

Nagios Installation and Configuration

July 24, 2014

Introduction

Goals

- Optional exercises for Nagios

Notes

- Commands preceded with “\$” imply that you should execute the command as a general user - not as root.
- Commands preceded with “#” imply that you should be working as root.
- Commands with more specific command lines (e.g. “rtrX>” or “mysql>”) imply that you are executing commands on remote equipment, or within another program.

Exercises

PART IX - Optional Exercises

1. Check that nagios is Running

As opposed to just checking that a web server is running on the classroom PCs, you could also check that the nagios3 service is available, by requesting the /nagios3/ path. This means passing extra options to the check_http plugin.

For a description of the available options, type this:

```
# /usr/lib/nagios/plugins/check_http          (short help)
# /usr/lib/nagios/plugins/check_http --help   (detailed help)
```

and of course you can browse the online nagios documentation or google for information on `check_http`. You can even run the plugin by hand to perform a one-shot service check:

```
# /usr/lib/nagios/plugins/check_http -H localhost -u /nagios3/
```

So the goal is to configure nagios to call `check_http` in this way.

```
{hint, /etc/nagios-plugins/config/http.cfg)
```

```
define command{
    command_name    check_http_url
    command_line    /usr/lib/nagios/plugins/check_http -H '$HOSTADDRESS$' -u '$ARG1$'
}
```

```
(hint, /etc/nagios3/conf.d/services_nagios2.cfg_
```

```
define service {
    hostgroup_name    nagios-servers
    service_description    NAGIOS
    check_command    check_http_url!/nagios3/
    use                generic-service
}
```

and of course you'll need to create a hostgroup called `nagios-servers` to link to this service check. (hint, `/etc/nagios3/conf.d/hostgroups_nagios2.cfg`)

Once you have done this, check that Nagios warns you about failing authentication (because it's trying to fetch the page without providing the username/password). There's an extra parameter you can pass to `check_http` to provide that info, so we need to define a new command with an additional argument:

```
define command{
    command_name    check_http_url_auth
    command_line    /usr/lib/nagios/plugins/check_http -H '$HOSTADDRESS$' -u '$ARG1$' -a
```

And you invoke it:

```
check_command    check_http_url_auth!/nagios3!/nagiosadmin:password
```

WARNING: in the tradition of “Debian Knows Best”, their definition of the `check_http` command in `/etc/nagios-plugins/config/http.cfg` is *not* the same as that recommended in the nagios3 documentation. It is missing `ARG1`, so

any parameters to pass to `check_http` are ignored. So you might think you are monitoring `/nagios3/` but actually you are monitoring `root`!

This is why we had to make a new command definition “`check_http_url`”. You could make a more specific one like “`check_nagios`”, or you could modify the Ubuntu `check_http` definition to fit the standard usage.

2. Check that SNMP is running on the classroom NOC

This exercise will not work if you did not complete the installation of additional SNMP MIBs at the start of the week and configure `/etc/snmp/snmp.conf` properly. Please refer to the original `snmp` exercises if you are unsure.

First you will need to add in the appropriate service check for SNMP in the file `/etc/nagios3/conf.d/services_nagios2.cfg`. This is where Nagios is impressive. There are hundreds, if not thousands, of service checks available via the various Nagios sites on the web. You can see what plugins are installed by Ubuntu in the `nagios3` package that we’ve installed by looking in the following directory:

```
# ls /usr/lib/nagios/plugins
```

As you’ll see there is already a `check_snmp` plugin available to us. If you are interested in the options the plugin takes you can execute the plugin from the command line by typing:

```
# /usr/lib/nagios/plugins/check_snmp                (short help)
# /usr/lib/nagios/plugins/check_snmp --help          (detailed help)
```

to see what options are available, etc. You can use the `check_snmp` plugin and Nagios to create very complex or specific system checks.

Now to see all the various service/host checks that have been created using the `check_snmp` plugin you can look in `/etc/nagios-plugins/config/snmp.cfg`. You will see that there are a lot of preconfigured checks using `snmp`, including:

```
snmp_load
snmp_cpustats
snmp_procname
snmp_disk
snmp_mem
snmp_swap
snmp_procs
snmp_users
snmp_mem2
snmp_swap2
```

```

snmp_mem3
snmp_swap3
snmp_disk2
snmp_tcpopen
snmp_tcpstats
snmp_bgpstate
check_netapp_uptime
check_netapp_cupuload
check_netapp_numdisks
check_compaq_thermalCondition

```

And, even better, you can create additional service checks quite easily. For the case of verifying that snmpd (the SNMP service on Linux) is running we need to ask SNMP a question. If we don't get an answer, then Nagios can assume that the SNMP service is down on that host. When you use service checks such as check_http, check_ssh and check_telnet this is what they are doing as well.

In our case, let's create a new service check and call it "check_system". This service check will connect with the specified host, use the private community string we have defined in class and ask a question of snmp on that host - in this case we'll ask about the System Description, or the OID "sysDescr.0" -

To do this start by editing the file /etc/nagios-plugins/config/snmp.cfg:

```
# editor /etc/nagios-plugins/config/snmp.cfg
```

At the top (or the bottom, your choice) add the following entry to the file:

```

# 'check_system' command definition
define command{
    command_name    check_system
    command_line    /usr/lib/nagios/plugins/check_snmp -H '$HOSTADDRESS$' -C '$ARG1$' -o
}

```

COPY and PASTE this. Do not type this by hand and make sure that the command_line line does not wrap.

Note that "command_line" is a single line. If you copy and paste in your editor, the line may not wrap properly and you may have to manually "join" the two lines so they are one.

Now you need to edit the file /etc/nagios3/conf.d/services_nagios2.cfg and add in this service check. We'll run this check against all our servers in the classroom, or the hostgroup "debian-servers"

Edit the file /etc/nagios3/conf.d/services_nagios2.cfg

```
# editor /etc/nagios3/conf.d/services_nagios2.cfg
```

At the bottom of the file add the following definition:

```
# check that snmp is up on all servers
define service {
    hostgroup_name      snmp-servers
    service_description  SNMP
    check_command        check_system!xxxxxx
    use                 generic-service
    notification_interval 0 ; set > 0 if you want to be renotified
}
```

The “xxxxxx” is the community string previously (or to be) defined in class.

Note that we have included our own community string here vs. hard-coding it in the snmp.cfg file earlier. You must change the “xxxxx” to be the snmp community string given in class or this check will not work.

Now we must create the “snmp-servers” group in our hostgroups_nagios2.cfg file. Edit the file /etc/nagios3/conf.d/hostgroups_nagios2.cfg and go to the end of the file. Add in the following hostgroup definition:

```
# A list of snmp-enabled devices on which we wish to run the snmp service check
define hostgroup {
    hostgroup_name  snmp-servers
    alias          snmp servers
    members        noc,localhost,pc1,pc2,pc3,pc4...pc36,rtr1,rtr2,rtr3...rtr9
}
```

Note that for “members” you can add in all PCs and routers as they should all have snmp up and running at this time. Remember to EXCLUDE our pc and use localhost instead.

Now verify that your changes are correct and restart Nagios.

```
# service nagios3 restart
```

**** Defect / Bug in Ubuntu 12.04 LTS ****

The net-snmp 5.6.x package appears to not install one of the IANA mibs (IANAifType-MIB). This causes a MIB error, which, in turn causes the snmp check plugin to fail. To fix this problem do the following (as root):

```
# cd /usr/share/mibs
# wget http://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib
# mv ianaiftype-mib ianaiftype-mib.my
```

And, now you can continue.

If you click on the Service Detail menu choice in web interface you should see the SNMP check appear for the noc host, or for any other hosts you may have included on the “members” line above.

3. Check other settings using SNMP

The real purpose for `check_snmp` is to poll devices for their status. It can be used, for example, to check that power supplies and fans are functioning normally.

In order to do this, you will need to find the OID(s) of interest and the values which you want to be alerted on for warning and critical status.

The following example checks the power supply status of a Netgear 72xx series switch with dual power supplies running 8.x firmware. Nagios doesn't care which file each definition goes in, but some locations are suggested.

```
# This could go in your switches.cfg or in services_nagios2.cfg

define service {
    hostgroup_name      netgear72xx-8x-switches
    service_description  PSUs
    check_command        check_netgear72xx_8x_power_dual!<community>
    use                 generic-service
}

# This could go in /etc/nagios-plugins/config/netgear-8x.cfg

define command{
    command_name      check_netgear72xx_8x_power_dual
    command_line       /usr/lib/nagios/plugins/check_snmp -H '$HOSTADDRESS$' \
        -o .1.3.6.1.4.1.4526.10.43.1.7.1.3.0,.1.3.6.1.4.1.4526.10.43.1.7.1.3.1 \
        -C '$ARG1$' -u 'PSU1,PSU2' -w @5:5,@5:5 -c @2:2,@2:2 -l "PSU status "
}
```

You'd also create a hostgroup “netgear72xx-8x-switches” and make the switches members of this group, so that Nagios runs this check on those devices.

Notice that the `-o` option contains the two OIDs we want to poll, and the `-w` and `-c` options give the values to check for. This makes use of a feature of `check_snmp` that is not well documented:

- `-w <x>:<y>` gives a warning if the value is *not* between x and y
- `-w @<x>:<y>` gives a warning if the value *is* between x and y

The [MIB](#) (fastpath_boxservices.my) contains the following definitions:

```
boxServicesPowSupplyItemState OBJECT-TYPE
    SYNTAX      INTEGER {
                                operational(1),
                                failed(2),
                                powering(3),
                                notpowering(4),
                                notpresent(5)
                        }
    MAX-ACCESS   read-only
    STATUS        current
    DESCRIPTION   "The status of power supply"
    ::= { boxServicesPowSuppliesEntry 3 }
```

Therefore, we get a warning if the status is `notpresent(5)`, and a critical error if the status is `failed(2)`.

Note: `notpowering(4)` means that the PSU is good but the device is being powered by the other PSU. This is not an error.

You should be able to adapt this recipe to other types of equipment, and for checking fan status and temperature, by adjusting the OIDs and values appropriately.

The OID `.1.3.6.1.4.1.4526.10.43.1.7.1.3` comes from:

```
netgear                OBJECT IDENTIFIER ::= { enterprises 4526 }

ng7000managedswitch     OBJECT IDENTIFIER ::= { netgear 10 }

fastPathBoxServices MODULE-IDENTITY
    LAST-UPDATED "200802220000Z" -- 22 Feb 2008 12:00:00 GMT
    ORGANIZATION "Netgear"
    CONTACT-INFO
        "...
    ::= { ng7000managedswitch 43 }

boxServicesGroup        OBJECT IDENTIFIER ::= { fastPathBoxServices 1 }

boxServicesPowSuppliesTable OBJECT-TYPE
    SYNTAX SEQUENCE OF BoxServicesPowSuppliesEntry
    MAX-ACCESS not-accessible
    STATUS        current
    DESCRIPTION   "Power supply"
```

```

 ::= { boxServicesGroup 7 }

boxServicesPowSuppliesEntry OBJECT-TYPE
    SYNTAX      BoxServicesPowSuppliesEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Box Services Power Supply Entry"
    INDEX { boxServicesPowSupplyIndex }
    ::= { boxServicesPowSuppliesTable 1 }

BoxServicesPowSuppliesEntry ::= SEQUENCE {
    boxServicesPowSupplyIndex
        Integer32,
    boxServicesPowSupplyItemType
        INTEGER,
    boxServicesPowSupplyItemState
        INTEGER
}

boxServicesPowSupplyIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Unique index of power supply table entry"
    ::= { boxServicesPowSuppliesEntry 1 }

```

A device with only one power supply connected reports under FASTPATH-BOXSERVICES-PRIVATE-MIB::boxServicesPowSupplyIndex.0 = INTEGER: 0

```

.1.3.6.1.4.1.4526.10.43.1.7.1.1.0 = INTEGER: 0
.1.3.6.1.4.1.4526.10.43.1.7.1.1.1 = INTEGER: 1
.1.3.6.1.4.1.4526.10.43.1.7.1.2.0 = INTEGER: fixed(1)
.1.3.6.1.4.1.4526.10.43.1.7.1.2.1 = INTEGER: removable(2)
.1.3.6.1.4.1.4526.10.43.1.7.1.3.0 = INTEGER: operational(1)
.1.3.6.1.4.1.4526.10.43.1.7.1.3.1 = INTEGER: notpresent(5)

```

or with translation of OIDs:

```

FASTPATH-BOXSERVICES-PRIVATE-MIB::boxServicesPowSupplyIndex.0 = INTEGER: 0
FASTPATH-BOXSERVICES-PRIVATE-MIB::boxServicesPowSupplyIndex.1 = INTEGER: 1
FASTPATH-BOXSERVICES-PRIVATE-MIB::boxServicesPowSupplyItemType.0 = INTEGER: fixed(1)
FASTPATH-BOXSERVICES-PRIVATE-MIB::boxServicesPowSupplyItemType.1 = INTEGER: removable(2)
FASTPATH-BOXSERVICES-PRIVATE-MIB::boxServicesPowSupplyItemState.0 = INTEGER: operational(1)
FASTPATH-BOXSERVICES-PRIVATE-MIB::boxServicesPowSupplyItemState.1 = INTEGER: notpresent(5)

```