Remotely monitor servers with Nagios' check_by_ssh plugin

By Vincent Danen

Nagios is a monitoring system that can be used to monitor a wide variety of services and criteria. Remotely, it can monitor anything that can be accessed remotely: Web sites, SMTP servers, FTP servers, and so forth. Locally, it can monitor even more: load average, swap and memory usage, disk space usage, hard drive temperatures, and the like. In fact, Nagios' extensible nature makes writing plugins a breeze, so it is possible to monitor anything for which you are able to get representable data.

Unfortunately, if you wish to monitor local resource usage on a remote site it can be a little trickier. There are a number of ways this can be done, from using NSCA (Nagios Service Check Acceptor) to using NRPE (Nagios Remote Plugin Executor). These solutions may be best if you are able to compile and install software on the other machine, but if that is not a possibility, there are other solutions.

One such solution is to execute checks via SSH. If you are able to access the remote machine via SSH and have the ability to run programs out of a home directory, and the ability to set an SSH public key, then the *check_by_ssh* plugin is perhaps your best bet.

The first step is to ensure that the central Nagios server is able to connect to the remote host via SSH in a manner that does not require a password. This would require creating a password-less public/private keypair as the user running the Nagios service (typically "nagios"), sending the public key to the remote server, and then (as user "nagios") logging into the remote system. For example:

```
nagios@nagiosserver:~/ > $ ssh-keygen -t dsa

Generating public/private dsa key pair.

Enter file in which to save the key (/home/nagios/.ssh/id_dsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/nagios/.ssh/id_dsa.

Your public key has been saved in /home/nagios/.ssh/id_dsa.pub.

The key fingerprint is:

6a:b4:cb:f1:7d:7b:7c:1b:c4:79:2a:5d:5a:16:da:b8 nagios@nagiosserver.com

nagios@nagiosserver:~/ > $ scp .ssh/id_dsa.pub
user@remotehost.com:~/.ssh/authorized_keys

nagios@nagiosserver:~/ > $ ssh user@remotehost.com

user@remotehost:~/ > $
```

This creates the key without a passphrase and then copies the newly-created *id_dsa.pub* public key file to the remote host. Make sure that the *~user/.ssh* directory already exists on the remote host and ensure that it is mode 0700 to protect it. If that is all correct, then using ssh to connect to the remote site as the specified user should yield a shell prompt. If so, then we can configure Nagios to use *check_by_ssh*.

One quick note: if you are able to create a dedicated account on the remote system for this, it would be best to do so. If, on the other hand, you are unable to, be sure to adequately protect your central Nagios server, because if anyone can obtain privileges as "nagios" on the central server, they will have an easy ticket to your user account on the remote server.

As well, copying whichever plugins you wish to execute on the remote machine into a *~/bin* or *~/plugins* directory would be the next step. To step up security, you can write a wrapper script to execute those specific commands and modify *~/.ssh/authorized_keys* on the remote server to only execute the wrapper script, which would prevent that key from being used for anything other than executing Nagios checks.

On the central Nagios server, in the *commands.cfg* configuration file, define the new checks. The example below defines a new *check_ssh_load* command:

```
# 'check_ssh_load' command definition

define command {

        command_name    check_ssh_load

        command_line    $USER1$/check_by_ssh -H $HOSTADDRESS$ -C
"/home/user/bin/check_load -w $ARG1$ -c $ARG2$"

}
```

This command will call the *check_by_ssh* plugin to connect to the specified host (via the $HOSTADDRESS$ macro) and execute the command */home/user/bin/check_load*, which is the *check_load plugin*, on the remote machine; you will need to adjust the path to match the location of that plugin on the remote server. As well, if paths and/or usernames differ on remote servers and you plan to monitor more than one, you may need to define multiple commands, one for each server (or use macros).

Next, edit *services.cfg* and add the following:

```
define service {

        use                             local-service           ; check current load
on machine

        hostgroup_name                  ssh-nagios-services

        service_description             Current Load

        check_command                   check_ssh_load!5.0,4.0,3.0!10.0,6.0,4.0

}
```

This defines a new service to execute for hosts in the *ssh-nagios-services* hostgroup. It calls the defined *check_ssh_load* command and will put the service in a warn state if the load average hits 5, and a critical state if it hits 10 (adjust to suit, of course).

Finally, edit *hostgroups.cfg* to create the *ssh-nagios-services* hostgroup. Systems added to this hostgroup will automatically begin to use the defined service.

```
define hostgroup {

        hostgroup_name  ssh-nagios-services

        alias           Nagios over SSH

        members         remote1,remote2
```

```
}
```

Here we define that remote1 and remote2 both belong to this hostgroup. As a result, both will start using the *check_ssh_load* command.

Using *check_by_ssh* is a convenient and secure way to execute Nagios plugins on remote servers. When all you can see of the status of a remote server is HTTP or SMTP availability, your view of the server is quite restricted. Being able to see local resource usage can allow you to spot problems, and correct them, before they are visible to users.