Multi-threaded Network Automation With Python

Brandon Hale
brandon@brandonhale.us

Goals

- Automate commands on Cisco IOS and similar devices
- Establish repeatable patterns for sending and receiving text from a router
- Attempt login with both Telnet and SSH
- Use threads to execute against several devices at once

Apologia

- This could be prettier
  - I won't be getting any calls from 37signals
- Working demo code should be posted at CPOSC after a good scrubbing
- Database schema is much too complicated for the purposes of this discussion – a suggested schema will be shared later
- CSV examples out of scope

Expect

- Expect was originally developed for TCL by Don Libes
- Expect is a set of commands to send and receive text from another process
- Combining Expect with Telnet or SSH makes managing your network fun again
- Expect rules – TCL sucks
  - No multi-dimensional arrays. Gar!

PExpect

- A deceptively clever name for an Expect work-alike written in pure Python
- Python doesn't suck
  - Hopefully all the Ruby guys went to another talk
- http://www.noah.org/wiki/Pexpect
  - Available in Debian/Ubuntu

Life of a PExpect App

- Import some modules
- Collect a list of devices to work on (SQL, CSV)
- Spawn a process (Telnet or SSH)
- Control the spawned process by sending and receiving data
- Repeat until device list is exhausted

Imports

```
#!/usr/bin/env python

import sys
import MySQLdb
import pexpect
import pxssh
import string
import re
import threading
import time
```

Spawning a Process

```
def ssh_connect(ip_address):
    session = pexpect.spawn ('ssh -l ' + username + ' ' + ip_address, timeout=5,
searchwindowsize=50)
   # Enable logging to the console
   session.logfile = sys.stdout
```

“Expecting” Data

```
def ssh_connect(ip_address):
    session = pexpect.spawn ('ssh -l ' + username + ' ' + ip_address, timeout=5,
searchwindowsize=50)
    session.logfile = sys.stdout
    i = session.expect (['assword', 'continue connecting', pexpect.TIMEOUT, pexpect.EOF,
'refused'])
    #Connected
    if i == 0:
        session.timeout = 10
        print "got ssh connected"
        return session
    # Host key prompt, accept unknown key
    elif i == 1:
        print "got host key prompt"
        session.sendline ('yes')
        return session
    # Timeout, EOF, or refused
    elif i == 2 or i == 3 or i == 4:
        print "could not ssh connect"
        return 0
```

Sending Data

```
def send_login(session):
    for password in device_info['passwords']:
        session.sendline(password)
        i = session.expect(['Authentication failed', '.+>', '.+#', 'Login invalid', 'sername', 'assword', 
pexpect.TIMEOUT])
        if i == 0 or i == 3 or i == 4 or i == 5:
            print "authentication failed"
            return 0
        if i == 1:
            print "got logged in"
            return session
        if i == 2:
            print "got enabled immediately"
            device_info['is_enabled'] = 1
            return session
        if i == 6:
            print "timeout"
            return 0
```

Do The Deed

```
# Poorly named, you can deal
def get_config(session):
    if device_info['os'] == "IOS":
        session.sendline ('term len 0')
        i = session.expect ([device_info['device_prompt'],
pexpect.TIMEOUT])
        if i == 0:
                session.sendline ('conf t')
                session.sendline ('snmp-server host 204.194.130.62 public')
                session.sendline ('snmp-server host 204.194.130.63 public')
                session.sendline ('snmp-server contact DENOC')
                session.sendline ('exit')
        elif i == 1:
            print "unknown timeout"
            logFile.write('unknown timeout\n')
            return 0
    return session
```

Get Classy

```
# We need a class inheriting from threading.Thread to get the job done.

class GetConfig(threading.Thread):

    def __init__ (self,device_id,name,ipaddress,site_id):
        threading.Thread.__init__(self)
        self.device_id = device_id
        self.name = name
        self.ipAddress = ipaddress
        self.site_id = site_id
        self.status = -1

......
```

GetConfig(threading.Thread).run()

```
class GetConfig(threading.Thread):
    def __init__ (self,device_id,name,ipaddress,site_id):
....
    def run(self):
        global device_info
        global passwords
        global enable_passwords

        passwords = ['kjlkjlkjl']
        enable_passwords = ['fsdfsdfsdfs', 'lkklkjl']

        device_info = {'name': self.name, 'os': '', 'device_prompt': '', 'is_enabled': 0, 'passwords':
passwords, 'enable_passwords': enable_passwords}
        # connect
        db = MySQLdb.connect(host="localhost", user="mysqluser", passwd="mysqlpass",
db="mysqldb")
        # create a cursor
        self.cursor = db.cursor()
        print self.getName() + " " + self.ipAddress
        print "trying " + self.ipAddress
        session = ssh_connect(self.ipAddress)
```

run() continued

```
if not session:
        print "couldn't ssh connect"
        session = telnet_connect(self.ipAddress)
        if session == 'no tacacs':
            print "no tacacs"
            query = "replace into configs (device_id,no_tacacs) values (%s,1)" % (str(self.device_id))
            print "query: ",  query
            #self.cursor.execute(query)
            self.cursor.close()
            db.close()
            sys.exit(1)
        elif not session:
            print "couldn't telnet connect"
            query = "replace into configs (device_id,connect_timeout) values (%s,1)" % (str(self.device_id))
            print "query: ",  query
            self.cursor.close()
            db.close()
            sys.exit(1)
        else:
            print "got telnet connected"
     else:
        print "got ssh connected"
```

run('would someone refactor this thing?')

```
#session.logfile = sys.stdout
session = send_login(session)
if device_info['is_enabled'] == 1:
    print "got enabled immediately"
elif session:
    print "got logged in"
    session = send_enable(session)
    if session:
        print "got enabled"
        print "os: " + device_info['os']
        print "prompt: " + device_info['device_prompt']
        session = get_config(session)
        if session:
            print "got config"
        else:
            print "didn't get config"
    else:
        print "couldn't enable"
        query = "replace into configs (device_id,enable_failure) values (%s,1)" % (str(self.device_id))
        print "query: ",  query
        #self.cursor.execute(query)
        self.cursor.close()
        db.close()
        sys.exit(1)
```

run() - last one, srsly

```
else:
        print "couldn't log in"
        query = "replace into configs (device_id,authentication_failure) values (%s,1)" % (str(self.device_id))
        print "query: ",  query
        #self.cursor.execute(query)
        self.cursor.close()
        db.close()
        sys.exit(1)

    query = "replace into configs (device_id, last_config) values (%s,NOW())" % (str(self.device_id))
    print "query: ",  query
    #self.cursor.execute(query)

    self.cursor.close()
    db.close()
```

Pull it all together – with threads!

```
db = MySQLdb.connect(host="localhost", user="mysqluser", passwd="mysqlpass", db="mysqldb")
cursor = db.cursor()
# See what I mean about schema?
cursor.execute("SELECT device_id, name, ip_address, site_id FROM devices where node_id IN
(1) AND (snmp_string = 'Pie2ahChoh' OR snmp_string = 'cujan1') AND (site_id = 107) AND
device_id > 7857 ORDER BY device_id ")

# get the resultset as a tuple
result = cursor.fetchall()
logFile=open('config-fetch-log', 'w')

totalThreads = 10
# OH LOL - RESULTING TUPLE IZ IMMUTABLE PLZ TO MAKE POP() WORK
newresult = list(result)

while len(newresult):
    if threading.activeCount() <= totalThreads:
        record = newresult.pop()
        print record[0] , record[1], record[2]
        print "active: " + str(threading.activeCount())
        current = GetConfig(record[0],record[1],record[2],record[3])
        current.start()
    else:
        time.sleep(2)
```

Wrap Up

- Ramble about other cool uses for Python
  - Config backup
  - Audit configurations
  - Import/Export arbitrary CSV from MySQL
- Please, someone ask some questions