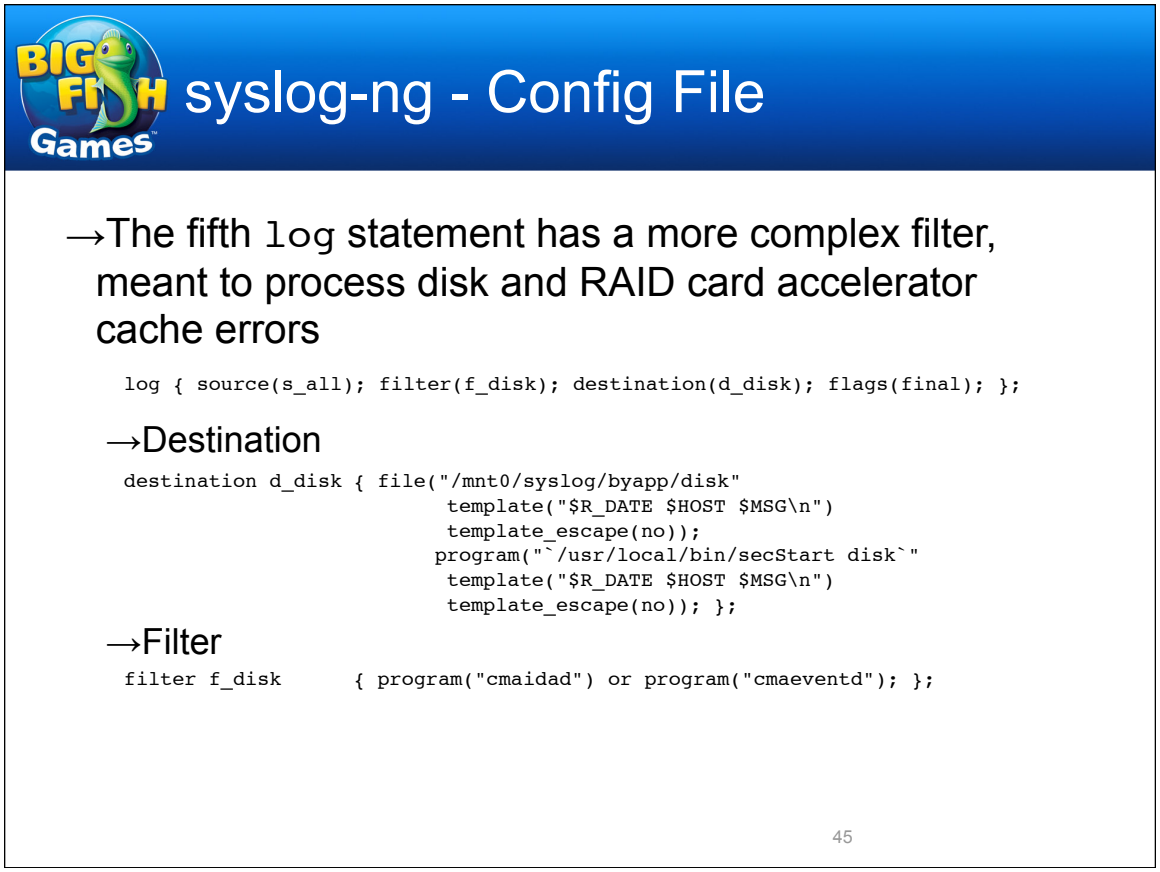
```
destination d_emerg     { file("/mnt0/syslog/byapp/emerg"
                  template("$R_DATE $HOST $MSG\n")
                  template_escape(no));
                 program("`/usr/local/bin/secStart emerg`"
                  template("$R_DATE $HOST $MSG\n")
                  template_escape(no)); };
```

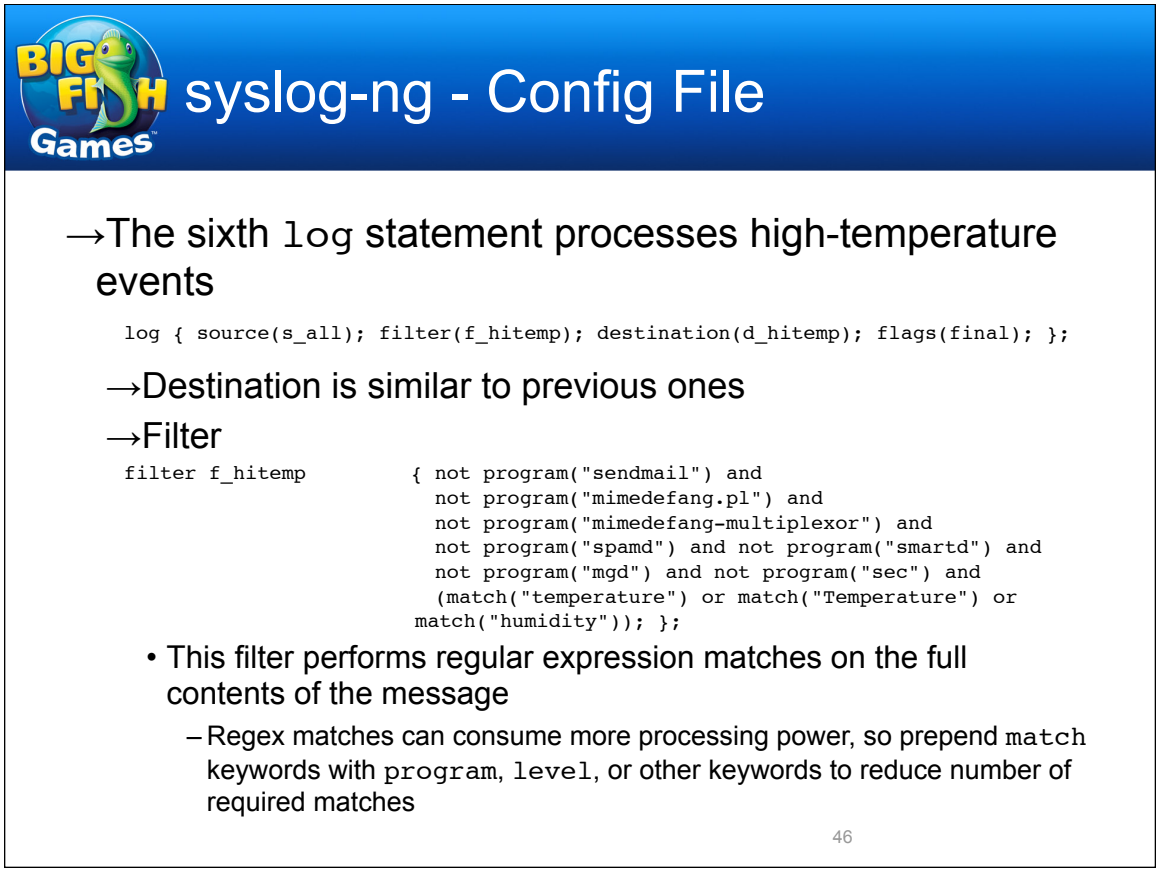
- Messages of `emerg` severity have a dedicated SEC process to generate immediate email notifications

→Filter

```
filter f_emerg          { level(emerg); };
```

- You can use keywords (such as `level`, `program`, or `host`) and Boolean logic to construct filters


syslog-ng - Config File
→The fifth log statement has a more complex filter, meant to process disk and RAID card accelerator cache errors
log { source(s_all); filter(f_disk); destination(d_disk); flags(final); };
→Destination
destination d_disk { file("/mnt0/syslog/byapp/disk"
template("$R_DATE $HOST $MSG\n")
template_escape(no));
program("`/usr/local/bin/secStart disk`"
template("$R_DATE $HOST $MSG\n")
template_escape(no)); };
→Filter
filter f_disk { program("cmaidad") or program("cmaeventd"); };
45


syslog-ng - Config File
→The sixth log statement processes high-temperature events
log { source(s_all); filter(f_hitemp); destination(d_hitemp); flags(final); };
→Destination is similar to previous ones
→Filter
filter f_hitemp { not program("sendmail") and
not program("mimedefang.pl") and
not program("mimedefang-multiplexor") and
not program("spamd") and not program("smartd") and
not program("mgd") and not program("sec") and
(match("temperature") or match("Temperature") or
match("humidity")); };
• This filter performs regular expression matches on the full contents of the message
– Regex matches can consume more processing power, so prepend match keywords with program, level, or other keywords to reduce number of required matches
46

syslog-ng - Config File

→Here's a statement I used to gather possible memory errors from Solaris clients

```
log { source(s_all); filter(f_mem); destination(d_mem); flags(final); };
```

→Destination is similar to previous ones

→Filter

```
filter f_hitemp         { program("SUNW,UltraSPARC") or
                          match("\\[AFT") or
                          (match("AFAR") and not match("SAFARI")) or
                          match("Fault_PC") or
                        match("Memory Module") or
                        match("Softerror") or
                        (program("unix") and match("remov") and match("age")); };
```


syslog-ng - Config File

→The final `log` statement funnels logged `su` and `sudo` commands into a separate file

```
log { source(s_all); filter(f_su); destination(d_su); flags(final); };
```

→Destination

```
destination d_su   { file("/mnt0/syslog/byapp/su"
                     template("$R_DATE $HOST $MSG\n")
                     template_escape(no)); };
```

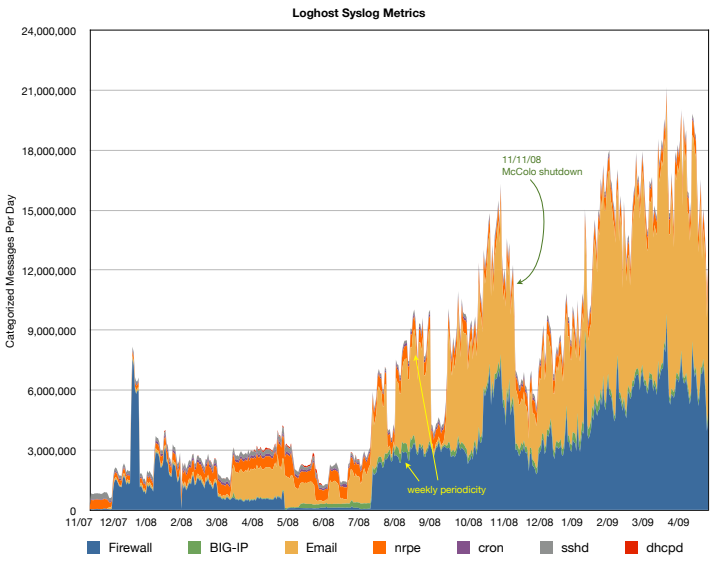
→Filter

```
filter f_su             { (program("su") and not program("sudo")) or
                          (program("sudo") and
                            (match("sh$") or match("COMMAND=/.*/su"))); };
```

syslog-ng - Performance

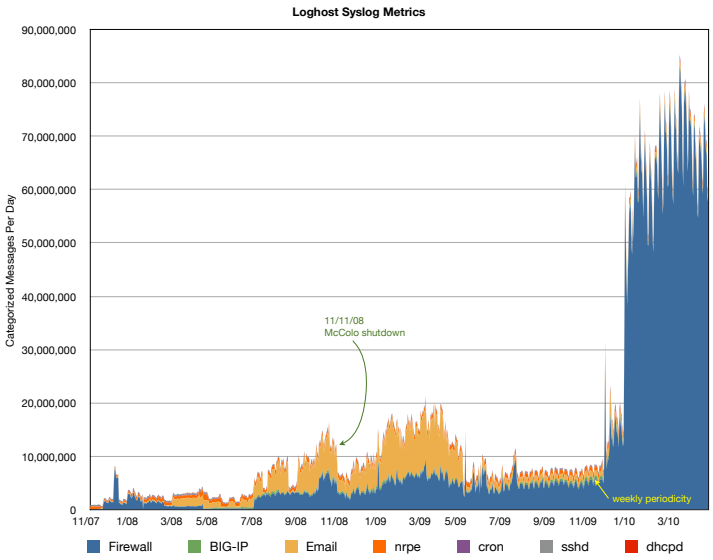
- →At its peak (early 2010), this installation of syslog-ng handled 60-80 million log messages per day (over 800 msgs/sec on average)
  - →Up from 10-20 million in 4/09, 4-5 million in 11/08, and 1.3 million in 6/08
  - →About 65 million from firewalls
  - →Nearly all the remaining logs are related to email, BIG-IPs, SSH, cron, and Nagios agents (`nrpe`)
- →At that rate, syslog-ng takes about 25 MB of RAM, using about 5% of CPU

syslog-ng - Performance



Messages Per Day

syslog-ng - Performance

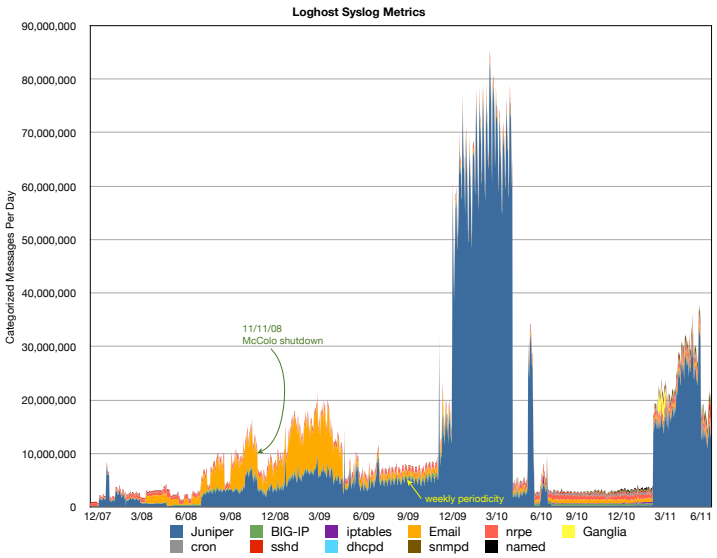


Messages Per Day

syslog-ng - Performance

Drinking from a firehose

syslog-ng - Performance



Messages Per Day

Ye Olden Days

The Olden Days - Swatch

→*Join me in those thrilling days of yesteryear...*

→Back in the 20th century, I set up a centralized loghost on a Pentium system running Caldera Linux, taking logs from AIX, HP-UX, Solaris, DYNIX/ptx, Linux and other UNIXy hosts, and Cisco border routers, Ascend and other network gear

→None of that is relevant now

→Tool used: Swatch

→Nice thing about Swatch: It's simple

The Olden Days - Swatch

```
ignore = /(above|last) message repeat(s|ed) \d+ time/
ignore = /-- MARK --/
ignore = /sendmail\[\d+\]: .+ stat=Sent/
ignore = /sendmail\[\d+\]: .+ relay=/
ignore = /printd\[\d+\]:/
ignore = /lp(stat)?\[\d+\]:/
ignore = /sshd\[\d+\]: log: Password authentication for \w+ accepted\./
watchfor = /\%SEC\-6\-IPACCESSLOGD?P:/
        append = /log/swatch/cisco_acl
        throttle = 15:00
watchfor = /\%\w+\-\d\-[\w_]+:/
        append = /log/swatch/cisco
        throttle = 15:00
watchfor = /sendmail\[\d+\]:/
        append = /log/swatch/mail
        throttle = 15:00
watchfor = /named(\-xfer)?\[\d+\]:/
        append = /log/swatch/named
        throttle = 15:00
watchfor = /.*/
        append = /log/swatch/misc
        throttle = 15:00
```

Excerpts from `swatchrc`

The Olden Days - Swatch

→Bad thing about Swatch: It's simple

→One line, one rule

→Can't deal with multi-line events

- Sendmail, boot, shutdown, scans, etc.

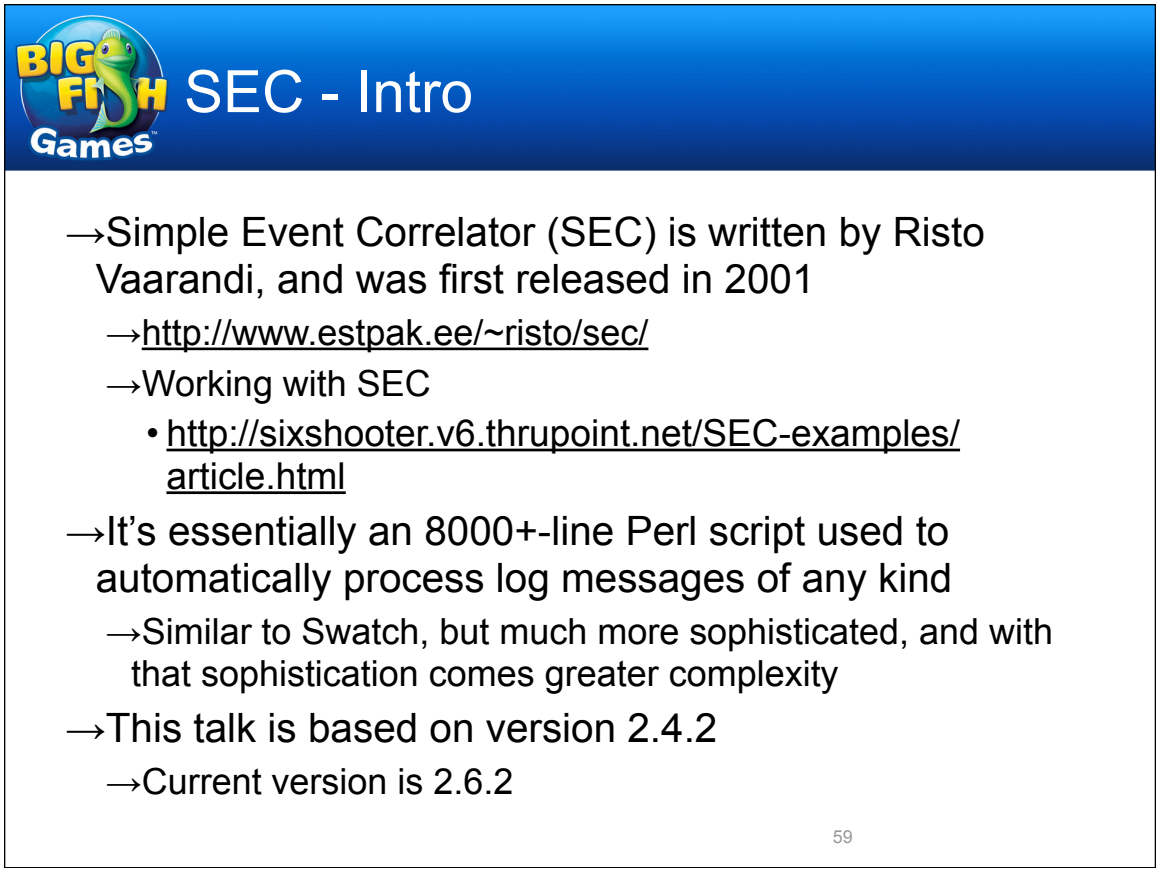
→My immediate motivation for setting up an SEC-based system c. 2005 was network scans

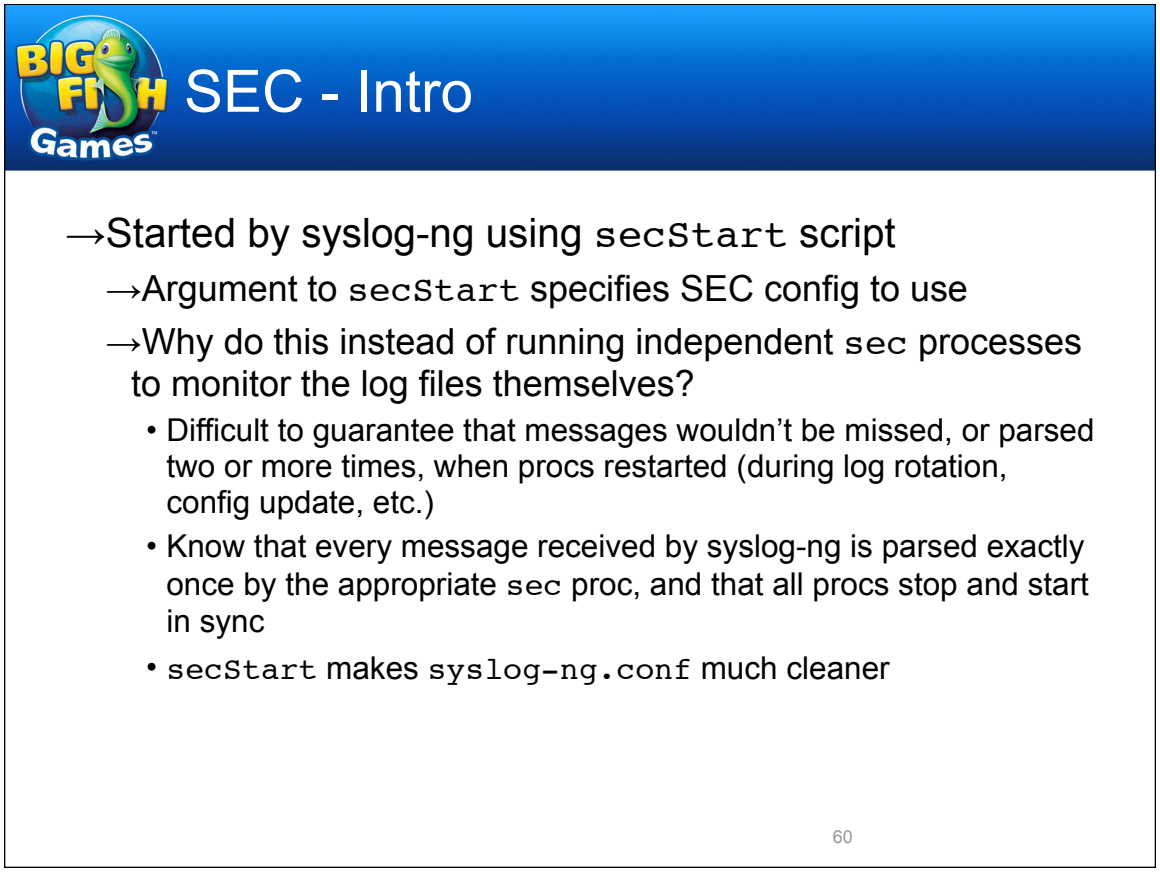
→Shop I was at had a shell script on each host, run by `cron`, that would track its place in the `messages` log, `grep` out some stuff, and email the rest

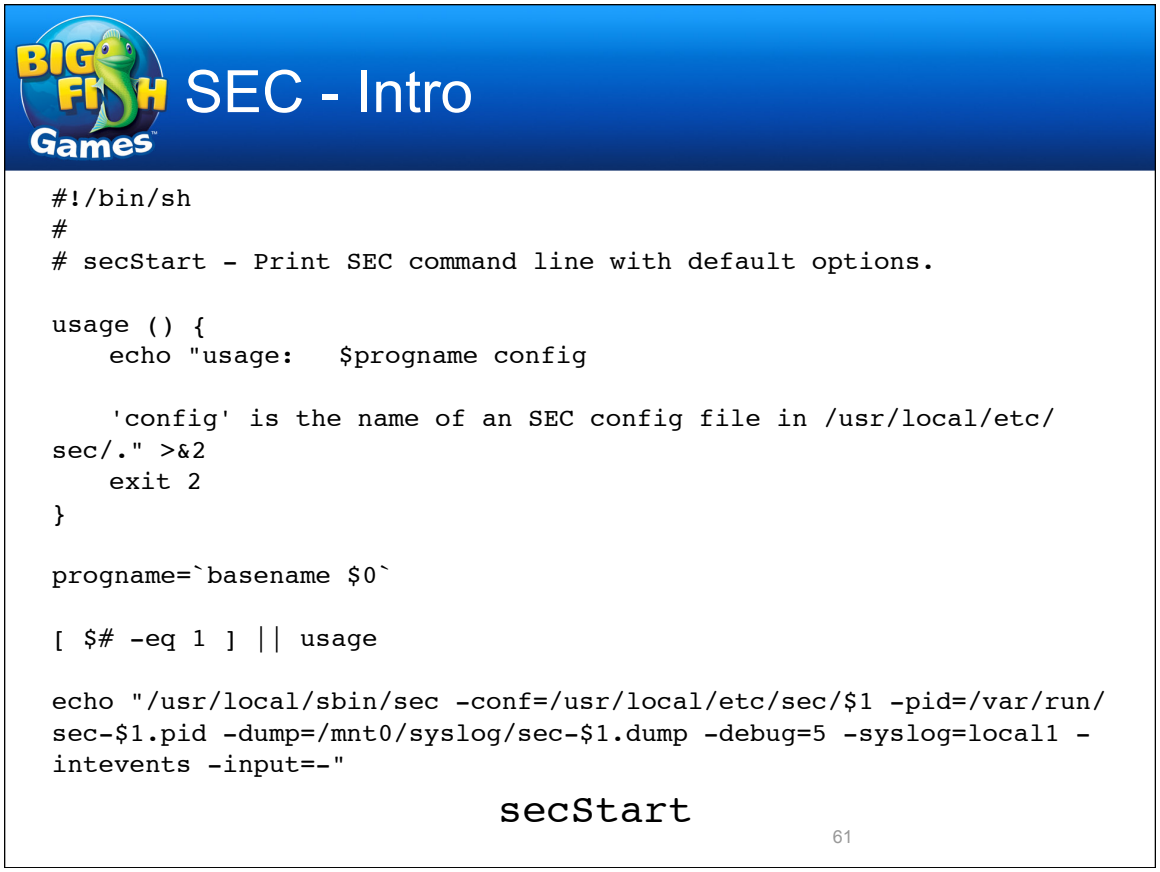
→Among other failings, during regular network scans we were bombarded with dozens of email messages

→Swatch wasn't going to cut it

Simple Event Correlator


SEC - Intro
→Simple Event Correlator (SEC) is written by Risto Vaarandi, and was first released in 2001
→http://www.estpak.ee/~risto/sec/
→Working with SEC
• http://sixshooter.v6.thrupoint.net/SEC-examples/article.html
→It's essentially an 8000+-line Perl script used to automatically process log messages of any kind
→Similar to Swatch, but much more sophisticated, and with that sophistication comes greater complexity
→This talk is based on version 2.4.2
→Current version is 2.6.2
59

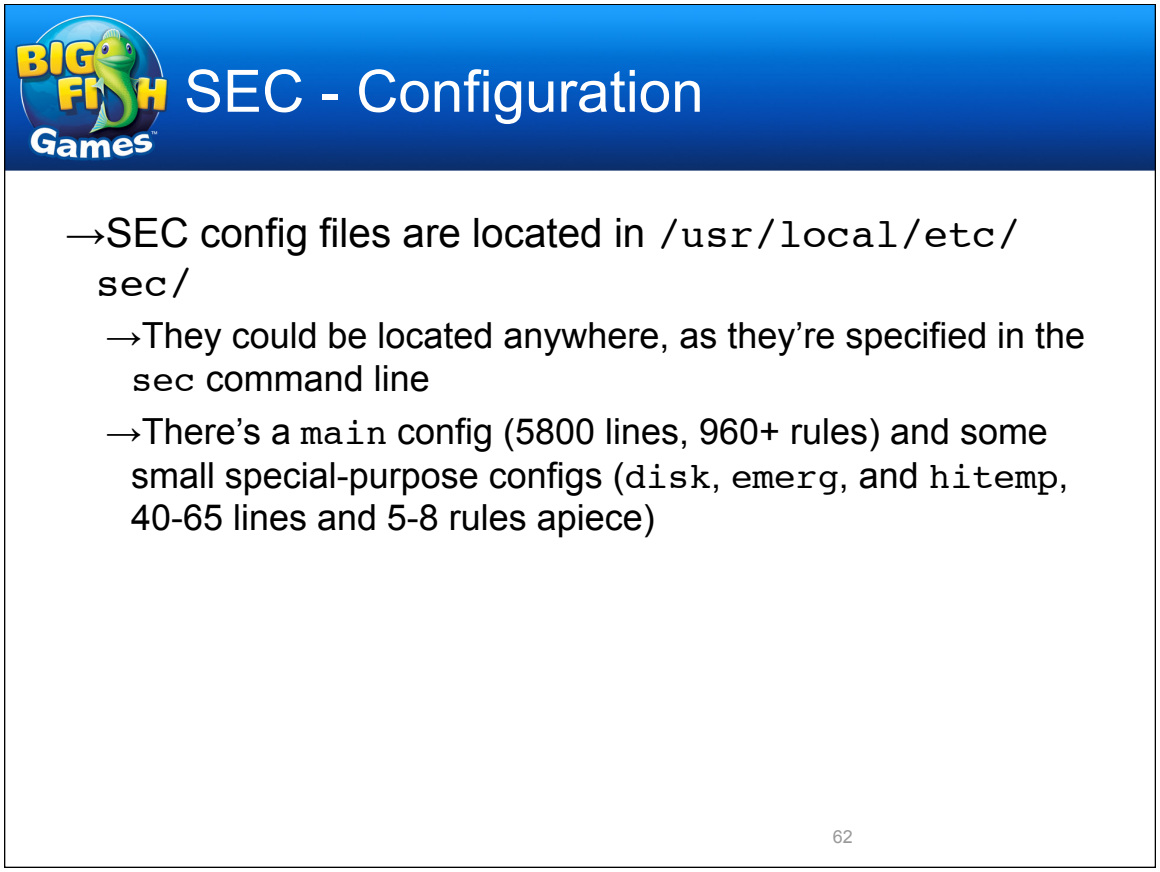

SEC - Intro
→Started by syslog-ng using secStart script
→Argument to secStart specifies SEC config to use
→Why do this instead of running independent sec processes to monitor the log files themselves?
• Difficult to guarantee that messages wouldn't be missed, or parsed two or more times, when procs restarted (during log rotation, config update, etc.)
• Know that every message received by syslog-ng is parsed exactly once by the appropriate sec proc, and that all procs stop and start in sync
• secStart makes syslog-ng.conf much cleaner
60



SEC - Intro

```
#!/bin/sh
#
# secStart - Print SEC command line with default options.

usage () {
    echo "usage:   $progname config

    'config' is the name of an SEC config file in /usr/local/etc/
sec/." >&2
    exit 2
}

progname=`basename $0`

[ $# -eq 1 ] || usage

echo "/usr/local/sbin/sec -conf=/usr/local/etc/sec/$1 -pid=/var/run/
sec-$1.pid -dump=/mnt0/syslog/sec-$1.dump -debug=5 -syslog=local1 -
intevents -input=-"
```

`secStart`



SEC - Configuration

→SEC config files are located in `/usr/local/etc/sec/`

→They could be located anywhere, as they're specified in the `sec` command line

→There's a `main` config (5800 lines, 960+ rules) and some small special-purpose configs (`disk`, `emerg`, and `hitemp`, 40-65 lines and 5-8 rules apiece)

SEC - Configuration

→An SEC configuration is composed of multi-line stanzas, or rule definitions, with each line containing a key and value

→Keys include:

- →`type` - Type of rule (examples later)
- →`desc` - Textual description of rule
- →`ptype` - Type of pattern (typically `regexp`)
- →`pattern` - String or Perl-style regular expression used to match log message
- →`context` - Apply rule only when named context in effect
- →`action` - What to do when rule is matched
- →`continue` - After this rule, continue or stop (default)

SEC - Configuration

→Rule types used in the examples

- →`suppress` - Simple rule to toss messages that match
- →`single` - If message matches, take immediate action
- →`singlewithsuppress` - If message matches, take immediate action, but then ignore similar messages for a time given by value of `window`
- →`singlewiththreshold` - Take action if the number of matching messages within a given `window` reaches a `threshold`
- →`pairwithwindow` - Specify 2 patterns; when 1st pattern matches, watch for 2nd pattern to appear within `window`; if it does, execute action; if not, execute different action

SEC - Configuration

→For each message, rules are processed one at a time, in order, until the message matches a rule without `continue=takenext`, or end-of-file is reached

→We'll start with a simple configuration, one that was used to rewrite simplified messages for outbound firewall connections (`/usr/local/etc/sec/outbound`)

SEC - Configuration Example: `outbound`

```
type=single
desc=Set log file and addressee list
ptype=substr
pattern=SEC_STARTUP
context=SEC_INTERNAL_EVENT
action=assign %f /mnt0/syslog/firewall/outbound
```

→A rule like this appears at the top of each config file

→It matches internally-generated messages used by SEC to mark startup, and sets variables for later use

- `%f` - file in which to record parsed messages

SEC - Configuration Example: outbound

→The internal startup messages look like this:

```
Apr 23 12:13:13 loghost.example.com sec[16832]: SEC (Simple Event Correlator) 2.4.2
Apr 23 12:13:13 loghost.example.com sec[16832]: Reading configuration from /usr/
local/etc/sec/outbound
Apr 23 12:13:13 loghost.example.com sec[16832]: Creating SEC internal context
'SEC_INTERNAL_EVENT'
Apr 23 12:13:13 loghost.example.com sec[16832]: Creating SEC internal event
'SEC_STARTUP'
Apr 23 12:13:13 loghost.example.com sec[16832]: Deleting SEC internal context
'SEC_INTERNAL_EVENT'
```

→More on contexts in a bit

SEC - Configuration Example: outbound

```
type=singlewithsuppress
desc=$2 $3 $4
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) .+NetScreen device_id=([\w.-]+)\s+\[Root\]system-
notification-00257\(traffic\):.+(policy_id=\d+).+(src=\d+\.\d+\.\d+\.\d+ dst=\d+\.\d
+\.\d+\.\d+) src_port=\d+ (dst_port=\d+)
action=write %f $1 $2 $3 $4 $5
window=600
```

→This rule looks for NetScreen firewall traffic logs

→Elements of the message (timestamp, hostname, policy ID, source and destination data) are captured in Perl regexp backreferences (`$1`, `$2`, etc.)

→A new log message is then written out to a log file

SEC - Configuration Example: `outbound`

```
type=singlewithsuppress
desc=$2 $3 $4
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) .+NetScreen device_id=([\w.-]+)\s+\[Root\]system-
notification-00257\(traffic\):.+(policy_id=\d+).+(src=\d+\.\d+\.\d+\.\d+ dst=\d+\.\d
+\.\d+\.\d+) src_port=\d+ (dst_port=\d+)
action=write %f $1 $2 $3 $4 $5
window=600
```

→For `singlewithsuppress` and other rules, the *event description* (the value of `desc`) is critical

→Subsequent messages are suppressed within the specified `window` only if their event descriptions are identical (so don't include timestamp, for example)

SEC - Configuration Example: `disk`

→Now we'll take a look at a configuration that makes use of a context (`/usr/local/etc/sec/disk`)

→A context is a named state that can be set by a rule, which affects the processing of other rules until the context lifetime runs out, it's deleted by another rule, or the SEC process dies

→A context can also store a set of related messages

→First, set up variables for notification email addresses

```
type=single
desc=Set addressee lists
ptype=substr
pattern=SEC_STARTUP
context=SEC_INTERNAL_EVENT
action=assign %a sec-notify;\
     assign %rt rt@example.com
```

SEC - Configuration Example: `disk`

```
type=suppress
desc=No new reports w/in timeout
ptype=regexp
pattern=\w+\s+\d+\s+\d+:\d+:\d+ ([\w.-]+) cmaidad\[\d+\]: Physical Drive
context=DISK_$1

type=single
desc=$1 $2 $3
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) cmaidad\[\d+\]: Physical Drive Status
Change: (Slot \d+ Port \w+ Box \d+ Bay \d+\. Status is now (Failed|Predictive
Failure)\.)
action=create DISK_$2 5; create OUT; add OUT %s; add OUT .; add OUT .;\
     add OUT You can check log1:/mnt0/syslog/byapp/disk for further status.;\
     report OUT /bin/mail -s "SEC: Disk failure on $2" %a;\
     report OUT /bin/mail -s "log issue: Disk failure on $2" %rt
```

→The first rule has no effect until a context is set, so we have to look at the second rule to make sense of this

→The second rule catches log messages that indicate a physical drive failure

SEC - Configuration Example: `disk`

```
type=suppress
desc=No new reports w/in timeout
ptype=regexp
pattern=\w+\s+\d+\s+\d+:\d+:\d+ ([\w.-]+) cmaidad\[\d+\]: Physical Drive
context=DISK_$1

type=single
desc=$1 $2 $3
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) cmaidad\[\d+\]: Physical Drive Status
Change: (Slot \d+ Port \w+ Box \d+ Bay \d+\. Status is now (Failed|Predictive
Failure)\.)
action=create DISK_$2 5; create OUT; add OUT %s; add OUT .; add OUT .;\
     add OUT You can check log1:/mnt0/syslog/byapp/disk for further status.;\
     report OUT /bin/mail -s "SEC: Disk failure on $2" %a;\
     report OUT /bin/mail -s "log issue: Disk failure on $2" %rt
```

→When the second rule matches, it creates a context named `DISK_`*`hostname`* which lasts for 5 seconds

→While this context is in effect, further messages are suppressed by the first rule to prevent something like a RAID disconnect from sending multiple alerts

SEC - Configuration Example: `disk`

```
type=suppress
desc=No new reports w/in timeout
ptype=regexp
pattern=\w+\s+\d+\s+\d+:\d+:\d+ ([\w.-]+) cmaidad\[\d+\]: Physical Drive
context=DISK_$1

type=single
desc=$1 $2 $3
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) cmaidad\[\d+\]: Physical Drive Status
Change: (Slot \d+ Port \w+ Box \d+ Bay \d+\. Status is now (Failed|Predictive
Failure)\.)
action=create DISK_$2 5; create OUT; add OUT %s; add OUT .; add OUT .;\
     add OUT You can check log1:/mnt0/syslog/byapp/disk for further status.;\
     report OUT /bin/mail -s "SEC: Disk failure on $2" %a;\
     report OUT /bin/mail -s "log issue: Disk failure on $2" %rt
```

→In addition, the rule creates a context named `OUT`

→The matched message is added to the *event store*, along with a comment to guide further investigation

→The report command then emails the contents of the event store, and creates an RT ticket (via email)

SEC - Configuration Example: `main`

→Now onto the `main` configuration

→This is a large file, so I'll choose a few excerpts

→First, some words about the overall structure

→Most of the work of reducing and correlating logs is done in the vast middle of the file

→At the end, any messages that ran the gauntlet are tagged (`PARSED:` is prepended to the message), and sent back through the rule set with an `event` action

→This is done so that duplicate messages can be suppressed; near the beginning of the file are rules that eliminates duplicates of `PARSED:` messages that show up within 15 minutes of each other

SEC - Configuration Example: `main`

→Structure (cont'd.)

→The same rules that suppress duplicates retag the remaining messages (prepending `UNDUPED:` to the message)

- The tag is necessary so that log messages don't match the following rules their first time through

→`UNDUPED:` messages are then counted in sliding time windows of 10 minutes; if the number reaches a threshold (currently 15), an email is sent immediately, as logging volume may indicate a problem

SEC - Configuration Example: `main`

→Structure (cont'd.)

→Finally, `UNDUPED:` messages are written out to log files (without the `UNDUPED:` tag)

- Most Sendmail messages are written to dedicated log files; their volume is so high, and their actionability so low, that they're written to separate files and not counted as described earlier
- All other messages go to `unix.tmp` or `drupal.tmp` (for periodic email reports) and to files in `sec/`
- Log messages written to these files are in standard syslog format, in case further processing is desired

→We'll see what these rules look like in a bit

SEC - Configuration Example: `main`

```
-rw-r--r--  1 syslog syslog   2055 Feb 27 09:04 attack
-rw-r--r--  1 syslog syslog   2622 Feb 27 06:37 drupal
-rw-r--r--  1 syslog syslog 663351 Feb 27 15:29 mail_custserv
-rw-r--r--  1 syslog syslog  14411 Feb 27 15:24 mail_inbound
-rw-r--r--  1 syslog syslog 117658 Feb 27 15:29 mail_outbound
-rw-r--r--  1 root   syslog 446453 Feb 27 15:32 mysql_err
-rw-r--r--  1 syslog syslog   3378 Feb 24 14:34 pdu
-rw-r--r--  1 syslog syslog 185794 Feb 27 14:23 unix
```

Contents of `/mnt0/syslog/sec/`

SEC - Configuration Example: `main`

→Here's how the file breaks down

- →**Setup rule** (1 rule, 29 lines)
  - Set variables for log pathnames, notification email
- →**Temporary rules** (23 rules, 130 lines)
  - Suppress logs for issues being worked on
- →**Deduplication rules** (6 rules, 55 lines)
  - Suppress duplicate messages
- →**Post-dedupe rules** (22 rules, 167 lines)
  - Work on deduplicated logs, mainly to correlate network outages
- →**Real-time alert rules** (10 rules, 75 lines)
  - Send alerts for specific messages

SEC - Configuration Example: `main`

→Breakdown (cont'd.)

- →**Output rules** (13 rules, 94 lines)
  - Count parsed messages and write to log files
- →**Scan rules** (27 rules, 177 lines)
  - Correlate messages from network security scans
- →**Network device rules** (22 rules, 124 lines)
  - Handle network device logs
- →**Syslog heartbeat rule** (1 rule, 14 lines)
  - Watch for devices that have stopped logging
- →**Misc. suppression rules** (87 rules, 436 lines)
  - Lots of basic `suppress` rules

SEC - Configuration Example: `main`

→Breakdown (cont'd.)

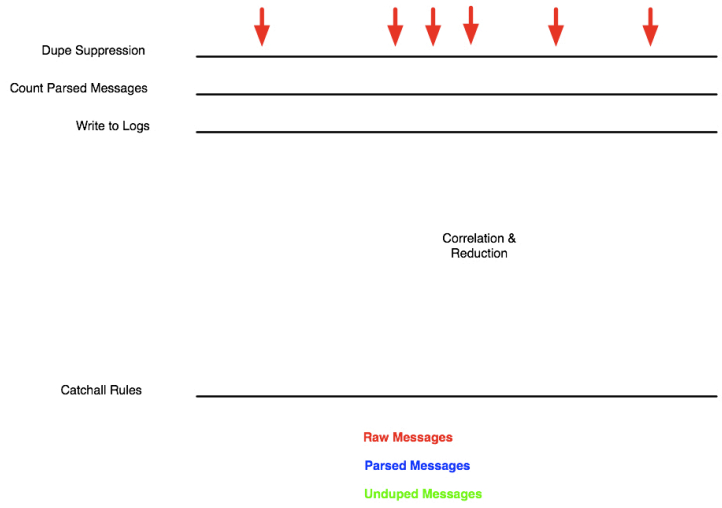
- →**Time-specific rules** (30 rules, 153 lines)
  - Suppress daily events
- →**Overload rules** (13 rules, 92 lines)
  - Correlate logs associated with overloaded hosts
- →**Service rules** (613 rules, 3588 lines)
  - Suppress or correlate messages for `iptables`, DHCP, BIND, Sendmail, SSH, Drupal, LDAP, NTP, MySQL, kernel, VMware, etc.
- →**Boot/Shutdown/Upgrade rules** (65 rules, 424 lines)
  - Correlate messages from system boots, shutdowns, and OS upgrades

SEC - Configuration Example: `main`

→Breakdown (cont'd.)

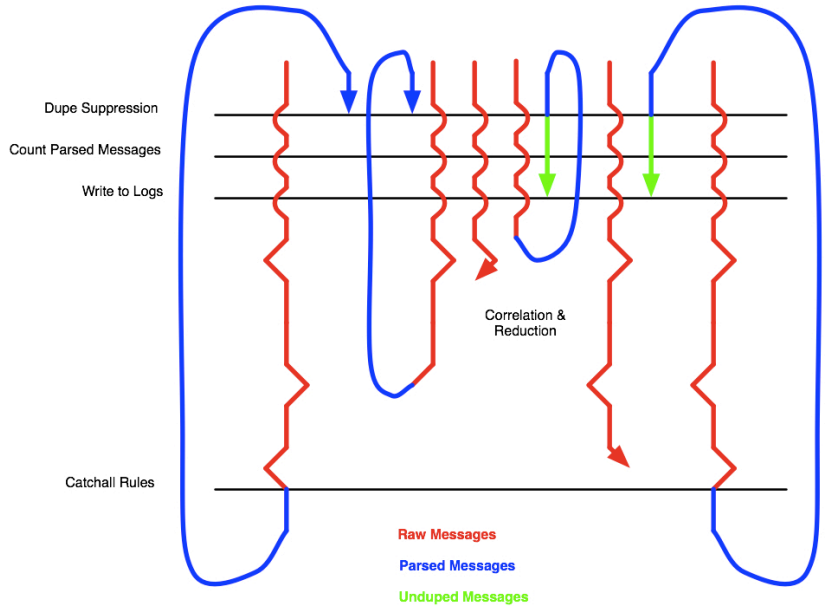
- →**Power & environmental systems rules** (14 rules, 99 lines)
  - Handle logs from PDUs, UPSes, EMUs, etc.
- →**Catchall rules** (14 rules, 87 lines)
  - Tag remaining messages and send them back through
- →**TOTAL:** 961 rules, 5744 lines

SEC - Configuration Example: `main`



Flow of Log Messages Through `main` Configuration

SEC - Configuration Example: `main`



Flow of Log Messages Through `main` Configuration

SEC - Configuration Example: `main`

→Here's an explanatory video that also demonstrates what happens to the log messages

→http://www.occam.com/sa/logging_video.html

SEC - Configuration Example: `main`

→For a detailed look, we'll start at the end of the file

→Here are some of the catchall rules to tag parsed messages

```
type=single
desc=Log messages w/o (usually) unhelpful PID and subprogram
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+ [\w.-]+ .+?)\([\w.-]+\)\[\d+\]: (.+)
action=event 0 PARSED:$1: $2

type=single
desc=Log messages w/o (usually) unhelpful PID
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+ [\w.-]+ .+?)\[\d+\]: (.+)
action=event 0 PARSED:$1: $2

type=single
desc=Log all remaining messages
ptype=regexp
pattern=.+
action=event 0 PARSED:$0
```


SEC - Configuration Example: `main`

→The catchall rules remove some elements that aren't usually helpful in evaluating importance (syslog message ID, process ID) with the use of Perl backrefs

→Then `event 0` puts the parsed and tagged message back into the message queue without delay

→Back toward the beginning of the file, the `PARSED:` messages are matched by the duplicate suppression rules

SEC - Configuration Example: `main`

```
type=singlewithsuppress
desc=$2
ptype=regexp
pattern=^PARSED:(\w+\s+\d+\s+\d+:\d+:\d+) (.+)
action=event 0 UNDUPED:$1 $2
window=900

# In case something somehow gets to here...
type=singlewithsuppress
desc=Malformed message $1
ptype=regexp
pattern=^PARSED:(.+)
action=event 0 UNDUPED:%s
window=900
```

→The `singlewithsuppress` rule uses the value of `desc` to determine whether messages are “similar”

→Since the timestamp isn’t included in the event description, messages are compared only on content

→These rules prevent flooding by lots of similar messages

SEC - Configuration Example: `main`

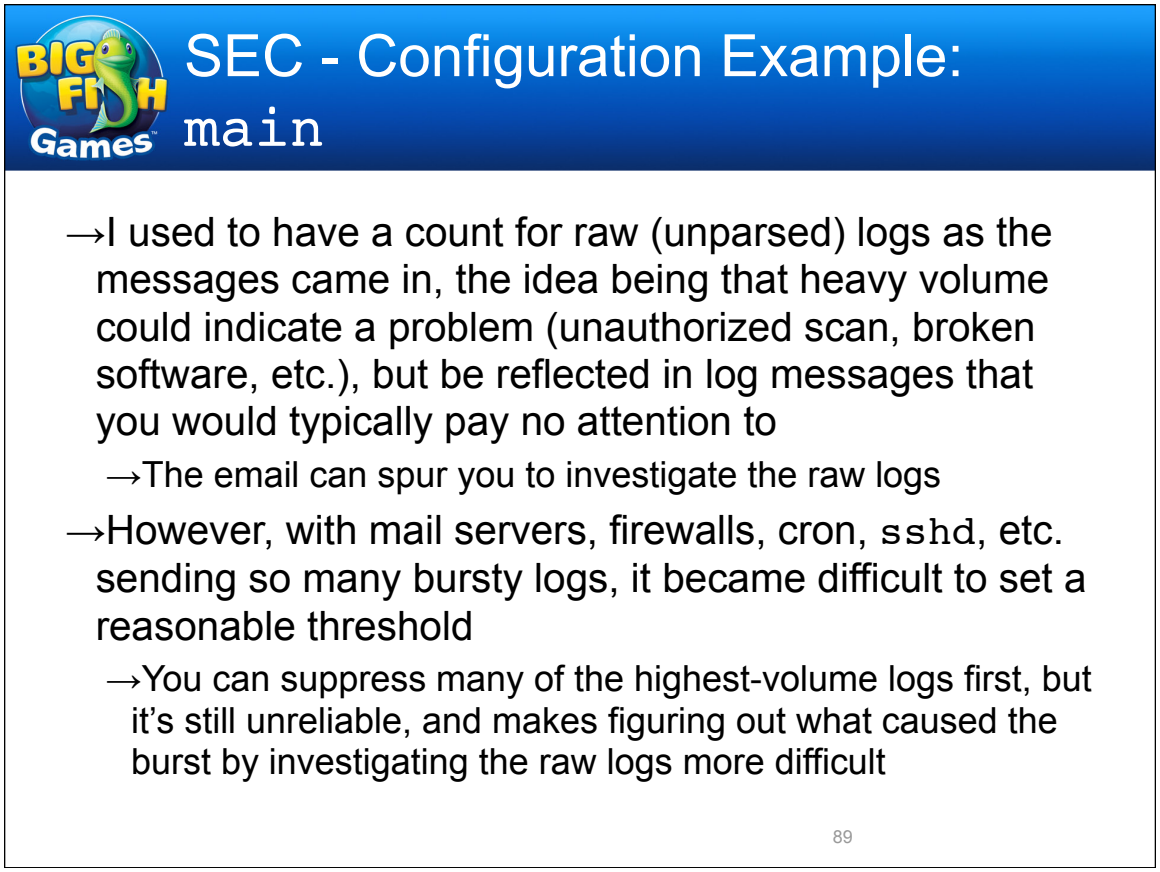
→After de-duplication, remaining messages are counted:

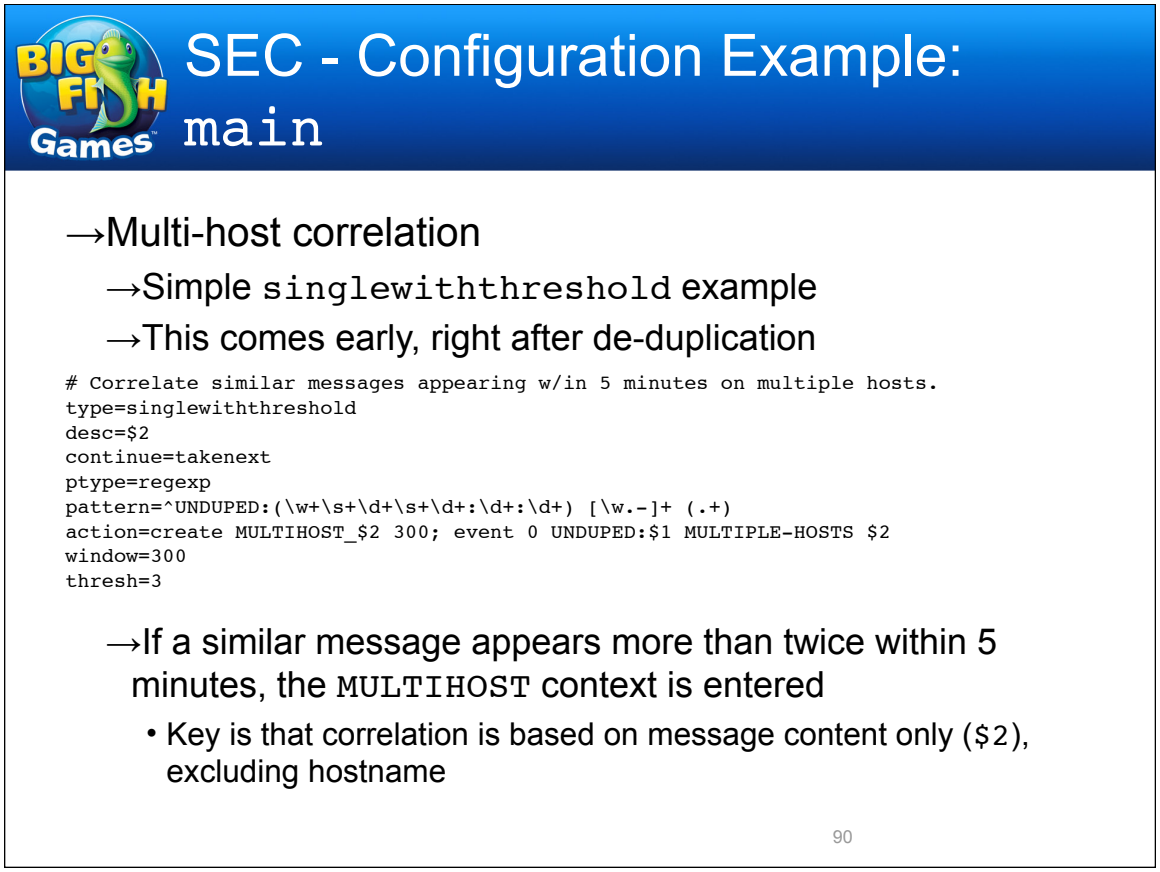
```
# Count parsed messages.
type=singlewiththreshold
desc=Over 15 interesting log messages received in the last 10 minutes.
continue=takenext
ptype=regexp
pattern=^UNDUPED:
action=create WARNED_OF_EXCESSIVE_INTERESTING_LOGS 1800;\
        pipe '' /usr/bin/mail -s "SEC: Excessive logging detected at %t" %a
context=!WARNED_OF_EXCESSIVE_INTERESTING_LOGS
window=600
thresh=15
```

→If at least 15 messages are counted in any 10-minute (600-second) period, an email is sent

→A context is used to prevent such emails from being sent more than twice an hour

→When the threshold is tripped, the context prevents this rule from operating for 30 minutes (1800 seconds)


SEC - Configuration Example: main
→I used to have a count for raw (unparsed) logs as the messages came in, the idea being that heavy volume could indicate a problem (unauthorized scan, broken software, etc.), but be reflected in log messages that you would typically pay no attention to
→The email can spur you to investigate the raw logs
→However, with mail servers, firewalls, cron, sshd, etc. sending so many bursty logs, it became difficult to set a reasonable threshold
→You can suppress many of the highest-volume logs first, but it's still unreliable, and makes figuring out what caused the burst by investigating the raw logs more difficult
89


SEC - Configuration Example: main
→Multi-host correlation
→Simple singlewiththreshold example
→This comes early, right after de-duplication
# Correlate similar messages appearing w/in 5 minutes on multiple hosts.
type=singlewiththreshold
desc=$2
continue=takenext
ptype=regexp
pattern=^UNDUPED:(\w+\s+\d+\s+\d+:\d+:\d+) [\w.-]+ (.+)
action=create MULTIHOST_$2 300; event 0 UNDUPED:$1 MULTIPLE-HOSTS $2
window=300
thresh=3
→If a similar message appears more than twice within 5 minutes, the MULTIHOST context is entered
• Key is that correlation is based on message content only ($2), excluding hostname
90

SEC - Configuration Example: `main`

→Multi-host correlation

→This rule actually comes earlier, right before de-dupe

```
# Suppress additional messages in multi-host events. See creation of correlation
# a few rules below. Need to put this here to suppress PARSED messages, because
# if we suppress UNDUPED messages, we suppress the multi-host message itself.
type=single
desc=Multi-host event
ptype=regexp
pattern=^PARSED:\w+\s+\d+\s+\d+:\d+:\d+ [\w.-]+ (.+)
action=set MULTIHOST_$1 300
context=MULTIHOST_$1
```

→This acts to suppress additional similar messages (since there's no `continue=takenext`), and extends the context lifetime with `set`

- The context will survive until 5 minutes after the last similar message is seen, providing a sliding window

SEC - Configuration Example: `main`

→Multi-host correlation

→Together, these rules (along with other correlation rules, as in the next example) can turn this

```
Apr 22 13:07:31 host2.example.com syslogd 1.4.1: restart.
Apr 22 13:07:31 host2.example.com syslog: syslogd startup succeeded
Apr 22 13:07:31 host2.example.com syslog: klogd startup succeeded
Apr 22 13:07:33 host2.example.com syslog: syslogd shutdown succeeded
Apr 22 13:08:06 host3.example.com syslogd 1.4.1: restart.
Apr 22 13:08:06 host3.example.com syslog: syslogd startup succeeded
Apr 22 13:08:06 host3.example.com syslog: klogd startup succeeded
Apr 22 13:08:08 host3.example.com syslog: syslogd shutdown succeeded
Apr 22 13:08:40 host4.example.com syslogd 1.4.1: restart.
Apr 22 13:08:40 host4.example.com syslog: syslogd startup succeeded
Apr 22 13:08:40 host4.example.com syslog: klogd startup succeeded
Apr 22 13:08:42 host4.example.com syslog: syslogd shutdown succeeded
Apr 22 13:09:13 host5.example.com syslogd 1.4.1: restart.
Apr 22 13:09:13 host5.example.com syslog: syslogd startup succeeded
Apr 22 13:09:13 host5.example.com syslog: klogd startup succeeded
Apr 22 13:09:16 host5.example.com syslog: syslogd shutdown succeeded
Apr 22 13:09:47 host6.example.com syslogd 1.4.1: restart.
Apr 22 13:09:47 host6.example.com syslog: syslogd startup succeeded
Apr 22 13:09:47 host6.example.com syslog: klogd startup succeeded
Apr 22 13:09:50 host6.example.com syslog: syslogd shutdown succeeded
```

SEC - Configuration Example: `main`

→Multi-host correlation

→Into this

```
Apr 22 13:07:31 host2.example.com syslogd: restarted
Apr 22 13:08:06 host3.example.com syslogd: restarted
Apr 22 13:08:40 host4.example.com syslogd: restarted
Apr 22 13:07:31 MULTIPLE-HOSTS syslogd: restarted
```


SEC - Configuration Example: `main`

→Service restart correlation (using `pairwithwindow`)

```
# Useful for simple services like xinetd, dhcpd, ...
# Disable the correlation if it's part of a reboot.
type=pairwithwindow
desc=Service $3 restart on $2
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) (\w+): \w+ shutdown succeeded
action=event 0 PARSED:$1 $2 $3: shutdown
context=!SHUTDOWN_$2
desc2=Service startup
ptype2=regexp
pattern2=(\w+\s+\d+\s+\d+:\d+:\d+) $2 $3: $3 startup succeeded
action2=event 0 PARSED:$1 %2 %3: restarted
window=10
```

→If we see “shutdown succeeded” followed shortly (within 10 secs) by “startup succeeded” for a service on a host, combine the two messages into a single “restarted” message

SEC - Configuration Example: `main`

→Service restart correlation (using `pairwithwindow`)

```
# Useful for simple services like xinetd, dhcpd, ...
# Disable the correlation if it's part of a reboot.
type=pairwithwindow
desc=Service $3 restart on $2
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) (\w+): \w+ shutdown succeeded
action=event 0 PARSED:$1 $2 $3: shutdown
context=!SHUTDOWN_$2
desc2=Service startup
ptype2=regexp
pattern2=(\w+\s+\d+\s+\d+:\d+:\d+) $2 $3: $3 startup succeeded
action2=event 0 PARSED:$1 %2 %3: restarted
window=10
```

→If a system shutdown context is in effect, the first message that triggers this rule will instead be left alone, so it can be suppressed by a later rule

SEC - Configuration Example: `main`

→Correlation of boot logs

```
##############
# BOOT RULES #
##############

type=single
desc=Create boot context
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) kernel: Linux version
action=create BOOT_$2 240; create SAN_UP_$2 240; create CFENGINE_BOOT_$2 900;\
        create NTP_STOPSTART_$2 900; event 0 UNDUPED:$1 $2 starting up...
context=!BOOT_$2
```

→This first rule sets up the host-specific boot context

→Also sets up contexts for SAN-related, Cfengine, and NTP logs that show up later than the rest

→Logs a message as `UNDUPED:`, rather than `PARSED:`, to bypass multi-host correlation and see every bootup

SEC - Configuration Example: main

→Correlation of boot logs

```
##############
# BOOT RULES #
##############

type=single
desc=Create boot context
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) (gen)?unix:.+(SunOS Release|Copyright
1983-20\d\d Sun Microsystems)
action=create BOOT_$2 600; event 0 PARSED:$1 $2 starting up...
context=!BOOT_$2
```

→Solaris version

SEC - Configuration Example: main

→Correlation of boot logs

→Remaining rules are suppressions, which can turn this

SEC - Configuration Example: main

→Correlation of boot logs

→Into this

```
Apr 24 14:22:50 host2.example.com starting up...
```

→:-)

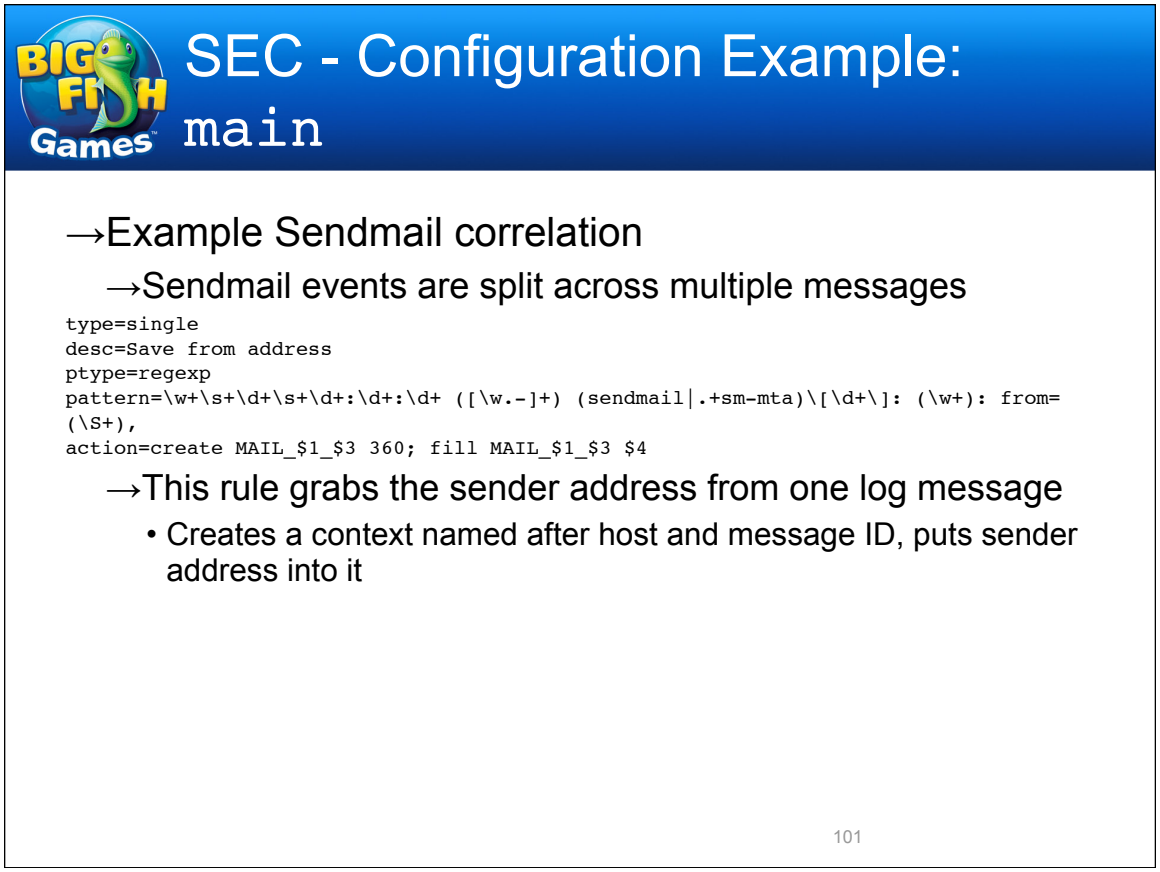
SEC - Configuration Example: main

→Correlation of boot logs

→Here are what anomalous boots can look like

```
Apr 25 10:07:31 host8.example.com starting up...
Apr 25 10:07:35 host8.example.com nscd: 4032 invalid persistent database file "/var/db/nscd/passwd":
verification failed


Apr 26 14:55:28 host1.example.com starting up...
Apr 26 14:55:30 host1.example.com /usr/sbin/gmond: Unable to create UDP client for ganglia.example.com:9450.
Exiting.


Apr 27 17:11:57 host9.example.com starting up...
Apr 27 17:12:01 host9.example.com mysqld: InnoDB: The log sequence number in ibdata files does not match
Apr 27 17:12:01 host9.example.com mysqld: InnoDB: the log sequence number in the ib_logfiles!
Apr 27 17:12:01 host9.example.com mysqld: InnoDB: Database was not shut down normally!
Apr 27 17:12:01 host9.example.com mysqld: Lots of unmatched messages
```

→Errors stand out clearly



SEC - Configuration Example: main

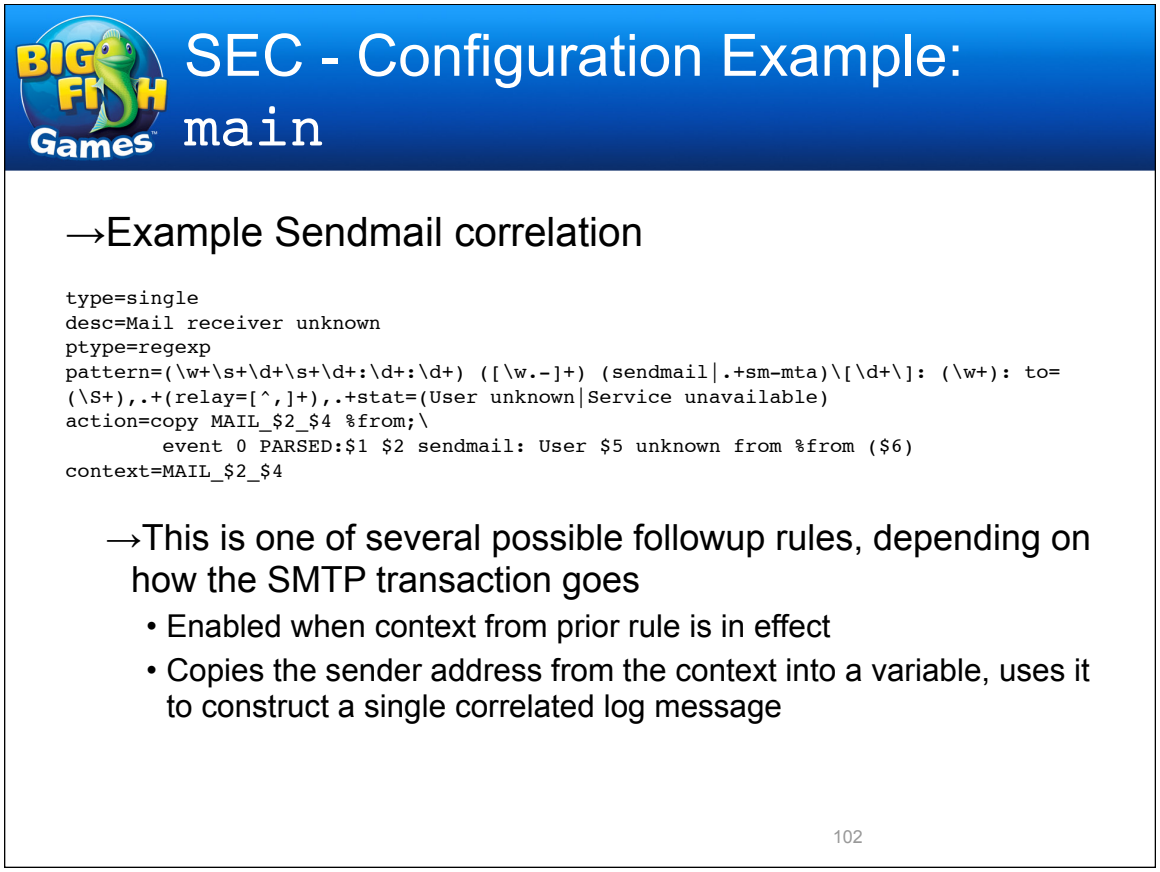
→Example Sendmail correlation

→Sendmail events are split across multiple messages

```
type=single
desc=Save from address
ptype=regexp
pattern=\w+\s+\d+\s+\d+:\d+:\d+ ([\w.-]+) (sendmail|.+sm-mta)\[\d+\]: (\w+): from=
(\S+),
action=create MAIL_$1_$3 360; fill MAIL_$1_$3 $4
```

→This rule grabs the sender address from one log message

- Creates a context named after host and message ID, puts sender address into it



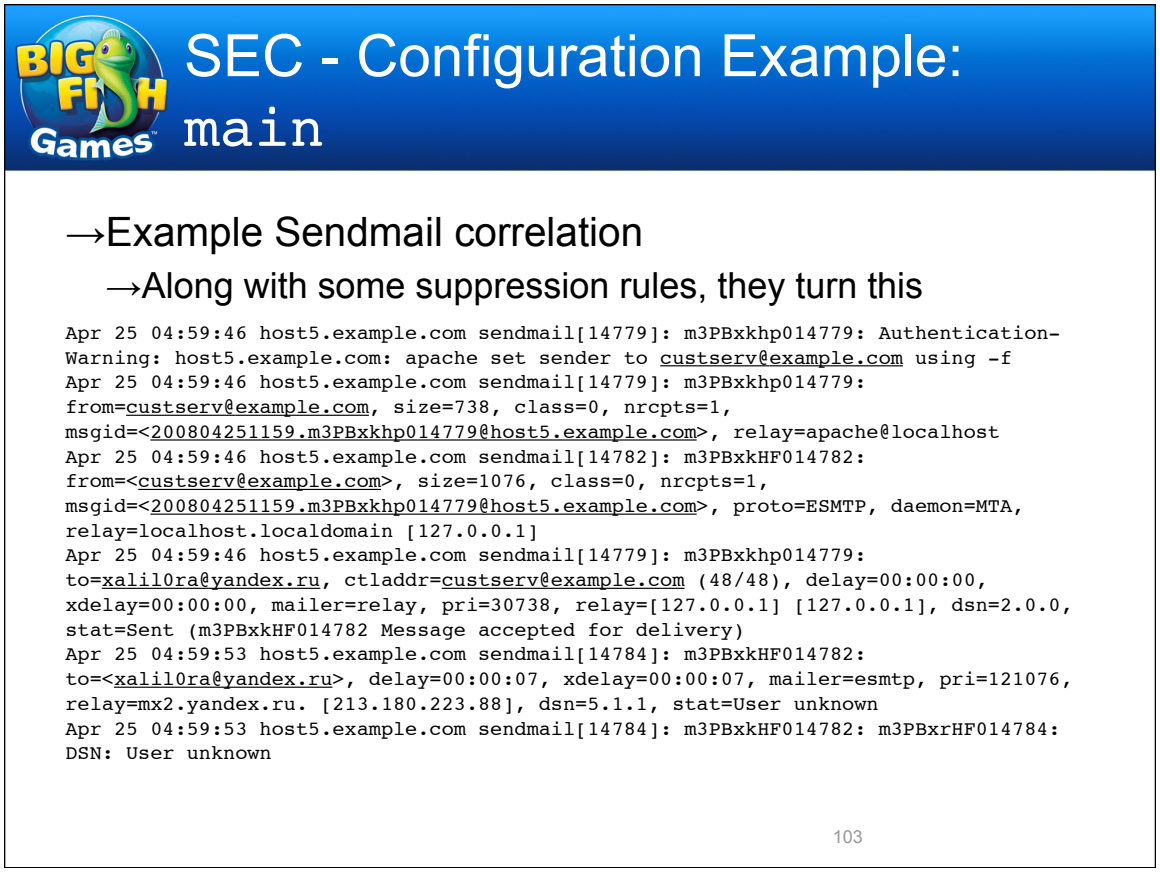
SEC - Configuration Example: main

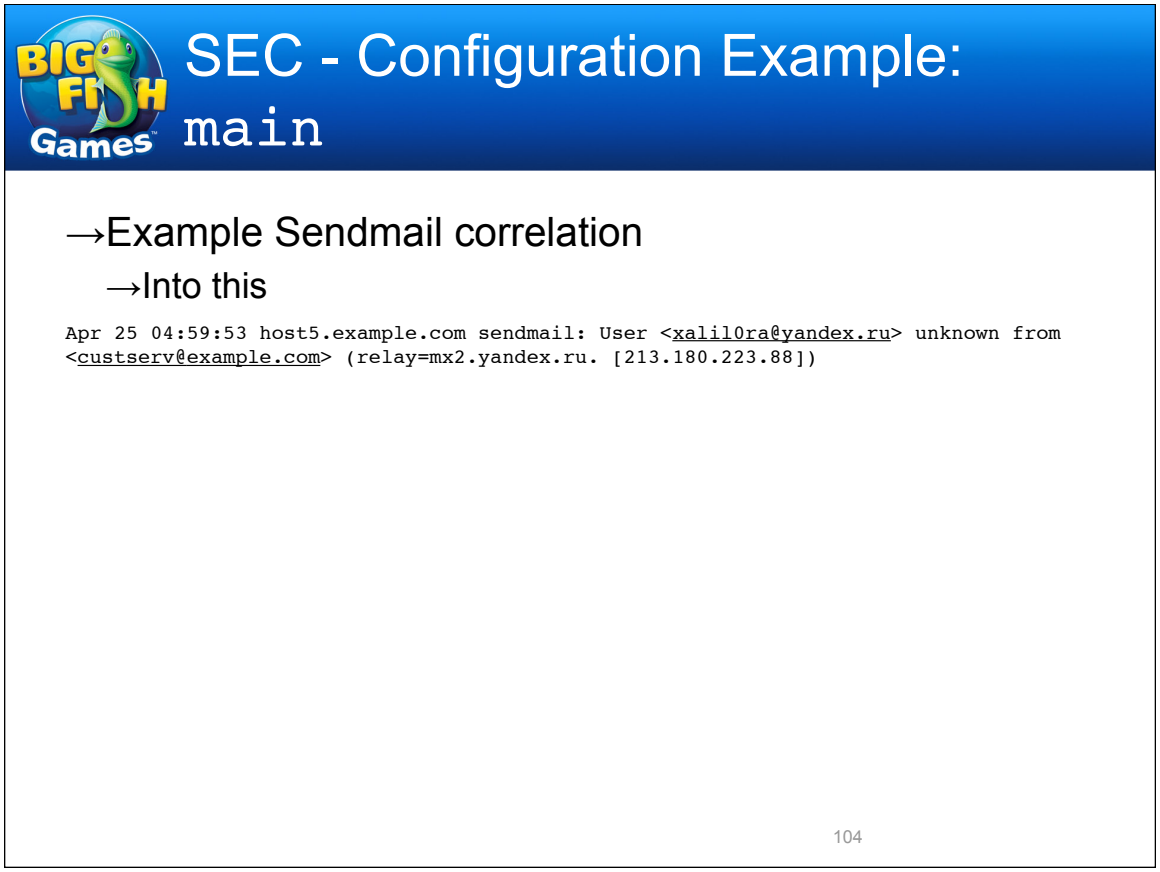
→Example Sendmail correlation

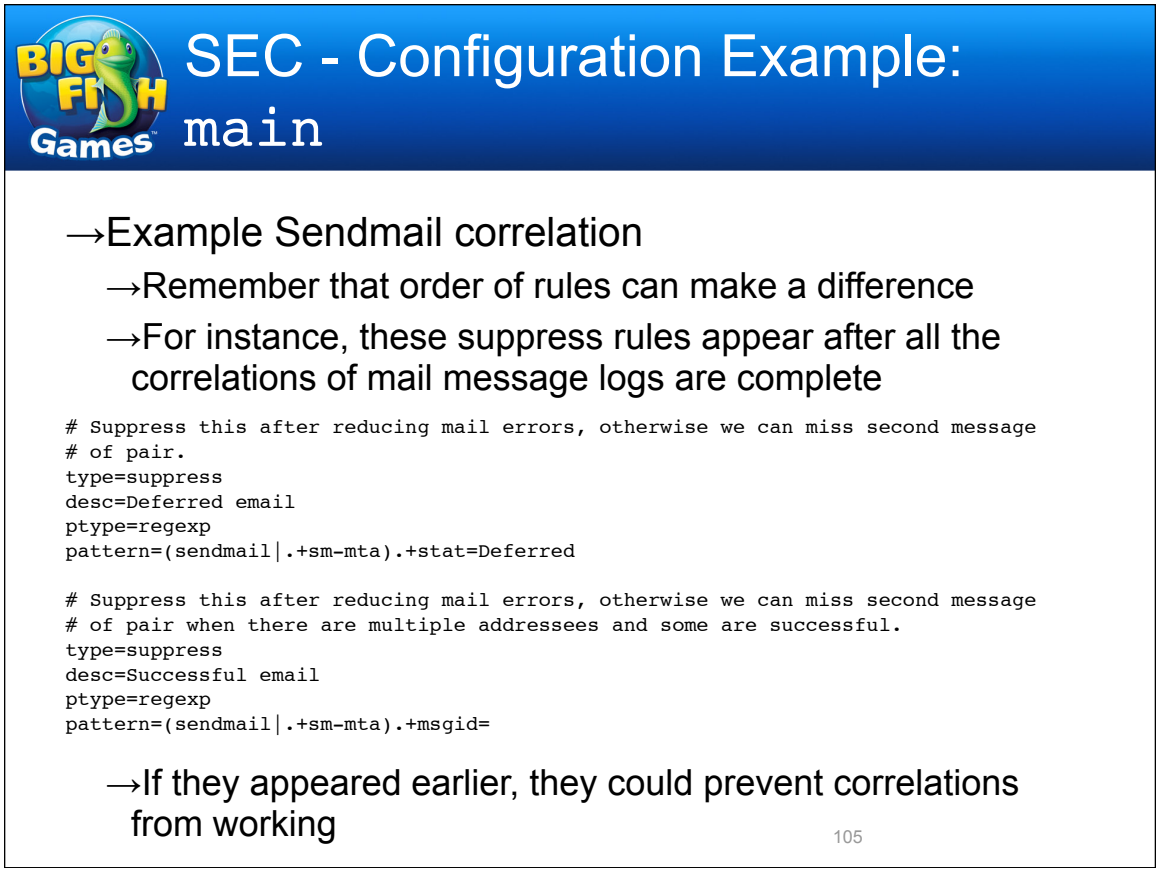
```
type=single
desc=Mail receiver unknown
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) (sendmail|.+sm-mta)\[\d+\]: (\w+): to=
(\S+),.+(relay=[^,]+),.+stat=(User unknown|Service unavailable)
action=copy MAIL_$2_$4 %from;\
        event 0 PARSED:$1 $2 sendmail: User $5 unknown from %from ($6)
context=MAIL_$2_$4
```

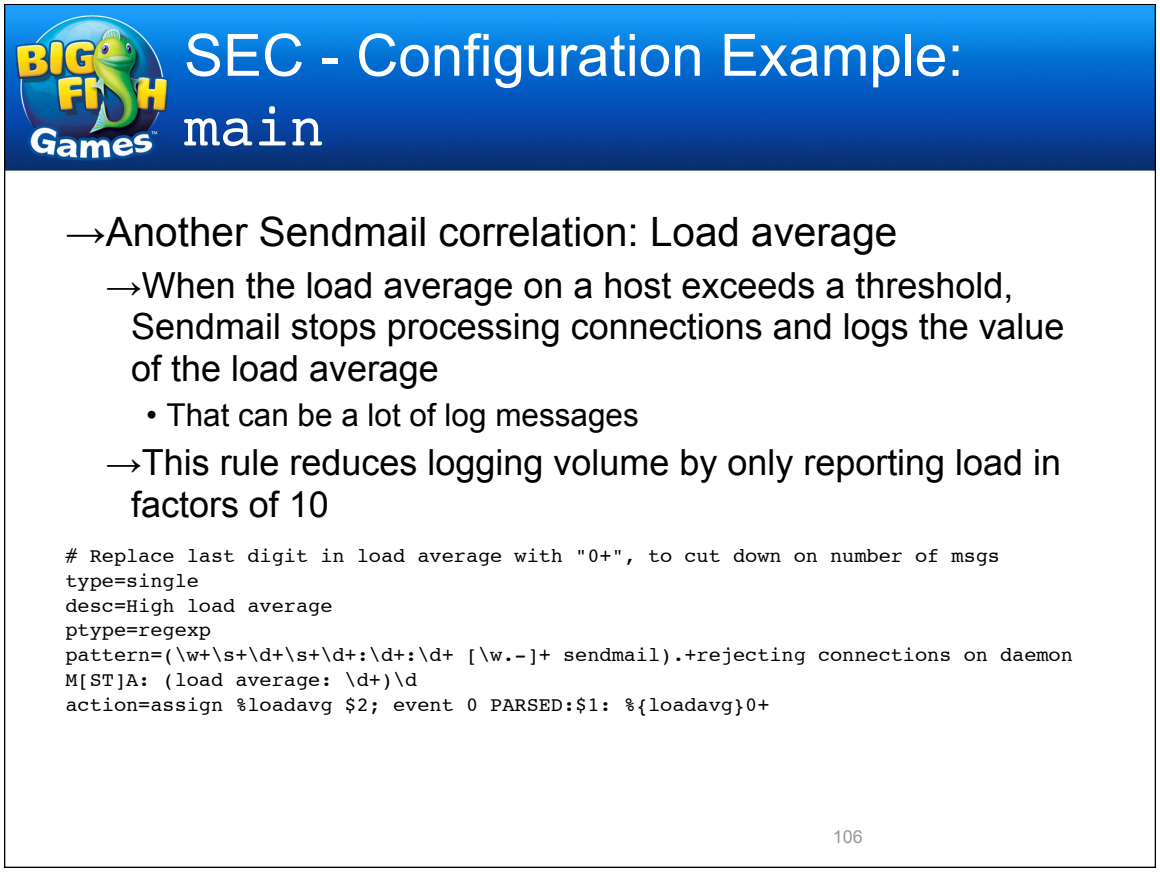
→This is one of several possible followup rules, depending on how the SMTP transaction goes

- Enabled when context from prior rule is in effect
- Copies the sender address from the context into a variable, uses it to construct a single correlated log message


SEC - Configuration Example:
main
→Example Sendmail correlation
→Along with some suppression rules, they turn this
Apr 25 04:59:46 host5.example.com sendmail[14779]: m3PBxkhp014779: Authentication-
Warning: host5.example.com: apache set sender to custserv@example.com using -f
Apr 25 04:59:46 host5.example.com sendmail[14779]: m3PBxkhp014779:
from=custserv@example.com, size=738, class=0, nrcpts=1,
msgid=<200804251159.m3PBxkhp014779@host5.example.com>, relay=apache@localhost
Apr 25 04:59:46 host5.example.com sendmail[14782]: m3PBxkHF014782:
from=<custserv@example.com>, size=1076, class=0, nrcpts=1,
msgid=<200804251159.m3PBxkhp014779@host5.example.com>, proto=ESMTP, daemon=MTA,
relay=localhost.localdomain [127.0.0.1]
Apr 25 04:59:46 host5.example.com sendmail[14779]: m3PBxkhp014779:
to=xalil0ra@yandex.ru, ctladdr=custserv@example.com (48/48), delay=00:00:00,
xdelay=00:00:00, mailer=relay, pri=30738, relay=[127.0.0.1] [127.0.0.1], dsn=2.0.0,
stat=Sent (m3PBxkHF014782 Message accepted for delivery)
Apr 25 04:59:53 host5.example.com sendmail[14784]: m3PBxkHF014782:
to=<xalil0ra@yandex.ru>, delay=00:00:07, xdelay=00:00:07, mailer=esmtp, pri=121076,
relay=mx2.yandex.ru. [213.180.223.88], dsn=5.1.1, stat=User unknown
Apr 25 04:59:53 host5.example.com sendmail[14784]: m3PBxkHF014782: m3PBxrHF014784:
DSN: User unknown


SEC - Configuration Example:
main
→Example Sendmail correlation
→Into this
Apr 25 04:59:53 host5.example.com sendmail: User <xalil0ra@yandex.ru> unknown from
<custserv@example.com> (relay=mx2.yandex.ru. [213.180.223.88])


SEC - Configuration Example: main
→Example Sendmail correlation
→Remember that order of rules can make a difference
→For instance, these suppress rules appear after all the correlations of mail message logs are complete
# Suppress this after reducing mail errors, otherwise we can miss second message
# of pair.
type=suppress
desc=Deferred email
ptype=regexp
pattern=(sendmail|.+sm-mta).+stat=Deferred
# Suppress this after reducing mail errors, otherwise we can miss second message
# of pair when there are multiple addressees and some are successful.
type=suppress
desc=Successful email
ptype=regexp
pattern=(sendmail|.+sm-mta).+msgid=
→If they appeared earlier, they could prevent correlations from working
105


SEC - Configuration Example: main
→Another Sendmail correlation: Load average
→When the load average on a host exceeds a threshold, Sendmail stops processing connections and logs the value of the load average
• That can be a lot of log messages
→This rule reduces logging volume by only reporting load in factors of 10
# Replace last digit in load average with "0+", to cut down on number of msgs
type=single
desc=High load average
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+ [\w.-]+ sendmail).+rejecting connections on daemon
M[ST]A: (load average: \d+)\d
action=assign %loadavg $2; event 0 PARSED:$1: %{loadavg}0+
106

SEC - Configuration Example: `main`

→Another Sendmail correlation: Load average

→Turns this

```
Apr 17 17:54:19 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 48
Apr 17 17:54:34 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 54
Apr 17 17:54:49 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 58
Apr 17 17:55:04 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 57
Apr 17 17:55:19 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 57
Apr 17 17:55:34 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 58
Apr 17 17:55:49 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 54
Apr 17 17:56:04 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 52
Apr 17 17:56:19 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 51
Apr 17 17:56:34 host3.example.com sendmail[2942]: rejecting connections on daemon MTA: load average: 48
```

→Into this

```
Apr 17 17:54:19 host3.example.com sendmail: load average: 40+
Apr 17 17:54:34 host3.example.com sendmail: load average: 50+
```


SEC - Configuration Example: `main`

→Detecting syslog-ng overflow

→syslog-ng logs statistics on messages it's processed

- Via internal source, every 10 minutes by default
- Messages look like this

```
Apr 25 08:40:05 loghost.example.com syslog-ng[16671]: Log statistics;
dropped='program(`/usr/local/bin/secStart emerg`)=0', dropped='program(`/usr/local/
bin/secStart main`)=0', dropped='program(`/usr/local/bin/secStart nmi`)=0',
dropped='program(`/usr/local/bin/secStart outbound`)=0', processed='center(queued)
=2419827', processed='center(received)=1205703', processed='destination(d_emerg)
=3204', processed='destination(d_fac)=1205703', processed='destination(d_all)
=1205703', processed='destination(d_su)=13', processed='destination(d_nmi)=0',
processed='destination(d_outbound)=5204', processed='destination(d_int)=0',
processed='source(s_int)=0', processed='source(s_all)=1205703'
```

→Most of the time we don't care to see these, but if syslog-ng drops any messages, particularly to the `main` SEC process, we want to know

SEC - Configuration Example: main

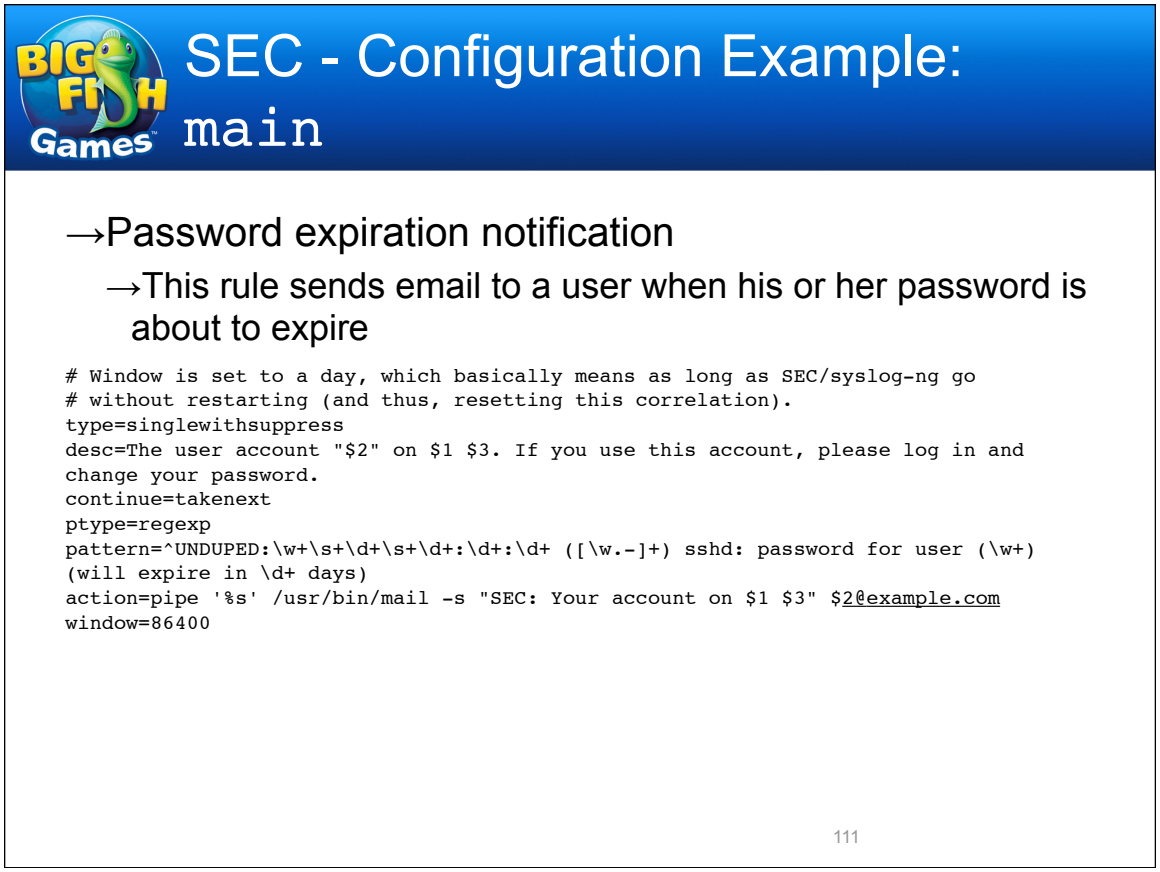
→Detecting syslog-ng overflow

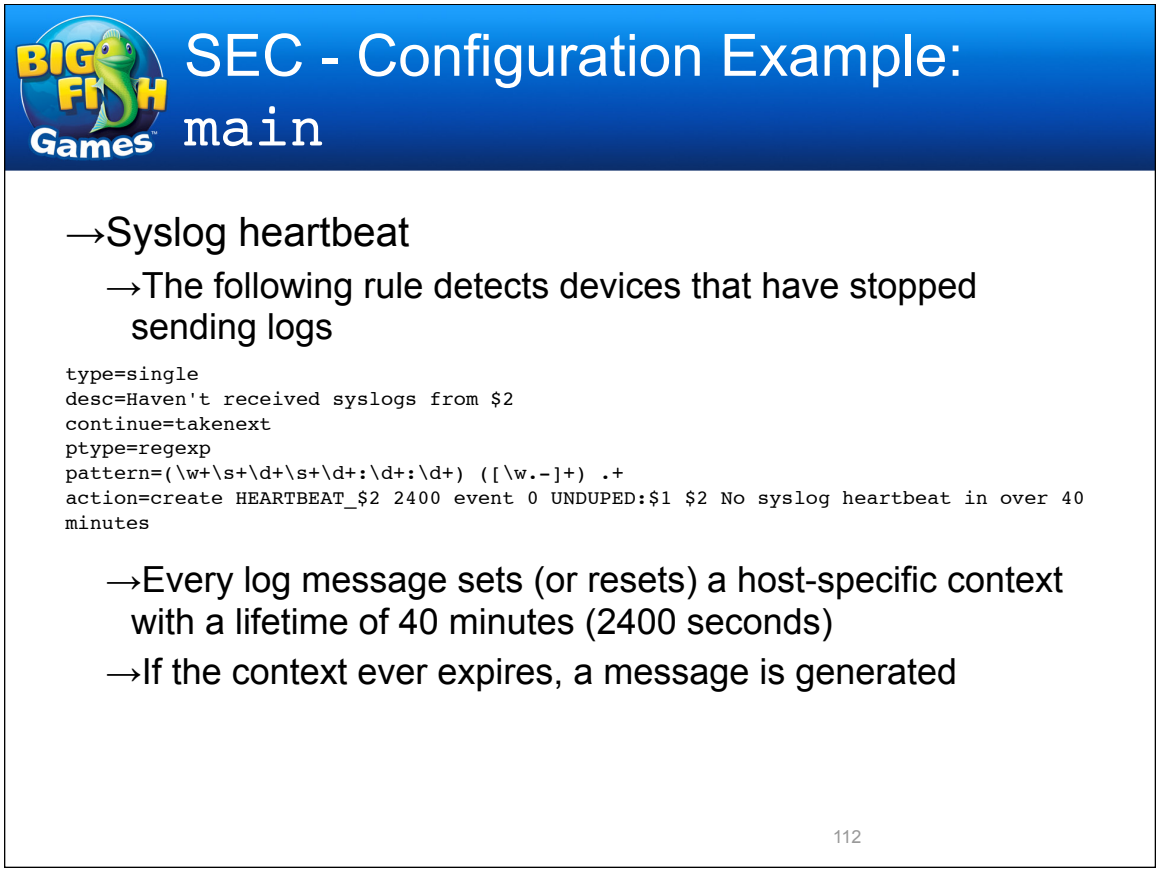
```
# Drop regular log stats reports unless messages get dropped. If that happens,
# send reduced message, but not too frequently, since this won't go away until
# syslog-ng is restarted.
type=suppress
ptype=regexp
pattern=loghost\.intelius\.com syslog-ng\[\d+\]: Log statistics\;.+dropped=\'program
\(\`\/usr\/local\/bin\/secStart main\`\)=0\'

type=singlewithsuppress
desc=Dropped $2 messages
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+ loghost\.intelius\.com syslog-ng)\[\d+\]: Log
statistics\;.+dropped=\'program\(\`\/usr\/local\/bin\/secStart main\`\)=(\d+)\'
action=event 0 PARSED:$1: dropped $2 messages
window=3600
```


SEC - Configuration Example: main

→Email sent when overflow detected

```
type=single
desc=syslog-ng overwhelmed
continue=takenext
ptype=regexp
pattern=^UNDUPED:(\w+\s+\d+\s+\d+:\d+:\d+ log1\.tuk\.intelius\.com syslog-ng:
dropped \d+ messages)
action=pipe '$1' /bin/mail -s "SEC: syslog-ng message buffer overrun" %a
```


SEC - Configuration Example:
main
→Password expiration notification
→This rule sends email to a user when his or her password is about to expire
# Window is set to a day, which basically means as long as SEC/syslog-ng go
# without restarting (and thus, resetting this correlation).
type=singlewithsuppress
desc=The user account "$2" on $1 $3. If you use this account, please log in and
change your password.
continue=takenext
ptype=regexp
pattern=^UNDUPED:\w+\s+\d+\s+\d+:\d+:\d+ ([\w.-]+) sshd: password for user (\w+)
(will expire in \d+ days)
action=pipe '%s' /usr/bin/mail -s "SEC: Your account on $1 $3" $2@example.com
window=86400
111


SEC - Configuration Example:
main
→Syslog heartbeat
→The following rule detects devices that have stopped sending logs
type=single
desc=Haven't received syslogs from $2
continue=takenext
ptype=regexp
pattern=(\w+\s+\d+\s+\d+:\d+:\d+) ([\w.-]+) .+
action=create HEARTBEAT_$2 2400 event 0 UNDUPED:$1 $2 No syslog heartbeat in over 40
minutes
→Every log message sets (or resets) a host-specific context with a lifetime of 40 minutes (2400 seconds)
→If the context ever expires, a message is generated
112

SEC - Configuration Example: `main`

→Syslog heartbeat

→This rule converts the message to an email

```
type=single
desc=Haven't received syslogs from $1 $2
ptype=regexp
pattern=^UNDUPED:\w+\s+\d+\s+\d+:\d+:\d+ ([\w.-]+) No syslog heartbeat (in over .+)
action=pipe '' /bin/mail -s "SEC: No contact from $1" %a
```

→Useful for a number of situations

- Host is down, and network monitoring not in place
- Syslog daemon dies, and process monitoring not in place
- Syslog misconfigured, and configuration management not in place
- Network device stops forwarding syslogs

SEC - Conclusion

→By far, the bulk of the setup work is creating the log filters

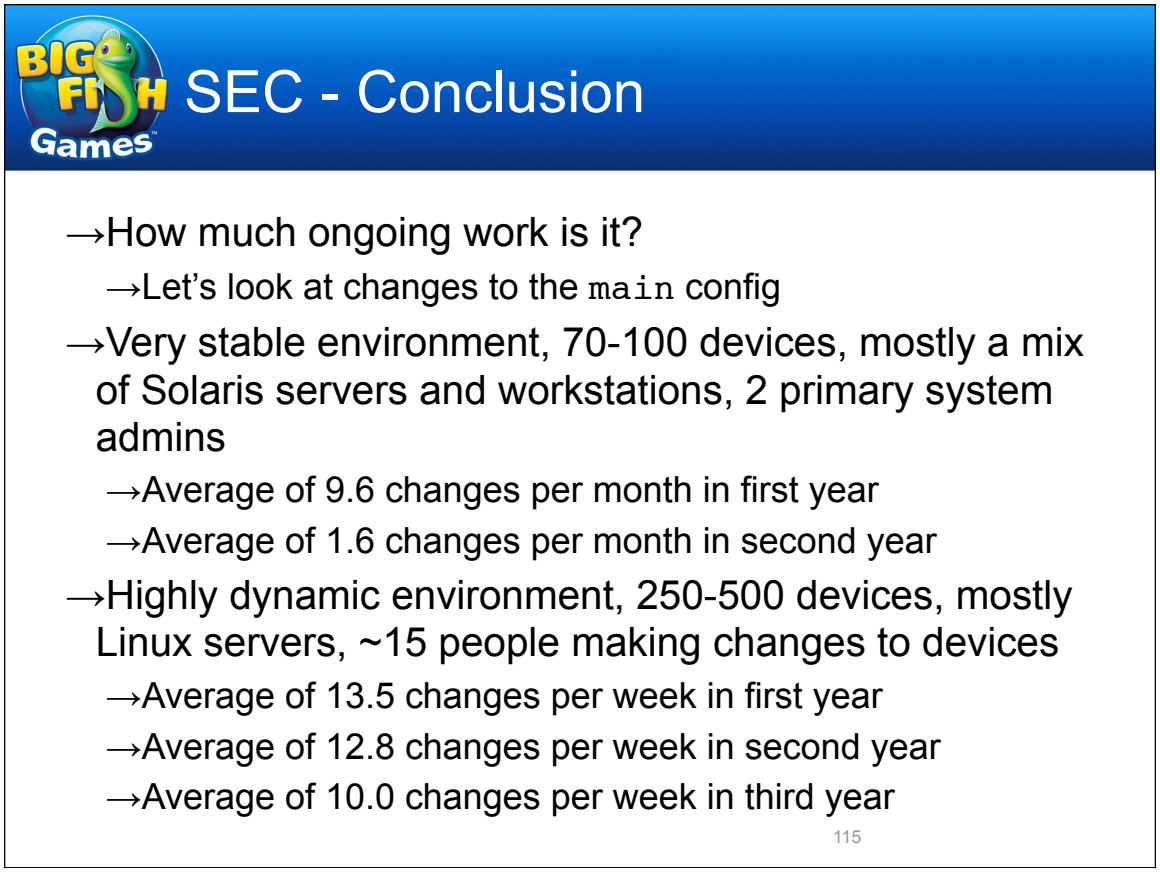
→The process is iterative

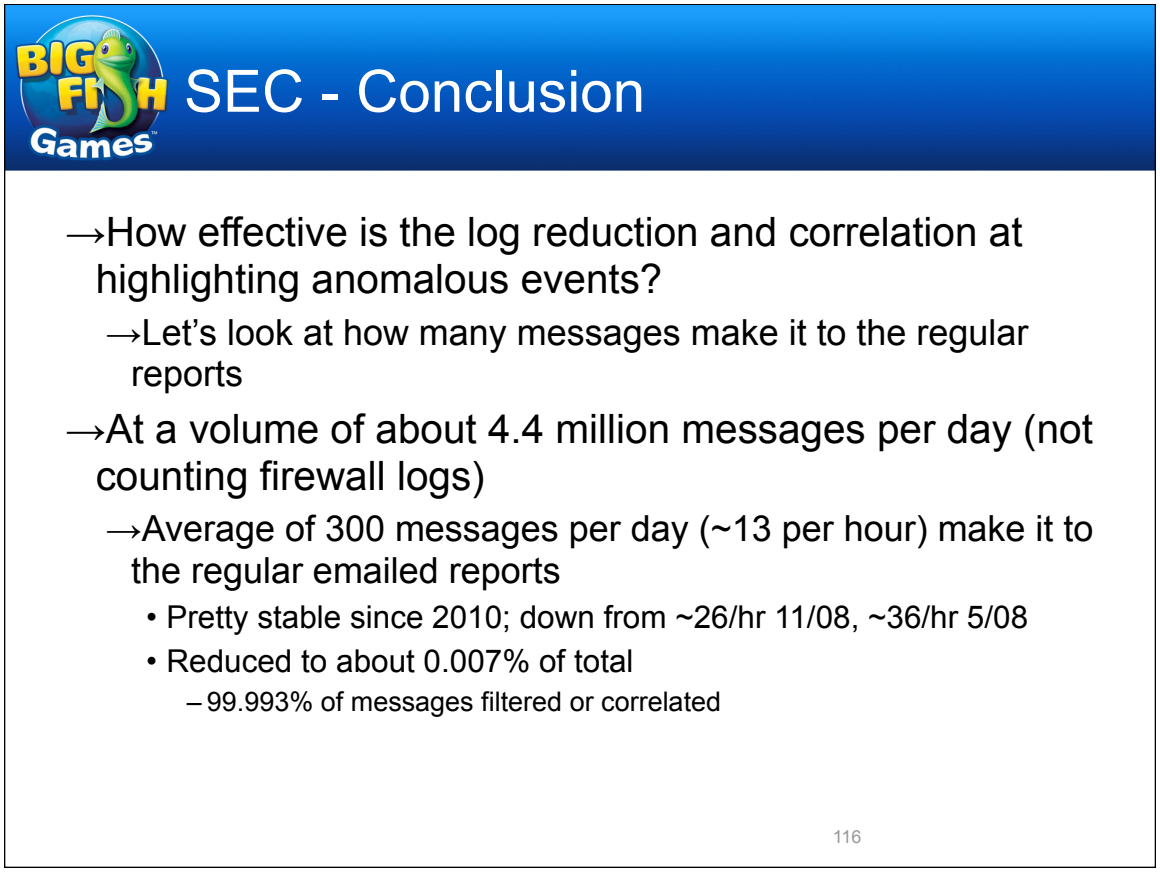
- Let logs through, figure out what you don't care to see, create filters to suppress or correlate
- Repeat until volume is bearable

→Learn your Perl regular expressions

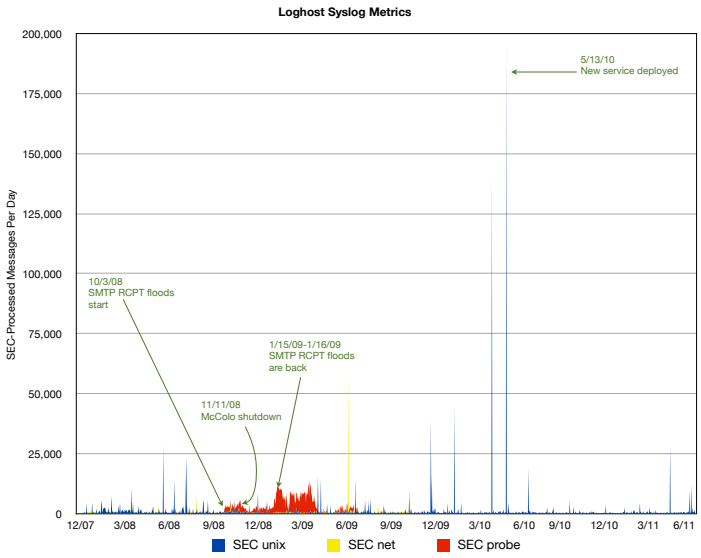
→Missing important log messages is bad

→But having so many to look at that you ignore them can be just as bad


SEC - Conclusion
→How much ongoing work is it?
→Let's look at changes to the main config
→Very stable environment, 70-100 devices, mostly a mix of Solaris servers and workstations, 2 primary system admins
→Average of 9.6 changes per month in first year
→Average of 1.6 changes per month in second year
→Highly dynamic environment, 250-500 devices, mostly Linux servers, ~15 people making changes to devices
→Average of 13.5 changes per week in first year
→Average of 12.8 changes per week in second year
→Average of 10.0 changes per week in third year
115


SEC - Conclusion
→How effective is the log reduction and correlation at highlighting anomalous events?
→Let's look at how many messages make it to the regular reports
→At a volume of about 4.4 million messages per day (not counting firewall logs)
→Average of 300 messages per day (~13 per hour) make it to the regular emailed reports
• Pretty stable since 2010; down from ~26/hr 11/08, ~36/hr 5/08
• Reduced to about 0.007% of total
– 99.993% of messages filtered or correlated
116

SEC - Conclusion



SEC-Processed Messages Per Day

SEC - Conclusion

→What is the drain on system resources imposed by SEC?

→As stated earlier, normal volume is ~4.4M msgs/day

→Each message is processed at least once by SEC, often multiple times

→Many messages are held in memory due to contexts, `pairwithwindow` rules, etc.

→At that rate

→Smaller processes (`disk`, `emerg`, `hitemp`) each take up about 8 MB of RAM and negligible CPU

→The `main` process uses ~14 MB RAM and 10% CPU

SEC - Conclusion

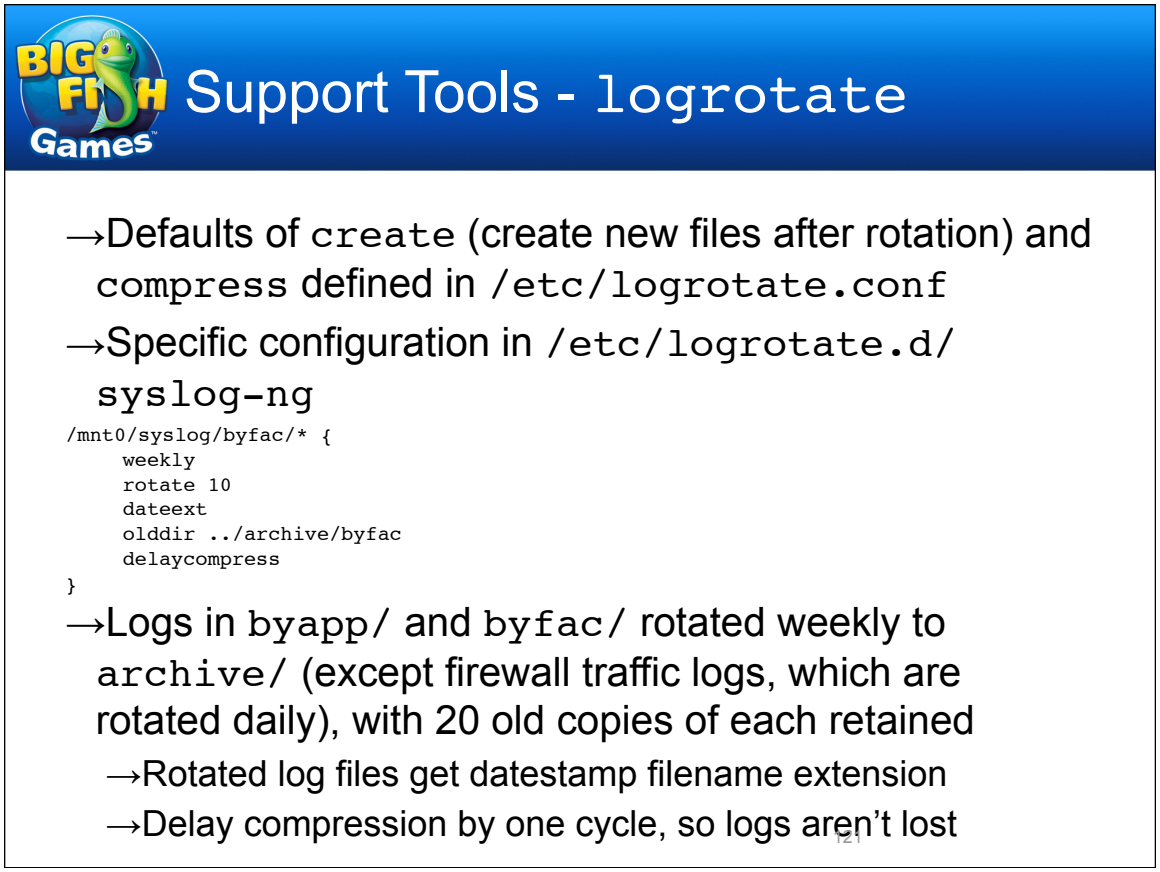
→Juniper NetScreen firewall traffic logs used to be processed by a specialized SEC config, bypassing the `main` config

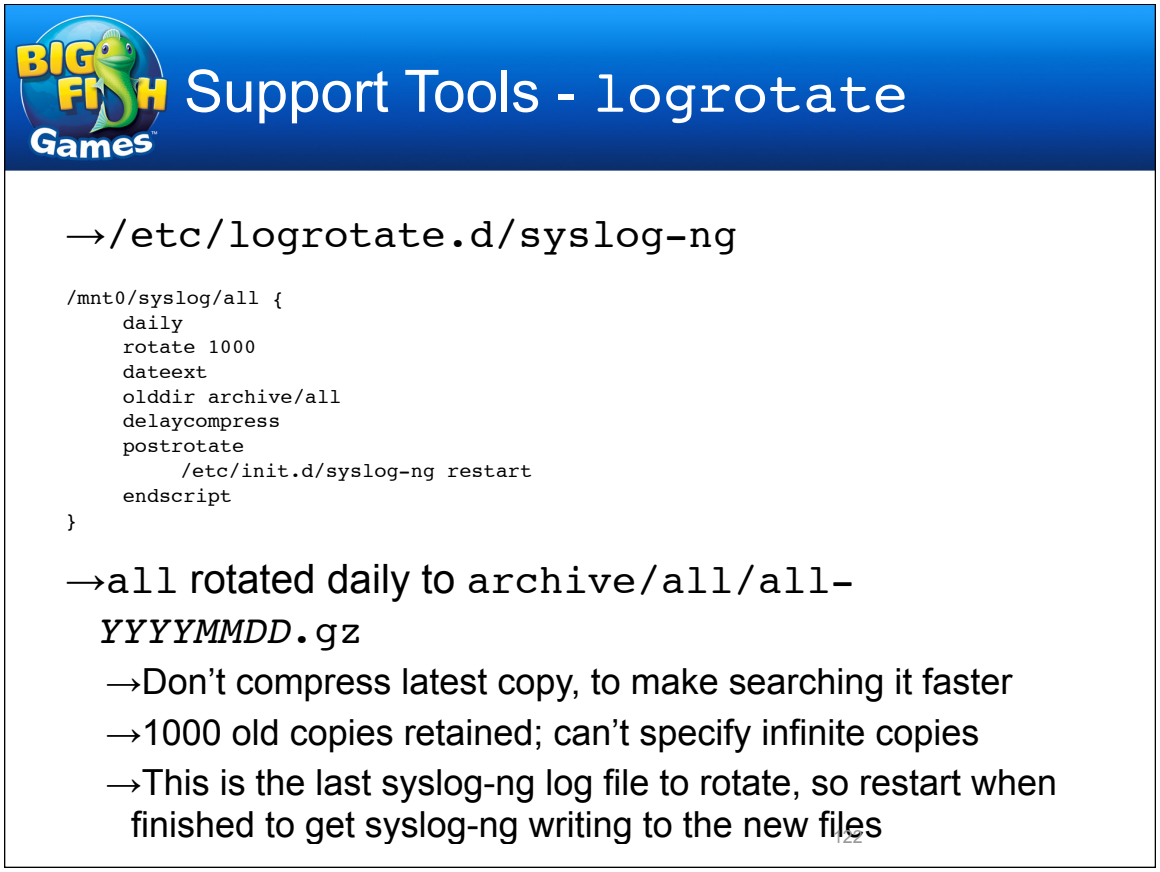
→Only one `singlewithsuppress` rule that rewrote the logs into a simpler format

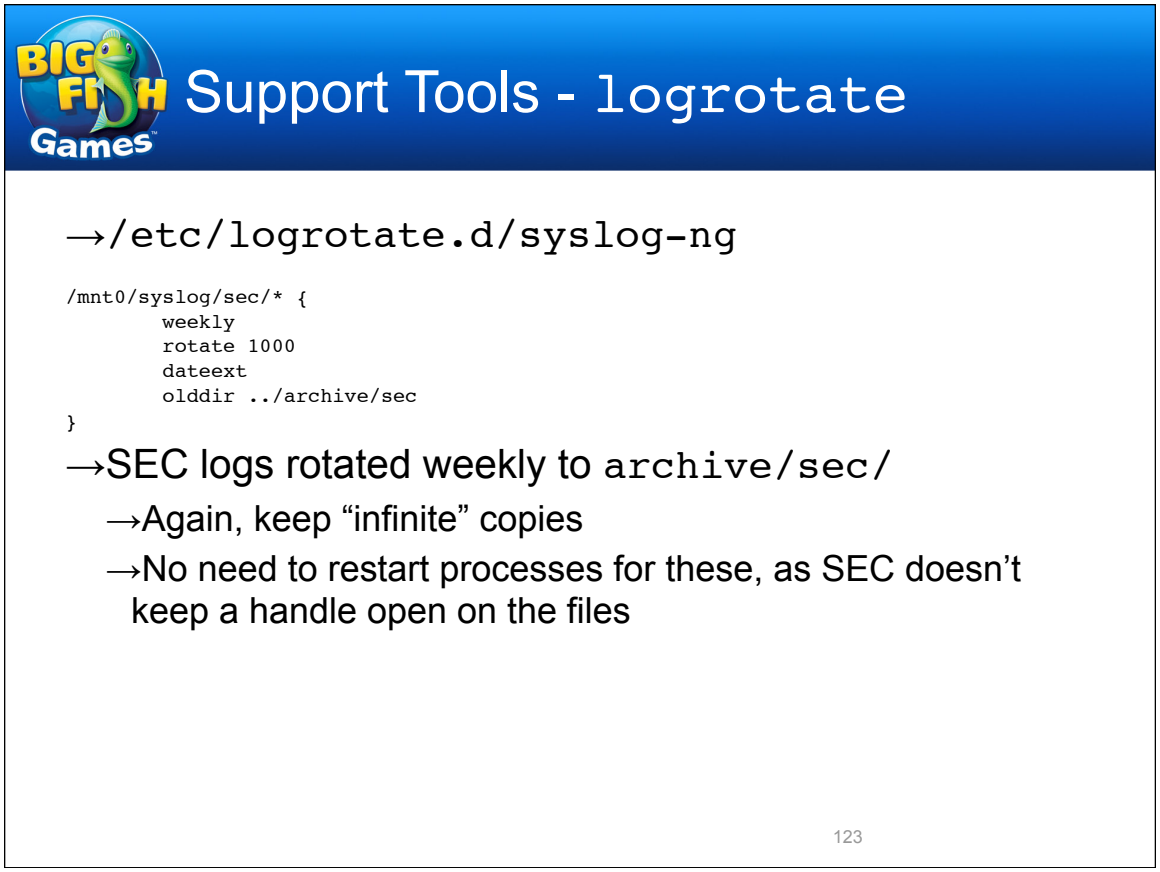
→Volume: 55-75 million msgs/day

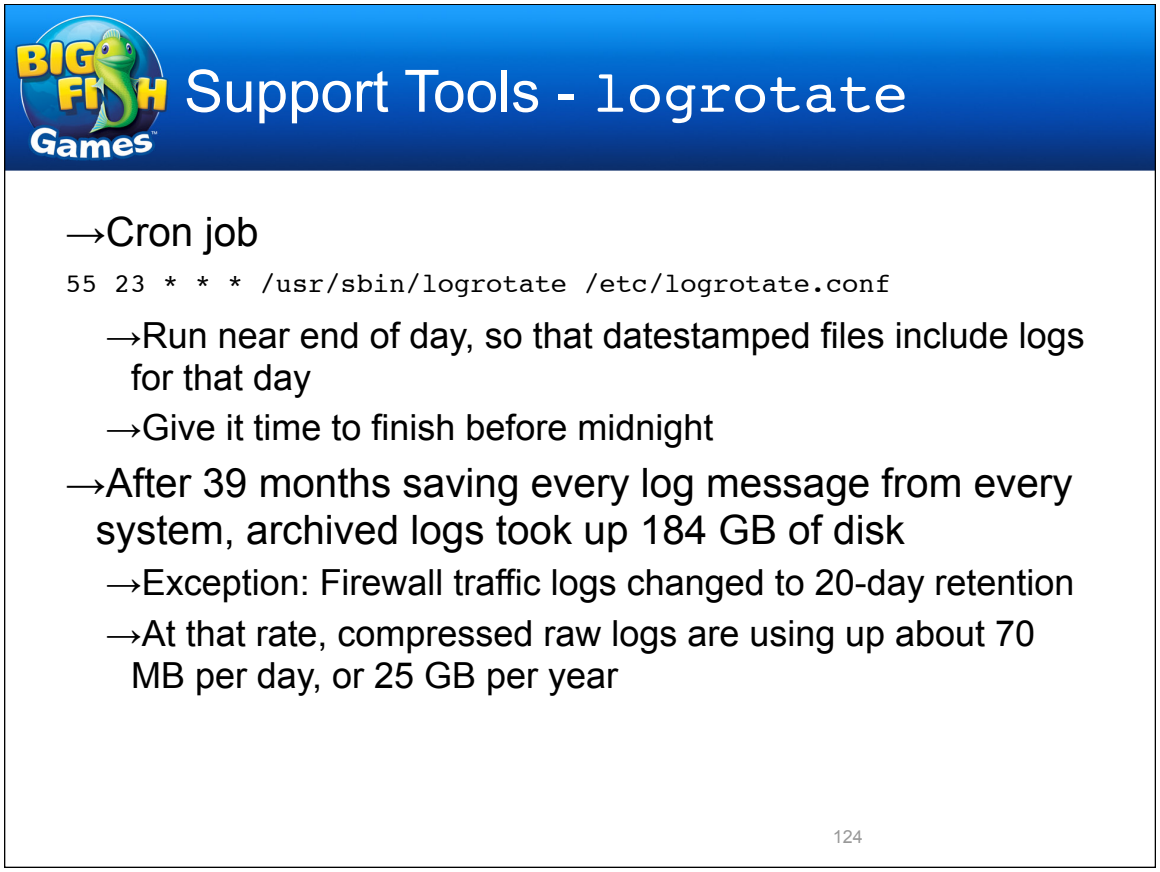
→This SEC process used nearly 200 MB and ~65% of a CPU

Support Tools


Support Tools - logrotate
→Defaults of create (create new files after rotation) and compress defined in /etc/logrotate.conf
→Specific configuration in /etc/logrotate.d/syslog-ng
/mnt0/syslog/byfac/* {
weekly
rotate 10
dateext
olddir ../archive/byfac
delaycompress
}
→Logs in byapp/ and byfac/ rotated weekly to archive/ (except firewall traffic logs, which are rotated daily), with 20 old copies of each retained
→Rotated log files get datestamp filename extension
→Delay compression by one cycle, so logs aren't lost
121


Support Tools - logrotate
→/etc/logrotate.d/syslog-ng
/mnt0/syslog/all {
daily
rotate 1000
dateext
olddir archive/all
delaycompress
postrotate
/etc/init.d/syslog-ng restart
endscript
}
→all rotated daily to archive/all/all-YYYYMMDD.gz
→Don't compress latest copy, to make searching it faster
→1000 old copies retained; can't specify infinite copies
→This is the last syslog-ng log file to rotate, so restart when finished to get syslog-ng writing to the new files
122


Support Tools - logrotate
→/etc/logrotate.d/syslog-ng
/mnt0/syslog/sec/* {
weekly
rotate 1000
dateext
olddir ../archive/sec
}
→SEC logs rotated weekly to archive/sec/
→Again, keep “infinite” copies
→No need to restart processes for these, as SEC doesn’t keep a handle open on the files
123


Support Tools - logrotate
→Cron job
55 23 * * * /usr/sbin/logrotate /etc/logrotate.conf
→Run near end of day, so that datestamped files include logs for that day
→Give it time to finish before midnight
→After 39 months saving every log message from every system, archived logs took up 184 GB of disk
→Exception: Firewall traffic logs changed to 20-day retention
→At that rate, compressed raw logs are using up about 70 MB per day, or 25 GB per year
124

Support Tools - `logadm`

→Included with Solaris since version 9

→Not quite as flexible as `logrotate` in most ways, and the config file is a little harder to understand, but certainly good enough

→Configured in `/etc/logadm.conf`

→Can be manually edited, or via `logadm` commands

Support Tools - `logadm`

→Key to example `logadm.conf` lines

→ `-C` - Retain this many old copies (0 for unlimited)

→ `-N` - Don't complain about missing log files

→ `-c` - Rotate by copying file then truncating

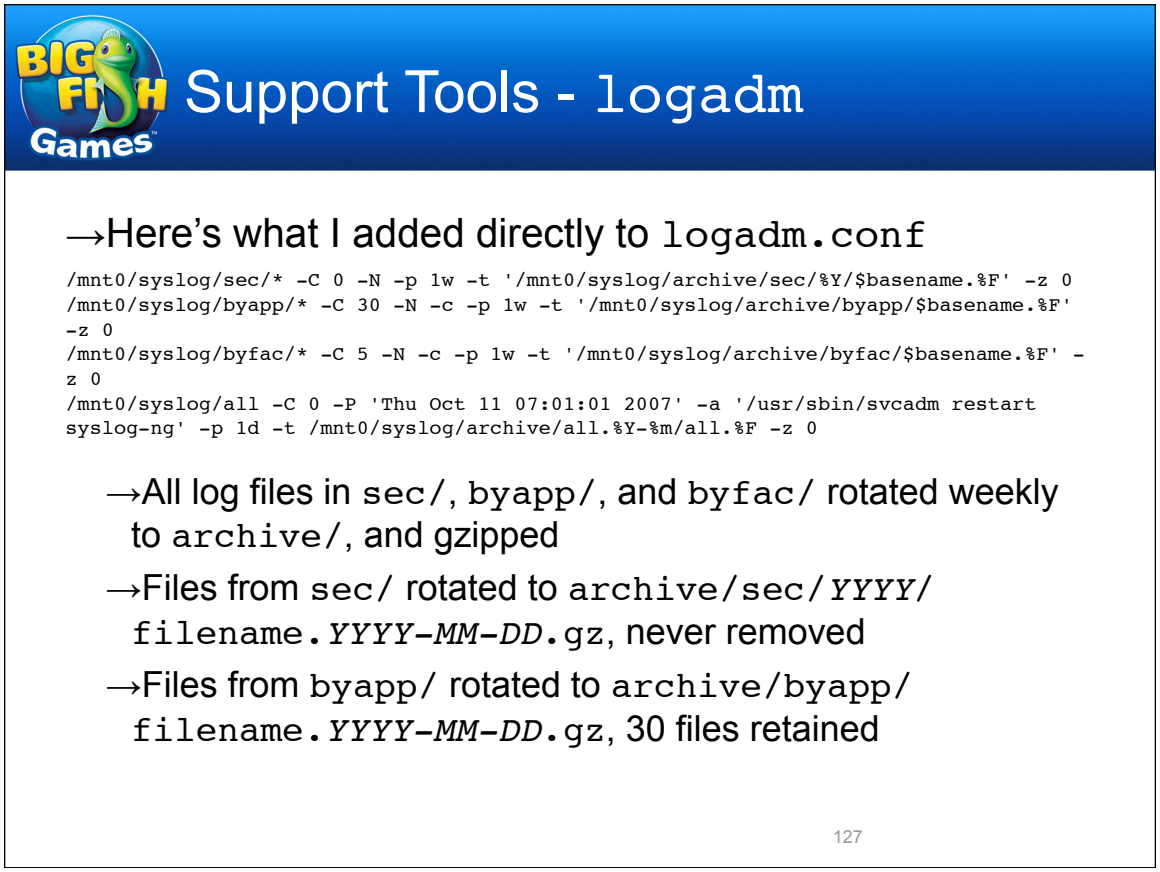
→ `-p` - Rotate this often

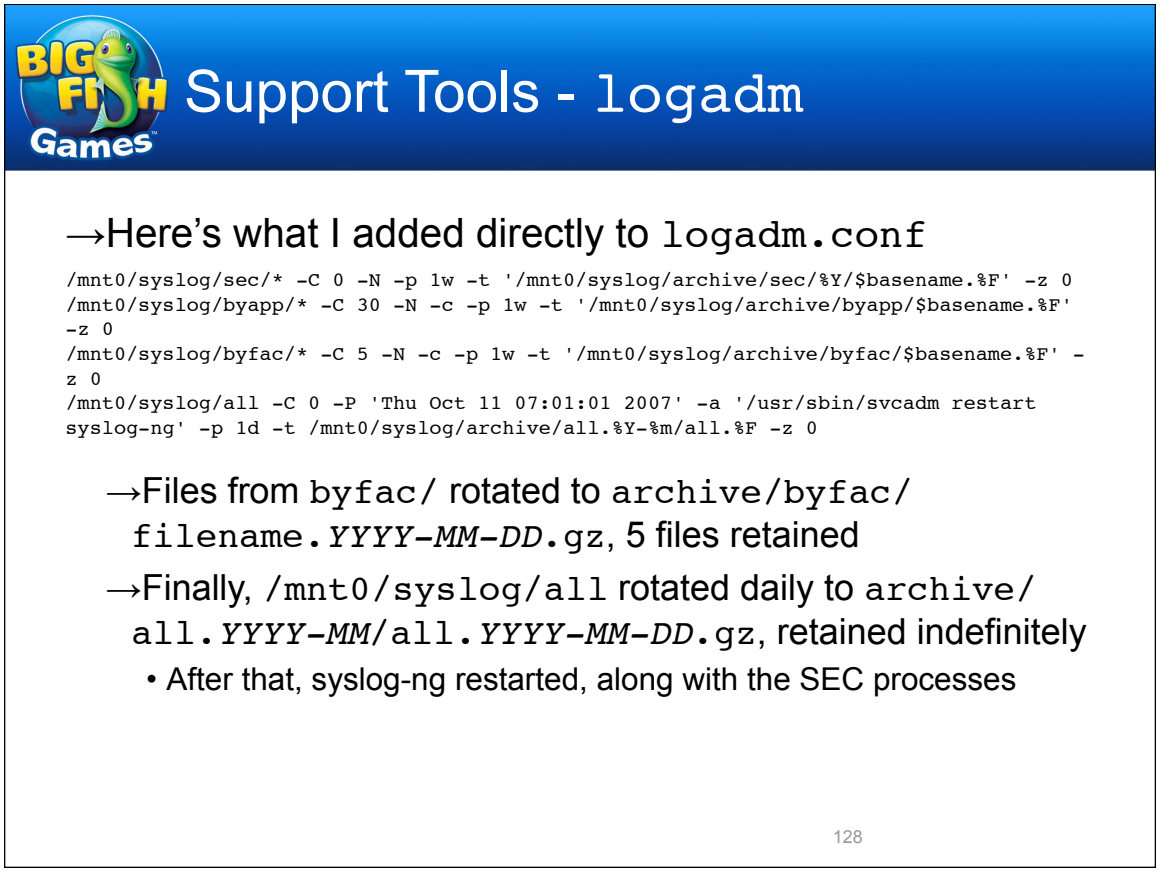
→ `-P` - Time of last rotation, in UTC (automatically updated)

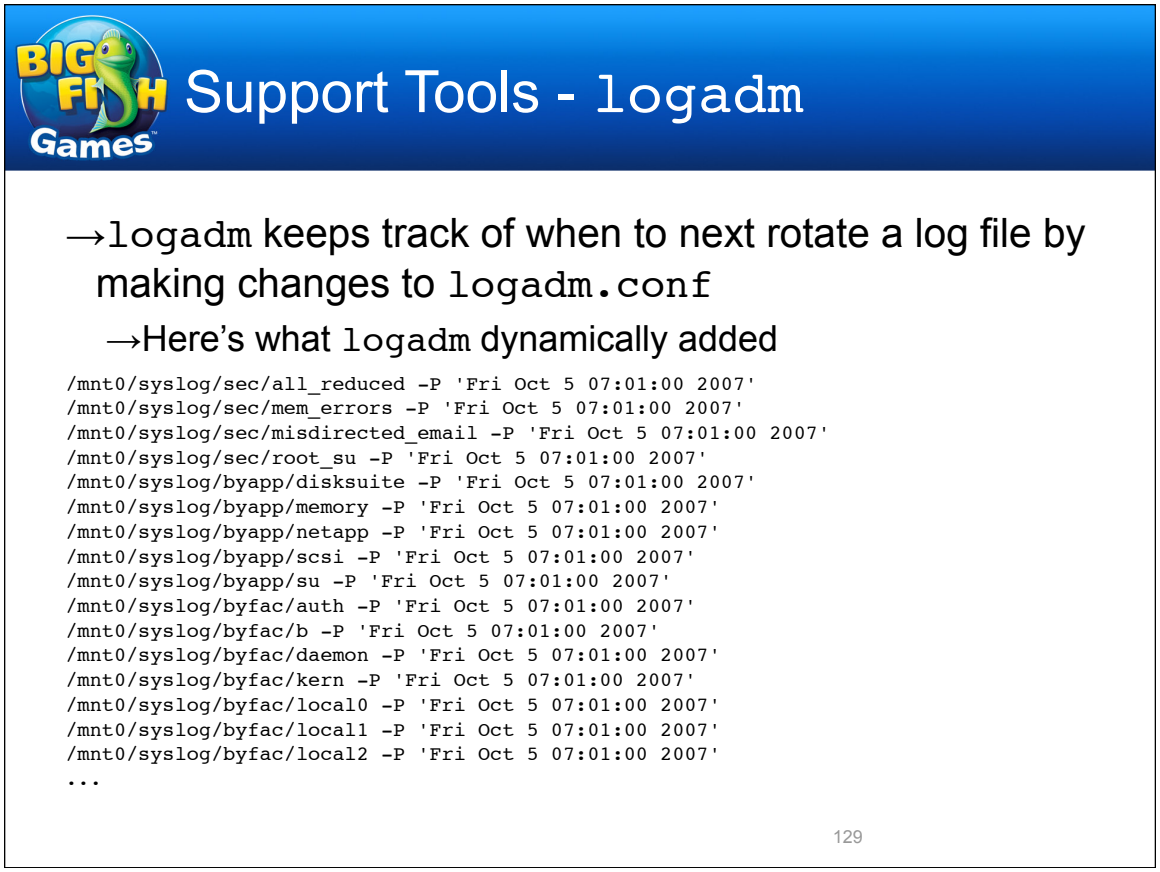
→ `-t` - Name of rotated file (including macros)

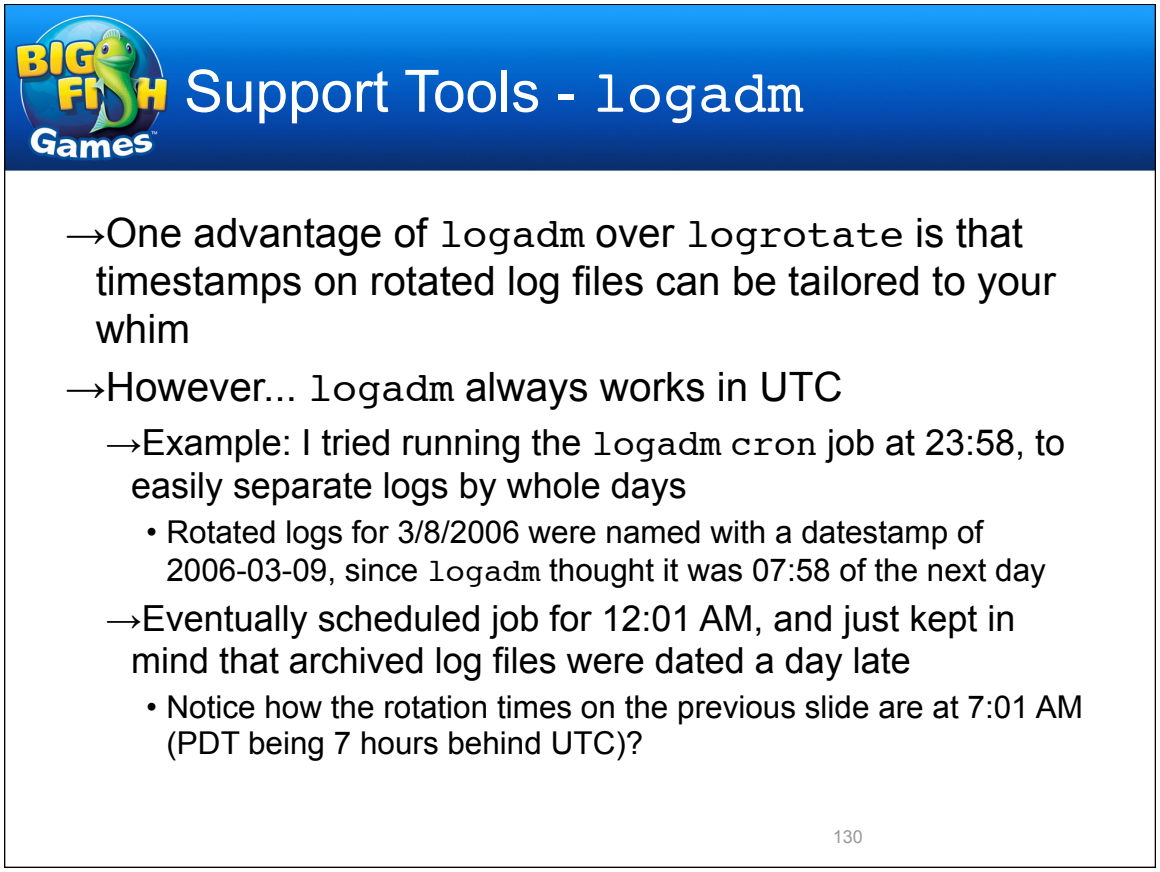
→ `-z` - Compress rotated files with `gzip`, keeping this many uncompressed (doesn't seem to work properly)

→ `-a` - Execute this command after rotation


Support Tools - logadm
→Here's what I added directly to logadm.conf
/mnt0/syslog/sec/* -C 0 -N -p 1w -t '/mnt0/syslog/archive/sec/%Y/$basename.%F' -z 0
/mnt0/syslog/byapp/* -C 30 -N -c -p 1w -t '/mnt0/syslog/archive/byapp/$basename.%F'
-z 0
/mnt0/syslog/byfac/* -C 5 -N -c -p 1w -t '/mnt0/syslog/archive/byfac/$basename.%F' -
z 0
/mnt0/syslog/all -C 0 -P 'Thu Oct 11 07:01:01 2007' -a '/usr/sbin/svcadm restart
syslog-ng' -p 1d -t /mnt0/syslog/archive/all.%Y-%m/all.%F -z 0
→All log files in sec/, byapp/, and byfac/ rotated weekly to archive/, and gzipped
→Files from sec/ rotated to archive/sec/YYYY/filename.YYYY-MM-DD.gz, never removed
→Files from byapp/ rotated to archive/byapp/filename.YYYY-MM-DD.gz, 30 files retained
127


Support Tools - logadm
→Here's what I added directly to logadm.conf
/mnt0/syslog/sec/* -C 0 -N -p 1w -t '/mnt0/syslog/archive/sec/%Y/$basename.%F' -z 0
/mnt0/syslog/byapp/* -C 30 -N -c -p 1w -t '/mnt0/syslog/archive/byapp/$basename.%F'
-z 0
/mnt0/syslog/byfac/* -C 5 -N -c -p 1w -t '/mnt0/syslog/archive/byfac/$basename.%F' -
z 0
/mnt0/syslog/all -C 0 -P 'Thu Oct 11 07:01:01 2007' -a '/usr/sbin/svcadm restart
syslog-ng' -p 1d -t /mnt0/syslog/archive/all.%Y-%m/all.%F -z 0
→Files from byfac/ rotated to archive/byfac/filename.YYYY-MM-DD.gz, 5 files retained
→Finally, /mnt0/syslog/all rotated daily to archive/all.YYYY-MM/all.YYYY-MM-DD.gz, retained indefinitely
• After that, syslog-ng restarted, along with the SEC processes
128


Support Tools - logadm
→logadm keeps track of when to next rotate a log file by making changes to logadm.conf
→Here's what logadm dynamically added
/mnt0/syslog/sec/all_reduced -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/sec/mem_errors -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/sec/misdirected_email -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/sec/root_su -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byapp/disksuite -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byapp/memory -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byapp/netapp -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byapp/scsi -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byapp/su -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byfac/auth -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byfac/b -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byfac/daemon -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byfac/kern -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byfac/local0 -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byfac/local1 -P 'Fri Oct 5 07:01:00 2007'
/mnt0/syslog/byfac/local2 -P 'Fri Oct 5 07:01:00 2007'
...
129


Support Tools - logadm
→One advantage of logadm over logrotate is that timestamps on rotated log files can be tailored to your whim
→However... logadm always works in UTC
→Example: I tried running the logadm cron job at 23:58, to easily separate logs by whole days
• Rotated logs for 3/8/2006 were named with a datestamp of 2006-03-09, since logadm thought it was 07:58 of the next day
→Eventually scheduled job for 12:01 AM, and just kept in mind that archived log files were dated a day late
• Notice how the rotation times on the previous slide are at 7:01 AM (PDT being 7 hours behind UTC)?
130

Support Tools - tidyLogArchives

→tidyLogArchives

- →Since logrotate can't generate timestamps the ways I'd like, I needed something else to do final archiving
- →Script cleans up after logrotate by moving old all logs into subdirectories named for month and year, and old SEC logs into subdirectories named for year
- →Runs from cron once a month, on the 2nd

Support Tools - tidyLogArchives

```
#!/bin/sh
#
# tidyLogArchives - Move archived logs into subdirs.

PATH=/bin:/usr/bin

LOG_DIR=/mnt0/syslog
ARCH_DIR=$LOG_DIR/archive

YEAR=`date +%Y`
MON=`date +%m`
case $MON in
       01)    lastMon="12"; year=`expr $YEAR - 1`;;
       02)    lastMon="01"; year=$YEAR;;
       03)    lastMon="02"; year=$YEAR;;
       04)    lastMon="03"; year=$YEAR;;
       05)    lastMon="04"; year=$YEAR;;
       06)    lastMon="05"; year=$YEAR;;
       07)    lastMon="06"; year=$YEAR;;
       08)    lastMon="07"; year=$YEAR;;
       09)    lastMon="08"; year=$YEAR;;
       10)    lastMon="09"; year=$YEAR;;
       11)    lastMon="10"; year=$YEAR;;
       12)    lastMon="11"; year=$YEAR;;
esac

cd $ARCH_DIR/all
mkdir -p -m 0750 ${year}-${lastMon}
chown syslog:syslog ${year}-${lastMon}
mv all-${year}${lastMon}*.gz ${year}-${lastMon}

cd $ARCH_DIR/sec
mkdir -p -m 0750 ${year}
chown syslog:syslog ${year}
mv *-${year}*.gz ${year}
```

tidyLogArchives

Support Tools - tidyLogArchives

```
drwxr-s---  2 syslog syslog      4096 Dec 21  2007 2007-11
drwxr-s---  2 syslog syslog      4096 Jan  1  2008 2007-12
drwxr-s---  2 syslog syslog      4096 Feb  4  2008 2008-01
drwxr-s---  2 syslog syslog      4096 Mar  1  2008 2008-02
drwxr-s---  2 syslog syslog      4096 Apr  1  2008 2008-03
drwxr-s---  2 syslog syslog      4096 May  1  2008 2008-04
drwxr-s---  2 syslog syslog      4096 Jun  2  2008 2008-05
drwxr-s---  2 syslog syslog      4096 Jul  1  2008 2008-06
...
drwxr-s---  2 syslog syslog      4096 Jul  2  2010 2010-06
drwxr-s---  2 syslog syslog      4096 Aug  2  2010 2010-07
drwxr-s---  2 syslog syslog      4096 Sep  2 00:50 2010-08
drwxr-s---  2 syslog syslog      4096 Oct  2 00:50 2010-09
drwxr-s---  2 syslog syslog      4096 Nov  2 00:50 2010-10
drwxr-s---  2 syslog syslog      4096 Dec  2 00:50 2010-11
drwxr-s---  2 syslog syslog      4096 Jan  2 00:50 2010-12
drwxr-s---  2 syslog syslog      4096 Feb  2 00:50 2011-01
-rw-r--r--  1 syslog syslog  59372496 Feb  2 23:57 all-20110201.gz
-rw-r--r--  1 syslog syslog  55499272 Feb  3 23:57 all-20110202.gz
-rw-r--r--  1 syslog syslog  60538970 Feb  4 23:57 all-20110203.gz
...
-rw-r--r--  1 syslog syslog  89945437 Feb 24 23:57 all-20110223.gz
-rw-r--r--  1 syslog syslog  80424155 Feb 25 23:57 all-20110224.gz
-rw-r--r--  1 syslog syslog  78360083 Feb 26 23:57 all-20110225.gz
-rw-r--r--  1 syslog syslog 846763243 Feb 26 23:57 all-20110226
```

Contents of `/mnt0/syslog/archive/all/`

Support Tools - tidyLogArchives

```
drwxr-s---  2 syslog syslog   4096 Jan  1  2008 2007
drwxr-s---  2 syslog syslog  16384 Jan  1  2009 2008
drwxr-s---  2 syslog syslog  20480 Jan  2  2010 2009
drwxr-s---  2 syslog syslog  12288 Jan  2 00:50 2010
drwxr-s---  2 syslog syslog   4096 Feb  2 00:50 2011
-rw-r--r--  1 syslog syslog    612 Feb  6 23:58 attack-20110206.gz
-rw-r--r--  1 syslog syslog    546 Feb 13 23:58 attack-20110213.gz
-rw-r--r--  1 syslog syslog    414 Feb 20 23:58 attack-20110220.gz
-rw-r--r--  1 syslog syslog    459 Feb  6 23:58 drupal-20110206.gz
-rw-r--r--  1 syslog syslog    875 Feb 13 23:58 drupal-20110213.gz
-rw-r--r--  1 syslog syslog    396 Feb 20 23:58 drupal-20110220.gz
-rw-r--r--  1 syslog syslog  73123 Feb  6 23:58 mail_custserv-20110206.gz
-rw-r--r--  1 syslog syslog  74295 Feb 13 23:58 mail_custserv-20110213.gz
-rw-r--r--  1 syslog syslog  71381 Feb 20 23:58 mail_custserv-20110220.gz
-rw-r--r--  1 syslog syslog   2475 Feb  6 23:58 mail_inbound-20110206.gz
-rw-r--r--  1 syslog syslog   2072 Feb 13 23:58 mail_inbound-20110213.gz
-rw-r--r--  1 syslog syslog   2073 Feb 20 23:58 mail_inbound-20110220.gz
-rw-r--r--  1 syslog syslog  15236 Feb  6 23:58 mail_outbound-20110206.gz
-rw-r--r--  1 syslog syslog  15139 Feb 13 23:58 mail_outbound-20110213.gz
-rw-r--r--  1 syslog syslog  14754 Feb 20 23:58 mail_outbound-20110220.gz
-rw-r--r--  1 root   syslog 330117 Feb  6 23:58 mysql_err-20110206.gz
-rw-r--r--  1 root   syslog 101416 Feb 13 23:58 mysql_err-20110213.gz
-rw-r--r--  1 root   syslog  39369 Feb 20 23:58 mysql_err-20110220.gz
-rw-r--r--  1 syslog syslog    751 Feb  6 23:58 pdu-20110206.gz
-rw-r--r--  1 syslog syslog   1014 Feb 13 23:58 pdu-20110213.gz
-rw-r--r--  1 syslog syslog    699 Feb 20 23:58 pdu-20110220.gz
-rw-r--r--  1 syslog syslog  46940 Feb  6 23:58 unix-20110206.gz
-rw-r--r--  1 syslog syslog  28980 Feb 13 23:58 unix-20110213.gz
-rw-r--r--  1 syslog syslog  56792 Feb 20 23:58 unix-20110220.gz
```

Contents of `/mnt0/syslog/archive/sec/`

Support Tools - UNIX Text Processing & Pipelines

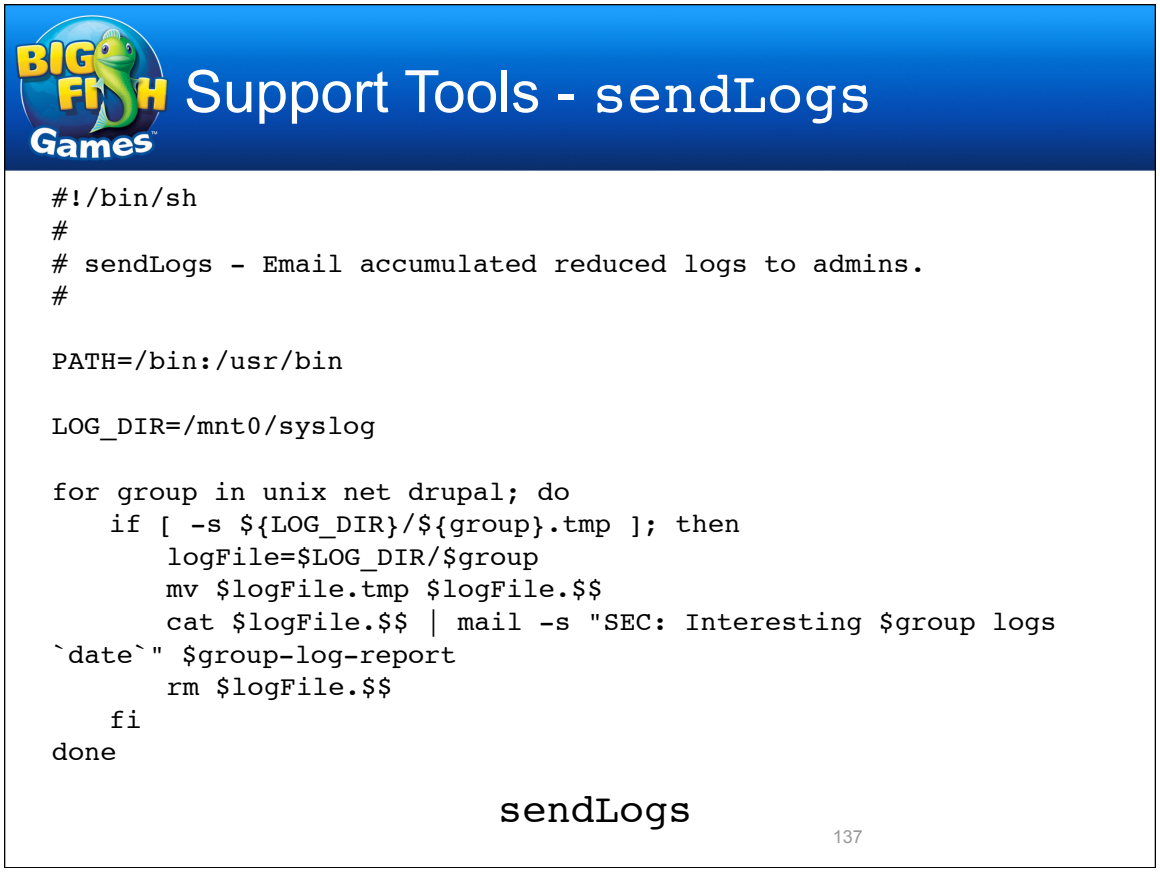
- →You'll often need to dive into the logs to follow up on an issue that SEC shows you
- →Get used to using `zcat`, `zgrep`, `grep`, `cut`, `awk`, `sort`, `uniq`, `wc`, `xargs`, and pipelines
  - →They will be your constant companions
    - Unless you have Splunk

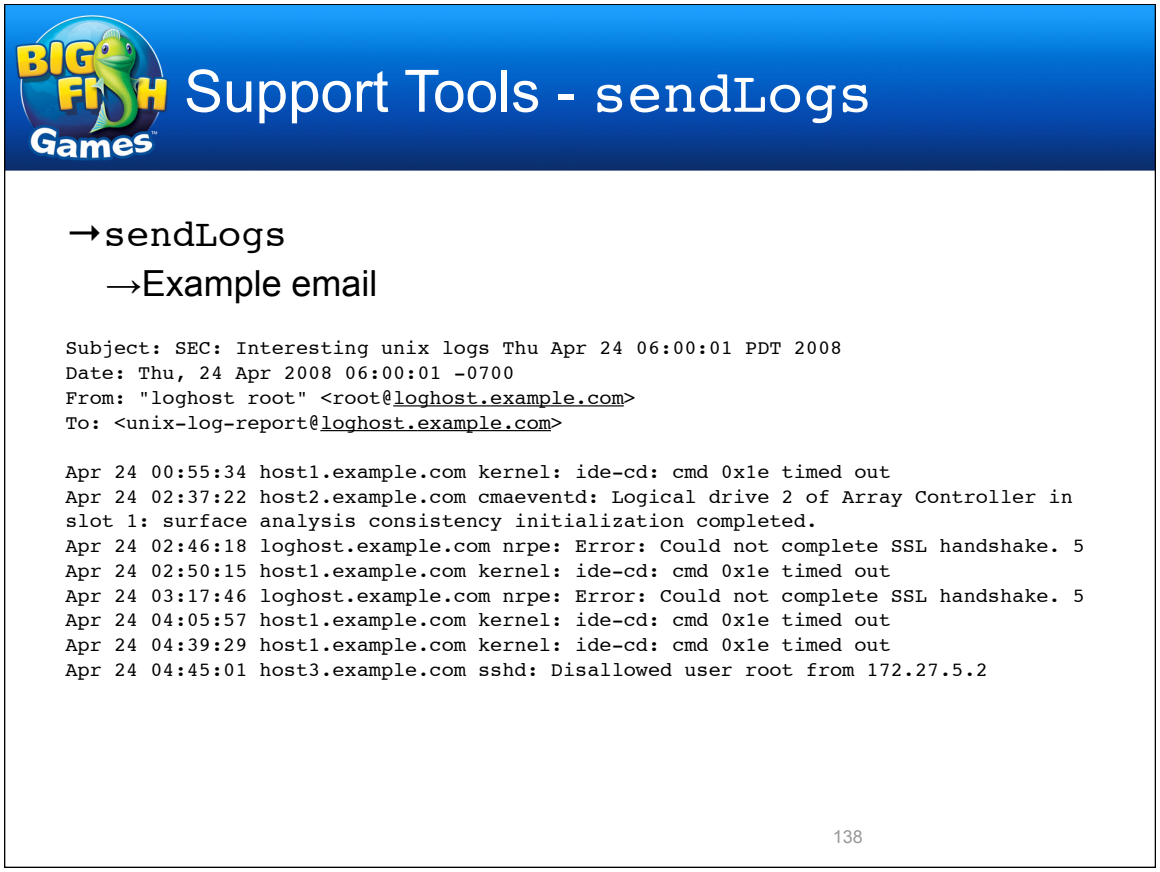
Support Tools - `sendLogs`

- →`sendLogs`
  - →`cron` calls `sendLogs` to issue regular reports of anomalous events

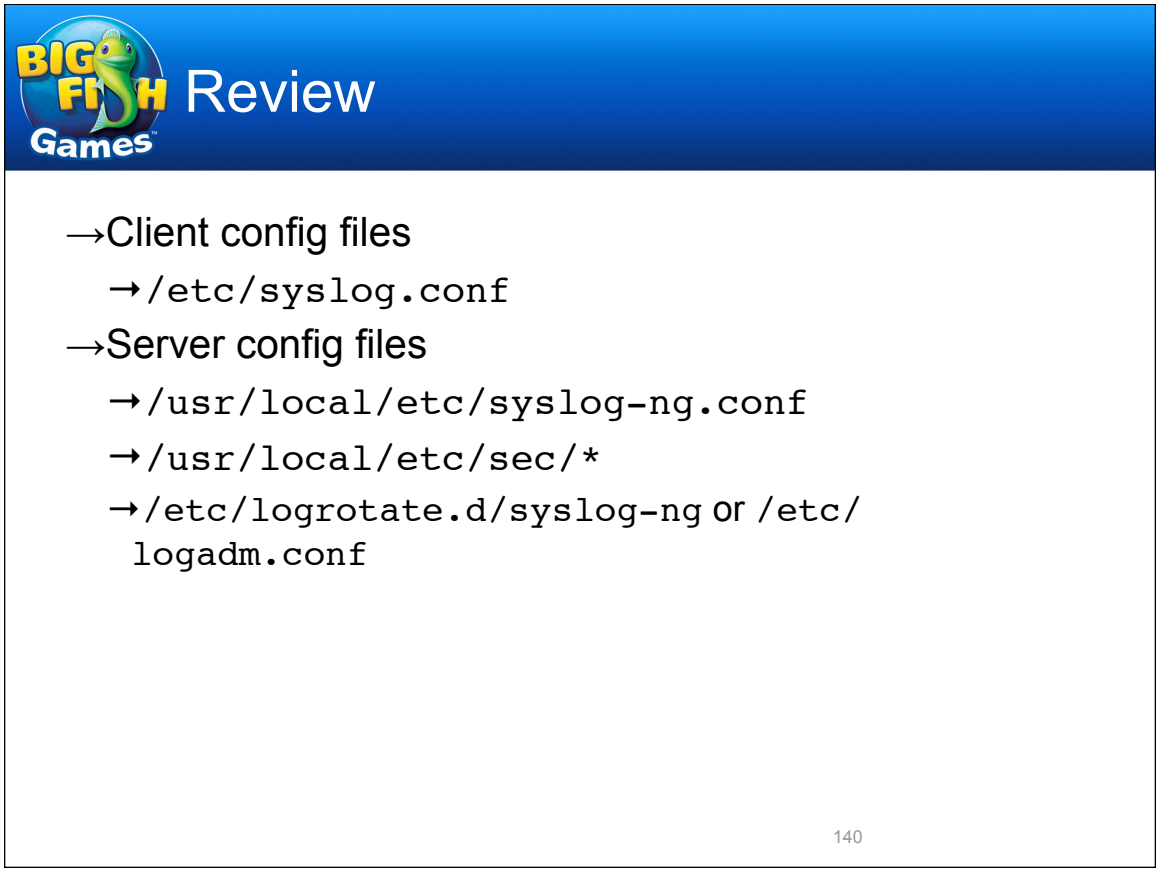
```
0 0,6-18 * * 1-5 /usr/local/bin/sendLogs
0 0,6,12,18 * * 0,6 /usr/local/bin/sendLogs
# Temporary holiday schedule
#0 0,6,12,18 * * * /usr/local/bin/sendLogs
```

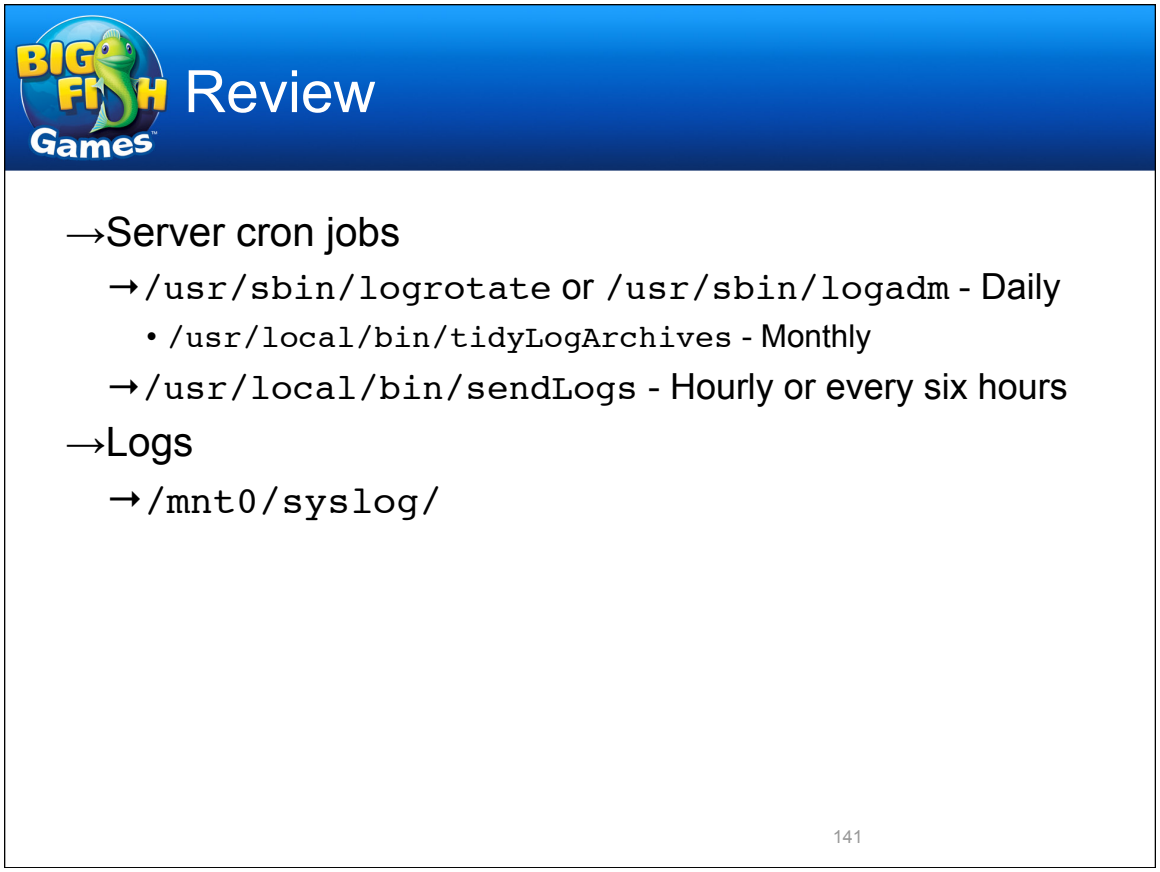
  - →Hourly during work hours (6 AM - 6 PM weekdays), every six hours otherwise

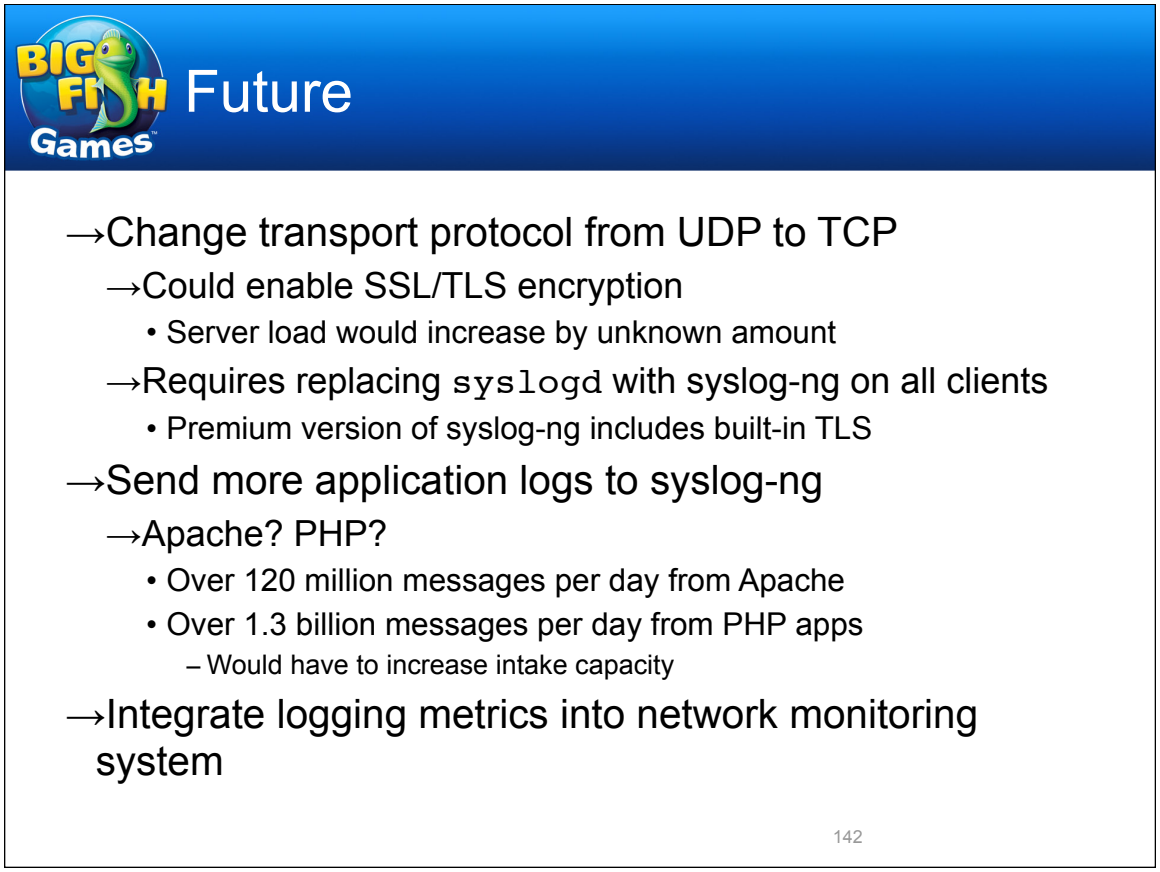

Support Tools - sendLogs
#!/bin/sh
#
sendLogs - Email accumulated reduced logs to admins.
#
PATH=/bin:/usr/bin
LOG_DIR=/mnt0/syslog
for group in unix net drupal; do
if [-s ${LOG_DIR}/${group}.tmp]; then
logFile=$LOG_DIR/$group
mv $logFile.tmp $logFile.$$
cat $logFile.$$ | mail -s "SEC: Interesting $group logs
`date`" $group-log-report
rm $logFile.$$
fi
done
sendLogs
137

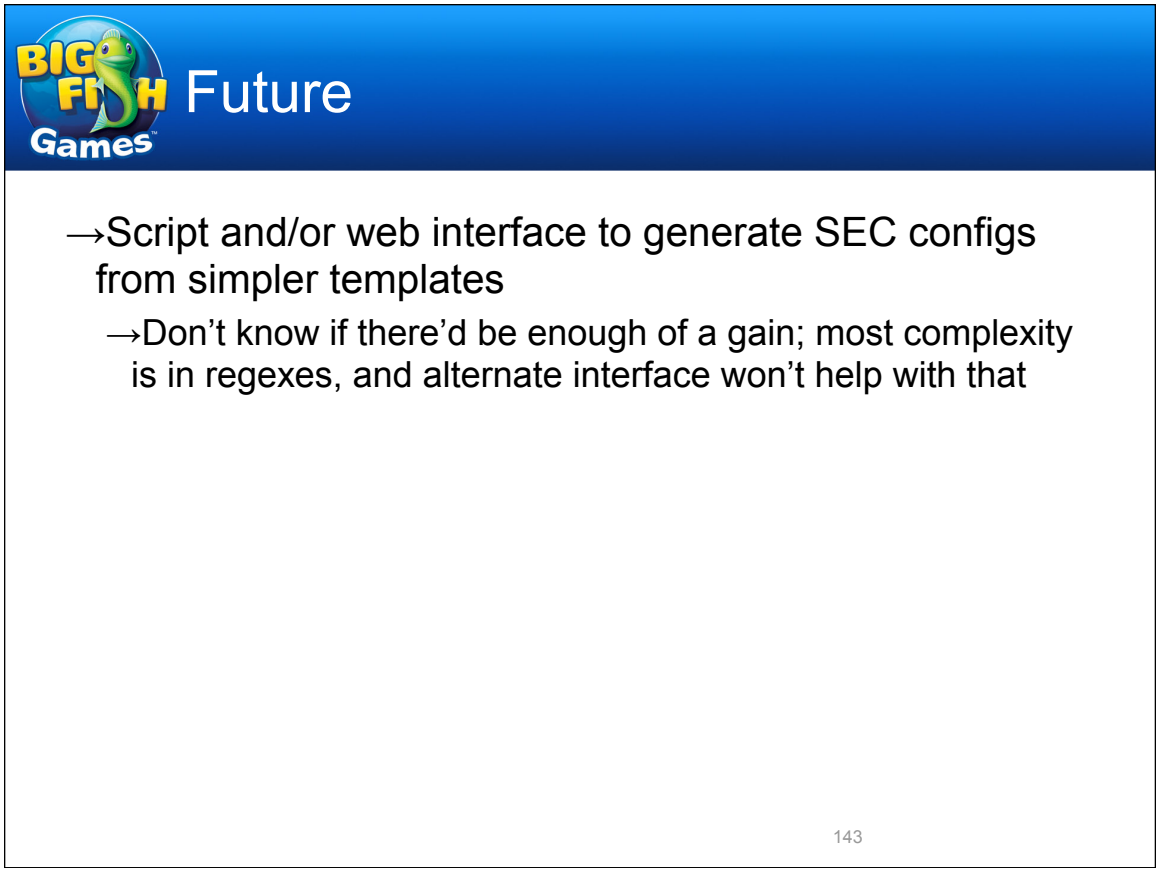

Support Tools - sendLogs
→sendLogs
→Example email
Subject: SEC: Interesting unix logs Thu Apr 24 06:00:01 PDT 2008
Date: Thu, 24 Apr 2008 06:00:01 -0700
From: "loghost root" <root@loghost.example.com>
To: <unix-log-report@loghost.example.com>
Apr 24 00:55:34 host1.example.com kernel: ide-cd: cmd 0x1e timed out
Apr 24 02:37:22 host2.example.com cmaeventd: Logical drive 2 of Array Controller in
slot 1: surface analysis consistency initialization completed.
Apr 24 02:46:18 loghost.example.com nrpe: Error: Could not complete SSL handshake. 5
Apr 24 02:50:15 host1.example.com kernel: ide-cd: cmd 0x1e timed out
Apr 24 03:17:46 loghost.example.com nrpe: Error: Could not complete SSL handshake. 5
Apr 24 04:05:57 host1.example.com kernel: ide-cd: cmd 0x1e timed out
Apr 24 04:39:29 host1.example.com kernel: ide-cd: cmd 0x1e timed out
Apr 24 04:45:01 host3.example.com sshd: Disallowed user root from 172.27.5.2
138


BIG FISH
Games
Review and
Future Activities


BIG FISH
Games
Review
→Client config files
→/etc/syslog.conf
→Server config files
→/usr/local/etc/syslog-ng.conf
→/usr/local/etc/sec/*
→/etc/logrotate.d/syslog-ng or /etc/logadm.conf
140


Review
→Server cron jobs
→/usr/sbin/logrotate or /usr/sbin/logadm - Daily
• /usr/local/bin/tidyLogArchives - Monthly
→/usr/local/bin/sendLogs - Hourly or every six hours
→Logs
→/mnt0/syslog/
141


Future
→Change transport protocol from UDP to TCP
→Could enable SSL/TLS encryption
• Server load would increase by unknown amount
→Requires replacing syslogd with syslog-ng on all clients
• Premium version of syslog-ng includes built-in TLS
→Send more application logs to syslog-ng
→Apache? PHP?
• Over 120 million messages per day from Apache
• Over 1.3 billion messages per day from PHP apps
– Would have to increase intake capacity
→Integrate logging metrics into network monitoring system
142


Future
→Script and/or web interface to generate SEC configs from simpler templates
→Don't know if there'd be enough of a gain; most complexity is in regexes, and alternate interface won't help with that
143


Centralized Logging with syslog-ng and SEC
Distilling Information from Data
Leon Towns-von Stauber
Cascadia IT, March 2012
http://www.occam.com/