From Laptop to Router: A Practical Project on Network Sharing and DNS Management

Nihal Pirjade
Email: nihalpirjade.clg@gmail.com
LinkedIn: linkedin.com/in/nihal-pirjade
GitHub: github.com/DEXTERxFTW

Siddhant Kadam
Email: sidddhantkadam@gmail.com
LinkedIn:
linkedin.com/in/siddhant-kadam-895b2b237
GitHub: github.com/smartytinker

Abstract

This document describes the process of repurposing an old laptop into a functional Linux-based Wi-Fi router. The built-in wireless interface on the laptop is assumed to be non-functional, motivating the use of an external USB Wi-Fi adapter. We detail the system architecture, necessary software tools, and step-by-step implementation including installing drivers, configuring a Wi-Fi access point with `hostapd`, a DHCP/DNS server with `dnsmasq`, and firewall/NAT rules with `iptables`. Possible future enhancements such as VPN support and IDS integration are also discussed.

1 Introduction

The goal of this project is to transform an outdated laptop into a dedicated wireless router and network gateway. In this case, the laptop's built-in Wi-Fi hardware is broken or unreliable, hence it cannot serve as an access point by itself. Instead, we connect a USB Wi-Fi adapter (`Realtek RTL8188FTV`) to provide wireless connectivity. The laptop runs a minimal Ubuntu Server (CLI-only) to maximize stability and performance. By configuring the laptop to forward packets between its Ethernet and wireless interfaces, the system can share an existing Internet connection.

2 Objectives

The main objectives of this project are:

- Convert an old laptop into a standalone Wi-Fi access point and router.
- Provide DHCP and DNS services for connected clients.
- Implement firewall and NAT rules to securely route traffic from Wi-Fi to the Internet.
- Document the step-by-step installation process and troubleshooting methods.

- Ensure the solution works in a headless (CLI-only) Ubuntu Server environment.
- To integrate **Unbound** as a recursive resolver with **DNS-over-TLS** support, thereby ensuring that all DNS queries are encrypted and protected from eavesdropping.

3 Tools and Technologies Used

The project uses the following software components and tools:

- **Operating System:** Ubuntu Server (CLI-only 22.04 LTS).
- **Wi-Fi driver:** Open-source Realtek driver (`rtl8188fu`) from GitHub, installed via `dkms`.
- **Hostapd:** Configures the laptop's Wi-Fi adapter as an access point.
- **Dnsmasq:** Lightweight DNS forwarder and DHCP server for assigning IPs and handling DNS queries.
- **Iptables:** Linux firewall utility for configuring NAT and packet filtering rules.
- **Sysctl:** Used to enable IP forwarding in the kernel (`net.ipv4.ip_forward`).
- **SSH:** For remote headless management of the server.
- **Unbound** – A validating, recursive, and caching DNS resolver. It improves privacy and security by resolving DNS queries locally.
- **DNS over TLS (DoT)** – Encryption methods integrated with Unbound to secure DNS queries. This prevents third parties from intercepting or modifying DNS traffic.

4 Hardware Setup

The hardware setup consists of:

- **Laptop:** An older model laptop with a working Ethernet port. No functional built-in Wi-Fi is required.
- **USB Wi-Fi Adapter:** A Realtek RTL8188FTV wireless USB adapter (802.11n) is connected to one of the laptop's USB ports and serves as the wireless interface (`wlan0`).
- **Ethernet Connection:** The laptop's Ethernet port (`eth0`) is connected to an upstream network (e.g., a modem or another router providing Internet access).

The Realtek RTL8188FTV adapter requires a Linux driver (`rtl8188fu`) that may not be included by default. We obtain and compile this driver using DKMS. Once installed, the adapter is recognized by the system as `wlan0`.

5 System Architecture

The system functions as a router by forwarding network traffic between two interfaces:

- `eth0` (Ethernet): Connected to the Internet source or upstream network.
- `wlan0` (Wireless): Provides a Wi-Fi access point for client devices.

The laptop is configured with NAT (Network Address Translation) so that multiple Wi-Fi clients can share the single Ethernet-based Internet connection. Key architectural components include:

- Kernel IP forwarding is enabled to route packets between interfaces.
- `hostapd` runs on the laptop to broadcast a Wi-Fi SSID and handle wireless authentication (WPA2).
- `dnsmasq` provides DHCP service to assign IP addresses (e.g., in a subnet like 192.168.10.x) to wireless clients, and forwards DNS queries to upstream DNS servers.
- `iptables` rules are applied to NAT and filter traffic, such as masquerading outgoing packets on `eth0` and allowing established connections to return.

Clients connecting via Wi-Fi receive an IP from the laptop (via `dnsmasq`) and use it as their gateway. All their outbound traffic is NAT-ed through the laptop to the upstream network.

6 Implementation

The following implementation steps outline the configuration of Ubuntu Server and necessary services:

6.1 Installing the Wireless Driver

Because the Realtek RTL8188FTV adapter may not be natively supported, we install the `rtl8188fu` driver manually. With a temporary Internet connection (e.g., via Ethernet or USB tethering), run:

```
sudo apt update
sudo apt install --reinstall build-essential dkms git
git clone https://github.com/kelebek333/rtl8188fu
sudo dkms add ./rtl8188fu
sudo dkms build rtl8188fu/1.0
sudo dkms install rtl8188fu/1.0
sudo cp ./rtl8188fu/firmware/rtl8188fufw.bin /usr/lib/firmware/rtlwifi/
```

Then blacklist any default driver that conflicts:

```
echo "blacklist r8188eu" | sudo tee -a /etc/modprobe.d/blacklist.conf
echo "rtl8188fu" | sudo tee -a /etc/modules
```

Reboot the system. The USB adapter should now be available as `wlan0`. Verify with `iwconfig` or `ip a`.

6.2 Enabling IP Forwarding

Edit `/etc/sysctl.conf` and ensure the following line is present (or uncommented):

```
net.ipv4.ip_forward=1
```

Apply this setting immediately with:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

6.3 Configuring `hostapd`

Create or edit `/etc/hostapd/hostapd.conf`. Example contents:

```
interface=wlan0
ssid=LinuxRouter
hw_mode=g
channel=6
wpa=2
wpa_passphrase=*********
```

Ensure `hostapd` uses this config by setting `DAEMON_CONF= "/etc/hostapd/hostapd.conf"` in `/etc/default/hostapd`. Enable and start `hostapd`:

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd
```

This will start broadcasting the Wi-Fi network.

6.4 Configuring `dnsmasq`

Edit `/etc/dnsmasq.conf` (or add a new file in `/etc/dnsmasq.d/`) and add:

```
interface=wlan0
dhcp-range=192.168.10.10,192.168.10.50,12h
dhcp-option=3,192.168.10.1
server=1.1.1.1
server=8.8.8.8
no-resolv
```

This sets `wlan0` to give IPs in the 192.168.10.0/24 range and uses public DNS servers (Cloudflare and Google). The laptop's IP (192.168.10.1) acts as the gateway and DNS for clients. Restart `dnsmasq`:

```
sudo systemctl restart dnsmasq
```

6.5 Configuring `iptables`

Set up NAT and forwarding rules (example commands):

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j
    ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

These rules masquerade outgoing traffic on `eth0` and allow forwarding between interfaces. To preserve rules across reboots, you can install `iptables-persistent` or add the commands to a startup script.

6.6 DNS-over-TLS with Unbound

Integrate Unbound as a recursive resolver with DNS-over-TLS (DoT) support, ensuring that all DNS queries are encrypted and protected from eavesdropping or manipulation. Unlike plain DNS forwarding in dnsmasq, which sends queries in clear text, DoT secures the communication channel between the router and upstream resolvers.

DNS resolution was tested using both dig and nslookup, confirming that queries were successfully answered through Unbound with DoT enabled.

To avoid conflicts with dnsmasq, Unbound was configured to listen on a local port (5353) and securely forward DNS requests to Cloudflare and Google public resolvers.

We configured Unbound to listen on a local port (5353) to avoid conflicts with `dnsmasq`, and set it to forward DNS requests securely to Cloudflare and Google public resolvers:

```
# /etc/unbound/unbound.conf.d/dns-over-tls.conf
server:
    interface: 127.0.0.1@5353
    access-control: 127.0.0.0/8 allow
    do-tcp: yes
    do-udp: yes

forward-zone:
    name: "."
    forward-tls-upstream: yes
    forward-addr: 1.1.1.1@853
    forward-addr: 1.0.0.1@853
    forward-addr: 8.8.8.8@853
    forward-addr: 8.8.4.4@853
```

Finally, `dnsmasq` was instructed to forward client queries to Unbound by setting `server=127.0.0.1#535` This allowed Wi-Fi clients to continue using the router transparently, while DNS queries were securely encrypted upstream.

Testing with `dig` confirmed that responses were obtained via Unbound with TLS encryption enabled. This upgrade significantly improves user privacy compared to plain DNS.

7 Troubleshooting

During setup, several issues may arise. Key problems encountered and their solutions include:

- **"dhclient not found" error:** If a DHCP client is missing, ensure `isc-dhcp-client` or `dhclient` is installed via `sudo apt install isc-dhcp-client`.
- **Ethernet interface not recognized:** Some laptops have proprietary Ethernet controllers. Install necessary firmware or enable the interface with `sudo ip link set eth0 up`. Check `lspci` for details.
- **DBus errors on start:** Services like `dnsmasq` or `hostapd` may complain if `dbus` is not running. Ensure the DBus daemon is active (`sudo systemctl start dbus`).
- **Iptables rules not working:** If clients cannot access the Internet, check that forwarding is enabled and iptables rules are correct. Use `sudo iptables -t nat -L -v` and `sudo iptables -L` to verify rules. Ensure the MASQUERADE rule is applied to the correct outbound interface.

8 Firewall and DNS Configuration

Below is an example set of firewall rules and DNS settings used in this setup:

```
# Enable IP forwarding
sudo sysctl -w net.ipv4.ip_forward=1

# Firewall/NAT rules
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j
    ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT

# Sample /etc/dnsmasq.conf entries:
interface=wlan0
dhcp-range=192.168.10.10,192.168.10.50,12h
server=1.1.1.1
server=8.8.8.8
no-resolv
```

These settings ensure that any client on the `wlan0` network (192.168.10.x) receives DNS services and can reach the Internet through NAT on `eth0`. The `FORWARD` chains allow traffic for established connections to flow properly.

9 Testing and Results

After configuration, tests were performed to verify functionality:

- Connecting multiple devices (e.g., smartphones, laptops) to the new SSID succeeded, and each device received an IP in the 192.168.10.x range.
- Clients could reach the Internet (verified by browsing and `ping 8.8.8.8`).
- DNS resolution through `dnsmasq` and Unbound functioned correctly (e.g., `dig example.com` and `nslookup example.com` returned valid IPs).
- Logs from Unbound confirmed that DNS queries were forwarded securely using DNS-over-TLS (DoT) to Cloudflare and Google resolvers on port 853, ensuring that DNS traffic was encrypted.
- Throughput was sufficient for typical home usage, limited mainly by the laptop's hardware and the USB adapter's capabilities.
- The firewall correctly masked internal IPs; externally, all traffic appeared to originate from the laptop's `eth0` address.

10 Future Scope

Future improvements to this project could include:

- Adding VPN server functionality (e.g., OpenVPN or WireGuard) so that remote clients can securely connect to the home network.
- Implementing an Intrusion Detection or Prevention System (IDS/IPS) such as Snort or Suricata to monitor network traffic for threats.
- Developing a lightweight GUI or web-based front end to simplify management of router settings.
- Porting the setup to a low-power platform like a Raspberry Pi with a compatible Wi-Fi adapter for a compact and energy-efficient solution.

11 Conclusion

This project demonstrates how an old laptop can be repurposed into a versatile Linux-based Wi-Fi router. By using a USB Wi-Fi adapter and configuring `hostapd`, `dnsmasq`, and `iptables`, we achieved a secure network gateway with custom DNS and firewall rules. To further enhance privacy and integrity, `Unbound` was integrated as a recursive DNS resolver with DNS-over-TLS (DoT) support, ensuring that all DNS queries between the router and upstream resolvers are encrypted and protected from eavesdropping or manipulation. All services were managed in a minimal Ubuntu Server environment, and various troubleshooting steps were required to address driver and configuration issues. The resulting system successfully provides Wi-Fi access with controlled and encrypted network services. This approach offers a cost-effective, privacy-focused, and customizable alternative to commercial routers.

References

[1] S. Kadam and N. Pirjade, *From Laptop to Router: A Practical Project on Network Sharing and DNS Management*, GitHub Repository, 2025. Available at: `https://github.com/siddhantkadam/router-dns`

[2] Jouni Malinen, *hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP Authenticator*, Official Documentation, `https://w1.fi/hostapd/`, accessed Aug 2025.

[3] Simon Kelley, *Dnsmasq – A lightweight DHCP and caching DNS server*, Official Documentation, `http://www.thekelleys.org.uk/dnsmasq/doc.html`, accessed Aug 2025.

[4] Netfilter Project, *iptables(8) Linux manual page*, `https://ipset.netfilter.org/iptables.man.html`, accessed Aug 2025.

[5] Canonical Ltd., *Ubuntu Server Guide*, `https://ubuntu.com/server/docs`, accessed Aug 2025.

[6] GitHub Repository, *Realtek RTL8188FU Linux Driver (kelebek333/rtl8188fu)*, `https://github.com/kelebek333/rtl8188fu`, accessed Aug 2025.

[7] NLnet Labs. *Unbound DNS Resolver Documentation*. Available at: `https://nlnetlabs.nl/documentation/unbound/`

[8] P. Hoffman and P. McManus, *Specification for DNS over Transport Layer Security (TLS)*, RFC 7858, IETF, 2016. Available at: `https://datatracker.ietf.org/doc/html/rfc7858`

[9] P. Hoffman and P. McManus, *DNS Queries over HTTPS (DoH)*, RFC 8484, IETF, 2018. Available at: `https://datatracker.ietf.org/doc/html/rfc8484`