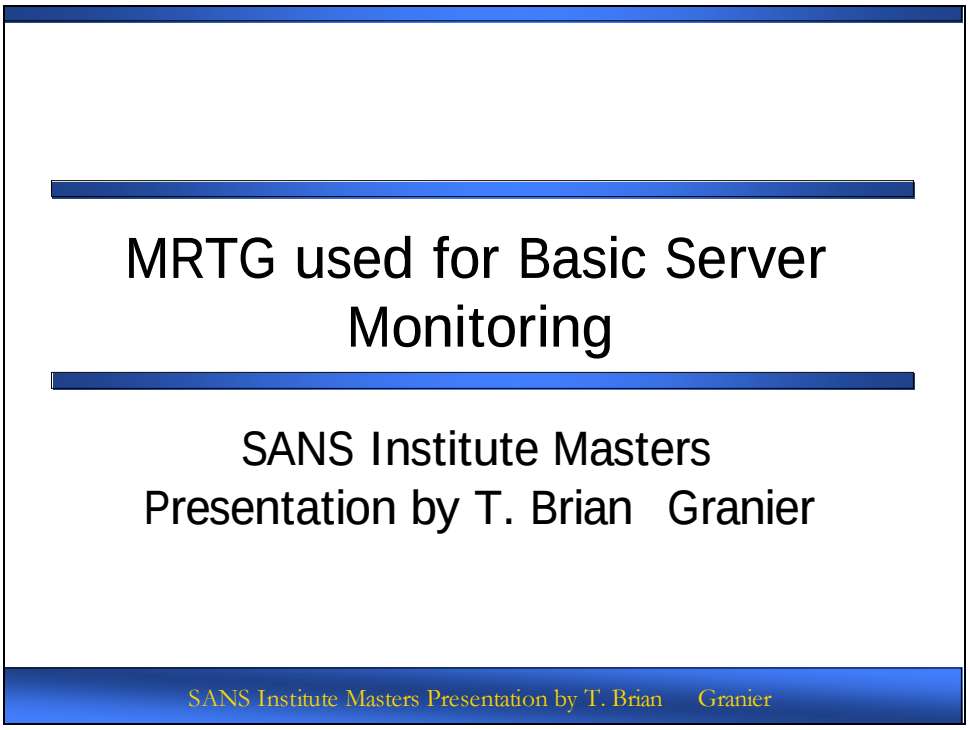
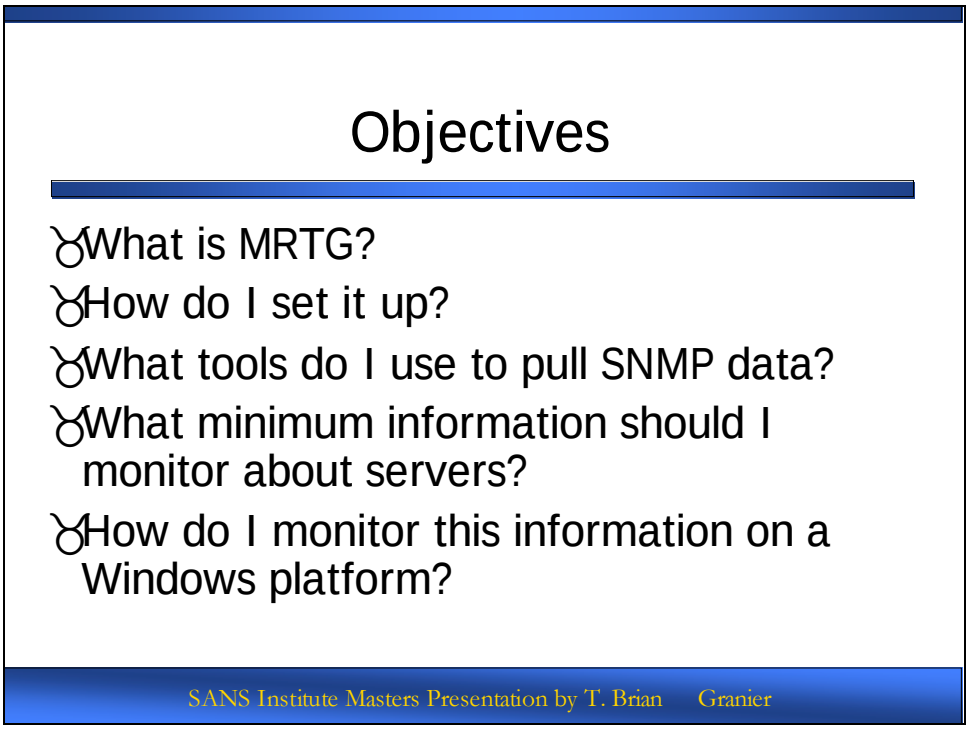


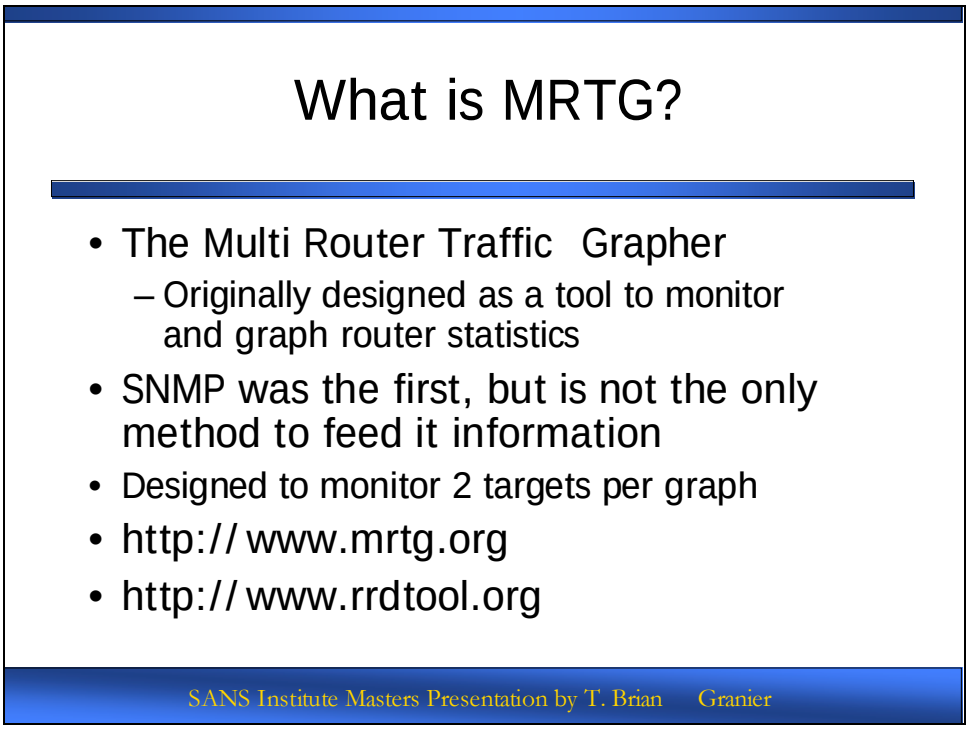
MRTG used for Basic Server Monitoring

This presentation covers how-to instructions to establish basic server monitoring of a server running a Microsoft Windows based operating system using MRTG to gather statistics from the target machine via SNMP. This minimal setup establishes the foundational baseline that is important to be regularly reviewed as part of a proactive strategy to manage the day to day operations of a server environment.



MRTG for Basic Server Monitoring – Outcome Statement

The Student Will understand what general steps are required to setup MRTG, how to configure SNMP on a Windows Server, how to setup SNMP-Informant, and how to use snmpwalk and snmpget in order to complete templates for monitoring disk, cpu, memory and network utilization against a Windows server using MRTG.

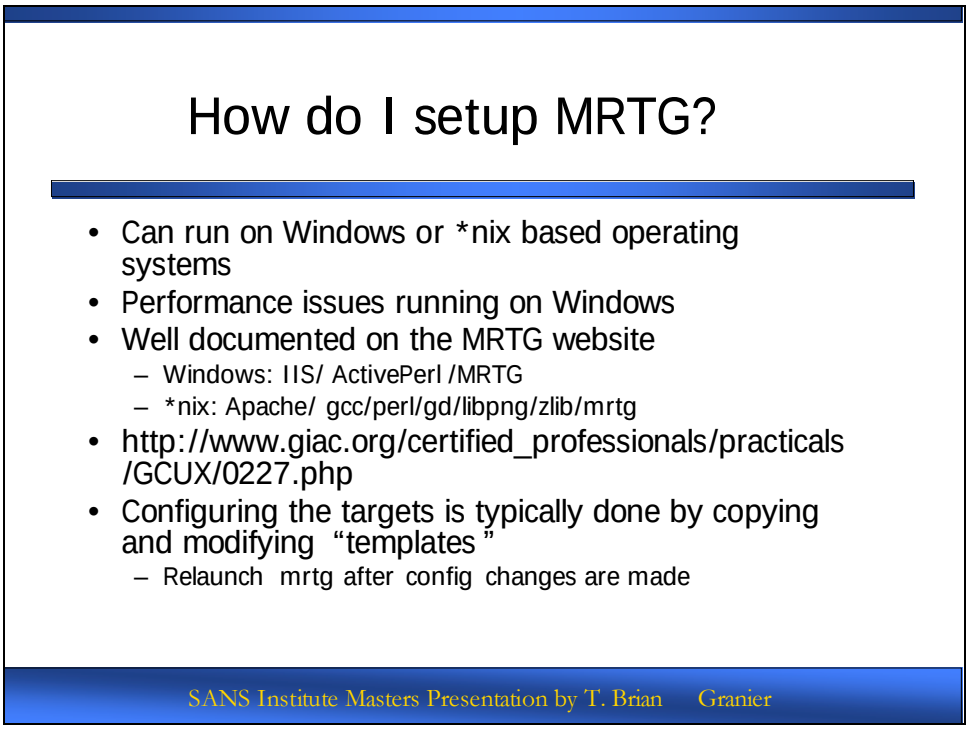


What is MRTG?

Multi Router Traffic Grapher (MRTG) is a tool that was created by Tobias Oetiker. The original purpose of this tool was to provide a convenient method to gather statistics from routers via SNMP and present them in an easy to understand graphical format through a web server. However, the tool quickly expanded to be a general purpose tool by which any information that can be represented in a machine readable numerical format can be captured and displayed. This works through traditional SNMP data gathering techniques, but also through custom scripts that can be created to present the information to MRTG in a manner it can understand, record and display.

After some time, MRTG began to demonstrate some operational issues with respect to inefficient methods of creating the graphs and having no theoretical cap on the size of the log files. To address these concerns, and many others, Tobias designed and built a tool that works in conjunction with MRTG called RRDTool. This set of code allows for on demand image creation and a fixed size logfile to help control disk space utilization by the application.

More details about MRTG and RRDTool can be found at http://www.mrtg.org and http://www.rrdtool.org respectively.



How do I setup MRTG?

Before we can begin setting up server elements to be tracked with MRTG, the base MRTG server must be up and running. It is possible to get MRTG up and functioning on both *nix based systems and on Windows based platforms. However, experience has taught me that you quickly run into performance based problems when you start to heavily use MRTG from a Windows based platform. For that reason, it is strongly recommended to run it from a *nix based platform. The remainder of the how-to steps in this presentation will assume that this was the chosen method.

The fundamental steps for installation are identified in the slide above, but more thorough details are available off of the main MRTG/RRDTool sites that give highly detailed step by step instructions. Furthermore, a somewhat outdated, but nevertheless even more thorough set of instructions on how to setup an MRTG server from scratch, including OS installation and configuration changes made with security in mind is available at http://www.giac.org/certified_professionals/practicals/GCUX/0227.php.

After MRTG has been installed and setup, the remainder of the effort involves setting up systems to be monitored and modifying configuration files. As changes are made to configuration files, MRTG will need to be stopped and restarted to read the new settings. For purpose of this presentation, it will be assumed that MRTG is being run in daemon mode. Given that expectation, the following instructions give a basic understanding of how to halt and re-launch MRTG:

more /var/mrtg/config/<configfilename>.pid

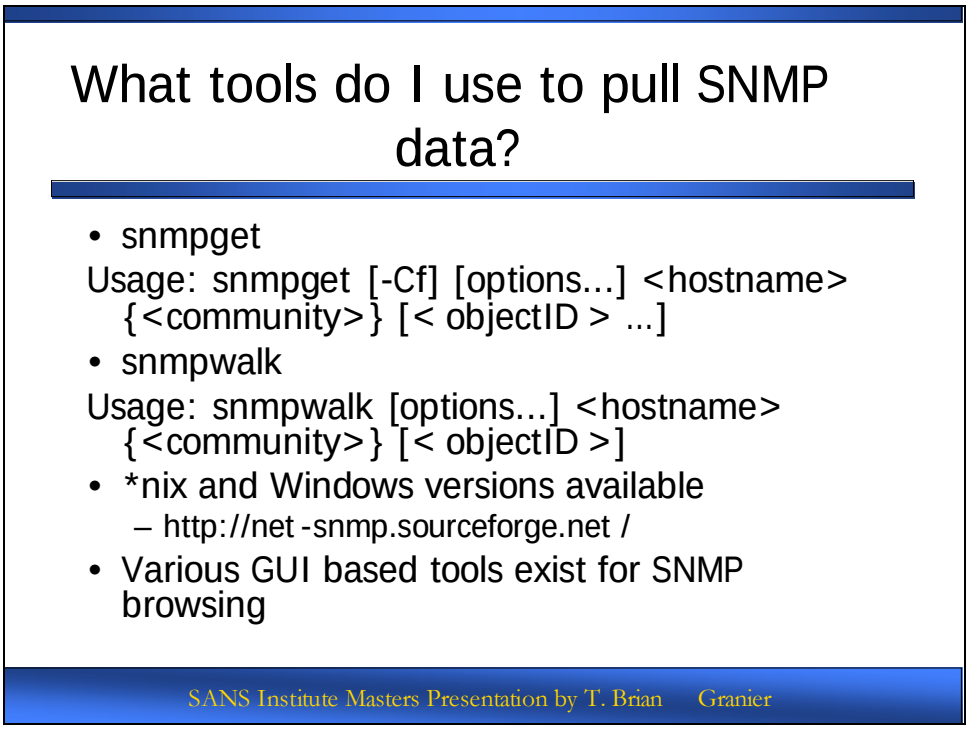
This will provide a process id that the current process is running by. Next, issue the command:

kill -9 #####;perl /usr/local/mrtg-2/bin/mrtg /var/mrtg/config/<configfilename>.cfg

needs to be replaced with the number shown from the "more" command above.

If the kill works, you will both kill and restart the application and receive the following message:

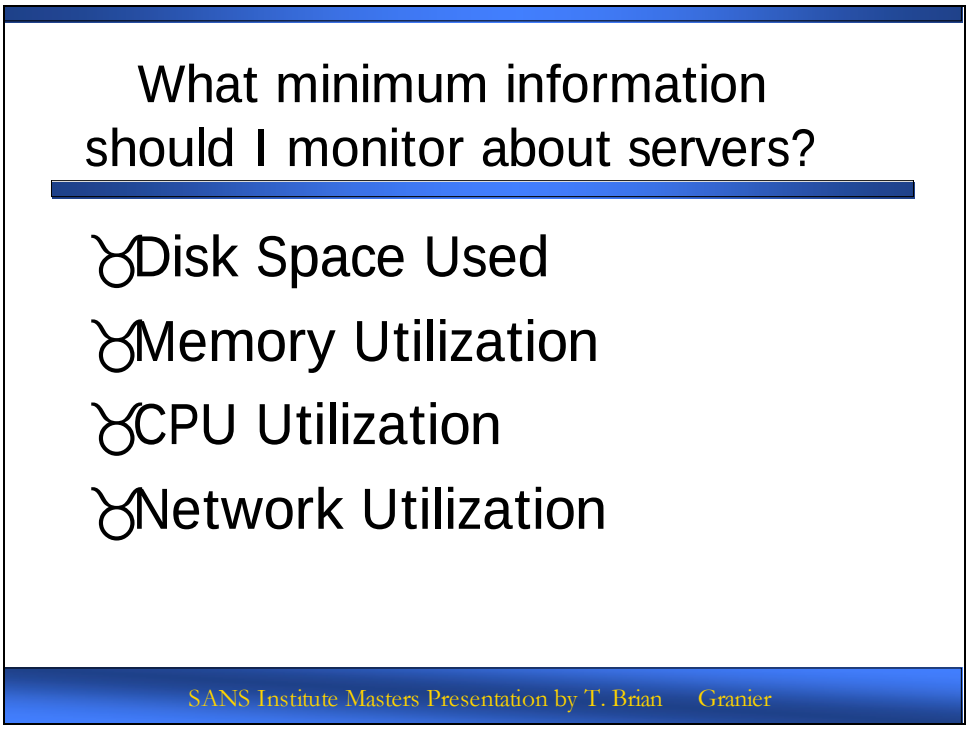
Daemonizing MRTG …



What tools do I use to pull SNMP data?

Snmpget and snmpwalk are the most direct ways to query snmp data from a command line. There are Unix and Windows versions of this tool available. However, there are slight differences between implementations and versions such that the results will report slightly different or the command line will vary between versions. Snmpget is typically used when you know the full OID of the data you want to query. Snmpwalk is useful when you want to view the OID tree beginning at a previously determined value. This is especially useful in learning what values are available after a certain point in the OID structure if you don't know the complete OID value for the data you want. We will be using both of these tools to gather the necessary information for the MRTG configuration file in this presentation.

For those who are more graphically inclined, there are also numerous GUI based Windows applications that will make it easier to query SNMP data.



What minimum information should I monitor about servers?

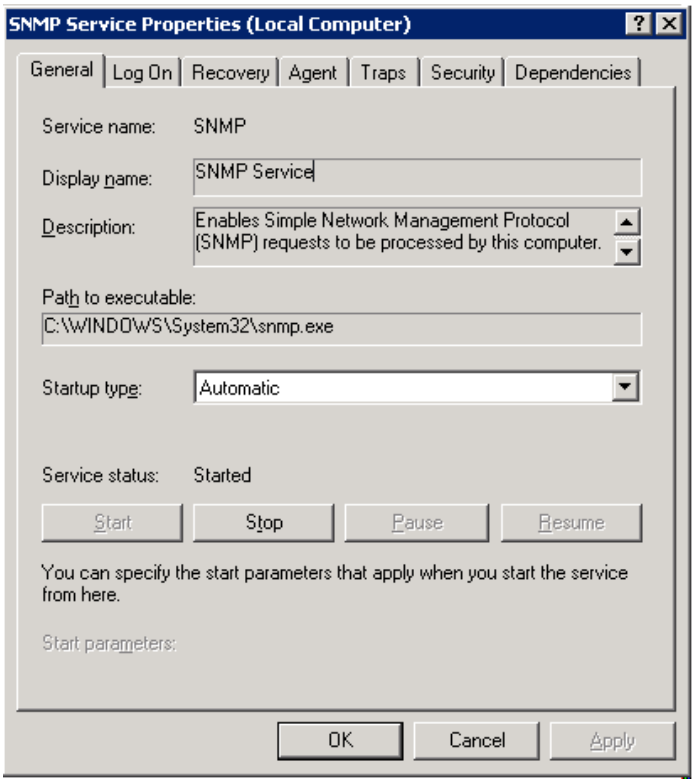
The minimum information that should be monitored on most servers are disk space usage, memory utilization, cpu utilization and network utilization. There are many more useful statistics that might apply to a specific server or environment, but from an operational perspective, these minimum details are going to be useful for day to day utilization and help establish a baseline to understand the normal operating parameters of the systems we are responsible for managing and to provide early warning indicators for when a system is going to exceed its hardware capacity.

SNMP-Informant

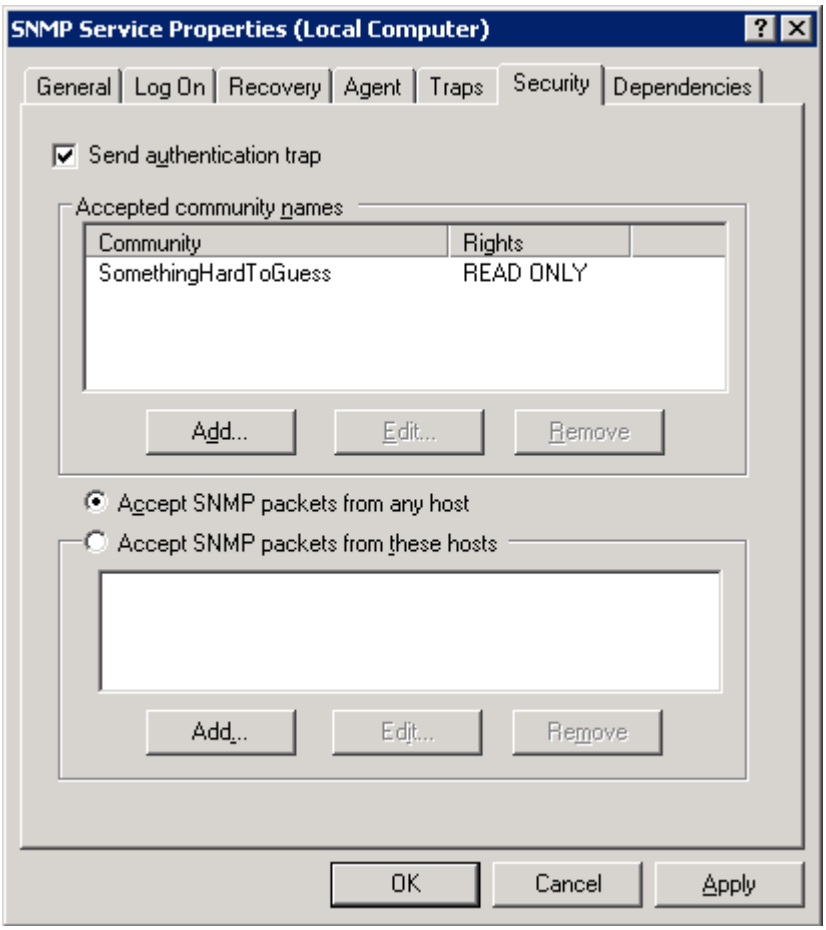
- Configure the Windows SNMP Service
- Windows built -in SNMP functionality has been historically problematic for MRTG
  - SNMP-Informant resolves this issue by extending the operating systems SNMP functionality
- "Stationary" OIDs
- FREE for the basic agent
  - Advanced agents, for a price, gives access to even more information
- http://www.snmp-informant.com /

SNMP-Informant

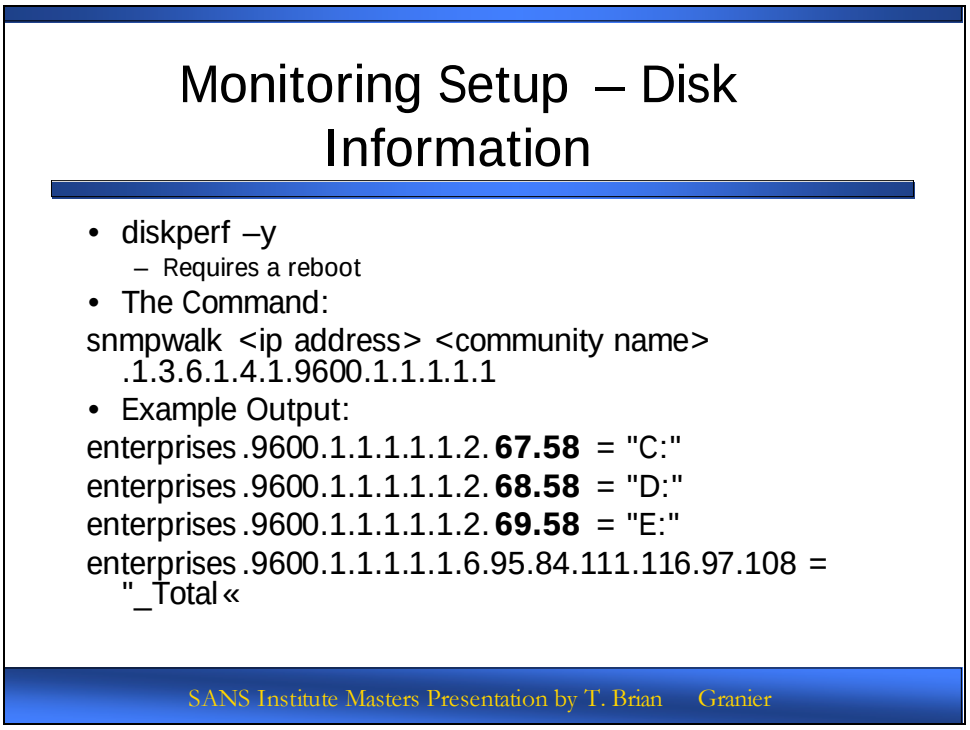
In order to begin querying a Windows server via SNMP to gather useful statistics, the SNMP service needs to be turned on and properly configured. To do this, open the "Services" administrative tool and locate the SNMP Server. Ensure that the service is set to automatically start.



Check the security tab and ensure that there is a Read only (at minimum) community name that has been created for the server. Finally, either select the “Accept SNMP packets from any host” option or ensure that authorized systems from which SNMP queries will come from are entered in the box beneath “Accept SNMP packets from these hosts”.



While you can begin querying some basic data through SNMP once this step has been done, historically there has been an issue where the specific OIDs that are important to query against a Microsoft system for hardware details changes due to reboots, patch levels or other seemingly arbitrary events. To counteract this problem, the basic version of snmp-informant (free) needs to be installed. By doing this, we ensure that the OIDs that we will be monitoring with MRTG via SNMP will remain constant. SNMP-Informant can be downloaded from http://www.snmp-informant.com.



Monitoring Setup – Disk Information

Now we will begin demonstrating specific templates and how to complete them. The template on this slide covers disk utilization. Before completing the template, some setup has to be done on the system to enable disk utilization statistics and some information needs to be obtained from the system using snmpwalk.

We will want to be able to get disk utilization statistics from the system. In order to do so, the "diskperf -y" command needs to be issued from a command line. The system will need to be rebooted after this command has been issued in order to take effect. Unfortunately, SNMP Informant is sometimes unable to register the fact that the command has been issued upon initial install. If the snmpwalk command returns no information, then issue the command diskperf -n and reboot, then diskperf -y and reboot.

The following snmpwalk command will be used to gather the necessary information to complete the provided template:

snmpwalk <ip address> <community name> .1.3.6.1.4.1.9600.1.1.1.1.1

In normal circumstances, you should be returned a list of drive letters. Here is an example output:

enterprises.9600.1.1.1.1.1.2.**67.58** = "C:"
enterprises.9600.1.1.1.1.1.2.**68.58** = "D:"
enterprises.9600.1.1.1.1.1.2.**69.58** = "E:"
enterprises.9600.1.1.1.1.1.6.95.84.111.116.97.108 = "_Total«

In this case, this system has three physical hard drives with volume letters of C, D and E. These will all be monitored. However, recall that MRTG is designed to monitor two data-points per each target configuration. To address this issue, be aware that the template will be used twice, with the first completion graphics drives C and D and the second template graphing drive E twice.

Here is the template that we will be using for Disk utilization:

Target[DSK#<host>]: 100 -
.1.3.6.1.4.1.9600.1.1.1.1.5.2.??.??&.1.3.6.1.4.1.9600.1.1.1.1.5.2.??.??:<SNMP Community>@<ip address>
MaxBytes[DSK#<host>]: 100
Options[DSK#<host>]: growright, gauge, nopercent
YLegend[DSK#<host>]: <host> Disk usage
ShortLegend[DSK#<host>]: %
Legend1[DSK#<host>]: ! Drive % Used
Legend2[DSK#<host>]: ! Drive % Used
Legend3[DSK#<host>]:
Legend4[DSK#<host>]:
LegendI[DSK#<host>]: ! Drive
LegendO[DSK#<host>]: ! Drive
Title[DSK#<host>]: <host> Disk Utilization
PageTop[DSK#<host>]: <H1><host> Disk Utilization</H1>

Modify the template as follows based upon the output from the snmpwalk command:

- Replace ?.? with the last two numbers from the lines shown from the snmpwalk.
- Replace the # sign as follows:
    - # is not present for the first config file set
    - # is 2 for the second (for 3 or more disk drives)
    - # is 3 for the third (for 5 or more disk drives)
    - Etc…
- <hostid> will match the host id of the system
- Replace <host> with the host name. Replace ! with the appropriate drive letters from the snmpwalk command.
- Replace <SNMP Community> with the appropriate community name
- Replace <ip address> with the appropriate IP address.

In order to illustrate what is being explained here, let's review the example output from snmpwalk above:

Example:

#snmpwalk 1.1.1.1 somecommunity .1.3.6.1.4.1.9600.1.1.1.1.1

enterprises.9600.1.1.1.1.1.2.**67.58** = "C:"
enterprises.9600.1.1.1.1.1.2.**68.58** = "D:"
enterprises.9600.1.1.1.1.1.2.**69.58** = "E:"
enterprises.9600.1.1.1.1.1.6.95.84.111.116.97.108 = "_Total«

For host ‘WebServer’ the resulting configuration would be:

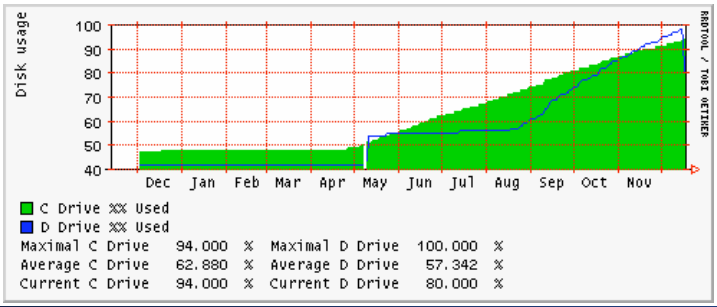
Target[DSKWEBSERVER]: 100 -
1.3.6.1.4.1.9600.1.1.1.1.5.2.67.58&.1.3.6.1.4.1.9600.1.1.1.1.5.2.68.58:somecomm
unity@1.1.1.1
MaxBytes[DSKWEBSERVER]: 100
Options[DSKWEBSERVER]: growright, gauge, nopercent
YLegend[DSKWEBSERVER]: WEBSERVER Disk usage
ShortLegend[DSKWEBSERVER]: %
Legend1[DSKWEBSERVER]: C Drive % Used
Legend2[DSKWEBSERVER]: D Drive % Used
Legend3[DSKWEBSERVER]:
Legend4[DSKWEBSERVER]:
LegendI[DSKWEBSERVER]: C Drive
LegendO[DSKWEBSERVER]: D Drive
Title[DSKWEBSERVER]: WEBSERVER Disk Utilization
PageTop[DSKWEBSERVER]: <H1>WEBSERVER Disk Utilization</H1>

AND

Target[DSK2WEBSERVER]: 100 -
1.3.6.1.4.1.9600.1.1.1.1.5.2.69.58&.1.3.6.1.4.1.9600.1.1.1.1.5.2.69.58:somecomm
unity@1.1.1.1
MaxBytes[DSK2WEBSERVER]: 100
Options[DSK2WEBSERVER]: growright, gauge, nopercent
YLegend[DSK2WEBSERVER]: WEBSERVER Disk usage
ShortLegend[DSK2WEBSERVER]: %
Legend1[DSK2WEBSERVER]: E Drive % Used
Legend2[DSK2WEBSERVER]: E Drive % Used
Legend3[DSK2WEBSERVER]:
Legend4[DSK2WEBSERVER]:
LegendI[DSK2WEBSERVER]: E Drive
LegendO[DSK2WEBSERVER]: E Drive
Title[DSK2WEBSERVER]: WEBSERVER Disk Utilization
PageTop[DSK2WEBSERVER]: <H1>WEBSERVER Disk Utilization</H1>

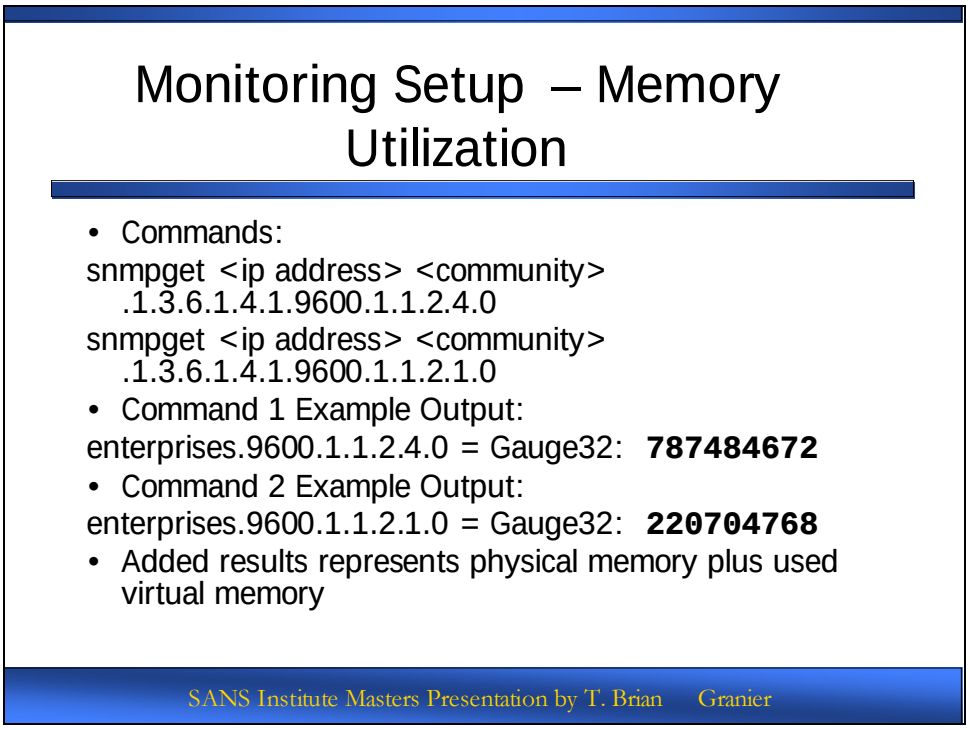
Disk Utilization – What to look for

- The primary purpose of monitoring disk space is to predict when a system will run out of space



Disk Utilization – What to look for

The main reason we will track disk utilization is to identify when a system will be running out of disk capacity. Ideally, a system should generally not exceed 80% utilization. Once it does, plans should be made to either increase the disk capacity or to remove old files that no longer need to be stored on the disk. By monitoring baseline graphs that show disk utilization statistics over a long period of time, it becomes easier to predict when these capacity issues will become an issue and to take action before they affect the operational capability of a system.



Monitoring Setup – Memory Utilization

Completing a template for memory utilization is a bit easier. The only thing that needs to be determined is an estimate of the amount of memory in the system. This is done by issued the following two commands:

snmpget <IP Address> <SNMP Community> .1.3.6.1.4.1.9600.1.1.2.4.0

snmpget <IP Address> <SNMP Community> .1.3.6.1.4.1.9600.1.1.2.1.0

The first value is the used memory (included physical and virtual memory) and the second value is the available memory. Add the two values to get your total…x + y = VALUE.

The following template can then be filled out:

Target[MEM<host>]:
.1.3.6.1.4.1.9600.1.1.2.4.0&.1.3.6.1.4.1.9600.1.1.2.1.0:<SNMP Community>@<IP Address>
MaxBytes[MEM<host>]: <2 x VALUE>
Options[MEM<host>]: growright, gauge, nopercent
YLegend[MEM<host>]: <host> Memory Usage
Legend1[MEM<host>]: Used Mem:
Legend2[MEM<host>]: Free Mem:
Legend3[MEM<host>]:
Legend4[MEM<host>]:
LegendI[MEM<host>]: Used :
LegendO[MEM<host>]: Free :

Title[MEM<host>]: <host> Memory Usage
PageTop[MEM<host>]: <H1><host> Memory Usage</H1>

Modify the template as follows based upon the output from the snmpget commands:

- Replace <ip address> with the appropriate IP address
- Replace <host> with the hostname for the system.
- Replace <2 x value> with the numerical sum of the values obtained from the snmpget commands above multipled by 2.
- Replace <SNMP Community> with the appropriate value.

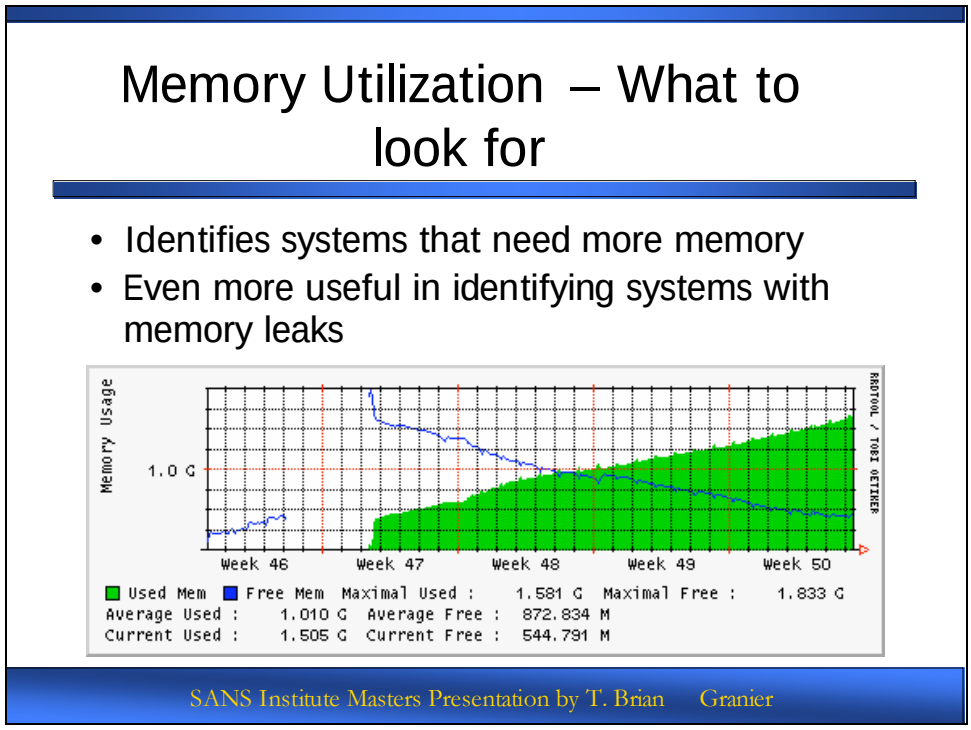
Given the following example output:

#snmpget 1.1.1.1 somecommunity .1.3.6.1.4.1.9600.1.1.2.4.0
enterprises.9600.1.1.2.4.0 = Gauge32: **787484672**

#snmpget 1.1.1.1 somecommunity .1.3.6.1.4.1.9600.1.1.2.1.0
enterprises.9600.1.1.2.1.0 = Gauge32: **220704768**

The template for WebServer would be filled out at follows:

Target[MEMWebServer]:
.1.3.6.1.4.1.9600.1.1.2.4.0&.1.3.6.1.4.1.9600.1.1.2.1.0:somecommunity@1.1.1.1
MaxBytes[MEMWebServer]: 2016378880
Options[MEMWebServer]: growright, gauge, nopercent
YLegend[MEMWebServer]: WebServer Memory Usage
Legend1[MEMWebServer]: Used Mem:
Legend2[MEMWebServer]: Free Mem:
Legend3[MEMWebServer]:
Legend4[MEMWebServer]:
LegendI[MEMWebServer]: Used :
LegendO[MEMWebServer]: Free :
Title[MEMWebServer]: WebServer Memory Usage
PageTop[MEMWebServer]: <H1>WebServer Memory Usage</H1>

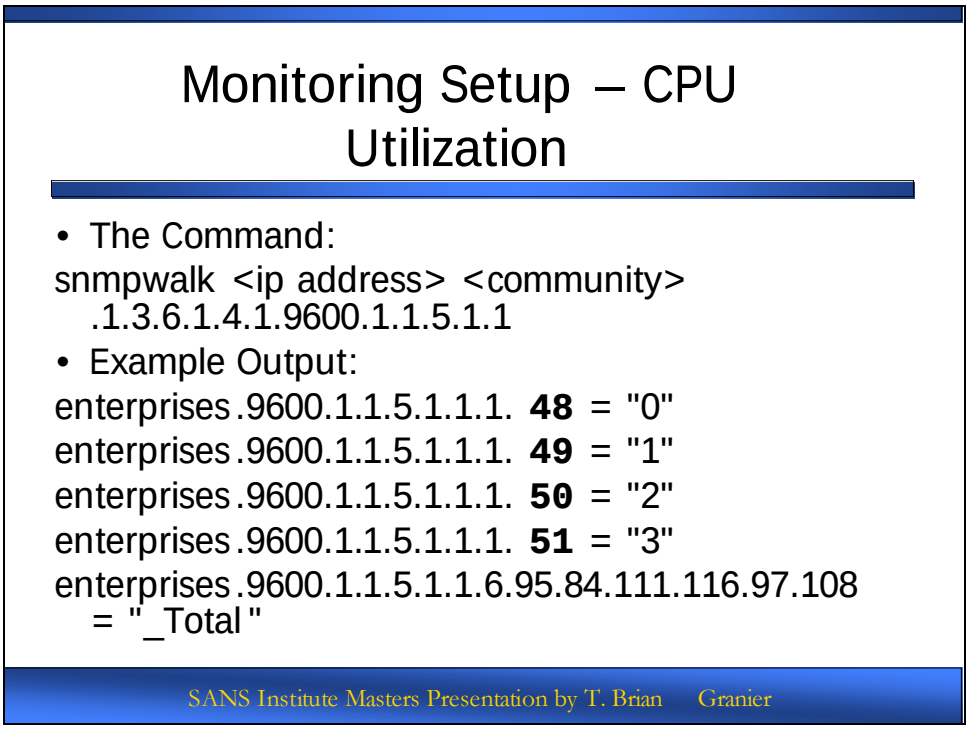


Memory Utilization – What to look for

Reviewing the graphs for memory utilization is very effective in identifying memory leaks, such as the one displayed on the graph in the slide. This information is very useful in maintaining the operational functionality of a system that exhibits characteristics of a memory leak. This issue is typically addressed in one of the following ways:

- Identifying and fixing the code that is the root cause of the memory leak
- Implementing scheduled reboots to "reset" the system and reduce the impact of a memory leak
- Prolonging the amount of time a system remains operationally viable by increasing the amount of memory in a system.

At the same time, this graph provides good information when determining if the memory available in the system is sufficient for its purpose. If the amount of free memory is exceedingly low even after a fresh reboot, the system likely needs more memory.



Monitoring Setup – CPU Utilization

The next template will demonstrate how to use the cpu utilization template. To complete this template, the following command will need to be run:

snmpwalk <IP Address> <SNMP Community> .1.3.6.1.4.1.9600.1.1.5.1.1

We would expect to see something similar to the following as output:

enterprises.9600.1.1.5.1.1.1.**48** = "0"
enterprises.9600.1.1.5.1.1.1.**49** = "1"
enterprises.9600.1.1.5.1.1.1.**50** = "2"
enterprises.9600.1.1.5.1.1.1.**51** = "3"
enterprises.9600.1.1.5.1.1.6.95.84.111.116.97.108 = "_Total"

What can change here is the number of responses and the last digit to the left of the equal sign. Each unique line identifies a single processor. In the case displayed above, there are four processors. Remember that MRTG retrieves information in pairs. If a system has one processor, the same OID will be queried twice. If it has 2 processors, then both OIDs will be used in one config file. If it has 3, then the first config file uses the first two OIDs and the second will gather information for the third OID twice, and so on. This is the same issue as with setting up monitoring for disks.

The CPU Utilization template is as follows:

Target[cpu#<host>]:
.1.3.6.1.4.1.9600.1.1.5.1.5.1.?&.1.3.6.1.4.1.9600.1.1.5.1.5.1.?:<snmp Community>@<IP Address>
MaxBytes[cpu#<host>]: 100

Options[cpu#<host>]: growright, gauge, nopercent
YLegend[cpu#<host>]: <host> CPU Usage
ShortLegend[cpu#<host>]: %
Legend1[cpu#<host>]: CPU ? Utilization
Legend2[cpu#<host>]: CPU ? Utilization
Legend3[cpu#<host>]:
Legend4[cpu#<host>]:
LegendI[cpu#<host>]: CPU ?
LegendO[cpu#<host>]: CPU ?
Title[cpu#<host>]: <host> CPU Utilization
PageTop[cpu#<host>]: <H1><host> CPU Utilization</H1>

Modify the template as follows based upon the output from the snmpwalk command:

- Replace the # sign as follows:
  - # is not present for the first config file set
  - # is 2 for the second (for 3 or more processors)
  - # is 3 for the third (for 5 or more processors)
  - Etc…
- Replace <host> with the host name.
- Replace ? with the appropriate Processor number (+1) from the snmpwalk command.
- Replace <SNMP Community> with the appropriate community name
- Replace <ip address> with the appropriate IP address.

To demonstrate, follow the example output for host ‘WebServer’ below:

#snmpwalk 1.1.1.1 somecommunity .1.3.6.1.4.1.9600.1.1.5.1.1
enterprises.9600.1.1.5.1.1.1.**48** = "0"
enterprises.9600.1.1.5.1.1.1.**49** = "1"
enterprises.9600.1.1.5.1.1.1.**50** = "2"
enterprises.9600.1.1.5.1.1.1.**51** = "3"
enterprises.9600.1.1.5.1.1.6.95.84.111.116.97.108 = "_Total"

So the template would be completed as follows:

Target[cpuWEBSERVER]:
.1.3.6.1.4.1.9600.1.1.5.1.5.1.48&.1.3.6.1.4.1.9600.1.1.5.1.5.1.49:somecommunity
@1.1.1.1
MaxBytes[cpuWEBSERVER]: 100
Options[cpuWEBSERVER]: growright, gauge, nopercent
YLegend[cpuWEBSERVER]: WEBSERVER CPU Usage
ShortLegend[cpuWEBSERVER]: %
Legend1[cpuWEBSERVER]: CPU 1 Utilization

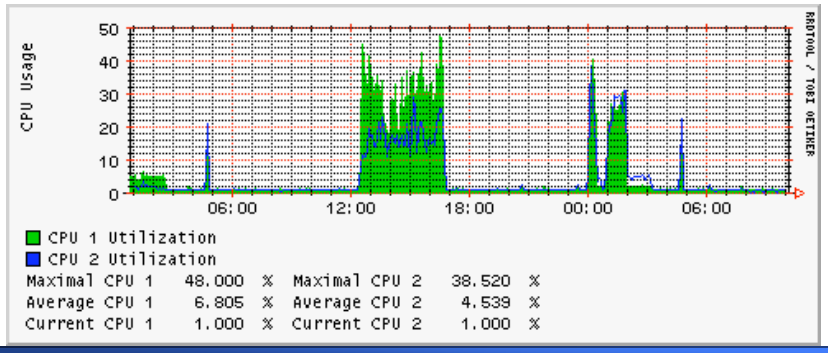
```
Legend2[cpuWEBSERVER]: CPU 2 Utilization
Legend3[cpuWEBSERVER]:
Legend4[cpuWEBSERVER]:
LegendI[cpuWEBSERVER]: CPU 1
LegendO[cpuWEBSERVER]: CPU 2
Title[cpuWEBSERVER]: WEBSERVER CPU Utilization
PageTop[cpuWEBSERVER]: <H1>WEBSERVER CPU Utilization</H1>
```

AND

```
Target[cpu2WEBSERVER]:
.1.3.6.1.4.1.9600.1.1.5.1.5.1.50&.1.3.6.1.4.1.9600.1.1.5.1.5.1.51:somecommunity
@1.1.1.1
MaxBytes[cpu2WEBSERVER]: 100
Options[cpu2WEBSERVER]: growright, gauge, nopercent
YLegend[cpu2WEBSERVER]: WEBSERVER CPU Usage
ShortLegend[cpu2WEBSERVER]: %
Legend1[cpu2WEBSERVER]: CPU 3 Utilization
Legend2[cpu2WEBSERVER]: CPU 4 Utilization
Legend3[cpu2WEBSERVER]:
Legend4[cpu2WEBSERVER]:
LegendI[cpu2WEBSERVER]: CPU 3
LegendO[cpu2WEBSERVER]: CPU 4
Title[cpu2WEBSERVER]: WEBSERVER CPU Utilization
PageTop[cpu2WEBSERVER]: <H1>WEBSERVER CPU Utilization</H1>
```

CPU Utilization - What to look for

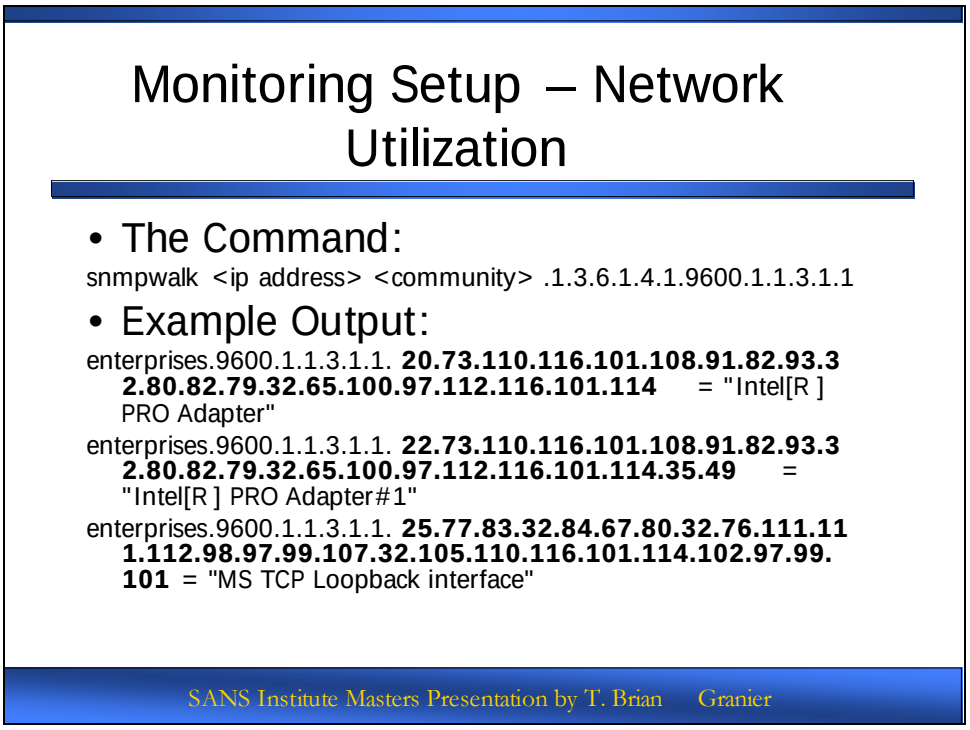
- Determine if more processor power is needed
- Establish processing baseline



SANS Institute Masters Presentation by T. Brian Granier

CPU Utilization – What to look for

When watching CPU utilization, we are mostly going to be looking for periods of time when processor utilization is very high and remains that way. This could indicate a hung application, that the processor is insufficient for the needs of the system or identify periods of time when scheduled tasks might need to be carefully considered to have certain events occur during off peak hours. Ultimately, these graphs help to establish a baseline of normal behavior for the system being monitored. For the graph illustrated in the slide, this happens to be an indication of normal backup behavior that occurs on the target system. This graph was taken from the daily utilization graph for a system on a Sunday morning. On Saturday, at approximately 12:30 PM, a full backup initiated on the target system. This full backup took approximately 4 and ½ hours. The following Sunday morning shortly after midnight, an incremental backup was initiated. The backup failed after approximately 20 minutes. The incremental backup was then restarted and took approximately 1 hour to complete. For this system, this is normal processor utilization behavior based upon the scheduled events that are known to be occurring for the system.



Monitoring Setup – Network Utilization

Identifying the correct OID for network utilization monitoring is fairly tricky to figure out. The information needed can be obtained from issuing the following command:

snmpwalk <ip address> <snmp community> .1.3.6.1.4.1.9600.1.1.3.1.1

You should see output similar to this:

enterprises.9600.1.1.3.1.1.**20.73.110.116.101.108.91.82.93.32.80.82.79.32.65.100.97.112.116.101.114** = "Intel[R] PRO Adapter"
enterprises.9600.1.1.3.1.1.**22.73.110.116.101.108.91.82.93.32.80.82.79.32.65.100.97.112.116.101.114.35.49** = "Intel[R] PRO Adapter#1"
enterprises.9600.1.1.3.1.1.**25.77.83.32.84.67.80.32.76.111.111.112.98.97.99.107.32.105.110.116.101.114.102.97.99.101** = "MS TCP Loopback interface"

Use this output to determine which interface you are going to use. The resulting output will tell you the name and of the actual physical interfaces on this particular server. Thus, it is necessary to get into the server itself via the console or through a Remote Desktop session, and verify the name of the interface that is used in the Network connections properties.

This output shows that there are two physical interfaces available on the server. After getting into the server, verify the name of the interface and use the interface that is associated with the one you wish to monitor. "MS TCP Loopback interface" is not an acceptable interface to use as it is a virtual interface that has no real meaning with respect to monitoring network utilization.

The bandwidth utilization template is as follows:

Target[net<host>]: .1.3.6.1.4.1.9600.1.1.3.1.3.<a long string of numbers>&.1.3.6.1.4.1.9600.1.1.3.1.2.<a long string of numbers>:<snmp community>@<ip address>
MaxBytes[net<host>]: 1000000000
Options[net<host>]: growright, gauge, nopercent
YLegend[net<host>]: <host> Bandwidth Usage
Legend1[net<host>]: Outbound bytes/sec:
Legend2[net<host>]: Inbound bytes/sec:
Legend3[net<host>]:
Legend4[net<host>]:
LegendI[net<host>]: Out :
LegendO[net<host>]: In :
Title[net<host>]: <host> Bandwidth Usage
PageTop[net<host>]: <H1><host> Bandwidth Usage</H1>

Modify the template as follows based upon the output from the snmpwalk command:

- Replace <host> with the host name.
- Replace <a long string of numbers> with string of numbers after "enterprises.9600.1.1.3.1.1." associated with the appropriate interface from the snmpwalk command
- Replace <SNMP Community> with the appropriate community name
- Replace <ip address> with the appropriate IP address.

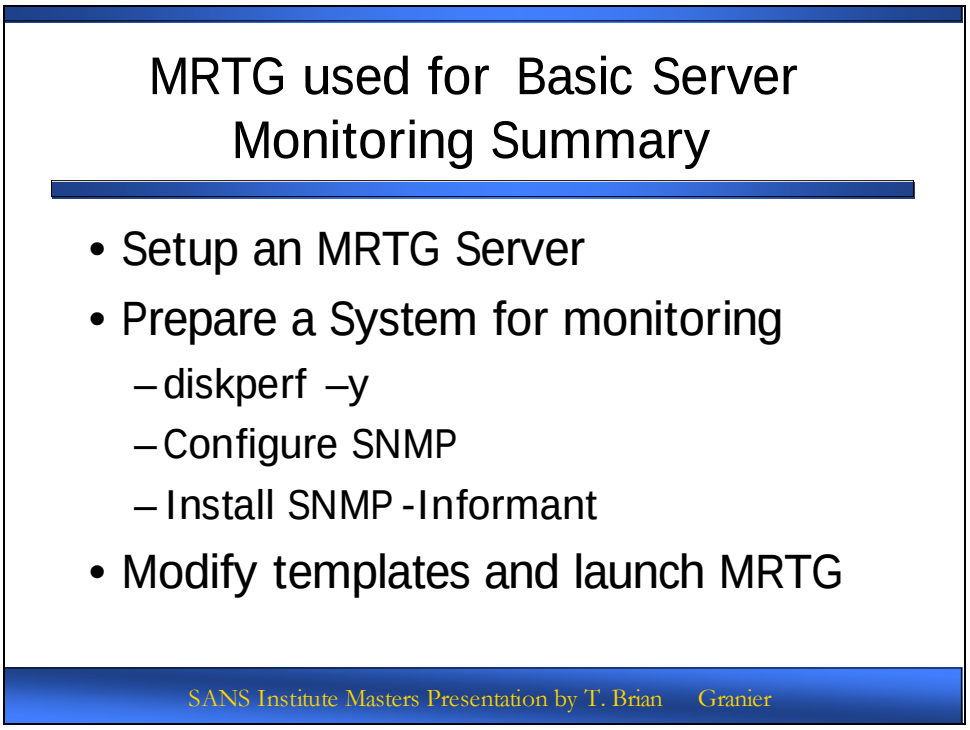
Given the example output for host 'WebServer':

#snmpwalk 1.1.1.1 somecommunity .1.3.6.1.4.1.9600.1.1.3.1.1
enterprises.9600.1.1.3.1.1.**20.73.110.116.101.108.91.82.93.32.80.82.79.32.65.100.97.112.116.101.114** = "Intel[R] PRO Adapter"
enterprises.9600.1.1.3.1.1.**22.73.110.116.101.108.91.82.93.32.80.82.79.32.65.100.97.112.116.101.114.35.49** = "Intel[R] PRO Adapter#1"
enterprises.9600.1.1.3.1.1.**25.77.83.32.84.67.80.32.76.111.111.112.98.97.99.107.32.105.110.116.101.114.102.97.99.101** = "MS TCP Loopback interface"

And given that we know we want the "Intel[R] PRO Adapter] the template would then be filled out like this:

Target[netWEBSERVER]:
.1.3.6.1.4.1.9600.1.1.3.1.3.20.73.110.116.101.108.91.82.93.32.80.82.79.32.65.100.97.112.116.101.114&.1.3.6.1.4.1.9600.1.1.3.1.2.20.73.110.116.101.108.91.82.93.32.80.82.79.32.65.100.97.112.116.101.114:somecommunity@1.1.1.1
MaxBytes[netWEBSERVER]: 1000000000
Options[netWEBSERVER]: growright, gauge, nopercent
YLegend[netWEBSERVER]: WEBSERVER Bandwidth Usage

Legend1[netWEBSERVER]: Outbound bytes/sec:
Legend2[netWEBSERVER]: Inbound bytes/sec:
Legend3[netWEBSERVER]:
Legend4[netWEBSERVER]:
LegendI[netWEBSERVER]: Out :
LegendO[netWEBSERVER]: In :
Title[netWEBSERVER]: WEBSERVER Bandwidth Usage
PageTop[netWEBSERVER]: <H1>WEBSERVER Bandwidth Usage</H1>



MRTG used for Basic Server Monitoring Summary

The goal of this presentation was to provide just enough information that you can begin implementing MRTG to be pointed at your Windows based servers in order to gather the minimal baseline statistics that should be monitored. An overview of the steps required are available on the slide, but this is only a beginning. Use the web sites referenced earlier in this presentation as an invaluable resource to explore the unending possibilities of MRTG and RRDtool in implementing a complete and thorough monitoring infrastructure in your environment.