Paramiko

A module for Python 3.4+ that implements the SSH2 protocol for secure (encrypted and authenticated) connections to remote machines.

https://en.wikipedia.org/wiki/Secure_Shell#Version_2.x

Written entirely in Python (though it depends on third-party C wrappers for low level crypto; these are often available pre-compiled).

Hamilton Python Users Group
Ian Stewart
13 July 2020
https://github.com/HamPUG/meetings/tree/master/2020/2020-07-13/paramiko

Paramiko: apt

```
$ sudo apt install python3-paramiko
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python3-bcrypt
Suggested packages:
  python3-gssapi
The following NEW packages will be installed:
  python3-bcrypt python3-paramiko
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 152 kB of archives.
After this operation, 813 kB of additional disk space will be used.
Do you want to continue? [Y/n] y


python3-paramiko/focal,focal,now 2.6.0-2 all [installed,automatic]
  Make ssh v2 connections (Python 3)
```

Paramiko: apt

```
$ sudo apt install python3-paramiko
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python3-bcrypt
Suggested packages:
  python3-gssapi
The following NEW packages will be installed:
  python3-bcrypt python3-paramiko
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 152 kB of archives.
After this operation, 813 kB of additional disk space will be used.
Do you want to continue? [Y/n] y


python3-paramiko/focal,focal,now 2.6.0-2 all [installed,automatic]
  Make ssh v2 connections (Python 3)
```

Paramiko: PyPI

```
$ pip3 search paramiko
paramiko (2.7.1)                - SSH2 protocol library
  INSTALLED: 2.6.0  <-- Ubuntu Mate 20.04
  LATEST:    2.7.1
```

paramiko 2.7.1

Latest version

```
pip install paramiko
```

Released: Dec 10, 2019

SSH2 protocol library

Navigation

- Project description
- Release history
- Download files

Project description

This is a library for making SSH2 connections (client or server). Emphasis is on using SSH2 as an alternative to SSL for making secure connections between python scripts. All major ciphers and hash methods are supported. SFTP client and server mode are both supported too.

To install the development version, `pip install -e git+https://github.com/paramiko/paramiko`

Paramiko: Home page ~ Github 224 Contributors

https://github.com/paramiko/paramiko/

Paramiko

build passing | codecov 79%

Paramiko:	Python SSH module
Copyright:	Copyright (c) 2009 Robey Pointer <robeypointer@gmail.com>
Copyright:	Copyright (c) 2020 Jeff Forcier <jeff@bitprophet.org>
License:	LGPL
Homepage:	http://www.paramiko.org/
API docs:	http://docs.paramiko.org
Development:	https://github.com/paramiko/paramiko

Paramiko: Home page ~ Description

https://github.com/paramiko/paramiko/

What

"Paramiko" is a combination of the Esperanto words for "paranoid" and "friend". It's a module for Python 2.7/3.4+ that implements the SSH2 protocol for secure (encrypted and authenticated) connections to remote machines. Unlike SSL (aka TLS), SSH2 protocol does not require hierarchical certificates signed by a powerful central authority. You may know SSH2 as the protocol that replaced Telnet and rsh for secure access to remote shells, but the protocol also includes the ability to open arbitrary channels to remote services across the encrypted tunnel (this is how SFTP works, for example).

It is written entirely in Python (though it depends on third-party C wrappers for low level crypto; these are often available precompiled) and is released under the GNU Lesser General Public License (LGPL).

The package and its API is fairly well documented in the `docs` folder that should have come with this repository.

Paramiko: Documentation

docs.paramiko.org/en/stable/

Paramiko

A Python implementation of SSHv2.

Watch 323

build passing

Navigation

Welcome to Paramiko's documentation!

This site covers Paramiko's usage & API documentation. For basic info on what Paramiko is, including its public changelog & how the project is maintained, please see the main project website.

API documentation

The high-level client API starts with creation of an `SSHClient` object. For more direct control, pass a socket (or socket-like object) to a `Transport`, and use `start_server` or `start_client` to negotiate with the remote host as either a server or client.

As a client, you are responsible for authenticating using a password or private key, and checking the server's host key. (Key signature and verification

Paramiko: Try Program... No ~/.ssh/known_hosts

```
$ python paramiko_demo.py
Traceback (most recent call last):
  File "paramiko_demo.py", line 266, in button_clicked
    copy_from_remote_server(args.server, args.port, args.username,
  File "paramiko_demo.py", line 313, in copy_from_remote_server
    client.load_host_keys(os.path.expanduser(os.path.join("~", ".ssh",
"known_hosts")))
  File "/usr/lib/python3/dist-packages/paramiko/client.py", line 127, in
load_host_keys
    self._host_keys.load(filename)
  File "/usr/lib/python3/dist-packages/paramiko/hostkeys.py", line 95, in
load
    with open(filename, "r") as f:
FileNotFoundError: [Errno 2] No such file or directory:
'/home/ian/.ssh/known_hosts'

$ ls ~/.ssh
ls: cannot access '/home/ian/.ssh': No such file or directory
```

Paramiko: ssh login creates ~/.ssh/known_hosts

```
$ ssh ian@1.2.3.4 -p 22
The authenticity of host '[1.2.3.4]:22 ([1.2.3.4]:22)' can't be
established.
RSA key fingerprint is SHA256:x/ZSGm55Az1MyZ47mihXjU91HwzhVVHCXDzKk/AnjqM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '[1.2.3.4]:22' (RSA) to the list of known
hosts.
ian@1.2.3.4's password:
Linux firewall 4.19.0-9-amd64 #1 SMP Debian 4.19.118-2+deb10u1 (2020-06-
07) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jul  6 16:34:26 2020 from 60.234.107.116
```

Paramiko: Exists ~/.ssh/known_hosts

```
$ ls -l ~/.ssh
total 4
-rw-r--r-- 1 ian ian 442 Jul 13 11:39 known_hosts

$ cat ~/.ssh/known_hosts
|1|tD1IlasdlCcufgRS6Lasdasu7aLA=|JTguLYGfExJpZ1KQtnxqU1bTQis= ssh-rsa
AAAAB3NzaC1yc2EAAAADAQAsdfsdfBAQC0DfXzMM+ylKEe4ePIIW7d5e6LbJ/dyDh1x/
m46uS2M+KlIiaeIar90Au/NFc86ng4mayg9YKmfekvBQ/
5G874qYxymIy4SIKqeo3PYMkVbkD9SwxGb3dMdmu/
TyfJBUiWYGSqTG7h+lH10UENY0SmEmSixg/EjGPkHx4fW3ukuadG5law5/
LiXchjwt2P0LDXp7EVuRgyM5WAtR1oRYtFgcQZmKUAfVtmlAGQ387dJHV7O8rWEfrN74IyQ/
mraK7IREiA6kPQLg+2koYcY68CgE5R5SjRmXmS4gB9FV9FcIcLNvES9Fvz+AJYvDvB
```

Paramiko: Code Examples - .get()

```
def copy_from_remote_server(server, port, username,
                            password, remote, local):

    client = paramiko.SSHClient()
    client.load_host_keys(os.path.expanduser(os.path.join(
                           "~", ".ssh", "known_hosts")))

    client.connect(server, port, username, password)

    sftp = client.open_sftp()
    sftp.get(remote, local)
    sftp.close()
    client.close()
```

Paramiko: Code Examples - .put()

```
def copy_to_remote_server(server, port, username,
                          password, local, remote):

    client = paramiko.SSHClient()
    client.load_host_keys(os.path.expanduser(os.path.join(
                          "~", ".ssh", "known_hosts")))

    client.connect(server, port, username, password)


    sftp = client.open_sftp()
    sftp.put(local, remote)
    sftp.close()
    client.close()
```

Paramiko: Code Examples - .exec_command()

```
def execute_command_on_remote_server(server, port, username,
                                     password, command):

    client = paramiko.SSHClient()
    client.load_host_keys(os.path.expanduser(os.path.join(
                          "~", ".ssh", "known_hosts")))

    client.connect(server, port, username, password)

    stdin, stdout, stderr = client.exec_command(command)

    # Type is <class 'paramiko.channel.ChannelFile'> read each line???
    string = ""
    for line in stdout:
        string += line

    client.close()
    return string
```

Paramiko: Code Examples – Security?

Passing...

- Server ip address or domain name.
- Port
- Username
- Password

Paramiko: Code Examples – Security: Embeded

```
SERVER = "1.2.3.4"
PORT = 22
USERNAME = "admin"
PASSWORD = "my_pass"
```

Paramiko: Code Examples – Security: Base64 file

```
def create_b64_message():
    """
    Convert the confidential remote server data to be a b64 string
    """
    message = input("Enter parameters string: ")

"SERVER='1.2.3.4', PORT=22, USERNAME='admin', PASSWORD='my_pass'"

    message_bytes = message.encode('utf-8')
    base64_bytes = base64.b64encode(message_bytes)
    #print("Encoded message in bytes:{}".format(base64_bytes))
    message_b64_ascii = base64_bytes.decode('utf-8')
    print("\nEncoded message in utf-8:{}".format(message_b64_ascii))

    # Check decoding:
    base64_bytes = message_b64_ascii.encode('utf-8')
    message_bytes = base64.b64decode(base64_bytes)
    message = message_bytes.decode('utf-8')
    print("Original message decoded:{}".format(message))
```

Paramiko: Code Examples – Security: Base64 file?

```
...continued...

    # Write b64 data to file
    with open("b64.data", "w") as fout:
        fout.write(message_b64_ascii)

    # Open file and check it decodes OK:
    with open("b64.data", "r") as fin:
        message_b64_ascii = fin.read()
        base64_bytes = message_b64_ascii.encode('utf-8')
        message_bytes = base64.b64decode(base64_bytes)
        message = message_bytes.decode('ascii')
        print("Original message from file decoded:{}".format(message))
```

```
$ cat b64.data
U0VSVkVMjA1LjEwMCcsIFBPUlQ9MjAU9J2lasdycycsIFBBU1NXT1JEPSdkasdZWN0GFuZCc=
```

Paramiko: Code Examples – Security: Base64 file

```
if SERVER == "" or USERNAME == "" or PASSWORD == "":

    try:
        with open("b64.data", "r") as fin:
            message_b64_ascii = fin.read()
            base64_bytes = message_b64_ascii.encode('utf-8')
            message_bytes = base64.b64decode(base64_bytes)
            message = message_bytes.decode('ascii')
            #print(message)
            constant_list = message.split(",")
            #print(data_list)
            for constant in constant_list:
                constant = constant.strip()
                exec(constant)
```

Paramiko: Code Examples – Create Base64 file

Launch paramiko demo with the following argument:

```
$ python paramiko_demo.py --make-b64
```

You will be prompted to enter a string, like this:

```
"SERVER='1.2.3.4', PORT=22, USERNAME='admin', PASSWORD='my_pass'"
```

The program will exit. On restarting then the b64.data file will be used

Paramiko: Code Examples – Security: Command Line

```
parser = argparse.ArgumentParser()


parser.add_argument("-s", "--server",
                    type = str,
                    default = SERVER,
                    help = "Server domain name or IP address string.")

parser.add_argument("-p", "--port",
                    type = int,
                    default = PORT,
                    help = "Port number for ssh")

Etc...

$ python paramiko_demo.py -s 1.2.3.4 -p 22 -u admin -p my_pass
```

Paramiko: Code Examples – Security: Prompted

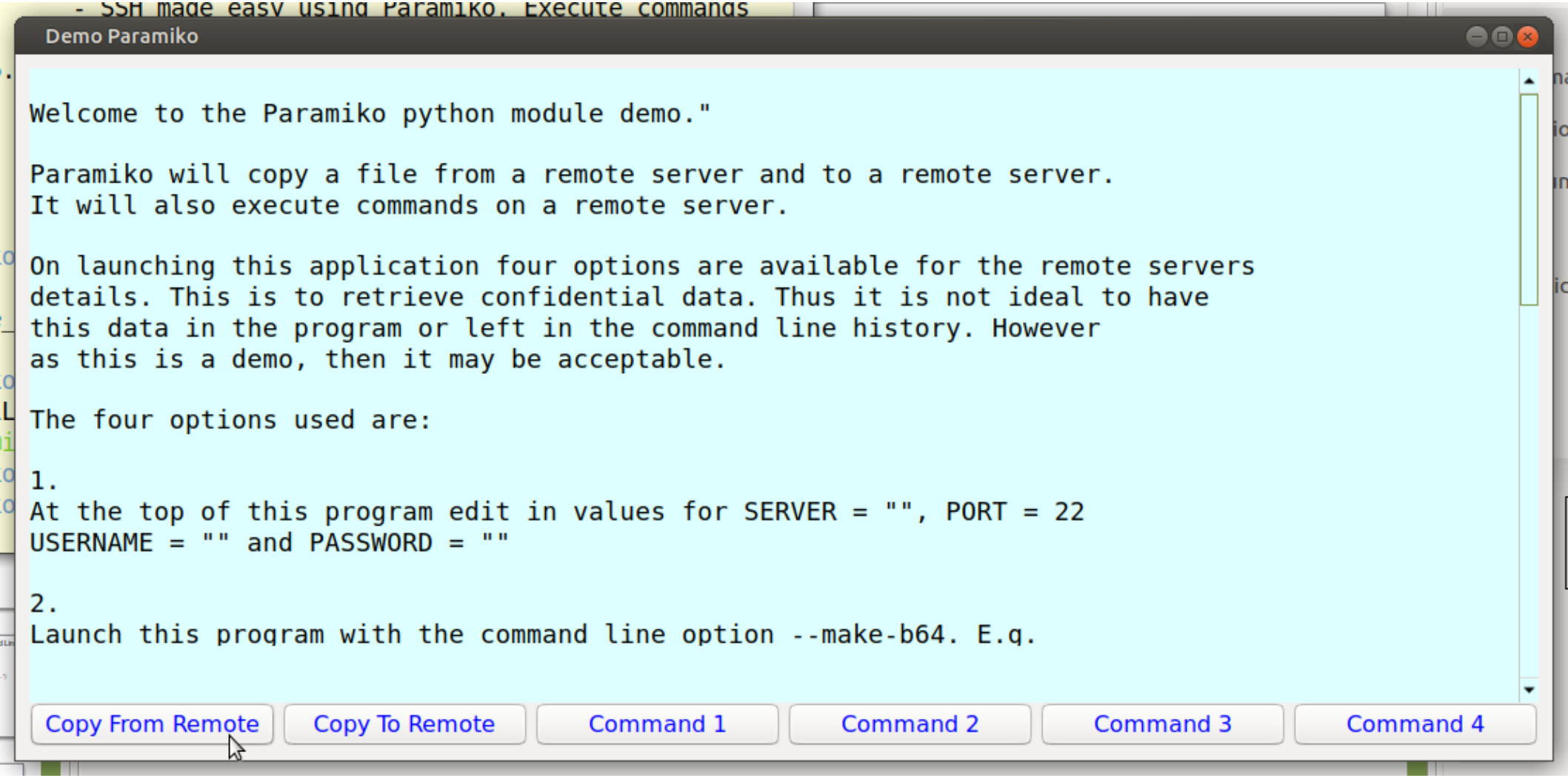
```
if args.server == "":
    args.server = input("\nEnter the name or ip address of the server: ")
1.2.3.4

if args.port == 22:
    args.port = int(input("Enter the ssh port number: "))
22

if args.username == "":
    args.username = input("Enter the Account name on the remote server: ")
admin

if args.password == "":
    args.password = input("Enter the password for the account on the
remote server: ")
my_pass
```

Paramiko: Demo with GTK GUI



Paramiko: Questions?

Question:
Using known_hosts, but need password.

Can I use authorized_key?...

i.e. On the remote server put the public key into the ~/.ssh/authorized_keys file.

===

Answer: Refer to...

http://docs.paramiko.org/en/stable/api/client.html#paramiko.client.SSHClient.connect

Use a private key ("pkey") in the connect method.

TODO: Try out this connection method.