NSCA (with Nagios)

Prerequisites

-Nagios should be previously installed and configured
-External commands should be enabled and configured for Nagios previously

Getting the source

The first step would be to get the source code for NSCA. You can get this from any Sourceforge mirror. I will list the mirror that I use but you can find this by searching the Sourceforge website. http://prdownloads.sourceforge.net/nagios/nsca-*version-number*.tar.gz. You can download this either using an internet browser, or if you are using a machine without a GUI or test based browser, you can retrieve the file using the wget command:

```
[root@localhost] wget http://prdownloads.sourceforge.net/nagios/nsca-version-number.tar.gz
```

Unpacking

After you have downloaded the compressed source, you need to unpack it to make it useable. Move the file to a working directory, perhaps your home directory:

```
[root@localhost] tar xzf nsca-version-number.tar.gz
```

The xzf are options for unpacking gzip files. x stands for extract, z for gzip type, and f to specify the filename.

Creating the binaries

To create the binaries for the NSCA add-on you is simple assuming you already have a C compiler such as gcc or preferably g++ installed. First run the configure script located in the base directory $DOWNLOADPATH$/nsca-2.6 with no arguments passed. After the script ends, assuming the computer met all requirements and finished without error, you can then compile with make.

```
[root@localhost]./configure
[root@localhost] make all
```

This should proceed to create two binaries and some daemon scripts. The most important of these for us is the nsca binary.

Installing

After compiling, we will put the compiled binary and the sample nsca config file in their respective nagios locations. This is not necessary, as it can run from any location as long as the config file is correct, but for the sake of clarity we will put the nsca binary in nagios’ bin directory and the config file in nagios’ etc directory.

```
[root@localhost] cp $DOWNLOADPATH$/nsca-version-number/src/ncsa $NAGIOSHOME$/bin
[root@localhost] cp $DOWNLOADPATH$/nsca-version-number/sample_config/nsca.cfg
$NAGIOSHOME$/etc
```

Configuring NSCA

The next step would be to configure the NSCA daemon. The config file is actually very self-explanatory. And if you have followed the documentation on the Nagios website to setup Nagios and external commands, the paths should all be right. Assuming you want all the default options, there is no configuration necessary here. The only options that I set are the server address and debug options. The server address option lets you specific an IP to bind to. This is used when there is more than one network interface card. It allows the daemon to determine which interface it should monitor by choosing the IP address on that network segment. To do this, uncomment the following line and enter that interface’s IP address on the network segment you wish to monitor.

```
server_address=192.168.1.207     # My local IP address
```

If you have only one network interface, then this is not necessary.

Second, the debug option is useful for logs. The NSCA daemon writes it logs to the standard syslog facility, so you can usually find messages in /var/log/messages. Enabling the "debug" option in the NSCA daemon config file, will cause more verbose information to be written to the logs. This is especially useful to see if the packets are being received at all. But I would enable it in any case simply to have a log of what actually comes through even if it is not interpreted by Nagios. To enable debug, change the 0 to 1 in the following line:

```
debug=1
```

Finally, we need to set the permissions for the cfg file. After saving and closing the nsca.cfg, set the owner and group to whatever you are running user you are running nsca as. Second, add read permissions for the rest of the group.

Once you are through the testing phase, it is highly recommended that you use a password with NSCA. Remember you must enter the same password in BOTH the nsca.cfg as well as your client (most likely send_nsca.cfg).

```
[root@localhost] chown nagios.nagcmd /usr/local/nagios/etc/nsca.cfg
[root@localhost] chmod g+r /usr/local/nagios/etc/nsca.cfg
```

Running NSCA

The next step would be to start up NSCA. Keep in mind that at this point, although it will be passing information to Nagios, Nagios doesn’t yet have any service to process this information, so it will simply throw out the data as irrelevant. This is where having debug information show up in our syslog is useful.
So we will call the executable nsca and provide it with the location of its conf file.

```
[root@localhost] /usr/local/nagios/bin/nsca –c /usr/local/nagios/etc/nsca.cfg
```

NSCA should now be running. Now to test it, we have two options. First of all, I like to port scan, so I scan the machine to see if the port specified (if left as default 5667) is open. This indicates that the program is in fact running and ready to receive information. If the port isn’t found open then you may have some other issue such as a firewall (iptables?) blocking it.
After determining that NSCA is running and accesable, we can try to send it some data. We can try from the same machine, or from another host using the send_nsca binary that was compiled at the

same time we compiled nsca. There are also plenty of third party software titles that incorporate send_nsca that you can use. I'll show an example using send_nsca from the local machine. Assuming you haven't yet put any password on the nsca host yet, we don't need to configure anything for send_nsca. Send_NSCA by default uses tab delimited format, since often we cannot enter tabs if in a GUI command prompt, the work-around is to create a file containing our packet to send and pipe it to send_nsca. The format for a service check packet using NSCA is

```
<hostname>[tab]<svc_description>[tab]<return_code>[tab]<plugin_output>[newline].
```

So create a normal text file named test with the following:

```
localhost       TestMessage     0       This is a test message. [return]
```

Save the file and then run send_nsca.

```
[root@localhost] $DOWNLOADPATH$/nsca-version-number/src/send_nsca  localhost –c
$DOWNLOADPATH$/nsca-version-number/sample-config/send_nsca.cfg < test
```

If it sent, then we should get a message saying "1 data packet(s) sent to host successfully." You should also be able to look in your syslogs and see the process as well as the actual packet received as well as perhaps Nagios taking the packet in as a External Command.

```
        May 23 15:46:49 localhost nsca[1731]: [ID 879649 daemon.info] Handling the connection...
        May 23 15:46:49 localhost nsca[1731]: [ID 691272 daemon.notice] SERVICE CHECK ->
        Host Name: 'localhost', Service Description: 'TestMessage', Return Code: '0', Output: 'This is
        a test message.'
        May 23 15:46:49 localhost nsca[1731]: [ID 862360 daemon.error] End of connection...
```

Installing NSCA as a service under inetd or xinetd

I recommend installing NSCA as a service if you are willing. It's not necessary, but it does simplify things in my opinion. Since the README file contained with NSCA did an excellent job on this section, I've chosen to simply include it in my tutorial.

**Note the following are mostly exerts from the README file contained with the NSCA package with some comments and reformatting by me:*

1) Add a line to your /etc/services file as follows (modify the port number as you see fit)

```
        nsca            5667/tcp        # NSCA
```

2) Add entries for the NSCA daemon to either your inetd or xinetd configuration files. Which one your use will depend on which superserver is installed on your system. Both methods are described below. NOTE: If you run nsca under inetd or xinetd, the server_port and allowed_hosts variables in the nrpe configuration file are ignored.

   ***** INETD *****

   If your system uses the inetd superserver WITH tcpwrappers, add an entry to /etc/inetd.conf as follows:

```
        nsca    stream  tcp     nowait  <user>  /usr/sbin/tcpd  <nscabin> -c <nscacfg> --inetd
```

   If your system uses the inetd superserver WITHOUT tcpwrappers, add an entry to /etc/inetd.conf as follows:

```
        nsca    stream  tcp     nowait  <user>  <nscabin> -c <nscacfg> --inetd
```

- Replace <user> with the name of the user that nsca server should run as.
  Example: nagios
- Replace <nscabin> with the path to the nsca binary on your system.
  Example: /usr/local/nagios/nsca
- Replace <nscacfg> with the path to the nsca config file on your system.
  Example: /usr/local/nagios/nsca.cfg

***** XINETD *****

If you are running under the default options in your config file, you can use the sample xinetd file contained in sample-config/ without changes except to add the IP address' of your client machines.

If your system uses xinetd instead of inetd, you'll probably want to create a file called 'nsca' in your /etc/xinetd.d directory that contains the following entries (a sample config file called nsca.xinetd should be created in the root folder of the distribution after you run the configure script):

```
# default: on
# description: NSCA
service nsca
{
Flags              = REUSE
socket_type        = stream
wait               = no
user               = <user>
group              = <group>
server             = <nscabin>
server_args        = -c <nscacfg> --inetd
log_on_failure     += USERID
disable            = no
only_from          = <ipaddress1> <ipaddress2> ...
}
```

- Replace <user> with the name of the user that the nsca server should run as.
- Replace <group> with the name of the group that the nsca server should run as.
- Replace <nscabin> with the path to the nsca binary on your system.
- Replace <nscacfg> with the path to the nsca config file on your system.
- Replace the <ipaddress> fields with the IP addresses of hosts which
  are allowed to connect to the NSCA daemon. This only works if xinetd was
  compiled with support for tcpwrappers.

You may omit the "only_from" line to accept connections from any host. This is useful in a DHCP environment when receiving from hosts without a fixed IP. In this case, it is even more important to include a password on your nsca communication as a way of authentication.

3) Restart inetd or xinetd will the following command (pick the one that is appropriate for your system:

First make sure that you are not still running the stand-alone daemon. You can this with the following command:

Kill < /var/run/nsca.pid

```
/etc/rc.d/init.d/inet restart

/etc/rc.d/init.d/xinetd restart
```

Check your syslog for any errors in restarting nsca in the daemon.

Configuring Nagios

If everything is running smoothly so far, the final step would be to create the service to process your passive checks in Nagios. I will show how to create a standard passive check service, but as with everything in Nagios, it should be customized for your own needs.
We are going to use the check_dummy as our check_command in the service we must define that command as well. Open up Nagios config file containing your check commands, if you are using minimal.cfg, this will be in here. Enter the following lines in the commands section:

```
define command{
        command_name            check_dummy
        command_line            $USER1$/check_dummy $ARG1$
        }
```

Next, we will create a service template for the passive checks. Enter the following lines at the bottom of the services section:

```
# Define a passive check template

define service{
        use                     generic-service
        name                    passive_service
        active_checks_enabled   0
        passive_checks_enabled  1                       # We want only passive checking
        flap_detection_enabled  0
        register                0                       # This is a template, not a real service
        is_volatile             0
        check_period            24x7
        max_check_attempts      1
        normal_check_interval   5
        retry_check_interval    1
        check_freshness         0
        contact_groups          admins, <additional_contact_groups>
        check_command           check_dummy!0
        notification_interval   120
        notification_period     24x7
        notification_options    w,u,c,r
        stalking_options        w,c,u
        }
```

After, we can create actual services to match our service checks being passed by NSCA. Keep in mind that the service_description must match the svc_description received in the nsca packet, in our above example using send_nsca, the svc-description was “TestMessage”. I will continue building a service check using that example:

```
define service{
        use                     passive_service
        service_description     TestMessage
        host_name               localhost
        }
```

Restart the Nagios daemon so that it loads the updates in your config file. If you have installed Nagios as a service you can use:

```
service nagios restart
```

You should now be able to see the service listed on the nagios web interface. Note that the service is PENDING until it receives it's first result. It has no scheduled updates because it is a passive service. We should be able to send a packet with our message now using send_nsca and have it processed and displayed on the web interface.

> **Note: Some documentation I've read says that it can take up to an hour for Nagios to fully activate this service, thus packets will not process immediately. But after it is "activated", results process at normal speeds. I'm still looking for verification of this, and why it may or may not happen.*

Repeat the same steps as our last send:

```
[root@localhost] $DOWNLOADPATH$/nsca-version-number/src/send_nsca  localhost –c
$DOWNLOADPATH$/nsca-version-number/sample-config/send_nsca.cfg < test.
```

After a moment the web interface should show the service status as OK and have our message under Status Information. One more test would be to send an error. Edit the test text file to say:

```
localhost          TestMessage     2          This is a Test Error [return]
```

Try sending again and the result should be a red ERROR under status. In addition, this should have triggered the notification check and send an email to the members of your admin contact group.

The host of your NSCA should now be up and running. Remember you will have to customize the Nagios services for actual use depending on the client and service check being received.

Problems

If you are not getting alerts, check the following:

1) Make sure the hostname defined in Nagios EXACTLY matches the hostname defined in your packet sent INCLUDING upper/lower case.
2) Make sure the service description used in the packet sent EXACTLY matches the Nagios service description, including case, embedded spaces, etc.
3) Check the NSCA log (syslog?) to make sure that the messages are being received by NSCA.
4) Check the nagios.log file to make sure they are being processed by Nagios. Have you enabled the external command file?
5) Check the nagios.log file. Are the NSCA messages being ignored because you have the wrong hostname or service description?
6) Check your filewall settings to allow 5667 (or other port as used)
7) If you have installed under xinetd, make sure you specify the IP address you are listening for, or remove the "only_from" line in your xinetd's nsca script.

If you are still having problems after reading my tutorial, feel free to contact me @ barahonc@cs.ucdavis.edu. I don't pretend to be an expert in this area, but I did go through a lot of reading and trial and error getting this running here so I'll help in what I can. If you felt this guide useful, please consider donating a few dollars to me. I am a struggling CS student who is currently not making enough to pay for school. Paypal: barahonc@cs.ucdavis.edu

Written by Carlos Barahona