Postfix: диагностируем и устраняем неисправности

Postfix прост и надежен в эксплуатации, словно автомат Калашникова. Но все же неискоренимое человеческое любопытство нет-нет, да и заставляет нас задумываться над вопросами: Что будет, если в один прекрасный день Postfix перестанет работать? Смогу ли я понять, почему это произошло? Удастся ли мне его починить?

В предыдущих статьях [1, 2] мы говорили о том, как Postfix устроен изнутри, почему я считаю его одним из лучших SMTP MTA и как с его помощью построить корпоративный почтовый сервер. Многие читатели, ознакомившись с этими материалами, принялись за создание собственных систем на основе Postfix. Судя по откликам, 99% из них преуспели в этом начинании, у подавляющего большинства почтовый сервер работает в штатном режиме без каких-либо проблем уже год или более.

Все, о чем будет говориться сегодня, проверялось на версиях Postfix 2.2 под FreeBSD, Solaris и несколькими дистрибутивами Linux. Под всеми перечисленными системами приемы, которым я вас научу, работают практически одинаково. Мелкие различия в формате вывода информации на экран или названии директорий, в которых лежит тот или иной файл, в данном случае несущественны. Поэтому я думаю, что у вас не возникнет проблем с применением полученных знаний.

Начинаем диагностику

Итак, представим, что неприятности все же случились. Произошло это по вине неловкого администратора или аппаратного сбоя — неважно, наша задача — найти неполадки и исправить их. Каменщики обычно пляшут от печки, а мы начнем диагностику с проверки, запущен ли главный процесс Postfix. Сделать это проще всего командой:

```
# ps -ax | grep master
22628         ?         Ss                     0:00
/usr/lib/postfix/master
```

В некоторых дистрибутивах Linux того же эффекта можно добиться командой:

```
# service postfix status
```

Убедившись в том, что процесс запущен, нужно переходить к следующему этапу и проверить, принимает ли он входящие соединения на 25-м порту нужных нам интерфейсов:

```
# netstat -na  | grep LISTEN | grep  25
tcp       0      0 0.0.0.0:25       0.0.0.0:*       LISTEN
```

Теперь неплохо было бы приказать Postfix самому провести внутреннюю диагностику.

```
# postfix check
postfix: fatal: /etc/postfix/main.cf, line 100: missing "="
after attribute name: "mynetworks_style"
```

Как видите, в данном случае проблема состоит в неправильном синтаксисе файла main.cf. Впрочем, эта команда работает не только с конфигурационными файлами, заодно она позволяет убедиться, что все необходимые служебные файлы и директории имеют надлежащие права и принадлежат кому положено. Стоит отметить тот факт, что, если в результате проверки будут обнаружены проблемы с какими-либо директориями, Postfix попытается самостоятельно исправить их атрибуты. В случае успешности этого действия на экране не появится никаких предупреждений. Ну а если самостоятельно разобраться с проблемами не удастся, сообщения об ошибках будут достаточно детализированы и легко понятны даже начинающему администратору.

Иногда случается так, что служебные директории бывают полностью удалены, в этом случае Postfix постарается самостоятельно восстановить их. Единственное, чего он не умеет создавать — бинарные выполняемые файлы взамен утраченных.

После выполнения команды check неплохо было бы перезагрузить Postfix, в идеале это требуется редко, но иногда лучше перестраховаться:

```
# postfix stop
# postfix start
```

Анализируем протоколы

Если все вышеперечисленные действия не спасли «отца русской демократии» и почта все еще не работает – значит пришло время заняться анализом протоколов работы почтовой системы. Обычно они находятся в /var/log/mail/ или /var/log/maillog. Если вам не удалось найти нужный файл, то в зависимости от того, какая система syslog у вас используется, посмотрите в /etc/syslog.conf или /etc/syslog-ng.conf. Поищите в этих файлах строки, где встречается mail, и вам сразу все станет понятно. Итак, с местонахождением протоколов мы определились, теперь следует посмотреть, что в них записано. В общем виде записи в протоколе должны выглядеть так:

```
Jun 12 17:23:41 altlinux postfix/master[23846]: daemon started
-- version 2.1.6
```

Я уже упоминал, что Postfix является не монолитной программой, а целым содружеством демонов, во главе которых стоит процесс master. С точки зрения безопасности это неоспоримое преимущество, потому что отказ одного компонента не приведет к падению всех остальных. Но в то же время благодаря такому подходу каждая программа подсистемы самостоятельно отвечает за протоколирование результатов своей работы. Получается, что, несмотря на свое главенство, процесс master обладает лишь необходимым минимумом информации о подчиненных процессах. К примеру, он может сообщить, что, судя по коду ошибки, у процесса с определенным ID проблемы, но сути их master все равно не знает. В большинстве случаев поиск по ID процесса, на который пожаловался master, дает исчерпывающую информацию о неполадках в системе.Каждая запись состоит из нескольких компонентов. Перечисляем по порядку: дата и время события, имя

хоста, имя компонента postfix и ID процесса, ну и, наконец, само сообщение.

Для указания серьезности ошибки служат специально зарезервированные слова. Давайте рассмотрим их значение.

- **panic** – произошло что-то из ряда вон выходящее. Вы, видимо, нашли какую-то серьезную ошибку в работе Postfix, которую вряд ли сможете исправить самостоятельно. Стоит отметить, что за несколько лет эксплуатации Postfix мне ни разу не приходилось встречать сообщения об ошибках подобного типа.
- **fatal** – продолжение работы невозможно до тех пор, пока ошибка не будет исправлена. Обычно причиной таких сообщений становится отсутствие важных файлов или прав на доступ к каким-либо объектам.
- **error** – ошибка может быть фатальной или нет, но обычно продолжение работы системы невозможно.
- **warning** – предупреждение о не фатальных ошибках. Может указывать на некоторые несущественные проблемы или ошибки в конфигурации.

Посмотрим на одну из самых типичных записей об ошибке:

```
Jun 12 17:41:28 altlinux postfix/smtpd[24506]: fatal: open database /etc/postfix/helo_restriction.db: No such file or directory
Jun 12 17:41:29 altlinux postfix/master[23846]: warning: process /usr/lib/postfix/smtpd pid 24506 exit status 1
Jun 12 17:41:29 altlinux postfix/master[23846]: warning: /usr/lib/postfix/smtpd: bad command startup -- throttling
```

DB-файлы – это бинарная форма вспомогательных таблиц, используемых Postfix во время работы. Обычно она создается из текстовых файлов специального формата с помощью команды postmap <имя текстового файла>. Начинающие администраторы при попытке расширить функциональность postfix довольно часто забывают создавать вспомогательные таблицы, которые сами же прописали в main.cf. Судя по сообщению от

postfix/master[23846], процесс postfix/smtpd[24506] аварийно завершил свою работу, потому что не смог найти файл служебных таблиц /etc/postfix/helo_restriction.db.

Обычно Postfix записывает в файлы протокола достаточно информации для того, чтобы понять, в чем именно заключается проблема, но иногда бывает полезно увеличить «разговорчивость» некоторых компонентов системы. Для этого открываем файл master.cf, выбираем нужный компонент системы и дописываем к его ключам запуска -v. К примеру, строка, заставляющая демона cleanup подробнее отчитываться о своих действиях, выглядит вот так:

```
cleanup   unix  n       -       -       -       0       cleanup -
v
```

Выполняем команду postfix reload и смотрим, что происходит. Если полученные сведения все еще не устраивают нас, добавляем через пробел к ключам запуска еще одну -v, и так до тех пор, пока не получим необходимую подробность. Подобный метод можно применять к любому из компонентов Postfix. При проблемах с входящими SMTP-соединениями добавляют подробности компоненту smtpd. Если нас интересует доставка, то ключ надо добавить к параметрам демона очереди qmgr. Плюс зависимости от направления, в котором должно уйти письмо, внести такие же изменения в параметры вызова агентов доставки smtp, lmtp, pipe, local, virtual. Также можно глобально указать ключ -v для программы postfix, которая в свою очередь передает параметры в master.

Делается это, к примеру, вот так:

```
# /usr/sbin/postfix -v
```

Как и в предыдущих примерах, количество повторений -v указывает, насколько подробными должны быть выводимые сообщения.

Следующей весьма полезной для отладки командой является postconf. Она выводит на экран огромное количество информации

о состоянии внутренних переменных и таким образом позволяет посмотреть, каковы на данный момент настройки postfix. Кстати, стоит отметить, что значение практически всех переменных, выводимых postconf, вы можете переопределить в файле main.cf. К примеру, узнать версию Postfix можно, выполнив:

```
# postconf | grep version
```

А с помощью postconf -m можно увидеть, с какими типами баз данных умеет работать ваш экземпляр Postfix.

```
# postconf -m
btree
cidr
environ
fail
hash
inline
internal
memcache
mysql
pcre
pipemap
proxy
randmap
regexp
socketmap
static
tcp
texthash
unionmap
unix
```

Представим, что нам необходимо посмотреть, что стало с тем или иным письмом. Подобная задача довольно часто встает перед администратором почтовой системы, когда пользователь звонит и жалуется, что ему два дня назад отправили письмо, а он его до сих пор не получил. Отследить путь прохождения письма довольно просто: нужно открыть файл протокола, найти сообщения о соединении с интересующего хоста. К примеру, нам интересно увидеть, куда делось письмо от vasa@unreal.net к

ira@unreal.net. Ищем в протоколе vasa@unreal.net, например, с помощью grep и находим следующие строки:

```
Jun 12 19:11:19 altlinux postfix/smtpd[27445]: connect from clif.unreal.net[10.10.21.75]
Jun 12 19:12:09 altlinux postfix/smtpd[27445]: F29D3562E: client=clif.unreal.net[10.10.21.75]
Jun 12 15:12:27 altlinux postfix/cleanup[27482]: F29D3562E: message-id=20050612151145.F29D3562E@mailhost.unreal.net
Jun 12 19:12:27 altlinux postfix/qmgr[27342]: F29D3562E: from=, size=363, nrcpt=1 (queue active)
Jun 12 15:12:29 altlinux postfix/smtpd[27445]: disconnect from clif.unreal.net[10.10.21.75]
```

Из них следует, что письмо было принято от clif.unreal.net[10.10.21.75], и ему присвоен ID F29D3562E почтовой очереди. Выполним следующую команду:

```
# grep /var/log/maillog F29D3562E
Jun 12 19:12:27 altlinux postfix/local[27491]: F29D3562E: to=, relay=local, delay=42, status=sent
(delivered to command: /usr/bin/procmail -a $DOMAIN -d $LOGNAME)
Jun 12 19:12:27 altlinux postfix/qmgr[27342]: F29D3562E: removed
```

Проверим доставку

Судя по выводу, письмо было успешно доставлено в почтовый ящик пользователя, что и требовалось доказать.

Иногда бывает нужно проверить, как передается почта между нашим сервером и каким-либо другим. В случае если администрируемый сервер доступен для нас по SMTP, выполнить это достаточно просто. А что делать, если сервер предназначен для внутренней почты компании и находится в закрытой сети, соответственно у нас к нему есть доступ только по ssh или telnet? Первый способ – проверить прохождение почты: отправить письмо из командной строки с помощью команды mail и затем проанализировать то, что будет написано в файлах протокола. Большинство администраторов делает именно так, но есть более

удобный путь. С помощью команды sendmail можно проверить прохождение почты гораздо более простым способом.

```
# sendmail -v vasa@unreal.net
```

После доставки письма к vasa@unreal.net все данные, касающиеся этого факта, будут не только записаны в протокол, но еще и отправлены почтой пользователю, от имени которого была запущена программа. В нашем случае это пользователь root.

```
# sendmail -bv vasa@unreal.net
```

Если воспользоваться такой командой, то на финальной стадии доставка письма будет прервана и пользователь не получит тестового письма, но все записи протокола так же, как и в предыдущем случае, придут к нам в почтовый ящик.

Еще одним удобным способом посмотреть, что происходит во время SMTP-сессии, является директива debug_peer_list из файла main.cf. Глобально включать отладку всех SMTP-соединений было бы очень накладно, поэтому в нее стоит добавлять только список хостов, взаимодействие с которыми мы хотим отслеживать тщательнее, чем обычно. К примеру, для наблюдения за передачей писем между нами и yandex.ru, mail.ru, pochta.ru и 10.10.10.23 строка могла бы выглядеть вот так:

```
debug_peer_level = 2
debug_peer_list =  yandex.ru, mail.ru pochta.ru 10.10.10.23/32
10.10.10.0/24
```

Как видите, хосты можно перечислять двумя путями — через запятую и через пробел. Впрочем, как вы уже убедились, никто не мешает в качестве хоста указывать IP-адреса или целые подсети вроде 10.10.10.0/32. С данной возможностью стоит обращаться очень осторожно, потому что каждая SMTP-сессия, попадающая под наш фильтр, записывает в файл протокола примерно 20 Кб текста. При большом потоке писем между нами и наблюдаемым хостом стоит немного зазеваться, и место на файловой системе /var может закончиться довольно быстро. Директива debug_peer_level указывает, насколько подробными

должны быть отчеты. По умолчанию ее значение равно 2.

Если хочется заглянуть чуть глубже в процесс работы Postfix, можно воспользоваться услугами стандартных трассировщиков системных вызовов. Для разных систем их имена могут несколько отличаться, но, скорее всего, вы легко найдете в своей системе что-то вроде trace, strace, truss или ktrace. Следить за каким-либо процессом довольно просто — нужно всего лишь вызвать команду, актуальную для вашей системы с ключом -p и номером процесса.

Ну а если и это не помогло, можно провести отладку с помощью интерактивного отладчика xgdb или его не интерактивного собрата gdb. За этот функционал отвечает директива debugger_command из файла main.cf. Впрочем, я сильно сомневаюсь, что она пригодится кому-то из вас. По крайней мере мне за последние три года ни разу не пришлось воспользоваться ею, даже под большой нагрузкой, Postfix работал без каких-либо нареканий.

Думаю, на данном этапе можно остановиться. вы уже достаточно хорошо подготовлены для самостоятельного поиска неисправностей в работе Postfix, если таковые встретятся на вашем пути. В следующей статье мы поговорим о методиках анализа узких мест в работе почтовой системы и способах тонкой настройки нацеленной на увеличение производительности.

источник