How to Handle a Few Hundred Million Extra Firewall Log Lines Per Day

Thanks to the Latest Virus or Worm

Jon Meek

Wyeth - Princeton, NJ

meekj 0x40 wyeth.com

meekj 0x40 ieee.org

High Volume Syslog Traffic

- It has been a while since the last high-volume worm / virus flooded log files
- Other activity can generate high denied traffic
  - Mis-configured systems
  - Poorly designed or implemented protocols
  - A few systems running malware
- We don't need to save every denied packet log line
- The record of allowed traffic is much more important
- A full disk will not be able to save anything

Zotob Worm Appeared in August 2005

- Scanned networks for open TCP port 445
- Little business impact for us
- Considerable IT impact, however
- Resulted in about 400 million additional log lines per day
- We use denied traffic at the firewalls as one method to detect malware activity
- So, we still want that information to locate infected PCs

Solutions, in Chronological Order

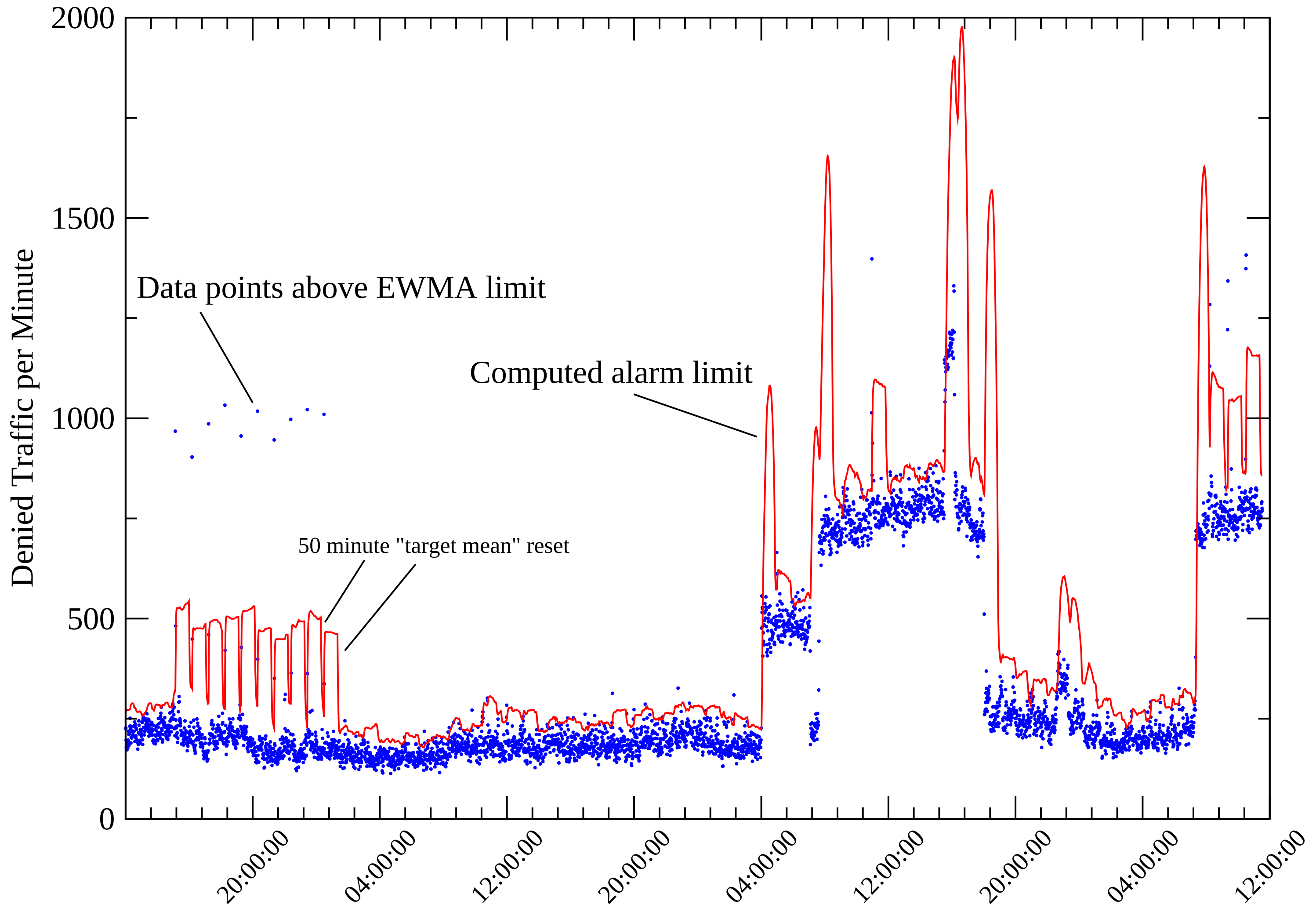
- **Stop logging port 445 traffic**
- **Setup packet capture script**
- **Collect 10,000 packets every five minutes**
- **Summarize source IPs on a Web page for each firewall**
- **Add subnet location to aid remediation effort**
- **Really need the firewall log data for real-time and daily "malware reports"**
- **Log only a 5% random sample of the denied traffic**
  - **Even 5% choked our OSV (outsourced security vendor)**

Longer Term Solution

- Automatic sampling rate filter
- Placed between named pipes in syslog-ng configuration
- Select a target daily total for denied traffic items
    - Usually 8 million lines per day, per Internet connection
- Filter measures per minute rate and scales sampling appropriately
    - Scale factor and rates are logged
- Under normal conditions 100% of denied traffic should be logged
- Can alarm on lower sampling rate, total denied traffic, etc.

Denied Traffic at Firewall

Saturday 16-Apr-2005 to following Tuesday



Typical Implementation

- **Command line:**

```
syslog-sample-auto --maxrate 5555 \
                   --log /usr2/log/denyrate/syslog-sampleYYYYMMDD.log \
                   --in /usr2/log/fractionpipe-in \
                   --out /usr2/log/fractionpipe-out \
                   --runas adetect
```

- **syslog-ng configuration snippets:**

```
# Input from syslog sampling filter, normally used only for denied traffic
#
source s_fractionpipe-out { pipe("/usr2/log/fractionpipe-out"); };

filter f_ns_io1_deny       { host(10.236.114.211) and match("Deny"); };

# IO denied traffic goes here for sampling if needed
#
destination d_ns_io_deny     { pipe("/usr2/log/fractionpipe-in"); };

# Send denied traffic packets through the sampling program via pipe
#
log { source(network); filter(f_ns_io1_deny); destination(d_ns_io_deny); flags(final); };

# Pick up denied traffic packets, send them to the user ID system and log file
#
log { source(s_fractionpipe-out);   destination(d_ns_io_deny2); };

# Then send them to the regular log files
#
log { source(s_fractionpipe-out);   destination(d_ns_io_access); flags(final); };
```

Using the Denied Packets / Minute Data

- Load data into RRDtool [1] for display and monitoring
- rrd_ewma.monitor for mon [2] watches data
- Alarm triggers on:
  - Statistical spike - EWMA control limit [3] is exceeded
  - Absolute threshold is exceeded
  - EWMA (Exponentially Weighted Moving Average) has reset feature to prevent control limit from remaining high for too long

References

[1] RRDtool: http://oss.oetiker.ch/rrdtool/

[2] mon Service Monitoring Daemon:
http://mon.wiki.kernel.org/

[3] EWMA: Introduction to Statistical Quality Control, 3rd Ed, Douglas C. Montgomery.

The Code

```
#!/usr/bin/perl
#
# Send only a sample of syslog data to disk
# to handle high loads!
#
#
# Auto rate adjust version

# Jon Meek
# Lawrenceville, NJ
# meekj  ieee.org
#
#    Copyright (C) 2007, Jon Meek
#
#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 2 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program; if not, write to the Free Software
#    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#

=head1 NAME

B<syslog-sample-auto> - Random sampling of syslog data with automatic rate
                        adjustments to handle high loads
```

```
=head1 DESCRIPTION

B<syslog-sample-auto> is typically deployed as a rate limiting filter
between two named pipes used by syslog-ng.

For each minute, or other specified interval, the program computes the
current syslog rate and makes an adjustment if a pre-set rate is
exceeded. The sample percentage is adjusted to keep the number of
messages reaching the log at, or below, the preset rate. The sample
percentage has only 2 digit precision and no adjustment is made unless
the change is 10% or more to prevent too many changes. Each minute the
current sample percentage, total message count, and sampled message
count are logged.

=head1 SYNOPSIS

B<syslog-sample-auto> --in in_pipe --out out_pipe --maxrate 5555 --runas adetect \
                      --interval 60 --log /usr2/logs/denyYYYYMMDD.log


=head1 OPTIONS

=over 5

=item B<--in>

Input, usually a named pipe.

=item B<--out>

Output, usually a named pipe.


=item B<--maxrate>

Maximum per interval rate (usually 60 second intervals).

 Log items per day  Per minute rate
```

```
      5 million       3472
      8 million       5555
     10 million       6944
     15 million      10416
     20 million      13888


=item B<--runas> username

Run as a designated user rather than root to enhance security.

=item B<--interval>

Seconds between rate checks and log entries. Defaults to 60 seconds.

=item B<--log log_file_template> /path/to/logs/denyYYYYMMDD.log

Current year, month, and day are substituted for YYYYMMDD, that is the
only possible template at this time.

=back


=head1 LOG FILE FORMAT

If the log option is specified, the following parameters are logged:
time in UNIX seconds, the scaling factor, the number of log lines seen
by the filter during the time period, and the number of lines passed
on to the syslog program.

A sample:

 1172016366 1.0e+00 5498 5498
 1172016426 1.0e+00 5301 5301
 1172016486 1.0e+00 5131 5131
 1172016546 1.0e+00 5704 5704
 1172016606 1.0e+00 5430 5430
 1172016666 8.3e-01 6673 6673
 1172016726 8.3e-01 7055 5859
 1172016786 8.3e-01 7107 5919
 1172016846 8.3e-01 6944 5782
 1172016906 7.4e-01 7477 6177
 1172016966 7.4e-01 7389 5490
 1172017026 7.4e-01 8311 6178
 1172017086 6.6e-01 8417 6250
 1172017146 6.6e-01 9064 5994


=head1 BUGS

=head1 AUTHOR

Jon Meek, meekj ieee.org

$Id: syslog-sample-auto,v 1.4 2007/11/07 04:34:25 meekj Exp $

=cut

use Getopt::Long;
use POSIX;
use IO::Handle;

my $TotalCount = 0;
my $PassedCount = 0;
my $SampleFraction = 1.0;       # Initially pass everything by default
my $MinimumFractionChange = 10; # Minimum change in percent difference from current value

GetOptions(
     "in=s" => \$INPIPE,
     "out=s" => \$OUTPIPE,
     "maxrate=f" => \$MaxRate,       # Items per minute
     "runas=s" => \$RunAsUser,       # Start as root, then change to specified user
     "interval=i" => \$IntervalTime, # Seconds between rate checks and logging, usually 60
     "log=s" => \$LogFile,           # If --runas is specified be sure directory is
                                     # writable by $RunAsUser
);


$IntervalTime = 60 unless $IntervalTime; # Seconds between rate checks and logging, usually 60
```

```
die "Input not readable: $INPIPE" if (! -r $INPIPE);
die "Output not writeable: $OUTPIPE" if (! -w $OUTPIPE);


# SIGPIPEs will occur if we write to a closed filehandle, it might happen during log rotation
#
$SIG{PIPE} = 'IGNORE';

if ($RunAsUser) {
  $uid = getpwnam($RunAsUser) or die "Can't get uid for $RunAsUser0;
  $> = $uid; # Change the effective UID
}

#
# Run as a deamon, fork a new process then exit this copy
#
$pid = fork;
exit if $pid;
die "Couldn't fork: $!" unless defined($pid);

#
# We are now running as a daemon, close the standard file handles
#
for my $handle (*STDIN, *STDOUT, *STDERR) {
  open($handle, "+<", "/dev/null") || die "Can't reopen $handle to /dev/null: $!";
}

#
# Dissociate from the controlling terminal
#
POSIX::setsid() or die "Can't start a new session: $!";

if ($RunAsUser) {
  $< = $>; # Set the real UID to the effective UID
}

$SIG{ALRM} = CatchAlarm;

alarm($IntervalTime);
```

```
while (1) {         # Loop to handle EOFs on input

 die "Input not readable: $INPIPE" if (! -r $INPIPE); # Be sure the pipes are there and ready
 die "Output not writeable: $OUTPIPE" if (! -w $OUTPIPE);

 open(IN, $INPIPE);      # Checking the status when opening a pipe fails, it appears
 open(OUT, ">$OUTPIPE"); # however, we checked the availability above
 OUT->autoflush(1);

  while ($in = <IN>) {
    $TotalCount++;
    $random = rand(1);               # Generate random number between 0.0 and 1.0
    if ($random < $SampleFraction) { # Should this item be passed?
      $PassedCount++;
      print OUT $in;
    }
  }
  close IN;
  close OUT;

}

#
# When the timer expires process data for current interval
#
sub CatchAlarm {
  my ($new_frac, $percent_change, $LOG, $DailyLogFile);
  my ($sec, $min, $hour, $mday, $month, $year, $wday, $yday, $isdst);
  my $TimeNow = time;
  my $rate = 60 * ($TotalCount / $IntervalTime); # Per minute rate

  #
  # Adjust rate to 2 digits
  #

  if ($rate == 0) {
    $new_frac = 1.0;
  } else {
    $new_frac = $MaxRate / $rate;
    $new_frac = 1.0 if ($new_frac > 1.0);
    $new_frac = sprintf("%.1e", $new_frac); # Two digit version,
                                            # we want a small set of possible values
                                            # + the scale factor
  }

  #
  # Check for appropriate change
  #

  $percent_change = 100 * abs($SampleFraction - $new_frac) / $SampleFraction;

  if ($percent_change >= $MinimumFractionChange) {
    $SampleFraction = $new_frac;
  }

  if ($LogFile) {
    ($sec, $min, $hour, $mday, $month, $year, $wday, $yday, $isdst) =
      localtime($TimeNow);
    $month++; $year += 1900;
    $YYYYMMDD = sprintf('%04d%02d%02d', $year, $month, $mday);
    $DailyLogFile = $LogFile;
    $DailyLogFile =~ s/YYYYMMDD/$YYYYMMDD/; # Fill in current year, month, day
    # Where will a warning go?
    open($LOG, ">>$DailyLogFile") || warn "$0 Can't open logfile: $LogFile0;
#   LOG->autoflush(1);
    print $LOG "$TimeNow $SampleFraction $TotalCount $PassedCount0;
    close $LOG;
  }

  $TotalCount = 0;
  $PassedCount = 0;

  alarm($IntervalTime); # Re-enable alarm for next cycle

}
```