воскресенье, 28 октября 2012 г.

Программирование работы с SSH при помощи Paramiko (Перевод)

> *OpenSSH - это вездесущий метод удалённого безопасного доступа к машине и передачи файлов. Многие - системные администраторы, инженеры автоматизации тестов, веб-разработчики и другие люди используют этот методы ежедневно. Написание скриптов для ssh на Python может быть тяжёлым занятием, но модуль Paramiko позволяет решить эту задачу проще.*

Это репринт статьи, написанной для Python Magazine в колонку Completely Different и опубликованной в октябрьском выпуске 2008 года. Тут она приводится в оригинальной форме, со всеми ошибками и т.д.

SSH везде. OS X, Linux. Solaris и даже Windows предлагают OpenSSH серверы для удалённого доступа и передачи файлов. Он уже давно заменил такие методы удалённого доступа как telnet и rlogin. И хотя эти системы ещё могут где-то применяться, их широкое использование ушло в прошлое вместе с быстрым принятием набора инструментов OpenSSH.

Сам OpenSSH - это набор инструментов, основанный на протоколе ssh2. Он содержит инструменты для удалённого безопасного входа (ssh), безопасной передачи файлов (scp и sftp) и инструмент управления ключами.

На большинстве ОС клиентские инструменты (ssh, scp, sftp) уже доступны для пользователя. Кроме того, пользователи могут легко установить и сконфигурировать серверные утилиты на той машине, к которой они хотят получить удалённый доступ.

Многие люди ежедневно пользуются OpenSSH и многие тратят уйму времени на попытки написать скрипты для автоматизации его использования. Большая часть этих скриптов оборачивает команды командной строки напрямую (ssh, scp и т.д.). Они используют инструменты типа Pexpect для получения паролей и пытаются работать напрямую с выводом исполняемых файлов.

Потратив на это кучу времени я пришёл, чтобы сказать Вам, что это приводит к ошибкам, сложностям в тестировании и всю эту машинерию очень тяжело поддерживать в рабочем состоянии.

Когда Вы пытаетесь работать с выводом утилит командной строки, следите за кодами завершения, жонглируете таймаутами - Вы на ложном пути. Вот для чего нужно Paramiko.

Я познакомился с Paramiko некоторое время назад. Он использует PyCrypto для предоставления Python'у интерфейса к протоколу ssh2. Этот модуль предоставляет всё, что Вы только можете желать, включая аутентификацию на основе ssh ключей, доступ к ssh-шелу и sftp.

После моего знакомства с Paramiko мой подход к использованию ssh изменился. Вместо изнурительных экспериметов работы с командной строкой, я получаю программный доступ к протоколу и ко всем его инструментам. При чём всё это в Python-стиле.

О Paramiko

Paramiko - это "чистый" модуль Python, который может быть легко установлен, как и любой другой модуль. Однако, PyCrypto написан большей частью на C, так что Вам может потребоваться компилятор, чтобы установить его на свою платформу.

Сам Paramiko имеет обширную документацию по API и активную рассылку. В качестве дополнительного бонуса, есть порт всего этого на Java (но давайте не будем об этом) если Вам понадобится сделать что-то на Java.

Начнем

Главный класс API Paramiko - это "paramiko.SSHClient". Он предоставляет базовый интерфейс, который Вам поднадобится для установки соединения и передачи файлов. Вот простой пример:

View Code PYTHON ?

```
1 import paramiko
2 ssh = paramiko.SSHClient()
3 ssh.connect('127.0.0.1', username='jesse',
4     password='lol')
```

Этот код создаёт новый объект SSHClient и затем вызывает метод connect() для подключения к локальному ssh серверу. Проще и быть не может!

Host keys

Один из сложнейших аспектов ssh-аутентификации - *host keys*. Когда Вы устанавливаете ssh соединения с удалённой машиной, ключ хоста автоматически сохраняется в файле в вашей домашней директории с названием .ssh/konwn_hosts. Если Вы когда-нибудь подключались к новому хосту через ssh, видели сообщение вроде этого:

```
The authenticity of host 'localhost (::1)' can't be
established.
RSA key fingerprint is
22:fb:16:3c:24:7f:60:99:4f:f4:57:d6:d1:09:9e:28.
Are you sure you want to continue connecting
(yes/no)?
```

и печатали в ответ на вопрос “yes” — то Вы добавляли ключ в файл konwn_hosts. Эти ключи важны потому как их принятие реализует уровень доверия между хостами. Если ключ был изменён и скомпроментирован каким-либо образом, то Ваш клиент отвергнет соединение даже не предупредив Вас об этом.

Paramiko использует то же правило. Вы должны принять и авторизовать использование и хранение этих ключей для каждого хоста. К счастью, вместо того, чтобы заниматься каждым ключом по отдельности, Вы можете настроить "волшебную" политику.

Поведение по умолчанию SSHClient'a - отвергать соединение с узлом ("paramiko.RejectPolicy"), чей ключ не сохранён в вашем файле known_hosts. Это может быстро надоесть при работе в тестовом окружении, где машины то появляются, то уходят, и где Вам постоянно приходится переустанавливать системы.

Установка политики работы с ключами требует одного метода, который вызывается у объекта ssh клиента (set_missing_host_key_poliy()), который задаёт поведение, которое должны быть применено к несвязаным ключам. Если Вы, как и я, ленивы, то Вы будете использовать paramiko.AutoAddPolicy(), что приводит к автоматическому принятию неизвестных ключей.

View Code PYTHON ?

```
1 import paramiko
2 ssh = paramiko.SSHClient()
3 ssh.set_missing_host_key_policy(
4     paramiko.AutoAddPolicy())
5 ssh.connect('127.0.0.1', username='jesse',
6     password='lol')
```

Конечно, не используйте эту политику если Вы работаете с машинами, которые Вам не известны или которым Вы не доверяете. Инструменты, построенные на Paramiko должны использовать эту чрезмерно либеральную политику лишь как опцию настройки.

Запускаем простые команды

После того, как мы подключились, мы можем запускать команды и получать их результат.
ssh использует тот же тип ввода / вывода / обработки ошибок, с которым Вы должны быть знакомы по другим UNIX-подобным приложениям. Ошибки посылаются на стандартный вывод ошибок, вывод - на стандартный вывод, а ввод берётся со стандартного ввода.
Итак, ответ от клиента возвращается в виде кортежа - (stdin, stdout, stderr) - где все три элемента - файлоподобные объекты, из которых Вы можете читать (или писать - в stdin). Например:

View Code PYTHON ?

```
1 ...
```

Обо мне

Ishayahu Lastov

Просмотреть профиль

Архив блога

► 2014 (1)
► 2013 (25)
▼ 2012 (104)
► ноября (15)
▼ октября (13)
Быстрая авторизация на pyramid при помощи persona ...
Используем Python для редактирования ярлыков (Пере...
Создать ярлык - the hard way (Перевод)
Создаём ярлыки при помощи Python (Перевод)
stevedore 0.6 (Перевод)
Python сообщество и конференции
Создаём QR коды на Python (Перевод)
Релиз Demakein 0.2: Шалмей (Перевод)
Программирование работы с SSH при помощи Paramiko ...
Тяжёлый свинец (Перевод)
Python 101: Перемещение файлов между серверами (Пе...
wxPython in Action. Глава 12. Манипуляции с изобра...
doctest для одной функции (Перевод)
► сентября (10)
► августа (4)
► июля (23)
► июня (9)
► мая (17)
► апреля (4)
► марта (9)

```
['13:35  up 11 days,  3:13, 4 users, load averages: 0.14 0.18 0.1
```

За кулисами Paramiko открывает новый объект ”paramiko.Channel”, который представляет безопасный тунель к удалённому хосту. Этот объект ведёт себя как обычный объект сокета в Python. Когда мы вызваем ”exec_command()”, открывается канал к хосту, а обратно мы получаем ”paramiko.ChannelFile” файлоподобный объект, который представляет данные, посылаемые на и с удалённого хоста.

Одна из документированных "фитч" объекта ChannelFile состоит в том, что Вам всё время надо читать из stderr и stdout. Если удалённый хост пошлёт обратно достаточно информации, чтобы забить буфер, то хост подвиснет, ожидая, пока ваша программа считает посланные данные. Два способа справиться с этим - вызывать readlines(), как мы сделали выше, или read(). Если Вам нужен внутренний буфер - можно проводить итерацию при помощи readline().

Это была простая форма подключения и запуска команд для получения ответа. Для многих админских задач этого вполне достаточно, так как работая со строками в Python Вы можете достать всё, что Вам нужно. Но давайте теперь посмотрим на что-то с большим объёмом вывода, что ещё и требует пароль:

View Code PYTHON ?

```
ssh.connect('127.0.0.1', username='jesse',
   password='lol')
stdin, stdout, stderr = ssh.exec_command(
   "sudo dmesg")
```

Упс. Я только что запустил команду sudo. А ведь ей нужен пароль для удалённого хоста... Без проблем:

View Code PYTHON ?

```
ssh.connect('127.0.0.1', username='jesse',
    password='lol')
stdin, stdout, stderr = ssh.exec_command(
    "sudo dmesg")
stdin.write('lol\n')
stdin.flush()
data = stdout.read.splitlines()
for line in data:
    if line.split(':')[0] == 'AirPort':
        print line
```

Вот так! Я удалённо залогинился и нашёл все сообщения от своей карты Airport. Ключом к этому послужила запись пароля в "файл" stdin, так что получив пароль, sudo меня распознал.

Если у Вас уже крутится на языке вопрос - да, таким образом Вы можете создать свою собственную интерактивную оболочку. Вы можете захотеть сделать что-то вроде этого для создания админской оболочки при помощи модуля cmd, чтобы управлять машинами в своей лаборатории.

Используя Paramiko сделать это становится просто. В листинге 1.1 показан простой способ достичь этого - мы оборачиваем манипуляции paramiko в методы RunCommand, позволяя пользователю добавлять столько хостов, сколько он захочет, соединяясь с ними и выполняя на них команды.

Листинг 1.1

View Code PYTHON ?

```
#!/usr/bin/python

import paramiko
import cmd

class RunCommand(cmd.Cmd):
    """ Simple shell to run a command on the host """

    prompt = 'ssh > '

    def __init__(self):
        cmd.Cmd.__init__(self)
        self.hosts = []
        self.connections = []

```

```
23
24      def do_connect(self, args):
25          """Connect to all hosts in the hosts list"""
26          for host in self.hosts:
27              client = paramiko.SSHClient()
28              client.set_missing_host_key_policy(
29                  paramiko.AutoAddPolicy())
30              client.connect(host[0],
31                  username=host[1],
32                  password=host[2])
33              self.connections.append(client)
34
35      def do_run(self, command):
36          """run <command>
37          Execute this command on all hosts in the list"""
38          if command:
39              for host, conn in zip(self.hosts, self.connections):
40                  stdin, stdout, stderr = conn.exec_command(comman
41                  stdin.close()
42                  for line in stdout.read().splitlines():
43                      print 'host: %s: %s' % (host[0], line)
44          else:
45              print "usage: run <command>"
46
47      def do_close(self, args):
48          for conn in self.connections:
49              conn.close()
50
51  if __name__ == '__main__':
52      RunCommand().cmdloop()
```

Вот пример вывода:

```
ssh > add_host 127.0.0.1,jesse,lol
ssh > connect
ssh > run uptime
host: 127.0.0.1: 14:49  up 11 days,  4:27, 8 users,
load averages: 0.36 0.25 0.19
ssh > close
```

Это всего лишь proof-of concept псевдо-интерактивной оболочки. Вот что Вы ещё можете к ней добавить:

- улучшенное отображение многострочного вывода
- обработчик стандартных ошибок
- что-нибудь в метод выхода
- поток для выполнения команды / возврата результата

Как и когда дело касается шелла границей тут будет только ваша фантазия. Такие инструменты, как pssh, osh, Fabric и т.д. все отображают данные по разному и каждый из них по своему группирует вывод от разных хостов.

Отправка и получение файлов

Работа с файлами в Paramiko управляется реализацией sftp, и, как и работа с клиентом ssh, она проста как два пальца.

Мы начинаем с создания нового paramiko.SSHClient, как и раньше:

View Code PYTHON ?

```
1  import paramiko
2  ssh = paramiko.SSHClient()
3  ssh.set_missing_host_key_policy(
4      paramiko.AutoAddPolicy())
5  ssh.connect('127.0.0.1', username='jesse',
6      password='lol')
```

Но на этот раз мы вызовем метод ”open_sftp()” после установки соединения. ”open_sftp()” возвращает объект клиента ”paramiko.SFTPClient”, который поддерживает все стандартный операции sftp (stat, put, get и т.д.). В нашем примере мы воспользуемся операцией “get”, чтобы скачать файл ”remotefile.py” с удалённой машины и сохранить его на локальной машине под именем ”localfile.py”.

Отправка файла на удалённую машину (операция “put”) работает точно так же. Мы лишь меняем порядок аргументов:

View Code PYTHON ?

```
1 ftp = ssh.open_sftp()
2 ftp.get('localfile.py', 'remotefile.py')
3 ftp.close()
```

То, что мне нравится в sftp клиенте, реализованном в Paramiko, так это то, что он поддерживает такие операции как stat, chmod, chown и т.д. Очевидно, что они могут работать по разному на разных удалённых машинах, так как не все сервера реализуют протокол целиком, но всё равно их наличие - большой плюс.

Вы можете запросто написать функцию типа glob.glob() для просмотра файлового дерева удалённой системы в поисках файлов, отвечающих заданному шаблону. Можно проводить поиск на основе разрешений, размера файлов и т.д.

Однако, стоит заметить (и я часто с этим сталкивался): sftp, как протокол, гораздо более ограничен, чем scp. scp позволяет использовать шаблоны UNIX в имени файла при получении его с удалённой машины. А вот sftp ожидает от Вас полного пути к файлу, который Вы хотите скачать. Вот пример:

View Code PYTHON ?

```
1 ftp.get('*.py', '.')
```

В большинстве случаев это значит "скачать все файлы с расширением .py в локальную директорию на моей машине". Но, к сожалению, sftp Вас не поймёт (см листинг 2). Мне это знание досталось тяжёлым путём, после того, как я потратил несколько часов на разбор реализации клиента sftp.

Листинг 2:

View Code PYTHON ?

```
1  >>> ftp.get("./*.py", '.')
2  Traceback (most recent call last):
3    File "<stdin>", line 1, in <module>
4    File "/Library/Python/2.5/site-packages/paramiko/sftp_client.p
5      line 567, in get
6      fr = self.file(remotepath, 'rb')
7    File "/Library/Python/2.5/site-packages/paramiko/sftp_client.p
8      line 238, in open
9      t, msg = self._request(CMD_OPEN, filename, imode, attrblock)
10   File "/Library/Python/2.5/site-packages/paramiko/sftp_client.p
11     line 589, in _request
12     return self._read_response(num)
13   File "/Library/Python/2.5/site-packages/paramiko/sftp_client.p
14     line 636, in _read_response
15     self._convert_status(msg)
16   File "/Library/Python/2.5/site-packages/paramiko/sftp_client.p
17     line 662, in _convert_status
18     raise IOError(errno.ENOENT, text)
19 IOError: [Errno 2] No such file
```

В заключение

Я надеюсь, что Вы увидели достаточно, чтобы оценить Paramiko. Это одна из тех жемчужин в сообществе Python, которая помогает мне в ежедневной работе. Я могу программировать удалённое администрирование, писать тестовые плагины, которые легко исполняют удалённые операции, и всё это без необходимости устанавливать дополнительных демонов на удалённой машине.
SSH везде, и раньше или позже Вы тоже столкнётесь необходимостью писать под него программы. Так почему бы сразу не сэкономить себе время и не воспользоваться Paramiko?

Связанные ссылки

OpenSSH — http://www.openssh.com/
Pexpect — http://www.noah.org/wiki/Pexpect
Fabric — http://www.nongnu.org/fab/
Paramiko Docs — http://www.lag.net/paramiko/docs/
Paramiko Mailing List — http://www.lag.net/mailman/listinfo/paramiko

- Fabric 0.0.3: A Capistrano-like deployment tool.
- osh: Object-Oriented Shell
- Favorite Posts
- AlmostVPN — Trac

Источник

Автор: Ishayahu Lastov на 21:32

Ярлыки: библиотеки, OpenSSH, paramiko, ssh

7 комментариев:

Unknown 11 июня 2015 г. в 16:27

Cool!)

Ответить

playnet 2 сентября 2015 г. в 13:13

>Но, к сожалению, sftp Bac не поймёт (см листинг 2).
и как быть?

Ответить

playnet 2 сентября 2015 г. в 14:26

в коннекте
if len(host) == 3:
client.connect(host[0],
username=host[1],
password=host[2])
else:
client.connect(host[0],
username=host[1])

чтобы можно было по ключу авторизоваться. А вообще можно было на гитхаб выложить, и пулреквесты принимать)

Ответить

Ответы

Ishayahu Lastov 3 сентября 2015 г. в 12:21

Я этим в итоге не воспользовался, так что для меня это не более чем перевод)

Ответить

Анонимный 14 декабря 2016 г. в 15:03

А на сайте: www.superplayers1.ru можно посмотреть курс по созданию карточной игры

Ответить

Unknown 19 апреля 2017 г. в 18:56

Статья хорошая, но обратите внимание на то, что
если вы хотите положить файл из папки бла/бла/файл.какойто нужно указать не просто куда его положить бла2/бла2/, а повторить название файла бла2/бла2/файл.какойто

Я убил на это день

http://stackoverflow.com/questions/3091326/put-in-sftp-using-paramiko

Ответить

Unknown 25 июля 2017 г. в 19:47

ВОЙТИ С АККАУНТОМ GOOGLE

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Yandex.money

Не удаётся установить соединение с сайтом

Не удалось найти IP-адрес сервера **money.yandex.ru**.

Попробуйте сделать следующее:

- Проверить, есть ли подключение к интернету.

GAnalytic