Шлюз SMTP в Telegram — блог cdnnow!

← Блог

ШЛЮЗ SMTP В TELEGRAM

30 марта 2020

С одной стороны, исторически сложилось, что многие сервисы в Unix/Linux уведомляют администратора о проблемах через электронную почту.

В качестве примеров можно привести cron, smartd и mdadm.

В самописных фоновых или долго выполняющихся сценариях неудачное завершение также не помешает сопровождать отправкой уведомлений, например:

```
Failed() {
        echo "$@" | mail -s "${0##*/} failed on $(hostname -f)" root
        logger -p user.err -t "${0##*/}" -- "$@"
        exit 1
}

first_command || Failed "first_command"

second_command || Failed "second_command"

third_command || Failed "third_command"
```

Разумеется, имея больше одного сервера, такую почту удобнее просматривать не локально, а централизованно.

С другой стороны, надёжная доставка почты в Интернете требует большого количества настроек, без которых письма либо вообще не дойдут до получателя, либо попадут в спам.

И наконец, электронная почта стремительно вытесняется системами мгновенной доставки сообщений: имеющими высокую скорость, простые протоколы взаимодействия и почти повсеместную распространённость.

С учётом изложенного, имеет смысл не собирать служебную почту в почтовый ящик, а настроить шлюз для доставки в Telegram.

Параметры Telegram:

В Телеграме нам потребуется зарегистрировать бота для отправки и получить token, а также узнать числовой идентификатор получателя (Chat ID).

Здесь проще сослаться на существующую документацию, чем пытаться сочинять свою:

- создание бота и получение токена: https://tlgrm.ru/docs/bots#botfather;
- определение Chat ID: найдите в Телеграме бота JsonDump и напишите ему что-нибудь.

Шлюз SMTP в Telegram — блог cdnnow!

Настройка шлюза:

Гугл находит две готовых реализации:

- https://github.com/KostyaEsmukov/smtp_to_telegram
- https://github.com/ircop/smtp2tg

Обе написаны на Golang. Первую из них мы проверили в работе и остались довольны.

Однако было решено написать вместо неё собственную утилиту, чтобы (а) уменьшить занимаемое на диске место в 10000 раз (с 10,7 мегабайт до 1140 байт) и (б) получить потенциальную гибкость для расширения (например, добавить в дальнейшем поддержку нескольких recipient_email_address=>TgChatId).

Забегая вперёд, имеет смысл отметить, что при выполнении smtp_to_telegram занимает в ОЗУ вдвое меньше места, чем наша утилита на Python (13 мегабайт против 26), хотя отчасти это компенсируется наличием других запущенных Python-утилит (в т.ч. tuned, networkd-dispatcher и т.д.), потому что ОЗУ под значительную часть среды выполнения выделяется системой однократно, независимо от числа использующих её процессов.

Пишем собственный вариант:

Этот файл следует сохранить как /usr/local/bin/smtp2tg и сделать исполняемым:

Шлюз SMTP в Telegram — блог cdnnow!

```
import os
import io
import asyncore
import requests               # yum install python-requests
import smtpd
from datetime import datetime

# Optional:
listen_addr = os.environ['SMTP2TG_LISTEN_ADDR'] if 'SMTP2TG_LISTEN_ADDR' in os.environ else
'localhost'
listen_port = os.environ['SMTP2TG_LISTEN_PORT'] if 'SMTP2TG_LISTEN_PORT' in os.environ else 2525
# Required:
bot_token   = os.environ['SMTP2TG_BOT_TOKEN']
chat_id     = os.environ['SMTP2TG_CHAT_ID']

class smtp2tg(smtpd.SMTPServer):
    def process_message(self, peer, mailfrom, rcpttos, data, mail_options=None, rcpt_options=None):
        nowstr   = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        markdown = '**%s:**\n\n%s' % (mailfrom, data.decode("utf-8"))
        msgfmt   = 'https://api.telegram.org/bot%s/sendMessage?chat_id=%s&parse_mode=Markdown&
text=%s'
        response = requests.get(msgfmt % (bot_token, chat_id, markdown))
        print("%s -- from=%s to=%s response=%s\n" %
            (nowstr, mailfrom, rcpttos, response.json()))

server = smtp2tg((listen_addr, int(listen_port)), None)
print("Started on %s:%s..." % (listen_addr, listen_port))

try:
    asyncore.loop()
except KeyboardInterrupt:
    pass
```

Чтобы уменьшить и упростить исходный текст, а также повысить безопасность, необходимые для работы параметры передаются не через командную строку, а через переменные окружения SMTP2TG_xx.

Сценарий проверен на совместимость с Python 2.7 и 3.6. Из дополнительных пакетов требуется только python-requests (или python3-requests), который называется одинаково и Debian/Ubuntu, и в CentOS.

При написании пришлось столкнуться со следующими проблемами:

- В отличие от большинства сетевых сервисов, модуль smtpd не позволяет заканчивать строки в запросах сокращённой односимвольной последовательностью "\n" и понимает только полную двухсимвольную "\r\n". Поэтому для тестирования надо запускать netcat с ключом "-C", а socat с флагом "crlf", и только telnet заработает правильно без дополнительных настроек (но потребует явную задержку, потому что без неё закроет сетевое соединение сразу, как только получит EOF из stdin, и не успеет из-за этого принять из сети ответ):

Шлюз SMTP в Telegram — блог cdnnow!

```
{ echo -e "HELO test\nQUIT"; sleep 1; } | telnet 127.0.0.1 2525
```

- Python не требует явно объявлять тип, но отказывается автоматически преобразовывать его даже в очевидных ситуациях, поэтому для listen_port и data необходимо ручное преобразование (из str в int и из ByteArray в str соответственно).
- Если в системе включён IPv6 и для localhost в /etc/hosts есть строки "127.0.0.1" и "::1", без параметра SMTP2TG_LISTEN_ADDR сценарий слушает только "::1" без 127.0.0.1
- И самое главное — в стандартной документации метод "process_message" содержит неверный набор параметров — без mail_options и rcpt_options (но эту ошибку успели найти до нас).

Автоматический запуск:

Сервис для systemd будет выглядеть так:

```
[Unit]
Description=SMTP to Telegram Gateway
Documentation=https://cdnnow.ru/blog/smtp2tg
After=nss-lookup.target
After=network.target

[Service]
User=smtp2tg
Environment='SMTP2TG_LISTEN_ADDR=0.0.0.0'
Environment='SMTP2TG_BOT_TOKEN=885500333:AAaabbcc_ddeeffgghhSSWW88hhnnzzkkQQ'
Environment='SMTP2TG_CHAT_ID=77113355'
ExecStart=/usr/local/bin/smtp2tg

[Install]
WantedBy=multi-user.target
```

Сохраните его в файл /etc/systemd/system/smtp2tg.service, активируйте и запустите, предварительно создав псевдопользователя:

- `useradd -d /nonexistent -s /bin/false -r smtp2tg`
- `systemctl daemon-reload`
- `systemctl enable smtp2tg`
- `systemctl start smtp2tg`
- `systemctl status smtp2tg`

Настройка отправителей почты:

Нам нужен любой консольный клиент (т.е. предоставляющий команду mail) и любой агент доставки почты, умеющий использовать relay:

```
apt-cache search mailx
apt-cache search mail-transport-agent
```

В Debian/Ubuntu мы предпочитаем для этого следующую связку:

Шлюз SMTP в Telegram — блог cdnnow!

Для CentOS в стандартных репозиториях нет готового пакета dma, но он собран в repo.cdnnow.pro:

```
yum install https://repo.cdnnow.pro/pub/linux/centos/7/x86_64/cdnnow-release-
1-1.cdnnow.el7.noarch.rpm
yum install dma mailx
```

В /etc/dma/dma.conf обязаны присутствовать следующие строки:

```
SMARTHOST 10.20.30.40
PORT 2525
```

IP-адрес должен указывать на сервер, на котором запущен smtp2tg. В данном случае рекомендуется использовать именно IP-адрес, т.к. это позволит сохранить работоспособность при проблемах с DNS.

Проверка:

```
echo This is test 1 | mail -s test1 qwe@asd.org
```

Email получателя может быть любым — шлюз в любом случае отправит сообщение пользователю Telegram, указанному в SMTP2TG_CHAT_ID.

Специалист по CDN онлайн

Шлюз SMTP в Telegram — блог cdnnow!

← Назад в Блог