Anomaly detection with cfenvd

Edition 2.2.1 for version 2.2.1

Mark Burgess
Faculty of Engineering, Oslo University College, Norway

This manual corresponds to CFENGINE Edition 2.2.1 for version 2.2.1 as last updated 30 October 2006.

1 Overview

One of the aims of cfengine version 2, in combination with our research efforts at Oslo University College, is to develop a real computer immune system, based on the detection of 'sicknesses' or anomalies in the behaviour of the system, not merely based on a policy template. Such a system would be far more dynamical and be able to change in response to changing external conditions. This work is in its early stages, but you can take advantage of it straight away, with a minimum of effort. This additional manual aims at getting you started, so that you can monitor systems, and learn about their behaviour, without having to watch control panels, or time-series traces.

1.1 Intrusion detection

What is an intrusion or an attempted intrusion? This can be difficult to define. If someone tries to login at root once? If someone tries to login at root fifty times? Port scanning, SATAN or ISS scan? Someone trying a known security hole?

The aim of an intrusion detection system is to detect break-ins in progress so that something can be done about them. Obviously the first thing one should worry about is how difficult it is to break in in the first place. If we have done the job of securing data well enough, why are we worried that anyone will be able to get in?

Intrusion detection is a form of fault-diagnosis. Faults (in a security system) are not supposed to happen, but the fact is that they do happen. As with all fault diagnosis systems, IDS give the wrong answers from time to time. Because it is so difficult to define what intrusion actually means in a generic sense (it's political) intrusion detection systems tend to err on the side of caution and report many false positives, i.e. false alarms.

This is a very difficult problem to do in real time. What does real-time mean? Some attacks are stealthy and occur over many hours or days. How can we make a prompt notification about such attempts? The intrusion detection will have to be fast to detect quick break-ins, but have a long memory in order to see slow ones (like the thief digging a tunnel into the bank with a tea-spoon).

How will we be alerted or notified about intrusions? By alarm on the screen? By E-mail or pager alert? What if the attacker first knocks out E-mail or the pager link?

User privacy is also a problem. If an intrusion detection system examines everything going on within the system, looking for suspicious behaviour, is that an intrusion of privacy? What if humans never see the data, but only the warnings? Where do we draw the line between justified and unjustified surveillance? Law enforcement agencies have been arguing about that one for years!

1.2 Cryptographic checksums

Change detection is about monitoring whether files and other aspects of a system change. The idea was originally advanced in the program Tripwire, which collected a "snaphot" of the system in the form of a database of file checksums (hashes) and permissions and rechecked the system against this database at regular intervals. Tripwire examines only files, and looks for any change at all. If a legitimate change is made to the system, it responds to this as a potential threat. Databases must then be altered, or rebuilt.

Cfengine adopted part of Tripwire's idea. It collects MD5 hash data for specified files. Its model for checking permissions is somewhat different however. Cfengine expects systems to change dynamically, so it allows users to define a policy for allowed change. It can also check for processes, not merely files. Integrity checks on files whose contents are supposed to be static are a good way to detect tampering with the system, from whatever source. Running MD5 or SHA1 checksums of files regularly provides us with a way of determining even the smallest changes to file contents. Here is an excerpt from a cfengine configuration program that would check the /usr/local filesystem for file changes. Note that it excludes files such as log files that are supposed to change:

```
control:

actionsequence = ( files )

####################################################################

files:

  /usr/local o=root,bin,man action=warnall mode=o-w r=inf checksum=md5
                    ignore=logs exclude=*.log
```

1.3 An environment detector: cfenvd

The '`cfenvd`' program serves two purposes: as an anomaly detection engine and as a source of 'entropy' for generating random numbers, such as for encryption keys. Although it is not a compulsory part of cfengine, it is highly recommended to run this daemon. It requires few resources and poses no vulnerability to the system. It will play an increasingly important role in future developments.

In cfengine 2.x, additional classes are automatically evaluated based on the state of the host, in relation to earlier times. This is accomplished by the additional '`cfenvd`' daemon, which continually updates a database of system averages and variances, which characterize "normal" behaviour. The state of the system is examined and compared to the database, and the state is classified in terms of the current level of activity, as compared to an average of equivalent earlier times. e.g.

```
RootProcs_low_dev2
netbiosssn_in_low_dev2
smtp_out_high_anomalous
www_in_high_dev3
ftp_in_high_microanomaly
```

The first of these tells us that the number of root processes is two standard deviations below the average of past behaviour, which might be fortuitous, or might signify a problem, such as a crashed server. The WWW item tells us that the number of incoming connections is three standard deviations above average. The smtp item tells us that outgoing smtp connections are more than three standard deviations above average, perhaps signifying a mail flood. The setting of these classes is transparent to the user, but the additional information is only visible to the privileged owner of the cfengine work-directory, where the data are cached.

The term 'microanomaly' is used to describe two standard deviations above normal, when the delta of the change is less than the arbitray value of 5. This is a small number, and anomalies of these kinds are generally noise.

Simply specifying statistical number anomalies is not sufficient to provide well-honed anomaly characteristics. Cfengine tries to organize the information surrounding an anomaly first in terms of statistical significance and then only later in terms of event characteristics. There are too many events in which the numerical values exceed thresholds determined by an arbitrary policy. Other criteria are needed to pin down which anomalies are interesting and which are not. As a second level of policy filtering, cfengine provides a measure of the entropy of the source IP addresses of the measured data. A low entropy value means that most of the events came from only a few (or one) IP addresses. A high entropy value implies that the events were spread over many IP sources. These conditions are described by classes of the form:

```
entropy_www_in_high
entropy_smtp_in_low
```

Thus, for example, in the first case the class will be set if incoming traffic at the peak event of the last data sample was spread evenly over all the incoming addresses. Such an event indicates that the resource usage is not due to a single source (e.g. an attacker from a single location) but is evenly spread — perhaps just a coincidental anomaly. In the second case, the low entropy smtp traffic must come from one or two addresses and is more likely to be spam or an attack of some kind. These classes can be combined with the specific anomaly thresholds (see example below).

Active incoming ports are also registered as “pin-portnumber”, but this is mainly an experimental feature for future research. The resulting class list, obtained from exploring the environment of the system, and after parsing a configuration, looks something like this:

```
host% cfagent -p -v

[snip]

Defined Classes = ( any Thursday Hr14 Min24 Min20_25
Day19 July Yr2001 solaris examplehost 32_bit sunos_5_7
sunos_sun4u sunos_sun4u_5_7 sparc solaris2_7 129_0_0
129_0_0_10 loghost OnTheHour peaktime DayTime
examplehost_example_org longjob Setup_SSH_OK y MailHub
percent_60 RootProcs_normal_dev2 nfsd_out_low_dev2
pin-1554 pin-80 pin-21 pin-6011 pin-5308 pin-139
pin-983 pin-10 )

[snip]
```

It is not yet known how the extra environment classes will be used in practice. One obvious possibility is to limit certain heavy-weight operations (such as file tree scans) when the host is very busy, and to increase the probability of their occurrence when the host is lightly loaded. See the example in section 11. It remains to be seen how users will respond to these possibilities.

1.4 Anomaly research

There is no system available in the world today which can claim to detect and classify the functioning state of a computer system. Cfengine does not attempt to provide a “product” solution to this problem; rather it incorporates a framework, based on the current state of knowledge, for continuing research into this issue. In version 2.x of cfengine, an extra daemon ‘`cfenvd`’ is used to collect statistical data about the recent history of each host (approximately the past two months), and classify it in a way that can be utilized by the cfengine agent.

The daemon may simply be started, with no arguments:

```
cfenvd
```

and it proceeds to collect data and work autonomously, without further supervision. The cf-environment daemon is meant be trivial to use. The current long-term data recorded by the daemon are: number of users, number of root processes, number of non-root processes, percentage disk full for root disk, number of incoming and outgoing sockets for netbiosns, netbiosdgm, netbiosssn, irc, cfengine, nfsd, smtp, www, ftp, ssh and telnet. These data have been studied previously, and their behaviour is relatively well understood. In future versions, it is expected to extend this repertoire, as more research is done.

The use of the daemon will not be reliable until about six to eight weeks after installing and running it, since a suitable training period is required to build up enough data for stable characterization. The daemon automatically adapts to the changing conditions, but has a built-in inertia which prevents anomalous signals from being given too much credence. Persistent changes will gradually change the 'normal state' of the host over an interval of a few weeks. Unlike some systems, cfengine's training period never ends. It regards normal behaviour as a relative concept, which has more to do with local stability than global constancy.

The final size of the database is approximately 2MB. Measurements are taken every five minutes (approximately). This interval is based on auto-correlation times measured for networked hosts in practice.

Cfenvd sets a number of classes in cfengine which describe the current state of the host in relation to its recent history. The classes describe whether a parameter is above or below its average value, and how far from the average the current value is, in units of the standard-deviation (see above). This information could be utilized to arrange for particularly resource-intensive maintenance to be delayed until the expected activity was low.

1.5 cfenvgraph

The '`cfenvgraph`' command can use used to dump a graph of averages for visual inspection of the normal state database. The format of the file is

```
t,y_1,y_2,y_3...
```

which can be viewed using '`gnuplot`' or '`xgmr`' or other graphical plotting program. This would allow the policy-maker to see what is likely to be a good time for such work (say 06:00 hours), and then use this time for the job, unless an anomalous load is detected.

The cfenvgraph command is used to extract data from the database used by the cfenvd environment daemon.

```
cfenvgraph -f filename.db [-r -T -t -s -e]
```

The command normally generates two files with format

```
t, y_1, y_2, y_3, y_4...
```

in a sub-directory of the current directory '`cfenvgraphs-snapshot`' (or '`cfenvgraphs-`'*TIMESTAMP* if '`-T`' is used).

The files are called

```
cfenv-average
cfenv-stddev
```

and contain, respectively, the weighted average values of all the recorded data and the square-root of the weighted variances with respect to the averages. Data are weighted in such a way that older values are gradually deprecated, becoming irrelevant after about two months.

Normally the vertical scale of each graph is scaled so that each line has a maximum value of 1 and a minimum value of 0, this allows all the lines to be seen in maximum detail. However, this makes it difficult to see the absolute values of the lines. With the '-n' option, no scaling is performed and true values are plotted.

The complete data span a one-week period, and the daily rhythm of the system may normally be viewed as a number of peaks, one per day.

The options are:

'--help (-h)'
: List command options

'--file (-f)'
: Specifiy file to plot.

'--titles (-t)'
: If the '-t' option is given, comments are generated at the start of the file which describe the columns. These are in a format understood by 'vvgraph' as title/label data.

'--timestamps (-T)'
: If the '-T' option is given, the output filenames are time-stamped with the current time, in order to give a unique name.

'--resolution (-r)'
: If the '-r' option is given then high resolution data are generated (five minute resolution), otherwise data are averaged over periods of one hour to generate simpler and smoother graphs.

'--separate (-s)'
: If the '-s' option is given, cfenvgraph generates separate files for each metric, in the format

```
t,y,dy
```

where dy is the height of a vertical error-bar. This set of graphs combines the average with the standard-deviation. (Note that the error bars show the standard-deviation, and not the standard error of the mean i.e. stddev/sqrt(N)); the latter has no obvious meaning here. If '-e' is specified, then error bars are omitted.

'--no-error-bars (-e)'
: No error bars are plotted.

'--no-scaling (-n)'
: The graphs are not scaled, so that (min,max) is mapped onto the interval (0,1).

Note that the values printed for sockets always look higher than they should for highly active services. This is because even those sockets which are in CLOSE_WAIT are counted.

This is the correct way to determine a normal state based on the recent past. It is a local averaging performed by the kernel. If one counts only those connections which are currently active, one gets a distorted view of activity with a 5-minute sample rate. To measure more often than this would place unacceptably high load on the system.

Graphs may be viewed in '`vvgraph`', '`xmgr`' (used in the pictures above) or '`gnuplot`', or other graphical viewer. These graphs are not meant for continuous viewing. The data are averages, not time-series.

For example, with gnuplot

```
host$ cfenvgraph -s
host$ gnuplot
gnuplot> plot "www-in.cfenv" with errorbars
gnuplot> plot "www-in.cfenv" with lines
```

The new version of `xmgr` is called `xmgrace`. It can be invoked as follows:

```
host$ xmgrace -nxy cfenv-averages
host$ xmgrace  rootprocs.cfenv
host$ xmgrace -settype xydy  rootprocs.cfenv
host$ xmgrace -settype xydy  rootprocs.cfenv -hardcopy -hdevice JPEG
```

If you see the error "Strings are not allowed", it might be because some "nan" values have come into the text file.

1.6 Fluctuation profiles

Any model of fluctuating values is based on the idea that the changing signal has a basic separation of signal and noise. The variability of the signal is generally characterized by a probability distribution. Some tools and many papers assume that the distribution of fluctuations is Gaussian. This is almost never the case in real computer systems.

Hurst exponent measures the degree of stability in the signal, in the following sense....

1.7 Starting with anomaly detection

Try importing the following file:

```
###
#
# BEGIN cf.environ
#
###

#
# Just a test for responses to measured anomalies
#

classes:

 anomaly_hosts = ( myhost1 myhost2 )


#################################################

alerts:

  nfsd_in_high_dev2::
```

```
  "High NFS server access rate 2dev at $(host)/$(env_time)
current value $(value_nfsd_in) av $(average_nfsd_in) pm
$(stddev_nfsd_in)"

  ShowState(incoming.nfs)


 # ROOT PROCS

  anomaly_hosts.RootProcs_high_dev2::

   "RootProc anomaly high 2 dev on $(host)/$(env_time) current value
$(value_rootprocs) av $(average_rootprocs) pm $(stddev_rootprocs)"
  ShowState(procs)



 # USER PROCS

 anomaly_hosts.UserProcs_high_dev2::

   "UserProc anomaly high 2 dev on $(host)/$(env_time) current
value $(value_userprocs) av $(average_userprocs) pm $(stddev_userprocs)"
   ShowState(procs)


 anomaly_hosts.UserProcs_high_anomaly::

   "UserProc anomaly high 3 dev!! on $(host)/$(env_time)"
   ShowState(procs)


 # WWW IN

 # This happens too often
 # anomaly_hosts.www_in_high_dev2::
 #

 entropy_www_in_high.anomaly_hosts.www_in_high_anomaly::

   "HIGH ENTROPY Incoming www anomaly high anomaly dev!!
on $(host)/$(env_time) - current value $(value_www_in)
av $(average_www_in) pm $(stddev_www_in)"

   ShowState(incoming.www)

 entropy_www_in_low.anomaly_hosts.www_in_high_anomaly::

   "LOW ENTROPY Incoming www anomaly high anomaly dev!! on
$(host)/$(env_time) - current value $(value_www_in) av
$(average_www_in) pm $(stddev_www_in)"

   ShowState(incoming.www)



 # SMTP IN
```

```
 entropy_smtp_in_high.anomaly_hosts.smtp_in_high_dev2::

   "HIGH ENTROPY Incoming smtp anomaly high 2 dev on $(host)/$(env_time)"

 entropy_smtp_in_high.anomaly_hosts.smtp_in_high_anomaly::

   "HIGH ENTROPY Incoming smtp anomaly high anomaly !! on $(host)/$(env_time)"


 entropy_smtp_in_low.anomaly_hosts.smtp_in_high_dev1::

   "LOW ENTROPY Incoming smtp anomaly high 1 dev on $(host)/$(env_time)
 current value $(value_smtp_in) av  $(average_smtp_in) pm $(stddev_smtp_in)"

   ShowState(incoming.smtp)


 entropy_smtp_in_low.anomaly_hosts.smtp_in_high_dev2::

   "LOW ENTROPY Incoming smtp anomaly high 2 dev on $(host)/$(env_time)
current value $(value_smtp_in) av  $(average_smtp_in) pm $(stddev_smtp_in)"

   ShowState(incoming.smtp)


 entropy_smtp_in_low.anomaly_hosts.smtp_in_high_anomaly::

   "LOW ENTROPY Incoming smtp anomaly high anomaly !! on $(host)/$(env_time)
current value $(value_smtp_in) av  $(average_smtp_in) pm $(stddev_smtp_in)"

   ShowState(incoming.smtp)


 # SMTP OUT

 anomaly_hosts.smtp_out_high_dev2::

   "Outgoing smtp anomaly high 2 dev on $(host)/$(env_time) current value
$(value_smtp_out) av  $(average_smtp_out) pm $(stddev_smtp_out)"

   ShowState(outgoing.smtp)

 anomaly_hosts.smtp_out_high_anomaly::

   "Outgoing smtp anomaly high anomaly dev!! on $(host)/$(env_time)
current value $(value_smtp_out) av $(average_smtp_out) pm $(stddev_smtp_out)"

   ShowState(outgoing.smtp)


 # SAMBA

 anomaly_hosts.netbiosssn_in_high_dev2::

   "SAMBA access high 2 on $(host)/$(env_time) current value
$(value_netbiosssn_in) av $(average_netbiosssn_in) pm $(stddev_netbiosssn_in)"
```

```
   ShowState(incoming.netbiosssn)

################################################################################

###
#
# END cf.environ
#
###
```

The output generated by this file shows the current value of the quantity and a summary of the highest values during the last 40 minutes:

```
cf:cube: LOW ENTROPY Incoming www anomaly high anomaly dev!! on cube/Fri Feb 20 19:57:23 2004 -
current value 53 av 9.9 pm 16.1
cf:cube: ---------------------------------------------------------------------------
------
cf:cube: In the last 40 minutes, the peak state was:
cf:cube: ( 1) tcp        0      0 128.39.74.16:80         157.158.24.40:4049      TIME_WAIT
cf:cube: ( 2) tcp        0      0 128.39.74.16:80         157.158.24.40:3796      TIME_WAIT
cf:cube: ( 3) tcp        0      0 128.39.74.16:80         157.158.24.40:3544      TIME_WAIT
cf:cube: ( 4) tcp        0      0 128.39.74.16:80         157.158.24.40:4063      TIME_WAIT
cf:cube: ( 5) tcp        0      0 128.39.74.16:80         157.158.24.40:4035      TIME_WAIT
cf:cube: ( 6) tcp        0      0 128.39.74.16:80         157.158.24.40:3782      TIME_WAIT
cf:cube: ( 7) tcp        0      0 128.39.74.16:80         157.158.24.40:3530      TIME_WAIT
cf:cube: ( 8) tcp        0      0 128.39.74.16:80         157.158.24.40:3824      TIME_WAIT
cf:cube: ( 9) tcp        0      0 128.39.74.16:80         157.158.24.40:3572      TIME_WAIT
cf:cube: (10) tcp        0      0 128.39.74.16:80         157.158.24.40:4091      TIME_WAIT
cf:cube: (11) tcp        0      0 128.39.74.16:80         157.158.24.40:3839      TIME_WAIT
cf:cube: (12) tcp        0      0 128.39.74.16:80         157.158.24.40:3810      TIME_WAIT
cf:cube: (13) tcp        0      0 128.39.74.16:80         157.158.24.40:4077      TIME_WAIT
cf:cube: (14) tcp        0      0 128.39.74.16:80         157.158.24.40:3993      TIME_WAIT
cf:cube: (15) tcp        0      0 128.39.74.16:80         157.158.24.40:3740      TIME_WAIT
cf:cube: (16) tcp        0      0 128.39.74.16:80         157.158.24.40:3712      TIME_WAIT
cf:cube: (17) tcp        0      0 128.39.74.16:80         157.158.24.40:3979      TIME_WAIT
cf:cube: (18) tcp        0      0 128.39.74.16:80         157.158.24.40:3726      TIME_WAIT
cf:cube: (19) tcp        0      0 128.39.74.16:80         157.158.24.40:4021      TIME_WAIT
cf:cube: (20) tcp        0      0 128.39.74.16:80         157.158.24.40:3768      TIME_WAIT
cf:cube: (21) tcp        0      0 128.39.74.16:80         157.158.24.40:3516      TIME_WAIT
cf:cube: (22) tcp        0      0 128.39.74.16:80         80.203.17.11:11487      ESTABLISHED
cf:cube: (23) tcp        0      0 128.39.74.16:80         157.158.24.40:4007      TIME_WAIT
cf:cube: (24) tcp        0      0 128.39.74.16:80         157.158.24.40:3754      TIME_WAIT
cf:cube: (25) tcp        0      0 128.39.74.16:80         66.196.72.28:6545       TIME_WAIT
cf:cube: (26) tcp        0      0 128.39.74.16:80         157.158.24.40:3923      TIME_WAIT
cf:cube: (27) tcp        0      0 128.39.74.16:80         157.158.24.40:3670      TIME_WAIT
cf:cube: (28) tcp        0      0 128.39.74.16:80         80.202.77.107:1567      TIME_WAIT
cf:cube: (29) tcp        0      0 128.39.74.16:80         157.158.24.40:4189      TIME_WAIT
cf:cube: (30) tcp        0      0 128.39.74.16:80         157.158.24.40:3909      TIME_WAIT
cf:cube: (31) tcp        0      0 128.39.74.16:80         157.158.24.40:3656      TIME_WAIT
cf:cube: (32) tcp        0      0 128.39.74.16:80         157.158.24.40:3698      TIME_WAIT
cf:cube: (33) tcp        0      0 128.39.74.16:80         157.158.24.40:3965      TIME_WAIT
cf:cube: (34) tcp        0      0 128.39.74.16:80         80.202.77.107:1568      TIME_WAIT
cf:cube: (35) tcp        0      0 128.39.74.16:80         157.158.24.40:3937      TIME_WAIT
cf:cube: (36) tcp        0      0 128.39.74.16:80         157.158.24.40:3684      TIME_WAIT
cf:cube: (37) tcp        0      0 128.39.74.16:80         157.158.24.40:4203      TIME_WAIT
cf:cube: (38) tcp        0      0 128.39.74.16:80         157.158.24.40:3951      TIME_WAIT
cf:cube: (39) tcp        0      0 128.39.74.16:80         157.158.24.40:3600      TIME_WAIT
```

```
cf:cube: (40) tcp        0      0 128.39.74.16:80         157.158.24.40:4119      TIME_WAIT
cf:cube: (41) tcp        0      0 128.39.74.16:80         157.158.24.40:3867      TIME_WAIT
cf:cube: (42) tcp        0      0 128.39.74.16:80         157.158.24.40:3614      TIME_WAIT
cf:cube: (43) tcp        0      0 128.39.74.16:80         157.158.24.40:3586      TIME_WAIT
cf:cube: (44) tcp        0      0 128.39.74.16:80         157.158.24.40:4105      TIME_WAIT
cf:cube: (45) tcp        0      0 128.39.74.16:80         157.158.24.40:3853      TIME_WAIT
cf:cube: (46) tcp        0      0 128.39.74.16:80         157.158.24.40:4147      TIME_WAIT
cf:cube: (47) tcp        0      0 128.39.74.16:80         157.158.24.40:3895      TIME_WAIT
cf:cube: (48) tcp        0      0 128.39.74.16:80         157.158.24.40:3642      TIME_WAIT
cf:cube: (49) tcp        0      0 128.39.74.16:80         80.213.238.106:4318     FIN_WAIT2
cf:cube: (50) tcp        0      0 128.39.74.16:80         80.213.238.106:4319     TIME_WAIT
cf:cube: (51) tcp        0      0 128.39.74.16:80         157.158.24.40:4133      TIME_WAIT
cf:cube: (52) tcp        0      0 128.39.74.16:80         157.158.24.40:3881      TIME_WAIT
cf:cube: (53) tcp        0      0 128.39.74.16:80         157.158.24.40:3628      TIME_WAIT
 {
 DNS key: 157.158.24.40 = arm.iele.polsl.gliwice.pl (47/53)
 DNS key: 80.203.17.11 = 11.80-203-17.nextgentel.com (1/53)
 DNS key: 66.196.72.28 = j3118.inktomisearch.com (1/53)
 DNS key: 80.202.77.107 = 107.80-202-77.nextgentel.com (2/53)
 DNS key: 80.213.238.106 = ti100710a080-3690.bb.online.no (2/53)
 -
 Frequency: 157.158.24.40    |***********************************************  (47/53)
 Frequency: 80.203.17.11     |*  (1/53)
 Frequency: 66.196.72.28     |*  (1/53)
 Frequency: 80.202.77.107    |**  (2/53)
 Frequency: 80.213.238.106   |**  (2/53)
 }
 -
 Scaled entropy of addresses = 12.7 %
 (Entropy = 0 for single source, 100 for flatly distributed source)
 -
cf:cube: -----------------------------------------------------------------------------
------
cf:cube: State of incoming.www peaked at Fri Feb 20 19:57:23 2004
```

Another example of a low entropy smtp (possible distributed spam operation in progress):

```
cf:nexus: HIGH ENTROPY Incoming smtp anomaly high 2 dev on nexus/Sat Aug  6 14:29:58 2005
cf:nexus: ----------------------------------------------------------------------------
-------
cf:nexus: In the last 40 minutes, the peak state was q = 25:
 {
 DNS key: 81.218.96.62 = bzq-218-96-62.red.bezeqint.net (1/25)
 DNS key: 85.152.128.208 = cm-85-152-128-208.telecable.es (1/25)
 DNS key: 82.40.141.102 = 82-40-141-102.cable.ubr04.uddi.blueyonder.co.uk (1/25)
 DNS key: 68.189.49.33 = 68-189-49-33.dhcp.rdng.ca.charter.com (1/25)
 DNS key: 61.47.218.167 = 61.47.218.167 (1/25)
 DNS key: 211.170.184.229 = 211.170.184.229 (2/25)
 DNS key: 64.65.134.186 = static-64-65-134-186.dsl.pdx.eschelon.com (1/25)
 DNS key: 61.9.82.105 = 61.9.82.105.mozcom.net (1/25)
 DNS key: 219.234.19.166 = 219.234.19.166 (1/25)
 DNS key: 68.60.199.206 = pcp07641774pcs.calhun01.ga.comcast.net (1/25)
 DNS key: 82.216.163.149 = ip-149.net-82-216-163.suresnes3.rev.numericable.fr (1/25)
 DNS key: 84.9.43.170 = host-84-9-43-170.bulldogdsl.com (1/25)
 DNS key: 84.59.55.171 = dsl-084-059-055-171.arcor-ip.net (1/25)
 DNS key: 200.104.102.141 = pc-141-102-104-200.cm.vtr.net (1/25)
 DNS key: 85.152.225.126 = cm-85-152-225-126.telecable.es (1/25)
```

```
 DNS key: 218.81.136.17 = 218.81.136.17 (1/25)
 DNS key: 83.84.225.251 = 5354E1FB.cable.casema.nl (1/25)
 DNS key: 60.198.145.92 = 60-198-145-92.static.tfn.net.tw (1/25)
 DNS key: 61.53.185.218 = 61.53.185.218 (1/25)
 DNS key: 81.53.86.235 = ANantes-154-1-63-235.w81-53.abo.wanadoo.fr (1/25)
 DNS key: 58.142.251.14 = 58.142.251.14 (1/25)
 DNS key: 220.234.173.3 = 220.234.173.3 (1/25)
 DNS key: 219.251.118.195 = 219.251.118.195 (1/25)
 DNS key: 68.44.158.165 = pcp04364785pcs.glstrt01.nj.comcast.net (1/25)
 -
 Frequency: 211.170.184.229  |**       (2/25)
 Frequency: 68.44.158.165    |*        (1/25)
 Frequency: 219.251.118.195  |*        (1/25)
 Frequency: 220.234.173.3    |*        (1/25)
 Frequency: 58.142.251.14    |*        (1/25)
 Frequency: 81.53.86.235     |*        (1/25)
 Frequency: 61.53.185.218    |*        (1/25)
 Frequency: 60.198.145.92    |*        (1/25)
 Frequency: 83.84.225.251    |*        (1/25)
 Frequency: 218.81.136.17    |*        (1/25)
 Frequency: 85.152.225.126   |*        (1/25)
 Frequency: 200.104.102.141  |*        (1/25)
 Frequency: 84.59.55.171     |*        (1/25)
 Frequency: 84.9.43.170      |*        (1/25)
 Frequency: 82.216.163.149   |*        (1/25)
 Frequency: 68.60.199.206    |*        (1/25)
 Frequency: 219.234.19.166   |*        (1/25)
 Frequency: 61.9.82.105      |*        (1/25)
 Frequency: 64.65.134.186    |*        (1/25)
 Frequency: 61.47.218.167    |*        (1/25)
 Frequency: 68.189.49.33     |*        (1/25)
 Frequency: 82.40.141.102    |*        (1/25)
 Frequency: 85.152.128.208   |*        (1/25)
 Frequency: 81.218.96.62     |*        (1/25)
 }
 -
 Scaled entropy of addresses = 4.0 %
 (Entropy = 0 for single source, 100 for flatly distributed source)
 -
cf:nexus: ----------------------------------------------------------------------------
-------
cf:nexus: State of incoming.smtp peaked at Sat Aug  6 14:29:58 2005
cf:nexus: HIGH ENTROPY Incoming smtp anomaly high anomaly !! on nexus/Sat Aug  6 14:29:58 2005█
```

2 Anomaly response

2.1 Anomaly Classes

When cfengine detects an anomaly, it classified the current statistical state of the system into a number of classes.

Cfengine classifies anomalies by whether the currently measured state of the system is higher or lower than the average for the current time of week. The amount of deviation is based on an estimate of the 'standard deviation'. The precise definition of the average and standard deviations is complex, and is discussed in the paper "M. Burgess, Probabilistic anomaly detection in distributed computer networks", (submitted to Science of Computer Programming, and available on the web).

The list of measured attributes is currently fixed to the following:

The first part of the string is from the list:

```
Users
RootProcs
UserProcs
DiskFree
LoadAvg
```

Socket counts of network services distinguish between incoming and outgoing sockets (to a service or from a client).

`netbiosns`
: Registers traffic to/from port 137.

`netbiosdgm`
: Registers traffic to/from port 138.

`netbiosssn`
: Registers traffic to/from port 139.

`irc` Registers traffic to/from port 194.

`cfengine` Registers traffic to/from port 5308.

`nfsd` Registers traffic to/from port 2049.

`smtp` Registers traffic to/from port 25.

`www` Registers traffic to/from port 80.

`ftp` Registers traffic to/from port 21.

`ssh` Registers traffic to/from port 22.

`wwws` Registers traffic to/from port 443.

If you have tcpdump program installed in a standard location, then `cfenvd -T` collects data about the network flows to your host.

`icmp` Traffic belonging to the ICMP protocol (ping etc).

`dns` Traffic to port 53, the Domain Name Service (usually a special case of UDP).

`udp` Miscellaneous UDP traffic that is not related to DNS.

`tcpsyn` Registers TCP packets with SYN flag set.

`tcpack` Registers TCP packets with ACK flag set.

`tcpfin` Registers TCP packers with FIN flag set.

`misc` Registers all other packets, not covered above.

When it has accurate knowledge of statistics, '`cfenvd`' classifies the current state into 3 levels:

`normal` means that the current level is less than one standard deviation above normal.

`dev1` means that the current level is at least one standard deviation about the average.

`dev2` means that the current level is at least two standard deviations about the average.

`anomaly` means that the current level is more than 3 standard deviations above average.

Each of these charaxterizations assumes that there are good data available. The '`cfenvd`' evaluates its data and decides whether or not the data are too noisy to be really useful. If the data are too noisy but the level *appears* to be more than two standard deviations above aaverage, then the category `microanomaly` is used.

Here are some example classes:

```
UserProcs_high_dev2
UserProcs_low_dev1
www_in_high_anomaly
smtp_out_high_dev2
```

2.1.1 Cfenvd Class list

Base classes:

```
users
rootprocs
otherprocs
diskfree
loadavg
netbiosns_in
netbiosns_out
netbiosdgm_in
netbiosdgm_out
netbiosssn_in
netbiosssn_out
irc_in
irc_out
cfengine_in
cfengine_out
nfsd_in
nfsd_out
smtp_in
smtp_out
www_in
www_out
ftp_in
ftp_out
ssh_in
ssh_out
```

```
  wwws_in
  wwws_out
  icmp_in
  icmp_out
  udp_in
  udp_out
  dns_in
  dns_out
  tcpsyn_in
  tcpsyn_out
  tcpack_in
  tcpack_out
  tcpfin_in
  tcpfin_out
  tcpmisc_in
  tcpmisc_out
```

Suffixes:

```
_high_microanomaly
_low_microanomaly

_high_dev1
_low_dev1

_high_dev2
_low_dev2

_high_anomaly
_low_anomaly

_high_ldt
_low_ldt
```

2.2 Variables

Cfenvd sets variables which cache the values that were valid at the time of the anomaly's occurrance. These are of the same form as above.

```
value_rootprocs
average_rootprocs
stddev_rootprocs

value_nsfd_in
average_nfsd_in
stddev_nfsd_in
```

The Leap Detection Test buffer is called

```
ldtbuf_users
ldtbuf_otherprocs
```

etc.

2.3 Entropy

Entropy is a statistical characterization of the distribution digits in an arrival process over long times. You can learn more about entropy in Mark Burgess, *Analytical Network and System Administration*.

For network related data, cfengine evaluates the entropy in the currently measured sample of measurements, with respect to the different IP addresses of the sources. You can use these to predicate the appearance of an anomaly, e.g.

```
entropy_www_in_high
entropy_smtp_in_low
```

For example, if you only want to know when a huge amount of SMTP traffic arrives from a single IP source, you would label your anomaly response:

```
entropy_smtp_in_low.smtp_in_high_anomaly::
```

since the entropy is low when the majority of traffic comes from only a small number of IP addresses (e.g. one). The entropy is maximal when activity comes equally from several different sources.

2.4 Log utilities

How shall we respond to an anomalous event? Alerts can be channelled directly to syslog:

```
SysLog(LOG_ERR,"Test syslog message")
```

Software that processes logs can thus be interfaced with via the syslog interface.

2.5 Persistent alerts

DEFCON 1

Another application for alerts is to pass signals from one cfengine to another by persistent, shared memory. For example, suppose a short-lived anomaly event triggers a class that relates to a security alert. The event class might be too short-lived to be followed up by cfagent in full. One could thus set a long term class that would trigger up several follow-up checks. A persistent class could also be used to exclude an operation for an interval of time.

Persistent class memory can be added through a system alert functions to give timer behaviour. For example, consider setting a class that acts like a non-resettable timer. It is defined for exactly 10 minutes before expiring.

```
SetState("preserved_class",10,Preserve)
```

Or to set a class that acts as a resettable timer. It is defined for 60 minutes unless the SetState call is called again to extend its lifetime.

```
SetState(non_preserved_class,60,Reset)
```

Existing persistent classes can be deleted with:

```
UnsetState(myclass)
```

Variable Index

(Index is nonexistent)

Concept Index

FAQ Index

(Index is nonexistent)

Table of Contents