SNMP Protocol and Nagios Plugins

Wiliam Leibzon
william@leibzon.org

Nagios® World Conference North America

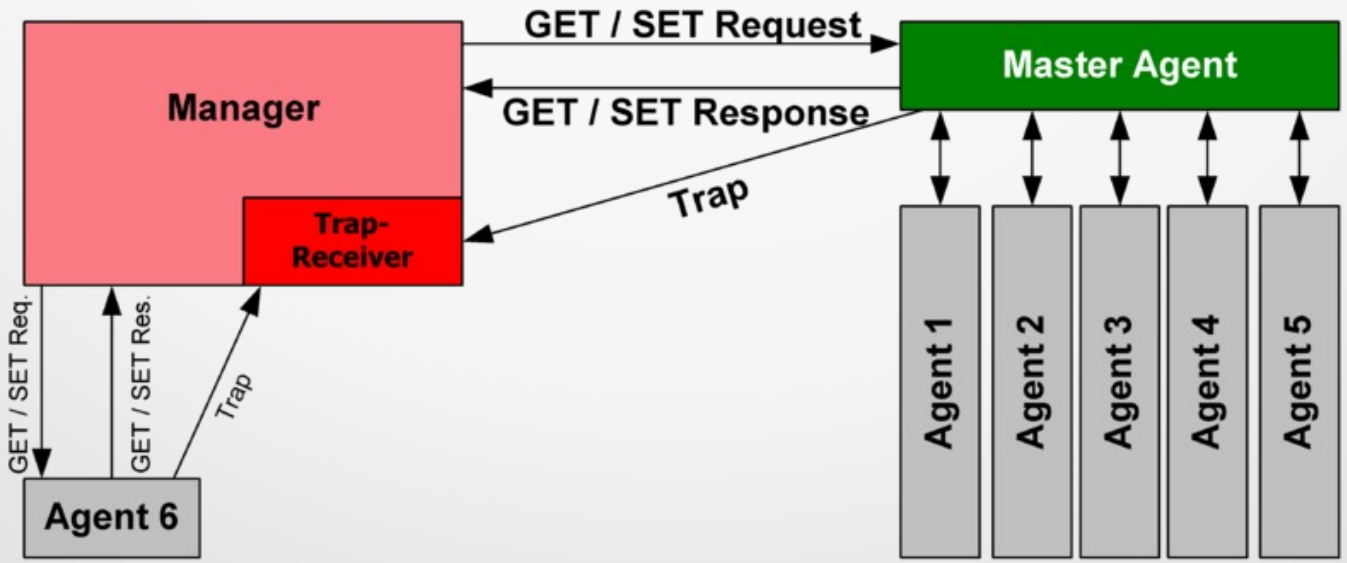
October 1, 2013

Saint Paul, MN

SNMP Overview

SNMP is “Simple Network Management Protocol”
(but its anything but simple....)

- Protocol is designed by IETF to have common means of monitoring and configuring any type of network device.
- In SNMP network systems run snmp agents, which is a software component that answers requests from Network Management System (NMS). Nagios Server servers using SNMP is NMS.
- Agent can also inform NMS of the events using TRAP messages.



SNMP Protocol

- SNMP can actually support more than just TCP/IP (also Appletalk and IPX) but TCP/IP s is the only thing I'll talk about
- With TCP/IP it uses UDP as a transport layer. Port 161 is used for most monitoring and configuration requests. Port 162 is used for Traps.
- There are 3 versions – SNMP v1, SNMP v2 and SNMP v3 which differ in message format, features and authentication
- SNMP is a session-less protocol. Each request is essentially by itself and agents do not keep record of requests. Both request and response messages may get lost in the network. There is no means to re-transmit or retry in the protocol but SNMP v2 and v3 do provide way to confirm receipt of Traps.

SNMP Objects

SNMP is a structured data query protocol

- Data is kept in variables (technically “scalar objects”) which can be one of several protocol defined types.
- Variables are organized into Tables (“table objects”) with each variable being at a certain numeric index of the table.
- Tables are organized into hierarchical tree with each table being a branch and index in its parent table and so forth. This creates an address such as “1.3.6.1.6.3.1.1” which is called “Object Access Identifier” or OID.
- MIB stands for “Management Information Base” and is a logical and administrative grouping of OIDs, typically all OIDs included and branched out of a certain table.
- With MIB definition it is possible to assign a name to each index and each table branch and through that give human readable name to OID. For example: 1.3.6.1.6.3.1.1 = iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObject
- IANA keeps track of the top hierarchy of OID assignments which are delegated to different organizations to manage on their own. Internet is technically under DoD tree which IANA and IETF also manage. Organization make MIBs from their trees available as text files which can be imported into NMS and used to create plugins.

SNMP Object Types

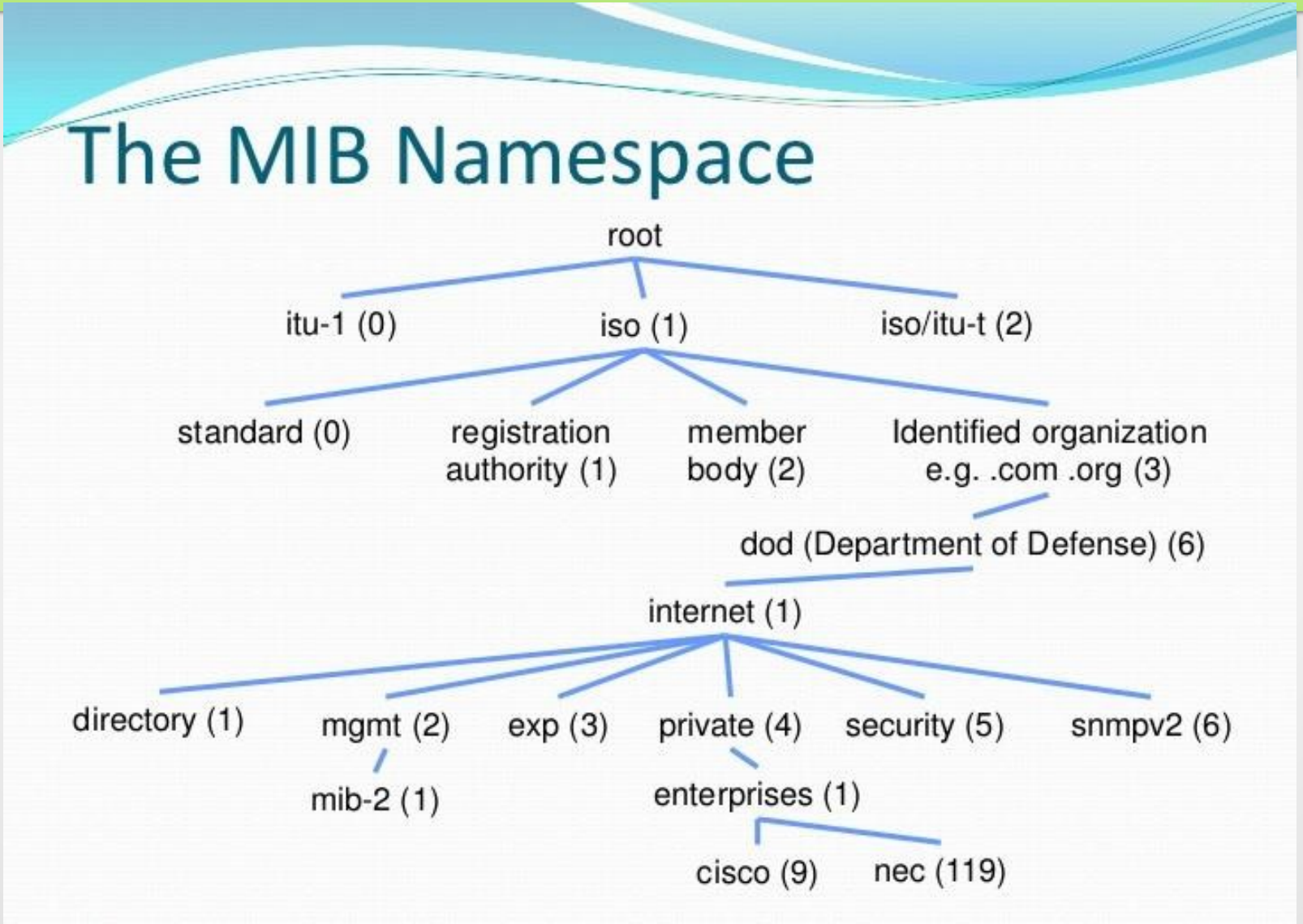
SNMP Objects (Variables) can have the following types:

Table 24.1 *Data Types*

Type	*Size*	*Description*
INTEGER	4 bytes	An integer with a value between -2^{31} and $2^{31}-1$
Integer32	4 bytes	Same as INTEGER
Unsigned32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string up to 65,535 bytes long
OBJECT IDENTIFIER	Variable	An object identifier
IPAddress	4 bytes	An IP address made of four integers
Counter32	4 bytes	An integer whose value can be incremented from zero to 2^{32}; when it reaches its maximum value it wraps back to zero
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter32, but when it reaches its maximum value, it does not wrap; it remains there until it is reset
TimeTicks	4 bytes	A counting value that records time in 1/100ths of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted string

Custom types ENUM types can also be defined based on Integer32

MIB Namespace Tree



SNMP Protocol Data Units

SNMP Protocol supports the following operations (Protocol Data Units = PDUs):

- GetRequest – retrieve data from specified variable or a list of variables
- SetRequest – change value of a variable
- GetNextRequet – Returns a response with variable binding for the lexicographically next variable in the MIB. The entire MIB of an agent can be walked by iterative application of GetNextRequest starting at OID 0.
- GetBulkRequest – Added in SNMP v2. Optimized version of GetNextRequest which returns a response with multiple variable bindings walked from the variable binding in the request.
- Trap - asynchronous notification from agent to manager. Includes current sysUpTime value, an OID identifying the type of trap and optional variable bindings. The format of the trap message was changed in SNMPv2 and the PDU was renamed SNMPv2-Trap.
- InformRequest – Acknowledgement of TRAP or other InformRequest.
- Response – this is what agent sends back as response to *Request PDUs

SNMP versions 1 and 2

There are 3 widely implemented versions of SNMP:

- SNMP v1 (the first version of the protocol)
  - defined in 1988 in RFC1065, RFC1066, RFC1067
  - Authentication mechanism is single password string called “Community” sent clear-text across network
  - Number of other issues with packet format and PDUs
- SNMP v2 (and why you know it as v2c)
  - 1992 IETF effort to fix v2 - RFC1441, RFC1452
  - Added GetBulkRequest and InformRequest PDUs and changed SNMP data packet format. Added 64-bit Counter and BITS variable types.
  - Proposal to add authentication (known as v2u) had no consensus as a result most companies decided to implement non-official “SNMP v2c” which is SNMP v2 but with “Community” password string as in v1

Both v1 and v2c are extremely insecure since password is sent openly across the network. You should never use these across open Internet and preferably not within you local intranet either. v1 is also open to attacks with malformed packets and walking the SNMP tree takes long time.

SNMP version 3

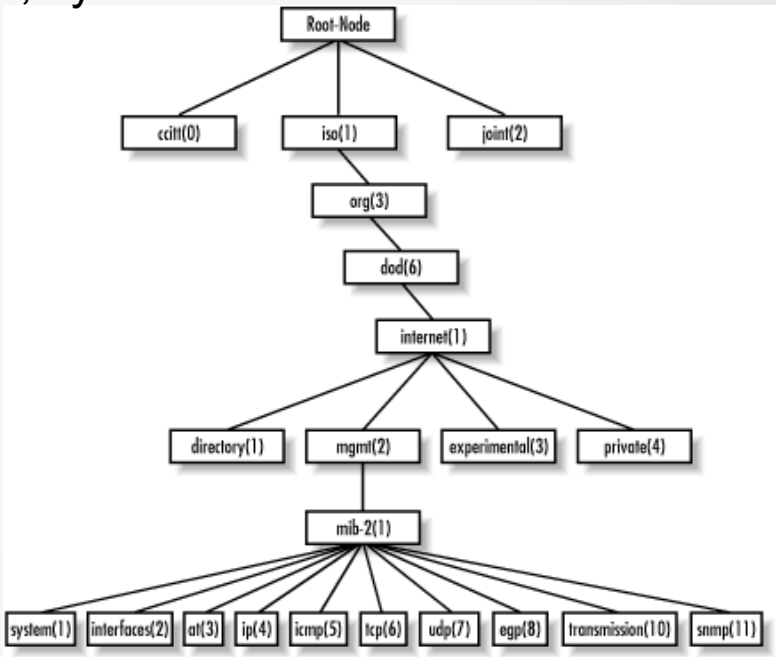
- SNMP v3 (the current IETF standard)
  - Defined in RFC 2271, RFC2272, RFC2273, RFC2274, RFC2275 as published by IETF in January 1998
  - Based on SNMP v2c but added user authentication for security and encryption for message privacy
  - For authentication uses Username and Password (authPass) which is hashed using MD5 or SHA1
  - For message encryption (privacy) can use DES or AES protocols with shared secret key (privPass)
  - Standard MIB for adding users and setting security context (USM table – RFC2574)
  - Can define users with access to certain sections of MIB tree (VASM table - RFC2575)
  - Other improvements like maxMsgSize included in request message so agent knows how large response can be
  - Also other extra complexity like matching EngineID which while it can be learned with requests becomes an issue with traps (i.e. may not be able to send traps unless trap server contacted monitoring device at least once)

For best security use SNMP v3 with SHA1 and AES and both passwords different

MIB-II

With SNMP v2 (in RFC1213) IETF defined standard MIB set for TCP/IP Protocol. These tables are supported by greater majority of systems (unix & windows servers, most network devices) and are known as MIB-2.

- System = 1.3.6.1.2.1.1 - Defines a list of objects that pertain to system operation, such as the snmp system uptime, system contact, and system name.
- Interfaces = 1.3.6.1.2.1.2 – Data on status of each interface and octets sent and received, errors and discards, etc. Updated as IF-MIB in RFC2863.
- IP = 1.3.6.1.2.1.4 - Keeps track of aspects of IP suc as IP routing. Updated as IP-MIB in RFC4293.
- ICMP = 1.3.6.1.2.1.5 - Tracks ICMP errors, discards, etc.
- TCP = 1.3.6.1.2.1.6 – Tracks the state of the TCP connections (closed, listen, synSent, etc.) and other info. Updated as TCP-MIB in RFC4022.
- UDP = 1.3.6.1.2.1.7 - Tracks UDP statistics, datagrams in and out. Updated as UDP-MIB in RFC4113.
- SNMP = 1.3.6.1.2.1.11 - Measures the performance of SNMP implementation and tracks things such as the SNMP packets sent and received. Updated as SNMPv2-MIB in RFC 3418.



Checking MIB-II with Nagios (part 1)

- System: sysDescr (1.3.6.1.2.1.1.1), sysUpTime (1.3.6.1.2.1.1.3),
  sysName (1.3.6.1.2.1.1.5), sysLocation (1.3.6.1.2.1.1.6), etc.
  Plugins: check_uptime.pl (WL-NagiosPlugins), uptime_by_snmp.sh

- Interfaces: ifNumber (1.3.6.1.2.1.2.1), ifDescr (1.3.6.1.2.1.2.2.1.2),
  ifSpeed (1.3.6.1.2.1.2.2.1.5), ifPhysAddress (1.3.6.1.2.1.2.2.1.6),
  ifAdminStatus (1.3.6.1.2.1.2.2.1.7), ifOperStatus (1.3.6.1.2.1.2.2.1.8),
  ifInOctets (1.3.6.1.2.1.2.2.1.10), ifOutOctets (1.3.6.1.2.1.2.2.1.16),
  ifInDiscards (1.3.6.1.2.1.2.2.1.13), ifInErrors (1.3.6.1.2.1.2.2.1.14), etc.
  Plugins: check_netint.pl (WL-NagiosPlugins), check_iftraffic3.pl, check_interface.pl, ...more...

- IP: ipInAddrErrors (1.3.6.1.2.1.4.5), pForwDatagrams (1.3.6.1.2.1.4.6),
  ipInDiscards (1.3.6.1.2.1.4.8), ipOutDiscards (1.3.6.1.2.1.4.11),
  ipInDelivers (1.3.6.1.2.1.4.9), ipOutRequests (1.3.6.1.2.1.4.10),
  ipReasmOKs (1.3.6.1.2.1.4.15), ipReasmFails (1.3.6.1.2.1.4.16),
  ipFragOKs (1.3.6.1.2.1.4.17), ipFragFails (1.3.6.1.2.1.4.18), ipFragCreates, etc
  Routing table: ipRouteIfIndex (1.3.6.1.2.1.4.21.1.2), etc

  Web References: http://www.ietf.org/rfc/rfc1213.txt,
  http://www.alvestrand.no/objectid/1.3.6.1.2.1.html
  http://www.net-snmp.org/docs/mibs/interfaces.html

  Nagios Plugins: http://exchange.nagios.org/directory/Plugins/Network-Protocols/SNMP
  WL-NagiosPlugins: http://github.com/willixix/WL-NagiosPlugins

Checking MIB-II with Nagios (part 2)

- ICMP: icmpInMsgs (1.3.6.1.2.1.5.1), icmpInErrors (1.3.6.1.2.1.5.2), icmpInDestUnreachs (1.3.6.1.2.1.5.3), icmpInTimeExcds (1.3.6.1.2.1.5.4), icmpInRedirects (1.3.6.1.2.1.5.7), icmpInEchos (1.3.6.1.2.1.5.8), icmpInEchoReps (1.3.6.1.2.1.5.9), icmpInTimestamps (1.3.6.1.2.1.5.10), icmpOutMsgs (1.3.6.1.2.1.5.14), icmpOutErrors (1.3.6.1.2.1.5.15), icmpOutRedirects (1.3.6.1.2.1.5.20), icmpOutEchos (1.3.6.1.2.1.5.21), etc

- TCP: tcpRtoAlgorithm (1.3.6.1.2.1.6.1), tcpActiveOpens (1.3.6.1.2.1.6.5), tcpPassiveOpens (1.3.6.1.2.1.6.6), tcpAttemptFails (1.3.6.1.2.1.6.7), tcpEstabResets (1.3.6.1.2.1.6.8), tcpCurrEstab (1.3.6.1.2.1.6.9), tcpInSegs (1.3.6.1.2.1.6.10), tcpOutSegs (1.3.6.1.2.1.6.11), etc.

  Example of Use:

```
define command {
    command_name check_snmp_tcpstats
    command_line $USER1$/check_snmp -l "TCP (ActiveOpens PassiveOpens CurrEstab
InErrs AttemptFails EstabResets RetransSegs)" -H $HOSTADDRESS$ -P 3 -L authPriv -a
SHA -x AES -U $_HOSTSNMP_V3_USER$ -A $_HOSTSNMP_V3_AUTH$ -X
$_HOSTSNMP_V3_PRIV$ -o
1.3.6.1.2.1.6.5.0,1.3.6.1.2.1.6.6.0,1.3.6.1.2.1.6.9.0,1.3.6.1.2.1.6.14.0,1.3.6.1.2.1.6.7.0,1.3.6.1.
2.1.6.8.0,1.3.6.1.2.1.6.12.0
 }
```

  PNP4Nagiostemplate for above at: https://github.com/willixix/WL-NagiosPlugins/blob/master/graphing_templates/pnp4nagios/check_snmp_tcpstats.php

  Web References: http://www.alvestrand.no/objectid/1.3.6.1.2.1.html, or for one document see http://jp.fujitsu.com/platform/server/primergy/products/note/other/NOS_MIB_v211.pdf

HOST-RESOURCES MIB

Another important standard MIB is HOST-RESOURCES-MIB defined in RFC2890 with a base at .1.3.6.1.2.1.25. For info on it see http://net-snmp.sourceforge.net/docs/mibs/host.html

- It has the following scalar objects:

hrSystemUptime(1.3.6.1.2.1.25.1.1) – unlike sysUpTime which is time since SNMPd was started, this is actual host system uptime
Plugins: check_uptime.pl (used when available in preference to sysUpTime)

hrSystemDate(1.3.6.1.2.1.25.1.2) – date on the remote host
hrSystemNumUsers(1.3.6.1.2.1.25.1.5) – number of logged-in users
hrSystemProcesses(1.3.6.1.2.1.25.1.6) – number of currently running processes
hrSystemMaxProcesses (1.3.6.1.2.1.25.1.7) – max number of processes on the server
hrMemorySize (1.3.6.1.2.1.25.2.2) – total RAM on the host in kilobytes

- And it has the following tables:

hrStorageTable (1.3.6.1.2.1.25.2.3) – info on storage devices (disks, partitions), including:
hrStorageDescr – desription, hrStorageSize – size, hrStorageUsed – how much in use
Plugins: check_snmp_storage.pl (http://nagios.manubulon.com/snmp_storage.html)

hrProcessorTable (1.3.6.1.2.1.25.3.3) – info on processors (CPUs) on the system
hrProcessorLoad – avg over 1 min that cpu was not idle
Plugins: check_snmp_load.pl (http://nagios.manubulon.com/snmp_load.html)

hrPrinterTable(1.3.6.1.2.1.25.3.5) – you can use this to tell if printer is out of paper
Plugins: check_snmp_printer (search Nagios Exchange for “SNMP Printer Check”)

hrDiskStorageTable(1.3.6.1.2.1.25.3.7), hrPartitionTable(1.3.6.1.2.1.25.3.7), more...

Reading MIBs

MIBs are provided as text files that contain bunch of entries like this:

```
sysUpTime OBJECT-TYPE
        SYNTAX  TimeTicks
         ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The time (in hundredths of a second) since the network management
portion of the system was last re-initialized."
    ::= { system 3 }
```

So what does it mean? Here is explanation line by line:

sysUpTime OBJECT-TYPE	=>	defines the object called sysUpTime.
SYNTAX TimeTicks	=>	Object type is TimeTicks
ACCESS read-only	=>	This can only be read via SNMP but can not be changed i.e. set-request will not work.
STATUS mandatory	=>	This must be implemented in a SNMP agent.
DESCRIPTION...	=>	Text description of the object. Read this carefully.
::= { system 3 }	=>	The “::=” entry tells how object fits in MIB tree. This says that sysUpTime is at index 3 branched out off of “system” objects table.

Reading MIBs (part 2)

Another type of MIB entry also exist to define new “enum” type assigning special meaning to numerical values and explaining what they are. Here is one defining Nagios exit codes:

```
ServiceStateID ::= TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION "A number that corresponds to the current state of the
                 service: 0=OK, 1=WARNING, 2=CRITICAL, 3=UNKNOWN."
    SYNTAX INTEGER {
        ok(0),
        warning(1),
        critical(2),
        unknown(3)
     }
```

This is coming from Nagios MIB which you can find at:
https://github.com/nagios-plugins/nagios-mib/blob/master/MIB/NAGIOS-ROOT-MIB
Here is how they are used in NAGIOS-NOTIFY-MIB:

```
NAGIOS-NOTIFY-MIB DEFINITIONS ::= BEGIN
  IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,  NOTIFICATION-TYPE, Integer32, Gauge32
      FROM SNMPv2-SMI
    nagios,NotifyType,HostStateID,HostStateType,ServiceStateID
      FROM NAGIOS-ROOT-MIB;
 ....
SvcEventEntry ::= SEQUENCE {
   nSvcEventIndex    Integer32,
   nSvcDesc          OCTET STRING,
    nSvcStateID       ServiceStateID,
```

Net-SNMP

Net-SNMP (homepage: http://www.net-snmp.org/) is an open-source SNMP package often included in linux and unix distributions. If this is not included doing “apt-get install snmp” for Debian-based or “yum install snmp” for RedHat/Suse

The project was previously known as UCD-SNMP. It includes:

- Command-Line Utilities:

  - snmpget, snmpgetnext – these are used for single OID checks

  - snmpbulkget, snmpwalk, snmptable - can walk the tree and retrieve multiple OIDs

  - snmpset - manipulate configuration information on an SNMP-capable device

  - snmpnetstat, snmppdf, snmpstatus - retrieve a fixed collection of information

  - snmptranslate – can translate named OID to numerical

- Graphical MIB browser:
  tkmib – graphical X-Window application written with TK for browsing MIB

- Daemon for receiving SNMP traps and notifications:

  snmptrapd – one of the ways to use is send SNMP traps to syslog and from there check them with Nagios

- SNMP agent server daemon:
  snmpd – snmp daemon that provides MIB-II info and more for various Unix systems

SNMP Security Levels

It is useful to give example with Net-SNMP while also explaining SNMP security levels. These are:

- v1 or v2 Community – clear-text community as the only means of authentication
- v3 noAuthnoPriv - communication without authentication and privacy (still requires known user)
- v3 authNoPriv - communication with authentication and without privacy
- v3 authPriv - communication with both authentication and privacy

- This is SNMP v2 request with community as the only means of authentication

  $ snmpgetnext -v 2c -C public test.net-snmp.org sysUpTime
  system.sysUpTime.0 = Timeticks: (83467101) 9 days, 15:51:11.01

- This is SNMP v3 “public” access which is no authentication, no encryption. Only requires valid user:

  $ snmpgetnext -v 3 -n "" -u noAuthUser -l noAuthNoPriv test.net-snmp.org sysUpTime
  system.sysUpTime.0 = Timeticks: (83467131) 9 days, 15:51:11.31

- This is SNMP v3 authenticated request but no encryption:

  $ snmpgetnext -v 3 -n "" -u MD5User -a MD5 -A "The Net-SNMP Demo Password" -l authNoPriv test.net-snmp.org sysUpTime

  system.sysUpTime.0 = Timeticks: (83491735) 9 days, 15:55:17.35

- And this is both authenticated and encrypted request:

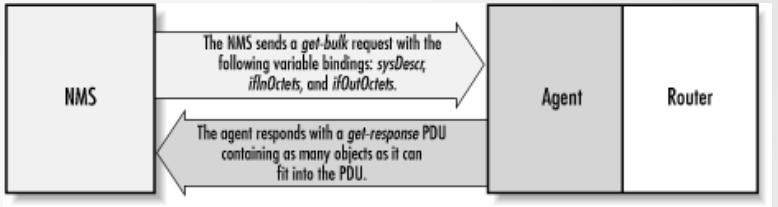
  $snmpgetnext -v 3 -n "" -u MD5DESUser -a MD5 -A "The Net-SNMP Demo Password" -x DES -X "The Net-SNMP Demo Password" -l authPriv test.net-snmp.org system

  system.sysUpTime.0 = Timeticks: (83493111) 9 days, 15:55:31.11

SNMP and bulk requests

- $ snmpbulkget -v2c -B 1 3 linux.ora.com public sysDescr ifInOctets ifOutOctets

  system.sysDescr.0 = "Linux linux 2.2.5-15 #3 Thu May 27 19:33:18 EDT 1999 i686"
  interfaces.ifTable.ifEntry.ifInOctets.1 = 70840
  interfaces.ifTable.ifEntry.ifOutOctets.1 = 70840
  interfaces.ifTable.ifEntry.ifInOctets.2 = 143548020
  interfaces.ifTable.ifEntry.ifOutOctets.2 = 111725152
  interfaces.ifTable.ifEntry.ifInOctets.3 = 0
  interfaces.ifTable.ifEntry.ifOutOctets.3 = 0

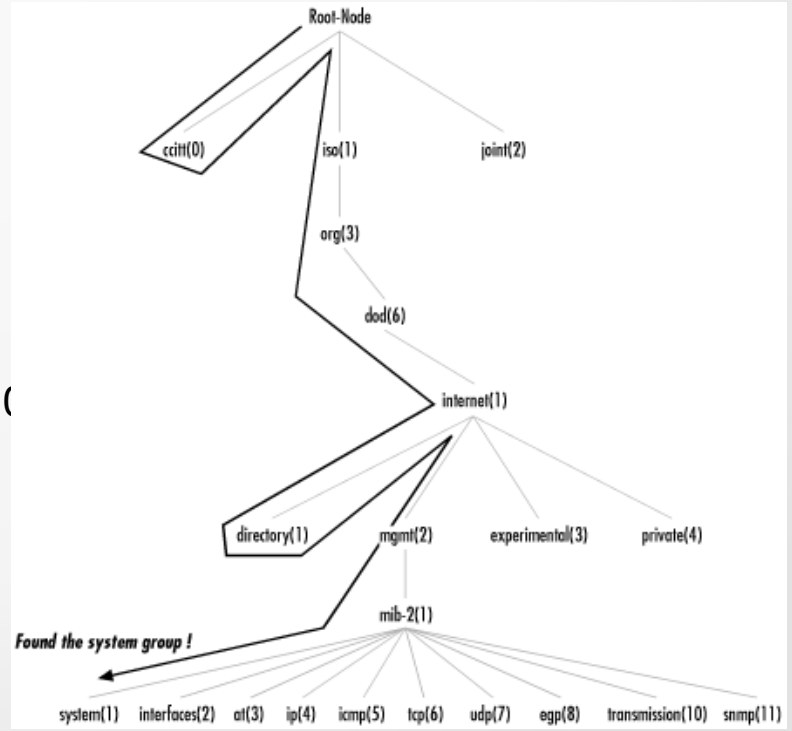


- $ snmpwalk cisco.ora.com public system

  system.ssDescr.0 = "Cisco Internetwork Operating System Software..IOS (tm) 2500 Software (C2500-I-L), Version 11.(5), RELEASE SOFTWARE (fc1).Copyright (c) 1986-1997 by cisco Systems, Inc. Compiled Mon 31-Mar-97 19:53 by ckralik"

  system.sysObjectID.0 = OID: enterprises.9.1.19
  system.sysUpTime.0 = Timeticks: (27210723) 3 days, 3:35:(
  system.sysContact.0 = ""
  system.sysName.0 = "cisco.ora.com"
  system.sysLocation.0 = ""
  system.sysServices.0 = 6

  Note: if you're writing shell plugin that uses Net-SNMP you should use snmpbulkget and NOT snmpwalk



Images courtesy of http://oreilly.com/catalog/esnmp/chapter/ch02.html

SNMP Tables

It is common for SNMP data for a list of components on a system to be located in tables. Instead of defining each OID, MIB defines what data belongs in each table. One of the tables would have names or other identifying id of the component and serve as map. Examples of these are:
network interfaces, temperature sensors, arp entries, list of ip routes, etc

```
$ snmptable -v 2c -c public localhost at.atTable

SNMP table: at.atTable RFC1213-MIB::atTable
atIfIndex  atPhysAddress    atNetAddress

        1  8:0:20:20:0:ab   130.225.243.33

$ snmptable localhost -Cl -CB -Ci -OX -Cb -Cc 16 -Cw 64 ifTable

SNMP table: ifTable

Index        Descr          Type        Mtu         Speed      PhysAddress AdminStatus     OperStatus
LastChange   InOctets       InUcastPkts InNUcastPkt InDiscards InErrors    InUnknownProtos OutOctets
OutUcastPkts OutNUcastPkts  OutDiscards OutErrors   OutQLen    Specific

index: [1]
1            lo             software    16436       10000000                         up          up
?            2837283786     3052466     ?           0          0               ?           2837283786
3052466      ?              0           0           0          zeroDotZero

Index: [2]
2            eth0           ethernet    1500        10000000   0:5:5d:d1:f7:cf up          up
?            2052604234     44252973    ?           0          0               ?           149778187
65897282     ?              0           0           0          zeroDotZero
```

Setting up SNMPd

Common question is:
What do I need to get basic SNMPd working on my Linux/Unix system?

- You need snmpd from Net-SNMP package. If not installing it from source you probably want to look for “snmpd” apt or yum package. You will also need “snmp” package.

- You need to add user for SNMP v3 access (or setup conext for v2). This is where you will need “snmp” package.

- You may need to edit /etc/snmp/snmpd.conf and make sure the server is open on your main network interface as by default its only open on 127.0.0.1

Here are instructions that for Ubuntu systems (remember to us different passwords !):

```
$ apt-get install snmp -y
# below adds read-only SNMP user “snmpuser” with authPriv security level,  SHA1
# auth and AES priv protocols,  authpass “@uthPass” and privpass “pr1vP@ss”
$ net-snmp-config --create-snmpv3-user -ro -a '@uthPass' -x 'pr1vP@ssr' -X AES -A
SHA snmpuser
$ apt-get install snmpd -y
$ vi /etc/snmp/snmpd.conf  # and comment out 'agentAddess udp:127.0.0.1:161'
                           # then uncomment 'agentAddress udp:161,udp6:[::1]:161'
$  vi /etc/default/snmpd   # If there remove 127.0.0.1 from the end of SNMPDOPTS
$ /etc/init.d/snmpd restart
```

Net::SNMP

Net::SNMP is a Perl Library which almost all nagios snmp plugins written in Perl use. Its located at:
http://search.cpan.org/~dtown/Net-SNMP-v6.0.1/lib/Net/SNMP.pm

You can install it as 'cpan Net::SNMP" or search your apt or yum repository for proper package.

This is an object-oriented perl library. It has one constructor returning session object:
Net::SNMP::session(...) - create SNMP v1/v2/v3 session

And a number of methods available for use with $session object:
get_request(-varbindlist=@OID_LIST) - retrieve a list of OIDs
get_table(-baseoid=$OID) - retrieve SNMP Table entries,
this will do either series of get_next_request with snmp v1 or
get_bulk_request with snmp v2 and snmp v3
timeout($seconds) – sets or gets timeout
max_msg_size($msgsize) - set or get msgsize
retries($retries) – sets or gets retries, default is 1 i.e. no retry
error() - returns error from the lat operation
debug() - enables and disabled debugging mode

Net::SNMP – Opening Session

- $session = Net::SNMP->session(..)
- Parameters:

  -version='1|2c|3' (Default is 1 if it is not passed!)

  -hostname='..' (default is localhost)

  -community='..' (only needed for v1 & v2c)

  -username='..' (only v3)

  -authpassword='..' and -authprotocol='md5|sha1' (authPriv or authNoPriv)

  -privpassword='..' and -privprotocol='des|aes' (only authPriv)

```
# open session and return handle to SNNP object
sub create_snmp_session {
  my ($session,$error);
  if ($opt_snmpversion eq '3') {
    if (!defined ($o_privpass)) { # SNMP v3 authNoPriv login
     ($session, $error) = Net::SNMP->session( --version => '3',
      -hostname => $o_host, -port => $o_port, -timeout => $timeout,
      -username => $o_login,-authpassword => $o_passwd,
      -authprotoccol => $o_authproto, );
    } else { # SNMP v3 AuthPriv login
     ($session, $error) = Net::SNMP->session( -version  => '3',
      -hostname => $o_host, -port => $o_port, -timeout => $timeout,
      -username => $o_login, -authpassword => $o_passwd,
      -authprotocol => $o_authproto,
      -privprotocol => $o_privproto, -privpassword => $o_privpass );
  }  }
  elsif ($opt_snmpversion eq '2') { # SNMP v2c Login
    ($session, $error) = Net::SNMP->session( -version => 2,
      -hostname  => $o_host, -port => $o_port, -timeout => $timeout,
      -community => $o_community );
  } else { # SNMPV1 login
    ($session, $error) = Net::SNMP->session(
    -hostname  => $o_host, -port => $o_port, -timeout => $timeout
    -community  => $o_community, );
  }
  if (!defined($session)) {
     printf("ERROR opening session: %s.\n", $error);
     exit $ERRORS{"UNKNOWN"};
  }
  return $session;
 }
```

Net::SNMP - Get_request and Get_table

- Get_Request
  - called as $session->get_request(--varbindlist=>@oid_list)
  - returns hash array with keys being oids in the array and data from SNMP with values that can be numeric or string
  - If request fails returns undef, error in $session->error
  - Example of get_request() based on code in check_uptime.pl (WL-NagiosPlugins):

```
$oid_sysSystem = '1.3.6.1.2.1.1.1.0';
$result = $session->get_request(-varbindlist=>[$oid_sysSystem]);
if (!defined($result)) { printf("ERROR: Problem retrieving $oid_sysSystem : %s", $session->error);
    $session->close(); exit $ERRORS{"UNKNOWN"}; }
verb("Result OID $oid_sysSystem: $result->{$oid_sysSystem}");
```

- Get_Table
  - called as $session->get_table(-baseoid => $table_oid)
  - returns has array with keys being OIDs in the table and values SNMP data
  - Example of get_table() based on code in check_snmp_temperature.pl (WL-NagiosPlugins):

```
verb("Retrieving SNMP table $oid_names to find sensor attribute names");
$result = $session->get_table( -baseoid => $oid_names );
if (!defined($result)) {
   printf("ERROR: Problem retrieving table %s : %s\n", $oid_names, $session->error);
   $session->close(); exit $ERRORS{"UNKNOWN"}; }
foreach $oid (Net::SNMP::oid_lex_sort(keys %{$result})) {
  $line=$result->{$oid};
  verb("got $oid : $line");
  ....
}
```

Optimizing SNMP code in plugins (slide 1)

1. Use numeric OIDs instead of OID names:

   The first thing you can optimize on is to replace named OIDs with numeric OIDs. This helps since named OID requires translation which is done by SNMP library which would read all MIB files in the system, index them and lookup correct name.

2. Retrieving OIDs together rather than individually:

   Many plugins retrieve a number of OIDs with individual get_request. You end up doing separate SNMP requests for each one then. If you know all OIDs then retrieve them together

   However be warned that sometimes doing get_bulk_request would be faster than get_request on extremely long list of OIDs that are all in the same table. I think this is due to agent bugs.

3. Optimize maxMsgSize

   SNMP uses UDP and you want data fit in one packet whenever possible. This can be achieved by setting maxMsgSize to size of packet in your network minus UDP header. 1472 is good number normally but with gigabit ethernet and jumbo frames enabled can set to 8000 or more.

   But don't set maxMsgSize too large causing UDP packet to be fragmented. This can happen if traffic goes through VPN. In that case decrease maxMsgSize to accommodate encapsulation.

   In general its a good idea to have maxMsgSize as parameter to plugin for user. Perl code for setting msgSize:

```
$oct_max=$session->max_msg_size();  verb(" current maxMsgSize: $oct_max");
if (defined($o_octetllength)) { $oct_resultat = $session->max_msg_size($o_octetlength); }
```

Optimizing SNMP code in plugins (slide 2)

4. Do not retrieve full SNMP tables:

Common case is plugin is called with specific name to be retrieved (network interface name, sensor name). This can be done as:

- Pugin retrieves full names table and data using get_table() every time and selects correct id once everything is ready

- Each time plugin first does a lookup in names table (get_table) and then retrieves data OIDs (get_request)

With SNMP its faster to retrieve specific OIDs with get_request than use get_table. So second case above is better than first (2 full tables)

But you can eve optimize out lookup for names table entirely by saving info from first time plugin was called since this isn't going to change. However be warned that this is not trivial and code gets much more complex, best to do this if dealing with multiple tables.

What I did in check_netint.pl (sniplets from it on next slide) is to save info as PERF data and on subsequent calls get PERF data as a parameter and process special perf “cached” perf variable

Optimizing SNMP code in plugins (slide 3)

```
# Load previous performance data
sub process_perf {
   my %pdh; my ($nm,$dt); use Text::ParseWords;
   foreach (quotewords('\s+',1,$_[0])) {
        if (/(.*)=(.*)/) {
           ($nm,$dt)=($1,$2); verb("prev_perf: $nm = $dt");
           $pdh{$nm}=$dt; $pdh{$nm}=$1 if $dt =~ /(\d+)c/; # 'c' is added as designation for octet
   }   }
   return %pdh;
}

# These are sniplets of code from check_netint that have to do with caching of interface name and port speed
my $descr_table = '1.3.6.1.2.1.2.2.1.2';
%prev_perf_data=process_perf($o_prevperf);
@tindex = split(',', prev_perf('cache_descr_ids')) if defined(prev_perf('cache_descr_ids'));
@portspeed = split(',', prev_perf('cache_int_speed')) if defined(prev_perf('cache_int_speed'));
for (my $i=0;$i<scalar(@tindex);$i++) {
   $interfaces[$i]={'descr' => $descr[$i]};
   $interfaces[$i]{'speed'} = $portspeed[$i] if defined(prev_perf('cache_int_speed'));
 }
if (scalar(@tindex)>0) {  verb("Using cached data:"); verb("  tindex=".join(',',@tindex)); … }
if (scalar(@tindex)==0) {
    # snmp_get_table() basically does “return $session->get_table(-baseoid => $descr_table)”
    $result1 = snmp_get_table($session, $descr_table, "Interfaces Description Table");
    foreach my $key (keys %$result1) {
        $data1 = clean_int_name($result1->{$key});
        verb(" OID: $key Clean Desc: '$data1' Raw Desc: ".$result1->{$key});
        if (int_name_match($data1) && $key =~ /$descr_table\.(.*)/) {
              $interfaces[$num_int] = { 'descr' => $data1, };
}   }   }
```

Nagios and SNMP Traps

- Nagios is a monitoring application primarily designed for actively checking and monitoring systems. But Traps are initiated from monitored systems.
- Dealing with them in Nagios requires defining passive checked service and a script that can process the trap message set this service on a proper host into CRITICAL. User action would be required to clear CRITICAL back into OK status on this Passive service

  For setup help read http://xavier.dusart.free.fr/nagios/en/snmptraps.html and http://askaralikhan.blogspot.com/2010/12/receiving-snmp-traps-in-nagios.html and http://www.net-snmp.org/wiki/index.php/TUT:Configuring_snmptrapd_to_parse_MIBS_from_3rd_party_Vendors
- Several nagios addons are available (some required) to help with setting it all up:
  - SNMP Trap Translator (required): http://www.sourceforge.net/projects/snmptt
  - NSTI (Nagios SNMP Trap Interface) – Web Interface for SNMPtt config http://exchange.nagios.org/directory/Addons/SNMP/Nagios-SNMP-Trap-Interface-(NSTI)/details
  - Nagios XI Trap Wizard: http://exchange.nagios.org/directory/Addons/Configuration/Configuration-Wizards/SNMP-Trap-Nagios-XI-Wizard/details
- Another alternative (which I do myself) is set up snmptrapd from Net-SNMP to log traps as syslog message (see snmptrapd manual). Then use check_logfile to check on these. See: http://mathias-kettner.de/checkmk_mkeventd_traps.html

Remote Execution with SNMP

- It is possible to use SNMP (or more specifically snmpd from Net-SNMP) to execute remote programs. See: http://www.net-snmp.org/wiki/index.php/Tut:Extending_snmpd_using_shell_scripts
  - There are two extensions: exec (no formal MIB) in older ucd-snmpd and extend (defined in NET-SNMP-EXTEND-MIB) in newer snmpd versions.
  - This can be used to replace NRPE and works very well for small scripts and plugins that execute fast and are unlikely to fail.
  - However this is known to cause SNMPd to block and even to die entirely if script does not execute fast
- Using with Nagios Plugins

  There are several plugins available for this that allow to get data into nagios from remotely executed plugin. I will use examples with my own check_by_snmp.pl
  - check_by_snmp.pl (WL-NagiosPlugins)
    this is the only plugin that allows both remote execution and cleanly saving remote data into files or passing it as STDIN to other nagios plugins
  - check_snmp_exec.sh / check_snmp_extend.sh (http://www.logix.cz/michal/devel/nagios/)
  - check_snmp_extend.py (https://github.com/nickanderson/check_snmp_extend)

Remote Execution with SNMP - Examples

- DRBD plugin remote executionwith snmp. DRBD plugin from http://exchange.nagios.org/directory/Plugins/Operating-Systems/Linux/check_drbd/details

  - Remote execution of check_drbd directly on a target host system with snmp exec. Add this to /etc/snmp/snmpd.conf:
    exec .1.3.6.1.4.1.2021.202 check_drbd /usr/lib/nagios/plugins/check_drbd-0.5 2 -D All

  - Command definition in nagios: define command {
    command_name check_drbd
    command_line $USER1$/check_by_snmp -S -O 1.3.6.1.4.1.2021.202 -H $HOSTADDRESS$ -L sha,aes -l $_HOSTSNMP_V3_USER$ -x $_HOSTSNMP_V3_AUTH$ -X $_HOSTSNMP_V3_PRIV$
    }

- DRBD plugin executed on Nagios systm using remote data from /proc/drbd retrieved by snmp

  - Here check_drbd is actually executed in Nagios, but it uses data from /proc/drbd onremote system. On remote host setup:
    echo 'exec .1.3.6.1.4.1.2021.202 procdrbd /bin/cat /proc/drbd' >> /etc/snmp/snmpd.conf

  - Command definition in nagios: define command {
    command_name check_drbd
    command_line $USER1$/check_by_snmp -S -O 1.3.6.1.4.1.2021.202 -H $HOSTADDRESS$ -L sha,aes -l $_HOSTSNMP_V3_USER$ -x $_HOSTSNMP_V3_AUTH$ -X $_HOSTSNMP_V3_PRIV$ --exec $USER1$check_drbd-0.5.2 -p - -d All
    }

- check_linux_procstat.pl pugin executed on nagios getting remote data from /proc/stat. Plugin from http://exchange.nagios.org/directory/Plugins/Operating-Systems/Linux/Check-Linux-CPU,-Process-Scheduler-and-I-2FO-Stats--(check_linux_procstat-2Epl)/details (or get it from https://william.leibzon.org/nagios/)

  - Can be used to get very full CPU utilization graph (download pnp plugin). Add the following to /etc/snmp/snmpd.conf:
    extend cpustat /bin/cat /proc/stat

  - Command definition in nagios: define command {
    command_name check_snmp_linuxcpustat
    command_line $USER1$/check_by_snmp.pl -T -E cpustat -H $HOSTADDRESS$ -L sha,aes -l $_HOSTSNMP_V3_USER$ -x $_HOSTSNMP_V3_AUTH$ -X $_HOSTSNMP_V3_PRIV$ --exec $USER1$/check_linux_procstat.pl -P %FILE1% -f -w $ARG1$ -c $ARG2$
    }

Where to read more

- Net-SNMP Tutorials and Documentation
  http://www.net-snmp.org/wiki/index.php/Tutorials,
  http://www.net-snmp.org/docs/man/
- O'Reilly (publicly available book chapters):
  http://oreilly.com/catalog/esnmp/chapter/ch02.html,
  http://oreilly.com/perl/excerpts/system-admin-with-perl/twenty-minute-snmp-tutorial.html
- Net::SNMP Documentation on CPAN
  http://search.cpan.org/~dtown/Net-SNMP-v6.0.1/lib/Net/SNMP.pm
- Nagios SNMP Plugins
  http://exchange.nagios.org/directory/Plugins/Network-Protocols/SNMP,
  https://github.com/willixix/WL-NagiosPlugins,
  http://nagios.manubulon.com/
- MIBs:
  http://net-snmp.sourceforge.net/docs/mibs/
  http://www.oidview.com/mibs/detail.html , http://www.mibdepot.com/ ,
  http://tools.cisco.com/Support/SNMP/do/BrowseOID.do

Questions ?

Questions? Feedback?

William Leibzon <william@leibzon.org>

My Plugins on GitHub:

https://github.com/willixix/WL-NagiosPlugins

My Nagios Page: http://william.leibzon.org/nagios/