Global Information Assurance Certification Paper

Logging, Monitoring and Trending

Practical: GIAC Security Essentials (GSEC)

GSEC practical requirements (1.4b) - Option 1

Author: Peter G Williams

June 2004

Submitted: June 22nd 2004

Logging, Monitoring and Trending

Abstract

The importance to the security practice of careful oversight of all network security policy enforcement point devices and their logging, trapping and monitoring cannot be overstated. This paper starts with discussing the basic difference between logging and monitoring. Following this is an elaboration on other metrics that could be gathered in addition to existing monitoring norms like bandwidth utilisation (the most commonly monitored metric). These include metrics such as memory, CPU and connection details. Finally the paper will provide some examples of operational anomalies and related security implications found by monitoring these metrics and correlating the data. The examples provided are from *Cisco* equipment using a simple tool, *MRTG*. Details of the *SNMP* Object IDs, how to find them and examples of their use are attached as the example appendix.

Preface

In depth defence concepts and aspects such as perimeter defence, access control and policy enforcement points, intrusion detection systems, virtual private networking and securing network (and host) entry points throughout the enterprise are all important components of most security strategies employed today. Most strategies are a combination of some or all of these components (amongst others) and they all require some degree of management. Host devices, appliances and network components contributing to the overall security strategy of an enterprise can usually provide a plethora of information of various types which comes in many forms.

One facet of the management of these components entails the collection and correlation of these various types of information and the presentation of them in a more human readable form that is easy to assimilate. Reporting that is easy to digest is one aspect and outcome of good monitoring, but correlation of events and interpretation of the output are probably more useful to the security professional.

The truism of a picture painting a thousand words is often the only way to take in the vast amounts of information provided by these components. These pictures or graphs are the most practical way to allow review and observe correlations of the interactions between these components.

Identifying an anomaly either by alert, report or observation followed by correlation of graphical summarisations can allow a security professional to quickly drill into a component or segment of interest. Historical summarisation can also be used to gauge the utilisation of a variety of aspects of these components and help baseline what constitutes normal operation. These quarterly or annual summaries also facilitate capacity planning and could be used to contribute to the justification for upgrade or change.

NTP and synchronised time stamps

The importance of synchronising time between devices and hosts is critical for setting the context to make any meaningful correlation possible. As logging and monitoring is often performed on different hosts and from a variety of different component devices or appliances that make up the security strategy, consistent time-stamps are critical. There are many other papers that discuss the Network Time Protocol, *NTP*, implementation and security implications. Mat William discusses *NTP* security issues at length in his paper "*NTP Security"* [1]. It is beyond the scope of this document to discuss or further elaborate on *NTP* or *SNTP*. However, as this paper discusses the correlation between various logging and monitoring events, establishing that time is set and consistent between both spoke (components) and hub (loggers and monitors) is mandatory.

Logging and Monitoring

Logging of an event occurs for various reasons, usually configurable, by one of the components in the security strategy. These logs can be kept locally on the component, or centralised to a loghost where the log data can be kept and analysed.

The logging stream is initiated from the component to the logging host, usually in the form of an *SNMP-trap* or *SYSLOG* message. Some proprietary systems such as Checkpoint use *RDP*. Others such as Windows use *NetBIOS* and *RPC* to send their log messages and open their event viewers.

Once received by the loghost the log or alert message content can traverse pattern matching scripts and alerting programs whilst being written to disk. Care must be taken in storage, retention and disposal of these logs as well as controlling access to them. Financial institutions have a good set of guidelines for 'what' to log, as well as for storage, retention, access and disposal issues. Appendix A (p13) of *"Privacy and Information Security Regulatory Compliance Guidelines October 2003"* has a 'good practices' summary for logging and monitoring requirements based on the FFIEC guidelines document. [2]

Monitoring components of the security strategy will vary from component to component and according to requirement. Some proprietary appliances or devices will only allow monitoring of specific aspects by proprietary applications. This discussion will limit 'monitoring' to polls or read-requests that are initiated from a monitoring sensor or host usually in the form of *SNMP get* request.

In some situations even running the *SNMP* daemon on a component has risks. Mitigating actions to address some of these risks include sufficient care in selecting complex community names and implementing access controls which restrict access to selected nodes. Only read access to the *SNMP* Object Identifiers or the Management Information Base (*OIDs* or MIB) is required. Where possible, use of the more advanced feature of *SNMP* such as those available with *SNMPv3* should be implemented. These include use of authenticated and encrypted channels. In a previous paper by Dan Keldsen

"An Oversight Layer for Layers of Defence" the insecurities of *SNMP* are discussed, along with *SNMPv3, v4* and *OPSEC* alternatives. [3]

Even so, it may still be the case that in be some instances running *SNMP* may pose an unacceptable risk. In those cases other steps will be required to gather the data, and these steps will vary. Checkpoint *OPSEC* for example provides an open platform for developers to interface their appliances, applications and tools. Pre-socialised monitoring solution alternatives exist for components adhering to the requirements of the ***O**pen **P**latform for **Sec**urity* initiative [4].

There are many papers and other *SANS* materials dealing with the monitoring of log files, and the utilities available to perform reporting and alerting against them, as well as their storage, retention and disposal. *Cisco* Open Systems community (*COSI*) tools and the *SWATCH* scripts [5] are two good examples of utilities for log monitoring. Weaknesses and vulnerabilities in various logging, monitoring and polling implementations have also been discussed as have some alternatives.

The focus of this paper is not on logging, the tools, protocols, vulnerabilities or mitigating actions. Suffice to state that a good logging strategy is an integral part of the overall management of the components that contribute to the security strategy. Monitoring of the components must also occur, the 'how' can vary, as can the basic 'what's. The `how' will vary subject to preference and funding available. The `what' is subject to requirements, but these requirements should include more than just bandwidth collections.

Monitoring

There are many commercial Enterprise and Network Management Systems and tools available capable of monitoring the various components using a variety of polling technology including *SNMP*. These include platforms like *HP Openview, Cabletron Spectrum, Statscout, SiteScope* and collaboratively developed and maintained works like *OpenNms* and *GNU GPL* works such as *MRTG*. [6] These commercial tools and *OpenNms* are all capable of providing the graphical representations under discussion. It is assumed that users of these types of tools will be familiar enough with their operation to collect these statistics.

In previous papers such as Seham Mohamed GadAllah's paper on *"The importance of Logging and Traffic Monitoring for Information Security"* [7] *MRTG* has been discussed in terms of its use in bandwidth monitoring and for the ability to detect anomalous conditions from the interpretation of the graphs. Some of these conditions include virus outbreak detection by observing the impact to bandwidth utilisation.

Monitoring Requirements, views and filtering

Bandwidth utilisation is arguably the most commonly collected metric. The relationship between bandwidth and cost is usually the reason for monitoring its use (or misuse). The collection and display of these interface traffic statistics along with other statistics such as interface error rates, congestion

rates, CPU utilisation, memory utilisation and connection table counters can also reveal anomalies. The display of these statistics and apparent anomalies can be useful in focusing the attention of the security professional on a component of interest.

Filtering the graphical representations facilitates review and correlation. The review of all CPU utilisations or all connection table graphs will allow for some correlation to occur between them. A similar review of specific devices or by device types may show a variety of different trends such as the CPU utilisation in all perimeter routers. This is also true of periodic graphs for a specific host or collection type such as the annual firewall connection table statistics or the weekly collections for a perimeter router. Having all the collections displayed on the one index does not prohibit correlation it just impedes the rate at which the correlation of anomalous events can occur.

One of the advantages of *MRTG* type utilities is the 50-mile-high-view and periodic trends provided by the weekly, monthly and yearly graphs. Although peaks on these tables appear lower due to the nature of averaging, the periodic data facilitates trend analysis. Conversely, if you are looking for a more granular resolution of instantaneous peaks, other utilities can provide a better display.

Bandwidth – used, usable and unusable

In today's switched networks, error rates are usually relatively low. An increase in error rate may be indicative of various conditions. In the older shared network segments, error rates include collisions and could just indicate increased traffic conditions. It is suggested that when setting up bandwidth utilisation collections or graphs the same interface's error rates are included. Changes in error rates can be used to correlate with other events in LAN or carrier utilisations, memory or CPU. These errors can come from a large variety of sources and for a large variety of reasons, which may include aging hardware, misbehaving or mis-configured hosts or just an increase in traffic. Errors equate to unusable bandwidth to some degree.

As well as error rates, collecting specific WAN metrics relevant to the technology is also useful such as **F**orward and **B**ackward **E**vent **C**ongestion **N**otification (*FECN* and *BECN*) rates in *Frame Relay* or *ATM* circuits. If the carrier is sending *FECN* or *BECN* frames when utilisation is below the subscribed CIR, then the carrier's network fabric may be underperforming or the carrier over charging for services provided. Anomalies such as the bandwidth not increasing dramatically, or not peaking as is expected, whilst the *FECN*/*BECN* rates change, may indicate congestion within either the carriers switch or networks. Careful cross correlation is required before consulting with the carrier. It is not unusual to have some *FECN* and *BECN* frames during peak load periods, or when ever bandwidth utilisation exceeds the subscribed *CIR* (or the *SCR* for *ATM*) on either end of the *SVC* or *PVC*.

Excessive error rates, congestion rates or the over utilisation of bandwidth, CPU or memory on any of the components that are part of the security strategy will impact performance. Impaired or under-performance in extremes

circumstances will in turn impact availability. This is one of the key ingredients in the CIA triangle of Confidentiality, Integrity and Availability.

CPU and Memory

Gathering CPU utilisation data is relatively straight forward as it is an integer less than 100. *Cisco* routers provide a last second, last minute and last 5 minutes CPU utilisation any of which can be graphed. Firewall CPU utilisation is also an important metric and can usually be gathered just as easily. With firewalls the availability of the CPU utilisation *OID* depends on the counter or *MIB* availability in the specific make and model.

Memory utilisation and the available *OIDs* are a little more complex. Initially only the amount of memory used and free is important, but subsequently plotting the number of specific size buffers and other memory utilisation *OIDs* can also be useful. Buffer counter collections will be of most use when memory free graphs indicate anything but a stationary point of inflection. In other words when the memory free to used graph displays a gradient in either direction the buffer collections should be established. Under most normal conditions an almost flat line of memory free to memory used is to be expected.

Memory and CPU *SNMP OIDs* for *Cisco* routers and PIX firewalls are provided in the example appendix. Also included are various size buffer counters for the PIX. Somix[8] provides further stub configuration templates to facilitate collection for various vendor's components.

Connection table

Some firewall type components of the security strategy will be state based, and will maintain some form of connection table. Some components are licensed (or maintained) based on concurrent connections, and are costed accordingly. Collecting connection table utilisation levels and connection rates can tell you various things about the performance of different components in the security strategy.

Along with the number of simultaneous concurrent connections and the peaks, the connection rate can also be useful in identifying a variety of operational anomalies such as syn-attacks or vulnerabilities in the configuration of components. Figures gathered from firewalls may be dampened by the firewalls own response to these attacks, if sufficient attacks occur to trip the firewalls syn-defence thresholds.

Depending on the component type, collecting these details can be straight forward. Example *OIDs* have been supplied in the examples appendix for the *Cisco* PIX. Some components may not provide this metric or rely on the underlying system *MIB* which may or may not make these counters available.

For components of the security strategy with maximum licensed connection limits, the peak connection count can be invaluable in determining the appropriate licensing requirements and future license size or upgrade

requirements. Correlations of peaks in these connection counters with peaks in other collected metrics can be used to plan and scale components of the strategy to suit the load.

Monitoring thresholds and Alerting

Collecting and graphing this data is one matter but at some point normal utilisation levels become abnormal and an investigation may be warranted. The threshold, or point at which these levels become abnormal, will be a function of the metric being monitored. This will be a static value or a dynamic level that will be based on a mathematical algorithm.

Once the metric being monitored passes the threshold a suitable message or trap needs to be sent and displayed. An operator needs to be informed that action is required to investigate and possibly remediate an (urgent) issue. This is a basic function of most network management systems mentioned above.

Any interface to an alerting or ticketing system will rely heavily on the tools used for monitoring as well as the ticketing, paging and display systems being used. Even the *MRTG* tool can be scripted to alert at a prescribed threshold and to send an email or *SNMP-trap* notifying relevant parties that the event has occurred. Thomas J Muggli [9] discusses "MRTG Thresholds" monitoring configuration on his webpage http://www.cloudnet.com/~tom/mrtg/thresh.html . The example that Thomas provides could easily be incorporated into the example appendix to add threshold alert monitoring.

The maintenance of a statically maintained threshold on each metric collected would become too costly and prone to error with larger deployments over time. Smarter tools like *HPOV* or *Spectrum* that can dynamically adjust the threshold or only raise an alert after it's been triggered a given number of times over a given period are better suited to performing this function.

It is not the primary focus of this paper to further discuss threshold triggers, the mathematics of dynamic threshold recalculations or some of the alerting and ticketing system integration scripts that are available. However, no discussion on monitoring would be complete without mentioning thresholds and alerting. With security strategies that deploy a number of components it could be argued that the automation of threshold alerting is as critical as the monitoring and logging of the entire strategy.

Examples of use

To illustrate the impact of correlating and trending the data from several collections, a few instances will be selected from each of the additional metrics advocated in this paper.

CPU

Initially a *CIR* threshold alert brought the attention of an operator to an over utilised *WAN* segment. In figure 1 the segment utilisation peak appears at

13:00 and then almost saturates this 512k *Frame Relay* service. The red dotted line indicates the 256k *CIR*.

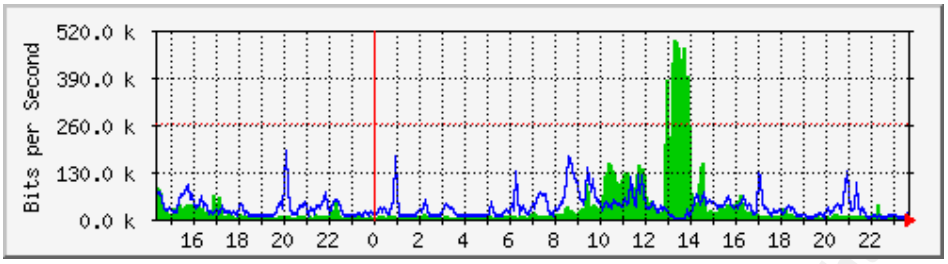

Figure 1 – Serial link utilisation

On quick inspection of the graphs filtered by CPU utilisation, it became apparent that a component of the security strategy (a firewall node) was using the CPU differently than normal. When correlated to the security strategy topology map the security component was identified as being downstream (the other side) of this WAN link. In the figure 1 above, the WAN utilisation spike that triggered the alert occurred at 13:00 and coincides with the CPU utilisation peak in figure 2. Normally the CPU on this component jumps from zero to one, with an occasional peak to 4 or 10. In figure 2, the size of the 13:00 peak dwarfed the normal utilisation rates to an insignificant flat line.

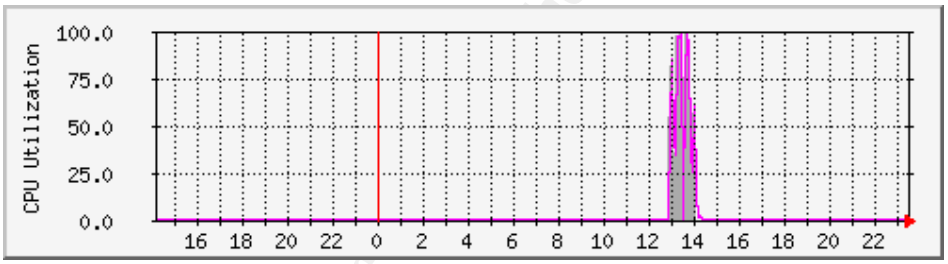

Figure 2 – CPU utilisation

The operator was then able to focus on the component, in this instance a firewall and establish the reason for this CPU hammering. The situation was quickly remedied. In this case a logging service had failed on one of two loghosts configured for this firewall. Every time the firewall had a log event to send to the loghosts, an *ICMP* 'service unreachable' packet was being sent from the broken loghost back to the firewall. The firewall in turn logged the *ICMP* received for a service unreachable event, which included sending the log message to both loghosts. The broken loghost then returned another unreachable, causing a loop, and the CPU utilisation almost saturated. The broken server was removed from the firewall configuration at 14:00 to stop the loop until the secondary loghost was fixed.

The logging failure was almost a denial of service by the affect it had on the resources and *WAN* links. The remaining loghost was very busy as a result of the flood. The possible loss of event information in the storm that resulted could well have masked various attacks or probes against or through the firewall.

Memory

In this example, it was observed that a component would occasionally reboot and was reporting errors when updating an access control list. The reboot was effectively terminating the sessions the component was carrying and therefore data integrity and service availability were both impaired. The inability to reliably update the access controls was increasing over time and posed a medium level risk. The access controls had to be updated regularly to facilitate changes in the business operation that in turn required changes to the permitted connectivity between the network segments.

Memory and CPU metrics were added to the established bandwidth utilisation collections. Initially the memory utilisation rate appeared to be low as only 25% (approximate) of available memory was used. Observation of the periodic memory utilisation rates for this component in figure 3 indicated the memory consumed was not released during the less busy or congested periods. The graphs in figures 3 and 4 display the monthly and yearly perspectives of the memory depletion rate for this component.

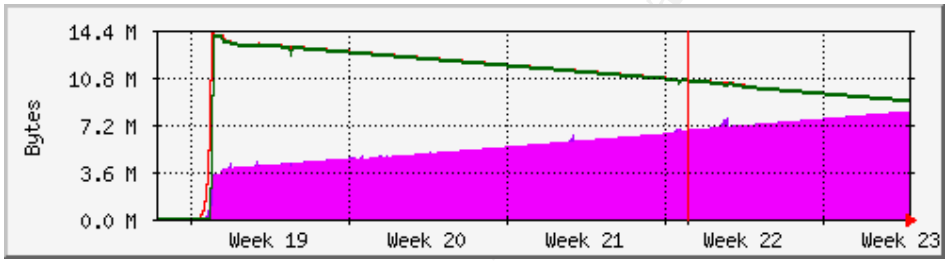

Figure 3 – Monthly memory utilisation

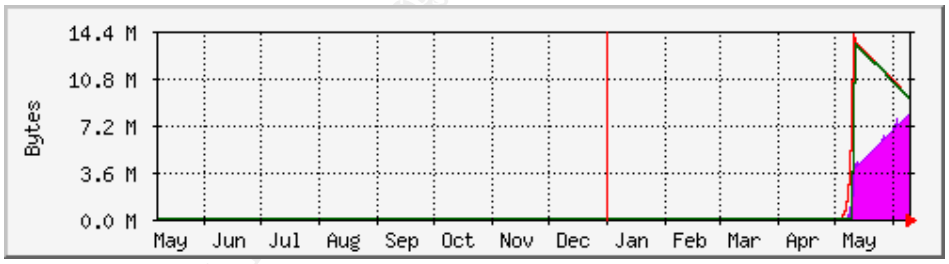

Figure 4 – Annual memory utilisation trend

Over time either load or configuration complexity had increased significantly, or a memory leak had been detected. Configuration had not changed drastically nor had load, and as a result the firmware was suspected and would most likely need to be upgraded. In the daily graph, the memory changes were only just detectable, but the weekly, monthly and yearly graphs showed the leakage trend. The yearly graph in figure 4 showed that free memory would eventually be exhausted and helped explain how the router performance was being degraded and why it would eventually reboot.

CPU and Memory

The following example was used to help justify an upgrade to a component which was performing badly according to user's observation and reports. Collections had been established to determine how much RAM was used and

how much was free. Various other collections had been established including the bandwidth, CPU and error rates. The error rates were negligible and bandwidth utilisation was still low, although trending had started to indicate an increase in utilisation when observed on the yearly utilisation graph, as shown in figure 5. In this instance the bandwidth monitoring had been established well before any issues with performance were raised and the CPU and memory collections established.

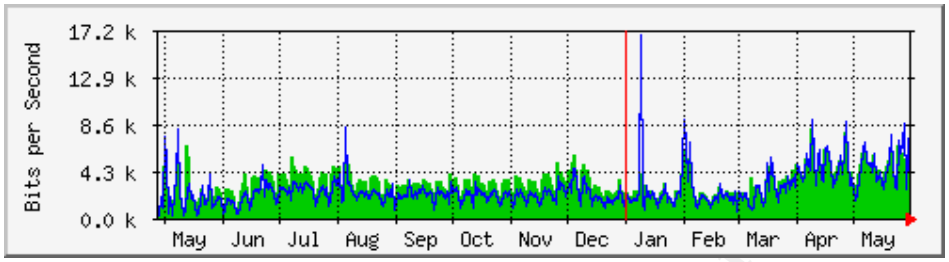

Figure 5 – Annual link utilisation

As there was bandwidth to spare and error rates were low to nonexistent, it was likely the responsibility lay with either the CPU and/or the memory. The CPU was operating at about 25% of the average utilisation mark over 5 minutes on a regular basis as shown in figure 6. Memory was low to begin with only 4Mbytes and was utilised at the 50/50 mark as shown in figure 7.

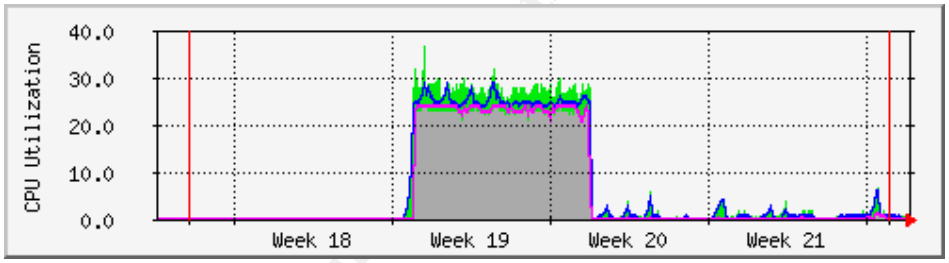

Figure 6 – Monthly CPU utilisation trend

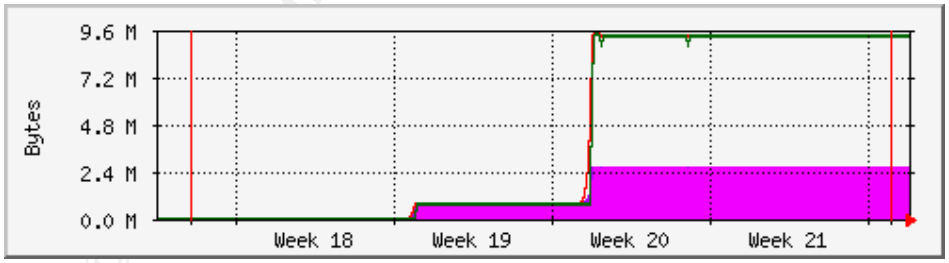

Figure 7 – Monthly memory utilisation trend

The combination of low memory and high CPU were likely to be contributing to the overall poor performance.

The correlations were provided as input to a router upgrade justification in week 19, and a contingency spare router was provided for commissioning early the next week. The changes in both graphs indicate the improvement in resource utilisation, more so than the annual utilisation in figure 5. User accounts report better response.

This perimeter filtering uses CPU cycles and consumes memory as a function of *ACL* length. Generally, the longer the Access Control List the more RAM will be required. Further refinement of the access control lists and other configuration optimisation was initially considered. The *ACL* was already short and simple; most of the security policy was enforced downstream in a firewall. Primarily because of the low RAM to begin with coupled with the general age of the equipment (had an *AUI* port, no *UTP*) and given the CPU intensive nature of it's primary purpose, an upgrade to the device was justified rather than just the RAM upgrade.

Perimeter filtering uses some resources within the security component, even with a relatively low speed link. Defending the perimeter must be resourced appropriately and reviewed periodically. Being an Internet facing *WAN* router this component was exposed to a higher packet rate over time. Even though most of the packets/noise was being discarded, there was more of it compared to when the router was commissioned.

Connection table

One of the more interesting examples of this instance was a pair of PIX firewalls that reportedly failed over to its partner every few days. A series of collections was established one of which was a connection table counter. During the first few days of the collection, it became apparent that number of connections was always increasing, even after hours. The graph in figure 8 below shows the number of connections increasing and pushing the red 'peak used' counter ever higher. This pair of firewalls has a maximum licensed connection limit of 64k (65535) and would require an upgrade in the short to mid term if these peak utilisation levels were sustained.

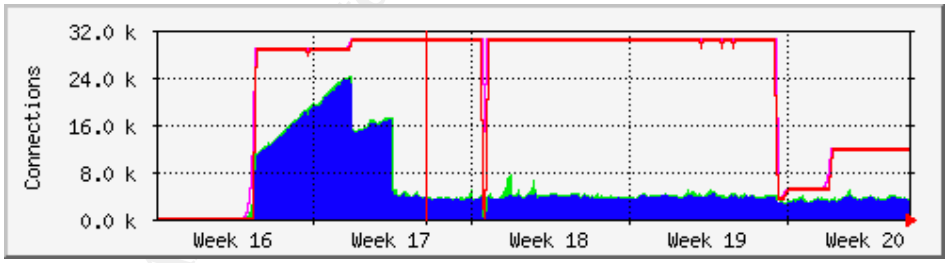

Figure 8 – Connection table counters

The syn-defender configuration had not detected any problems and was processing the connection requests as received. It appeared to be a denial of service type activity. Correlation of summarised *SYSLOG* data from this firewall for the period pinpointed the problem, which turned out to be a mis-configuration of just one of the hosts on one of the segments. Subsequently it was suspected that the threshold for the syn-defender defence to cut-in had been configured too low on this component for this device. This mis-configuration of the threshold left the component susceptible to an effective form of denial of service attack, by just one device.

On the Sunday of week19 the pair was rebooted to apply patches that were released by the vendor and were not related to any issues or problems at the time. The application of patched firmware is consistent with good patch management which should be practiced for all components of the security strategy at the very least; if not a requirement for every OS based component of the enterprise. As the patching requires a reboot, the peak counters are reset to zero allowing peaks created by anomalous situations to be identified as such.

Congestion Event

The example of a congestion notification event (*FECN/BECN*) collection that was established during the time of writing is included to highlight potential security issues observed. The performance of a service was reported to be slow by a consumer. Simple maintenance issues on the default gateway had also been noted; errors had been observed loading and saving configuration files. The components between the consumer and their service were identified and several additional metrics were added to the usual bandwidth utilisation collections including *FECN* and/or *BECN* rates, error rates, CPU utilisation and memory consumption.

The bandwidth utilisation rates look normal to low as shown in figure 9. Using a ping with large timeouts between the two routers, directly across the *WAN* link, it was observed that the serial interface was occasionally dropping packets even though the bandwidth utilisation appeared low at that time. No noticeable errors were reported by the interface during the ping testing. The bandwidth utilisation graph in figure 9 indicates several utilisation peaks over the *CIR* (dotted red line) were occurring as expected throughout the week.

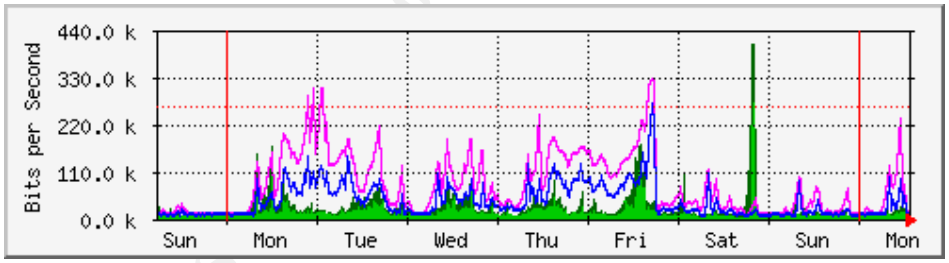

Figure 9 – Weekly bandwidth utilisation

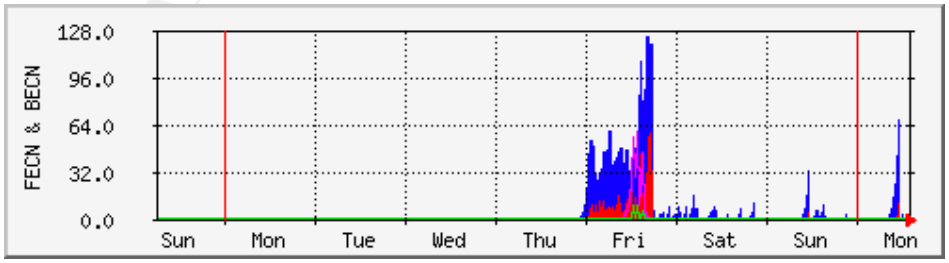

Figure 10 – Weekly congestion rates

Observation of the weekly bandwidth utilisation graph indicated the service appeared to be able to peak over the subscribed 256k CIR. This established that bursts occur and could coincide with a minimal number of congestion

events being recorded. This would indicate normal burst type operation, as observed on Saturday night in the figures 9 and 10 above.

Correlation of the more granular daily graphs in figures 11 and 12 and the congestion events of Friday from figure 10 above, it became apparent that these congestion events were not directly peak load related. In the graphs in figure 12 the 11:00 am peak of 68 BECN/sec corresponds to the Monday peak of only 180k in figure 11. Even on the previous day, a Sunday, the congestion peaks at 11am do not coincide with a utilisation peak.

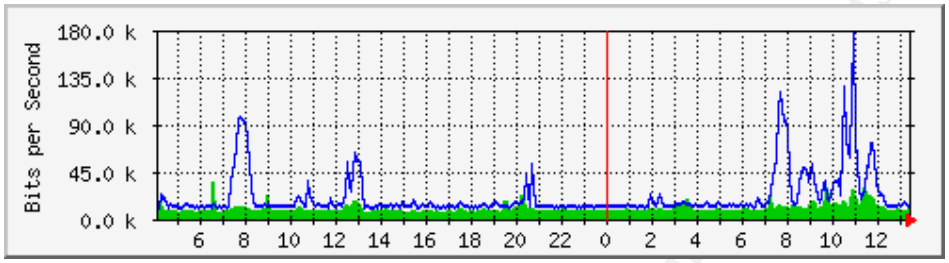

Figure 11 – Daily bandwidth utilisation

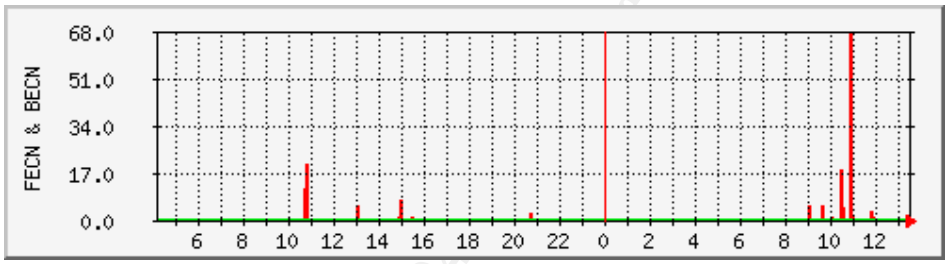

Figure 12 – Daily congestion peaks

Closer correlation of the congestion notification rate to the interface utilisation rate showed periods with low bandwidth and high congestion rates for the *PVC*. This is abnormal behaviour for a *Frame Relay* service and is indicative of an issue within the carrier's network. There were no observable changes in memory consumption and only normal to marginal CPU change during the period. As the congestion notification event rate peaks appear to be occurring when the bandwidth is below the *CIR*, further investigation and consultation with the carrier is required.

Both service availability and session integrity are compromised in this situation, as packets are randomly dropped with varying impact to the service or application. Data integrity is a function of the protocols used at both the network and application layers. In this packet loss situation, some protocols will fail integrity checks and the application will appear to hang, disconnect or fail without reason. Other applications may pass bad data completely compromising the integrity of the file or application. Some more tolerant applications written to perform over a variety of communications mediums may retransmit lost data fragments and continue to perform, with erratic or slow response.

The configuration maintenance failures could have been more serious than just chunks of a configuration file missing on the *TFTP* host. Due to the nature of *TFTP*, dropped packets will be lost and the result can be that random parts of the data can go missing. Had an ACL update been loaded with errors and then applied to the interface, the missing configuration lines would have failed to provide or protect to some degree.

An increase in false alarms from systems and security components on the other side of this service can be directly related to the packet loss, given the connectionless nature of *ICMP* and *UDP* traffic types. Logging and monitoring could all easily be adversely affected by this packet loss, as *ICMP* and *UDP* are the basis of many monitoring and logging protocols. As an example, the loss of *SYSLOG* event messages can have serious ramifications for the security professional.

The response to and servicing of these false alerts also has an immediate cost implication for on-call operations personal and therefore the total cost of the operation. Increased exposure to risk by failure to implement policy in the form of a failure to successfully apply access controls to an asset or service is a significant issue for the security professional.

Bandwidth

The final brief example will show the affects of a slammer (SQL virus) outbreak on bandwidth. The graph in figure 13 shows the dramatic affect of this virus as a new outbreak occurs at 20:00. The SQL-slammer infection was isolated to one un-patched machine at the end of this *WAN* link, a 30Mbit Peak Cell rate *ATM* service with 10Mbps Sustained Cell Rate. This service was in affect running at 130% of the subscribed capacity of 10Mbps for two hours, and so all the traffic over the 10Mbps rate could have been legitimately discarded by the carrier.

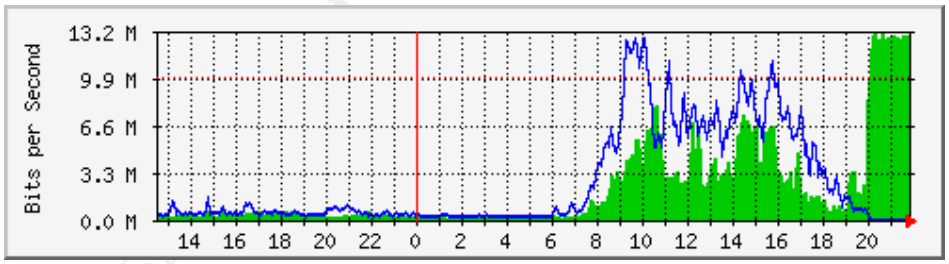

Figure 13 – The affect of SQL-slammer on a WAN link.

The impact to applications and services during this time would have been similar to those for the congestion events discussed above, but far worse. The resulting utilisation extreme and potential for high packet loss would significantly affect both availability and integrity of any application or service.

Recommendations

In some environments the risk of running *SNMP* is considered too great, and alternatives should be sought. In considering risk and the higher cost of monitoring alternatives, the single loss expectancy (*SLE*) calculation for a major event or failure should include the potential cost of lost service availability. Poor performance of a business critical service may be quantifiable in terms of cost to the business operation. The subsequent damage to the integrity of a brand or an enterprise is less tangible and can be difficult to measure.

Where monitoring or polling the components of the security strategy permits, consideration should be given to monitoring more than just bandwidth metrics. Error rates, memory utilisation and CPU metrics should be collected, and where appropriate, connection table statistics. On occasions, congestion rate collections can also be useful. Filtering these collections to review by type or device can facilitate their correlation.

Threshold monitoring and alerting of these metrics has the potential to detect viral outbreaks, congestion or failure in components, contributing to the overall integrity and availability of the security strategy. It is suggested that automated alerting of threshold triggers be established for these metrics.

Using simple tools like *MRTG*, the collection of these statistics are not hard to include. The collection data and associated graphs have fixed impact to disk, and a degree of impact to memory and CPU on the host they are run. Impact to system resources will vary according to the number of collections, tools and hosts used.

Histories over the last month, quarter or year can also provide useful information for the security professional. This trending or observing the change in the baseline data helps pinpoint times of abnormal activity and facilitates further analysis or log searching and summarisation. Over time histories help provide inputs to capacity planning cycles and help avoid congestion extremes. Avoiding over utilisation extremes improves the availability and integrity.

Monitoring may not provide direct tangible benefit to or for ‘confidentiality’ or provide benefit to the ‘integrity’ of data. Monitoring of these metrics will provide positive impact to ‘availability’ which in turn contributes to the overall integrity of a brand, organisation, enterprise or business.

Sound logging and monitoring practices for components of the security strategy are essential for the security practice in maintaining the Confidentiality, Integrity and Availability of a service or business.

Reference list and Citations

[1] William, Matt. “NTP Security” August 2002
URL: http://www.giac.org/practical/Matt_Willman_GSEC.doc (1-June-2004)

[2] Addamark Technologies. “Privacy and Information Security Regulatory Compliance Guidelines” October 2003
URL: http://www.mbipr.com/whitepaper1.pdf (1-June-2004)

[3] Keldsen, Dan. “An Oversight Layer for Layers of Defence“ August 2003
URL: http://www.sans.org/rr/papers/index.php?id=1191 (1-June-2004)

[4] Checkpoint “Open Platform Security”
URL: http://www.opsec.com/index.html (30-May-2004)

Checkpoint OPSEC “Certification for pre-socialisation of solutions”
URL: http://www.opsec.com/aboutcert.html (1-June-2004)

[5] COSI “Cisco-centric Open Source Community “
URL: http://cosi-nms.sourceforge.net/alpha-progs.html (30-May-2004)

SWATCH “The active log file monitoring tool”
URL: http://swatch.sourceforge.net/ (1-June-2004)

[6] HP OpenView “Network Node Manager”
URL: http://www.managementsoftware.hp.com/products/nnm/index.html (10-June-2004)

Cabletron (now Aprisma) “Spectrum”
URL: http://www.aprisma.com (7-June-2004)

Statscout Pty Ltd “Blanket network monitoring” 2004.
URL: http://www.statscout.com/products5.html (1-June-2004)

Sitescope “Infrastructure Availability and Performance Monitoring”
URL: http://www.mercury.com/us/products/application-management/foundation/monitors/sitescope/ (1-June-2004)

Opennms “OpenSource Enterprise Network Management project”
URL: http://www.opennms.org/

Oetiker, Tobias. Rand, Dave. (et al) MRTG – Multi Router Traffic Grapher
URL: www.mrtg.com or http://people.ee.ethz.ch/~oetiker/webtools/mrtg/ (1-June-2004)

[7] GadAllah, Seham Mohamed. “The Importance of Logging and Traffic Monitoring for Information Security” December 2003
URL: http://www.sans.org/rr/papers/index.php?id=1379 (1-June-2004)

[8] Somix Technologies Inc “MRTG repository” 2004
URL: http://www.somix.com/support/mrtg_repository.php (1-June-2004)

[9] Muggli, Thomas J. “MRTG Thresholds” December 1999
URL: http://www.cloudnet.com/~tom/mrtg/thresh.html (1-June-2004)

Example Appendix – OIDs and configs

Throughout this paper references are made to *MRTG* script to do the *SNMP* collection. The tool used is irrelevant, the *OID* is important, and so those *OID*s utilised in gathering data for this paper are summarised below. It is assumes that bandwidth utilisation and error rates collections can be established.

The example *MRTG* configuration that follows has basic interface summary, memory, CPU and connection-table graph sections, and are based on;

- a preimiter router, a *Cisco*, LocalRouterName, with version 12 IOS with an ip address of 192.168.1.1 and a community name of YourCommunityNameInHere, with the two interfaces of interest having an ifIndex of 1 and 2
- and a firewall, a *Cisco*PIX, PIX1, with 6.3.3 code, and ip address 10.1.1.1 and same community name place holder as the router.

The router *OID's* are for a *Cisco* but the relevant ones can be found using *snmpwalk* or a similar utility to query your router.

Similarly with the firewall, an *snmpwalk* will usually find the OIDs in question, if *SNMP* is permitted. The ones supplied are for a *Cisco* PIX.

Due to the way Checkpoint Firewall-1 can sit on a variety of platforms and appliances, the *SNMP* OIDs will vary and it's not particle to provide them all here. A quick search will provide various ways to plot metrics for a variety of platforms, e.g. http://www.joerg.cc/cgi-bin/wiki.pl?MyMRTG_Page

Router OIDs

CPU (last second, last 60): 1.3.6.1.4.1.9.2.1.56.0 & 1.3.6.1.4.1.9.2.1.57.0

Memory (free and used): 1.3.6.1.4.1.9.9.48.1.1.1.5.1 & 1.3.6.1.4.1.9.9.48.1.1.1.6.1

BECN and FECN: 1.3.6.1.2.1.10.32.2.1.4.2.16 & 1.3.6.1.2.1.10.32.2.1.5.2.16
(Actual final two digits will vary with DLCI)

Firewall OIDs

Connection counter tables (current and peak used)
.1.3.6.1.4.1.9.9.147.1.2.2.2.1.5.40.6 & .1.3.6.1.4.1.9.9.147.1.2.2.2.1.5.40.7

CPU .1.3.6.1.4.1.9.9.109.1.1.1.1.3.1 & .1.3.6.1.4.1.9.9.109.1.1.1.1.5.1

Memory (free and used): 1.3.6.1.4.1.9.9.48.1.1.1.5.1 & 1.3.6.1.4.1.9.9.48.1.1.1.6.1

Memory: (1550, 256, 80 and 4 byte buffers)
1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.1550.5&1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.1550.8
1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.256.5&1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.256.8
1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.80.5&1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.80.8
1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.4.5&1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.4.8

Example of parts of an MRTG.cfg file

```
Target[LocalRouterName-1]: 1:YourCommunityNameInHere@192.168.1.1:
MaxBytes[LocalRouterName-1]: 12500000
Options[LocalRouterName-1]: growright, bits
WithPeak[LocalRouterName-1]: ym
Title[LocalRouterName-1]: Analysis for FastEthernet0/0
PageTop[LocalRouterName-1]: <H1>Analysis for FastEthernet0/0</H1>
#---------------------------------------------------------
Target[LocalRouterName-2]: 2:YourCommunityNameInHere@192.168.1.1:
MaxBytes[LocalRouterName-2]: 125000
AbsMax[LocalRouterName-2]: 250000
Options[LocalRouterName-2]: growright, bits
WithPeak[LocalRouterName-2]: ym
Title[LocalRouterName-2]: Analysis for Serial1
PageTop[LocalRouterName-2]: <H1>Analysis for Serial1</H1>
#---------------------------------------------------------
Target[LocalRouterName-e1]: ifInErrors.1&ifOutErrors.1:YourCommunityNameInHere@192.168.1.1:
MaxBytes[LocalRouterName-e1]: 12500000
Options[LocalRouterName-e1]: growright, bits
WithPeak[LocalRouterName-e1]: ym
Title[LocalRouterName-e1]: Error   Analysis for FastEthernet0/0
PageTop[LocalRouterName-e1]: <H1>Error rate Analysis for FaEth0/0</H1>
#---------------------------------------------------------
Target[LocalRouterName-e2]: ifInErrors.2&ifOutErrors.2:YourCommunityNameInHere@192.168.1.1:
MaxBytes[LocalRouterName-e2]: 12500000
Options[LocalRouterName-e2]: growright, bits
WithPeak[LocalRouterName-e2]: ym
Title[LocalRouterName-e2]: Error   Analysis for S1
PageTop[LocalRouterName-e2]: <H1>Error rate Analysis for Serial1</H1>
#---------------------------------------------------------
Target[R1-cpu]:
1.3.6.1.4.1.9.2.1.56.0&1.3.6.1.4.1.9.2.1.57.0:YourCommunityNameInHere@192.168.1.1
YLegend[R1-cpu]: CPU Utilization
ShortLegend[R1-cpu]: %
MaxBytes[R1-cpu]: 100
Options[R1-cpu]: growright, nopercent, gauge, unknaszero, integer
Legend1[R1-cpu]: CPU last sec:
Legend2[R1-cpu]: CPU last min:
LegendI[R1-cpu]: CPU:
LegendO[R1-cpu]: .
Title[R1-cpu]: LocalRouterName CPU Utilisation
Colours[R1-cpu]: GRAY#AAAAAA,VIOLET#ff00ff,GREEN#00eb0c,BLUE#0000ff
WithPeak[R1-cpu]: ymw
PageTop[R1-cpu]: <H1>LocalRouterName CPU</H1>
#---------------------------------------------------------------
Target[R1-mem]:
1.3.6.1.4.1.9.9.48.1.1.1.5.1&1.3.6.1.4.1.9.9.48.1.1.1.6.1:YourCommunityNameInHere@192.168.1.1
Options[R1-mem]: growright, nopercent, unknaszero, gauge, integer
WithPeak[R1-mem]: wmyd
Colours[R1-mem]: VIOLET#f000ff,DARK GREEN#006600,PURPLE#AA00ff,RED#ff0000
MaxBytes[R1-mem]: 134217728
Title[R1-mem]: LocalRouterName yMemory
YLegend[R1-mem]: Bytes
ShortLegend[R1-mem]: Bytes
Legend2[R1-mem]: Memory Free
Legend1[R1-mem]: Memory Used
LegendI[R1-mem]:   Used:
LegendO[R1-mem]:   Free:
PageTop[R1-mem]: <H1>Memory in use on LocalRouterName </H1>
```

```
#-------------------------------------------------------------
Target[pix1-in]:   2:YourCommunityName@10.1.1.1
MaxBytes[pix1-in]: 12500000
AbsMax[pix1-in]:   12500000
Title[pix1-in]:    PIX1 Inside interface at Timbuktoo
PageTop[pix1-in]:  <H1>PIX1 Inside interface </H1>
#.............................................................
Target[pix1-out]:   1:YourCommunityName@10.1.1.1
MaxBytes[pix1-out]: 12500000
AbsMax[pix1-out]:   12500000
Title[pix1-out]:    PIX1 Outside interface at Timbuktoo
PageTop[pix1-out]:  <H1>PIX1 Outside interface </H1>
#.............................................................
Target[pix1-ins-ERR]: ifInErrors.2&ifOutErrors.2:YourCommunityName@10.1.1.1
MaxBytes[pix1-ins-ERR]: 6500000
WithPeak[pix1-ins-ERR]: ymw
Title[pix1-ins-ERR]: PIX1 xError rate Inside
ShortLegend[pix1-ins-ERR]: Errors/Min
YLegend[pix1-ins-ERR]: Errors/Min
Legend1[pix1-ins-ERR]: ifInErrors
Legend2[pix1-ins-ERR]: IfOutErrors
Legend3[pix1-ins-ERR]: Maximal 5 Minute IfInErrors
Legend4[pix1-ins-ERR]: Maximal 5 Minute IfOutErrors
LegendI[pix1-ins-ERR]:  In
LegendO[pix1-ins-ERR]:  Out
Options[pix1-ins-ERR]: nopercent, growright, perminute
PageTop[pix1-ins-ERR]: <H1> PIX1 Inside interface error rate</H1>
#.............................................................
Target[pix1-out-ERR]: ifInErrors.1&ifOutErrors.1:YourCommunityName@10.1.1.1
MaxBytes[pix1-out-ERR]: 6500000
WithPeak[pix1-out-ERR]: ymw
Title[pix1-out-ERR]: PIX1 xError rate Outde
ShortLegend[pix1-out-ERR]: Errors/Min
YLegend[pix1-out-ERR]: Errors/Min
Legend1[pix1-out-ERR]: ifInErrors
Legend2[pix1-out-ERR]: IfOutErrors
Legend3[pix1-out-ERR]: Maximal 5 Minute IfInErrors
Legend4[pix1-out-ERR]: Maximal 5 Minute IfOutErrors
LegendI[pix1-out-ERR]:  In
LegendO[pix1-out-ERR]:  Out
Options[pix1-out-ERR]: nopercent, growright, perminute
PageTop[pix1-out-ERR]: <H1> PIX1 Outside interface error rate</H1>
#-------------------------------------------------------------
Target[pix1-mem-4]:
1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.4.5&1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.4.8:YourCommunityName@10.1.1.1
Options[pix1-mem-4]: growright, nopercent, unknaszero, gauge, integer
WithPeak[pix1-mem-4]: wmyd
Colours[pix1-mem-4]: GRAY#AAAAAA,GREEN#00e00c,DARK
GREEN#006600,PURPLE#AA00ff,RED#ff0000,VIOLET#ff00ff
MaxBytes[pix1-mem-4]: 1048576
AbsMax[pix1-mem-4]: 2097152
Title[pix1-mem-4]: PIX1 xMemory - 4 byte buffers
YLegend[pix1-mem-4]: Buffers
ShortLegend[pix1-mem-4]: Buffers
Legend1[pix1-mem-4]: Fewest Number of 4 byte blocks
Legend2[pix1-mem-4]: Current Number of 4 byte blocks
LegendI[pix1-mem-4]:   Fewest;
LegendO[pix1-mem-4]:   Current;:
PageTop[pix1-mem-4]: <H1> 4 byte buffers in use on PIX1 in Timbuktoo </H1>
#
```

```
#------------------------------------------------------------
Target[pix1-mem-80]:
1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.80.5&1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.80.8:YourCommunityName@10.1.
1.1
Options[pix1-mem-80]: growright, nopercent, unknaszero, gauge, integer
WithPeak[pix1-mem-80]: wmyd
Colours[pix1-mem-80]: GRAY#AAAAAA,GREEN#00e00c,DARK
GREEN#006600,PURPLE#AA00ff,RED#ff0000,VIOLET#ff00ff
MaxBytes[pix1-mem-80]: 1048576
AbsMax[pix1-mem-80]: 2097152
Title[pix1-mem-80]: PIX1 xMemory - 80 byte buffers
YLegend[pix1-mem-80]: Buffers
ShortLegend[pix1-mem-80]: Buffers
Legend1[pix1-mem-80]: Fewest Number of 80 byte blocks
Legend2[pix1-mem-80]: Current Number of 80 byte blocks
LegendI[pix1-mem-80]:   Fewest;
LegendO[pix1-mem-80]:   Current;:
PageTop[pix1-mem-80]: <H1> 80 byte buffers in use on PIX1 in Timbuktoo </H1>
#------------------------------------------------------------
Target[pix1-mem-256]:
1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.256.5&1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.256.8:YourCommunityName@10.
1.1.1
Options[pix1-mem-256]: growright, nopercent, unknaszero, gauge, integer
WithPeak[pix1-mem-256]: wmyd
Colours[pix1-mem-256]: GRAY#AFAAAA,GREEN#00e00c,DARK
GREEN#006600,PURPLE#AA00ff,RED#ff0000,VIOLET#ff00ff
MaxBytes[pix1-mem-256]: 6552500
Title[pix1-mem-256]: PIX1 xMemory - 256 byte buffers
YLegend[pix1-mem-256]: Buffers
ShortLegend[pix1-mem-256]: Buffers
Legend1[pix1-mem-256]: Fewest Number of 256 byte blocks
Legend2[pix1-mem-256]: Current Number of 256 byte blocks
LegendI[pix1-mem-256]:   Fewest;
LegendO[pix1-mem-256]:   Current;:
PageTop[pix1-mem-256]: <H1> 256 byte buffers in use on PIX1 in Timbuktoo </H1>
#------------------------------------------------------------
Target[pix1-mem-2]:
1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.1550.5&1.3.6.1.4.1.9.9.147.1.2.2.1.1.4.1550.8:YourCommunityName@
10.1.1.1
Options[pix1-mem-2]: growright, nopercent, unknaszero, gauge, integer
WithPeak[pix1-mem-2]: wmyd
Colours[pix1-mem-2]: GRAY#AAAAAA,GREEN#00e00c,DARK
GREEN#006600,PURPLE#AA00ff,RED#ff0000,VIOLET#ff00ff
MaxBytes[pix1-mem-2]: 6552500
Title[pix1-mem-2]: PIX1 xMemory - 1550 byte buffers
YLegend[pix1-mem-2]: Buffers
ShortLegend[pix1-mem-2]: Buffers
Legend1[pix1-mem-2]: Fewest Number of 1550 byte blocks
Legend2[pix1-mem-2]: Current Number of 1550 byte blocks
LegendI[pix1-mem-2]:   Fewest;
LegendO[pix1-mem-2]:   Current;:
PageTop[pix1-mem-2]: <H1> 1550 byte buffers in use on PIX1 in Timbuktoo </H1>
#------------------------------------------------------------
Target[pix1-mem]:
1.3.6.1.4.1.9.9.48.1.1.1.5.1&1.3.6.1.4.1.9.9.48.1.1.1.6.1:YourCommunityName@10.1.1.1
Options[pix1-mem]: growright, nopercent, unknaszero, gauge, integer
WithPeak[pix1-mem]: wmyd
Colours[pix1-mem]: VIOLET#f000ff,DARK GREEN#006600,PURPLE#AA00ff,RED#ff0000
MaxBytes[pix1-mem]: 134217728
Title[pix1-mem]: PIX1 xMemory
```

```
YLegend[pix1-mem]: Bytes
ShortLegend[pix1-mem]: Bytes
Legend2[pix1-mem]: Memory Free
Legend1[pix1-mem]: Memory Used
LegendI[pix1-mem]:   Used:
LegendO[pix1-mem]:   Free:
PageTop[pix1-mem]: <H1>Memory in use on PIX1 in Timbuktoo </H1>
#---------------------------------------------------------------
Target[pix1-
cpu]: .1.3.6.1.4.1.9.9.109.1.1.1.1.3.1&.1.3.6.1.4.1.9.9.109.1.1.1.1.5.1:YourCommunityName@10.1.1.1
YLegend[pix1-cpu]: CPU Utilization
ShortLegend[pix1-cpu]: %
MaxBytes[pix1-cpu]: 100
Options[pix1-cpu]: growright, nopercent, gauge, unknaszero, integer
Legend1[pix1-cpu]: CPU Utilization
Legend2[pix1-cpu]: .
Legend3[pix1-cpu]: Max value per interval on graph
Legend4[pix1-cpu]: .
LegendI[pix1-cpu]: CPU:
LegendO[pix1-cpu]: .
Title[pix1-cpu]: PIX1 xCPU Utilisation
Colours[pix1-cpu]: GRAY#AAAAAA,VIOLET#ff00ff,GREEN#00eb0c,BLUE#0000ff
WithPeak[pix1-cpu]: ymw
PageTop[pix1-cpu]: <H1>PIX1 CPU</H1>
#---------------------------------------------------------------
Target[pix1-conn-
cou]: .1.3.6.1.4.1.9.9.147.1.2.2.2.1.5.40.6&.1.3.6.1.4.1.9.9.147.1.2.2.2.1.5.40.7:YourCommunityName
@10.1.1.1
Options[pix1-conn-cou]: growright, nopercent, unknaszero, gauge, integer
WithPeak[pix1-conn-cou]: wmyd
Colours[pix1-conn-cou]: BLUE#1000ff,RED#ff0000,GREEN#00eb0c,VIOLET#ff00ff
MaxBytes[pix1-conn-cou]: 65525
Title[pix1-conn-cou]: PIX1 xConnections
YLegend[pix1-conn-cou]: Connections
ShortLegend[pix1-conn-cou]: Connections
Legend1[pix1-conn-cou]: Number of concurrent connections
Legend2[pix1-conn-cou]: Maximun Number of connections peak
Legend3[pix1-conn-cou]: Maximal 5 Minute concurrent connections
Legend4[pix1-conn-cou]: Maximal 5 Minute peak used connections
LegendI[pix1-conn-cou]:   Connections/s:
kMG[pix1-conn-cou]:
LegendO[pix1-conn-cou]:   Peak:
PageTop[pix1-conn-cou]: <H1>Connections in use on PIX1 in Timbuktoo </H1>
#---------------------------------------------------------------
```