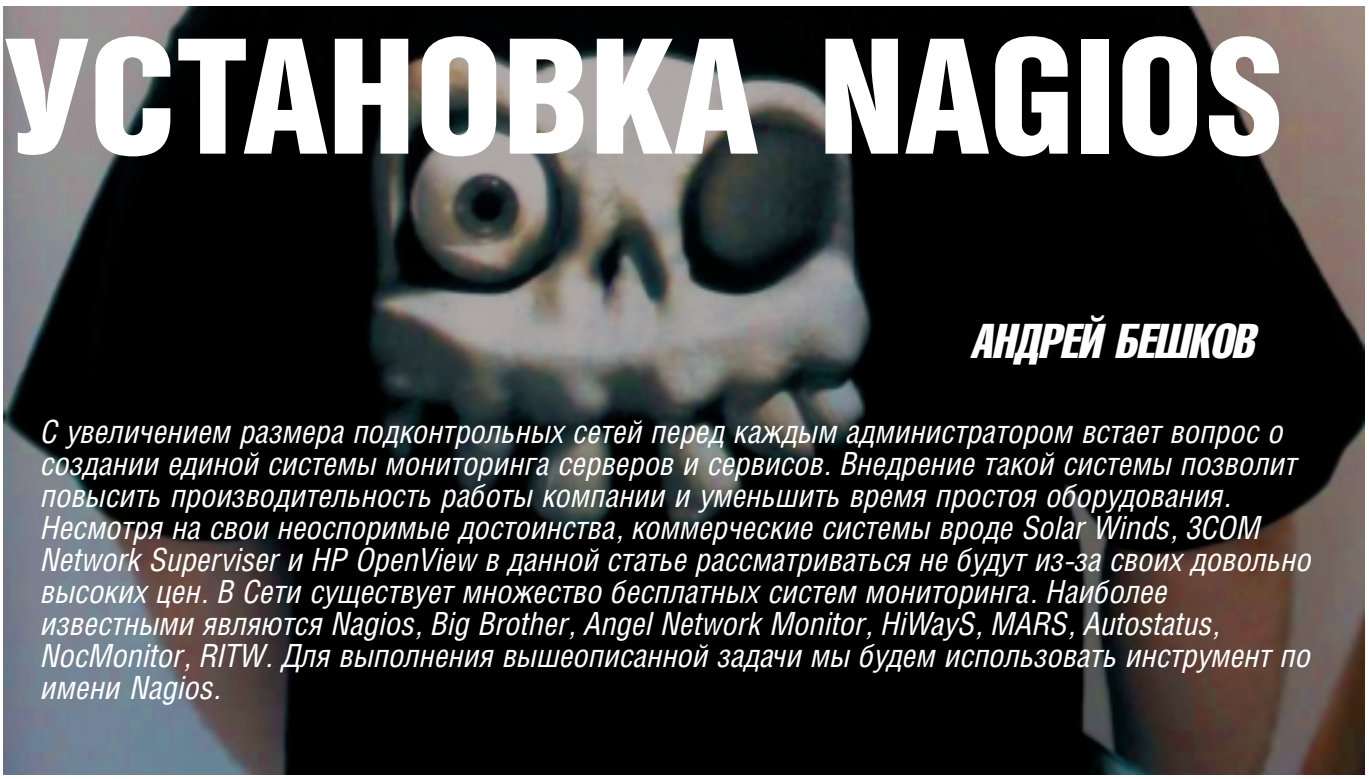
УСТАНОВКА NAGIOS



АНДРЕЙ БЕШКОВ

С увеличением размера подконтрольных сетей перед каждым администратором встает вопрос о создании единой системы мониторинга серверов и сервисов. Внедрение такой системы позволит повысить производительность работы компании и уменьшить время простоя оборудования. Несмотря на свои неоспоримые достоинства, коммерческие системы вроде Solar Winds, 3COM Network Superviser и HP OpenView в данной статье рассматриваться не будут из-за своих довольно высоких цен. В Сети существует множество бесплатных систем мониторинга. Наиболее известными являются Nagios, Big Brother, Angel Network Monitor, HiWayS, MARS, Autostatus, NocMonitor, RITW. Для выполнения вышеописанной задачи мы будем использовать инструмент по имени Nagios.

Давайте рассмотрим главные задачи системы мониторинга. Она должна оповещать администратора с помощью электронной почты, пейджера или sms-сообщений, если с наблюдаемыми объектами случаются какие-либо неприятности. Также оповещения должны отсылаться, когда состояние объекта возвращается в норму. Вдобавок ко всем способам, перечисленным выше, Nagios позволяет использовать в качестве инструмента оповещения программы, разработанные пользователями. Перед тем как мы станем более подробно обсуждать возможности Nagios, вам стоит посмотреть на демонстрационную систему, которую Tom Welsh установил специально для этих нужд. Быстренько идем на сайт http://nagios.squarebox.com/. Для авторизации используем имя пользователя guest и пароль guest. Набродившись по разным меню и полюбовавшись на все красоты веб-интерфейса, давайте поговорим о достоинствах Nagios. На основе увиденного мы можем сделать выводы:

- Nagios позволяет производить мониторинг таких сетевых сервисов, как SMTP, TELNET, SSH, HTTP, DNS, POP3, IMAP, NNTP и многих других.
- Можно следить за использованием ресурсов серверов, таких как расход дискового пространства, свободная память и загруженность процессора.
- Существует возможность создавать свои собственные обработчики событий.

Эти обработчики будут выполняться при возникновении тех или иных событий, инициированных проверками сервисов или серверов. Такой подход позволит активно реагировать на происходящие события и пытаться автоматически решать возникшие проблемы. К примеру, можно создать обработчик событий, который будет самостоятельно перезапускать повисший сервис.

Еще одним достоинством системы мониторинга Nagios является возможность управлять ею удаленно с помощью wap-интерфейса мобильного телефона. Используя концепцию “родительских” хостов, легко описать иерархию и зависимости между всеми хостами. Такой подход чрезвычайно полезен для больших сетей, потому что позволяет производить сложную диагностику. А это качество, в свою очередь, помогает отличить неработающие хосты от тех, что недоступны в данный момент из-за неполадок в работе промежуточных звеньев. Nagios умеет строить графики работы наблюдаемых систем и карты контролируемой сетевой инфраструктуры.

Из своей практики работы с Nagios приведу пример, показывающий, насколько полезен он оказался для меня. На внешнем сетевом интерфейсе нашего брандмауэра с периодичностью в несколько часов начиналась потеря пакетов. Из-за неисправности терялось до 20% проходящего трафика. По истечении минуты-другой интерфейс снова начинал работать как положено. По причине плавающего характера этой неполадки несколько недель я не мог понять, почему при работе с Интернетом периодически происходят кратковременные сбои. Без Nagios поиск неисправности затянулся бы надолго.

Многим из администраторов хорошо знаком предок Nagios по имени NetSaint. Несмотря на то что сайт проекта NetSaint все еще исправно функционирует, новые разработки базируются уже на исходном коде Nagios. Поэтому всем рекомендуется потихоньку перебираться на Nagios. Изначально проект разрабатывался для Linux, но наша система мониторинга будет работать на основе FreeBSD 4.5. В документации, поставляемой с

Nagios, говорится, что он будет стабильно работать и со многими другими Unix-подобными системами. Для отображения веб-интерфейса Nagios нам понадобится сервер Apache. Вы вольны использовать любой другой, но в данной статье будет рассматриваться именно Apache, как наиболее распространенный на Unix-платформах веб-сервер. Можно установить систему мониторинга вообще без веб-интерфейса, но мы так делать не станем, потому что это существенно уменьшает удобство пользования.

Для создания графиков нам понадобится библиотека GD Graphics, которую написал Thomas Boutell. Она необходима для создания изображений в формате PNG, JPEG и WBMP. WBMP чаще всего используется клиентами беспроводных устройств. Подавляющее большинство браузеров пока не умеет работать с WBMP, поэтому для нас он интереса не представляет. До перехода на версию 2.0.6 была возможность работать с форматом GIF, но от нее пришлось отказаться из-за проблем с патентным законодательством. Для сжатия информации внутри изображений GIF используется алгоритм LZW. Эксклюзивными правами на него обладает Unisys. В связи с вышеуказанными проблемами, сетевому сообществу рекомендуется использовать свободные от подобных неудобств форматы PNG и JPEG. В свою очередь, библиотека GD опирается на библиотеки zlib, LibPng и jpeg. У нас есть три пути, следуя которым мы можем установить все необходимые библиотеки. В статье они будут подробно описаны по очереди. Я люблю все собирать и устанавливать вручную, потому как подобный подход дает более полное понимание происходящих процессов и возможность использовать самое свежее программное обеспечение. Итак, начнем нашу эпопею со способа, красноречиво называемого в народе «закат солнца вручную».

Первый ингредиент – библиотека zlib, написанная двумя программистами – Jean-loup Gailly и Mark Adler. Она проектировалась как API общего назначения для сжатия данных без потерь. Как ни странно, но в нашем проекте она будет использоваться по своему прямому назначению, то есть для сжатия изображений. Берем новейшую версию http://www.gzip.org/zlib/. На момент написания это была 1.1.4. Распаковываем и компилируем.

```
# tar zvxf zlib-1.1.4.tar.gz
# cd zlib-1.1.4
# ./configure
# make test
# make install
```

Как вы могли бы предположить, LibPng служит для создания изображений в формате PNG. Я использовал версию 1.2.5. Взять ее можно на сайте: http://www.graphicswiz.com/png/libpng.html.

Собирается LibPng довольно-таки странным и нестандартным путем.

```
# tar zxvf  libpng-1.2.5.tar.gz
# cd libpng-1.2.5
```

Ориентируясь на тип операционной системы, из директории scripts выбираем нужный нам makefile и копируем его в директорию с исходными файлами. Для нашего случая исходный файл называется makefile.freebsd.

```
# cp ./scripts/makefile.freebsd
makefile
```

Затем создаем директорию zlib и кладем в нее из дистрибутива zlib все файлы с расширением *.h:

```
# mkdir zlib
```

Вот теперь можно проводить компиляцию. Что мы с радостью выполняем.

```
# make all
# make test
```

В результате на экране вы должны увидеть что-то вроде этого:

```
Testing pngtest.png:
  Pass 0: rwrwrwrwrwrwrwrwrw
  Pass 1: rwrwrwrwrwrwrwrwrw
  Pass 2: rwrwrwrwrwrwrwrw
  Pass 3: rwrwrwrwrwrwrwrwrwrwrwrw
rwrwrwrwrw
  Pass 4: rwrwrwrwrwrwrwrwrwrwrwrw
rwrwrwrwrw
  Pass 5: rwrwrwrwrwrwrwrwrwrwrwrw
rwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrw
  Pass 6: rwrwrwrwrwrwrwrwrwrwrwrw
rwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrw
  PASS (9782 zero samples)
  Filter 0 was used 21 times
  Filter 1 was used 15 times
  Filter 2 was used 52 times
  Filter 3 was used 10 times
  Filter 4 was used 33 times
  tIME = 7 Jun 1996 17:58:08 +0000
  Current memory allocation:
0 bytes
  Maximum memory allocation:
337346 bytes
  Total   memory allocation:
1023516 bytes
      Number of allocations:
198
 libpng passes test
 mkdir /usr/local/include/libpng
```

И наконец, пришло время инсталляции. Тут уж все должно пройти легко и просто.

```
# make install
```

Теперь берем по адресу http://www.ijg.org/files/ библиотеку jpeg версии 6b. Я надеюсь, самые догадливые читатели уже поняли, что она понадобится нам для создания изображений в формате jpeg. Поверив моим заявлениям о том, что процедура установки довольно проста, начнем ее выполнение.

```
# tar zxvf jpegsrc.v6b.tar.gz
# cd jpeg-6b
# ./configure
# make
```

После тестовой сборки должны увидеть такую картину:

```
# make test
 rm -f testout*
 ./djpeg -dct int -ppm -outfile
testout.ppm   ./testorig.jpg
 ./djpeg -dct int -bmp -colors 256
-outfile testout.bmp   ./testorig.jpg
 ./cjpeg -dct int -outfile
testout.jpg   ./testimg.ppm
 ./djpeg -dct int -ppm -outfile
testoutp.ppm  ./testprog.jpg
 ./cjpeg -dct int -progressive -opt
-outfile testoutp.jpg ./testimg.ppm
 ./jpegtran -outfile testoutt.jpg ./
testprog.jpg
 cmp ./testimg.ppm testout.ppm
 cmp ./testimg.bmp testout.bmp
 cmp ./testimg.jpg testout.jpg
 cmp ./testimg.ppm testoutp.ppm
 cmp ./testimgp.jpg testoutp.jpg
 cmp ./testorig.jpg testoutt.jpg
```

Следующей командой устанавливаем библиотеку jpeg.

```
# make install
```

Наконец, после всех этих мытарств, можно заняться инсталляцией библиотеки GD. Скачиваем исходные тексты с сайта проекта http://www.boutell.com/gd/. Как всегда, распаковываем и конфигурируем дистрибутив.

```
# tar zxvf  gd-2.0.6.tar.gz
# cd gd-2.0.6

# ./configure --x-includes=/usr/
local/include --x-libraries=/usr/
local/lib
```

Ключи –x-includes и –x-libraries указывают скрипту конфигурации, где у нас находятся заголовочные и библиотечные файлы всех установленных ранее библиотек.

```
# make all
```

Если компиляция завершилась с ошибками, то нам необходимо удалить временные файлы, созданные в процессе работы команды make.

```
# make devclean
```

После выполнения чистки нужно принять меры к устранению причин возникновения ошибок. При необходимости подправьте ключи команды configure, а затем проведите повторную компиляцию. И если предыдущие шаги завершились нормально, запускаем инсталляцию.

```
# make install
```

Теперь давайте обсудим второй способ установки. При наличии устойчивого соединения с Интернетом можно попытаться установить библиотеки GD, zlib, LibPng и jpeg с помощью механизма портов. Мы запустим установку GD, а она, в свою очередь, автоматически выполнит скачивание, компиляцию и установку всех необходимых библиотек на основе зависимостей между ними.

```
# cd /usr/ports/graphics/gd
# make install
```

Третий способ установки не требует подключения к сети Интернет. Скорее всего, он является самым простым и, видимо, лучше всего подходит для начинающих администраторов. Как и во втором способе, все необходимое будет автоматически установлено из пакетов, поставляющихся вместе с операционной системой FreeBSD. По аналогии со вторым способом за удобство придется заплатить свежестью устанавливаемого программного обеспечения. В CD-привод кладем компакт диск с пакетами и запускаем программу sysinstall.

```
# /stand/sysinstall
```

А затем, последовательно пройдя через все перечисленные ниже меню, выполняем инсталляцию.

```
Configure->Packages->CD/DVD-
>graphics->gd-1.8.4_3->ok->install->ok
```

Аналогичного эффекта можно добиться, используя команду pkg_add. Посмотрим, куда у нас все это установилось. Файлы библиотек лежат в /usr/local/lib, а заголовочные файлы в – /usr/local/include/gd. Не стоит огорчаться, если, несмотря на все попытки, библиотеку GD не удалось установить ни одним из трех способов. Хотя я с трудом представляю, как такое возможно. Отложите установку до лучших времен. Наличие вышеназванных библиотек необходимо только для правильной работы модулей построения графиков и генерирования сетевой карты. Nagios работает стабильно и остается очень полезным инструментом даже без этих модулей.

Пришло время детально разобраться с архитектурой системы мониторинга. Обычно комплекс на основе Nagios состоит из двух частей: главной программы, в документации чаще всего называемой core program, и подключаемых модулей, соответственно plugins. Модульный дизайн позволяет отделить логику основной программы от процесса выполнения проверок. Это, в свою очередь, дает возможность всем желающим без особых усилий расширять область применения Nagios через написание собственных подключаемых модулей.

Скачиваем последнюю версию главной программы и подключаемых модулей с официального сайта проекта http://www.nagios.org/download. На момент написания статьи были доступны версии nagios-1.0b6 и nagiosplug-1.3-beta1. Засучив рукава, приступим к установке главной программы.

```
# tar zxvf  nagios-1.0b6.tar.gz
# cd nagios-1.0b6
```

Теперь нужно решить, в какой каталог мы хотим инсталлировать Nagios. По умолчанию предполагается каталог /usr/local/nagios/. Думаю, нам тоже стоит придерживаться его, потому что в прилагающейся к дистрибутиву документации указан именно этот каталог. При последующем устранении ошибок или проблем с конфигурированием будет гораздо проще читать документацию. Создаем каталог, куда мы впоследствии будем проводить инсталляцию. Не совсем понятно, почему ее должен создавать администратор, а не программа инсталляции. Надеюсь, в следующих версиях этот недочет будет исправлен.

```
# mkdir /usr/local/nagios
```

С помощью скрипта adduser либо каким-нибудь другим способом создаем пользователя nagios, состоящего в группе nagios.

```
# adduser nagios

# ./configure --prefix=/usr/local/
nagios --with-cgi-url=/nagios/cgi-bin
--with-html-url=/nagios/ \
  --with-nagios-user=nagios --with-
nagios-grp=nagios --with-gd-lib=/usr/
local/lib \
  --with-gd-inc=/usr/local/include/gd
```

Давайте быстренько разберемся с назначением используемых при конфигурировании ключей.

- —prefix=/usr/local/nagios – сюда мы будем устанавливать файлы после компиляции. Вот и пригодилась созданная вручную директория /usr/local/nagios.
- —with-cgi-url=/nagios/cgi-bin – URL, с помощью которого мы будем обращаться к CGI-скриптам веб-интерфейса nagios. В конце URL символ «/» добавлять не нужно. В качестве примера для вымышленного сайта somesite.ru можно привести абсолютный URL одного из CGI-скриптов httt://somesite.ru/nagios/cgi-bin/statusmap.cgi.
- —with-html-url=/nagios/ – URL, где будут находиться html-файлы, в которых хранятся главное меню веб-интерфейса и документация.
- —with-nagios-user=nagios – пользователь, которому будут принадлежать файлы инсталляции.
- —with-nagios-grp=nagios – соответственно группа этого пользователя.

- —with-gd-lib=/usr/local/lib – тут у нас лежит библиотека GD.
- —with-gd-inc=/usr/local/include/gd – а здесь находятся ее заголовочные файлы.

Осмыслив все прочитанное и разобравшись с опциями команды configure, проводим сборку и инсталляцию.

```
# make all
# make install
```

Следующий шаг является необязательным, но желательно его все же выполнить. С помощью команды make install-init можно создать скрипт, который будет выполнять запуск Nagios после перезагрузки системы. Вдобавок к этому у нас появится возможность управлять работой процесса Nagios с помощью команд stop, reload, start, restart, reload, force-reload. Давайте подробнее рассмотрим каждую из этих команд:

- start – запускает процесс Nagios;
- stop – останавливает текущий процесс Nagios;
- restart – останавливает выполняющийся процесс Nagios и запускает новый;
- reload – отправляет процессу Nagios сигнал SIGHUP, заставляя его перечитать конфигурационные файлы, а затем продолжить мониторинг;
- force-reload – производит принудительную перезагрузку конфигурационных файлов;
- status – показывает статус работающего процесса.

Для FreeBSD файл скрипта должен располагаться в директории /usr/local/etc/rc.d и называться nagios.sh. В зависимости от типа операционной системы, инсталлятор должен сам выбрать подходящие права доступа, местоположение и имя скрипта. Например, для систем на основе Slackware должны получить соответственно /etc/rc.d/init.d и rc.nagios. Ну что же, давайте попытаемся выполнить установку скрипта.

```
# make install-init
```

Получаем следующие ошибки:

```
/usr/bin/install -c -m 755 -d -o
root -g root /usr/local/etc/rc.d
install: unknown group root
*** Error code 67

Stop in /tmp/nagios-1.0b6.
```

Жаль, но установка с наскоку не удалась. Ну и ладно, вот исправим ошибки программистов, писавших скрипт, и все заработает как надо. С помощью любимого текстового редактора начинаем редактировать файл Makefile. Идем на строку 32 и ищем символы:

```
INIT_OPTS=-o root -g root
```

Обдумав увиденную строку, понимаем, что под FreeBSD пользователь root входит в группу wheel. Вносим изменения.

```
INIT_OPTS=-o root -g wheel
```

Переходим на строку 154:

```
$(INSTALL) -m 774 $(INIT_OPTS)
daemon-init $(DESTDIR)$(INIT_DIR)/
nagios
```

Правильно задаем имя файла nagios.sh вместо nagios. Не стоит давать общий доступ на чтение для файла скрипта. Вполне хватит режима доступа 100. Измененная строка будет выглядеть так:

```
$(INSTALL) -m 100 $(INIT_OPTS)
daemon-init $(DESTDIR)$(INIT_DIR)/
nagios.sh
```

Сохраняем файл и снова выполняем команду инсталляции.

```
# make install-init
```

Итак, скрипт автозапуска создан, но он все еще не готов к работе. Все дело в том, что при перезагрузке файл nagios.sh будет выполнен без единого параметра командной строки, а для безукоризненной работы нужно запускать его с параметром start. Сейчас мы это исправим. Открываем файл для редактирования и выполняем поиск строки:

```
echo "Usage: nagios
{start|stop|restart|reload|force-
reload|status}"
```

Затем сразу после нее вставляем вот это:

```
/usr/local/etc/rc.d/nagios.sh start
```

Внесенными изменениями мы заставляем nagios.sh, запущеный без обязательных параметров, рекурсивно выполнить самого себя с параметром start.

Еще одна ошибка – это путь к файлу, в котором будут храниться внешние команды, переданные на выполнение процессу Nagios. После инсталляции подразумевается, что файл должен располагаться в /usr/local/nagios/var/rw/nagios.cmd. Но, к сожалению, директория /usr/local/nagios/var/rw/ автоматически не создается. К тому же не совсем понятен смысл создания отдельной директории для единственного файла nagios.cmd. Чтобы исправить вышеуказанную ошибку, ищем строку:

```
NagiosCmd=${prefix}/var/rw/
nagios.cmd
```

и заменяем ее на такую:

```
NagiosCmd=${prefix}/var/nagios.cmd
```

Теперь файл nagios.cmd будет создаваться в директории /usr/local/nagios/var/.

К сожалению, это не последняя проблема, связанная с файлом nagios.cmd. При каждом перезапуске процесса Nagios файл внешних команд сначала удаляется, а затем создается вновь. Беда в том, что создается nagios.cmd с правами процесса. Соответственно, он принадлежит пользователю nagios и группе nagios. На первый взгляд, здесь нет никакой проблемы. Поразмышляв некоторое время, мы понимаем, что главным поставщиком внешних команд для процесса Nagios будет веб-интерфейс. Сервер Apache обычно запускается с правами пользователя nobody группы nogroup. А это значит, что процесс веб-сервера не сможет вносить новые данные в файл nagios.cmd. Самым простым решением было бы дать доступ на запись всем, но с точки зрения безопасности такое решение выглядит довольно неприглядно. Вторым способом разрешения конфликта могло бы стать введение пользователя nobody в группу nagios. Но с безопасностью опять возникают неувязки. Самым лучшим выходом из этой ситуации является

частичная передача прав на файл пользователю nobody. В результате такой операции файлом должен владеть пользователь nobody и группа nagios. Таким образом, доступ к файлу получат и процесс nagios, и вебсервер. Ищем в файле nagios.sh вот такие строки:

```
sleep 1
status_nagios  nagios
```

и добавляем сразу за ними команду:

```
chown nobody $NagiosCmd
```

Покончив с файлом nagios.sh, можно быть уверенным, что уж теперь-то он будет работать как положено.

Заглянув в директорию /usr/local/nagios/, мы видим, что после инсталляции внутри нее образовалось еще 5 директорий.

```
# ls   /usr/local/nagios
bin      etc      sbin     share    var
```

Опишем, для чего нужна каждая из этих директорий:

- bin – здесь находится выполняемый файл основной программы. Он же демон Nagios;
- etc – конфигурационные файлы;
- sbin – тут лежат файлы CGI для веб-интерфейса;
- share – здесь находятся html-файлы веб-интерфейса и документация;
- var – файлы протоколов и данные о состоянии проверяемых объектов.

С помощью команды make install-config создаем файлы конфгурации, заполняем их значениями по умолчанию и помещаем в директорию /usr/local/nagios/etc/.

```
# make install-config
```

Можете поздравить себя с окончанием установки главной программы Nagios. Можно было сделать все то же самое с помощью портов или пакетов. К сожалению, на такой способ установки времени у меня не хватило. Поэтому сказать об удобстве или подводных камнях такого пути ничего не могу. Есть подозрение, что возможно возникновение проблем с конфигурированием, да и свежесть установленного программного обеспечения будет весьма сомнительна. Еще одной причиной не использовать порты стало желание заняться русификацией системы. Подробнее о результатах этого эксперимента будет рассказано в следующей статье.

Несмотря на то что инсталляция главной программы закончена, толку нам от этого пока никакого. Без установленных модулей Nagios совершенно бесполезен, так как не имеет возможности взаимодействовать с объектами, за которыми нужно наблюдать. Настало время приняться за инсталляцию этих самых модулей. С помощью make, поставлявшегося вместе с операционной системой, собрать их не удалось. Посему для компиляции будем пользоваться gmake.

```
  # tar zxvf nagiosplug-1.3-
beta1.tar.gz
  # cd nagiosplug-1.3-beta1
  # ./configure --prefix=/usr/local/
nagios --with-nagios-user=nagios --
with-nagios-grp=nagios
```

Значение ключей конфигурации аналогично тем, что мы использовали при компиляции главной программы. Почитать детальные объяснения о назначении каждого ключа можно с помощью команды ./configure –help.

```
  # gmake all
  # gmake install
```

После компиляции подключаемые модули устанавливаются в директорию /usr/local/nagios/libexec. Некоторые из них написаны на C, другие на perl. В принципе модуль может быть написан на любом языке. Каждый модуль является выполняемым файлом. Это дает возможность использовать модули не только в комплексе с Nagios, но и с другими программами. Можно выполнять проверки даже из командной строки. Каждый модуль должен поддерживать опцию –help или -h, иначе он не может считаться совместимым с Nagios. Такие жесткие требования позволяют легко разобраться с любым модулем. Давайте посмотрим, какие параметры командной строки должен получать модуль check_dns.

```
  # check_dns -h
  check_dns (nagios-plugins 1.3.0-
alpha1) 1.1.1.1
  The nagios plugins come with
ABSOLUTELY NO WARRANTY. You may
redistribute copies of the plugins
under the terms of the GNU General
Public License.
  For more information about these
matters, see the file named COPYING.
  Copyright (c) 1999 Ethan Galstad
(nagios@nagios.org)

  Usage: check_dns -H host [-s
server] [-t timeout]
         check_dns --help
         check_dns --version

  -H, --hostname=HOST
     The name or address you want to
query
  -s, --server=HOST
     Optional DNS server you want to
use for the lookup
  -t, --timeout=INTEGER
     Seconds before connection times
out (default: 10)
  -h, --help
     Print detailed help
  -V, --version
     Print version numbers and
license information

  This plugin uses the nslookup program
to obtain the IP address for the given
host/domain query.  A optional DNS
server to use may be specified.  If no
DNS server is specified, the default
server(s) specified in /etc/resolv.conf
will be used.
```

Прочитав и обдумав вывод предыдущей команды, можно понять, что модуль check_dns принимает один обязательный параметр –H и два необязательных параметра -s, -t. Итак, по порядку: –H – имя или адрес проверяемого хоста. В параметре -s определяется адрес сервера DNS, используемого для перевода имени в адрес, если в первом параметре вместо адреса передано имя хоста. Затем -t устанавливает тайм-аут, по истечении которого служба считается нефункционирующей. Для проверки сервиса DNS машины с адресом 192.168.10.254 нужно выполнить следующую команду:

```
  # check_dns -H 192.168.10.254
  DNS ok - 0 seconds response time,
Address(es) is/are  192.168.10.254
```

Интуиция странным образом подсказывает мне, что DNS работает нормально.

Закончив установку и разобравшись с механизмом работы подключаемых модулей, перейдем к настройке Nagios. Давайте осмотрим содержимое директории конфигурационных файлов.

```
  # cd /usr/local/nagios/etc
```

```
  # ll
  total 92
  -rw-rw-r--  1 nagios  nagios  17028
Nov 11 15:41 cgi.cfg-sample
  -rw-rw-r--  1 nagios  nagios   4480
Nov 11 15:41 checkcommands.cfg-sample
  -rw-rw-r--  1 nagios  nagios   1577
Nov 11 15:41 contactgroups.cfg-sample
  -rw-rw-r--  1 nagios  nagios   1485
Nov 11 15:41 contacts.cfg-sample
  -rw-rw-r--  1 nagios  nagios   1651
Nov 11 15:41 dependencies.cfg-sample
  -rw-rw-r--  1 nagios  nagios   2022
Nov 11 15:41 escalations.cfg-sample
  -rw-rw-r--  1 nagios  nagios   1658
Nov 11 15:41 hostgroups.cfg-sample
  -rw-rw-r--  1 nagios  nagios   5774
Nov 11 15:41 hosts.cfg-sample
  -rw-rw-r--  1 nagios  nagios   4277
Nov 11 15:41 misccommands.cfg-sample
  -rw-rw-r--  1 nagios  nagios  21332
Nov 11 15:41 nagios.cfg-sample
  -rw-rw----  1 nagios  nagios   3074
Nov 11 15:41 resource.cfg-sample
  -rw-rw-r--  1 nagios  nagios  17668
Nov 11 15:41 services.cfg-sample
  -rw-rw-r--  1 nagios  nagios   1594
Nov 11 15:41 timeperiods.cfg-sample
```

В названии каждого файла есть фрагмент -sample, значит, все вышеперечисленные файлы являются не полноценными конфигурациями, а всего лишь примерами. На случай, если нам вдруг захочется посмотреть, что там у них внутри, скопируем их в директорию sample. Может быть, когда-то они нам еще пригодятся.

```
  # mkdir sample
  # cp * ./sample
```

Переименовываем все файлы, отсекая цепочку символов –sample. Таким образом мы готовим файлы к тому, чтобы Nagios смог их заметить и правильно воспринять. В результате должно получиться что-то вроде:

```
  # ll
  total 94
  -rw-rw-r--  1 nagios  nagios  17028
Nov 11 16:13 cgi.cfg
  -rw-rw-r--  1 nagios  nagios   4480
Nov 11 16:13 checkcommands.cfg
  -rw-rw-r--  1 nagios  nagios   1577
Nov 11 16:13 contactgroups.cfg
  -rw-rw-r--  1 nagios  nagios   1485
Nov 11 16:13 contacts.cfg
  -rw-rw-r--  1 nagios  nagios   1651
Nov 11 16:13 dependencies.cfg
  -rw-rw-r--  1 nagios  nagios   2022
Nov 11 16:13 escalations.cfg
  -rw-rw-r--  1 nagios  nagios   1658
Nov 11 16:13 hostgroups.cfg
  -rw-rw-r--  1 nagios  nagios   5774
Nov 11 16:13 hosts.cfg
  -rw-rw-r--  1 nagios  nagios   4277
Nov 11 16:13 misccommands.cfg
  -rw-rw-r--  1 nagios  nagios  21332
Nov 11 16:13 nagios.cfg
  -rw-rw----  1 nagios  nagios   3074
Nov 11 16:13 resource.cfg
  drwxr-xr-x  2 root    nagios    512
Nov 11 16:07 sample
  -rw-rw-r--  1 nagios  nagios  17668
Nov 11 16:13 services.cfg
  -rw-rw-r--  1 nagios  nagios   1594
Nov 11 16:13 timeperiods.cfg
```

К сожалению, файлы все еще не готовы к употреблению. Nagios даже не сможет стартовать, если мы попытаемся его запустить. Разобраться в содержимом файлов-примеров, созданных во время инсталляции, довольно сложно. Самым лучшим выходом из подобной ситуации будет уничтожение всех конфигурационных данных и создание их вручную под моим чутким руководством. Такой подход принесет более глубокое понимание используемой методики настройки. Обнуляем все необходимые файлы.

```
  # cp /dev/null hosts.cfg
services.cfg contacts.cfg
contactgroups.cfg hostgroups.cfg
  # cp /dev/null dependencies.cfg
escalations.cfg
```

Начиная с версии 1.0b2, в Nagios появилась возможность использовать шаблоны внутри конфигурационных файлов. Такая методика настройки позволяет многократно снизить время создания рабочей конфигурации. Перед нами стоит задача описать два хоста mail.regata.ru и proxy.regata.ru. Каждый сервер, находящийся под наблюдением, должен быть описан в файле хостов. Поэтому первым делом заполняем именно этот файл.

Содержимое файла hosts.cfg

```
  # Описываем шаблон хоста
  define host{
   name
generic-host
  # Имя шаблона - будем ссылаться на
  # него позже при описании каждого
  # хоста
  notifications_enabled          1
  # Включаем уведомления
  event_handler_enabled          1
  # Включаем обработчик событий
  flap_detection_enabled         1
  # Включаем обнаружение мерцания
  process_perf_data              1
  # Собирать статистику об эффектив-
  # ности работы процесса
  retain_status_information      1
  # Сохранять статусную информацию
  # между перезагрузками программы
  retain_nonstatus_information   1
  # Сохранять нестатусную информацию
  # между перезагрузками программы
  register                       0
  # Указываем, что все вышеописанное -
  # это шаблон. Запрещаем регистриро-
  # вать это описание как хост
  }

  # Описываем хост по имени mail
  define host{
  use               generic-host
# Ссылка на используемый шаблон
host_name         mail
# Имя хоста
alias             Mail Server
# Псевдоним хоста
address           mail.regata.ru
# Имя хоста в формате fqdn или его
# адрес
check_command     check-host-alive
# Команда проверки хоста выполняется
# обычно с помощью ping. Подробнее
# можно почитать в файле
# checkcommands.cfg
max_check_attempts      10
# Количество попыток повторного тес-
# тирования, после того как одна из
# попыток возвратила ошибочный
# статус.

 notification_interval   120
# Интервал в минутах, по прошествию
# которого нужно повторно отсылать
# уведомление, если сервер все еще
# не работает.
notification_period     24x7
# Период времени, в течение которого
# серверу разрешено беспокоить адми-
# нистратора своими уведомлениями.
# В данном случае это круглые сутки.
# Подробнее о периодах можно
# почитать в файле timeperiods.cfg.
notification_options    d,u,r
# Список событий, при наступлении
# которых необходимо отсылать
# уведомления. Соответственно d,u,r
# (DOWN, UNREACHABLE, RECOVERY)
# означает события "работает",
# "недоступен", "восстановлен".
}

# Описываем хост по имени proxy
define host{
use               generic-host
host_name         proxy
# Имя хоста. Также стоит обратить
# внимание на изменившиеся записи
# alias и address.
alias             Proxy Server
address           proxy.regata.ru
check_command     check-host-alive
max_check_attempts      10
notification_interval   120
notification_period     24x7
notification_options    d,u,r
}
```

Каждый хост должен состоять хотя бы в одной группе хостов. Опираясь на группы хостов, Nagios сможет определить, какую из групп администраторов нужно оповещать. Данные, полностью описывающие группы хостов, находятся в файле hostgroups.cfg. Формат этого файла настолько прост, что механизм шаблонов применять смысла нет. Посмотрим, чем заполнен этот файл.

Содержимое файла hostgroups.cfg

```
define hostgroup{
 hostgroup_name   regata-servers
# Имя группы
 alias            Regata Servers
# Псевдоним группы
 contact_groups   regata-admins
# Список групп контактов, которые
# нужно уведомлять
 members          mail proxy
```

```
# Список серверов, принадлежащих к
# группе
}
```

Пришло время вплотную заняться описанием сервисов, за которыми нужно наблюдать. Как и во всех остальных конфигурационных файлах, первым делом необходимо создать шаблон сервиса. Мы разберем его подробно. Затем второй и третьей записью будут описаны сервисы HTTP и PING, базирующиеся на хосте proxy.regata.ru. Соответственно, далее мы работаем с mail.regata.ru, на котором у нас базируются SMTP и IMAP.

Содержимое файла services.cfg

```
# Описываем шаблон сервиса
define service{
name    generic-service
# Имя шаблона
active_checks_enabled   1
# Включить активные проверки
passive_checks_enabled   1
# Принимать результаты пассивных
# проверок
parallelize_check   1
# Активные проверки лучше выполнять
# параллельно - такой подход
# повышает скорость работы
obsess_over_service   0
# Эту опцию стоит включать только
# при создании распределенной
# системы мониторинга
check_freshness   0
# Следить за свежестью результатов
# проверок. По умолчанию отключено
notifications_enabled   1
# Уведомления включены
event_handler_enabled   1
# Включить обработчики событий
flap_detection_enabled   1
# Использовать обнаружение мерцания
process_perf_data   1
# Собирать данные об эффективности
# выполнения проверок
retain_status_information 1
# Сохранять информацию о статусе
# между перезапусками Nagios
retain_nonstatus_information 1
# Сохранять нестатусную информацию
# между перезапусками Nagios
register   0
#  Запрещаем регистрировать это
# описание как сервис
 }

# Описываем сервис HTTP для хоста
# proxy.regata.rudefine service{
use    generic-service
# Имя используемого шаблона
hostname proxy.regata.ru
# Имя хоста
service_description HTTP
# Описание сервиса
is_volatile 0
# Для стандартных сервисов лучше
# оставить значение 0. К нестандарт-
# ным сервисам стоит относить те
# сервисы, которые после каждой
# проверки автоматически возвращаются
# в состояние "OK" вне зависимости
# от режима, в котором они
# находились до проверки.
check_period    24x7
# Период, в течение которого можно
# выполнять проверки
max_check_attemps    3
# Максимальное количество повторных
# проверок
normal_check_interval   5
# Интервал между нормальными
# проверками
retry_check_interval   1
# Интервал между повторными провер-
# ками. Применяется, если нормальная
# проверка завершилась неудачно
contact_groups  regata-admins
# Группа контактов, используемая
# для оповещения
notification_interval    120
# Интервал (в минутах), после
# которого нужно послать повторное
# уведомление, если сервис так и не
# восстановился
notification_period    24x7
# Период, в течение которого можно
# производить отправку уведомлений
notification_options    w,u,c,r
# Список событий, при наступлении
# которых необходимо
# отсылать уведомления.
check_command check_http
#  Имя команды, выполняемой для
# проверки сервиса
 }


# Описываем сервис PING для хоста
# proxy.regata.rudefine service{
use    generic-service
# Имя используемого шаблона
hostname proxy.regata.ru
# Имя хоста
service_description PING
# Описание сервиса
is_volatile 0
check_period    24x7
# Период, в течение которого можно
# выполнять проверки
max_check_attemps    3
# Максимальное количество повторных
# проверок
normal_check_interval   5
# Интервал между нормальными
# проверками
retry_check_interval   1
# Интервал между повторными провер-
# ками. Применяется, если нормаль-
# ная проверка завершилась неудачно
contact_groups  regata-admins
# Группа контактов, используемая
# для оповещения
notification_interval    120
# Интервал (в минутах), после
# которого нужно послать повторное
# уведомление, если сервис так и не
# восстановился
notification_period    24x7
# Период, в течение которого можно
# производить отправку уведомлений
notification_options    w,u,c,r
# Список событий, при наступлении
# которых необходимо
# отсылать уведомления.
check_command check_ping
#  Имя команды, выполняемой для
# проверки сервиса
 }

# Описываем сервис SMTP для хоста
# mail.regata.rudefine service{
use    generic-service
hostname mail.regata.ru
service_description  SMTP
# Описание сервиса
is_volatile 0
check_period    24x7
max_check_attemps    3
normal_check_interval   5
retry_check_interval   1
contact_groups  regata-admins
notification_interval    120
notification_period    24x7
notification_options    w,u,c,r
check_command check_smtp
# Имя команды, выполняемой для
# проверки сервиса
 }

# Описываем сервис IMAP для хоста
# mail.regata.rudefine service{
use    generic-service
hostname mail.regata.ru
service_description  IMAP
# Описание сервиса
is_volatile 0
check_period    24x7
max_check_attemps    3
normal_check_interval   5
retry_check_interval   1
contact_groups  regata-admins
notification_interval    120
notification_period    24x7
notification_options    w,u,c,r
check_command check_imap
# Имя команды, выполняемой для
# проверки сервиса
 }
```

Как вы, возможно, сумели догадаться, имена команд, вызываемых для выполнения проверки того или иного сервиса, определяются параметром check_command. Сами же команды описываются в файле checkcommands.cfg. По странному стечению обстоятельств команда check_imap в этом файле не описана, несмотря на то что среди модулей, установленных в директорию /usr/local/nagios/libexec/, файл check_im-ap присутствует. Видимо, это еще одна ошибка программистов Nagios. Ну и ладно, нам не привыкать делать все собственными руками. Разместим на любом месте файла checkcommands.cfg следующие строки:

```
# 'check_imap' command definition
define command{
        command_name    check_imap
        command_line    $USER1$/
check_imap -H $HOSTADDRESS$
        }
```

Создав описание сервисов, самое время перейти к определению списка людей, которым система будет отсылать оповещения. В терминологии Nagios каждая запись в файле contacts.cfg, описывающая человека, – это контакт. Формат записи довольно прост, поэтому и в данном случае шаблоны нам не пригодятся.

Содержимое файла contacts.cfg

```
define contact{
contact_name     tigrisha
# Имя контакта
alias            Andrei Beshkov
# Имя человека
service_notification_period    24x7
```

```
  # Период времени, в течение которого
  # разрешено посылать уведомления
  # о состоянии сервисов
  host_notification_period        24x7
  # Период времени, в течение которого
  # разрешено посылать уведомления
  # о состоянии хостов
  service_notification_options  w,u,c,r
  # Список событий сервисов, при
  # наступлении которых необходимо
  # отсылать уведомления.
  host_notification_options    d,u,r
  # Список событий хостов, при
  # наступлении которых необходимо
  # отсылать уведомления
  service_notification_commands
notify-by-email, notify-by-epager
  # Команды рассылки уведомлений о
  # событиях сервиса
  host_notification_commands     host-
notify-by-email, host-notify-by-epager
  # Команды рассылки уведомлений о
  # событиях хоста
  email      admin@regata.ru
  # Почтовый адрес
  pager      150345865@pager.icq.com
  # Адрес системы пейджинга
   }

  define contact{
  contact_name      amon
  # Имя контакта
  alias             Dmitry Larin
  # Имя человека
  service_notification_period    24x7
  # Период времени, в течение которого
  # разрешено посылать уведомления о
  # состоянии сервисов
  host_notification_period       24x7
  # Период времени, в течение которого
  # разрешено посылать уведомления о
  # состоянии хостов
  service_notification_options  w,u,c,r
  # Список событий сервисов, при
  # наступлении которых необходимо
  # отсылать уведомления.
  host_notification_options    d,u,r
  # Список событий хостов, при
  # наступлении которых необходимо
  # отсылать уведомления.
  service_notification_commands
notify-by-email, notify-by-epager
  # Команды рассылки уведомлений о
  # событиях сервиса
  host_notification_commands     host-
notify-by-email, host-notify-by-epager
  # Команды рассылки уведомлений о
  # событиях хоста
  email      amon@regata.ru
  # Почтовый адрес
  pager      150345865@pager.icq.com
  # Адрес системы пейджинга
  }
```

Каждый контакт должен принадлежать как минимум одной контактной группе. Давайте создадим такую группу для контактов, описанных выше.

Содержимое файла contactgroups.cfg

```
  define contactgroup{
   contactgroup_name     regata-admins
  # Имя контактной группы
   alias               regata.ru Admins
  # Псевдоним группы
   members             tigrisha amon
  # Список контактов, состоящих в
  # группе
  }
```

Файл dependencies.cfg отвечает за зависимости между хостами. Ну а файл escalations.cfg, в свою очередь, описывает эскалацию оповещений. В связи с тем, что рассматриваемый нами пример состоит всего из двух хостов, файлы зависимостей и эскалации нам не пригодятся. Отсюда вытекает приятный факт, говорящий, что необходимости заполнять их у нас нет. Возможно, в следующей статье я подробно опишу работу с файлами зависимостей и эскалации.

Конфигурирование, наконец-то, завершено. Перед пробным запуском стоит упомянуть, что Nagios может быть запущен четырьмя способами. Давайте перечислим их.

- Вручную – как процесс переднего плана.
- Вручную – как фоновый процесс.
- Вручную – как демон.
- Автоматически – как демон, при старте системы.

Для проверки конфигурации воспользуемся способом № 3. Собравшись с духом, можно попытаться запустить систему и начать мониторинг. Воспользуемся для этих целей скриптом nagios.sh.

```
  # /usr/local/etc/rc.d/nagios.sh
```

После этого должно произойти одно из двух: либо Nagios начнет работать, либо на экране появится следующая ошибка:

```
Starting network monitor: nagios
/bin/sh:  -l: unrecognized option
```

Не пугайтесь, если вам довелось наступить на эти грабли. Приведенное выше сообщение об ошибке значит, что операционная система, на которой вы работаете, не поддерживает опцию -l команды su. В случае с FreeBSD дело обстоит именно таким образом. Поэтому команда должна выглядеть так: su -. В файле /usr/local/etc/rc.d/ nagios.sh ищем строку, содержущую символы su -l, и удаляем из нее букву l. Вот теперь-то все должно работать как положено. Снова запускаем Nagios. Проверяем, как он себя чувствует с помощью ключа командной строки status.

```
  # /usr/local/etc/rc.d/nagios.sh
status
    PID  TT  STAT      TIME COMMAND
    101  ??  Ss     0:20.81 /usr/
local/nagios/bin/nagios -d /usr/local/
nagios/etc
```

Если, несмотря на все принятые предосторожности, Nagios продолжает кричать об ошибках, то стоит убедиться, что директория /usr/local/ nagios/ вместе со всем своим содержимым принадлежит пользователю nagios и группе nagios. В случае ежели подобные ожидания не оправдываются, выполняем смену владельца.

```
  # chown -R nagios:nagios /usr/
local/nagios/
```

Ну если и это не помогло, то вам стоит запустить Nagios в режиме отладки.

```
  #  /usr/local/nagios/bin/nagios -v
/usr/local/nagios/etc/nagios.cfg
```

В режиме отладки тексты сообщений обо всех найденных неполадках будут более подробными. Это обстоятельство поможет легче и быстрее обнаружить ошибки. Процедура исправления ошибок в таком случае довольно проста. Исходя из этого, я надеюсь, что у вас все получится, если не с первого, то со второго раза точно.

Nagios уже выполняет мониторинг сервисов и при наступлении критических событий отправляет оповещения всем, кому положено. Но по одним оповещениям понять, что происходит, не так уж и легко. Согласитесь, что в интерактивном режиме смотреть статистику работы и разбираться с проблемами гораздо легче. Исходя из этого, давайте обратим наши взоры к процедуре настройки веб-интерфейса. В данной статье предполагается, что на машине, занимающейся системой мониторинга, работает веб-сервер Apache. Он обязан быть хотя бы минимально настроен и должен функционировать достаточно стабильно.

В файле настроек Apache httpd.conf ищем такой фрагмент текста:

```
      <Directory "/usr/local/apache/
cgi-bin">
          AllowOverride None
          Options None
          Order allow,deny
          Allow from all
      </Directory>
```

Сразу же после них добавляем такие строки:

```
  ScriptAlias /nagios/cgi-bin/ "/usr/
local/nagios/sbin/"
      <Directory "/usr/local/nagios/
sbin/">
          AllowOverride AuthConfig
          Options ExecCGI
          Order allow,deny
          Allow from all
      </Directory>

  Alias /nagios/ /usr/local/nagios/
share/
     <Directory "/usr/local/nagios/
share/">
         AllowOverride AuthConfig
         Options None
         Order allow,deny
         Allow from all
     </Directory>
```

Выполнив эту процедуру, мы создали два псевдонима. Первый – для cgi директории nagios, находящейся в «/usr/local/nagios/sbin/». Доступ к ней можно получить, выполнив подобный http-запрос: http://ваш сайт/nagios/cgi-bin/. Второй псевдоним указывает, что в директории /usr/local/nagios/share/ находятся html-файлы веб-интерфейса и справочной документации . Просмотреть эти страницы можно, посетив адрес: http://ваш сайт /nagios/. При попытке посетить эти страницы, получаем ошибку. Для того чтобы псевдонимы и авторизация заработали, нужно создать файл .htaccess в директории /usr/local/nagios/sbin/ и внести в него следующие строки:

```
  AuthName "Nagios Access"
  AuthType Basic
  AuthUserFile /usr/local/nagios/etc/
htpasswd.users
  require valid-user
```

Для вступления в силу выполненных изменений осталось лишь перезапустить Apache.

```
  # /usr/local/apache/bin/apachectl
restart
```

Таким образом мы объясняем веб-серверу, что доступ к файлам директории /usr/local/nagios/sbin/ может получить только авторизованный пользователь. Если есть желание, чтобы пользователи не смогли войти даже на главную страницу Nagios без ввода пароля, то нужно скопировать файл .htaccess из директории /usr/local/nagios/sbin/ в /usr/local/nagios/share/.

Список паролей и имен пользователей должен находиться в файле /usr/local/nagios/etc/htpasswd.users, который на данный момент еще не существует. Заводим нового пользователя и заодно с помощью ключа –c создаем файл.

```
  # /usr/local/apache/bin/tpasswd -c
/usr/local/nagios/etc/htpasswd.users
tigrisha
  New password: ******
  Re-type new password: ******
  Adding password for user tigrisha
```

С помощью той же команды, но без ключа –c, создаем еще одного пользователя.

```
  # /usr/local/apache/bin/htpasswd  /
usr/local/nagios/etc/htpasswd.users
amon
  New password: ******
  Re-type new password: ******
  Adding password for user amon
```

Теперь давайте займемся конфигурационным файлом /usr/local/nagios/etc/cgi.cfg. Нам нужно изменить некоторые параметры:

```
  authorized_for_system_information=tigrisha,
amon
  # Список пользователей, которым
  # разрешен просмотр информации о
  # работе процесса Nagios

  authorized_for_configuration_information=
tigrisha, amon
  # Список пользователей, которым
  # разрешен просмотр информации о
  # конфигурации всех хостов и
  # сервисов. По умолчанию пользователь
  # может смотреть конфигурацию только
  # хостов и сервисов, принадлежащих
  # к его контактной группе.


  authorized_for_system_commands=tigrisha,
amon
  # Список пользователей, авторизо-
  # ванных для выполнения через
  # cmd.cgi команд управления
  # процессом Nagios.

  authorized_for_all_services=tigrisha,
amon
  authorized_for_all_hosts=tigrisha,
amon
  # Эти два параметра определяют список
  # пользователей, которым разрешен
  # просмотр информации обо всех
  # наблюдаемых хостах и сервисах.
  # По умолчанию пользователь может
  # видеть только те хосты и сервисы,
  # которые принадлежат к его
  # контактной группе.

  refresh_rate=10
  # Частота обновления (в секундах)
  # информации, просматриваемой через
  # веб-интерфейс. По умолчанию
  # установлено 90 секунд, но нам
  # такой большой интервал не подходит.
  # Поэтому поставим 10 секунд.
```

Теперь в файл nagios.cfg вносим следующие изменения:

```
  use_authentication=1
  # Включаем аутентификацию
  # пользователей

  check_external_commands=1
  # Разрешаем обрабатывать команды,
  # передаваемые процессу Nagios
  # через веб-интерфейс.

  command_file=/usr/local/nagios/var/
nagios.cmd
  # Местоположение файла внешних
  # команд.
```

Для того чтобы внесенные изменения вступили в силу, перезапускаем процесс Nagios:

```
  # /usr/local/etc/rc.d/nagios.sh
restart
```

Если все сделали правильно, то теперь самое время наслаждаться отлично работающей системой мониторинга. Мы установили наблюдение за двумя тестовыми хостами. Процедуру добавления остальных хостов оставляю читателю в качестве само-