www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

How to Setup basic Windows monitoring checks with Nagios

Bradley Tinder, Systems Integrator, SPK and Associates

In the below example, it is assumed you have setup NSClient++ on the Windows server you wish to monitor and it is functioning properly. If you have not done this, check out my blog post about setting this up here:

http://www.spkaa.com/blog/how-to-monitor-windows-servers-using-nagios-and-nsclient

First, we will be setting up a disk space check. This will monitor disk usage to ensure that none of your disks run out of space and bring down the server. We will need to add a new command to the Nagios configuration that will call NRPE on the Windows server and query the disk(s) in question. NSClient includes a built in disk check command, so we will use that.

Add this command definition:

```
define command{
        command_name check_win_drive
        command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -p 5666 -c CheckDriveSize -a ShowAll
MaxWarnUsed=$ARG2$% MaxCritUsed=$ARG3$% Drive=$ARG1$
        }
```

This command will monitor the used space on a fixed drive. For example, when the drive space used gets up to around 90% you might want to receive a warning email or page. If it gets up to 98%, you probably want a critical alert so you can free up space on the drive. This command syntax is what we use to monitor drives at our clients.

Next, we will add the actual service check in Nagios as detailed here:

```
define service{
        use                             local-service
        host_name                       host1
        service_description             Disk Space C:
        normal_check_interval           5
        notification_interval           15
        contact_groups                  alertgroup
        check_command                   check_win_drive!C!90!98
        }
```

This service definition will now send out a warning email and page when the C: drive reaches 90% used space, and a critical page when the drive space used reaches 98%. It also checks every 5 minutes and pages every 15 until the alert is acknowledged or resolved.

You can then copy this definition and add it for every fixed drive you want to monitor, all you need to do is change the C argument to whatever drive it is you want to monitor.

Our next check will be a CPU load check. Monitoring the CPU load is important for critical servers. If a server is overloaded, everyone who uses it will be affected. This is especially important with Exchange servers and Active Directory domain controllers. I like to use a Unix-like load average check, which checks CPU usage at the 1 minute, 5

minute and 15 minute intervals. If the CPU is being bogged down by something, this check will catch it. It also filters out spikes in CPU load and will alert on real events instead.
So, first we need to add our command definition as follows:

```
define command{
        command_name check_win_cpuload
        command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -p 5666 -c CheckCPU -a warn=100 crit=100 
time=1 warn=95 crit=99 time=5 warn=90 crit=95 time=15
        }
```

This command will call the CheckCPU command within the NSClient process and ask for the CPU usage percentage at the 1, 5, and 15 minute marks. The parameters to the warn= and crit= are in percentage. I found these values to be perfect to catch any rogue processes on Windows servers.

Next, we need to add the service definition:

```
define service{
        use                             local-service
        host_name                       host1
        service_description             CPU Load Average
        normal_check_interval           5
        notification_interval           15
        contact_groups                  alertgroup
        check_command                   check_win_cpuload
        }
```

Our final basic monitor check will be for a specific Windows service. This check will ensure that the defined service is up and running. This is great for monitoring critical services like DHCP or DNS or any other 3rd party service.

First, we add the command:

```
define command{
        command_name check_win_service
        command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -p 5666 -c CheckServiceState -a ShowAll 
"$ARG1$"
}
```

This is a simple command, it'll return OK if the service is started, or CRITICAL if the service is stopped. The double quotes are important, as this allows you to use the entire service name with spaces. Next, let's add the actual service definition:

```
define service{
        use                             local-service
        host_name                       host1
        service_description             DNS service
        normal_check_interval           5
        notification_interval           15
        contact_groups                  alertgroup
        check_command                   check_win_service!DNS Server
        }
```

The argument to the check_win_service needs to be the EXACT service name as shown in the Administrative Tools -> Services display on the Windows service. This example is checking to make sure the DNS server service is running. Note the spaces, that's why we used double quotes in the command definition.

Summary

Now, we can restart Nagios and our new checks will go into effect. There are a vast number of other checks you can do with Windows servers, but these are some of the basic checks that all Windows servers should have. Feel free to comment on our blog if you have any spiffy Nagios checks for your servers!