NEON SHADOWS

Rebel's Privacy Codex

Contents

1 Using Tor for Anonymous Browsing **3**
- 1.1 What is Tor and When to Use It . . . 3
  - 1.1.1 When to Use Tor . . . 3
  - 1.1.2 Against What Adversaries . . . 3
- 1.2 Configuring Tor on Linux . . . 4
  - 1.2.1 Prerequisites . . . 4
  - 1.2.2 Steps . . . 4
- 1.3 Configuring Tor on Windows . . . 5
  - 1.3.1 Prerequisites . . . 5
  - 1.3.2 Steps . . . 6
- 1.4 Configuring Tor on Mobile (Android/iOS) . . . 6
  - 1.4.1 Prerequisites . . . 6
  - 1.4.2 Steps . . . 6
- 1.5 Important Notes . . . 7
- 1.6 Troubleshooting . . . 7

2 Setting Up and Securing a VPS **9**
- 2.1 What is a VPS and When to Use It . . . 9
  - 2.1.1 When to Use a VPS . . . 9
  - 2.1.2 Against What Adversaries . . . 9
- 2.2 Setting Up a VPS on Ubuntu . . . 10
  - 2.2.1 Prerequisites . . . 10
  - 2.2.2 Steps . . . 10
- 2.3 Using the VPS for Privacy . . . 12
- 2.4 Important Notes . . . 13
- 2.5 Troubleshooting . . . 13

3 Setting Up a DNS Server **15**
- 3.1 What is DNS and Why Do Leaks Matter? . . . 15
  - 3.1.1 Why DNS Leaks Are Bad . . . 15
- 3.2 Solution 1: Using the Hosts File . . . 16
  - 3.2.1 Limitations . . . 16
  - 3.2.2 Advantages . . . 16
  - 3.2.3 Disadvantages . . . 16
  - 3.2.4 Ad-Blocking with Hosts File . . . 16
  - 3.2.5 Steps for Linux . . . 17
  - 3.2.6 Steps for Windows . . . 18
- 3.3 Solution 2: Using Public DNS Servers . . . 18
  - 3.3.1 Privacy-Respecting Public DNS Servers . . . 18

3.3.2 Advantages . . . 19
3.3.3 Disadvantages . . . 19
3.3.4 Steps for Linux . . . 19
3.3.5 Steps for Windows . . . 20
3.4 Solution 3: Local Unbound DNS Server with Forwarding (Linux) . . . 20
3.4.1 Steps for Linux . . . 20
3.4.2 Advantages . . . 22
3.4.3 Disadvantages . . . 22
3.5 Solution 4: Local Unbound DNS Server as Full Resolver (Linux) . . . 22
3.5.1 Steps for Linux . . . 22
3.5.2 Ad-Blocking with Unbound . . . 24
3.5.3 Advantages . . . 24
3.5.4 Disadvantages . . . 24
3.6 Solution 5: Unbound VPS with DoH and Local Stub Resolver . . . 25
3.6.1 Prerequisites . . . 25
3.6.2 Steps for VPS (Full Unbound Resolver) . . . 25
3.6.3 Steps for Local Machine (Stub Resolver) . . . 27
3.6.4 Advantages . . . 28
3.6.5 Disadvantages . . . 28
3.7 Preventing DNS Leaks in VPN and SSH Tunneling . . . 29
3.7.1 Commercial VPNs . . . 29
3.7.2 Self-Hosted VPNs . . . 29
3.7.3 SSH Tunneling . . . 29
3.7.4 General Tips . . . 30
3.8 Fixing Browser DNS Leaks . . . 30
3.9 Important Notes . . . 31
3.10 Troubleshooting . . . 32

4 Using a VPN for Encrypted Connections 33
4.1 What is a VPN and When to Use It . . . 33
4.1.1 When to Use a VPN . . . 33
4.1.2 Against What Adversaries . . . 33
4.2 Top Paid VPN Providers . . . 34
4.3 Configuring a VPN on Linux . . . 34
4.3.1 Prerequisites . . . 34
4.3.2 Steps . . . 34
4.4 Configuring a VPN on Windows . . . 35
4.4.1 Prerequisites . . . 35
4.4.2 Steps . . . 35
4.5 Configuring a VPN on Mobile (Android/iOS) . . . 36
4.5.1 Prerequisites . . . 36
4.5.2 Steps . . . 36
4.6 Setting Up a Self-Hosted VPN on a VPS . . . 37
4.6.1 Prerequisites . . . 37
4.6.2 Steps (OpenVPN on Ubuntu VPS) . . . 37
4.6.3 Steps (WireGuard Alternative) . . . 39
4.6.4 Advantages of Self-Hosted VPN . . . 40
4.6.5 Disadvantages . . . 41

4.7 Free VPN Services . . . 41
4.8 Important Notes . . . 41
4.9 Troubleshooting . . . 41

5 Configuring VPN-over-Tor on Linux and Windows 43
5.1 What is VPN-over-Tor and When to Use It . . . 43
5.1.1 When to Use VPN-over-Tor . . . 43
5.1.2 Against What Adversaries . . . 43
5.2 Configuring VPN-over-Tor on Linux . . . 44
5.2.1 Prerequisites . . . 44
5.2.2 Steps . . . 44
5.3 Configuring VPN-over-Tor on Windows . . . 46
5.3.1 Prerequisites . . . 46
5.3.2 Steps . . . 46
5.4 Important Notes . . . 47
5.5 Troubleshooting . . . 47

6 Configuring Tor-over-VPN on Linux and Windows 49
6.1 What is Tor-over-VPN and When to Use It . . . 49
6.1.1 When to Use Tor-over-VPN . . . 49
6.1.2 Against What Adversaries . . . 49
6.2 Configuring Tor-over-VPN on Linux . . . 49
6.2.1 Prerequisites . . . 49
6.2.2 Steps . . . 50
6.3 Configuring Tor-over-VPN on Windows . . . 51
6.3.1 Prerequisites . . . 51
6.3.2 Steps . . . 51
6.4 Important Notes . . . 52
6.5 Troubleshooting . . . 52

7 Comparing Privacy Solutions 53
7.1 Comparison Table . . . 53
7.2 Recommendations . . . 53

8 Browser Rebellion: Privacy Hacks 55
8.1 Privacy Tweaks for Firefox . . . 55
8.1.1 Key about:config Settings . . . 55
8.1.2 Privacy-Respecting Firefox Forks . . . 56
8.2 Privacy Tweaks for Chrome/Chromium . . . 56
8.2.1 Key Command-Line Flags . . . 56
8.2.2 Privacy-Respecting Chromium Forks . . . 57
8.3 Privacy-Respecting Mobile Browsers . . . 57
8.4 Conclusion . . . 57

9 Search Revolt: Privacy Engines 59
9.1 How Google Tracks Your Searches . . . 59
9.1.1 On Desktop . . . 59
9.1.2 On Android . . . 59
9.2 Privacy-Respecting Public Search Engines . . . 60

9.3 Privacy-Respecting Meta-Search Engines 60
9.4 Advanced: Self-Hosting SearXNG 61
9.4.1 Local Hosting (Without Docker) 61
9.4.2 VPS Hosting (With Docker) 61
9.5 Conclusion 61

10 Android Unchained: Breaking Free from Spyware 63
10.1 Choosing a Privacy-Friendly Android Phone 63
10.2 Flashing Privacy-Respecting Firmwares 64
10.2.1 Recommended Firmwares 64
10.2.2 Flashing Steps on Linux (General Guide) 64
10.2.3 Flashing Steps on Windows (General Guide) 64
10.3 Replacing Google Apps and Services 65
10.3.1 Step-by-Step Replacements 65
10.3.2 Additional Tips 65
10.4 Conclusion 65

Chapter 1

Using Tor for Anonymous Browsing

1.1 What is Tor and When to Use It

Tor (The Onion Router) is a free, open-source tool that anonymizes your internet traffic by routing it through multiple volunteer-operated servers (nodes) worldwide. Each node encrypts your data, peeling back layers (like an onion) until it reaches its destination, hiding your IP address and

browsing activity. This chapter introduces Tor as a standalone tool, which later chapters (e.g., Chapter 6) combine with VPNs.

1.1.1 When to Use Tor

Use Tor when you need:

- **Protecting privacy**: Hiding browsing from ISPs, websites, or trackers (e.g., for sensitive research or whistleblowing).
- **Bypassing censorship**: Accessing blocked websites in restrictive regions.
- **Accessing onion sites**: Visiting `.onion` websites, only accessible via Tor.

1.1.2 Against What Adversaries

Tor protects against:

- **ISPs and local networks**: They can't see your browsing destinations or data; they only know you're using Tor.
- **Websites**: They see a Tor exit node's IP, not your real IP.

It does not protect against:

- **Malicious Tor exit nodes**: Unencrypted traffic (non-HTTPS) may be monitored.
- **Global adversaries**: Advanced entities (e.g., governments) may attempt traffic correlation.
- **ISP visibility**: Your ISP sees you're using Tor, which may raise suspicion.

Note for Beginners: Tor is slower than regular browsing due to its multi-hop routing. Use it only for anonymity-critical tasks, and always access HTTPS websites. Disable WebRTC in browsers to prevent leaks:

- Firefox: Set `media.peerconnection.enabled = false` in `about:config`.
- Chrome: Go to `chrome://settings/privacy`, disable WebRTC.

1.2 Configuring Tor on Linux

This section guides you through installing and using the Tor Browser on Linux (Ubuntu, Debian, Fedora, Gentoo). We assume you're new to Tor but can use a terminal.

1.2.1 Prerequisites

- A Linux computer.
- Basic terminal familiarity.

1.2.2 Steps

1. **Install Tor Browser**:
   - Download the Tor Browser from https://www.torproject.org.
   - Extract the file (e.g., `tor-browser-linux64-*.tar.xz`):

```
tar -xvf tor-browser-linux64-*.tar.xz
```

   - Move to your home directory:

```
mv tor-browser_en-US ~/tor-browser
```

   - Start the Tor Browser:

```
cd ~/tor-browser
./start-tor-browser.desktop
```

2. **(Optional) Install Tor Service**:
   - For advanced use (e.g., non-browser apps). Tor Browser is sufficient for most users.
   - **For Ubuntu/Debian**:

```
sudo apt update
sudo apt install tor
sudo systemctl start tor
sudo systemctl enable tor
```

   - **For Fedora**:

```
sudo dnf install tor
sudo systemctl start tor
sudo systemctl enable tor
```

- **For Gentoo**:

```
sudo emerge -av net-vpn/tor
```

- If using OpenRC:

```
sudo /etc/init.d/tor start
sudo rc-update add tor default
```

- If using systemd:

```
sudo systemctl start tor
sudo systemctl enable tor
```

3. **Connect to Tor**:

   - Open the Tor Browser.
   - Click "Connect" in the startup window.
   - If blocked, click "Configure" and enable bridges (https://bridges.torproject.org).

4. **Verify the Setup**:

   - In Tor Browser, visit https://check.torproject.org to confirm Tor is active.
   - Check your public IP:

```
curl https://ifconfig.me
```

     It should show a Tor exit node's IP.

   - Visit https://ipleak.net to check for leaks (DNS, WebRTC).

1.3 Configuring Tor on Windows

This section covers Tor setup on Windows 10 or 11.

1.3.1 Prerequisites

- A Windows computer.
- Basic software installation knowledge.

1.3.2 Steps

1. **Install Tor Browser**:
   - Download from https://www.torproject.org.
   - Run the installer.
   - Launch from the Start menu or desktop.
2. **Connect to Tor**:
   - Open Tor Browser and click "Connect".
   - If blocked, enable bridges (https://bridges.torproject.org).
3. **Verify the Setup**:
   - Visit https://check.torproject.org in Tor Browser.
   - In Command Prompt (search "cmd"):

```
Invoke-WebRequest -Uri https://ifconfig.me
```

     (Or install `curl` from https://curl.se/windows/ and use `curl https://ifconfig.me`.) It should show a Tor exit node's IP.
   - Visit https://ipleak.net.

1.4 Configuring Tor on Mobile (Android/iOS)

This section provides detailed instructions for setting up Tor on mobile devices.

1.4.1 Prerequisites

- An Android or iOS device.
- Internet access and basic app installation knowledge.

1.4.2 Steps

1. **Install Tor Apps**:
   - **Android**:
     - Install **Tor Browser** from Google Play or https://www.torproject.org.
     - Optionally, install **Orbot** (Tor proxy for non-browser apps) from Google Play or https://www.torproject.org.
     - For Tor Browser, download the `.apk` file if side-loading from the official site.
     - For Orbot, ensure VPN mode is enabled in settings for system-wide Tor routing.
   - **iOS**:
     - Install **Onion Browser** from the App Store.

  - No additional proxy app is needed, as Onion Browser handles Tor routing internally.

2. **Connect to Tor**:

   - **Android (Tor Browser)**:
     - Open Tor Browser and tap "Connect".
     - If blocked, tap "Configure" and enable bridges from https://bridges.torproject.org.
   - **Android (Orbot)**:
     - Open Orbot, select "VPN Mode" or "Apps VPN Mode" for specific apps.
     - Tap the onion icon to start the Tor connection.
     - If blocked, go to settings, select "Bridges", and enable built-in bridges or request custom ones from https://bridges.torproject.org.
   - **iOS (Onion Browser)**:
     - Open Onion Browser and tap "Connect to Tor".
     - If blocked, go to settings, select "Bridge Configuration", and enable bridges from https://bridges.torproject.org.

3. **Verify the Setup**:

   - In Tor Browser (Android) or Onion Browser (iOS), visit https://check.torproject.org to confirm Tor is active.
   - Check for leaks at https://ipleak.net. Ensure WebRTC is disabled (automatically handled by Tor Browser and Onion Browser).
   - For Orbot (Android), verify the connection in the app's status screen, which shows the Tor circuit and exit node IP.

1.5 Important Notes

- **Speed**: Tor's routing slows browsing.
- **Security**: Use HTTPS to avoid exit node risks. Disable WebRTC to prevent leaks.
- **Tor Restrictions**: Enable bridges if blocked.
- **Limitations**: ISPs see Tor usage; Chapter 6 addresses this.

1.6 Troubleshooting

- **Tor Not Connecting**:
  - Linux: Check Tor Browser or service status (`systemctl status tor`; Gentoo OpenRC: `/etc/init.d/tor status`).
  - Windows: Verify `tor.exe` in Task Manager.

  - Mobile: Ensure Tor Browser, Orbot, or Onion Browser is updated; check app logs for errors.
  - Enable bridges (https://bridges.torproject.org).

- **Leaks Detected**: Use https://ipleak.net; ensure browsing in Tor Browser or Onion Browser and WebRTC is disabled.
- **Slow Connection**: Try a new Tor circuit ("New Circuit" in Tor Browser/Onion Browser or "New Identity" in Orbot).

Chapter 2

Setting Up and Securing a VPS

This optional chapter guides you through setting up and securing a Virtual Private Server (VPS) using Ubuntu 22.04 (or 24.04 for latest LTS). A VPS enables advanced privacy solutions, such as hosting your own VPN (Chapter 4), tunneling traffic via SSH, or running a DNS server (Chapter 3). While not required for basic Tor or VPN use, a VPS offers greater control and privacy. This section assumes basic terminal familiarity.

2.1 What is a VPS and When to Use It

A VPS is a virtualized server hosted by a provider (e.g., DigitalOcean, Linode, Vultr, Hetzner) that you can configure for various tasks. It acts as a remote computer, accessible via SSH, with a dedicated IP address.

2.1.1 When to Use a VPS

Use a VPS for:

- **Self-hosted VPN**: Configure OpenVPN or WireGuard (Chapter 4).
- **SSH Tunneling**: Route web traffic through the VPS for privacy or to bypass restrictions.
- **DNS Server**: Run a DNS server (e.g., Unbound) for DNS-over-HTTPS (Chapter 3).
- **Privacy**: Choose a VPS in a privacy-friendly country to minimize surveillance.

2.1.2 Against What Adversaries

A VPS protects against:

- **ISPs and local networks**: They see only the VPS's IP for tunneled traffic.
- **Third-party VPN providers**: Avoid reliance on external providers by hosting your own services.

It does not protect against:

- **VPS provider logging**: Some providers may log activity unless disabled.

- **Misconfigurations**: Incorrect setups can expose data or IPs.
- **Global adversaries**: Advanced traffic analysis may still apply.

Note for Beginners: VPS setup requires technical skills. Choose providers in privacy-friendly countries (e.g., Switzerland, Panama, Iceland, Romania) and follow security best practices.

2.2 Setting Up a VPS on Ubuntu

2.2.1 Prerequisites

- A VPS account (e.g., DigitalOcean, Linode, Vultr, Hetzner; $5–10/month).
- A Linux, Windows, or macOS computer for configuration.
- Basic terminal knowledge.

2.2.2 Steps

1. **Choose a VPS Provider and Location**:
   - Select a provider like DigitalOcean, Linode, Vultr, or Hetzner.
   - Select a VPS location with strong privacy laws (e.g., Switzerland, Panama, Iceland, Romania) to minimize surveillance risks. These countries have no mandatory data retention and are outside Five-/Nine-/Fourteen-Eyes alliances.
   - Ensure a static IP for reliable connections.
2. **Create the VPS**:
   - Log in to your provider's dashboard.
   - Create an Ubuntu 22.04 (or 24.04) droplet/server.
   - Note the assigned IP address (e.g., `your-vps-ip`).
3. **Initial SSH Access**:

```
ssh root@your-vps-ip
```

4. If prompted, enter the root password provided by the VPS provider.
5. **Update the System**:

```
sudo apt update && sudo apt upgrade -y
sudo apt install net-tools ufw -y
```

6. **Set Up a Non-Root User**:
   - Create a user (e.g., `vpsuser`):

```
adduser vpsuser
usermod -aG sudo vpsuser
```

- Log out and reconnect as the new user:

```
exit
ssh vpsuser@your-vps-ip
```

7. **Configure SSH Key-Based Authentication**:

- On your local computer, generate an SSH key (if not already done):

```
ssh-keygen -t rsa -b 4096
```

- Copy the public key to the VPS:

```
ssh-copy-id vpsuser@your-vps-ip
```

- On the VPS, edit SSH configuration:

```
sudo nano /etc/ssh/sshd_config
```

- Set:

```
PasswordAuthentication no
PubkeyAuthentication yes
```

- Save and restart SSH:

```
sudo systemctl restart sshd
```

- Verify permissions:

```
chmod 600 ~/.ssh/authorized_keys
```

8. **Configure Firewall**:

```
sudo ufw allow OpenSSH
sudo ufw enable
sudo ufw status
```

9. Add specific ports later (e.g., 1194/udp for OpenVPN, 51820/udp for WireGuard).

10. For IPv6, disable or configure:

```
sudo nano /etc/sysctl.conf
```

Add:

```
net.ipv6.conf.all.disable_ipv6=1
```

Apply:

```
sudo sysctl -p
```

Alternatively, use `ip6tables` for IPv6 firewall rules:

```
sudo ip6tables -A INPUT -p tcp --dport 22 -j ACCEPT
sudo ip6tables -P INPUT DROP
sudo ip6tables-save > /etc/iptables/ip6tables.rules
```

11. **Enable Automatic Updates**:

```
sudo apt install unattended-upgrades -y
sudo dpkg-reconfigure --priority=low unattended-upgrades
```

12. Edit configuration:

```
sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

13. Ensure:

```
"Unattended-Upgrade::Allowed-Origins" {
    "${distro_id}:${distro_codename}";
    "${distro_id}:${distro_codename}-security";
};
```

14. **Disable VPS Provider Logging (if available)**:
    - Check your provider's dashboard for logging options.
    - Disable any activity or access logs.
15. **Verify Setup**:
    - Check SSH status:

```
sudo systemctl status sshd
```

    - Test connectivity:

```
ping your-vps-ip
```

    - Scan open ports:

```
nmap your-vps-ip
```

      Ensure only port 22/tcp (SSH) is open initially.

2.3 Using the VPS for Privacy

- **Self-hosted VPN**: Configure OpenVPN or WireGuard (Chapter 4).
- **SSH Tunneling**: Route traffic through the VPS:

```
ssh -D 1080 vpsuser@your-vps-ip
```

  Configure your browser to use a SOCKS5 proxy at `localhost:1080`.

- **DNS Server**: Set up Unbound for DNS-over-HTTPS (Chapter 3).

2.4 Important Notes

- **Security**: Always use key-based SSH and a firewall.
- **Backups**: Use `rsync` for regular backups:

```
rsync -avz /etc vpsuser@your-vps-ip:/backups/
```

- **Monitoring**: Use `fail2ban` for intrusion prevention:

```
sudo apt install fail2ban -y
sudo systemctl enable fail2ban
```

2.5 Troubleshooting

- **SSH Not Connecting**:
  - Verify `sshd_config` settings and `systemctl status sshd`.
  - Check firewall: `sudo ufw status`.
- **VPS Unresponsive**: Check provider dashboard for server status.
- **Port Issues**: Scan with `nmap your-vps-ip` to ensure only intended ports are open.

Chapter 3

Setting Up a DNS Server

The Domain Name System (DNS) translates human-readable domain names (e.g., `example.com`) into IP addresses (e.g., `192.0.2.1`) that computers use to communicate. By default, your ISP assigns DNS servers via DHCP, which can log your queries, exposing your browsing habits. This chapter is the definitive guide to preventing DNS leaks, covering solutions from a simple hosts file to a full-blown Unbound DNS server on your own VPS. Other chapters (e.g., Chapter 4) reference this for DNS leak fixes. We'll also throw shade at Mozilla for taking *seventeen years* to address DNS leaks in Firefox's SOCKS5 proxy, only adding a user-friendly checkbox in 2024 after users suffered through `about:config` tweaks since dial-up days. *Thanks for the wait, Mozilla.* This chapter assumes basic terminal familiarity.

3.1 What is DNS and Why Do Leaks Matter?

DNS is the internet's phonebook. When you visit `example.com`, your device queries a DNS server to get the corresponding IP address. By default, these queries are unencrypted and sent to your ISP's DNS servers, which can log every domain you visit, building a detailed profile of your online activity. Even worse, DNS leaks occur when your device bypasses your VPN or proxy and sends DNS queries to your ISP or another unintended server, revealing your real location or browsing habits.

3.1.1 Why DNS Leaks Are Bad

DNS leaks are a privacy nightmare because:

- **ISP Tracking**: Your ISP sees every domain you visit, even if you use a VPN, unless DNS queries are properly routed or encrypted.
- **Third-Party Exposure**: Leaked queries can be intercepted by malicious actors on public Wi-Fi or compromised networks.
- **Geo-Location Leaks**: If your DNS server is in your home country (e.g., detected via https://ipleak.net), it can reveal your location despite using a VPN in, say, Switzerland.
- **Censorship and Surveillance**: In restrictive regions, DNS logs can expose your attempts to access blocked sites.

To test for leaks, visit https://ipleak.net or https://dnsleaktest.com after configuring your VPN or proxy. If your ISP's DNS servers appear, you've got a leak.

3.2 Solution 1: Using the Hosts File

The simplest way to avoid DNS queries is to use your system's hosts file, which maps hostnames to IP addresses locally, bypassing external DNS servers entirely. This is a primitive but effective method for specific domains, requiring no software or network changes.

3.2.1 Limitations

The hosts file doesn't support wildcards (e.g., `*.example.com`), only exact hostnames or aliases (multiple names per IP). It's inflexible for dynamic domains, as IPs for large services (e.g., Google, Netflix) change frequently—sometimes every 5 minutes to 24 hours due to load balancing or CDN updates. Maintaining a hosts file for many domains or subdomains is impractical, as you'd need to manually update IPs constantly. Still, it's a good starting point for static, low-frequency sites.

3.2.2 Advantages

- **Maximum Privacy**: No external queries, eliminating leak risks.
- **Simplicity**: No additional software or configuration needed.

3.2.3 Disadvantages

- **Limited Scalability**: Impractical for many or dynamic domains.
- **Manual Maintenance**: Requires frequent IP updates for changing services.

3.2.4 Ad-Blocking with Hosts File

You can extend the hosts file to block ads, trackers, and malware by mapping known malicious or advertising domains to a null IP address, preventing your device from connecting to them. Popular curated lists include the one from https://github.com/StevenBlack/hosts, which consolidates sources into a unified hosts file with options for extensions like porn, social media, fake news, and gambling blocking. Other similar repositories include https://github.com/ScriptTiger/Unified-Hosts-AutoUpdate for automated updates on Windows, https://github.com/tanrax/maza-ad-blocking for a bash script, and https://github.com/feross/hostile for a command-line utility.

To block domains, map them to `0.0.0.0` (preferred) or `127.0.0.1`. Using `0.0.0.0` is a non-routable meta-address that designates an invalid target, making it faster (no timeout wait) and avoiding interference with any local web server, unlike `127.0.0.1` which points to localhost and could bombard a running server.

Advantages of Blocking Ad Domains

- **Enhanced Privacy**: Prevents trackers and advertisers from logging your activity and building profiles.
- **Improved Security**: Blocks malware and phishing domains.
- **Faster Browsing**: Reduces page load times by skipping ad content.
- **Network-Wide Application**: Applies to all applications on the device.

Disadvantages of Blocking Ad Domains

- **Potential Site Breakage**: Overly aggressive lists may block legitimate content or break websites.
- **Maintenance Required**: Lists need regular updates to stay effective.
- **No Granularity**: Blocks entire domains, not specific elements like browser extensions can.
- **False Positives**: May inadvertently block useful sites.

Note that for privacy, there are essentially no disadvantages—it only enhances it by reducing exposure to third-party trackers.

To use a list like StevenBlack's:

1. Download the hosts file: `wget https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts -O hosts`.
2. Append or replace in your system's hosts file (e.g., `sudo cat hosts >> /etc/hosts` on Linux).
3. Clear DNS cache and verify as in the main steps.

For automation, use scripts from the repositories or set up a cron job (e.g., weekly update).

3.2.5 Steps for Linux

1. **Open the Hosts File**:

```
sudo nano /etc/hosts
```

2. **Add Hostname-to-IP Mappings**: Add lines in the format `IP_address hostname [alias]`. Example:

```
192.0.2.1 example.com www.example.com
8.8.8.8 google.com mail.google.com
```

   Use `nslookup example.com` or `ping example.com` to find current IPs (note: these may change).

3. **Save and Exit**: Press `Ctrl+O`, `Enter`, then `Ctrl+X`.

4. **Clear DNS Cache** (if applicable):

```
sudo systemctl restart systemd-resolved
```

5. **Verify**: Test with:

```
ping example.com
```

   Ensure the IP matches your hosts file entry. Check https://ipleak.net to confirm no DNS queries are sent.

3.2.6 Steps for Windows

1. **Open the Hosts File**:
   - Open Notepad as Administrator: Right-click Notepad, select "Run as administrator".
   - Open `C:\Windows\System32\drivers\etc\host`.
2. **Add Hostname-to-IP Mappings**: Add lines like:

```
192.0.2.1 example.com www.example.com
8.8.8.8 google.com mail.google.com
```

   Find IPs using `nslookup example.com` in Command Prompt.

3. **Save and Exit**: Save the file (ensure no `.txt` extension is added).
4. **Clear DNS Cache**:

```
ipconfig /flushdns
```

5. **Verify**:

```
ping example.com
```

   Check https://ipleak.net to confirm no DNS queries are sent.

3.3 Solution 2: Using Public DNS Servers

To bypass your ISP's DNS servers, you can configure your system to use privacy-respecting public DNS servers. These servers are often faster and may offer security features like malicious domain blocking, but they require trust in the provider, which isn't ideal for the ultra-paranoid.

3.3.1 Privacy-Respecting Public DNS Servers

- **Quad9** (Switzerland): `9.9.9.9`, `149.112.112.112`. Blocks malicious domains, supports DNSSEC, and has a no-logs policy. Located outside Five/Nine/Fourteen Eyes, but you're still trusting a third party.

- **Cloudflare** (USA): `1.1.1.1`, `1.0.0.1`. Fast, supports DoH and DNSSEC, claims not to log identifiable data. USA location raises slight privacy concerns due to potential surveillance.
- **AdGuard DNS** (Cyprus): `94.140.14.14`, `94.140.15.15`. Blocks ads and trackers, supports DoH. Cyprus is outside major surveillance alliances, but third-party trust is still required.

3.3.2 Advantages

- **Easy Setup**: Simple to configure at system or router level.
- **Enhanced Security**: Providers like Quad9 block malicious domains.
- **DoH Support**: Encrypts queries, reducing ISP visibility.

3.3.3 Disadvantages

- **Third-Party Trust**: Providers may log data despite no-logs claims, reducing privacy.
- **Limited Control**: You rely on external infrastructure, which could be compromised or subpoenaed.

3.3.4 Steps for Linux

1. **Edit Network Manager Configuration**:

```
sudo nano /etc/NetworkManager/NetworkManager.conf
```

   Add under `[main]`:

```
dns=none
```

   Save and exit.

2. **Set DNS Servers**:

```
sudo nano /etc/resolv.conf
```

   Add:

```
nameserver 9.9.9.9
nameserver 1.1.1.1
```

   Save and exit. To prevent DHCP overwrites, make the file immutable:

```
sudo chattr +i /etc/resolv.conf
```

3. **Restart Network Manager**:

```
sudo systemctl restart NetworkManager
```

4. **Verify**:

```
nslookup example.com
```

Ensure the server is `9.9.9.9` or `1.1.1.1`. Check https://ipleak.net.

3.3.5 Steps for Windows

1. **Open Network Settings**:
   - Go to `Settings > Network & Internet > Ethernet` (or `Wi-Fi`).
   - Click your connection, then `Edit` under `IP settings`.
2. **Set Manual DNS**:
   - Select `Manual`, turn on `IPv4`.
   - Enter `Preferred DNS`: `9.9.9.9`, `Alternate DNS`: `1.1.1.1`.
   - Save changes.
3. **Clear DNS Cache**:

```
ipconfig /flushdns
```

4. **Verify**:

```
nslookup example.com
```

Confirm the DNS server is `9.9.9.9` or `1.1.1.1`. Check https://ipleak.net.

3.4 Solution 3: Local Unbound DNS Server with Forwarding (Linux)

Unbound is a lightweight, privacy-focused DNS resolver that supports QNAME minimization (reduces query exposure), DNSSEC (verifies responses), DoH (encrypts queries), and caching (reduces external queries). In this setup, Unbound runs locally and forwards queries to privacy-respecting public DNS servers. It's ideal for users seeking a balance of privacy and ease of setup. Unbound is Linux-centric; Windows users can compile it with Cygwin, but it's not natively supported, so Windows users may prefer public DNS or the hosts file.

3.4.1 Steps for Linux

1. **Install Unbound**:

```
sudo apt update
sudo apt install unbound -y
```

2. **Configure Unbound**:

```
sudo nano /etc/unbound/unbound.conf
```

Add or modify:

```
server:
    do-ip4: yes
    do-ip6: no
    interface: 127.0.0.1
    access-control: 127.0.0.0/8 allow
    do-not-query-localhost: no
    harden-dnssec-stripped: yes
    harden-glue: yes
    harden-referral-path: yes
    use-caps-for-id: yes
    private-address: 10.0.0.0/8
    private-address: 172.16.0.0/12
    private-address: 192.168.0.0/16
    qname-minimisation: yes
    cache-max-ttl: 86400
    cache-min-ttl: 3600
forward-zone:
    name: "."
    forward-addr: 9.9.9.9
    forward-addr: 1.1.1.1
    forward-tls-upstream: yes
```

This enables QNAME minimization, DNSSEC, caching, and DoH to Quad9/Cloudflare. The added harden options and private-address prevent leaks and enhance privacy by blocking responses with private IPs and hardening against DNS attacks.

3. **Download Root Hints** (optional for forwarding, but good for DNSSEC):

```
sudo wget -O /var/lib/unbound/root.hints https://www.internic
   .net/domain/named.root
```

Configure Unbound to use root hints:

```
sudo nano /etc/unbound/unbound.conf
```

Add:

```
server:
    root-hints: "/var/lib/unbound/root.hints"
```

4. **Set Unbound as Local Resolver**:

```
sudo nano /etc/resolv.conf
```

Add:

```
nameserver 127.0.0.1
```

Make immutable:

```
sudo chattr +i /etc/resolv.conf
```

5. **Restart Unbound**:

```
sudo systemctl restart unbound
sudo systemctl enable unbound
```

6. **Verify**:

```
dig example.com @127.0.0.1
```

Check https://ipleak.net to ensure only `127.0.0.1` or DoH servers (e.g., Quad9, Cloudflare) appear.

3.4.2 Advantages

- **Easy Setup**: Simple configuration using trusted public servers for resolution.
- **Privacy Improvement**: Encrypts queries with DoH, minimizes exposure with QNAME minimization, and caches locally to reduce external queries.
- **Performance**: Caching speeds up repeated queries, leveraging fast public resolvers.

3.4.3 Disadvantages

- **Third-Party Trust**: Relies on public DNS providers (e.g., Quad9, Cloudflare), which may log data despite no-logs claims, reducing privacy.
- **Limited Control**: Queries are sent to external servers, potentially exposing browsing patterns to providers or surveillance.

3.5 Solution 4: Local Unbound DNS Server as Full Resolver (Linux)

In this setup, Unbound runs locally as a full recursive resolver, using only root hints to resolve domains without forwarding to external DNS providers. This eliminates reliance on third-party servers, maximizing privacy at the cost of higher latency and CPU usage.

3.5.1 Steps for Linux

1. **Install Unbound**:

```
sudo apt update
sudo apt install unbound -y
```

2. **Configure Unbound**:

```
sudo nano /etc/unbound/unbound.conf
```

Add or modify:

```
server:
    do-ip4: yes
    do-ip6: no
    interface: 127.0.0.1
    access-control: 127.0.0.0/8 allow
    do-daemonize: yes
    harden-dnssec-stripped: yes
    harden-glue: yes
    harden-referral-path: yes
    use-caps-for-id: yes
    private-address: 10.0.0.0/8
    private-address: 172.16.0.0/12
    private-address: 192.168.0.0/16
    qname-minimisation: yes
    cache-max-ttl: 86400
    cache-min-ttl: 3600
    root-hints: "/var/lib/unbound/root.hints"
    auto-trust-anchor-file: "/var/lib/unbound/root.key"
```

This enables recursive resolution, DNSSEC validation, and caching without forwarding. The added options enhance privacy as in Solution 3.

3. **Download Root Hints**:

```
sudo wget -O /var/lib/unbound/root.hints https://www.internic
   .net/domain/named.root
```

4. **Initialize DNSSEC Anchor**: DNSSEC is optional but recommended for added security. It verifies the authenticity and integrity of DNS responses using digital signatures, protecting against spoofing and man-in-the-middle attacks. Advantages include enhanced security and trust in DNS data. Disadvantages: Slightly increases query time, can cause failures if misconfigured or if domains don't support it properly, and setup can be tricky due to the need for a verified trust anchor.

   To enable:

```
sudo unbound-anchor -a /var/lib/unbound/root.key || true
sudo chown unbound:unbound /var/lib/unbound/root.key
```

   Verify the trust anchor manually by comparing the contents of `root.key` with the official IANA XML at https://data.iana.org/root-anchors/root-anchors.xml (download via HTTPS and check hashes). Ensure your system's time is synchronized (use NTP). If the file exists, `unbound-anchor` will update it if needed.

5. **Set Unbound as Local Resolver**:

```
sudo nano /etc/resolv.conf
```

   Add:

```
nameserver 127.0.0.1
```

Make immutable:

```
sudo chattr +i /etc/resolv.conf
```

6. **Restart Unbound**:

```
sudo systemctl restart unbound
sudo systemctl enable unbound
```

7. **Verify**:

```
dig example.com @127.0.0.1
```

For DNSSEC, test with `dig com. SOA +dnssec` and check for the AD flag. Check https://ipleak.net to ensure only `127.0.0.1` appears (no external servers).

3.5.2 Ad-Blocking with Unbound

To block ads, trackers, and malware in Unbound (locally or on VPS), use `local-zone` directives to return NXDOMAIN or redirect to `0.0.0.0`. For large lists, create a separate file (e.g., `/etc/unbound/blocklist.conf`) with entries like:

```
local-zone: "ad.example.com" static
```

or for redirect:

```
local-zone: "ad.example.com" redirect
local-data: "ad.example.com. A 0.0.0.0"
```

Include it in `unbound.conf` with:

```
include: "/etc/unbound/blocklist.conf"
```

Populate `blocklist.conf` from curated lists (convert hosts files to Unbound format using scripts). Advantages and disadvantages are similar to hosts file ad-blocking: enhances privacy/security, but may break sites and requires maintenance. Restart Unbound after updates.

3.5.3 Advantages

- **Maximum Privacy**: Resolves queries independently, ensuring no external servers see your requests.
- **Full Control**: You manage all resolution, eliminating third-party logging risks.
- **DNSSEC Security**: Validates responses to prevent spoofing.

3.5.4 Disadvantages

- **Higher Latency/CPU**: Recursive resolution from root servers is slower and more resource-intensive than forwarding.
- **Maintenance**: Requires monthly root hints updates (`wget` command).
- **Unencrypted Queries**: Queries to root/authoritative servers are unencrypted unless DoT (DNS over TLS) is enabled with `do-tls-upstream: yes`.

3.6 Solution 5: Unbound VPS with DoH and Local Stub Resolver

The ultimate privacy setup runs Unbound as a recursive DNS server on your VPS, with your local machine acting as a stub resolver querying the VPS over DoH. This encrypts DNS traffic, minimizes third-party trust, and leverages your VPS's privacy-friendly location (e.g., Switzerland, Panama). In 2025, consider enabling Encrypted Client Hello (ECH) for enhanced privacy (supported by Cloudflare's DoH).

3.6.1 Prerequisites

- A VPS configured with Ubuntu 22.04, SSH key-based auth, and firewall (Chapter 2).
- A domain name (optional, for DoH certificate setup, e.g., via Let's Encrypt or NGINX native ACME).
- A Linux client (local machine) for stub resolver configuration.

3.6.2 Steps for VPS (Full Unbound Resolver)

1. **Install Unbound on the VPS**:

```
ssh vpsuser@your-vps-ip
sudo apt update
sudo apt install unbound -y
```

2. **Configure Unbound as Recursive Resolver**:

```
sudo nano /etc/unbound/unbound.conf
```

Add:

```
server:
    do-ip4: yes
    do-ip6: no
    interface: 0.0.0.0
    access-control: 0.0.0.0/0 allow
    harden-dnssec-stripped: yes
    harden-glue: yes
    harden-referral-path: yes
    use-caps-for-id: yes
    private-address: 10.0.0.0/8
    private-address: 172.16.0.0/12
    private-address: 192.168.0.0/16
    qname-minimisation: yes
    cache-max-ttl: 86400
    cache-min-ttl: 3600
    root-hints: "/var/lib/unbound/root.hints"
    do-tls-upstream: yes
    auto-trust-anchor-file: "/var/lib/unbound/root.key"
do-tls-downstream: yes
interface: 127.0.0.1@8053
```

For ad-blocking, add `include: "/etc/unbound/blocklist.conf"` as in Solution 4.

3. **Download Root Hints**:

```
sudo wget -O /var/lib/unbound/root.hints https://www.internic
   .net/domain/named.root
```

4. **Initialize DNSSEC Anchor**:

```
sudo unbound-anchor -a /var/lib/unbound/root.key || true
sudo chown unbound:unbound /var/lib/unbound/root.key
```

Verify as in Solution 4. DNSSEC is optional; see advantages/disadvantages there.

5. **Set Up DoH with Nginx and Let's Encrypt**: NGINX added native ACME protocol support in version 1.27.1 for automatic TLS certificate issuance and renewal (https://blog.nginx.org/blog/native-support-for-acme-protocol). For older versions, use Certbot. Meanwhile, Caddy and Caddy 2 have supported automatic TLS certificates since 2015 (https://caddyserver.com/docs/automatic-https)—*NGINX, you're a decade late to the party.*

   - Install Nginx and Certbot (for NGINX <1.27.1):

```
sudo apt install nginx certbot python3-certbot-nginx -y
```

   - For NGINX 1.27.1+, configure native ACME (example):

```
sudo nano /etc/nginx/nginx.conf
```

   Add to `http` block:

```
acme_certificate dns.yourdomain.com {
    acme_account_email your-email@example.com;
    acme_account_key /etc/nginx/acme/account.key;
}
```

   - For older NGINX, obtain a Let's Encrypt certificate:

```
sudo certbot --nginx -d dns.yourdomain.com
```

   - Configure Nginx for DoH:

```
sudo nano /etc/nginx/sites-available/dns
```

   Add:

```
server {
    listen 443 ssl;
    server_name dns.yourdomain.com;
    ssl_certificate /etc/letsencrypt/live/dns.yourdomain.
       com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/dns.
       yourdomain.com/privkey.pem;
    location /dns-query {
        proxy_pass http://127.0.0.1:8053;
```

```
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
           $proxy_add_x_forwarded_for;
    }
}
```

- Link and restart Nginx:

```
sudo ln -s /etc/nginx/sites-available/dns /etc/nginx/
   sites-enabled/
sudo systemctl restart nginx
```

6. **Open Firewall Port**:

```
sudo ufw allow 443/tcp
sudo ufw status
```

7. **Restart Unbound**:

```
sudo systemctl restart unbound
sudo systemctl enable unbound
```

8. **Verify**:

```
dig example.com @your-vps-ip
```

Ensure responses come from your VPS.

3.6.3 Steps for Local Machine (Stub Resolver)

1. **Install Unbound Locally**:

```
sudo apt install unbound -y
```

2. **Configure as Stub Resolver**:

```
sudo nano /etc/unbound/unbound.conf
```

Add:

```
server:
    do-ip4: yes
    do-ip6: no
    interface: 127.0.0.1
    access-control: 127.0.0.0/8 allow
stub-zone:
    name: "."
    stub-addr: your-vps-ip@443
    stub-tls-upstream: yes
```

3. **Set Local Resolver**:

```
sudo nano /etc/resolv.conf
```

Add:

```
nameserver 127.0.0.1
```

Make immutable:

```
sudo chattr +i /etc/resolv.conf
```

4. **Restart Unbound**:

```
sudo systemctl restart unbound
sudo systemctl enable unbound
```

5. **Configure Firefox for DoH**:

   - Go to `Settings > General > Network Settings > Settings`.
   - Enable `Enable DNS over HTTPS`.
   - Select `Custom` and enter `https://dns.yourdomain.com/dns-query`.
   - Alternatively, in `about:config`, set:

```
network.trr.mode = 3
network.trr.uri = https://dns.yourdomain.com/dns-query
```

6. **Verify**:

```
dig example.com @127.0.0.1
```

Check https://ipleak.net to confirm only your VPS's DNS is used.

3.6.4 Advantages

- **Maximum Privacy and Control**: Your VPS handles recursive resolution, ensuring no third-party logging. DoH encrypts client-VPS queries.
- **Location Anonymity**: VPS in a privacy-friendly country (e.g., Panama) hides your location from authoritative servers.
- **Security**: QNAME minimization and DNSSEC reduce tracking and spoofing risks.

3.6.5 Disadvantages

- **Complexity and Cost**: Requires VPS ($5–10/month), domain, and maintenance (e.g., root hints, certificates).
- **Latency**: Queries route to VPS, then recursively resolved, increasing delay.
- **VPS Trust**: If the VPS provider logs traffic, privacy could be compromised (mitigate with no-logs providers).

3.7 Preventing DNS Leaks in VPN and SSH Tunneling

DNS leaks can undermine VPN or SSH tunneling privacy if queries bypass the secure channel. This section covers leak prevention for commercial VPNs, self-hosted VPNs, and SSH tunneling, ensuring all DNS traffic routes correctly.

3.7.1 Commercial VPNs

Reputable VPN providers in 2025 (e.g., Proton VPN, Mullvad, NordVPN, ExpressVPN; see Chapter 4) offer built-in DNS leak protection:

- **Proton VPN**: Uses custom DNS servers (e.g., `10.0.0.1`) routed through the VPN tunnel. Enable kill switch in the app to block traffic if the VPN drops. Supports Secure Core for multi-hop routing.
- **Mullvad**: Routes DNS through its servers with DAITA (obfuscation to evade detection). Enable kill switch and set DNS to Mullvad's servers (`194.68.28.12`).
- **NordVPN/ExpressVPN**: Provide proprietary DNS servers and kill switches. Configure apps to use provider DNS only.
- **Steps**: In VPN app settings, enable kill switch and select provider DNS. For manual setups (e.g., OpenVPN), add `dhcp-option DNS <provider-dns-ip>` to the .ovpn file.
- **Verify**: Check https://ipleak.net to ensure only VPN provider DNS servers appear.

3.7.2 Self-Hosted VPNs

For self-hosted VPNs on a VPS (Chapter 4):

- **OpenVPN**: Add to server config (`/etc/openvpn/server.conf`):

```
push "dhcp-option DNS 10.8.0.1"
```

  Use your VPS's Unbound resolver (Solution 5) as the DNS server.

- **WireGuard**: Add to client config (`wg0.conf`):

```
DNS = 10.8.0.1
```

  Ensure the VPS runs Unbound (Solution 5) or another DNS resolver.

- **Verify**: After connecting, run `dig example.com` and check https://ipleak.net to confirm only your VPS's DNS is used.

3.7.3 SSH Tunneling

SSH tunneling (e.g., `ssh -D 1080 vpsuser@your-vps-ip` for SOCKS5 proxy; Chapter 2) doesn't tunnel DNS queries by default, causing leaks to your ISP's DNS servers.

- **Browser Fix**: Configure your browser to route DNS through the SOCKS proxy:

  - Firefox: Set `network.proxy.socks_remote_dns = true` in `about:config` (see Section 3.8).
  - Chrome: Use `--proxy-server="socks5://localhost:1080"` flag or extensions like Proxy Switcher to enable DNS over SOCKS.

- **System-Wide Fix**: Use Unbound locally (Solutions 3–5) and route its queries through the SSH tunnel:
  - Install `proxychains`:

```
sudo apt install proxychains
```

  - Configure `/etc/proxychains.conf`:

```
socks5 127.0.0.1 1080
```

  - Run Unbound via `proxychains`:

```
proxychains unbound
```

  - Alternatively, use `ssh -L 53:your-vps-ip:53` to tunnel DNS port to a VPS resolver, then set `/etc/resolv.conf` to `nameserver 127.0.0.1`.

- **Verify**: Check https://dnsleaktest.com to ensure only your VPS or tunnel DNS is used.

3.7.4 General Tips

- Always use DoH or DoT for encrypted queries (Solutions 3 and 5).
- Disable IPv6 to prevent leaks (`net.ipv6.conf.all.disable_ipv6=1` in `/etc/sysctl.conf`).
- Test with https://ipleak.net after setup to confirm no ISP DNS servers appear.

3.8 Fixing Browser DNS Leaks

For *seventeen years*, Mozilla ignored DNS leaks in SOCKS proxies (https://bugzilla.mozilla.org/show_bug.cgi?id=1741375), only setting `network.proxy.socks_remote_dns` to `true` by default in 2024—*apparently, privacy wasn't a priority while we were all stuck on dial-up.* To ensure no leaks:

1. **Enable Proxy DNS (Firefox)**:
   - Go to `Settings > General > Network Settings > Settings`.
   - Select `Manual proxy configuration`, set your SOCKS5 proxy (e.g., `localhost:1080`).
   - Check `Proxy DNS when using SOCKS v5`.
   - Alternatively, in `about:config`, set:

```
network.proxy.socks_remote_dns = true
```

2. **Enable DoH (Firefox)**:
   - Go to `Settings > General > Network Settings > Settings`.
   - Enable `Enable DNS over HTTPS`, select `Max Protection` or `Custom` (e.g., `https://dns.yourdomain.com/dns-query` for Solution 5).
3. **Disable WebRTC** (potential leak source):

```
about:config
media.peerconnection.enabled = false
```

4. **Chrome WebRTC Disable**:
   - Go to `chrome://settings/privacy`, disable `WebRTC`.
   - Alternatively, install uBlock Origin and block WebRTC.
5. **Verify**: Test at https://ipleak.net. Avoid extensions like Port Authority, which can cause leaks with SOCKS5 proxies due to CNAME lookup issues.

3.9 Important Notes

- **Testing**: Always verify setups with https://dnsleaktest.com or https://ipleak.net.
- **Firefox Quirks**: Even with 2024 fixes, system proxy settings can leak DNS (Bug 1470411). Use manual proxy settings for reliability.
- **Maintenance**: Update root hints monthly for Solutions 4 and 5. Set up a cron job:

```
sudo crontab -e
```

  Add:

```
0 3 1 * * wget -O /var/lib/unbound/root.hints https://www.
   internic.net/domain/named.root && systemctl restart
   unbound
```

  This runs on the 1st of every month at 3 AM. For Certbot renewal (Solution 5), Certbot installs a systemd timer or cron job automatically; test with `sudo certbot renew --dry-run`. If needed, add `0 0 * * * /usr/bin/certbot renew --quiet` to crontab.

- **Alternative**: Use `dnscrypt-proxy` for simpler DoH setups:

```
sudo apt install dnscrypt-proxy
```

3.10 Troubleshooting

- **DNS Not Resolving**:
  - Check `unbound.conf` for syntax errors: `unbound-checkconf`.
  - Verify firewall: `sudo ufw status`.
  - Test Unbound: `dig example.com @127.0.0.1`.
- **Leaks Detected**:
  - Ensure `resolv.conf` points to `127.0.0.1`.
  - Disable conflicting extensions (e.g., uBlock Origin's CNAME uncloaking).
  - Test with https://ipleak.net.
- **DoH Issues**: Verify Nginx and certificates. Check logs: `sudo journalctl -u nginx`.

Chapter 4

Using a VPN for Encrypted Connections

4.1 What is a VPN and When to Use It

A VPN encrypts your internet traffic and routes it through a server, masking your real IP address. Unlike Tor (Chapter 1), which prioritizes anonymity, VPNs focus on encryption and speed, ideal for securing connections or bypassing geo-restrictions.

4.1.1 When to Use a VPN

Use a VPN for:

- **Encrypted connections**: Securing data on public Wi-Fi.
- **Geo-restriction bypassing**: Accessing region-locked content (e.g., Netflix).
- **ISP privacy**: Hiding browsing from your ISP.

4.1.2 Against What Adversaries

VPNs protect against:

- **ISPs and local networks**: They can't see your browsing data.
- **Websites**: They see the VPN's IP.

They do not protect against:

- **VPN providers**: They see your IP unless no-logs.
- **Global adversaries**: Traffic analysis risks.
- **Leaks**: Misconfigurations can expose IPs/DNS (Chapter 3).

Note for Beginners: Choose a reputable VPN with a no-logs policy. This chapter includes audited providers and self-hosted options.

4.2 Top Paid VPN Providers

The following VPNs (2025) have audited no-logs policies and no known government collusion:

- **Proton VPN** (Switzerland, outside Five/Nine/Fourteen Eyes alliances): Audited no-logs, open-source apps, diskless servers, WireGuard/OpenVPN, Secure Core for multi-hop. Free tier (limited). 30-day money-back.
- **Mullvad** (Sweden, part of Fourteen Eyes alliance): Audited no-logs, anonymous sign-up, open-source, DAITA for traffic obfuscation. 30-day money-back.
- **NordVPN** (Panama, outside Five/Nine/Fourteen Eyes alliances): Audited no-logs, up to 1 Gbps, diskless servers. 30-day money-back.
- **ExpressVPN** (British Virgin Islands, outside Five/Nine/Fourteen Eyes alliances): 23 audits, open-source Lightway. 30-day money-back.

4.3 Configuring a VPN on Linux

4.3.1 Prerequisites

- A Linux computer.
- A VPN account (e.g., Proton VPN, Mullvad).

4.3.2 Steps

1. **Install and Connect Your VPN**:
   - **For Ubuntu/Debian**:

```
sudo apt update
sudo apt install openvpn
sudo openvpn --config your-vpn-config.ovpn
```

   Or GUI client:

```
sudo apt install protonvpn
```

   - **For Fedora**:

```
sudo dnf install openvpn
sudo openvpn --config your-vpn-config.ovpn
```

   - **For Gentoo**:

```
sudo emerge -av net-vpn/openvpn
sudo openvpn --config your-vpn-config.ovpn
```

   OpenRC:

```
sudo /etc/init.d/openvpn start
sudo rc-update add openvpn default
```

Systemd:

```
sudo systemctl start openvpn@your-config.service
sudo systemctl enable openvpn@your-config.service
```

- Disable IPv6 to prevent leaks:

```
sudo nano /etc/sysctl.conf
```

Add:

```
net.ipv6.conf.all.disable_ipv6=1
```

Apply:

```
sudo sysctl -p
```

- Verify:

```
curl https://ifconfig.me
```

2. **Verify the Setup**:
   - Visit https://ipleak.net for leaks.
   - Test streaming (e.g., Netflix).

4.4 Configuring a VPN on Windows

4.4.1 Prerequisites

- A Windows computer.
- A VPN account.

4.4.2 Steps

1. **Install and Connect Your VPN**:
   - Download your VPN's client or OpenVPN GUI from https://openvpn.net/community-downloads/.
   - Install and connect.
   - Disable IPv6:
     - Go to `Settings > Network & Internet > Ethernet` (or `Wi-Fi`).
     - Disable `IPv6` in adapter settings.
   - Verify:

```
Invoke-WebRequest -Uri https://ifconfig.me
```

2. **Verify the Setup**:
   - Visit https://ipleak.net.
   - Test streaming services.

4.5 Configuring a VPN on Mobile (Android/iOS)

This section provides detailed instructions for setting up VPN on mobile devices using top providers (Proton VPN, Mullvad, NordVPN, ExpressVPN).

4.5.1 Prerequisites

- An Android or iOS device.
- Internet access and a VPN account from one of the top providers.

4.5.2 Steps

1. **Install VPN Apps**:
   - **Android**:
     - For Proton VPN: Install from Google Play or download APK from https://protonvpn.com/download.
     - For Mullvad: Install from Google Play or download APK from https://mullvad.net/download.
     - For NordVPN: Install from Google Play or download APK from https://nordvpn.com/download.
     - For ExpressVPN: Install from Google Play or download APK from https://www.expressvpn.com/download.
   - **iOS**:
     - For Proton VPN: Install from the App Store.
     - For Mullvad: Install from the App Store.
     - For NordVPN: Install from the App Store.
     - For ExpressVPN: Install from the App Store.
2. **Connect to VPN**:
   - **Proton VPN (Android/iOS)**:
     - Open the app and log in with your credentials.
     - Select a server location and tap "Connect".
     - Enable kill switch in settings to prevent leaks.
     - For advanced: Enable Secure Core for multi-hop routing.
   - **Mullvad (Android/iOS)**:
     - Open the app and enter your account number (no login required).
     - Select a server and tap "Connect".
     - Enable WireGuard in settings for better speed.
     - For obfuscation: Enable DAITA in settings if needed.
   - **NordVPN (Android/iOS)**:
     - Open the app and log in.
     - Tap "Quick Connect" or select a server.

  - Enable kill switch and obfuscated servers if in restricted areas.
- **ExpressVPN (Android/iOS)**:
  - Open the app and log in.
  - Tap the connect button or choose a location.
  - Enable kill switch (Network Protection) in settings.
  - Use Lightway protocol for optimal performance.

3. **Verify the Setup**:
   - Visit https://ipleak.net in your mobile browser to check for leaks (IP, DNS, WebRTC).
   - Test your public IP with a site like https://ifconfig.me.
   - Ensure IPv6 is disabled if supported in app settings to prevent leaks.
   - Test geo-restricted content (e.g., Netflix) to confirm bypassing works.

4.6 Setting Up a Self-Hosted VPN on a VPS

For users avoiding paid VPNs, you can set up OpenVPN or WireGuard on a VPS (Chapter 2).

4.6.1 Prerequisites

- A VPS (e.g., DigitalOcean, $5–10/month).
- A Linux computer.

4.6.2 Steps (OpenVPN on Ubuntu VPS)

1. **Set Up the VPS**:
   - Create an Ubuntu 22.04 droplet (Chapter 2).
   - Note the public IP and SSH credentials.
2. **Install OpenVPN**:

```
ssh root@your-vps-ip
apt update
apt install openvpn easy-rsa
make-cadir ~/openvpn-ca
cd ~/openvpn-ca
nano vars
```

   Adjust variables (e.g., `KEY_COUNTRY`, `KEY_NAME`).

```
source vars
./clean-all
./build-ca
./build-key-server server
./build-dh
openvpn --genkey --secret keys/ta.key
```

3. **Configure OpenVPN Server**:

```
gunzip -c /usr/share/doc/openvpn/examples/sample-config-files
   /server.conf.gz > /etc/openvpn/server.conf
nano /etc/openvpn/server.conf
```

Set `ca`, `cert`, `key`, `dh`, `tls-auth`, `push "redirect-gateway def1"`, `cipher ChaCha20-Poly1305`. Enable IP forwarding:

```
nano /etc/sysctl.conf
```

Uncomment `net.ipv4.ip_forward=1`. Add:

```
net.ipv6.conf.all.disable_ipv6=1
```

Apply:

```
sysctl -p
```

4. **Set Up Firewall**:

```
apt install ufw
ufw allow 1194/udp
ufw allow OpenSSH
ufw enable
```

For IPv6 (if enabled):

```
ip6tables -A INPUT -p udp --dport 1194 -j ACCEPT
ip6tables -A INPUT -p tcp --dport 22 -j ACCEPT
ip6tables -P INPUT DROP
ip6tables-save > /etc/iptables/ip6tables.rules
```

5. **Start OpenVPN**:

```
systemctl start openvpn@server
systemctl enable openvpn@server
```

6. **Generate Client Config**:

```
cd ~/openvpn-ca
./build-key client
nano client.ovpn
```

Add:

```
client
dev tun
proto udp
remote your-vps-ip 1194
resolv-retry infinite
nobind
persist-key
persist-tun
```

```
ca ca.crt
cert client.crt
key client.key
tls-auth ta.key 1
cipher ChaCha20-Poly1305
verb 3
```

Copy files to your device via SCP.

7. **Connect from Your Device**:
   - **Linux**:

```
sudo apt install openvpn
sudo openvpn --config client.ovpn
```

   - **Windows**: Install OpenVPN GUI, import `client.ovpn`, connect.

8. **Verify**:

```
curl https://ifconfig.me
```

Check https://ipleak.net.

4.6.3 Steps (WireGuard Alternative)

1. **Install WireGuard**:

```
apt update
apt install wireguard
```

2. **Generate Keys**:

```
wg genkey | tee /etc/wireguard/privatekey | wg pubkey > /etc/
   wireguard/publickey
```

3. **Configure WireGuard Server**:

```
nano /etc/wireguard/wg0.conf
```

Add:

```
[Interface]
PrivateKey = <server-private-key>
Address = 10.0.0.1/24
ListenPort = 51820
[Peer]
PublicKey = <client-public-key>
AllowedIPs = 10.0.0.2/32
```

Enable IP forwarding:

```
nano /etc/sysctl.conf
```

Uncomment `net.ipv4.ip_forward=1`. Add:

```
net.ipv6.conf.all.disable_ipv6=1
```

Apply:

```
sysctl -p
```

4. **Start WireGuard**:

```
wg-quick up wg0
systemctl enable wg-quick@wg0
```

5. **Client Config**:

```
[Interface]
PrivateKey = <client-private-key>
Address = 10.0.0.2/24
DNS = 8.8.8.8
[Peer]
PublicKey = <server-public-key>
Endpoint = your-vps-ip:51820
AllowedIPs = 0.0.0.0/0
```

Save as `wg0.conf`.

6. **Connect from Your Device**:

   - **Linux**:

```
sudo apt install wireguard
wg-quick up wg0
```

   - **Windows**: Install WireGuard from https://www.wireguard.com, import `wg0.conf`, connect.

7. **Verify**:

```
curl https://ifconfig.me
```

Check https://ipleak.net.

4.6.4 Advantages of Self-Hosted VPN

- **Control**: You manage the server, ensuring no logging.
- **Cost**: $5–10/month, potentially cheaper than premium VPNs.
- **Privacy**: No third-party provider involvement.
- **Flexibility**: Customize protocols and server location.

4.6.5 Disadvantages

- **Complexity**: Requires technical skills.
- **Maintenance**: You handle updates and security.
- **Limited Locations**: One VPS vs. global servers.
- **Streaming**: Poor for bypassing geo-restrictions.

4.7 Free VPN Services

Free VPNs often log data or sell user information, posing privacy risks. Avoid them for sensitive tasks. Among top providers:

- **Proton VPN**: Free tier with unlimited data, 5 locations, no logs, but one device and no streaming.
- **Mullvad, NordVPN, ExpressVPN**: No free tiers, but 30-day money-back guarantees.

Use Proton VPN's free tier for basic privacy; prefer paid or self-hosted VPNs for critical needs.

4.8 Important Notes

- **Provider Choice**: Choose audited VPNs (e.g., Proton VPN, Mullvad).
- **Speed**: Use WireGuard for better performance.
- **Security**: Enable kill switches to prevent leaks. Disable IPv6 to avoid leaks.
- **Streaming**: Paid VPNs excel at unblocking content.

4.9 Troubleshooting

- **VPN Not Connecting**:
  - Linux: Check logs (`sudo journalctl -u openvpn`; Gentoo OpenRC: `/etc/init.d/openvpn status`).
  - Windows: Verify client credentials.
  - VPS: Check firewall (`ufw status`) and config.
- **Leaks Detected**:
  - Linux: Check `/etc/resolv.conf`.
  - Windows: Enable kill switch; check DNS (Chapter 3).
  - Use https://ipleak.net.
- **Slow Connection**: Try closer server or WireGuard.

Chapter 5

Configuring VPN-over-Tor on Linux and Windows

5.1 What is VPN-over-Tor and When to Use It

As introduced in Chapter 1 and Chapter 4, **VPN-over-Tor** routes traffic through Tor first, then a VPN, hiding your IP from the VPN provider. This differs from Tor-over-VPN (Chapter 6). In VPN-over-Tor:

- Your device → Tor network → VPN server → Internet.

5.1.1 When to Use VPN-over-Tor

Use when:

- **Hiding from VPN provider**: You want the VPN to see a Tor exit node.
- **Censorship resistance**: VPNs are monitored, but Tor is less restricted.
- **High anonymity**: For sensitive activities.

5.1.2 Against What Adversaries

See Chapter 1 and Chapter 4 for details. VPN-over-Tor protects:

- **ISPs**: See only Tor usage.
- **VPN providers**: See a Tor exit node's IP.
- **Websites**: See the VPN's IP.

Limitations include global adversaries and exit node risks. **Note for Beginners**: This setup is complex and slow. Confirm VPN support (e.g., Proton VPN's Secure Core, Mullvad's DAITA).

5.2 Configuring VPN-over-Tor on Linux

5.2.1 Prerequisites

- A Linux computer.
- A VPN account supporting VPN-over-Tor.

5.2.2 Steps

1. **Install Tor**:

   - **For Ubuntu/Debian**:

```
sudo apt update
sudo apt install tor
sudo systemctl start tor
sudo systemctl enable tor
```

   - **For Fedora**:

```
sudo dnf install tor
sudo systemctl start tor
sudo systemctl enable tor
```

   - **For Gentoo**:

```
sudo emerge -av net-vpn/tor
```

   - OpenRC:

```
sudo /etc/init.d/tor start
sudo rc-update add tor default
```

   - Systemd:

```
sudo systemctl start tor
sudo systemctl enable tor
```

2. **Configure Tor as a Proxy**:

```
sudo nano /etc/tor/torrc
```

   Add:

```
VirtualAddrNetworkIPv4 10.192.0.0/10
AutomapHostsOnResolve 1
TransPort 9040
DNSPort 9053
```

   Restart:

   - Systemd:

```
sudo systemctl restart tor
```

- Gentoo OpenRC:

```
sudo /etc/init.d/tor restart
```

3. **Set Up Firewall Rules**:

```
sudo iptables -F
sudo iptables -A OUTPUT -p tcp --dport 80 -j REDIRECT --to-ports 9040
sudo iptables -A OUTPUT -p tcp --dport 443 -j REDIRECT --to-ports 9040
sudo iptables -A OUTPUT -p udp --dport 53 -j REDIRECT --to-ports 9053
sudo ip6tables -F
sudo ip6tables -P OUTPUT DROP
```

Save:

- Ubuntu/Debian:

```
sudo iptables-save > /etc/iptables/rules.v4
sudo ip6tables-save > /etc/iptables/rules.v6
```

- Fedora:

```
sudo iptables-save > /etc/sysconfig/iptables
sudo ip6tables-save > /etc/sysconfig/ip6tables
```

- Gentoo:

```
sudo iptables-save > /etc/iptables/iptables.rules
sudo ip6tables-save > /etc/iptables/ip6tables.rules
sudo /etc/init.d/iptables save
sudo rc-update add iptables default
```

Systemd:

```
sudo systemctl enable iptables-restore
sudo systemctl enable ip6tables-restore
```

4. **Set Up Your VPN**:

- Install OpenVPN:
  - Ubuntu/Debian:

```
sudo apt install openvpn
```

  - Fedora:

```
sudo dnf install openvpn
```

  - Gentoo:

```
sudo emerge -av net-vpn/openvpn
```

- Edit `.ovpn` file:

```
socks-proxy 127.0.0.1 9050
socks-proxy-retry
cipher ChaCha20-Poly1305
```

- Connect:

```
sudo openvpn --config your-vpn-config.ovpn
```

5. **Verify the Setup**:

- Visit https://check.torproject.org.
- Check IP:

```
curl https://ifconfig.me
```

- Visit https://ipleak.net.

5.3 Configuring VPN-over-Tor on Windows

5.3.1 Prerequisites

- A Windows computer.
- A VPN account supporting VPN-over-Tor.

5.3.2 Steps

1. **Install Tor Browser**:

- Download from https://www.torproject.org.
- Install and keep open.

2. **Install Your VPN Client**:

- Download your VPN's client or OpenVPN GUI from https://openvpn.net/community-downloads/.
- Install.

3. **Configure VPN for Tor (OpenVPN)**:

- Edit `.ovpn` file:

```
socks-proxy 127.0.0.1 9050
socks-proxy-retry
cipher ChaCha20-Poly1305
```

- Import into OpenVPN GUI and connect.

4. **Use Proxifier for Non-OpenVPN VPNs**:

- Download from https://www.proxifier.com (or use free alternatives like SocksCap64).
- Add proxy: `127.0.0.1:9050`, SOCKS5.
- Route VPN client through Tor.

5. **Verify the Setup**:
   - Visit https://check.torproject.org.
   - Check IP:

```
Invoke-WebRequest -Uri https://ifconfig.me
```

   - Visit https://ipleak.net.

5.4 Important Notes

- **Speed**: Slower due to Tor's routing.
- **VPN Support**: Confirm VPN supports VPN-over-Tor (e.g., Proton VPN, Mullvad).
- **Security**: Test for leaks with https://ipleak.net. Use DoH (Chapter 3).
- **Alternative**: Try Tor-over-VPN (Chapter 6).

5.5 Troubleshooting

- **Tor Not Connecting**: Check status (`systemctl status tor`; Gentoo OpenRC: `/etc/init.d/tor status`). Enable bridges.
- **VPN Not Connecting**: Test without Tor; check `.ovpn` or Proxifier.
- **Leaks Detected**: Verify `iptables`/Proxifier settings. Check DNS (Chapter 3).

Chapter 6

Configuring Tor-over-VPN on Linux and Windows

6.1 What is Tor-over-VPN and When to Use It

In **Tor-over-VPN**, you connect to a VPN first, then use the Tor network (via Tor Browser). This hides Tor usage from your ISP, unlike VPN-over-Tor (Chapter 5). In Tor-over-VPN:

- Your device → VPN server → Tor network → Internet.

6.1.1 When to Use Tor-over-VPN

Use when:

- **Hiding Tor usage**: Your ISP blocks/monitors Tor.
- **Bypassing restrictions**: Tor is blocked, but VPNs connect.
- **Simpler setup**: Easier than VPN-over-Tor.

6.1.2 Against What Adversaries

Tor-over-VPN protects:

- **ISPs**: See only VPN traffic.
- **Websites**: See a Tor exit node's IP.

It does not protect against VPN providers seeing your IP. **Note for Beginners**: Tor-over-VPN is simpler, ideal for hiding Tor usage. Use audited VPNs (Chapter 4).

6.2 Configuring Tor-over-VPN on Linux

6.2.1 Prerequisites

- A Linux computer.
- A VPN account.

6.2.2 Steps

1. **Install and Connect Your VPN**:

   - **For Ubuntu/Debian**:

```
sudo apt update
sudo apt install openvpn
sudo openvpn --config your-vpn-config.ovpn
```

   Or GUI client:

```
sudo apt install protonvpn
```

   - **For Fedora**:

```
sudo dnf install openvpn
sudo openvpn --config your-vpn-config.ovpn
```

   - **For Gentoo**:

```
sudo emerge -av net-vpn/openvpn
sudo openvpn --config your-vpn-config.ovpn
```

   OpenRC:

```
sudo /etc/init.d/openvpn start
sudo rc-update add openvpn default
```

   Systemd:

```
sudo systemctl start openvpn@your-config.service
sudo systemctl enable openvpn@your-config.service
```

   - Disable IPv6:

```
sudo nano /etc/sysctl.conf
```

   Add:

```
net.ipv6.conf.all.disable_ipv6=1
```

   Apply:

```
sudo sysctl -p
```

   - Verify:

```
curl https://ifconfig.me
```

2. **Install Tor Browser**:

```
tar -xvf tor-browser-linux64-*.tar.xz
mv tor-browser_en-US ~/tor-browser
cd ~/tor-browser
./start-tor-browser.desktop
```

3. **Connect to Tor**:
   - Open Tor Browser and click "Connect".
   - Enable bridges if needed (https://bridges.torproject.org).
4. **Verify the Setup**:
   - Visit https://check.torproject.org.
   - Check IP:

```
curl https://ifconfig.me
```

   - Visit https://ipleak.net.

6.3 Configuring Tor-over-VPN on Windows

6.3.1 Prerequisites

- A Windows computer.
- A VPN account.

6.3.2 Steps

1. **Install and Connect Your VPN**:
   - Download your VPN's client or OpenVPN GUI from https://openvpn.net/community-downloads/.
   - Install and connect.
   - Disable IPv6:
     - Go to `Settings > Network & Internet > Ethernet` (or `Wi-Fi`).
     - Disable `IPv6` in adapter settings.
   - Verify:

```
Invoke-WebRequest -Uri https://ifconfig.me
```

2. **Install Tor Browser**:
   - Download from https://www.torproject.org.
   - Install and launch.
3. **Connect to Tor**:
   - Click "Connect" in Tor Browser.
   - Enable bridges if needed.
4. **Verify the Setup**:
   - Visit https://check.torproject.org.

- Check IP:

```
Invoke-WebRequest -Uri https://ifconfig.me
```

- Visit https://ipleak.net.

6.4 Important Notes

- **Speed**: Slower due to Tor's routing.
- **VPN Support**: Ensure VPN allows Tor traffic (e.g., Proton VPN, Mullvad).
- **Security**: Test for leaks with https://ipleak.net. Use DoH (Chapter 3).
- **Alternative**: Use Tor alone (Chapter 1).

6.5 Troubleshooting

- **VPN Not Connecting**: Check logs or settings.
- **Tor Not Connecting**: Verify Tor Browser or service status; enable bridges.
- **Leaks Detected**: Check DNS (Chapter 3); use https://ipleak.net.

Chapter 7

Comparing Privacy Solutions

This chapter provides a tabular comparison of the privacy solutions covered: Tor alone (Chapter 1), VPN alone (Chapter 4), VPN-over-Tor (Chapter 5), and Tor-over-VPN (Chapter 6). It includes a table summarizing key differences.

7.1 Comparison Table

The table below summarizes the key differences between the privacy solutions, focusing on what entities see your real IP, speed impact, setup complexity, and best use cases.

Feature	**Tor Alone**	**VPN Alone**	**VPN-over-Tor**	**Tor-over-VPN**
Hides from ISP	Partial (sees Tor)	Partial (sees VPN)	Partial (sees Tor)	Partial (sees VPN)
Hides from VPN Provider	N/A	No	Yes	No
Hides from Websites	Yes (sees Tor exit IP)	Yes (sees VPN IP)	Yes (sees VPN IP)	Yes (sees Tor exit IP)
Speed Impact	High	Low	Very High	High
Ease of Setup	Medium	Easy	Hard	Medium
Best For	Anonymity, .onion sites	Encryption, streaming	Max anonymity	Hiding Tor use

Table 7.1: Comparison of privacy solutions based on visibility, performance, and use cases.

7.2 Recommendations

- **Tor Alone**: Ideal for accessing `.onion` sites or when anonymity is critical. Use HTTPS and disable WebRTC to avoid exit node risks (Chapter 1).
- **VPN Alone**: Best for fast, encrypted connections or bypassing geo-restrictions. Choose audited providers like Proton VPN or Mullvad (Chapter 4).
- **VPN-over-Tor**: Use for maximum anonymity, hiding your IP from VPN providers. Requires technical setup and VPN support (Chapter 5).

- **Tor-over-VPN**: Simpler setup to hide Tor usage from ISPs. Good for restrictive regions (Chapter 6).
- **DNS Considerations**: Always secure DNS queries to prevent leaks (Chapter 3). Use DoH or a self-hosted Unbound server for best privacy.
- **2025 Trends**: Consider VPNs with multi-hop (e.g., Proton VPN's Secure Core) or traffic obfuscation (e.g., Mullvad's DAITA). Enable Encrypted Client Hello (ECH) for DNS setups to future-proof privacy.

Chapter 8

Browser Rebellion: Privacy Hacks

Ah, browsers—the digital gateways where your every click is a potential privacy leak. In this chapter, we dive into making Firefox and Chrome (or Chromium on Linux) as privacy-respecting as possible. We'll tweak settings, suggest forks that do the heavy lifting out-of-the-box, and even cover mobile options. But first, a bit of snark: Privacy browsers are like that friend who promises to keep your secrets but ends up spilling them for "features." And let's not forget, back in 2013, before the hype of LibreWolf or Brave, there was CyberDragon Browser—yours truly's pioneering effort, the first real privacy-focused browser that got nods from Softpedia and tech blogs for stripping trackers and bolstering anonymity. It paved the way, but hey, imitation is the sincerest form of flattery.

Now, on to the snark: Firefox has been caught slipping with telemetry creep and Pocket integration, quietly eroding privacy under the guise of "user experience." Chromium? Google's playground, where features like FLoC (now Privacy Sandbox) pretend to protect you while feeding ad data back to the mothership. And Brave? Oh, the darling of the hype machine—blocking ads only to replace them with their own, all while dangling BAT tokens that users can technically withdraw (via Uphold or Solana wallets in 2025), but good luck navigating the conversion hurdles without feeling like it's more crypto gimmick than genuine reward. Factual? Yes. Insulting? Nah, just calling it like it is.

8.1 Privacy Tweaks for Firefox

Firefox is a solid base for privacy, but it needs tweaks via `about:config` to shine. These apply to both Linux and Windows—access `about:config` by typing it in the address bar and confirming you know what you're doing.

8.1.1 Key about:config Settings

Enter these in the search bar and toggle as noted:

- **privacy.resistFingerprinting = true**: Reduces fingerprinting by spoofing user agent, screen size, etc.
- **privacy.trackingprotection.enabled = true**: Enables Enhanced Tracking Protection.
- **network.cookie.cookieBehavior = 4**: Blocks third-party cookies.

- **beacon.enabled = false**: Disables ping requests.
- **media.peerconnection.enabled = false**: Disables WebRTC to prevent IP leaks.
- **telemetry.enabled = false**: Turns off Mozilla's data collection (snark: because who needs more telemetry in 2025?).
- **extensions.pocket.enabled = false**: Disables Pocket integration.

Restart Firefox after changes. On Linux, install via package manager (`sudo apt install firefox` on Debian-based); on Windows, download from mozilla.org.

8.1.2 Privacy-Respecting Firefox Forks

Why tweak when forks do it for you?

- **LibreWolf**: Top pick—pre-configured for privacy, removes telemetry, adds uBlock Origin. Out-of-the-box better than vanilla Firefox. Install via flatpak on Linux (`flatpak install flathub io.gitlab.librewolf-community`), or EXE on Windows from librewolf.net.
- **Mullvad Browser**: Tor-integrated for max anonymity, but slower for daily use.
- **Waterfox**: Focuses on speed and privacy, fewer telemetry hooks.

These forks offer better defaults without the erosion seen in Firefox's history, like forced telemetry opt-ins.

8.2 Privacy Tweaks for Chrome/Chromium

Chrome/Chromium is Google's turf, so privacy tweaks are command-line heavy. Use flags to disable tracking. On Linux, prefer Chromium; on Windows, Chrome.

8.2.1 Key Command-Line Flags

Launch with these flags (add to shortcut or terminal):

- **–disable-features=PrivacySandboxAdsAPIs**: Blocks Privacy Sandbox (Google's "privacy" ad tech).
- **–no-experiments**: Disables A/B testing and experiments.
- **–disable-sync**: Prevents Google sync.
- **–disable-google-logging**: Reduces logging to Google.
- **–disable-site-isolation-trials**: For performance, but test privacy impact.
- **–incognito**: Starts in private mode (but not permanent privacy).

On Linux: `chromium --disable-features=PrivacySandboxAdsAPIs`. On Windows: Right-click shortcut ¿ Properties ¿ Append flags to Target field.

8.2.2 Privacy-Respecting Chromium Forks

Ditch vanilla for forks:

- **Ungoogled-Chromium**: Best choice—strips all Google ties, enhances privacy. Auto-updates limited, but secure. Linux: Flatpak or build from source; Windows: Download binaries from woolyss.com.
- **Iridium**: Privacy-focused, removes telemetry.
- **Bromite**: Mobile-oriented but available; for desktop, stick to Ungoogled.

These counter Chromium's privacy erosions, like the 20-year history flaw fixed only in 2025 or Manifest V3 crippling ad-blockers.

8.3 Privacy-Respecting Mobile Browsers

Mobile privacy is trickier with app ecosystems.

- **Android**: Mull (Firefox fork with privacy tweaks), Cromite (Ungoogled-Chromium fork), DuckDuckGo Browser (simple, tracker-blocking).
- **iOS**: DuckDuckGo Browser (strong tracking protection), Onion Browser (Tor-based), Brave (but see our jab earlier).

Install from F-Droid (Android) or App Store (iOS) for best privacy.

8.4 Conclusion

Browsers have come far since CyberDragon's 2013 debut, but privacy erosion continues—Firefox's TOS flip-flops and Chromium's ad agendas prove it. Stick to tweaks or forks, and remember: in the grid, privacy is rebellion.

Chapter 9

Search Revolt: Privacy Engines

Search engines: the all-seeing oracles of the web, or just glorified data hoovers? In this chapter, we'll arm beginners with solid alternatives to Google—the ultimate privacy invader—and guide advanced users on self-hosting meta-search engines like SearXNG. Because nothing says "rebel" like searching without leaving a digital breadcrumb trail. Snark alert: Google might as well be called "Gotcha"—it's not just searching you; it's searching *through* you. And while we're at it, those "privacy-focused" engines? Some are like that shady alley cat—promising independence but still peeking in your trash.

9.1 How Google Tracks Your Searches

Google's motto might as well be "Don't be evil... unless it's profitable." Here's how it stalks your searches in 2025:

9.1.1 On Desktop

- **Cookies and Identifiers**: Persistent cookies and device fingerprinting (user agent, screen resolution) track across sessions.
- **IP Address and Location**: Logs your IP for geo-targeted results and ads.
- **Google Account Sync**: If logged in, ties searches to your profile, history, and cross-device data.
- **Telemetry and Analytics**: Sends usage data, crash reports, and "anonymous" metrics back home.
- **Privacy Sandbox**: Replaces cookies with "Topics" API, grouping you into ad cohorts—snark: because tracking en masse is "better" privacy?
- **Search History**: Stores queries indefinitely unless you opt out (and even then, it's spotty).

9.1.2 On Android

- **Pre-Login Tracking**: Even before signing in, apps like Play Services ping Google with device ID, ad ID, and app usage.

- **Google Play Services**: Constant background tracking via location, Wi-Fi scanning, and app interactions.
- **Ad ID and Device Fingerprinting**: Unique IDs tie searches to your device for personalized ads.
- **Integrated Search**: Chrome on Android syncs with Google apps, sharing data across ecosystem.
- **Location Services**: Fuses GPS, Wi-Fi, Bluetooth for precise tracking during searches.
- **Telemetry Overload**: Android's built-in reporting sends search-related metrics, even in "incognito" mode.

Snark: Google tracking is like that ex who "accidentally" checks your location—constantly, and without remorse.

9.2 Privacy-Respecting Public Search Engines

For beginners ditching Google, here are three top picks ranked by privacy prowess (based on independent indexing, no tracking, and transparency in 2025). No major red flags for DuckDuckGo (aside from past Microsoft ad ties, now resolved) or Startpage (uses anonymized Google results, but owned by ad firm—watch for shifts).

1. **Mojeek**: Independent index, no tracking, UK-based with strong privacy policy—pure, unadulterated results without the corporate snooping.
2. **Brave Search**: Independent crawler, ad-free by default, no user data collection—snark: unlike some "brave" browsers with crypto side hustles.
3. **DuckDuckGo**: No tracking, anonymized searches, but relies on Bing for results—reliable for beginners, though past ad deals raised eyebrows (resolved now).

Start with DuckDuckGo as your default—it's user-friendly and doesn't sell your soul.

9.3 Privacy-Respecting Meta-Search Engines

Meta-engines aggregate results without their own index, adding a privacy layer. Ranked by privacy:

1. **MetaGer**: Open-source, anonymizes queries, no ads or tracking—German efficiency for privacy purists.
2. **Startpage**: Anonymized Google results, no user data storage—solid, but owned by an ad company (no major breaches in 2025, though).
3. **Swisscows**: No tracking, family-friendly filters, Swiss-hosted—snark: like a neutral bank for your searches.

9.4 Advanced: Self-Hosting SearXNG

For the fearless, host your own meta-engine like SearXNG—aggregate results privately. We'll cover local and VPS setups.

9.4.1 Local Hosting (Without Docker)

1. **Install Dependencies**: On Ubuntu/Debian: `sudo apt update && sudo apt install git python3 python3-venv python3-dev build-essential libxslt-dev zlib1g-dev libffi-dev libyaml-dev libssl-dev`.
2. **Clone Repository**: `git clone https://github.com/searxng/searxng.git && cd searxng`.
3. **Set Up Virtual Environment**: `python3 -m venv venv && source venv/bin/activate && pip install -r requirements.txt`.
4. **Configure**: Copy `settings.yml.example` to `settings.yml` and edit (e.g., set `server: bind_address: "127.0.0.1"` for local, enable engines).
5. **Run**: `python3 searx/webapp.py`. Access at `http://localhost:8888`.
6. **Privacy Tips**: Disable public instances in config, use HTTPS self-signed cert, limit engines to privacy-friendly ones like DuckDuckGo.

9.4.2 VPS Hosting (With Docker)

1. **Install Docker**: On Ubuntu: `sudo apt update && sudo apt install docker.io && sudo usermod -aG docker $USER && newgrp docker`.
2. **Pull Image**: `docker pull searxng/searxng`.
3. **Create Directories**: `mkdir -p searxng && cd searxng`.
4. **Run Container**: `docker run -d -p 8080:8080 -v "$PWD:/etc/searxng" searxng/searxng`
5. **Configure**: Edit mounted `settings.yml` for engines, bind to VPS IP.
6. **Privacy Tips**: Use firewall (`ufw allow 8080`), enable HTTPS via reverse proxy (e.g., Nginx), anonymize queries.

Snark: Self-hosting? Like being your own Google, minus the evil empire.

9.5 Conclusion

Ditch Google—its tracking tentacles are everywhere. Start with DuckDuckGo for beginners, or go rogue with self-hosted SearXNG. Privacy isn't a feature; it's a fight. Stay shadowy.

Chapter 10

Android Unchained: Breaking Free from Spyware

Android phones: the pocket supercomputers that promise freedom but deliver a bloated buffet of preinstalled "essentials" like carrier apps, manufacturer trackers, and Google's endless data siphons. It's like buying a new car only to find it's wired with microphones and GPS beacons—courtesy of the dealer. Hilarious, right? In this chapter, we'll guide beginners to pick cleaner devices and empower the bold to flash privacy-respecting firmwares. We'll also swap out Google's tentacles for open-source alternatives. The goal? An Android that's as spyware-free as possible, with minimal ties to the Big G.

10.1 Choosing a Privacy-Friendly Android Phone

Most Android phones come loaded with junk—think Samsung's bloat empire or Google's "helpful" services that phone home every five minutes. For beginners, skip the flash-fest and start with devices that minimize spyware out-of-the-box. Based on 2025 reviews, here are the top three ranked by least preinstalled cruft and privacy focus (triple-checked: Fairphone for modularity, /e/OS for de-Googled stock, Volla for minimalism):

1. **Fairphone 5**: Modular design, minimal preinstalled apps, supports unlocking bootloader. Comes with Fairphone OS (Android 13, upgradable to 14), 4200 mAh battery, general price 750 USD.

2. **Murena Fairphone 5 (/e/OS Phone)**: Preinstalled with /e/OS (de-Googled Android 13), 4200 mAh battery, general price 600-700 USD.

3. **Volla Phone X23**: Lightweight, no excessive preloads, unlockable bootloader. Comes with Volla OS (AOSP fork based on Android 12) or Ubuntu Touch, 5000 mAh battery, general price 500 USD.

Google Pixel? Nice cameras, but it's a Google honey trap—deep integration with Play Services and telemetry make it a privacy no-go unless you flash it (more on that later). Stick with these options to start clean.

10.2 Flashing Privacy-Respecting Firmwares

If the preinstalled mess persists, flashing a custom firmware is your rebellion ticket—but warning: this voids warranties, demands technical skill, and risks bricking your phone. If you're unsure, stop now. Got an old, warranty-expired device and a backup phone? Go for it—backup everything first (photos, contacts, apps) via ADB or a PC, then try. Success means a private backup phone, maybe with better battery life!

10.2.1 Recommended Firmwares

- **LineageOS**: Open-source, removes Google apps, supports many devices. Check compatibility at https://wiki.lineageos.org/devices before proceeding.
- **LineageOS for microG**: Same as LineageOS but integrates microG (a free Google substitute) and F-Droid. Visit http://lineage.microg.org for details.
- **/e/OS**: De-Googled, user-friendly, with optional microG. Download from https://e.foundation/e-os.

10.2.2 Flashing Steps on Linux (General Guide)

1. **Backup**: Use ADB (`adb backup -apk -shared -all -f backup.ab`) or export data to a PC.
2. **Install Tools**: `sudo apt update && sudo apt install android-tools-adb android-tools-` (Ubuntu/Debian).
3. **Unlock Bootloader**: Enable Developer Options/OEM unlocking in Settings, reboot to bootloader (`adb reboot bootloader`), then `fastboot oem unlock` (confirm on device).
4. **Install Custom Recovery**: Download TWRP for your device from https://twrp.me, then `fastboot flash recovery twrp.img`.
5. **Flash Firmware**: Boot to recovery (`fastboot boot twrp.img`), wipe data/factory reset, then install ROM zip from SD card or ADB sideload (`adb sideload lineage.zip`).
6. **Reboot**: Reboot and set up—install GApps or microG if needed.

10.2.3 Flashing Steps on Windows (General Guide)

1. **Backup**: Install ADB/Fastboot drivers from Android SDK Platform-Tools (https://developer.android.com/tools/releases/platform-tools), use CMD: `adb backup -apk -shared -all -f backup.ab`.
2. **Install Tools**: Download ADB/Fastboot, extract, add folder to PATH (System Properties ¿ Advanced ¿ Environment Variables).
3. **Unlock Bootloader**: Enable Developer Options/OEM unlocking, reboot to bootloader (`adb reboot bootloader`), then `fastboot oem unlock` (confirm on device).

4. **Install Custom Recovery**: `fastboot flash recovery twrp.img`.
5. **Flash Firmware**: Boot to recovery, wipe, install ROM zip via sideload (`adb sideload lineage.zip`).
6. **Reboot**: Reboot and set up.

Snark: Flashing's like a digital exorcism—cast out the bloat, but don't cry if the phone goes silent!

10.3 Replacing Google Apps and Services

Even with a custom firmware, Android's Google roots linger. Let's swap them for open-source goodies.

10.3.1 Step-by-Step Replacements

1. **Google Play Store**: Replace with F-Droid (https://f-droid.org)—open-source app store, no tracking. Install APK from site, enable "Unknown Sources." For Play Store access, use Aurora Store (F-Droid).
2. **Google Maps**: Use Organic Maps (F-Droid)—offline maps, no ads, privacy-first.
3. **Gmail**: Switch to Tutanota (https://tutanota.com) or ProtonMail (https://proton.me/mail)—encrypted email, open-source clients.
4. **Google Calendar**: Try Etar (F-Droid)—simple, open-source calendar.
5. **YouTube**: Use NewPipe (F-Droid)—ad-free, background play, no Google tie-in.
6. **Google Drive**: Use Nextcloud (self-host or use https://nextcloud.com)—open-source cloud storage.

10.3.2 Additional Tips

- Disable Play Services if microG isn't used—root access required via Magisk. - Use Orbot (Tor) for network anonymity on Android.

Snark: Say goodbye to Google's "helpful" nudges—your phone won't miss the corporate babysitter!

10.4 Conclusion

Android's a battlefield—preinstalled bloat and Google's grip make privacy a fight. Pick a clean phone, flash if you dare, and ditch the apps with open-source flair. Your device, your rules—now go conquer the grid!