Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)
Fachbereich Informatik, Mathematik und Naturwissenschaften

Projektarbeit

Nagios - Grundlegende Einführung und Überwachung von Windows-Rechnern

Vorgelegt von

Matthias Jauernig (06INM),
Michael Lahl (03IN-P)

Leipzig, Januar 2007

Lehrveranstaltung: Netzwerk- und Systemmanagement, WS 06/07
Verantwortlicher: Prof. Dr. Klaus Hänßgen

Zusammenfassung

Diese Arbeit behandelt Teilaspekte des System- und Netzwerkmanagement-Paketes *Nagios*. Zunächst wird ein grundlegender Überblick über Nagios gegeben und auf Installation sowie Konfiguration des Systems eingegangen. Diese Schritte gestalten sich aufgrund ihrer Komplexität als nicht einsteigerfreundlich, erlauben aber eine hohe Flexibilität und Anpassung an eine Vielzahl von Problemen. Desweiteren wird besonders auf die Funktionalität der Weboberfläche eingegangen, wobei sich zeigt, dass diese flexibel genug ist, um Administratoren stets die von ihnen benötigten Informationen bereit zu stellen. Im zweiten Teil der Arbeit wird umfassend die Überwachung von Windows-Systemen mit Nagios betrachtet, wobei hierzu gleich 5 Varianten untersucht werden. Diese unterscheiden sich teilweise stark in Bezug auf Aspekte der Funktionalität, Sicherheit, Komplexität, Aktualität, der aktiven Weiterentwicklung und dem erforderlichen Konfigurationsaufwand. Es folgt eine Bewertung und ein Vergleich der Pakete, um Empfehlungen für die Auswahl einer Lösung geben zu können. Eine wichtige Erkenntnis ist hierbei, dass sich trotz dieser Bewertung keine allgemein gültigen Aussagen über den Einsatz eines Paketes treffen lassen. So bleibt die Wahl der Lösung zur Windows-Überwachung ein individuelles Problem, bei dessen Bearbeitung die vorliegende Übersicht behilflich ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Das Internet wächst unaufhaltsam. Was 1969 mit der Vernetzung von vier Universitäten begann, ist heute zu einem unüberschaubaren Netz mit mehreren Millionen Teilnehmern geworden. Mit der Größe des weltweiten Rechnerverbundes wächst auch der Bedarf nach Informationen, was in hohen Forderungen nach zusätzlichem Speicher und Rechenzeit resultiert. Doch nicht nur das Internet lässt den Ressourcenbedarf enorm anwachsen, auch neue Software-Technologien und die allumfassende Digitalisierung erfordern immer mehr Speicherkapazität und höhere Rechenleistungen. Damit Firmen und Organisationen expandieren können, stellt es eine Grundvoraussetzung dar in neue Server zu investieren, um IT-Engpässen von vornherein entgegen zu wirken. Die Größe der entsprechenden Rechnernetze wächst dadurch stetig an und Administratoren stehen vor dem Problem für solche Netze verantwortlich zu sein. Der Ausfall von Servern kann nicht nur hohe Kosten verursachen, sondern auch den Ruf bei Kunden ruinieren. Jede Minute, die ein Server nicht erreicht werden kann, ist dabei entscheidend. Neben dem Ausfall von Rechnern können auch das nicht ordnungsgemäße Funktionieren einzelner Dienste und die Knappheit von System-Ressourcen von großer Bedeutung sein. Administratoren müssen daher rund um die Uhr über eventuelle Probleme in ihrem Netzwerk Bescheid wissen und vor allem den Überblick bewahren. Doch auch wenn eine schnelle Alarmierung erfolgt und die Problembehebung initialisiert wurde, gestaltet sich der Prozess der Fehlersuche nicht selten als zeit- und nervenraubende Aufgabe.

Um ihre Systeme zu pflegen und Administratoren bei deren Arbeit zu unterstützen ist es für viele Firmen unerlässlich, Pakete für das System- und Netzwerkmanagement einzusetzen. Diese sind in der Lage Administratoren bei Problemen mit Rechnern, Diensten und Ressourcen umgehend zu benachrichtigen und bei der Fehlersuche zu assistieren. Informationen können dabei häufig verdichtet werden, bei Bedarf aber auch umfassend informieren. Kommerzielle Lösungen stellen beispielsweise Tivoli (IBM), Openview (HP), Unicenter (Computer Associates) und Patrol (BMC) dar, die in der Lage sind, tausende von Servern zu verwalten und zu überwachen. Der Anschaffungspreis, den ein Unternehmen für solch ein umfassendes Schwergewicht zahlen muss, rechnet sich meist jedoch auch. Anders sieht es bei kleinen bis mittelständigen Unternehmen aus, welche die Kosten für große kommerzielle Lösungen nicht tragen können oder nur einen Teil des angebotenen Funktionsumfangs wirklich benötigen.

Auf der Suche nach Alternativen werden solche Unternehmen im Open-Source-Umfeld fündig. Die wichtigsten Funktionen werden hier übersichtlich und gebündelt zur Verfügung gestellt, weiterhin spricht die leichte Erweiterbarkeit der meisten Systeme für den Einsatz von Open-Source-Management-Werkzeugen. Vergessen werden darf auch nicht der Kostenfaktor, der durch die nicht vorhandenen Anschaffungskosten und die geringen laufenden Kosten ein großes Plus für solche Lösungen darstellt. Neben „Cacti“, „Big Sister“, „Ganglia“, „Zenoss“ und anderen stellt vor allem „Nagios“ ein quelloffenes Werkzeug dar, welches in vielen tatsächlichen Installationen zum Einsatz kommt. Mit Nagios lassen sich auch komplexere Netzwerk-Strukturen überwachen, die Verfolgung eines Plugin-Konzepts sorgt zudem für leichte Erweiterbarkeit der Funktionalität. Nagios lebt auch von seiner aktiven Community, durch welche nicht nur das Paket selbst weiter entwickelt wird, sondern auch zusätzliche Funktionen und Programme zur Verfügung gestellt werden.

1.2 Inhalte der Arbeit

Diese Arbeit betrachtet das System- und Netzwerk-Management-Werkzeug *Nagios*. Dabei kann nicht die gesamte Palette an Funktionen und Einsatzmöglichkeiten beleuchtet werden, wodurch nur Grundkonzepte und spezielle Teilaspekte des Systems zur Sprache kommen werden. Die Arbeit lässt sich in zwei Teile untergliedern. Der erste Teil beschäftigt sich mit der Installation, Konfiguration und Grundkonzepten von Nagios. Dabei stellt sich heraus, dass diese Schritte aufgrund ihrer Komplexität Einsteiger überfordern können, sie erlauben aber auch eine hohe Flexibilität und Anpassung an eine Vielzahl von Problemen. Weiterhin wird besonders die Funktionalität der Weboberfläche beschrieben, welche es dem Administrator ermöglicht, genau die Informationen angezeigt zu bekommen, die für ihn wichtig sind. Der zweite Teil gibt dann einen umfangreichen Überblick über verschiedene Möglichkeiten der Überwachung von Windows-Rechnern mithilfe von Nagios, wobei auch eine Bewertung sowie ein Vergleich erfolgen. Allgemein gültige Empfehlungen für eine Lösung lassen sich dabei nicht treffen, als Kernaussage gilt, dass die Wahl eines Paketes von der jeweiligen Rechnerumgebung und dem Einsatzzweck abhängt und individuell erfolgen muss. Die angegebene Übersicht der Lösungen und die Bewertung geben eine Orientierung und sollen bei der Wahl eines Systems behilflich sein.

Zunächst wird in Abschnitt 2 eine kurze Zusammenfassung der Basis-Funktionalität von Nagios gegeben. In 3 wird grundlegend auf die Installation und Konfiguration des Nagios-Paketes eingegangen. In Abschnitt 4 wird zunächst eine kleine Beispielkonfiguration erzeugt, um dabei einen ersten Eindruck von der Weboberfläche zu erhalten (Unterabschnitt 4.2). In 4.3 wird deren Funktionalität dann näher beleuchtet und konkretisiert. Mit Plugins (4.4), Addons (4.5) und Überwachungsarten (4.6) werden weitere Konzepte von Nagios besprochen, die auch für das Verständnis der Windows-Überwachung von Bedeutung sind.

In Abschnitt 5 erfolgt schließlich eine umfangreiche Darstellung der Möglichkeiten zur Überwachung von Windows-Rechnern mit Nagios. Den Autoren ist dabei keine Publikation bekannt, die in ähnlich umfangreicher Weise die besprochenen Varianten vorstellt und bewertet. Es wird für jedes Paket zunächst eine kurze Beschreibung abgeliefert, um dann auf die Konfiguration von Windows-Client und Nagios-Host einzugehen. In Unterabschnitt 5.7 erfolgt dann eine zusammenfassende Bewertung der einzelnen Varianten, um sie in 5.8 miteinander vergleichen zu können.

In Abschnitt 6 wird abschließend ein Resümee gezogen und die Inhalte der Arbeit zusammengefasst.

1.3 Verwendete Nagios-Hosts

Folgende Rechner dienten dieser Arbeit als Nagios-Hosts. Auf ihnen wurde Nagios installiert, um damit andere Rechner überwachen zu können:

Host `rtc1`:

- Rechner: AMD Athlon XP 2400+, 768 MB RAM
- Betriebssystem: OpenSUSE 10.2
- Software-Versionen:
  - Nagios: 2.6

  - Apache: 2.2.3
  - gd library: 2.0.33
  - gd-devel: 2.0.33

Host `Router-litschinetz`:

- Rechner: Intel Pentium III 700Mhz, 384 MB RAM
- Betriebssystem: Debian Linux 3.1
- Software-Versionen:
  - Nagios: 2.6
  - Apache: 1.3.33
  - gd library: 2.0.33
  - libgd2-dev: 2.0.33

Dabei wurden auf beiden Rechnern die Schritte durchgeführt, welche zur Installation und Konfiguration erforderlich sind und in Abschnitt 3 beschrieben werden. Der Host „Router-litschinetz“ wurde dazu verwendet, um die Beispielkonfiguration und die Übersicht über die Weboberfläche in Abschnitt 4 zu erhalten. Der Host „rtc1“ diente als Grundlage für die Windows-Überwachung, welche in Abschnitt 5 beschrieben wird.

2 Funktionsumfang von Nagios

Bei Nagios handelt es sich um eine Open-Source-Applikation für das System- und Netzwerkmanagement. Es lassen sich damit Rechner sowie Dienste bzw. Ressourcen auf diesen überwachen und Alarme auslösen, sobald definierte Grenzen überschritten werden. Nagios wurde ursprünglich für Linux entwickelt, ist aber auch auf vielen anderen Unix-Derivaten lauffähig.

Die Dokumentation von Nagios zählt einige der wichtigsten Merkmale auf:

- Überwachung von Netzwerk-Diensten (SMTP, POP3, HTTP, NNTP, PING etc.),
- Überwachung von Host-Ressourcen (Prozessorlast, Festplattenbelegung etc.),
- Einfache Plugin-Schnittstelle für eigene, individuelle Überwachungsaufgaben,
- Parallele Dienst-Abfragen,
- Möglichkeit zur Festlegung von Netzwerk-Hierarchien, um z. B. bei Ausfall eines Routers unnötige Fehlalarme im jeweiligen Subnetz zu vermeiden,
- Verschiedene Kontaktierungsmöglichkeiten bei auftretenden Problemen (z. B. via Pager, E-Mail, SMS etc.),
- Optionale Integration einer Weboberfläche zur übersichtlichen Überwachung.

3 Installation und Konfiguration von Nagios

Nagios ist flexibel konfigurierbar, gestaltet sich jedoch als nicht einsteigerfreundlich. Das ist vor allem auf die langwierige Installation und Konfiguration zurück zu führen, die komplett auf der Konsole zu erfolgen hat. Nach der Erstkonfiguration steht einer effektiven Nutzung des Systems jedoch nichts mehr im Wege.

Im diesem Abschnitt wird *nicht* umfassend auf die Installation und Konfiguration von Nagios eingegangen, dafür sei an dieser Stelle auf die umfangreiche Dokumentation verwiesen, die der Nagios-Distribution beiliegt. Vielmehr wird ein Gesamteindruck des Installations- und Konfigurationsprozesses und ein Überblick über die dabei nötigen Schritte gegeben.

3.1 Systemvoraussetzungen

Es gibt nur wenige Voraussetzungen für die Installation von Nagios, wodurch das System vielerorts einsetzbar ist. Benötigt werden systemseitig:

- ein installiertes Linux oder anderes Unix-Derivat,
- ein C-Compiler,
- eine konfigurierte TCP/IP-Umgebung für Netzwerkfunktionen,
- zur Nutzung der enthaltenen CGIs:
  - ein Webserver (vorzugsweise Apache, siehe [2]),
  - die gd-Bibliothek 1.6.3 oder höher (siehe [8]).

Zudem müssen sowohl ein Benutzer als auch eine Gruppe vorhanden sein, denen Berechtigungen für die Nagios-Administration zugewiesen werden. Im Standardfall wird „nagios“ als Benutzer- und Gruppenname verwendet, beide werden angelegt mit:

```
#> adduser nagios
#> groupadd nagios
```

3.2 Installation

Die Installation von Nagios gestaltete sich auf dem verwendeten Rechner als problemlos. Zunächst wurde Nagios 2.6 als Tarball von [9] herunter geladen und auf der lokalen Festplatte entpackt. Im dadurch entstandenen Verzeichnis wurde dann eine vollständige Installation über die folgenden Schritte durchgeführt:

```
%> ./configure
%> make all
#> make install
#> make install-init
#> make install-config
```

Zur kurzen Erklärung der einzelnen Aufrufe:
`configure`: Dieser Aufruf konfiguriert die Installation für die aktuelle Umgebung und prüft benötigte Programmpakete.
`make all`: Dieser Aufruf kompiliert Nagios und die CGIs.
`make install`: Hierdurch werden die Binaries und die HTML Dokumentation installiert.
`make install-init`: Installiert ein Init-Skript, um Nagios bei Systemstart auszuführen.
`make install-config`: Erzeugt eine Beispielkonfiguration für Nagios.

3.3 Installation von Plugins

Ein frisch installiertes Nagios-Paket bringt zunächst keinen Nutzen. Dieser wird erst durch die Installation von Plugins erbracht. Hierfür lädt man das Paket *nagiosplug* (hier benutzt in Version 1.4.5) von [21] ebenfalls als Tarball herunter. Nachdem das Archiv entpackt wurde, werden die Plugins kompiliert und installiert mit:

```
%> ./configure
%> make
#> make install
```

3.4 Einrichten der Weboberfläche

Zum Betrieb von Nagios ist die Weboberfläche nicht zwingend erforderlich, sie ist aber unentbehrlich, um einen administrativen Überblick über verfügbare Dienste und eventuelle Konflikte/Probleme zu erhalten (siehe auch Abschnitt 4.3).

Zunächst hat die Konfiguration des Webservers auf die Verwendung von Nagios hin zu erfolgen. Dafür enthält die Datei `httpd.conf` im Verzeichnis „sample-config“ der Nagios-Distribution eine Beispiel-Konfiguration. Diese ist in die Konfigurationsdatei des HTTP-Servers (hier: `/etc/apache2/httpd.conf`) einzufügen und der aktuellen Umgebung entsprechend anzupassen. Danach muss der Webserver neu gestartet werden.

Zu guter Letzt sind noch die Berechtigungen für den Zugriff auf die Weboberfläche zu setzen. Dazu muss jeder Benutzer, der Zugriff erhalten soll, mittels des Befehls `htpasswd2` (der Bestandteil von Apache2 ist) berechtigt werden. Für den ersten Benutzer ist die Option `-c` zu übergeben, um die Berechtigungsdatei zu erstellen. Will man weitere Benutzer hinzufügen, so darf diese Option nicht mehr verwendet werden:

```
#> htpasswd2 -c /usr/local/nagios/etc/htpasswd.users <username1>
   New password:
   Re-type new password:
```

```
    Adding password for user <username1>
#> htpasswd2 /usr/local/nagios/etc/htpasswd.users <username2>
    New password:
    Re-type new password:
    Adding password for user <username2>
```

Nach diesen Schritten kann man sich mittels der eben erstellen Benutzerdaten bereits auf der Weboberfläche unter der URL `http://localhost/nagios` einloggen, allerdings ist noch keine Funktionalität nutzbar. Für weitere Informationen gibt die Datei `html/docs/installweb.html` Auskunft, die in der Nagios-Distribution enthalten ist.

3.5 Basiskonfiguration

Zur Konfiguration von Nagios gibt es eine Reihe von Konfigurationsdateien, auf die kurz eingegangen wird. Vor allem das Anlegen von Hosts und entsprechenden Diensten, die zu überwachen sind, stellen elementare Schritte dar, denen man viel Sorgfalt einzuräumen hat. Es empfiehlt sich, mittels `make install-config` bei der Nagios-Installation bereits Beispiel-Konfigurationsdateien zu erstellen und dann von Hand anzupassen. Diese sind bereits umfangreich durch Kommentare dokumentiert, sodass die Einrichtung kein Problem darstellt. In der nachfolgenden Beschreibung der einzelnen Dateien bezeichnet `<nagios>` das Verzeichnis, in welches Nagios installiert wurde (Standard: `/usr/local/nagios`).

Im Sinne eines Überblicks wird an dieser Stelle nicht detailliert auf die einzelnen Dateien eingegangen. In Abschnitt 4.2 wird jedoch eine Beispielkonfiguration erstellt, bei der einige Punkte konkretisiert werden.

Haupt-Konfigurationsdatei:

Diese ist unter `<nagios>/etc/nagios.cfg` zu finden. Sie wird sowohl von den CGIs als auch vom eigentlichen Nagios-Prozess gelesen, sodass es von Vorteil ist sie auch als erstes anzupassen. Hier lassen sich Pfade zu anderen Konfigurationsdateien definieren und weitere Grundeinstellungen vornehmen, die z. B. im Anpassen der Prüfoptionen für Hosts und Dienste, der Konfiguration des Loggings, dem Einstellen von Nagios-Benutzer und -Gruppe oder auch im Eintragen von Adressen für e-Mail und Pager des Nagios-Administrators bestehen, um bei kritischen Ereignissen informiert zu werden. Weitere Informationen findet man in der Datei `html/docs/configmain.html` des Nagios-Pakets.

Objektdefinitions-Dateien:

Dies stellt eine wichtige Gruppe von Dateien dar, werden hierdurch doch die eigentlichen Entitäten definiert, welche zu überwachen sind und die Art ihrer Überwachung. Die Haupt-Entitäten stellen Hosts dar, die logisch zu Hostgruppen zusammen gefasst werden können. Weiterhin lassen sich Dienste definieren, mit denen Funktionalitäten auf Hosts überwacht werden sollen, auch hier kann eine Verschmelzung zu Dienst-Gruppen erfolgen. Als elementar sind Befehle (Commands) zu nennen, welche die Syntax für den Zugriff auf Nagios-Plugins festlegen und von Dienst-Definitionen benutzt werden. Ebenso lassen sich Kontakte erstellen und zusammenfassen, um bei Problemen die jeweiligen Verantwortlichen direkt informieren zu können. Die Dateien zur Definition von Objekten werden in der Haupt-Konfigurationsdatei eingetragen, typisch sind dabei:

```
cfg_file=<nagios>/etc/localhost.cfg
```

```
cfg_file=<nagios>/etc/hosts.cfg
cfg_file=<nagios>/etc/hostgroups.cfg
cfg_file=<nagios>/etc/services.cfg
cfg_file=<nagios>/etc/servicegroups.cfg
cfg_file=<nagios>/etc/commands.cfg
cfg_file=<nagios>/etc/contacts.cfg
```

Zur besseren Übersichtlichkeit lassen sich auch ganze Konfigurationsverzeichnisse angeben, aus denen alle Dateien mit der Endung `.cfg` als Konfigurationsdateien entnommen werden. Zum Beispiel: `cfg_dir=<nagios>/etc/hosts`. Die eigentliche Definition von Objekten wird über ein Template-basiertes Format vorgenommen, welches ausführlich in der Datei `html/docs/xodtemplate.html` des Nagios-Pakets erläutert wird.

CGI-Konfigurationsdatei:

Diese dient der Einstellung von CGI-Pfaden sowie grundlegenden Berechtigungen und besitzt standardmäßig den Pfad `<nagios>/etc/cgi.cfg`, der sich ebenfalls in der Haupt-Konfigurationsdatei anpassen lässt. Beispiele für Konfigurationsparameter sind das Einstellen der Authentisierungs-/Autorisierungs-Funktionen für CGIs, das Verteilen von grundlegenden Rechten wie Anzeige von Systeminformationen oder Ausführen von Systemkommandos an bestimmte Benutzer oder die Konfiguration einiger Parameter für bestimmte CGIs. Weiterführende Beschreibungen sind auch hier in der umfangreichen Dokumentation unter `html/docs/configcgi.html` des Nagios-Pakets zu finden.

Ressourcen-Dateien

In diesen werden benutzerdefinierte Makros zur Prüfung von Diensten gespeichert. Ebenso lassen sich andere Informationen wie z. B. Datenbank-Verbindungen hier eintragen. Der Sinn von Ressourcendateien ist sensitive Informationen zu speichern ohne sie in CGIs verfügbar machen zu müssen. In der Haupt-Konfigurationsdatei lassen sich mehrere Ressourcen-Dateien angeben, vorgeschlagen wird eine Ressourcendatei mit dem Pfad `<nagios>/etc/resource.cfg`.

3.6 Verifizieren der Konfiguration

Hat man die Basis-Konfiguration durchgeführt und alle Entitäten hinzugefügt, so lässt sich Nagios mit der Option `-v` starten, um die Konfiguration auf syntaktische Fehler hin zu überprüfen:

```
<nagios> %> bin/nagios -v etc/nagios.cfg
```

In der Ausgabe sind die ausgelesenen Entitäten sowie etwaige Probleme aufgeführt, in einer separaten Zusammenfassung wird die Anzahl von Warnungen und Fehlern ausgegeben.

4 Grundlegender Überblick

Nachdem sich der letzte Abschnitt zusammenfassend mit der Installation und Konfiguration von Nagios beschäftigt hat, werden nun die tatsächlichen Funktionen des Paketes betrachtet. Vor

allem der Weboberfläche wurde einige Aufmerksamkeit geschenkt, so werden in Abschnitt 4.2 zunächst eine kleine Konfiguration erstellt und daran grundlegende Funktionen des Web-Frontends beschrieben, um dann in Abschnitt 4.3 detaillierter auf die wichtigsten davon einzugehen. Danach werden mit Plugins (4.4), Addons (4.5) und Überwachungsarten (4.6) weitere Basiskonzepte von Nagios besprochen, die hauptsächlich von Interesse sind.

4.1 Start von Nagios

Um Nagios zu starten, ist lediglich die entsprechende binäre Datei auszuführen. Als Parameter muss dabei die zu nutzende Hauptkonfigurationsdatei angegeben werden:

```
<nagios> %> bin/nagios etc/nagios.cfg

Nagios 2.6
Copyright (c) 1999-2006 Ethan Galstad (http://www.nagios.org)
Last Modified: 11-27-2006
License: GPL

Nagios 2.6 starting... (PID=xxxx)
```

4.2 Einrichten einer Beispielkonfiguration für die Weboberfläche

Die Nutzung der Weboberfläche ist nicht zwingend erforderlich, stellt für Administratoren allerdings einen komfortablen Weg dar, um überwachte Hosts und Dienste (Services) umfassend im Blick zu halten und sich Statistiken anzeigen zu lassen. Nachfolgend wird eine kleine Beispielkonfiguration erzeugt, um einen ersten Eindruck von der Funktionalität der Weboberfläche zu erhalten.

Nach dem Start von Nagios lassen sich auf der Weboberfläche (wenn diese korrekt konfiguriert wurde, siehe 3.4) die bereit gestellten CGIs nutzen. Unter der Adresse `http://localhost/nagios` kann man sich mittels eines angelegten Benutzers (siehe 3.4) einloggen und die in Abbildung 1 dargestellte Übersicht sehen. Am linken Rand des Browserfensters ist eine vertikale Menüleiste zu sehen, über welche sich die einzelnen Funktionen auswählen lassen. Diese sind in Kategorien eingeteilt und werden bei Auswahl im rechten Teils des Fensters dargestellt.

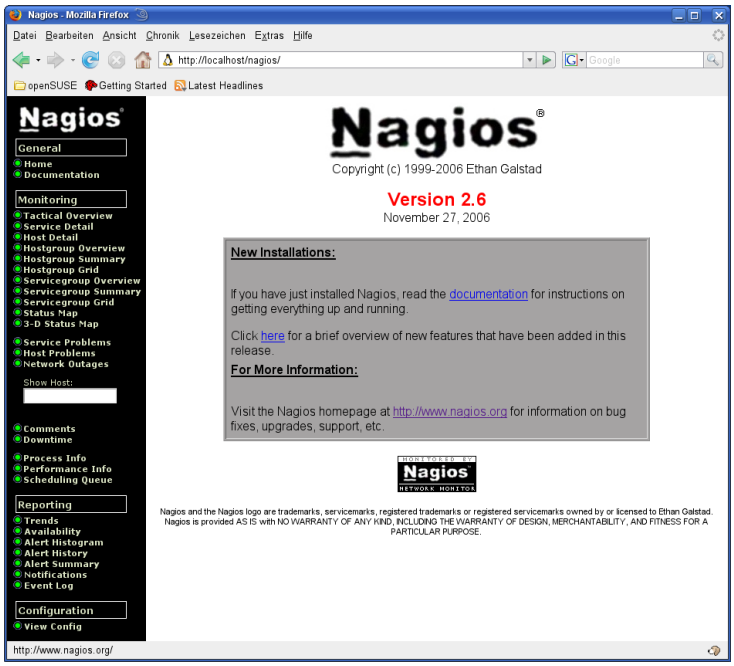

Abbildung 1: Startseite der Nagios-Weboberfläche

4.2.1 Zu überwachende Entitäten

Folgende Hosts mit den dabei angegebenen Diensten werden in der Beispielkonfiguration überwacht:

Host `localhost`:

- Rechner: Intel Pentium III 700Mhz, 384 MB RAM
- Betriebssystem: Debian Linux 3.1
- Software-Versionen:
  - Nagios: 2.6
  - Apache: 1.3.33
  - gd library: 2.0.33
  - libgd2-dev: 2.0.33
- Überwachte Objekte:
  - `Current Load`: Aktuelle CPU-Auslastung,
  - `Current Users`: Anzahl aktuell angemeldeter Benutzer,
  - `PING`: *ping* zum lokalen Rechner,
  - `Root Partition`: Nutzung der Root-Partition,
  - `Total Processes`: Gesamtanzahl gestarteter Prozesse.

Host `litschi2`:

- Rechner: AMD Athlon XP-M 2600+, 512 MB RAM

- Betriebssystem: Windows XP Professional SP1
- Software-Versionen:
  - NRPE_NT Version 0.8
- Überwachte Objekte (über NRPE_NT, siehe auch 5.3):
  - `PING`: Prüfen auf Erreichbarkeit mit Hilfe des *ping*-Kommandos,
  - `Services`: Status des Windows-Dienstes „Sygate Personal Firewall“,
  - `cpu load`: Aktuelle CPU-Nutzung,
  - `disk c`: Belegung der Festplatte C:,
  - `disk d`: Belegung der Festplatte D:,
  - `mem load`: Aktuelle Hauptspeicher-Benutzung.

Host `litschi3`:

- Rechner: AMD Athlon XP 2400+, 512 MB RAM
- Betriebssystem: Windows XP Professional SP2
- Software-Versionen:
  - NRPE_NT Version 0.8
- Überwachte Objekte:
  - `PING`: Prüfen auf Erreichbarkeit mit Hilfe des *ping*-Kommandos.

4.2.2 Anlegen von Hosts und Services

Bevor die Weboberfläche Informationen der überwachten Hosts und Dienste anzeigen kann, sind diese zunächst in den Objekt-Konfigurationsdateien von Nagios einzutragen (siehe auch Abschnitt 3.5).

Zu überwachende Dienste (Services) müssen immer Rechnern (Hosts) zugeordnet werden, auf denen sie zu überwachen sind. Diese werden beispielsweise in der Datei `hosts.cfg` definiert. Dazu wird ein Objekt-Template benutzt, durch welches ein physikalischer Rechner als zu überwachende Ressource in Nagios eingebunden wird.

Um unnötige Fehlalarme zu vermeiden, wird die Aktivität eines Dienstes von Nagios nur dann geprüft, wenn eine Verbindung zu dem entsprechenden Host aufgebaut werden kann. Dazu lässt sich in der Host-Definition ein Parameter angeben, der den Rechner auf Erreichbarkeit prüft. Klassischerweise wird dies durch einen `ping` realisiert, allerdings sind auch andere Varianten denkbar, z. B. wenn ICMP-Pakete von einer Firewall gefiltert werden, der HTTP-Port hingegen für eine Verbindung offen steht. In diesem Fall greift man auf das `check_http`-Plugin zurück. Antwortet der besagte Rechner, so können die für ihn definierten Services überwacht werden. Ist er nicht erreichbar, so kommt es auch zu keiner Prüfung der jeweiligen Dienste, d. h. der gesamte Rechner ist für das Management-Tool offline.

Nachfolgend ist ein Basiskonfigurations-Template für den Host „litschi2“ dargestellt:

```
define host{
  host_name              litschi2
  alias                  litschi2 - WinXP
  address                192.168.1.2
  check_command          check_host alive
  max_check_attempts     5
  check_period           24x7
  contact_groups         admins
  notification_interval  30
  notification_period    24x7
  notification_options   d,u
}
```

Erläuterungen:

- **host_name**
  Kurzname für die Identifikation des Hosts, dieser wird bei der Definition von Diensten (Services) und Hostgroups zur Identifikation verwendet.
- **alias**
  Alias-Name zur besseren Beschreibung des Hosts.
- **address**
  IP-Adresse des zu definierenden Hosts (*Hinweis*: wenn diese Direktive fehlt, wird der `host_name` als `address` verwendet, womit auch Namensauflösung mittels System-Host-Datei oder DNS benutzt werden kann).
- **check_command**
  Diese Direktive wird verwendet um einen Befehl festzulegen, der einen Host auf Erreichbarkeit prüft. Meist wird hierfür ein `ping` an die entsprechende `address` gesendet.
- **max_check_attempts**
  Hierüber lässt sich definieren, wie oft der `check_command` bei erfolgloser Kontaktierung des Hosts wiederholt wird.
- **check_period**
  Festlegen eines definierten Zeitraums, in dem aktive Host-Prüfungen durchgeführt werden können. Der entsprechende Zeitraum lässt sich über die Objektdefinition *timeperiod* realisieren (wird hier nicht weiter beschrieben).
- **contact_groups**
  Personen, die in den hier definierten Gruppen enthalten sind, werden bei Problemen mit dem Host benachrichtigt.
- **notification_interval**
  Benachrichtigungsintervall für die Mitglieder der definierten `contact_groups` bei Problemen. Wenn dies auf 0 gesetzt ist, wird nur eine Benachrichtigung erfolgen.
- **notification_period**
  Definition eines Zeitraums, in dem Benachrichtigungen versendet werden dürfen.

- **notification_options**
  Mit dieser Direktive wird festgelegt, wann eine Benachrichtigung erfolgen soll. (**d**: Benachrichtigung bei Status „DOWN“, **u**: Benachrichtigung bei Unerreichbarkeit (Status „UNREACHABLE“), **r**: Benachrichtigung bei Status „OK“, **f**: Benachrichtigung bei schnellen Status-Wechseln des Hosts („Flapping“)).

Nachdem auf diese Weise alle benötigten Hosts konfiguriert wurden, werden sie auf der Weboberfläche im Menüpunkt `Host Detail` dargestellt. Dabei werden Name, Status und weitere Informationen zur Prüfung angezeigt. Die hier verwendete Beispielkonfiguration wird in Abbildung 2 gezeigt. Hierbei wird im oberen Teil der Darstellung eine Zusammenfassung für Hosts und Dienste abgeliefert, während dann im unteren Teil der Status eines jeden Hosts angezeigt wird. Grün hinterlegte Hosts weisen dabei keine Probleme auf, während eine rote Hinterlegung für kritische Probleme mit dem jeweiligen Host steht.

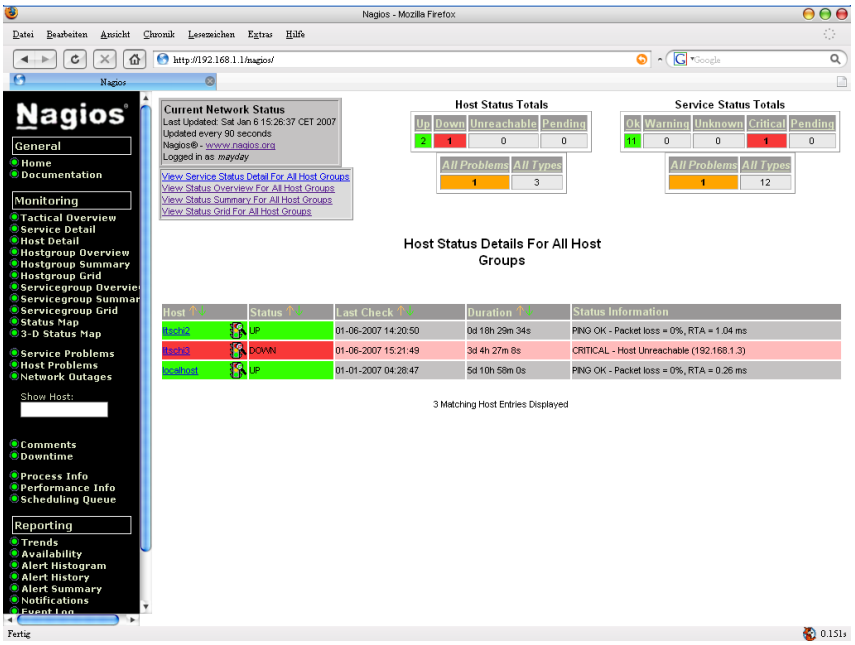

Abbildung 2: Anzeige eingerichteter Hosts im Menüpunkt „Host Detail“

Nach dem Hinzufügen von Hosts besteht der nächste Schritt in der Einrichtung von Diensten (Services), die auf den entsprechenden Hosts zu überwachen sind. Elementar werden hierfür zunächst Kommandos definiert, über welche sich Plugins von Services aus aufrufen lassen. Durch Kommandos muss bei Services nicht die gesamte Plugin-Syntax angegeben werden. Stattdessen lässt sich über die Parameter der Kommandos festlegen, welche Argumente dem Plugin zu übergeben sind. Kommandos stellen somit eine Schicht zwischen Plugins und Services dar, verbergen die eigentliche Syntax für Plugin-Aufrufe und erlauben die Kapselung gleicher Funktionalitäten, indem Argumente für Plugins über feste Einstellungen des Kommandos angegeben werden. Die Standard-Datei zur Definition eines Kommandos ist `commands.cfg`, in der das folgende Template anzugeben ist (hier für das `check_ping`-Plugin):

```
define command{
  command_name  check_ping
  command_line  $USER1$/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ -p 5
}
```

Erläuterungen:

- **command_name**
  Gibt den Namen des Kommandos an, über welchen dieses auch von Services benutzt werden kann.

- **command_line**
  Gibt die Kommandozeile an, über welche das jeweilige Plugin aufgerufen werden soll. Plugins befinden sich standardmäßig *nur* im Ordner `libexec` der Nagios-Installation, wodurch der Pfad dorthin nicht angegeben werden muss. Argumente, die beim Aufruf des Kommandos übergeben werden, sind in zwei Dollarzeichen eingebettet anzugeben. Die Argumentnamen sind fest vorgegeben: `$USER1$` gibt den Benutzer an, von dem das Plugin aufgerufen wird, `$HOSTADDRESS$` erlaubt die Übergabe der Adresse des Zielrechners. Diese beiden Parameter werden automatisch eingesetzt, wenn Nagios einen Service aufruft, sodass sie in den Service-Definitionen nicht mit angegeben werden müssen. Schlussendlich können mit `$ARG1$`, `$ARG2$` usw. Parameter angegeben werden, welche bei tatsächlichen Service-Definitionen Verwendung finden und bei Kommando-Aufrufen für diese Platzhalter eingesetzt werden. Im Beispiel des Ping-Plugins stellt `ARG1` die Marke für Warnungen und `ARG2` die Marke für kritische Meldungen dar. Über `-p 5` wird die Anzahl maximal zu sendender ICMP ECHO Pakete auf 5 begrenzt, was einen festen Wert darstellt, der durch Service-Definitionen aufbauend auf diesem Kommando nicht verändert werden kann.

Nach der Spezifikation von Kommandos können die eigentlichen Services eingerichtet werden. Eine Service-Definition wird benutzt um einen Prozess oder Ressourcen auf dem lokalen oder einem entfernten Rechner zu überwachen (z. B. nach außen angebotene Dienste wie POP, SMTP, HTTP etc.). Um einen Service zu überwachen, muss dieser in den Konfigurationsdateien definiert werden, wobei hier entsprechend der Beispielkonfiguration ebenfalls die Datei `hosts.cfg` verwendet wurde. Dabei ist mindestens folgendes Template auszufüllen (beispielhaft am `check_ping`-Plugin):

```
define service{
  use                 local-service
  host_name           localhost
  service_description PING
  check_command       check_ping!100.0,20%!500.0,60%
}
```

Erläuterungen:

- **use**
  Bindet ein zu verwendendes Template ein, in dem Direktiven bereits vordefiniert sind und von Diensten benutzt werden können.

- **host_name**
  Legt (komma-separiert) die Hosts fest, auf welchen der Service zu prüfen ist.

- **service_description**
  Diese Direktive wird verwendet um eine Beschreibung für den Service festzulegen, die auch auf der Weboberfläche angezeigt wird. Jeder Service ist durch den `host_name` und die `service_description` eindeutig identifiziert.

- **check_command**
  Das hier festgelegte Kommando wird ausgeführt um den definierten Service auf dem Host zu überwachen. Dieses Kommando muss bereits über ein `commands`-Template definiert sein, üblicherweise in der Datei `commands.cfg`. Ausrufezeichen trennen die einzelnen Argumente voneinander, welche dem Kommando übergeben werden und auf die dort über die Parameter `$ARG1$`, `$ARG2$` etc. zugegriffen werden kann.

Die so erstellten Services mit den dazu gehörigen Informationen werden auf der Weboberfläche im Menüpunkt `Service Detail` angezeigt, wie es in Abbildung 3 nach dem Hinzufügen aller benötigten Services für das hier verwendete Beispiel zu sehen ist. Im oberen Teil des Browserfensters wird eine Zusammenfassung der Hosts und Dienste angezeigt, im unteren Teil werden zu jedem Host die darauf überwachten Dienste und ihre Status dargestellt.

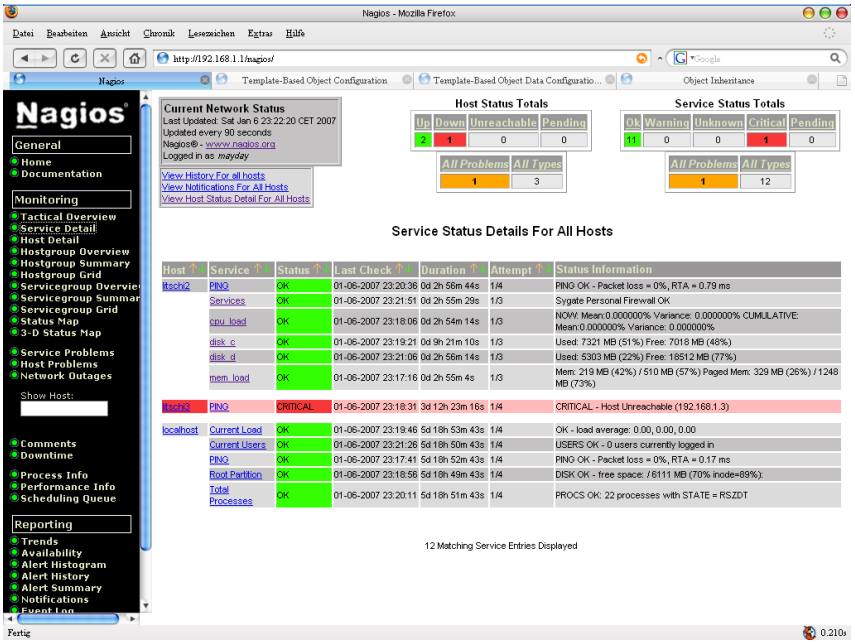

Abbildung 3: „Service Detail“ Übersicht

4.2.3 Definition von Hostgroups und Servicegroups

Gruppendefinitionen dienen dem Zweck, Hosts oder Services logisch zusammen zu fassen und gegliedert in den CGIs darzustellen. Sie dienen damit der Beschränkung von Informationen auf relevante Bereiche und der logischen Gliederung sowie der Abbildung tatsächlicher Strukturen von Netzwerken. Folgende Beispiele zeigen je ein ausgefülltes Template für eine Hostgroup und eine Servicegroup:

```
define hostgroup{
  hostgroup_name      Lan
  alias               Local Area Network
  members             litschi3, litschi2, localhost
}
```

```
define servicegroup{
  servicegroup_name   pings
  alias               PING Services
  members             litschi2,PING,litschi3,PING,localhost,PING
}
```

Erläuterungen:

- **hostgroup_name** und **servicegroup_name**
  Identifikationsname für die Hostgroup bzw. Servicegroup.
- **alias**
  Kurzbeschreibung für die Gruppe.
- **members**
  hostgroup: Eine Liste von Hosts, welche in der Gruppe enthalten sein sollen (<host1>,<host2>,...).
  servicegroup: Eine Liste von Services, welche gruppiert werden sollen. Diese besteht alternierend aus einem Host und dem dazu gehörigen Dienst (<host1>,<service1>,<host2>,<service2>,...), da Services nur durch solch eine Kombination eindeutig identifiziert werden können.

Die Abbildungen 4 und 5 zeigen definierte Host- und Service-Gruppen auf der Weboberfläche. Dabei werden im unteren Teil des jeweiligen Browserfensters die Status der einzelnen Dienste und Hosts in Gruppen zusammen gefasst dargestellt. Solche Gruppen zu definieren kann sich als sehr nützlich erweisen und dem Administrator einen geordneteren Überblick über festgelegte Hosts und Dienste verschaffen.

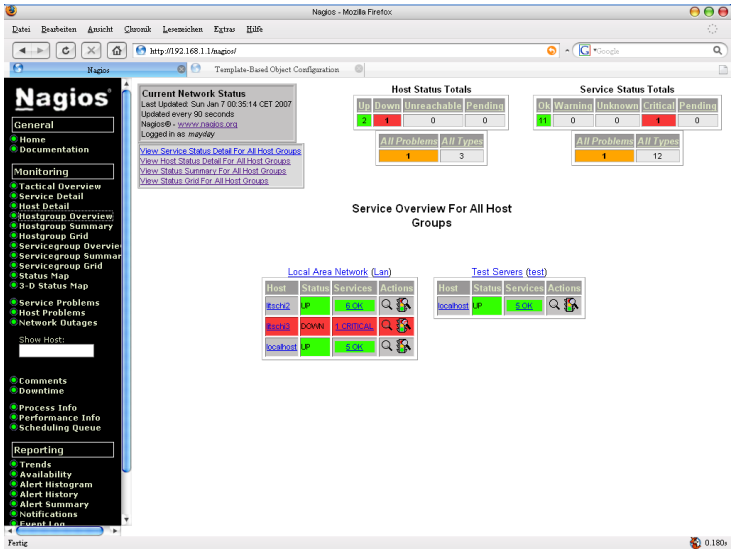

Abbildung 4: Definierte Host-Gruppen *Lan* und *Test*

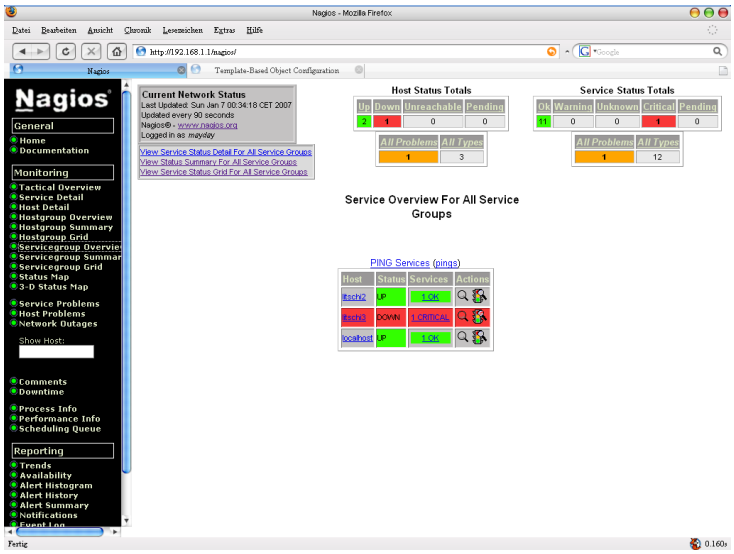

Abbildung 5: Definierte Service-Gruppe *pings*

4.3 Beschreibung der Weboberflächen-Funktionalität

Nachdem im vorigen Abschnitt eine Beispielkonfiguration erstellt wurde um grundlegende Eigenschaften der Weboberfläche zu zeigen, werden jetzt verstärkt die wichtigsten hier verfügbaren Funktionen betrachtet. Dazu werden die Beispielkonfiguration mit weiteren zu überwachenden Hosts und Services ausgebaut und auf einzelne Menüpunkte des Web-Frontends eingegangen.

4.3.1 Menüpunkt „Monitoring“

Hier enthaltene Funktionen können grob unter dem Schlagwort *Überwachung* zusammen gefasst werden. Mit ihnen lassen sich Hosts, Services und ganze Teil-Netzwerke überwachen.

Unterpunkt „Tactical Overview“

Das `tac.cgi`, welches über den Menüpunkt `Tactical Overview` erreichbar ist, gibt einen komprimierten Überblick über alle Netzwerkprobleme. So erhält ein Administrator schnell grundlegende Informationen über Ausfälle sowie den Status von Hosts und Services. Wenn viele Hosts und Services zu überwachen sind, kann hierdurch besonders schnell auf Probleme im vorhandenen Netzwerk aufmerksam gemacht werden. Abbildung 6 zeigt die „Tactical Overview“ im hier vorliegenden Beispiel. Im Kasten „Monitoring Performance“ wird dabei eine komprimierte Statistik zur Ausführung von Plugins angezeigt, die „Network Health“ zeigt anhand eines Balkendiagramms auf einen Blick, wie viele der überwachten Hosts und Dienste den Status OK haben. In den Kästen „Network Outages“, „Hosts“ und „Services“ werden jeweils Probleme mit (Teil-) Netzwerken, Hosts und Diensten aufgezeigt, unter „Monitoring Features“ findet sich schließlich ein Überblick über die in der Konfiguration aktivierten Optionen zur Überwachung.

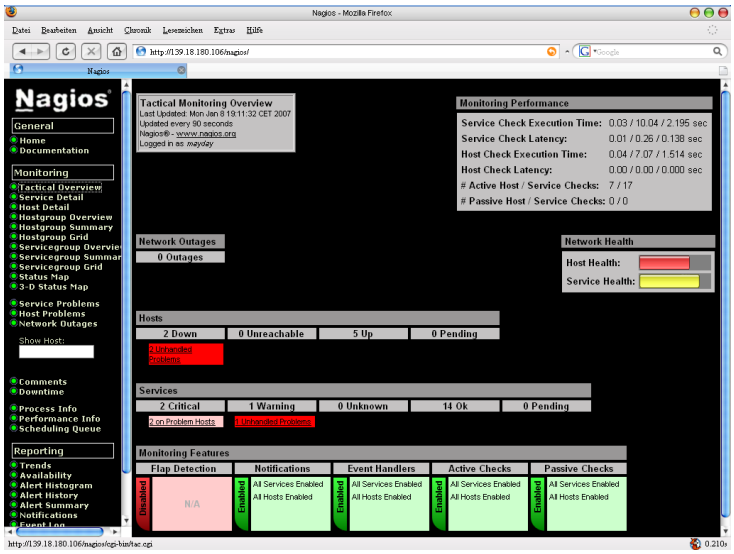

Abbildung 6: Weboberflächen-Menüpunkt „Tactical Overview“

Unterpunkte zu Hosts und Services

Details zu den überwachten Hosts und Services werden von dem Skript `status.cgi` aufbereitet und strukturiert dargestellt. Sie sind dann über 8 Menüpunkte von „Service Detail“ bis „Servicegroup Grid“ verfügbar. Dies stellt die wichtigste Funktion der Weboberfläche dar. Die Status aller Hosts und Services lassen sich hier einsehen. Weiterhin können detailliertere Informationen zu Hosts oder Services abgerufen werden um somit die Fehlersuche zu vereinfachen. Durch die Übergabe verschiedener Optionen besitzt das Skript mehrere Darstellungsvarianten, so können beispielsweise Hosts (Abbildung 7), Services (Abbildung 8), Hostgroups (Abbildung 9) sowie Servicegroups (Abbildung 10) angezeigt werden. In der jeweiligen Abbildung findet eine farbliche Hinterlegung der überwachten Objekte statt, sodass auf einen Blick ersichtlich ist, welcher Status momentan anliegt. „Grün“ steht dabei für den Status OK, „gelb“ gibt eine Warnung an und bei „rot“ gibt es ein kritisches Problem. Alle diese Abbildungen stellen die gleiche Information auf unterschiedlichen Ebenen und in unterschiedlichen Gruppierungen an, entweder verdichtet oder expandiert. Durch diese Flexibilität kann es einem Administrator ermöglicht werden, immer die für ihn wichtige Informationen im Blick zu halten und sich bei Bedarf tiefgründiger anzeigen zu lassen.

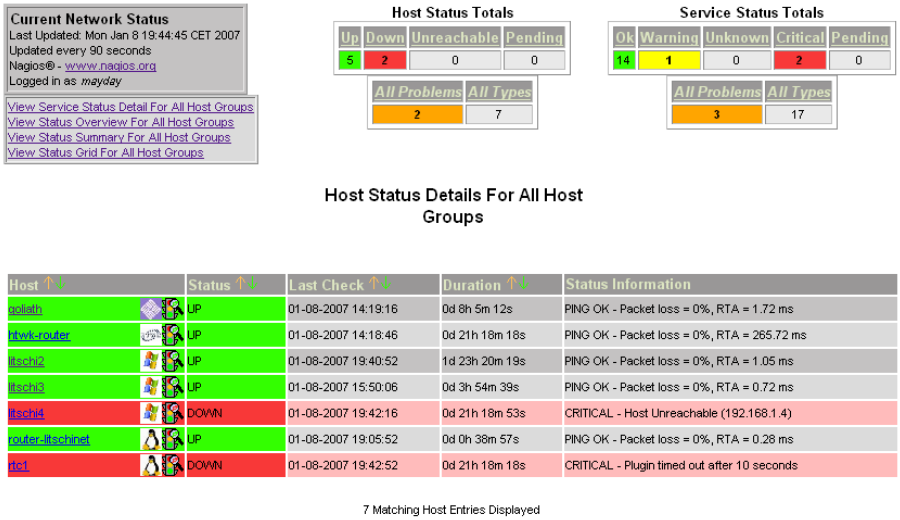

Abbildung 7: Menüpunkt „Host Detail“

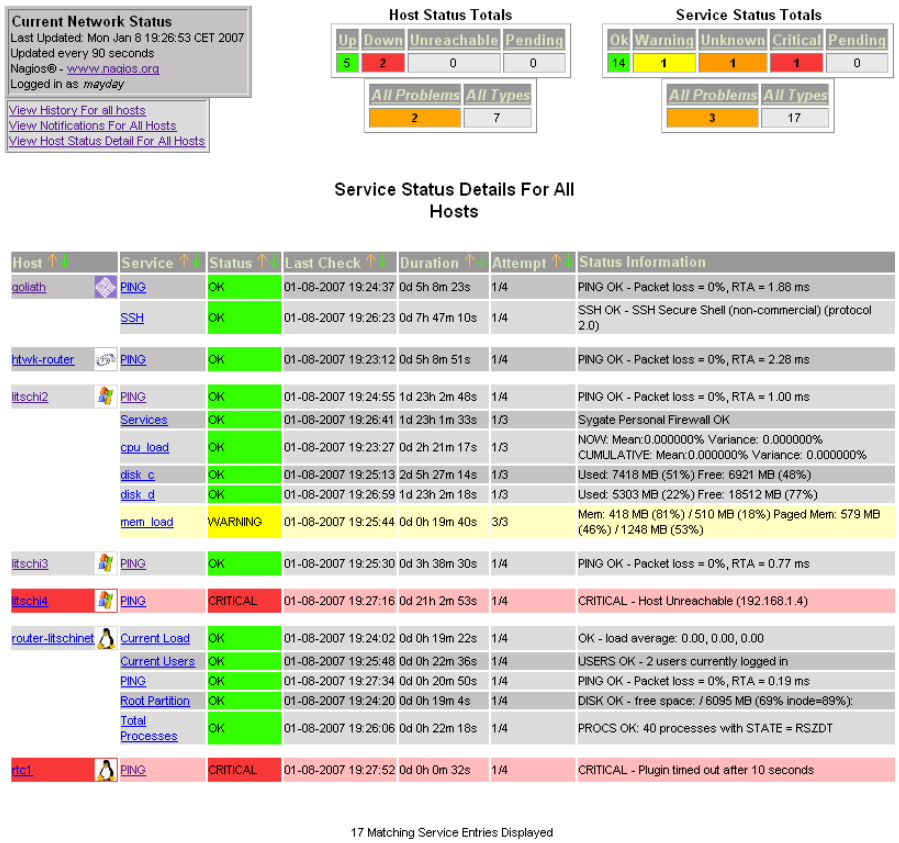

Abbildung 8: Menüpunkt „Service Detail“

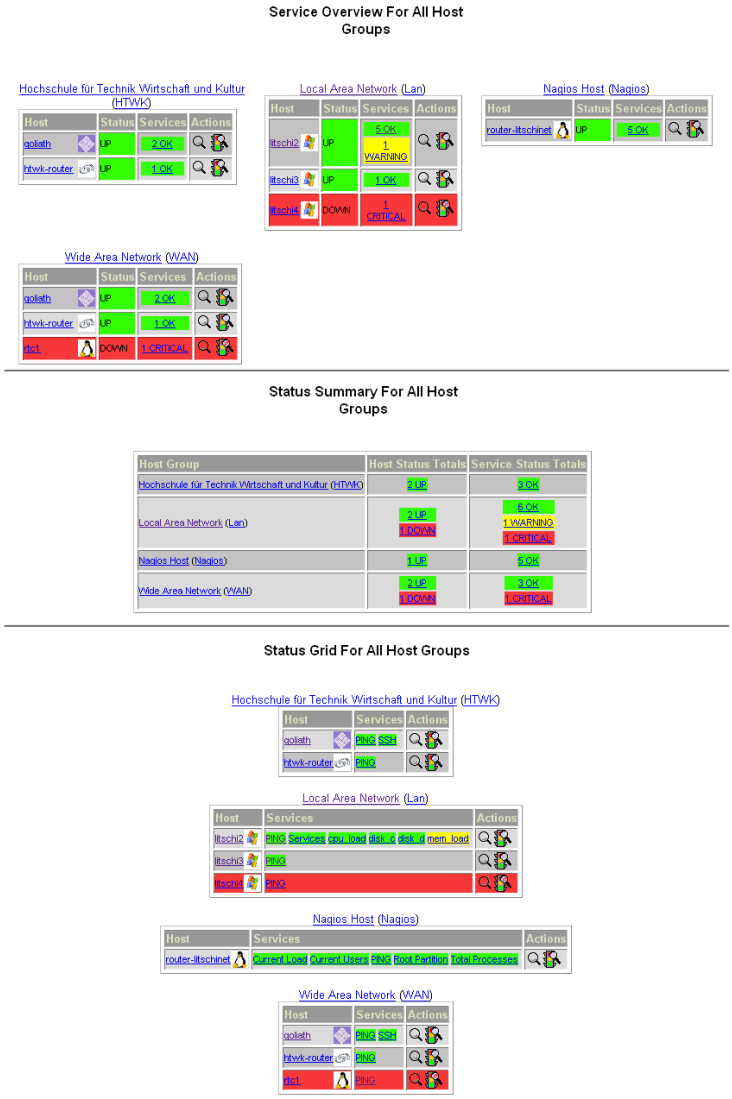

Abbildung 9: Menüpunkte zu Host Groups

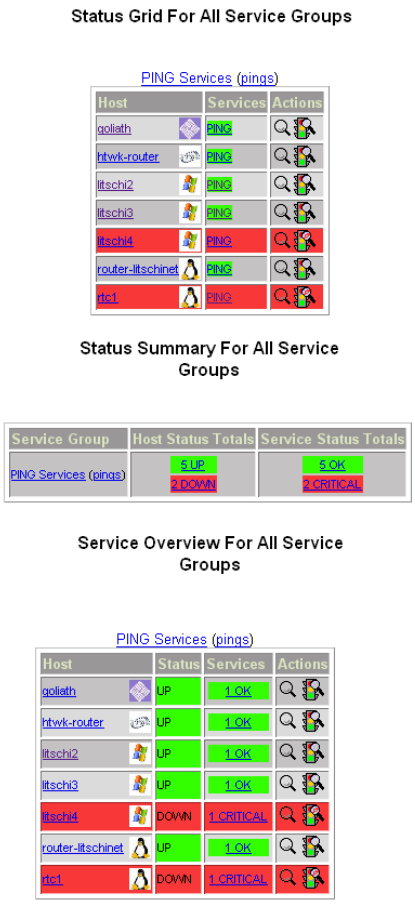

Abbildung 10: Menüpunkte zu Service Groups

Unterpunkt „Status Map“

Das Skript `statusmap.cgi` benutzt die gd library (siehe [8]) um ein Bild der vorhandenen Netzwerkstruktur zu erstellen. Dabei gibt es vordefinierte Layouts für die hierarchische Darstellung der Hosts und Netzwerke. Wem das nicht genügt, der kann mit einer `hostextinfo`-Definition ein individuelles Layout festlegen. So empfiehlt sich z. B. bei größeren Netzen eine baumartige Struktur. Abbildung 11 zeigt die Status Map für das hier verwendete Netzwerk. Im rechten oberen Teil des Browser-Fensters lässt sich die Darstellungsmethode ändern und auf einzelne Hostgruppen beschränken, im unteren Teil findet dann die eigentliche hierarchische Darstellung statt.

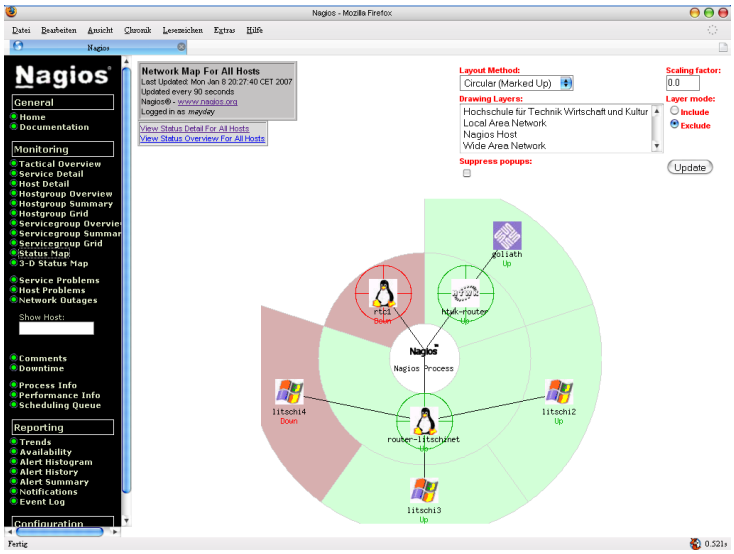

Abbildung 11: Menüpunkt „Status Map“

Für diejenigen, die sich mit einer zweidimensionalen Darstellung nicht zufrieden geben, hat das Nagios-Team eine 3D Status Map entworfen. Um das Skript im Browser richtig darstellen zu können, wird ein Plugin zur Anzeige von VRML-Dateien benötigt (für Windows z. B. „Cortona VRML“ von [4]). Wenn alles richtig eingerichtet wurde, erscheint die 3D-Karte wie in Abbildung 12 dargestellt im Browser. Am linken und unteren Rand der Map finden sich Knöpfe für die Navigation in dieser, wodurch sich die Darstellung des Netzes rechts beispielsweise rotieren und skalieren lässt.

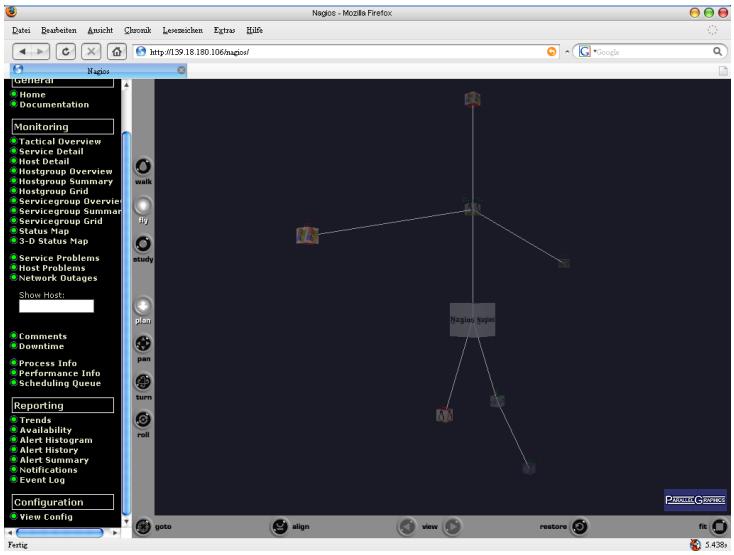

Abbildung 12: Menüpunkt „3-D Status Map“

Unterpunkte zu Netzwerk-Problemen

Das integrierte Network Outages CGI (`outages.cgi`) gibt einen Überblick über vorhandene Probleme im Netzwerk. Dies ist bei riesigen Netzwerken nützlich, bei denen der Überblick in den anderen Darstellungen schnell verloren gehen kann. Das CGI bedient die Menüpunkte `Service`

`Problems`, `Host Problems` und `Network Outages`, wovon `Host Problems` in Abbildung 13 zu sehen ist. Neben der Zusammenfassung von Hosts und Diensten im oberen Teil des Browser-Fensters werden im unteren Teil alle Hosts aufgelistet, mit denen es aktuell Probleme bei der Überwachung gibt.

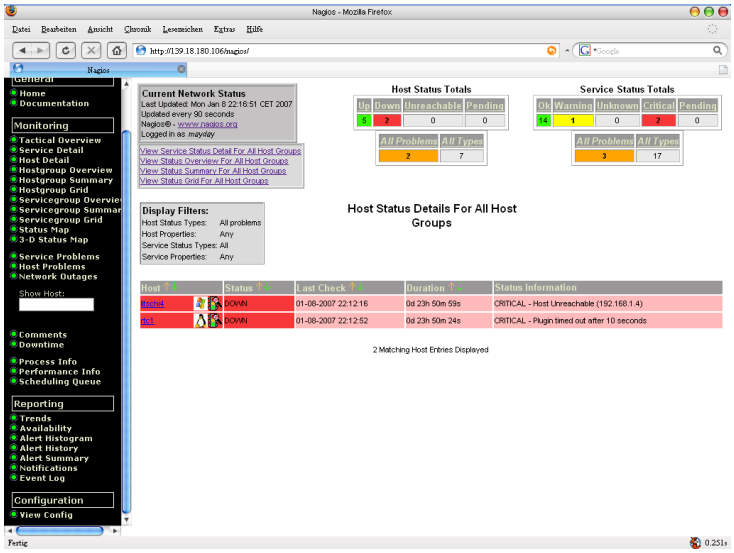

Abbildung 13: Menüpunkt „Host Problems“

4.3.2 Menüpunkt „Reporting“

Wie der Name schon andeutet werden durch diese Gruppe von Funktionen Berichte über mögliche Probleme überwachter Entitäten generiert. Dies geschieht über einen Zeitraum in der Vergangenheit und kann somit für Statistiken, aber auch zur Fehlersuche herangezogen werden.

Unterpunkt „Trends“

In dieser Übersicht (CGI `trends.cgi`) wird ein Graph generiert, welcher den Status von einzelnen Hosts oder Services in einem bestimmten Zeitraum darstellt. Abbildung 14 zeigt den Status des Hosts „litschi2“ über die letzten 7 Tage. Hiermit kann sich der Administrator darüber informieren, wann ein Host oder Service nicht erreichbar war und beispielsweise Rückschlüsse auf damit in Verbindung stehende Probleme ziehen. Im rechten Teil des Browser-Fensters ist ein Zeitstrahl zu sehen, der übersichtlich die Erreichbarkeit des jeweiligen Hosts darstellt, unter „State Breakdowns“ sind alle Host-Status noch einmal prozentual aufgelistet.

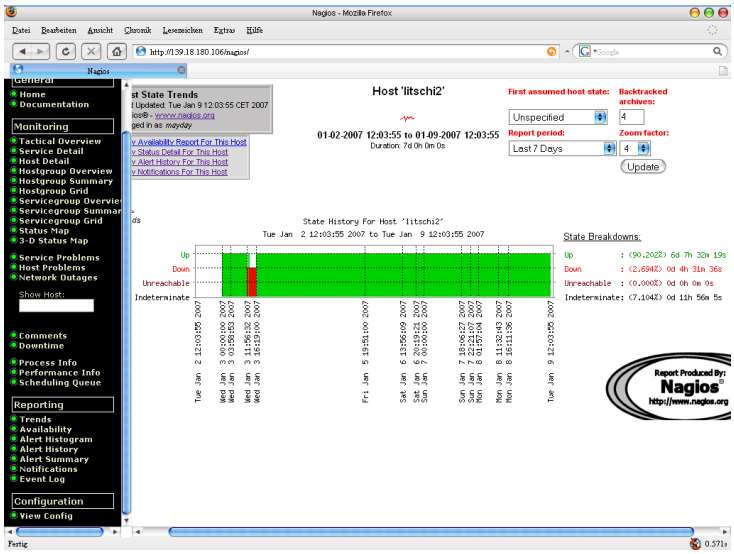

Abbildung 14: Menüpunkt „Trends“ für Host `litschi2`

Unterpunkt „Availability“

Dieser Menüpunkt (CGI `avail.cgi`) informiert über die Erreichbarkeit von Hosts, Services, Hostgroups und Servicegroups. Die Daten werden als eine Art Statistik aufbereitet und können z. B. der Fehlerbehebung im Netzwerk oder der Erstellung von Erreichbarkeits-Übersichten dienen. Abbildung 15 zeigt die Statistik für alle Hostgroups im Zeitraum der letzten 7 Tage. Dabei wird jede Hostgroup einzeln dargestellt, für jeden Host darin findet eine prozentuale Status-Angabe statt, die letzte Zeile einer jeden Gruppe dient der Mittelwert-Bildung über alle enthaltenen Hosts.

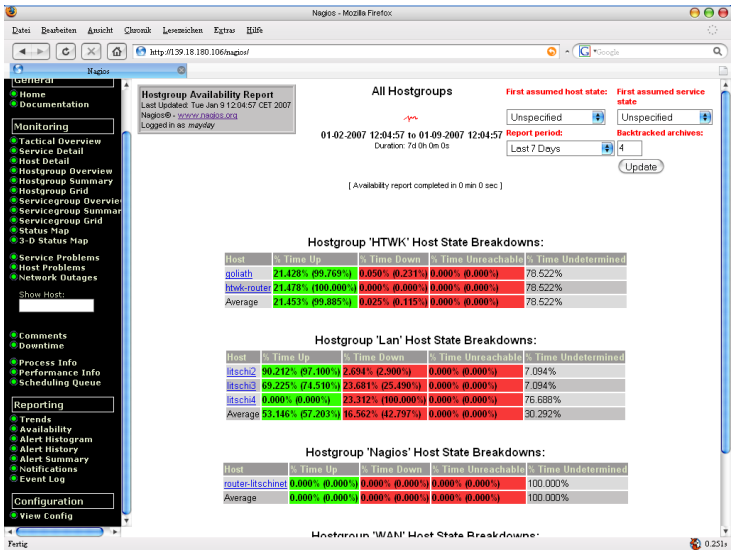

Abbildung 15: Menüpunkt „Availability“ für alle Hostgroups

Alert-Unterpunkte

Die Alert CGI Skripte (`histogram.cgi`, `history.cgi`, `summary.cgi`) zeigen ín tabellarischer Form Probleme des ganzen Netzwerkes in der Vergangenheit an.

Das **Alert Histogram** (CGI `histogram.cgi`) zeigt ein Diagramm über die Erreichbarkeit einzelner Hosts und Services, wie in Abbildung 16 für den Host `litschi2` zu sehen ist. Dabei wird tagesweise die Anzahl der Ereignisse dargestellt, für jeden Ereignis-Typ wird zudem aufgelistet, wie oft er minimal, maximal und im Durchschnitt eingetreten ist.

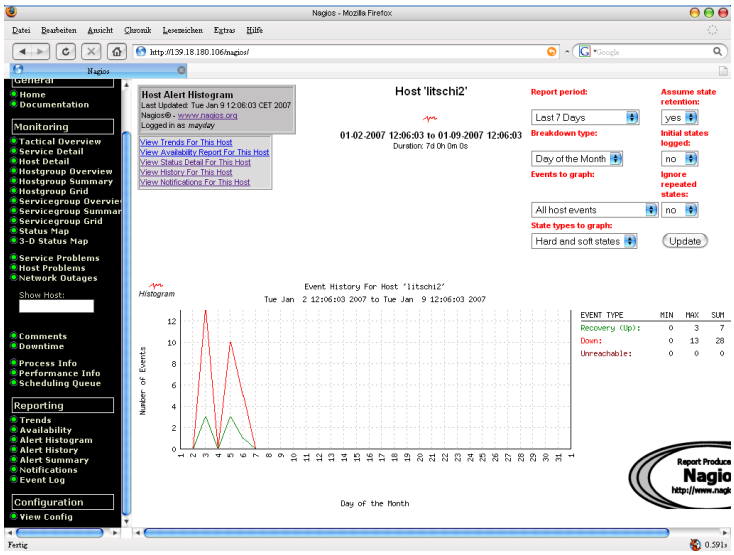

Abbildung 16: Menüpunkt „Alert Histogram“ für Host `litschi2`

Die **Alert History** (CGI `history.cgi`) stellt alle Probleme im vorhandenen Netzwerk nach Tagen gruppiert sehr übersichtlich in sequentieller Form dar und zeigt auch Informationen zu den einzelnen Problemen an (Abbildung 17).

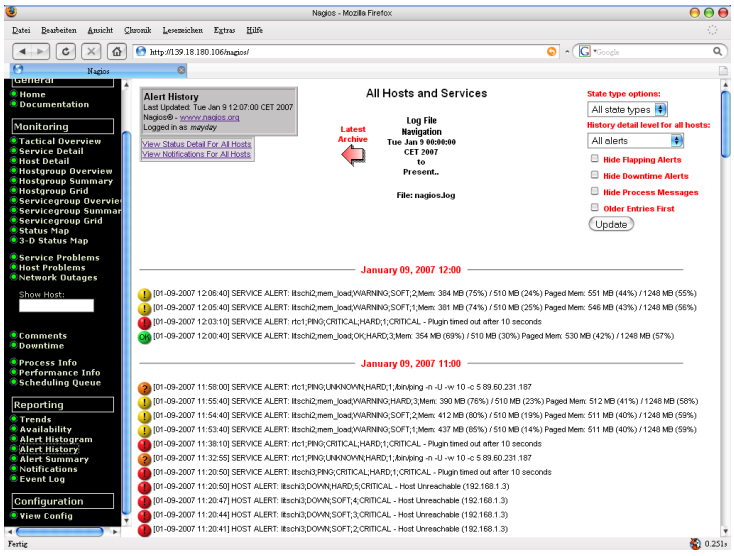

Abbildung 17: Menüpunkt „Alert History“

Unter **Alert Summary** (CGI `summary.cgi`) lassen sich zusammenfassende Berichte über Host- und Service-Alarmierungen erstellen, z. B. Gesamtanzahl von Alarmierungen und größte Problemverursacher. Letzteres ist in Abbildung 18 dargestellt. Entsprechend einer Rangliste werden die größten Problemverursacher und die Anzahl der von ihnen ausgelösten Alarme dargestellt.

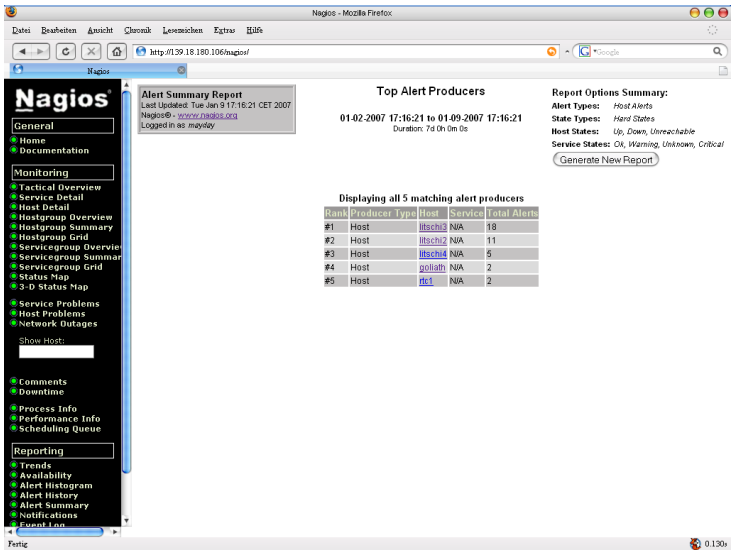

Abbildung 18: Menüpunkt „Alert Summary“ für die 25 größten Problemverursacher

Abschließend lässt sich zusammen fassen, dass die Weboberfläche ein optionales Werkzeug des Nagios-Paketes ist, auf das ein Administrator schwerlich verzichten kann. Durch die vielen erstellbaren Übersichten und Zusammenfassungen lassen sich Informationen auf den Punkt verdichten, im Bedarfsfall aber auch detaillierter expandieren. Die unter dem Menüpunkt „Reporting“ verfügbaren Übersichten stellen zudem komfortable Statistiken bereit, die für Berichte und zur Analyse von Netzwerkproblemen verwendet werden können.

4.4 Plugins

Wie bereits in Abschnitt 3.3 angeklungen ist, wird erst durch Plugins die Überwachungs-Funktionalität von Nagios zur Verfügung gestellt. Bei Plugins handelt es sich um kompilierte Programme oder um Skripte, die von der Kommandozeile aus gestartet werden können. Jedes Plugin, das sich an die Guideline des Projekts hält, stellt eine Beschreibung seiner selbst über die Kommandozeilenoption `-h` zur Verfügung. Weiterhin ist die Entwicklung eigener Plugins problemlos möglich.

Die Plugin-Philosophie von Nagios steht im Gegensatz zu vielen anderen Überwachungs-Tools, welche die angebotene Funktionalität gleich mitliefern, somit aber teilweise auch einen begrenzten Funktionsumfang besitzen. Bei Nagios lassen sich beliebige Objekte überwachen, solange ein Plugin dafür zur Verfügung steht. Der Nachteil von Plugins besteht in der abstrakten Sicht, die Nagios darauf hat. Nagios weiß nicht was durch ein Plugin überwacht wird und kann so auch keine Graphen zu Ressourcen erstellen, die durch ein Plugin möglicherweise observiert werden. Es kann nur Informationen zu Statusänderungen auswerten und darauf reagieren.

4.5 Addons

Neben Plugins gibt es Addons, die weitere grundlegende Funktionalitäten für Nagios bereitstellen und bei denen es sich im Gegensatz zu Plugins um eigenständige Programmpakete handelt, die entweder auf dem Nagios-Host, auf den zu überwachenden Rechnern oder mit einzelnen Komponenten auf beiden Seiten zu installieren sind. Die folgenden beiden Addons werden auf der

Nagios-Downloadseite (siehe [9]) bereitgestellt und finden in vielen größeren Umgebung Verwendung:

`NRPE`:

Das Ziel dieses Addons ist die Ausführung lokaler Nagios Plugins auf entfernten (Linux/Unix) Rechnern. Dazu wird auf dem entfernten Rechner der NRPE-Dienst gestartet und vom Nagios-Prozess mittels eines speziellen Plugins (`check_nrpe`) geprüft.

`NSCA`:

Mit diesem Addon lassen sich Resultate von Service-Checks über eine gesicherte Verbindung an einen zentralen Überwachungs-Server senden, auf dem Nagios zur Auswertung verwendet wird. Entgegen der aktiven Prüfung durch den Nagios-Server findet hier also eine passive Prüfung statt, bei welcher der Client den Server kontaktiert.

Weitere Addons umfassen das Management von Konfigurationsdateien via Web-Browser, Integration in diverse Applikationen sowie Frontend-Erweiterungen, um nur drei zu nennen. Eine Übersicht verfügbarer Addons gibt die Community-Seite „Nagios Exchange“ (siehe [10]). Auch die im Abschnitt 5 beschriebenen Pakete zur Windows-Überwachung stellen (bis auf SNMP) Addons für Nagios dar.

4.6 Überwachungs-Arten

Generell lässt sich zwischen aktiver und passiver Überwachung unterscheiden:

Aktive Überwachung:

Dies stellt die Standardmethode zur Überwachung von Diensten auf Netzwerkgeräten dar. Dabei führt der Nagios-Host periodische Prüfungen der jeweiligen Dienste durch, d. h. er selbst stellt *aktiv* eine Verbindung zu dem gewünschten Gerät her und löst eine Prüfung aus. Der Nachteil besteht hier in der notwendigen Installation von Diensten auf den Netzwerkgeräten, über die Informationen von außen abgerufen werden können, was ein nicht unerhebliches Sicherheitsrisiko darstellt.

Passive Überwachung:

Passive Überwachung wird ermöglicht, wenn auf dem Nagios-Host das NSCA-Addon (siehe 4.5) installiert ist. Dann können Netzwerkgeräte Ergebnisse von Prüfungen an den NSCA-Dämon auf dem Nagios-Host senden, d. h. die Aktivität der Ergebnis-Ermittlung und -Übertragung liegt bei dem zu prüfenden Gerät, der Nagios-Prozess selbst nimmt diese nur *passiv* entgegen. Passive Prüfungen können notwendig sein, wenn sich ein Zielgerät hinter einer Firewall verbirgt und somit keine aktiven Verbindungen zu einem bereitgestellten Dienst auf diesem Gerät möglich sind oder wenn Prüfergebnisse nur asynchron bereit gestellt werden können. Der Nachteil von passiver Überwachung liegt im dezentralen Konfigurationsaufwand der einzelnen Prüfungen auf den jeweiligen Netzwerkgeräten.

5 Überwachung von Windows-Rechnern

Windows ist in der heutigen Rechnerlandschaft weit verbreitet und deswegen muss ein gutes Überwachungs-Werkzeug auch windowsbasierte Systeme umfassend einbeziehen können. Vor allem da Nagios für Unix konzipiert wurde, ist eine Untersuchung der Möglichkeiten zur Integration von Windows-Rechnern von großem Interesse.

Der folgende Abschnitt stellt mehrere Möglichkeiten der Windows-Überwachung vor und beschreibt kurz ihre Funktionalität. Dazu wird ein Szenario erstellt, welches mit allen Varianten umzusetzen ist. In Abschnitt 5.7 werden die Pakete einzeln bewertet um in 5.8 einen Vergleich durchführen zu können. Bei den angegebenen Konfigurationen der einzelnen Lösungen auf dem Nagios-Host werden die Dateien `hosts.cfg` sowie `commands.cfg` editiert. Die Syntax der dabei vorgenommenen Einstellungen wurde bereits in Abschnitt 4.2 erläutert, sodass hier nicht mehr darauf eingegangen wird.

5.1 Überwachungs-Szenario

Folgende Daten sind auf einem Windows-System zu überwachen. Über den Sinn dieser Elemente soll dabei nicht weiter nachgedacht werden, wichtig ist vielmehr, ob die vorgestellten Varianten flexibel genug sind, um in der Lage zu sein diese Probleme zu lösen. Die vorangestellte Abkürzung in Großbuchstaben gibt eine Marke an, um sich in der nachfolgenden Darstellung schnell auf das jeweilige Datum beziehen zu können. Zudem werden Schranken definiert, bei denen Warnungen bzw. kritische Meldungen auszulösen sind:

1. `CURCPU`: Aktuelle CPU-Last; Warnungs-Marke: 90%, kritische Marke: 95%,
2. `MIDCPU`: CPU-Last im Mittel über die letzten 15 Minuten; Warnungs-Marke: 80%, kritische Marke: 90%,
3. `CURRAM`: Aktuell belegter Arbeitsspeicher; Warnungs-Marke: 85%, kritische Marke: 95%,
4. `CURPAGE`: Aktuelle Belegung der Auslagerungsdatei; Warnungs-Marke: 70%, kritische Marke: 80%,
5. `DISKC`: Belegung der Festplatte C: ; Warnungs-Marke: 80%, kritische Marke: 90%,
6. `PERFMON`: Perfmon-Objekt „\System\Prozesse“ (Anzahl laufender Prozesse); Warnungs-Marke: 50, kritische Marke: 100,
7. `SERVICES`: Aktivität der Dienste *AntiVir PersonalEdition Classic Service (AntivirService)*, *AntiVir Scheduler (AntiVirScheduler)*, *Sygate Personal Firewall (SmcService)*, *Automatische Updates (wuauserv)* und *DHCP-Client (Dhcp)*; kritische Meldung, sobald ein Dienst nicht aktiv ist.

Zum Einsatz kommt auf Client-Seite als Betriebssystem „Windows XP Professional SP2“, welches auf dem Rechner „rtc3“ über eine 100MBit-Ethernet-Verbindung an den Nagios-Host angeschlossen ist. Auf dem Nagios-System sollen als Objektdefinitions-Dateien lediglich `commands.cfg` für Kommandos und `hosts.cfg` für zu überwachende Dienste und Rechner genutzt werden. Die durch die Nagios-Installation erstellte Beispielkonfiguration sollte diese Dateien bereits beinhalten und sie können als Ausgangspunkt für die nachfolgenden Erläuterungen dienen.

5.2 SNMP

SNMP (*Simple Network Management Protocol*) ist ein Netzwerkprotokoll zur Überwachung (und Steuerung) von Entitäten eines Netzwerks von einem zentralen Rechner aus. Die Werte von zu überwachenden Elementen auf einem Netzwerkgerät werden dabei in der *Management Information Base* (MIB) abgespeichert und können auf Anfrage abgerufen bzw. verändert werden. Die in einer MIB enthaltenen Informationen können als Baumstruktur dargestellt werden, überwachte Objekte lassen sich durch Angabe der Position in diesem Baum (OID) adressieren. Hier kann und soll nicht umfassend auf die Arbeitsweise von SNMP eingegangen werden, für weitere Hilfe bieten sich die entsprechenden RFCs als Informationsquellen an (eine Liste findet sich unter [20]).

Da SNMP großflächig zur Überwachung von Netzwerkgeräten eingesetzt wird und auch Windows von Haus aus einen SNMP-Dienst anbietet, stellt die Integration in Nagios einen interessanten Aspekt der Überwachung von Windows-Rechnern dar.

5.2.1 Konfiguration des Windows-Clients

Windows implementiert schon seit den ersten Tagen von Windows NT das SNMP-Protokoll. Um Informationen über SNMP von einem Windows-Rechner abrufen zu können, muss auf diesem zunächst der „SNMP-Dienst“ als Windows-Komponente installiert sein. Daraufhin lässt er sich in der Dienste-Verwaltung (Ausführen von `services.msc`) konfigurieren und starten. Am wichtigsten ist dabei in der Registerkarte *Sicherheit* das Einstellen von akzeptierten Communitynamen und von Hosts, welche zum Abrufen von Informationen über SNMP berechtigt sind (siehe Abbildung 19). Communitynamen müssen angegeben werden, wenn sich ein Rechner mit dem SNMP-Dienst verbindet und Informationen abfragen will. Der jeweilige Name der Community gibt dabei an, mit welchen Rechten sich Rechner verbinden dürfen. Hier sei angemerkt, dass die dadurch erreichte Sicherheit lediglich als rudimentär bezeichnet werden kann bzw. keinen großen Beitrag leistet. Wenn z. B. keine erlaubte Host-Liste gepflegt wird, so kann jeder Rechner, der einen konfigurierten Communitynamen kennt, Informationen mit den dabei verteilten Rechten abrufen, ein zusätzliches Passwort wird nicht benötigt. Es kann zwar versucht werden, den Communitynamen so kompliziert wie möglich zu vergeben, da die Kommunikation jedoch unverschlüsselt stattfindet, lässt sich dieser Name durch Abhören der Verbindung schnell identifizieren. Selbst wenn die Hostliste begrenzt wird, lassen sich durch kombiniertes IP/ARP-Spoofing Angriffe über SNMP fahren und Klartext-Informationen auslesen, das Abfangen dieser ist ohnehin möglich.

Die Abfrage des Perfmon-Objekts „Prozesse“ bedarf etwas Handarbeit, da von Haus aus kein Abfragen von Performance Countern mittels SNMP möglich ist. Auf [1] steht allerdings eine Anleitung nebst benötigter Dateien bereit, womit man Perfmon in SNMP einbinden kann. Hier soll eine Zusammenfassung der benötigten Schritte gegeben werden. Zunächst gilt es dafür notwendige Dateien herunter zu laden. Unter [6] ist die von Microsoft bereitgestellte Bibliothek `perfmib.dll` verfügbar, die für die SNMP-Integration sorgt. Das ZIP-Archiv unter [5] enthält weitere notwendige Dateien, am wichtigsten ist hierbei die Datei `perfmib.ini`, welche die Daten beinhaltet, die über SNMP bereitgestellt werden sollen. Nützlich ist auch die Datei `perfmib.mib`, über die gängige MIB-Browser auf die jeweiligen Objekte zugreifen können. Im nächsten Schritt sind die beiden Dateien `perfmib.dll` und `perfmib.ini` in das Verzeichnis „System32“ des Windows-Ordners zu kopieren. Nachfolgend muss SNMP die neue DLL bekannt gemacht werden. Dazu editiert man die Windows-Registrierung (Ausführen von `regedit`) und fügt die folgenden Schlüssel hinzu:

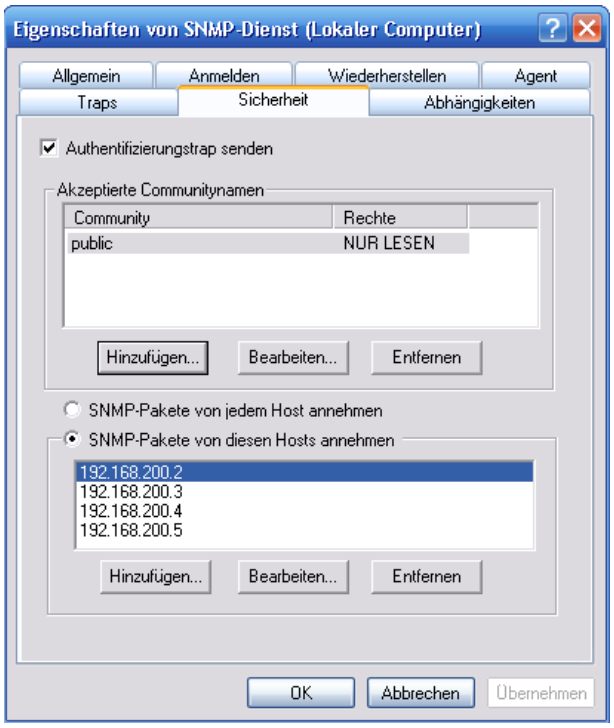

Abbildung 19: Konfiguration des SNMP-Dienstes auf dem Windows-Client

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\
  Parameters\ExtensionAgents]
 Neu->Zeichenfolge:  nächste freie Nummer in Liste
 Wert ändern: "SOFTWARE\Microsoft\PerformanceAgent\CurrentVersion"

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PerformanceAgent\CurrentVersion]
 Neu->Wert der erweiterbaren Zeichenfolge : "Pathname"
 Wert ändern: "%SystemRoot%\System32\perfmib.dll"
```

Damit ist die Konfiguration abgeschlossen und nach einem Neustart des SNMP-Dienstes kann über die OIDs .1.3.6.1.4.1.311.1.1.3.1.1.* (.iso .identified-organization .dod .internet .private .enterprises .microsoft .software .ms-systems .os .winnt .performance .*) auf die Perfmon-Objekte zugegriffen werden.

5.2.2 Konfiguration des Nagios-Hosts

Die Hauptlast der Konfiguration liegt beim Nagios-Host, der die einzelnen Werte über SNMP abrufen muss. Eine grundlegende Möglichkeit hierfür bietet das Standard-Nagios-Plugin `check_snmp`, welchem man eine OID übergibt, deren Wert dann zurück gegeben wird. Allerdings stellt dieses Plugin eher einen Test von SNMP dar, hingegen ist für tatsächliche Daten eine Aufbereitung der abgerufenen Werte nötig, was eigenständige Plugins erfordert. Hier sollen zur Lösung des vorgestellten Szenarios die Plugins `check_snmp_storage.pl`, `check_snmp_load.pl`, und `check_snmp_win.pl` zum Einsatz kommen, welche auf [19] zur Verfügung stehen. Die Nutzung von `check_snmp_mem.pl` wäre ebenfalls wünschenswert gewesen, da hierdurch der Arbeitsspeicher und ausgelagerte Daten überwacht werden können. Leider ist dieses Plugin nicht mit der

Windows-SNMP-Implementierung kompatibel und scheidet somit aus.

Zur Umsetzung der einzelnen Objekte des Szenarios: CURCPU könnte mittels `check_snmp_load.pl` realisiert werde, welches die Prozessorauslastung von SNMP abruft. Dabei gibt der in SNMP verfügbare Wert die mittlere Prozessoraktivität über den Zeitraum von 1 Minute an, es handelt sich also nicht wie gefordert um einen Momentanwert. Dieser kann nur durch Abfrage eines Perfmon-Objekts über SNMP ermittelt werden, was nach der Integration von Perfmon wie in 5.2.1 beschrieben kein Problem darstellt. Die OID für die Abfrage der Prozessorauslastung lautet .1.3.6.1.4.1.311.1.1.3.1.1.6.1.3.1.48. Im Anhang B.1 ist der Quelltext des selbst entwickelten Shell-Skripts `check_snmp_win_curcpu` zu finden, welches den tatsächlichen momentanen Wert für die Belegung des Prozessors zurückgibt.

Durch eine Direktabfrage von SNMP gar nicht lösen lässt sich MIDCPU, da ein freies Zeitintervall für Mittelwerte nicht angegeben werden kann. Es ist auch auf Seite des Nagios-Hosts kein einfaches Plugin vorstellbar, welches diese Funktionalität realisiert. Es müsste z. B. minütlich über das Plugin `check_snmp_load.pl` eine Prüfung der CPU-Last erfolgen um dann nach 15 Minuten einen Mittelwert bilden zu können und an Nagios zurückzugeben. Da Plugins aber nicht minütlich vom Nagios-Host aufgerufen werden, ist solch eine Lösung nicht denkbar. Eine Möglichkeit ist die Entwicklung eines eigenständigen Programms, welches im Hintergrund läuft, eine minütliche Abfrage startet und die Daten in einer Datei speichert, auf die dann ein Nagios-Plugin zugreift. Da hier aber nur die grundlegende Flexibilität einzelner Varianten untersucht werden soll, wird auf eine solche Lösung nicht weiter eingegangen.

CURRAM kann über `check_snmp_storage.pl` realisiert werden, da sich über dieses Plugin neben Festplatten auch die Belegung von Arbeitsspeicher und virtuellem Speicher prüfen lässt. Dasselbe Plugin wird über andere Parameter für DISKC benutzt.

Für die Implementierung von SERVICES lässt sich direkt das Plugin `check_snmp_win.pl` nutzen, dem eine Liste der zu prüfenden Windows-Dienste übergeben wird.

Das Abrufen des Perfmon-Objekts „Prozesse“ für die Anzahl gestarteter Prozesse ist nach der Konfiguration des Performance Monitors für SNMP problemlos über die OID .1.3.6.1.4.1.311.1.1.3.1.1.4.12.0 möglich, allerdings ist für die Aufbereitung der Ausgabe ein eigenes Skript zu schreiben, welches auch die korrekten Meldungen an Nagios zurückgibt. Das hierfür entwickelte Shell-Skript `check_snmp_win_procs` ist in Anhang B.2 zu finden.

Die Realisierung von CURPAGE bereitet ebenfalls einige Probleme. Sowohl Arbeitsspeicher als auch virtueller Speicher lassen sich mittels SNMP abrufen, jedoch nicht die Belegung der Auslagerungsdatei. Approximativ lässt sich diese als Differenz von belegtem virtuellen Speicher und Arbeitsspeicher betrachten, die sich beide über SNMP abrufen lassen. Das im Rahmen dieser Arbeit entwickelte Plugin `check_snmp_win_page` (siehe Anhang B.3) realisiert diese Idee. Dabei ergeben sich jedoch keine genauen Werte für die Größe und quantitative Belegung der Auslagerungsdatei, da der abgerufene „virtuelle Speicher“ lediglich den Speicher darstellt, der den Benutzerprozessen zur Verfügung steht, dabei fehlen allerdings die vom System (Kernel und Treiber) belegten Ressourcen. Die prozentuale Belegung stimmt indes trotzdem annäherungsweise mit dem tatsächlichen Wert überein, sodass dieser Lösungsweg hier ausreicht.

Nachdem für jedes Datum des Szenarios (mit Ausnahme von MIDCPU) Plugins zum Erhalten der erforderlichen Werte ermittelt bzw. entwickelt wurden, ist Nagios so zu konfigurieren, dass es die neuen Kommandos erkennt und periodisch damit definierte Dienste überprüft. Dies erfordert die Modifikation der verwendeten Konfigurationsdateien `commands.cfg` und `hosts.cfg`.

In `commands.cfg` wird die Syntax zum Aufruf der jeweiligen Plugins eingetragen, wohingegen in `hosts.cfg` die Kommandos als Dienste definiert und an die zu überprüfenden Hosts zugewiesen werden.

Folgende Änderungen werden in die beiden Dateien eingefügt, um den Host „rtc3“ zu überwachen:

Datei `commands.cfg`:

```
# SNMP check commands
# get very current CPU load
define command{
  command_name  check_snmp_win_curcpu
  command_line  $USER1$/check_snmp_win_curcpu $HOSTADDRESS$ $ARG1$ $ARG2$
}
# get current (physical) mem load
define command{
  command_name  check_snmp_memload
  command_line  $USER1$/check_snmp_storage.pl -2 -H $HOSTADDRESS$ \
              -C public -m Physical -w $ARG1$ -c $ARG2$
}
# get current pagefile usage
define command{
  command_name  check_snmp_win_page
  command_line  $USER1$/check_snmp_win_page $HOSTADDRESS$ $ARG1$ $ARG2$
}
# get usage of disk C:
define command{
  command_name  check_snmp_diskc
  command_line  $USER1$/check_snmp_storage.pl -2 -H $HOSTADDRESS$ \
              -C public -m C -w $ARG1$ -c $ARG2$
}
# get process count
define command{
  command_name  check_snmp_win_procs
  command_line  $USER1$/check_snmp_win_procs $HOSTADDRESS$ $ARG1$ $ARG2$
}
# check services running
define command{
  command_name  check_snmp_services
  command_line  $USER1$/check_snmp_win.pl -2 -H $HOSTADDRESS$ -C public -n $ARG1$
}
```

Den Plugins „check_snmp_win_curcpu“, „check_snmp_win_page“ und „check_snmp_win_procs“ werden als Argumente die Host-Adresse und die prozentualen Marken für Warnungen und kritische Meldungen übergeben. „check_snmp_storage.pl“ erhält über die Option `-H` die Adresse des zu prüfenden Hosts und über `-C` den zu benutzenden Community-Namen. Über `-m` wird festgelegt, welcher Speicher-Typ geprüft werden soll und mit `-w` und `-c` lassen sich die Marken für Warnungen und kritische Meldungen festlegen. Die Option `-2` gibt an, dass SNMPv2 verwendet werden soll. Dem Plugin „check_snmp_win.pl“ werden dieselben Optionen `-2`, `-H` und

`-C` übergeben, weiterhin wird jedoch über `-n` festgelegt, welche Windows-Dienste auf Aktivität zu prüfen sind.

Datei `hosts.cfg`:

```
# SNMP services
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             SNMP: check current CPU load
        check_command                   check_snmp_win_curcpu!90!95
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             SNMP: check physical memory
        check_command                   check_snmp_memload!85!95
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             SNMP: check pagefile usage
        check_command                   check_snmp_win_page!70!80
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             SNMP: check usage of disk C:
        check_command                   check_snmp_diskc!80!90
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             SNMP: check process count
        check_command                   check_snmp_win_procs!50!100
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             SNMP: check Windows services
        check_command                   check_snmp_services!\
          "AntiVir PersonalEdition Classic Service,AntiVir Scheduler,\
           Sygate Personal Firewall,Automatische Updates,DHCP-Client"
}
```

Im laufenden Betrieb von Nagios sieht das auf der Weboberfläche dann wie in Abbildung 20 zu sehen aus. Neben dem Host und den überwachten Diensten werden der Status, der Zeitpunkt der letzten Prüfung sowie die verstrichene Zeit seitdem, die Anzahl von Prüf-Versuchen sowie die Beschreibung der Prüfung (i. Allg. die zurück gelieferten Ergebnisse textuell aufbereitet) dargestellt.

rtc3	SNMP: check Windows services	OK	05-01-2007 13:58:34	0d 0h 6m 24s	1/4	5 services active (matching "Anti\Vir PersonalEdition Classic Service,Anti\Vir Scheduler,Sygate Personal Firewall,Automatische Updates,DHCP-Client") : OK
	SNMP: check current CPU load	OK	05-01-2007 13:54:55	0d 0h 5m 3s	1/4	cpuload: 39% - OK
	SNMP: check pagefile usage	OK	05-01-2007 13:56:16	0d 0h 3m 42s	1/4	swap usage: 132 of 654 MB (20% used) - OK
	SNMP: check physical memory	OK	05-01-2007 13:57:37	0d 0h 2m 21s	1/4	Physical Memory: 73%used(356MB/487MB) (<85%) : OK
	SNMP: check process count	OK	05-01-2007 13:58:46	0d 0h 6m 12s	1/4	process count: 47 - OK
	SNMP: check usage of disk C:	OK	05-01-2007 13:55:07	0d 0h 4m 51s	1/4	C:\ Label: Serial Number 78b415b1: 57%used(16447MB/28608MB) (<80%) : OK

Abbildung 20: Konfigurierte SNMP-Abfragen

5.3 NRPE_NT

Wie in 4.5 bereits kurz angeklungen ist, handelt es sich bei NRPE um ein Nagios-Addon, welches das Ausführen lokaler Plugins auf entfernten Rechnern erlaubt. Dabei werden jedoch zunächst nur Unix-Derivate unterstützt, auf denen der NRPE-Dämon gestartet wird, Windows bleibt außen vor. Diese Lücke wird durch NRPE_NT geschlossen, welches mit der Absicht entwickelt wurde, die volle NRPE-Funktionalität für Windows-Rechner zur Verfügung zu stellen.

NRPE_NT stellt auf dem zu überwachenden Windows-System einen Dienst bereit, der die Aufgaben des NRPE-Dämons von Unix übernimmt. Weiterhin werden durch Plugins ausführbare Programme zur Verfügung gestellt, die Überwachungsfunktionen implementieren und Resultate an den Nagios-Host senden können, der diese über das Host-Plugin `check_nrpe` entgegen nimmt. Die Konfiguration der an ein Plugin zu übergebenden Argumente kann dabei entweder auf Client- oder Host-Seite erfolgen. Werden die Argumente vom Nagios-Host übergeben, so ist dies die flexibelste Lösung, da bei einer Änderung die Konfiguration auf einem NRPE_NT-Client gleich bleibt. Allerdings stellt dies auch ein Sicherheitsrisiko dar, wenn Hosts beliebige Argumente an bereitgestellte Plugins des Dienstes übergeben können. Die Liste erlaubter Hosts lässt sich zwar wie bei SNMP begrenzen, dieser Schutz lässt wie dort auch durch IP- und ARP-Spoofing jedoch aushebeln.

Vorteile von NRPE_NT bestehen in der Nutzung von OpenSSL und damit einer Verschlüsselung der Verbindung zwischen Client und Nagios-Host sowie in der leichten Erweiterbarkeit des Pakets durch Plugins. Fehlt ein benötigtes Plugin, so lässt sich über die allgemeine Plugin-Schnittstelle leicht ein Programm entwickeln, welches die gewünschte Funktionalität implementiert und zur Nutzung durch NRPE_NT eingebunden werden kann. Vorgefertigte Plugins stehen auf [15] für eine Vielzahl von Anwendungsgebieten bereit.

5.3.1 Konfiguration des Windows-Clients

Zunächst ist das NRPE_NT-Paket von [14] herunter zu laden. Zum Einsatz kam hier Version 0.8. Die in dem Archiv-Ordner „bin“ enthaltenen Dateien sind in ein zu verwendendes Installationsverzeichnis zu entpacken und das Paket nachfolgend zu konfigurieren. NRPE_NT liegen keine Test-Programme bei, diese müssen mit dem Paket „*Basic NRPE_NT Plugins*“ von [15] separat herunter geladen und in das NRPE_NT-Verzeichnis entpackt werden. Es handelt sich dabei um einfache ausführbare Dateien, die sich wie die bekannten Nagios-Plugins auf Unix-Systemen verhalten. Derzeit stellt das Paket folgende Programme bereit:

- `cpuload_nrpe_nt.exe`: Gibt die aktuelle Auslastung des Prozessors zurück. Dazu wird die CPU 20 mal innerhalb von 5 Sekunden abgefragt und daraus ein Mittelwert errechnet. Weiterhin wird über 10 Anfragen gemittelt und daraus eine Varianz berechnet, die ebenfalls

zurück gegeben wird, allerdings nur zur Information. Die anzugebenden Schwellwerte für eine Warnung und eine kritische Meldung beziehen sich nur auf die aktuelle Anfrage.

- `diskspace_nrpe_nt.exe`: Überwacht den aktuell verfügbaren Speicherplatz einer ausgewählten Festplatte.
- `eventlog_nrpe_nt.exe`: Prüft, ob Dienste im Windows Eventlog Fehler oder Warnungen eingetragen haben. Dabei ist anzugeben, wie viele Minuten in der Vergangenheit einzubeziehen sind.
- `memload_nrpe_nt.exe`: Gibt die aktuelle Auslastung des Arbeitsspeichers sowie der Auslagerungsdatei zurück. Marken für Warnungen und kritische Meldungen können angegeben werden, beziehen sich aber lediglich auf den Arbeitsspeicher und nicht auf die Auslagerungsdatei.
- `service_nrpe_nt.exe`: Dieses Plugin überwacht den Status von Diensten, welche als kommaseparierte Liste mit ihrem vollständigen Namen anzugeben sind. Ist ein Dienst nicht aktiv, so wird eine kritische Meldung gesendet.

Ein weiteres Paket von [14] mit dem Namen „*NagiosPluginsNT*“ stellt zusätzliche Plugins zur Verfügung. Nachteilig ist hierbei, dass diese in C# geschrieben sind und das .NET Framework in Version 2.0 oder höher erfordern. Das Laden des Frameworks zum Start eines solchen Plugins nimmt auf älteren Maschinen eine gewisse Zeit in Anspruch, die mit der Laufzeit des Plugins im Extremfall zur Zeitüberschreitung einer aktiven Prüfung führen kann (vor allem bei hoher Prozessorauslastung). Folgendes Bild zeigt eine Situation, in der dieser Fall eingetreten ist:

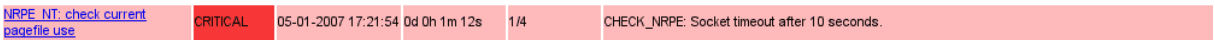

Abbildung 21: Zeitüberschreitung bei Nutzung eines .NET-Plugins in NRPE_NT

Neben den o. a. Funktionalitäten von „*Basic NRPE_NT Plugins*“ (mit Ausnahme der Prüfung des Eventlogs) werden folgende zusätzlich durch „*NagiosPluginsNT*“ bereitgestellt:

- `check_disk_time.exe`: Ermittelt die prozentuale Anzahl der Zeit, die ein physisches Laufwerk in Benutzung ist.
- `check_http.exe`: Prüft einen Host (z. B. localhost) auf einen laufenden HTTP-Server.
- `check_ping.exe`: Sendet einen Ping an einen definierten Host (Request) und wartet auf eine Rückantwort (Response).
- `check_reg.exe`: Überprüft, ob in der lokalen Windows-Registrierung ein übergebener Schlüssel vorhanden ist und gibt im positiven Fall dessen Wert zurück.
- `check_snmp.exe`: SNMP „get“ Plugin, welches eine bestimmte OID von einem SNMP-Server erfragen kann.
- `check_swap.exe`: Prüft die Belegung der Windows-Auslagerungsdatei.
- `check_tcp.exe`: Führt einen TCP-Verbindungsaufbau zu einem angegebenen Rechner auf einem bestimmten Port durch.

- `check_uname.exe`: Ermittelt die Betriebssystemversion und gibt diese zurück. Damit hat das Plugin lediglich Informationscharakter.
- `check_uptime.exe`: Gibt die aktuelle Betriebslaufzeit des Systems seit dem letzten Start von Windows zurück und dient lediglich der Information.

In Hinblick auf die Probleme des Szenarios lassen sich mit den beiden oben genannten Plugin-Paketen für NRPE_NT bereits viele Aufgaben lösen. CURCPU durch `cpuload_nrpe_nt.exe`, CURRAM durch `memload_nrpe_nt.exe`, CURPAGE durch `check_swap.exe`, DISKC durch `diskspace_nrpe_nt.exe` und SERVICES durch `service_nrpe_nt.exe`. Nur MIDCPU und PERFMON bereiten Probleme. Ähnlich wie bei SNMP müsste für MIDCPU ein eigenständiges Programm geschrieben werden, welches unabhängig von den NRPE_NT-Plugins läuft und z. B. minütlich die Prozessorlast prüft und in eine Datei schreibt. Diese Datei wäre dann bei einer Anfrage von einem Plugin auf dem Windows-Rechner auszulesen und ein Mittelwert über die letzten 15 Minuten zu bilden. Dies wird hier nicht weiter verfolgt.

Zur Realisierung von PERFMON wird ein Plugin benötigt, welches auf beliebige Perfmon-Objekte zugreifen kann. Auf [15] ist das NRPE_NT-Plugin „Check Counter“ zu finden, welches das in VBScript geschriebene Plugin `wincheck_counter.exe` zur Verfügung stellt, mit dem sich beliebige Performance Counter abrufen lassen.

So ausgestattet lässt sich das erstellte Szenario wie bei der Benutzung von SNMP bis auf MIDCPU komplett lösen. Dazu ist nun die Datei `nrpe.cfg` so zu konfigurieren, dass die Prüf-Programme bekannt gemacht werden und vom Nagios-Host abgerufen werden können. Weiterhin kann hier beispielsweise zur Erhöhung der Sicherheit eine Liste von Rechnern angegeben werden, denen der Zugriff auf den Dienst erlaubt ist. Im vorliegenden Szenario findet trotz der Sicherheitsbedenken die Festlegung der Argumente für die Plugins auf Seite des Nagios-Hosts statt, womit die entsprechenden Einträge in der Datei `nrpe.cfg` wie folgt aussehen (Installationsordner von NRPE_NT ist hierbei C:\NRPE):

```
# activate command argument processing
dont_blame_nrpe=1

# define commands with arguments
command[nt_cpuload]=C:\NRPE\cpuload_nrpe_nt.exe $ARG1$ $ARG2$
command[nt_memload]=C:\NRPE\memload_nrpe_nt.exe $ARG1$ $ARG2$
command[nt_pageload]=C:\NRPE\check_swap.exe $ARG1$ $ARG2$
command[nt_diskc]=C:\NRPE\diskspace_nrpe_nt.exe $ARG1$ $ARG2$ $ARG3$
command[nt_perfmon]=C:\NRPE\wincheck_counter.exe -C "$ARG1$" \
  -P "$ARG2$" -f "$ARG3$" -w $ARG4$ -c $ARG5$
command[nt_service]=C:\NRPE\service_nrpe_nt.exe "$ARG1$"
```

Bei der Definition von Kommandos wird in eckigen Klammern der Name angegeben, unter welchem das Kommando von außen abrufbar sein soll. Hinter dem Gleichheitszeichen folgen der Pfad zur ausführbaren Datei und die zu übergebenden Argumente. Bei den Kommandos „nt_cpuload“, „nt_memload“ und „nt_pageload“ handelt es sich dabei um die Marken für Warnungen und kritische Meldungen, bei „nt_diskc“ wird als erstes Argument noch der Buchstabe des zu prüfenden Laufwerks zwischen geschoben. „nt_perfmon“ erhält über `-C` das zu prüfende Perfmon-Objekt, über `-P` die jeweilige Instanz und über `-f` die Formatierung der Ausgabe. `-w` und `-c` geben

abermals die warnende und kritische Marke an. Das Kommando „nt_service“ übernimmt in nur einem Argument die Liste der auf Aktivität zu überwachenden Windows-Dienste.

Um den NRPE_NT-Dienst zu installieren und dann bei jedem Windows-Boot automatisch starten zu lassen, ist der Befehl `nrpe_nt -i` auszuführen. In der Liste verfügbarer Dienste wird nun ein neuer Dienst „Nagios Remote Plugin Executor for NT/W2K“ aufgeführt, der mit `net start nrpe_nt` gestartet wird. Damit ist der Windows-Client soweit konfiguriert, dass Anfragen von einem Nagios-Host entgegen genommen werden können. Wird die Konfigurationsdatei nachträglich geändert, so ist der NRPE_NT-Dienst wie jeder andere Windows-Dienst neu zu starten mittels:

```
> net stop nrpe_nt
> net start nrpe_nt
```

5.3.2 Konfiguration des Nagios-Hosts

Der Zugriff auf den NRPE_NT-Dienst erfolgt aus Sicht des Nagios-Hosts wie der Zugriff auf den NRPE-Dämon eines zu überwachenden Unix-Systems. Daher wird auch dasselbe Plugin zum Kontaktieren des Dienstes benötigt. Es handelt sich dabei um `check_nrpe` aus dem Addon-Paket NRPE (siehe 4.5), welches nach der Kompilierung des Pakets im Unterverzeichnis „src“ der NRPE-Distribution bereitsteht und in das Plugin-Verzeichnis „libexec“ des Nagios-Installationsordners kopiert werden muss.

Im Anschluss daran ist wiederum die Datei `commands.cfg` zu erweitern, um das NRPE-Plugin als Kommando verfügbar zu machen. Die Option `-c` des Plugins „check_nrpe“ gibt dabei den clientseitig zu prüfenden Befehl an, mit `-a` lässt sich diesem Befehl eine Liste von Argumenten übergeben.

```
# NRPE_NT command definition
define command{
        command_name  check_nrpe
        command_line  $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$ -a $ARG2$
}
```

In der Datei `hosts.cfg` wird das Kommando `check_nrpe` nun dazu verwendet, um abzurufende Dienste und dazugehörige Rechner (hier „rtc3‘’) zu definieren:

```
# NRPE services
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NRPE_NT: check current cpuload
        check_command                   check_nrpe_arg!nt_cpuload!90 95
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NRPE_NT: check current memload
```

```
        check_command                   check_nrpe_arg!nt_memload!85 95
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NRPE_NT: check current pagefile usage
        check_command                   check_nrpe_arg!nt_pageload!70 80
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NRPE_NT: check disk space of drive C
        check_command                   check_nrpe_arg!nt_diskc!C: 80 90
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NRPE_NT: check process count
        check_command                   check_nrpe_arg!nt_perfmon!"System" \
          "Prozesse" "process count: %f" 50 100
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NRPE_NT: check services defined on client
        check_command                   check_nrpe_arg!nt_service!"AntiVir \
          PersonalEdition Classic Service,AntiVir Scheduler,Sygate Personal \
          Firewall,Automatische Updates,DHCP-Client"
}
```

Nach einem Neustart von Nagios ergibt sich bei der Überwachung von Diensten auf der Weboberfläche folgendes Bild, welches alle definierten Dienste und die Ergebnisse ihrer letzten Prüfungen darstellt:

rtc3	NRPE_NT: check current cpuload	OK	06-01-2007 17:15:00	0d 4h 32m 22s	1/4	NOW: Mean:29.842106% Variance: 48.683217% CUMULATIVE: Mean:29.842106% Variance: 48.683217%
	NRPE_NT: check current memload	OK	06-01-2007 17:15:55	0d 23h 31m 18s	1/4	Mem: 378 MB (77%) / 487 MB (22%) Paged Mem: 398 MB (34%) / 1141 MB (65%)
	NRPE_NT: check current pagefile usage	OK	06-01-2007 17:17:51	0d 0h 25m 49s	1/4	Page File Utilization OK - C:\pagefile.sys = 17%
	NRPE_NT: check disk space of drive C	OK	06-01-2007 17:17:46	0d 23h 28m 48s	1/4	Used: 16449 MB (57%) Free: 12158 MB (42%)
	NRPE_NT: check process count	OK	06-01-2007 17:13:17	0d 0h 44m 48s	1/4	process count: 49
	NRPE_NT: check services	OK	06-01-2007 17:14:13	0d 0h 37m 54s	1/4	AntiVir PersonalEdition Classic Service OK AntiVir Scheduler OK Sygate Personal Firewall OK Automatische Updates OK DHCP-Client OK

Abbildung 22: Konfigurierte NRPE_NT-Abfragen

5.4 NSClient

Die nächste Möglichkeit der Windows-Überwachung ist durch NSClient (NetSaint Windows Client, siehe [18]) gegeben. Bei NSClient handelt es sich um einen Windows-Dienst, der beliebige

Systemparameter auslesen kann, die durch Perfmon zur Verfügung gestellt werden. Über ein spezielles Plugin (`check_nt`) kann der Nagios-Prozess dann diese Parameter erfragen. Es ist allerdings keine Nutzung eigener Plugins möglich, was die Flexibilität des Pakets erheblich hemmt.

Das letzte Update der NSClient-Webseite unter [18] bezieht sich auf den Februar des Jahres 2004 und aus Mailinglisten-Einträgen (siehe [7]) geht hervor, dass die Entwicklung von NSClient eingestellt wurde. Das unter 5.5 vorgestellte NC_Net stellt eine aktive Weiterentwicklung von NSClient dar und ist stattdessen in Betracht zu ziehen.

Nachfolgend werden die eingebauten Funktionen von NSClient im Überblick gegeben. Die Abkürzung am Anfang stellt die Option dar, mit der das Nagios-Host-Plugin `check_nt` über den Parameter `-v` für die jeweilige Funktionalität aufgerufen werden muss:

- `CPULOAD`: Prüft die Auslastung der CPU im Mittel in einem konfigurierbaren Zeitintervall von 1 bis 1440 Minuten.
- `MEMUSE`: Überprüft die aktuelle Belegung des virtuellen Speichers (Hauptspeicher + Auslagerungsdatei).
- `USEDDISKSPACE`: Überwacht die Belegung einer mit Laufwerkbuchstaben anzugebenden Festplatte.
- `UPTIME`: Dadurch werden weder Warnungen noch kritische Meldungen ausgelöst, es wird ausschließlich die Betriebszeit des Systems seit dem letzten Booten zurück gegeben.
- `SERVICESTATE`: Prüft den Status einer zu übergebenden Liste von Windows-Diensten und liefert eine kritische Meldung zurück, sobald ein Dienst nicht gestartet ist. Dabei ist der tatsächliche Name des Dienstes anzugeben, nicht der in der Diensteliste angezeigte Name.
- `PROCSTATE`: Testet eine zu übergebende Liste von Prozessen (deren Namen aus dem Windows Task Manager entnommen werden können) auf Aktivität und liefert eine kritische Meldung zurück, falls ein Prozess nicht gestartet ist.
- `FILEAGE`: Prüft das Alter einer angegebenen Datei. Warnungen oder kritische Meldungen werden ausgelöst, wenn die Minuten seit der letzten Aktualisierung die jeweiligen Grenzwerte übersteigen.
- `COUNTER`: Überwacht ein frei definierbares Perfmon-Counter-Objekt.

5.4.1 Konfiguration des Windows-Clients

Zunächst ist NSClient als ZIP-Archiv von [18] herunter zu laden und in ein separates Verzeichnis zu entpacken (hier in Version 2.0.1 vom 02.06.2003). In den Unterverzeichnissen „Win_2k_XP_Bin“ bzw. „Win_NT4_Bin“ steht dann mit `pNSClient.exe` der benötigte Windows-Dienst zur Verfügung. Mit dem Befehl `pNSClient.exe /install` wird der Dienst registriert und nachfolgend unter dem Namen „Nagios Agent“ in der Dienste-Liste geführt. Mit `net start nsclient` wird der Dienst gestartet.

Eine weitere Konfiguration von NSClient findet nicht statt, da die zur Verfügung stehenden Überwachungsfunktionen bereits vordefiniert sind. Dies erfreut Einsteiger, stellt insgesamt jedoch eine Einschränkung gegenüber Varianten wie NRPE_NT dar, die durch ihre individuelle Konfiguration flexibler sind.

5.4.2 Konfiguration des Nagios-Hosts

Der Nagios-Host kontaktiert den NSClient-Dienst auf einem Windows-Rechner über das Plugin `check_nt`, welches dem ZIP-Archiv von NSClient sowohl in den Quellen als auch vorkompiliert beiliegt. Nachdem das Plugin in das „libexec“-Verzeichnis der Nagios-Installation kopiert wurde, sind Nagios die zu überwachenden Dienste mitzuteilen.

Trotz der geringen Flexibilität von NSClient lassen sich fast alle Aufgaben des Szenarios lösen, was vor allem auch dem Zugriff auf alle Performance Counter zuzuschreiben ist. So werden neben PERFMON auch CURPAGE und CURCPU durch den Zugriff auf die entsprechenden Performance Counter realisiert. Bei CURCPU ist die Verwendung eines Performance Counters notwendig, da der in NSClient eingebaute Zugriff auf CPU-Werte lediglich ein Zeitintervall von 1 bis 1440 Minuten vorsieht, sich aber keine direkten Momentanwerte abrufen lassen. Dafür kann MIDCPU über diese Funktionalität gelöst werden. Dies ist im Gegensatz z. B. zu NRPE_NT möglich, da das Prüfen der CPU-Auslastung direkt in NSClient eingebaut ist und somit periodisch Werte durch den ständig aktiven Dienst erfasst werden können, um daraus schließlich einen Durchschnitt zu bilden. DISKC und SERVICES lassen sich schließlich ohne Umwege durch die eingebauten Funktionen `USEDDISKSPACE` und `SERVICESTATE` lösen.

Das elementare Datum CURRAM zur Prüfung der aktuellen Arbeitsspeicher-Belegung konnte nicht umgesetzt werden. Zunächst bietet sich die Funktion `MEMUSE` an, diese prüft aber die Belegung des gesamten virtuellen Speichers und nicht nur die Auslastung des Arbeitsspeichers. Weitere Optionen, die dies ermöglichen, können hier nicht angegeben werden. Daher ist man auf den Abruf eines Perfmon-Objektes angewiesen und auch hier liegt ein Stein im Weg: anstatt dass sich wie z. B. bei der Auslagerungsdatei eine prozentuale Anzahl der Belegung abrufen lässt, gibt es für den physischen Speicher nur den quantitativen Wert belegter Bytes, KB oder MB. Hier muss man zuvor selbst prüfen wieviel Arbeitsspeicher auf dem Rechner insgesamt zur Verfügung steht und anhand prozentualer Schranken entsprechende Grenzwerte für Nagios berechnen. Doch selbst wenn man diesen Umweg in Kauf nimmt, gibt es ein Problem: alle Perfmon-Objekte, welche den freien Arbeitsspeicher zurückgeben, enthalten auf dem verwendeten deutschen Windows-System Umlaute, z. B. „\Speicher\Verfügbare MB“. Es hat sich herausgestellt, dass ausgerechnet Perfmon-Counter mit Umlauten weder von NSClient noch von NC_Net, NRPE_NT oder NSClient++ abgerufen werden können, für sie ist der entsprechende Counter nicht existent. Ob dies auf Kodierungsprobleme auf dem Nagios-Host, bei der Übertragung oder auf dem Windows-System zurückzuführen ist konnte nicht festgestellt werden, eine Internet-Recherche förderte auch keine Ergebnisse zu Tage. Eine Alternative stellt die Verwendung der englischen Counter-Definition „\Memory\Available Mbytes“ dar, die aus Kompatibilitätsgründen selbst auf einem deutschen Windows-System noch definiert sein muss. NSClient prüft jedoch anscheinend nur deutsche Counter, sodass sich für das Problem CURRAM hier keine Lösung ergibt.

Dem Plugin `check_nt` wird über den Parameter `-v` mitgeteilt, welche Funktion aufgerufen werden soll. Dementsprechend werden einzelne Kommandos in der `commands.cfg` für NSClient hinzugefügt. Das Plugin verfügt über eine Reihe von Argumenten, die abhängig von der zu prüfenden Funktion angegeben werden müssen oder nicht. `-H` legt die Host-Adresse fest, `-p` gibt den Port an und `-v` definiert die zu verwendende Funktion. `-l` spezifiziert eine Liste von Argumenten, die von der jeweiligen Funktion abhängig sind. `-w` und `-c` geben wie üblich die Marken für Warnungen und kritische Meldungen an.

```
# NSClient command definitions
# check CPU load
define command{
        command_name check_nt_cpu
        command_line $USER1$/check_nt -H $HOSTADDRESS$ \
          -p $ARG1$ -v CPULOAD -l $ARG2$
}
# check disk usage
define command{
        command_name check_nt_disk
        command_line $USER1$/check_nt -H $HOSTADDRESS$ -p $ARG1$ \
          -v USEDDISKSPACE -l $ARG2$ -w $ARG3$ -c $ARG4$
}
# check the state of one or more services
define command{
        command_name check_nt_service
        command_line $USER1$/check_nt -H $HOSTADDRESS$ -p $ARG1$ \
          -v SERVICESTATE -l $ARG2$
}
# check an individual counter
define command{
        command_name check_nt_counter
        command_line $USER1$/check_nt -H $HOSTADDRESS$ -p $ARG1$ \
          -v COUNTER -l $ARG2$,$ARG3$ -w $ARG4$ -c $ARG5$
}
```

In der Datei `hosts.cfg` werden diese Kommandos schließlich benutzt, um von Nagios zu überwachende Dienste auf dem Host „rtc3“ anzulegen:

```
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient: current CPU load
        check_command                   check_nt_counter!1248!\
          "\\Prozessor(_Total)\\Prozessorzeit (%)"!\
          "current CPU load: %.2f %%"!90!95
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient: median CPU load over 15 min
        check_command                   check_nt_cpu!1248!15,80,90
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient: current pagefile usage
        check_command                   check_nt_counter!1248!\
```

```
          "\\Auslagerungsdatei(_Total)\\Belegung (%)"!\
          "paging file usage: %.2f %%"!70!80
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient: current usage of disk C:
        check_command                   check_nt_disk!1248!C!80!90
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient: current service states
        check_command                   check_nt_service!1248!AntiVirService,\
          AntiVirScheduler,SmcService,wuauserv,Dhcp
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient: current process count
        check_command                   check_nt_counter!1248!"\\System\\Prozesse"!\
          "process count: %.0f"!50!100
}
```

Die folgende Abbildung stellt die Dienste-Übersicht der Weboberfläche nach dieser Konfiguration dar und enthält dabei die Ergebnisse der letzten Prüfungen:

rtc3	NSClient: current CPU load	OK	06-01-2007 11:03:56	0d 0h 39m 29s	1/4	current CPU load: 28.00 %
	NSClient: current pagefile usage	OK	06-01-2007 11:05:42	0d 0h 52m 30s	1/4	paging file usage: 18.00 %
	NSClient: current process count	OK	06-01-2007 11:06:50	0d 0h 51m 24s	1/4	process count: 46
	NSClient: current service states	OK	06-01-2007 11:07:58	0d 0h 40m 27s	1/4	All services are running
	NSClient: current usage of disk C:	OK	06-01-2007 11:04:06	0d 0h 39m 19s	1/4	C: - total: 27.94 Gb - used: 16.06 Gb (57%) - free 11.87 Gb (43%)
	NSClient: median CPU load over 15 min	OK	06-01-2007 11:05:13	0d 0h 38m 12s	1/4	CPU Load (15 min. 32%)

Abbildung 23: Konfigurierte NSClient-Abfragen

5.5 NC_Net

Bei NC_Net handelt es sich um eine auf der Funktionalität von NSClient aufsetzende Entwicklung, die zur Bewahrung der Kompatibilität dieselben Ein- und Ausgaben benutzt und somit auch über dasselbe Plugin `check_nt` auf dem Nagios-Host abgerufen wird. Wie NSClient wird auch NC_Net als Windows-Dienst registriert und kann dann von außen abgerufen werden. NSClient und NC_Net können desweiteren nicht parallel betrieben werden, wenn die Grundeinstellung beider Pakete verwendet wird, da sie denselben Port (1248) belegen.

Die Verbesserungen von NC_Net gegenüber NSClient bestehen hauptsächlich aus passiven Überprüfungen, bei denen der Client ausschlaggebend ist für neue Statusinformationen und diese dann an den Nagios-Host sendet, im Gegensatz zur normalen aktiven Überwachung, bei welcher sich der Nagios-Prozess *aktiv* über neue Statusinformationen bei einem Client informiert (siehe auch

Abschnitt 4.6. Weiterhin kann NC_Net von einem entfernten Rechner aus konfiguriert werden und bindet zusätzliche Überwachungsfunktionen ein.

NC_Net 3.05 benötigt das .NET Framework 2.0 und höher, was eine Software-Abhängigkeit darstellt, die andere Varianten der Windows-Überwachung nicht besitzen und die für Administratoren durchaus auch ein Argument gegen die Verwendung von NC_Net darstellen kann.

NC_Net bietet die gleichen Funktionen wie NSClient und zusätzlich (die Abkürzung zu Beginn stellt wie bei NSClient im Unterabschnitt 5.4 die Option des Parameters `-v` des Host-Plugins `check_nt` dar):

- `EVENTLOG`: Prüft das Windows Ereignisprotokoll auf Fehler. Die Anzahl der Fehler definiert die Marken für Warnungen bzw. kritische Meldungen.
- `FREEDISKSPACE`: Ähnlich USEDDISKSPACE, nur dass hierbei der freie Plattenplatz geprüft und für die Grenzwerte verwendet wird.
- `WMICHECK`: Prüft WMI mit einer bestimmten Anfrage und gibt das Resultat zurück, ohne Warnungen oder kritische Meldungen auszulösen.
- `WMICOUNTER`: Führt eine Anfrage an WMI durch. Das Ergebnis wird mit den Marken für Warnungen und kritische Meldungen verglichen und ein entsprechendes Resultat geliefert.
- `CONFIG`: Hierüber lassen sich Einstellungen von NC_Net vom Nagios-Host aus verändern, ohne dass NC_Net neu gestartet werden muss. Dies kann allerdings auch zu einem großen Sicherheitsproblem werden, weswegen diese Funktionalität standardmäßig deaktiviert ist und bei der lokalen Konfiguration von NC_Net explizit erlaubt werden muss. Es ist zu beachten, dass diese Funktionalität nicht als periodischer Aufruf in Nagios eingetragen wird.

Weitere verfügbare Optionen haben lediglich sekundären Informationscharakter und wurden deswegen hier nicht mit aufgeführt.

5.5.1 Konfiguration des Windows-Clients

Zunächst ist das aktuelle NC_Net-Paket herunter zu laden und zu installieren. Verwirrend sind dabei unterschiedliche Bezugsquellen. Unter [12] findet sich die ausgepriesene „offizielle Seite“ von NC_Net, die allerdings seit Juli 2005 nicht mehr aktualisiert wurde. Aus einem Eintrag des deutschsprachigen Nagios-Forums unter [3] geht hervor, dass der Entwickler von NC_Net seine ehemalige Firma *Shatterit* verlassen hat, wo er für das Projekt verantwortlich war. NC_Net hat er nachfolgend auf Sourceforge (siehe [13]) eingetragen und stellt neue Versionen nur noch dort zur Verfügung.

Zur Verwendung kam hier NC_Net Version 3.05 von [13]. Das darin enthaltene Windows-Installer-Paket `NC_Net_setup.msi` installiert NC_Net auf dem Windows-System. Unter anderem wird dabei auch der benötigte Windows-Dienst registriert, welcher in der Dienste-Liste unter dem Namen „NC_Net“ wieder zu finden ist. Mittels `net start nc_net` kann er gestartet werden.

Versionen von NC_Net vor 3.05 besitzen wie auch NSClient nur einen (wenn auch im Vergleich erweiterten) festen Funktionsumfang, was die Flexibilität einschränkt. Nur die verwendete Version

3.05 bietet die Möglichkeit, eigene Programme im Ordner „script“ des Installationsverzeichnisses abzulegen und NC_Net so zu konfigurieren, dass es diese berücksichtigt. Der Nagios-Host kann dann über den Parameter „RUNSCRIPT“ darauf zugreifen. Die Dokumentation wurde an dieser Stelle noch nicht angepasst, sodass diese Funktionalität vor allem auch durch Blick in die Quellen selbst erforscht werden musste.

5.5.2 Konfiguration des Nagios-Hosts

NC_Net kann vom Nagios-Host aus mit demselben `check_nt` Plugin wie NSClient abgerufen werden. Dies erlaubt NC_Net jedoch nur die Nutzung der bereits in NSClient enthaltenen Funktionalität. Um die erweiterten Funktionen von NC_Net nutzen zu können, muss man auf dem Nagios-Host die C-Datei `check_nc_net.c` aus dem Verzeichnis „other files“ des NC_Net-Windows-Installationsverzeichnisses kompilieren und anstelle von `check_nt` verwenden. Für weitere Informationen gibt die Datei `readme.html` der NC_Net-Installation Auskunft. Das *neue* `check_nt`-Plugin ist abwärtskompatibel, kann also auch von NSClient verwendet werden.

Zur Lösung der Probleme des vorliegenden Szenarios kann die in 5.4.2 beschriebene Konfiguration zunächst ohne Anpassungen übernommen werden, womit bis auf CURRAM auch wieder alle geforderten Daten abgefragt werden können. Mit NC_Net lässt sich im Gegensatz zu NSClient allerdings auch CURRAM realisieren. Dies geschieht wie in 5.4.2 schon gewünscht durch den Zugriff auf den englischsprachigen Perfmon-Counter „\Memory\Available Mbytes“, was bei NSClient nicht funktioniert, bei NC_Net hingegen schon. Der Grund ist die Verwendung von .NET durch NC_Net und damit geänderte Abfragestrategien von Performance-Countern, die neben deutschen auch die englischen Pendants einbeziehen. So kann CURRAM durch Erweiterung der Datei `hosts.cfg` gelöst werden, als Marken für Warnungen bzw. kritische Meldungen sind dabei quantitative Werte zu übergeben, die den prozentualen Angaben von CURRAM entsprechen, was die Kenntnis der Arbeitsspeichergröße voraussetzt (73 MB ≈ 85%, 24 MB ≈ 95%):

```
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NC_Net: available RAM
        check_command                   check_nt_counter!1248!\
          "\\Memory\\Available Mbytes"!"available RAM (MB)"!73!24
}
```

Folgendes Bild zeigt das Ergebnis auf der Weboberfläche nach der Konfiguration und die Resultate der entsprechenden Prüfungen:

rtc3	NC_Net: available RAM	OK	12-01-2007 08:00:52	0d 0h 1m 9s	1/4	available RAM (MB) = 126,00
	NC_Net: current CPU load	OK	12-01-2007 08:01:37	0d 22h 7m 23s	1/4	current CPU load = 43,64 %
	NC_Net: current pagefile usage	OK	12-01-2007 07:57:24	4d 12h 16m 52s	1/4	paging file usage = 7,41 %
	NC_Net: current process count	OK	12-01-2007 07:58:11	4d 12h 20m 39s	1/4	process count = 43
	NC_Net: current service states	OK	12-01-2007 07:58:58	4d 12h 20m 18s	1/4	All services are running
	NC_Net: current usage of disk C:	OK	12-01-2007 07:54:44	4d 12h 19m 32s	1/4	C: - total: 27,94 Gb - used: 16,04 Gb (57%) - free 11,90 Gb (43%)
	NC_Net: median CPU load over 15 min	OK	12-01-2007 08:00:58	4d 12h 18m 45s	1/4	CPU Load 48% (15 min average)

Abbildung 24: Konfigurierte NC_Net-Abfragen

5.6 NSClient++

Bei NSClient++ (auch *nscp* genannt) handelt es sich um eine Variante der Windows-Überwachung, die an der Funktionalität von NSClient (siehe 5.4) angelehnt ist, allerdings darüber hinausgeht und eine komplette Neuentwicklung darstellt. Es wird ebenfalls ein Windows-Dienst zur Verfügung gestellt, der von einem Nagios-Host entweder über das NRPE-Plugin `check_nrpe` (im Gegensatz zu NSClient oder NC_Net) oder über das NT-Plugin `check_nt` abgerufen wird, wobei `check_nrpe` die sicherere Lösung darstellt. Funktionalität kann entweder über interne Plugins oder über externe Programme bereit gestellt werden und ist vom Umfang her vergleichbar mit NC_Net (siehe 5.5).

Auf der informativen Webseite des Projekts (siehe [17]) lässt sich eine aktive Entwicklung und Mitarbeit erkennen. So wurde erst am 05.12.06 Version 0.2.6 veröffentlicht, auch gehen immer wieder neue Kommentare und Hinweise ein. Für den 31.12.06 war Version 0.3.0 als Meilenstein angesetzt, dieses Ziel scheint jedoch nicht erreicht worden zu sein bzw. finden sich keine Informationen über den weiteren Fortgang. Es ist zu hoffen, dass die Entwicklung nicht ins Stocken gerät.

Ein wichtiges Detail von NSClient++ im Gegensatz zu NSClient oder NC_Net ist die Sicherheit. Da die Kommunikation mit dem Nagios-Host über dessen NRPE-Plugin stattfindet, ist die Verbindung grundsätzlich SSL-verschlüsselt. Weiterhin lässt sich die Liste von Rechnern mit Zugriffsrecht auf den NSClient++ Dienst beschränken, womit die Sicherheit mit der von NRPE_NT vergleichbar ist.

Der Funktionsumfang von NSClient++ beinhaltet (der Name zu Beginn steht dabei für das zu übergebende NRPE-Kommando):

- `CheckFileSize`: Erlaubt die Prüfung der Größe einer Datei oder des Inhaltes eines Verzeichnisses (rekursiv mit Unterverzeichnissen). Es können untere sowie obere Grenzen für Warnungen und kritische Meldungen angegeben werden.
- `CheckDriveSize`: Prüft die Größe eines Laufwerks und erlaubt untere sowie obere Grenzen für die Belegung bzw. den Freispeicher.
- `CheckEventLog`: Überprüft den Status des Windows Eventlogs und erlaubt eine maximale Anzahl von Fehlern, bis eine Warnung bzw. eine kritische Meldung ausgelöst werden.
- `CheckCPU`: Prüft die Prozessorauslastung in einem gegebenen Zeitintervall. Dieses ist nicht wie bei NSClient und NC_Net auf Minuten beschränkt, sondern lässt sich von Sekunden bis hin zu ganzen Wochen definieren.
- `CheckUpTime`: Überwacht die Betriebszeit eines Windows-Systems und erlaubt die Definition von minimalen und maximalen Werten.
- `CheckServiceState`: Überprüft den Status eines oder mehrerer Dienste auf dem System. Erlaubt im Gegensatz zu NSClient / NC_Net die Definition eines Status, den ein Dienst haben sollte. Als Name für einen Dienst kann dessen realer Name oder der in der Diensteliste angezeigte Name verwendet werden. Eine kritische Meldung wird zurück gegeben, sobald ein Dienst nicht den erwünschten Status besitzt.
- `CheckProcState`: Prüft den Status eines oder mehrerer Prozesse, deren Namen äquivalent ihrer Auflistung im Windows Task Manager anzugeben sind. Es kann weiterhin angegeben

werden, ob der jeweilige Prozess den Status „started“ oder „stopped“ besitzen soll. Eine kritische Meldung wird zurück gegeben, sobald ein Prozess nicht den angegebenen Status besitzt.

- `CheckMem`: Dieser Test überwacht die Speicherbenutzung. Dabei ist er flexibler als die Pendants von NSClient und NC_Net. Neben physikalischem Hauptspeicher lässt sich auch die Belegung der Auslagerungsdatei prüfen. Minimale und maximale Grenzen können frei definiert werden.
- `CheckCounter`: Fragt einen beliebigen Perfmon-Counter ab und erlaubt die Festlegung minimaler und maximaler Werte für das jeweilige Objekt.
- `CheckWMI`: Führt eine WMI-Anfrage aus und prüft das Ergebnis auf Überschreiten der festgelegten Grenzen. Diese Funktionalität befindet sich in einem sehr frühen Entwicklungsstadium und sollte nicht in Produktivsystemen eingesetzt werden.

5.6.1 Konfiguration des Windows-Clients

Zunächst ist ein aktuelles Paket von [17] herunter zu laden. wobei hier Version 0.2.6 zum Einsatz kam. Das ZIP-Archiv ist in ein zu verwendendes Installationsverzeichnis zu entpacken, daraufhin muss die Konfigurationsdatei `NSC.ini` angepasst werden. Hier finden sich viele Einstellungen und in den Kommentaren wird deutlich darauf hingewiesen, dass einige Optionen experimenteller Natur sind und noch nicht im produktiven Betrieb eingesetzt werden sollten (z. B. das CheckWMI-Plugin). Dem Paket liegt keine Offline-Dokumentation bei, diese kann jedoch auf der Projektseite unter [16] eingesehen werden.

`NSC.ini` *muss* editiert werden, da alle Plugins aus Sicherheitsgründen initial deaktiviert sind und die Grundeinstellungen individuell angepasst werden müssen. Die folgende Konfiguration fand hier Verwendung:

```
# modules that should be used
[modules]
CheckSystem.dll
CheckDisk.dll
NRPEListener.dll

[Settings]
allowed_hosts=127.0.0.1,192.168.200.2,192.168.200.3,192.168.200.4,192.168.200.5

[log]
file=NSC.log
date_mask=%Y-%m-%d %H:%M:%S

[Check System]
# Save CPU values for the last 16 minutes
CPUBufferSize=16m
# Check the CPU every second (one check every CheckResolution/10 seconds)
CheckResolution=10
```

```
[NRPE]
command_timeout=10
# arguments should be allowed for remote accessing of local commands with args
allow_arguments=1
# metachars have to be allowed for CheckCounter
allow_nasty_meta_chars=1
use_ssl=1
```

Kurz zur Erläuterung der einzelnen Sektionen. In `[modules]` werden die zu verwendenden Module definiert, welche in Form einer Liste von DLL-Dateien angegeben werden. `[Settings]` stellt allgemeine Einstellungen dar, in diesem Fall werden über die Option „allowed_hosts“ genau die Rechner angegeben, welche für einen Zugriff auf den NSClient++ Dienst berechtigt sind. `[log]` konfiguriert das Logging des Dienstes. Hier kann u. a. angegeben werden, welche Datei verwendet werden soll und wie das Log zu formatieren ist. In der Sektion `[Check System]` können Einstellungen für das Modul „CheckSystem“ vorgenommen werden. Beispielsweise lässt sich hier angeben, wie lange abgefragte CPU-Werte zu speichern sind (Befehl „CPUBufferSize“) und wie oft die CPU abgefragt werden soll (Befehl „CheckResolution“). Unter `[NRPE]` lassen sich Optionen für die Abfrage per NRPE festlegen, z. B. die maximale Abfragedauer bis zu einem Timeout („command_timeout“), das Entgegennehmen von Argumenten über NRPE-Befehle von außen („allow_arguments“) und die Benutzung von SSL für verschlüsselte Verbindungen („use_ssl“). In der vorliegenden Konfiguration wurde zudem die Option „allow_nasty_meta_chars“ gesetzt, um Meta-Zeichen wie ’>%&#’ etc. bei der Befehlsübergabe zu erlauben, was für die Abfrage von Performance-Countern nötig ist.

Nach der Konfiguration kann NSClient++ als Dienst registriert und gestartet werden. Dazu wechselt man in das Installationsverzeichnis und führt aus:

```
> NSClient++.exe /install
> net start nsclientpp
```

NSClient++ wird nachfolgend in der Dienste-Liste unter dem Namen „NSClientpp (Nagios) 0.2.6 2006-12-05“ aufgeführt.

5.6.2 Konfiguration des Nagios-Hosts

Wie bereits erwähnt erfolgt der Zugriff auf den NSClient++ Dienst hier über die NRPE-Schnittstelle, wodurch das Plugin `check_nrpe` zum Einsatz kommt. Die Probleme des vorliegenden Szenarios lassen sich alle direkt lösen. CURCPU und MIDCPU werden durch das Kommando *CheckCPU* abgedeckt, dem als Zeitintervall 1 Sekunde bzw. 15 Minuten übergeben wird. CURRAM wird durch *CheckMem* gelöst, DISKC durch *CheckDriveSize* und SERVICES durch *CheckServiceState*. Die Anzahl der laufenden Prozesse (PERFMON) sowie die Belegung der Auslagerungsdatei (CURPAGE) werden über Performance Counter mithilfe des Kommandos *CheckCounter* abgerufen.

Wie bei allen anderen Varianten ist zunächst wieder die Datei `commands.cfg` anzupassen, um das `check_nrpe`-Plugin als Kommando verfügbar zu machen. Die Option `-H` spezifiziert die Host-Adresse, über `-c` wird die auszuführende Funktion festgelegt und über `-a` kann eine von dieser Funktion abhängige Liste von Argumenten angegeben werden.

```
# NSClient++ command definition
define command{
        command_name  check_nscp_arg
        command_line  $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$ -a $ARG2$
}
```

Anschließend sind Nagios wieder die zu überwachenden Dienste auf Host „rtc3“ bekannt zu machen, was durch die Erweiterung von `hosts.cfg` geschieht:

```
# NSClient++ services
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient++: check current cpuload
        check_command                   check_nscp_arg!CheckCPU!Time=1s warn=90% \
          crit=95% ShowAll
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient++: check median cpuload over 15 min
        check_command                   check_nscp_arg!CheckCPU!Time=15m warn=80% \
          crit=90% ShowAll
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient++: check current memload
        check_command                   check_nscp_arg!CheckMem!type=physical \
          MaxWarn=85% MaxCrit=95% ShowAll
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient++: check current pagefile usage
        check_command                   check_nscp_arg!CheckCounter!\
          "Counter:pagefile usage=\\Auslagerungsdatei(_Total)\\Belegung (%)" \
          MaxWarn=70% MaxCrit=80% ShowAll
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient++: check disk space of drive C
        check_command                   check_nscp_arg!CheckDriveSize!Drive=C \
          MaxWarnUsed=80% MaxCritUsed=90% ShowAll
}
define service{
        use                             remote-service
```

```
        host_name                       rtc3
        service_description             NSClient++: check process count
        check_command                   check_nscp_arg!CheckCounter!\
          "Counter:process count=\\System\\Prozesse" MaxWarn=50 MaxCrit=100 ShowAll
}
define service{
        use                             remote-service
        host_name                       rtc3
        service_description             NSClient++: check services
        check_command                   check_nscp_arg!CheckServiceState!\
          AntiVirService AntiVirScheduler SmcService wuauserv Dhcp ShowAll
}
```

Als Resultat auf der Weboberfläche ergibt sich Abbildung 25, welche sowohl die konfigurierten Dienste als auch die Ergebnisse ihres Abfragens darstellt.

rtc3	NSClient++: check current cpuload	OK	06-01-2007 18:13:10	0d 0h 10m 21s	1/4	OK: 1s: 31%
	NSClient++: check current memload	OK	06-01-2007 18:13:36	0d 0h 25m 34s	1/4	OK: physical memory: 383M
	NSClient++: check current pagefile usage	OK	06-01-2007 18:14:23	0d 0h 24m 47s	1/4	OK: pagefile usage: 17.6262
	NSClient++: check disk space of drive C	OK	06-01-2007 18:15:10	0d 0h 24m 0s	1/4	OK: C:: 16.1G
	NSClient++: check median cpuload over 15 min	OK	06-01-2007 18:15:56	0d 0h 12m 35s	1/4	OK: 15m: 33%
	NSClient++: check process count	OK	06-01-2007 18:11:43	0d 0h 22m 27s	1/4	OK: process count: 48
	NSClient++: check services	OK	06-01-2007 18:12:30	0d 0h 21m 40s	1/4	OK: All services are running.

Abbildung 25: Konfigurierte NSClient++ Abfragen

5.7 Einzelbewertungen

Mit **SNMP** ließen sich dank der Integration von Performance Countern bis auf MIDCPU alle Probleme des Szenarios lösen. Dank der Hilfe von Perfmon-Objekten stehen viele Informationen zur Verfügung, die einfach über die jeweiligen SNMP-OIDs abgerufen werden können. Dies hat den Vorteil, dass man keine oder nur kleine Plugins entwickeln muss, um auf die meisten benötigten Daten Zugriff zu haben. Der Nachteil liegt in der geringen Flexibilität dieser statischen Informationen, durch die nicht jeder erdenkliche Problemfall abgedeckt werden kann. In einer solchen Situation ist man auf andere Lösungen angewiesen. Ein weiteres grundlegendes Problem an SNMP stellt die Sicherheit dar. SNMPv3 bietet umfassende Sicherheitsmechanismen, derzeit unterstützt der Windows SNMP-Dienst aber lediglich SNMPv2, welches nicht für umsonst auch nicht ganz ernst gemeint als Abkürzung für *Security is Not My Problem* steht. Da bei Kenntnis eines Community-Namens jeder detaillierte Informationen über einen Rechner einholen und sogar manipulieren kann (wenn Schreibrechte gesetzt sind) und weder Passwörter vergeben werden noch eine Verschlüsselung der Kommunikation stattfindet, ist diese Variante als unsicher anzusehen. Das Begrenzen der Hostliste stellt dabei nur einen rudimentären Schutz dar, der durch spezielle Angriffstechniken leicht ausgehebelt werden kann. In Umgebungen, wo SNMP breitflächig zum Einsatz kommt (z. B. durch die Verwendung weiterer Überwachungswerkzeuge, die sich auf SNMP stützen) und zusätzliche Schutzmaßnahmen der Rechner getroffen werden, ist die Überwachung von Windows-Maschinen mittels SNMP trotzdem empfehlenswert.

NRPE_NT stellt die direkte Übertragung des NRPE-Konzeptes von Unix auf Windows dar.

Das hat einige Vorteile. So kann auf dem Nagios-Host dasselbe Plugin zum Überwachen von Windows- und Unix-Maschinen zum Einsatz kommen. Weiterhin ist NRPE_NT sehr flexibel aufgrund der beliebig integrierbaren Plugins. Dies erlaubt die Anwendung des Pakets in vielen erdenklichen Situationen, vorausgesetzt es steht ein Plugin zur Verfügung oder wird zusätzlich entwickelt. Im Gegensatz zu SNMP wird bei NRPE_NT grundlegend auch auf Sicherheit geachtet. Übertragene Daten werden SSL-verschlüsselt und die Liste erlaubter Hosts lässt sich begrenzen. Dies darf jedoch nicht darüber hinwegtäuschen, dass weiterhin Angriffe möglich sind und sich Plugins mit frei definierbaren Argumentlisten ausführen lassen, wenn dies in der Konfiguration aktiviert wurde. Außerdem ist zu beachten, dass in NRPE_NT initial keine Plugins zur Verfügung stehen und theoretisch erst zu entwickeln sind, um Funktionalität zu erhalten. Jedoch ist die Plugin-Basis für NRPE_NT auf Seiten wie NagiosExchange (siehe [15]) so groß, dass für viele Anwendungsgebiete entsprechende Programme bereitstehen, die jederzeit genutzt werden können. Im vorliegenden Szenario konnten alle Fälle mit Ausnahme von MIDCPU gelöst werden, insgesamt lässt sich NRPE_NT empfehlen, wenn im Netzwerk bereits auf NRPE gesetzt wird und man ähnliche Funktionalitäten zur Überwachung von Windows-Rechnern einsetzen will.

NSClient stellt die einfachste Variante der Windows-Überwachung dar. Der Dienst muss lediglich installiert werden, eine weitere Konfiguration ist indes nicht nötig, aber auch gar nicht möglich. Auf dem Nagios-Host sind alleinig die Kommandos zu spezifizieren, mit denen NSClient zu kontaktieren ist. Diese müssen dann nur noch für die Überwachung durch Nagios konfiguriert werden. Der feste Funktionsumfang von NSClient ist besonders durch den Zugriff auf Performance Counter umfangreich genug, um die meisten Aufgaben des vorliegenden Szenarios zu lösen. Vor allem kann so z. B. die CPU über einen längeren Zeitraum überwacht und daraus ein Mittelwert errechnet werden, was durch Lösungen wie NRPE_NT oder SNMP nicht möglich ist. CURRAM ließ sich nicht umsetzen, da von NSClient nur der gesamte virtuelle Speicher überwacht werden kann. Eine Lösung mit Performance Countern scheiterte an Umlautproblemen (siehe Abschnitt 5.4.1). Weiterhin ist es nicht möglich eigene Plugins zu verwenden, wodurch sich viele Probleme nicht lösen lassen. Negativ ist zudem das komplette Fehlen jeglicher Sicherheit. Da keine Konfiguration des NSClient-Dienstes möglich ist, lassen sich weder Host-Listen begrenzen noch Passwörter setzen. Zudem findet die Verbindung unverschlüsselt statt, sodass ein Angreifer Daten abfangen und verfälscht an den Nagios-Host senden kann, womit weitere Angriffe verschleiert werden können. Aufgrund dessen und der Tatsache, dass das Paket keiner Weiterentwicklung unterliegt, ist es nur in einfachen Überwachungs-Umgebungen empfehlenswert.

NC_Net als auf NSClient aufbauende Entwicklung ist kompatibel mit diesem, wodurch beide mit demselben Plugin von Nagios angesprochen werden können. Auch NC_Net besitzt im Gegensatz zu NRPE_NT bereits einen festen Funktionsumfang, mit dem im Gegensatz zu NSClient alle Probleme des erstellten Szenarios gelöst werden konnten. Weiterhin lassen sich bei NC_Net jedoch auch eigene Plugins nutzen, was der Funktionsweise von NRPE_NT entspricht und als sehr flexibel zu bewerten ist. So lassen sich auch die von NRPE_NT verwendeten Plugins nutzen. Damit vereint NC_Net bezüglich der Funktionalität die Vorzüge von NRPE_NT und NSClient, zusätzlich ist die Unterstützung von passiver Überwachung und der Definition von Kommandos zur Realisierung dieser positiv hervorzuheben. Die Sicherheit kommt dabei jedoch zu kurz. Da man sich dafür entschlossen hat dieselbe Abfragemethodik wie NSClient einzusetzen, ist Sicherheit quasi nicht vorhanden. Es lassen sich keine Hostlisten anlegen, eine Verschlüsselung der Übertragung findet nicht statt. Da hilft es auch nicht, dass sich ein Passwort für den Zugriff auf den NC_Net-Dienst festlegen lässt. Da alle Daten unverschlüsselt übertragen werden, kann ein Angreifer, der die Verbindung abhört, leicht auch die Passwort-Daten ermitteln. Insgesamt ist dies der größte Nachteil bei NC_Net, welches ansonsten durch die Verbindung von fester

Funktionalität und beliebigen Plugins als sehr gut zu bewerten ist. Es bleibt zu hoffen, dass die nicht sehr aktive Entwicklung des Projekts zukünftig wieder belebt und fortgeführt wird.

NSClient++ hat zwar den Namen von NSClient geerbt, ansonsten unterscheiden sich beide Pakete aber grundlegend. Es besitzt zwar auch einen festen Funktionsumfang, dieser geht jedoch über den von NSClient hinaus und gestaltet sich zudem als flexibler, was die einzelnen Optionen angeht. So beispielsweise bei *CheckCPU*, welchem Zeitintervalle von einer Sekunde bis zu ganzen Wochen übergeben werden können. NSClient++ erfüllt wie NC_Net alle Aufgaben des Szenarios und ist darüber hinaus wie auch NRPE_NT und NC_Net für das Einbinden eigener Plugins offen. Nicht nur hier vereint es die Vorteile beider Pakete. Im Gegensatz zu NC_Net lässt sich NSClient++ über die NRPE-Schnittstelle abrufen, wodurch die Sicherheit der übertragenen Daten gewährleistet wird. Neben der dort stattfindenden Verschlüsselung lässt sich auch die Liste von Hosts begrenzen, die Zugriff auf den Dienst haben. Weiterhin erlaubt die Konfigurationsdatei `NSC.ini` eine umfangreiche Einstellung vieler sicherheitsrelevanter Parameter, sodass das gesamte Sicherheitskonzept noch vor NRPE_NT als umfangreichstes aller hier besprochenen Varianten der Windows-Überwachung anzusehen ist. Aufgrund der Flexibilität von NSClient++, der umfangreichen Sicherheit und der aktiven Entwicklung des Projekts ist es als sehr gut zu bewerten.

5.8 Vergleich und Fazit zur Windows-Überwachung

Tabelle 1 stellt einen übersichtlichen Vergleich und eine abschließende Gesamtbewertung der einzelnen untersuchten Varianten zur Windows-Überwachung dar.

	SNMP	NRPE_NT	NSClient	NC_Net	NSClient++
Version	v2	0.8	2.0.1	3.05	0.2.6
Letzte Aktualisierung	—	16.02.2006	02.06.2003	17.06.2006	05.12.2006
Windows-Dienst	snmp	nrpe_nt	nsclient	nc_net	nsclientpp
Standard-Port	161	5666	1248	1248	5666
Nagios-Plugin	diverse	check_nrpe	check_nt	check_nt	check_nrpe
Überwachungsart	aktiv	aktiv	aktiv	aktiv/passiv	aktiv
Eingebaute Funktionen	ja	nein	ja	ja	ja
Eigene Plugins	nein	ja	nein	ja	ja
Verschlüsselung	nein	ja (SSL)	nein	nein	ja (SSL)
Host-Begrenzung	ja	ja	nein	ja	ja
Passwortabfrage	nein	nein	nein	ja	nein
Bewertung	+	++	+	++	+++

Tabelle 1: Vergleich der Varianten zur Windows-Überwachung

NSClient schneidet am schlechtesten ab, was vor allem auf die fehlende Sicherheit und die geringe Flexibilität zurück zu führen ist, welche sich aufgrund des ausschließlich festen Funktionsumfangs ergibt. Das Prüfen von Performance Countern macht diese Variante trotzdem für einfache Überwachungsfunktionen einsetzbar.

SNMP folgt an vorletzter Stelle. In Umgebungen, wo es standardmäßig eingesetzt wird und auch andere Aufgaben mittels SNMP erledigt werden, ist dieses Ergebnis nach oben zu korrigieren,

für die reine Windows-Überwachung bietet es sich aufgrund der Umständlichkeit des Einbindens neuer Funktionen und der mangelnden Sicherheit in Version 2 jedoch nicht an.

NRPE_NT ist eine Variante, die vor allem auch durch ihre Sicherheitsmerkmale wie der SSL-Verschlüsselung punktet. Eine hohe Flexibilität wird hier durch das beliebige Einbinden eigener Plugins erreicht, allerdings sind standardmäßig keine Funktionen enthalten, was z. B. zur Lösung von MIDCPU nötig ist. Durch die aktive Community steht jedoch eine Vielzahl von Plugins für NRPE_NT zur Verfügung, wodurch das Paket an Wert gewinnt.

NC_Net besitzt in der neuesten Version sowohl feste Funktionen, lässt sich jedoch auch über beliebige eigene Plugins erweitern, was es flexibler einsetzbar macht als NRPE_NT und daher einen Platz vor diesem Paket einnimmt. Negativ zu werden ist insbesondere die geringere Sicherheit, dafür kann es durch die Möglichkeit passiver Überwachung punkten. Wer Windows-Rechner passiv mit Nagios und NSCA überwachen will, kommt an NC_Net nicht vorbei.

NSClient++ hat sich im Vergleich als beste Variante herausgestellt. Neben festen Funktionen bietet es die Möglichkeit der Verwendung eigener Plugins, womit es flexibel einsetzbar ist. Zudem sind die eingebauten Befehle individuell konfigurierbar. Weiterhin ist die Sicherheit umfassend, neben der Verschlüsselung übertragener Daten kann die Liste erlaubter Hosts begrenzt werden, zudem erlaubt die Konfigurationsdatei umfangreiche Einstellungen.

Um ein Fazit zu ziehen: für welches Paket man sich schlussendlich entscheidet, hängt stark von der übrigen Rechnerumgebung und den benötigten Funktionen ab. So kann auch NSClient häufig schon ausreichen. Man darf jedoch das Thema Sicherheit nicht vergessen und hier bieten sich vor allem die Lösungen NRPE_NT und NSClient++ an.

6 Zusammenfassung

Die vorliegende Arbeit behandelte Teilaspekte des freien Management-Werkzeuges Nagios. Im ersten Teil wurde Nagios an sich beschrieben und ein kurzer Überblick über Installation und Konfiguration des Systems gegeben. Als Erkenntnis hieraus lässt sich festhalten, dass Einsteiger aufgrund des umfangreichen Konfigurationsprozesses und der Fülle von Optionen, die textuell in Konfigurationsdateien einzutragen sind, überfordert werden. Doch gerade der Konfiguration muss viel Zeit und Sorgfalt eingeräumt werden. Andererseits erlaubt die Vielzahl an Einstellungen eine flexible und individuelle Anpassung an jede größere Rechnerumgebung, die klare Syntax aller Optionen sorgt dafür, dass erfahrene Administratoren zu überwachende Entitäten schnell hinzufügen oder ändern können. Bei der Beschreibung der Weboberfläche stellte sich heraus, dass durch die Vielzahl von Darstellungsarten stets der Überblick über Netzwerke, Hosts, Dienste und Probleme gewahrt werden kann. So wird es Administratoren ermöglicht, Informationen komprimiert und übersichtlich dargestellt zu bekommen und sich diese bei Bedarf auch tiefgründiger anzeigen zu lassen.

Der zweite Teil der Arbeit ist als umfangreiche Darstellung der Überwachung von Windows-Rechnern mit Nagios zu sehen. Dies stellte genau deswegen ein spannendes Thema dar, als dass Nagios für Unix-Systeme konzipiert wurde und Windows zunächst nicht unterstützt. Trotzdem muss ein Management-Tool in der Lage sein jede Art von Geräten überwachen zu können. Mit SNMP, NRPE_NT, NSClient, NC_Net und NSClient++ konnten gleich fünf Varianten anhand ihrer Konzepte und bereit gestellten Funktionalität untersucht werden. Dabei wurde für

jeden Fall zunächst ein Überblick gegeben und die Konfiguration von Client und Nagios-Host beschrieben. Einzelbewertungen konnten dann Vor- und Nachteile aufdecken, bevor ein Vergleich konkrete Empfehlungen gegeben hat, wobei mehrere Kriterien einbezogen wurden. Als wichtigste Erkenntnis lässt sich festhalten, dass sich die Pakete teilweise stark in Bezug auf Funktionalität, Sicherheit, Komplexität, Aktualität, aktive Weiterentwicklung und erforderlichem Konfigurationsaufwand unterscheiden. Ebenso hat sich gezeigt, dass man zwar Bewertungen für jede Lösung verteilen kann, allgemein gültige Empfehlungen für die Wahl eines Paketes jedoch nicht abgegeben werden können. Dieses individuelle Problem ist abhängig von der übrigen Rechnerumgebung und dem Einsatzzweck. Zwar geht NSClient++ mit der besten Bewertung aus dem Vergleich hervor, allerdings sind auch Fälle denkbar, in denen eine andere Lösung zu bevorzugen ist. Für einfache Anwendungsgebiete reicht so auch NSClient aus, bei großflächigem Einsatz von SNMP wird auch auf Abfragen per SNMP zurück gegriffen werden, für NRPE_NT spricht eine Umgebung, wo bereits andere Rechner mit NRPE überwacht werden. Als Ausblick bleibt zu hoffen, dass die angesprochenen bestehenden Mängel z. B. bei der Sicherheit beseitigt werden können und es bei der aktiven Entwicklung von Lösungen wie NSClient++ bleibt. Gerade hier spielt die Community eine entscheidende Rolle, durch welche über den Wunsch nach neuen Funktionen und die Bereitstellung weiterer Plugins, aber auch durch das Aufspüren von Fehlern und Schwachstellen die Weiterentwicklung und Stabilisierung der bestehenden Systeme gewährleistet werden kann.

Abschließend lässt sich sagen, dass Nagios zwar nicht an die großen kommerziellen Produkte (Unicenter, Tivoli etc.) heranreichen kann, für kleine bis mittelständige Unternehmen aber eine ernst zu nehmende und kostensparende Alternative zur Überwachung von Netzwerken darstellt. Die verwendeten Methoden und Funktionalitäten sind vielfältig und lassen sich durch die Integration von Plugin- und Addon-Konzepten auf praktisch jeden Anwendungsfall hin erweitern. Noch können proprietäre Lösungen ihre Marktmacht behaupten, doch je mehr sich Open-Source-Pakete wie Nagios durchsetzen, umso größer wird auch deren Akzeptanz und umso stärker werden sie gefördert. So scheint es nur eine Frage der Zeit, bis quelloffene Management-Software selbst in den ganz großen Rechenzentren der Welt umfassend zum Einsatz kommen wird.

A Glossar

.NET Framework Von Microsoft entwickelte Technologie, die viele Betriebssystem-Funktionen an einem zentralen Punkt anbietet, was veraltete Konzepte ablöst.

Addon Im Sinne von Nagios eine Funktionalitäts-Erweiterung, welche über die von → Plugins hinausgeht und eigene Dienste auf dem Nagios-Host oder den Clients anbietet.

Apache Meist verbreitetster Webserver im Internet.

CGI Abkürzung für „Common Gateway Interface“; Standard für den Datenaustausch zwischen Webserver und Programmen von Drittanbietern, womit diese über den Webserver von außen zugänglich werden.

Client In dieser Arbeit ein Rechner, auf dem Dienste oder Ressourcen durch Nagios überwacht werden sollen.

Counter Abrufbarer Leistungsparameter eines Systems, der durch → Perfmon zur Verfügung gestellt wird.

Dämon auch „Daemon“; in der Unix-Welt ein Programm, welches meist beim Hochfahren des Systems gestartet wird, im Hintergrund läuft und dort seine Aufgaben verrichtet; vergleichbar mit dem von Windows bekannten Begriff → „Dienst (Windows)“.

Dienst (Nagios) auch „Service“; stellt eine von Nagios zu überwachende Entität auf einem Client dar. Die Definition eines Dienstes wird in den Nagios-Konfigurationsdateien eingetragen. Dienste stellen Semantik für den Aufruf eines Plugins bereit und binden einen oder mehrere zu überwachende Rechner.

Dienst (Windows) Übersetzung des Begriffs → Dämon in die Windows-Welt.

Eventlog Das Windows-Eventlog kann als zentrale Stelle angesehen werden, in der Programme Fehler oder Warnungen hinterlassen können. Damit stellt es einen ersten Ansatzpunkt dar um Probleme einzugrenzen, aber auch um die fehlerfreie Arbeit von Windows-Systemen zu überwachen.

Frontend Programmoberfläche, auf der interne Daten meist grafisch aufbereitet dargestellt werden und welche die komfortable Interaktion mit Benutzern erlaubt.

gd Bibliothek Open-Source Bibliothek, welche das dynamische Erstellen von Grafiken ermöglicht und hauptsächlich bei der Webseiten-Entwicklung eingesetzt wird. In Nagios werden damit die Übersichten des „Reporting“ Menüpunktes sowie die 2D Statusmap erzeugt.

Guideline Englisch für „Richtlinie“; erweiterbare Softwareprojekte stellen i. d. R. Richtlinien zur Verfügung, um Dritt-Entwicklungen problemlos einbinden zu können, wenn diese sich an die gestellten Vorgaben halten.

Host Im Sinne von „Nagios-Host“: der Rechner, auf dem der Nagios-Prozess läuft; sonst: in der Nagios-Konfiguration eingetragene Rechner, die zu überwachen sind (siehe → Client).

ICMP Abkürzung für „Internet Control Message Protocol“; Teil der Internetprotokollfamilie (wie auch TCP und UDP), dient in Netzwerken zum Austausch von Informations- und Fehlermeldungen.

Monitoring Englisch für „Überwachung“; in Nagios kann dabei die Überwachung von Hosts, Diensten oder Gruppen von beiden gemeint sein.

Nagios Werkzeug zum System- und Netzwerkmanagement, welches vielfältige Möglichkeiten der Überwachung von Netzwerken, Rechnern, Diensten und Ressourcen anbietet.

Open Source Open Source bezeichnet Software, die kostenlos kopiert werden darf und deren Quellen frei zugänglich sind sowie von jedermann eingesehen werden können. Der Quelltext darf dabei je nach Lizenz unter bestimmten Bedingungen angepasst und weiter gegeben werden.

Perfmon „Performance Monitor"; Windows-Dienst, mit dem verschiedene Parameter (→ Counter) eines Windows-Systems in Echtzeit überwacht werden können. Es ist somit zum einen zur Diagnose im Fehlerfall und zum anderen zur aktuellen Leistungsmessung einsetzbar.

Ping Mit dem Befehl `ping` kann geprüft werden, ob ein Rechner in einem Netzwerk erreichbar ist. Dazu wird das ICMP-Protokoll verwendet. Durch Firewalls wird diese Funktionalität manchmal unterbunden, sodass ein Rechner trotzdem auf bestimmten Ports (z.B. HTTP) erreichbar sein kann, obwohl ein `ping` zu ihm nicht funktioniert.

Plugin Im Sinne von Nagios ein kleines Programm, welches zur Überwachung von Diensten oder Ressourcen auf Hosts aufgerufen wird und Resultate an den Nagios-Prozess zur Auswertung zurückgibt.

Reporting Englisch für „Berichterstattung"; in Nagios lassen sich auf der Weboberfläche Problemberichte und Statistiken erstellen, die komprimiert Informationen zu überwachten Entitäten liefern.

Service (Nagios) Siehe → Dienst (Nagios).

Tarball Datei, welche ein Archiv weiterer Dateien und Verzeichnisse darstellt. Die Archivierung erfolgt dabei über den Unix-Befehl „tar", von dem heute Portierungen auf viele andere Betriebssysteme existieren. Meist werden die Daten dabei zusätzlich komprimiert.

Template Englisch für „Schablone"; diese muss ausgefüllt werden, um individuell eingesetzt werden zu können. Nagios erlaubt die Definition von Hosts, Services, Kommandos, Hostgroups und Servicegroups über ein template-basiertes Schema, wobei eine solche Schablone zur Definition einer solchen Entität auszufüllen ist.

Unix-Derivat Von Unix abgeleitetes Betriebssystem, das auf Quelltext von Unix beruht oder Teile der Unix-Funktionalität nachbildet.

VRML Abkürzung für „Virtual Reality Modeling Language"; stellt eine Beschreibungssprache für 3D-Szenen dar.

WBEM Abkürzung für „Web Based Enterprise Management"; stellt einen Satz von Standardfunktionalitäten zur Fernüberwachung und -administration von Rechnern zur Verfügung. Im Vordergrund steht dabei vor allem auch die Unabhängigkeit von Hardware und Betriebssystem.

Windows Installer Stellt eine Ausführungsumgebung für Installationsroutinen von Programmen unter Windows bereit.

WMI Abkürzung für „Windows Management Instrumentation"; Microsoft-Implementierung des „Common Information Model", einer Kernfunktionalität des standardisierten → WBEM. Über WMI kann lesend oder schreibend auf viele Einstellungen von Windows zugegriffen werden, lokal oder von entfernten Rechnern. Es stellt mittlerweile einen Ersatz für → Perfmon dar und wird von Microsoft stattdessen gepflegt.

B Quelltexte

B.1 check_snmp_win_curcpu

```
#!/bin/sh

#####
# shell script to check the current cpuload via snmp with perfmon from a windows host
#####

STATE_OK=0
STATE_WARNING=1
STATE_CRITICAL=2
STATE_UNKNOWN=3
STATE_DEPENDENT=4

PORT=161
OID=1.3.6.1.4.1.311.1.1.3.1.1.6.1.3.1.48

if [ $# -ne 3 ]
then
  echo "    checks the current cpuload via snmp with perfmon from a windows host"
  echo "    usage: $0 <IP address> <warning %> <critical %>"
  exit $STATE_CRITICAL
fi

HOST=$1
WARNPERC=$2
CRITPERC=$3

SNMPGETPATH=‘which snmpget‘

if [ ! -x $SNMPGETPATH ]
then
  echo "’snmpget’: command not found or not executable"
  exit $STATE_CRITICAL
fi

NUMOK=1
CPULOAD=‘snmpget -t 1 -r 5 -v 2c -O qvU -L n -c public $HOST:$PORT $OID‘
echo $CPULOAD | egrep -q ’[[:space:]]*[[:digit:]]+[[:space:]]*$’ || NUMOK=0
if [ $NUMOK = 0 ]
then
  echo "’snmpget’: couldn’t receive cpuload information"
  exit $STATE_CRITICAL
fi

echo -n "cpuload: $CPULOAD%"
```

```
if [ $CPULOAD -ge $CRITPERC ]
then
  echo " - CRITICAL"
  exit $STATE_CRITICAL
fi
if [ $CPULOAD -ge $WARNPERC ]
then
  echo " - WARNING"
  exit $STATE_WARNING
fi

echo " - OK"
exit $STATE_OK
```

B.2 check_snmp_win_procs

```
#!/bin/sh

#####
# shell script to check the process count via snmp with perfmon from a windows host
#####

STATE_OK=0
STATE_WARNING=1
STATE_CRITICAL=2
STATE_UNKNOWN=3
STATE_DEPENDENT=4

PORT=161
OID=1.3.6.1.4.1.311.1.1.3.1.1.4.12.0

if [ $# -ne 3 ]
then
  echo "    Checks the number of processes on a Windows host via SNMP and perfmon"
  echo "    usage: $0 <IP address> <warning count> <critical count>"
  exit $STATE_CRITICAL
fi

HOST=$1
WARNPERC=$2
CRITPERC=$3

SNMPGETPATH=‘which snmpget‘

if [ ! -x $SNMPGETPATH ]
then
  echo "’snmpget’: command not found or not executable"
```

```
  exit $STATE_CRITICAL
fi

NUMOK=1
PROCCOUNT=‘snmpget -t 1 -r 5 -v 2c -O qvU -L n -c public $HOST:$PORT $OID‘
echo $PROCCOUNT | egrep -q ’[[:space:]]*[[:digit:]]+[[:space:]]*$’ || NUMOK=0
if [ $NUMOK = 0 ]
then
  echo "’snmpget’: couldn’t receive process count information"
  exit $STATE_CRITICAL
fi

echo -n "process count: $PROCCOUNT"

if [ $PROCCOUNT -ge $CRITPERC ]
then
  echo " - CRITICAL"
  exit $STATE_CRITICAL
fi
if [ $PROCCOUNT -ge $WARNPERC ]
then
  echo " - WARNING"
  exit $STATE_WARNING
fi

echo " - OK"
exit $STATE_OK
```

B.3 check_snmp_win_page

```
#!/bin/sh

#####
# shell script to check the used pagefile memory on Windows hosts via SNMP
#####

STATE_OK=0
STATE_WARNING=1
STATE_CRITICAL=2
STATE_UNKNOWN=3
STATE_DEPENDENT=4

PORT=161
OIDBASE=1.3.6.1.2.1.25.2.3.1

if [ $# -ne 3 ]
then
  echo "    Get used pagefile memory on Windows hosts via SNMP"
```

```
  echo "    usage: $0 <IP address> <warning %> <critical %>"
  exit $STATE_CRITICAL
fi

HOST=$1
WARNPERC=$2
CRITPERC=$3

SNMPGETPATH=‘which snmpget‘
BCPATH=‘which bc‘

if [ ! -x $SNMPGETPATH ]
then
  echo "’snmpget’: command not found or not executable"
  exit $STATE_CRITICAL
fi
if [ ! -x $BCPATH ]
then
  echo "’bc’: command not found or not executable"
  exit $STATE_CRITICAL
fi

NUMOK=1
VIRTALLOC=‘snmpget -t 1 -r 5 -v 2c -O qvU -L n -c public $HOST:$PORT $OIDBASE.4.3‘
#echo $VIRTALLOC | egrep -q ’[[:space:]]*[[:digit:]]+[[:space:]]*$’ || NUMOK=0
if [ $NUMOK = 0 ]
then
  echo "’snmpget’: couldn’t receive virtual alloc information"
  exit $STATE_CRITICAL
fi

VIRTSIZE=‘snmpget -t 1 -r 5 -v 2c -O qvU -c public $HOST:$PORT $OIDBASE.5.3‘
echo $VIRTSIZE | egrep -q ’[[:space:]]*[[:digit:]]+[[:space:]]*$’ || NUMOK=0
if [ $NUMOK = 0 ]
then
  echo "’snmpget’: couldn’t receive virtual size information"
  exit $STATE_CRITICAL
fi

VIRTUSED=‘snmpget -t 1 -r 5 -v 2c -O qvU -c public $HOST:$PORT $OIDBASE.6.3‘
echo $VIRTUSED | egrep -q ’[[:space:]]*[[:digit:]]+[[:space:]]*$’ || NUMOK=0
if [ $NUMOK = 0 ]
then
  echo "’snmpget’: couldn’t receive virtual used information"
  exit $STATE_CRITICAL
fi

PHYSALLOC=‘snmpget -t 1 -r 5 -v 2c -O qvU -c public $HOST:$PORT $OIDBASE.4.4‘
echo $PHYSALLOC | egrep -q ’[[:space:]]*[[:digit:]]+[[:space:]]*$’ || NUMOK=0
```

```
if [ $NUMOK = 0 ]
then
  echo "'snmpget': couldn't receive physical alloc information"
  exit $STATE_CRITICAL
fi

PHYSSIZE=`snmpget -t 1 -r 5 -v 2c -O qvU -c public $HOST:$PORT $OIDBASE.5.4`
echo $PHYSSIZE | egrep -q '[[:space:]]*[[:digit:]]+[[:space:]]*$' || NUMOK=0
if [ $NUMOK = 0 ]
then
  echo "'snmpget': couldn't receive physical size information"
  exit $STATE_CRITICAL
fi

PHYSUSED=`snmpget -t 1 -r 5 -v 2c -O qvU -c public $HOST:$PORT $OIDBASE.6.4`
echo $PHYSUSED | egrep -q '[[:space:]]*[[:digit:]]+[[:space:]]*$' || NUMOK=0
if [ $NUMOK = 0 ]
then
  echo "'snmpget': couldn't receive physical used information"
  exit $STATE_CRITICAL
fi

VIRTSIZEMB=`echo "$VIRTSIZE*$VIRTALLOC/1024^2" | bc`
VIRTUSEDMB=`echo "$VIRTUSED*$VIRTALLOC/1024^2" | bc`
PHYSSIZEMB=`echo "$PHYSSIZE*$VIRTALLOC/1024^2" | bc`
PHYSUSEDMB=`echo "$PHYSUSED*$VIRTALLOC/1024^2" | bc`

SWAPSIZEMB=`echo "$VIRTSIZEMB-$PHYSSIZEMB" | bc`
SWAPUSEDMB=`echo "$PHYSSIZEMB-$PHYSUSEDMB" | bc`
SWAPUSEDPERC=`echo "$SWAPUSEDMB*100/$SWAPSIZEMB" | bc`

echo -n "swap usage: $SWAPUSEDMB of $SWAPSIZEMB MB ($SWAPUSEDPERC% used)"

if [ $SWAPUSEDPERC -ge $CRITPERC ]
then
  echo " - CRITICAL"
  exit $STATE_CRITICAL
fi
if [ $SWAPUSEDPERC -ge $WARNPERC ]
then
  echo " - WARNING"
  exit $STATE_WARNING
fi

echo " - OK"
exit $STATE_OK
```

Literatur

[1] *Adding support of SNMP Performance Monitoring on Windows 2000/XP.* http://www.loriotpro.com/ServiceAndSupport/How_to/How_to_add_performance_MIB_on_Windows_2000.php, letzter Zugriff: 05.01.07

[2] *Apache Webserver Homepage.* http://www.apache.org, letzter Zugriff: 10.12.06

[3] *Beitrag zu NC_Net im deutschen Nagios-Forum.* http://www.nagios-portal.de/forum/thread.php?threadid=4236, letzter Zugriff: 31.12.06

[4] *Cortona VRML Client.* http://www.parallelgraphics.com/products/cortona/, letzter Zugriff: 09.01.07

[5] *Download benötigter Dateien zur Perfmon-Integration in SNMP.* http://www.loriotpro.com/Download/Files/Perfmibfile.zip, letzter Zugriff: 05.01.07

[6] *Download der Datei perfmib.dll.* http://www.microsoft-engineer.com/files/perfmib.dll, letzter Zugriff: 05.01.07

[7] *Eintrag in der Mailingliste nagios-users zu NSClient.* https://sourceforge.net/mailarchive/message.php?msg_id=10174460, letzter Zugriff: 02.01.07

[8] *gd Graphics Library.* http://www.boutell.com/gd, letzter Zugriff: 10.12.06

[9] *Nagios Download Page.* http://www.nagios.org/download, letzter Zugriff: 10.12.06

[10] *Nagios Exchange Homepage.* http://www.nagiosexchange.org, letzter Zugriff: 17.12.06

[11] *Nagios Homepage.* http://www.nagios.org, letzter Zugriff: 10.12.06

[12] *NC_Net Official Site.* http://www.shatterit.com/nc_net, letzter Zugriff: 31.12.06

[13] *NC_Net Sourceforge Project Page.* http://sourceforge.net/projects/nc-net, letzter Zugriff: 31.12.06

[14] *NRPE_NT.* http://www.miwi-dv.com/nrpent, letzter Zugriff: 31.12.06

[15] *NRPE_NT Plugins.* http://www.nagiosexchange.org/NRPE_Plugins.66.0.html, letzter Zugriff: 31.12.06

[16] *NSClient++ Documentation.* http://trac.nakednuns.org/nscp/wiki/Documentation, letzter Zugriff: 03.01.07

[17] *NSClient++ Homepage.* http://trac.nakednuns.org/nscp, letzter Zugriff: 02.01.07

[18] *NSClient Official site.* http://nsclient.ready2run.nl, letzter Zugriff: 31.12.06

[19] *SNMP Nagios Plugins.* http://www.manubulon.com/nagios, letzter Zugriff: 05.01.07

[20] *SNMP RFCs.* http://www.snmp.org/protocol/snmp_rfcs.shtml, letzter Zugriff: 31.12.06

[21] *Sourceforge Nagios Plugin Page.* http://sourceforge.net/projects/nagiosplug, letzter Zugriff: 10.12.06