Design and Deployment of a Hardware Based Multi Layered Global DNS Filtering System for Secure Internet Access

by

Sohel Al Ashraf
ID: CSE2202026063

Marinal Biswas
ID: CSE2202026066

Rashida Khatun
ID: CSE2202026008

Sumon Hossen
ID: CSE2202026140

Supervised by
Md. Shamim Hossain

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

May 2026

APPROVAL

The project titled "**Design and Deployment of a Hardware Based Multi Layered Global DNS Filtering System for Secure Internet Access**" submitted by Sohel Al Ashraf (CSE2202026063), Marinal Biswas (CSE2202026066), Rashida Khatun (CSE2202026008) and Sumon Hossen (CSE2202026140) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Supervisor

Md. Shamim Hossain
Lecturer
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Md. Shamim Hossain, Lecturer,** Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Md. Shamim Hossain)
Supervisor

Signature

Sohel Al Ashraf
ID: CSE2202026063

Marinal Biswas
ID: CSE22020206066

Rashida Khatun
ID: CSE2202026008

Sumon Hossen
ID: CSE2202026140

ABSTRACT

The increase in online ads, user tracking scripts, advertisements, malicious domains, adult content and bad actors via the internet has created many challenges with respect to the amount of bandwidth used by a network, how much privacy users have while using a network, and the types of content that are safe to use on a network.

This project will describe the design, implementation, and analysis of an economical network security appliance built around a Raspberry Pi 2 W platform. Because of this project where the AdGuard Home was deployed as a DNS sinkhole filtering system, it has successfully reduced unapproved data traffic and allowed for parental controls to be put in place for an otherwise mixed Local Area Network (LAN).

Additionally, as a result of this project, there were statistically significant reductions in the time it took to load web pages, a vast reduction in the amount of tracker data collected from web sites and full enforcement of the content filtering policies in place. Furthermore, the integration of Tailscale creates a secure, encrypted tunnel that extends these protective features to remote clients, allowing for consistent DNS filtering regardless of geographical location. The resulting architecture provides a low-power, high-efficiency solution for unified threat management and privacy preservation across both local and Wide Area Networks (WAN).

Keywords: Ads; User tracking; Malicious domains; Adult content; Privacy; Raspberry Pi 2 W; AdGuard Home; DNS sinkhole filtering system; LAN; Tailscale; WAN;

ACKNOWLEDGMENT

At the very beginning, we would like to express my deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the scheduled time.

We are auspicious that we had the kind association as well as supervision of **Md. Shamim Hossain**, Lecturer, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary recourse to carry out our project.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to my parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

AGH	AdGuard Home
ARM	Advanced RISC Machines
BLE	Bluetooth Low Energy
CGNAT	Carrier-Grade NAT
CIDR	Classless Inter-Domain Routing
CLI	Command Line Interface
CPU	Central Processing Unit
DERP	Designated Encrypted Relay for Packets
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoH	DNS over HTTPS
DoT	DNS over TLS
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IOT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
MAC	Media Access Control
NAT-PMP	Network Address Translation Port Mapping Protocol
OpenGL	Open Graphics Library
OS	Operating System
OSI	Open Systems Interconnection
OTG	On-The-Go
PC	Personal Computer
PCP	Port Control Protocol
PCP	Primary Care Physician
RAM	Random Access Memory
ROM	Read Only Memory
RPZ	Reduced Pressure Zone Device
SDRAM	Synchronous Dynamic Random Access Memory
SoC	System on a Chip
SSH	Secure Shell
SSID	Service Set Identifier.
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UI	User Interface
UPnP IGD	Universal Plug and Play - Internet Gateway Device
URL	Uniform Resource Locator

VPN	Virtual Private Network
VSM	Value Stream Mapping
WAN	Wide Area Network

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

The Internet has become one of the most dominant aspects of modern life and online communication. While it supports free digital services, the uncontrolled growth of intrusive advertisements has created serious challenges for users and organizations alike. A large portion of web traffic today is occupied by unwanted ads, trackers, and malicious scripts, which degrade user experience, consume bandwidth, and introduce potential security and privacy risks. Studies found that advertising code can lead up to 25-30% of a typical webpage's total data volume. Beyond mere annoyance, these trackers pose privacy risks by harvesting user browsing habits. Furthermore, the unregulated nature of the internet exposes children and sensitive users to adult content, violence, and phishing scams. Similar to how spam once overwhelmed email systems, online advertisements have now become a persistent and evolving problem across the internet.

Over the past decade, advertisements have evolved from simple banner displays into sophisticated tracking mechanisms capable of monitoring user behavior and delivering targeted content. These ads not only interrupt browsing but also place a significant burden on network infrastructure and device performance. Furthermore, many ads act as carriers for phishing attempts, malware distribution, and other cyber threats, making them a critical concern in modern cybersecurity.

Various approaches have been proposed to mitigate this problem, including browser based ad blockers, content filtering extensions, and DNS level filtering systems. Among these, networkwide ad blocking has emerged as a highly effective solution, as it provides centralized protection for all devices connected to a network without requiring individual configuration. This approach is particularly valuable in environments with multiple devices such as homes, offices, and IoT ecosystems.

In this project, we explore the development of a network level ad blocking system using a Raspberry Pi and AdGuard Home. The system operates by filtering domain name requests and blocking access to known advertising, tracking, and malicious domains before content is delivered to the user. By leveraging DNS based filtering, the system ensures fast, efficient, and scalable ad blocking across all connected devices.

Unlike traditional methods that rely on client side filtering, this solution offers a lightweight, low cost, and energy efficient alternative. The Raspberry Pi serves as a dedicated local server, while AdGuard Home provides advanced filtering capabilities, customizable rules, and real time monitoring. This integration allows the system to adapt to evolving advertising techniques and maintain high effectiveness against new threats.

1.2 Problem Statement

DNS based filtering is an effective way to block unwanted traffic; however, it is traditionally limited to the local area network (LAN). Once a mobile device leaves the home Wi-Fi environment, it loses its connection to the secure DNS resolver, leaving it vulnerable to tracking and inappropriate content. Additionally, running high level network filtering on standard PC hardware is energy inefficient. This project addresses the need for a low power, high availability 'DNS over VPN' appliance that maintains security persistence across disparate network architectures

1.3 Objectives

The primary objectives of the project are as follows:

- Centralized Filtering: Block ads, trackers, viruses and malware at the DNS level for every device on the network (Laptop, IoT, Smartphones) without installing individual software.
- Remote Protection: Use Tailscale to extend home grade protection to mobile devices anywhere in the world.
- Parental Oversight: Implement blocked Services to restrict access to specific platforms (e.g., YouTube, TikTok, or Adult content) during specific hours or permanently.
- Performance Optimization: Reduce bandwidth consumption and improve page load times by preventing heavy ad scripts from ever downloading.

1.4 System Architecture

- Hardware: Raspberry Pi Zero 2 W (Quad core CPU clocked at 1 GHz and 512 MB of LPDDR2 SDRAM).
- Primary Filter (AdGuard Home): Acts as the DNS resolver. It checks every outgoing request against a blocklist. If a request is for ads.google.com, it returns a null address, effectively killing the ad.
- The Mesh VPN (Tailscale): Creates a secure, encrypted "Tailnet." By setting the Pi Zero as the Global Nameserver in the Tailscale admin console, the phone sends its DNS queries through the encrypted tunnel to the Pi at home.

1.5 Key Features & Implementation

Table 1 : Key Features & Functionality of the System

Feature	Implementation
Blocking	Uses adguard dns blocklists (e.g., HaGeZi, OISD) to stop ads and trackers, viruses and malware.
Parental Controls	One click blocking for social media, gaming, or gambling sites via the block services tab in adguard.
Remote Access	Tailscale nameservers capability to connect to home dns while traveling.
Low Footprint	Operates on less than 2 Watts of power.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Early research in this domain focused on two distinct areas, the classification of content itself (software algorithms) and the management of the network policies used to block access (infrastructure/hardware logic). However, with the rise of the Internet of Things (IoT) and encrypted traffic (HTTPS), the focus has moved from simply determining what to block to deciding where to block, increasingly at the DNS layer. This review synthesizes foundational papers on content classification with modern studies on DNS blocking, IoT security, and network policy management, exploring the technical challenges, economic drivers of ad avoidance, and the ongoing quest for a secure browsing experience.

2.2 Existing Software for DNS and Content Blocking

Software based filtering operates primarily at the application or transport layers of the OSI model. Research in this area has evolved from simple string matching to complex heuristic and statistical analysis.

a. **Firewall Logic and Policy Software**
   Al-Shaer and Hamed (2003)[1] focused on the software logic required to manage firewall rules. Their research led to the development of software tools capable of detecting "Shadowing" and "Redundancy" in filtering policies, ensuring that DNS or IP blocking rules do not conflict with legitimate traffic.

a. **Statistical and Compression Models**
   Bratko et al. (2006)[3] introduced the use of statistical data compression for spam filtering. By utilizing Dynamic Markov Compression, the software could identify malicious or unwanted content without the need for manual feature engineering, making it more resilient to the "obfuscation" techniques used by advertisers and spammers.

b. **Text and Image Multimodal Software**
   Systems like "WebGuard," as explored by Chen et al. (2006)[4] and Hu et al. (2007)[6], represent the software evolution toward visual textual fusion. These software frameworks analyze both the HTML text and the embedded image metadata to block content that purely text based filters might miss.

c. **Browser Level Ad Avoidance**
   Anderson and Gans (2011)[2] examine the economic software user relationship, where ad blocking software acts as a gatekeeper. This research highlights the shift where software moved from being a security tool to a consumer utility tool for improving user experience.

d. **Browser Extensions and Filter Lists**
   As web complexity increased, blocking software shifted toward matching URLs and domain names against community maintained blocklists (e.g., EasyList). Nikiforakis

et al. (2014)[11] provided a comprehensive analysis of the ad blocking ecosystem, noting that modern software relies heavily on browser extensions that use declarative rules to block HTTP requests. They highlighted that while this software is effective for desktop browsers, it fails to protect devices that do not support browser extensions (e.g., smart TVs, refrigerators), creating a security gap that DNS blocking attempts to fill.

2.3 Hardware and IoT for DNS Blocking

Hardware based filtering implementations provide a critical, centralized "bottleneck" architecture that ensures uniform protection across an entire network. Unlike software agents that must be installed on individual machines, hardware solutions extend security to all connected endpoints, including restricted Internet of Things (IoT) devices that lack the processing power or OS flexibility to run third party security software.

a. **Network Policy and Infrastructure**
   The foundation of hardware level blocking is the management of network rules. Al-Shaer and Hamed (2003)[1] laid the groundwork for this with their Firewall Policy Advisor, which addressed the complexity of managing rule sets in network hardware. In modern IoT contexts, this translates to configuring the home router (the hardwire) to act as a DNS sinkhole.
b. **Hardware Firewalls and Proxy Servers**
   Historically, hardware based filtering was the domain of enterprise grade equipment. Du et al. (2003)[5] discussed the integration of text classification directly into network interface hardware/gateways. By moving the filtering logic to the "chokepoint" of the network, administrators can enforce policies across all connected hardware without installing individual agents.
c. **Embedded System Constraints**
   Research into hardware blocking emphasizes the need for lightweight algorithms. Unlike software running on a powerful PC, IoT hardware for DNS blocking must handle high throughput with minimal latency. The statistical models proposed by Bratko et al. (2006)[3] are particularly relevant here, as compression based models often require less memory than large neural networks.
d. **Device Identification and Traffic Control**
   Effective IoT blocking requires distinguishing between legitimate device traffic and ad tracking traffic. Miettinen et al. (2017)[10] proposed "IoT SENTINEL," a system that identifies IoT devices based on network traffic fingerprints. While not a blocking tool per se, this research is vital for "hardwire" blocking, as it allows the network to apply specific DNS policies to specific devices (e.g., blocking ads on a Smart TV but not on a PC).
e. **DNS Sinkholing (Hardware Implementation)**
   Vixie (2012)[13] introduced the concept of Response Policy Zones (RPZ), a critical piece of "hardware" infrastructure. RPZ allows DNS resolvers to apply policy logic to name resolution, effectively enabling a network wide switch to block malicious or advertising domains. This standard is widely used in modern "Pi-hole" style solutions for IoT.

The most prominent recent evolution in hardware filtering is the DNS sinkhole. Devices like the Pi-hole (leveraging Raspberry Pi hardware) act as a local DNS server. When an IoT device requests an ad serving domain, the hardware intercepts the request and returns a null address. This method effectively neutralizes unwanted traffic before it ever reaches the user's device, providing a seamless protective layer that requires zero configuration on the client side.

2.4 Challenges Faced in Previous Research

The challenges identified in previous research underscore the inherent complexity of maintaining robust filtering systems within an ever changing digital landscape. These obstacles are not merely technical but represent a multifaceted struggle involving logical consistency, adversarial evasion, administrative sustainability, and economic impact.

Table 2: Challenges Faced in Previous Research & Description

Challenge Category	Description	Primary Sources
Encryption & Granularity	The rise of HTTPS obscures URL paths, forcing a shift to DNS level blocking. This lacks granularity, often resulting in blocking an entire site rather than just a specific ad element.	Naylor et al. (2015)[12]; Anderson & Gans (2011)[2]
Classification Accuracy	The persistent struggle between "over blocking" (breaking legitimate content) and "under blocking" (missing obfuscated ads/threats) due to adversaria[8][8]l tactics.	Falahrastegar et al. (2015)[8]; Hu et al. (2007)[6]
Computational Resource Constraints	The difficulty of running sophisticated filtering models on low power IoT hardware without causing significant network latency or memory exhaustion.	Ikram et al. (2019)[9] ; Bratko et al. (2006)[3]
Policy & Rule Anomalies	Conflicts between blocking rules that cause "legal" traffic to be dropped or "illegal" traffic to pass.	Al-Shaer & Hamed (2003)[1]
Maintenance	The unsustainable nature of manually updating blacklists for millions of dynamic domains.	Du et al. (2003)[5]
Economic Disruption	The friction caused by ad avoidance technology on media revenue models, leading to "anti blocking" countermeasures.	Anderson & Gans (2011)[2]
Content Evasion	The use of images, JavaScript, or unconventional encoding to bypass text based filters.	Hu et al. (2007)[6] ; Chen et al. (2006)[4]

2.5 Previous Research and Related Works

The synthesis of these papers reveals a clear technological lineage:

- **Efficiency in Classification:** Du et al. (2003)[5] were among the first to argue that for a filter to be viable, it must classify content in "near real time." Their use of the Vector Space Model (VSM) for web filtering remains a cornerstone for how modern software categorizes domains. Chen et al. (2006)[4], and Hu et al. (2007)[6] focused on the software problem of identifying unwanted content using text and images.
- **Security Policy Verification:** Related work by Al-Shaer and Hamed (2003)[1] established the "Firewall Policy Advisor" model. This research is vital for modern DNS hardware (like pfSense or Cisco Umbrellas) to ensure that adding a new ad blocking list doesn't inadvertently break critical IoT communication protocols.
- **The Move to Automated Learning:** Early work relied heavily on static lists. However, the shift toward "statistical data compression" (Bratko et al., 2006)[3] paved the way for modern AI driven filters that can recognize a "malicious looking" domain or packet even if it has never been seen before.
- **Economic Drivers**: Anderson and Gans (2011)[2] explained the user motivation for adopting these technologies.
- **Modern DNS Adaptation:** Recent work by Vixie (2012)[13] (RPZ) and Nikiforakis et al. (2014)[11] shifts the focus from client side analysis to network level DNS enforcement, specifically to address the rise of IoT and encryption that renders older content analysis methods obsolete.

CHAPTER 3

ARCHITECTURE REVIEW

3.1 Overview of System Design

The project is designed to provide an integrated DNS filtering and parental control tool for Wi-Fi networks by leveraging hardware based security solutions and virtual private networking. The system architecture consists of several components working in unison to enable network wide ad blocking, privacy protection, remote access and protection from viruses and malware. The network is designed to operate seamlessly across both local and remote environments, ensuring that security policies remain active regardless of the device's physical location. By utilizing a **Raspberry Pi Zero 2 W**, **AdGuard Home**, and **Tailscale**, the system provides a scalable, robust, and efficient solution for personal network management.

The system is intended to be deployed as a centralized network gatekeeper, using DNS level interception to manage traffic for every device on a wi-fi network. This setup eliminates the need for individual software installations on every client device. A web based user interface allows administrators to manage blocked services, monitor real time network queries, and configure parental control settings to restrict specific platforms or malicious domains. All of the hardware and software used in the system to build are free, open source.

This visual represents how data flows when you are at home versus when you are remote.

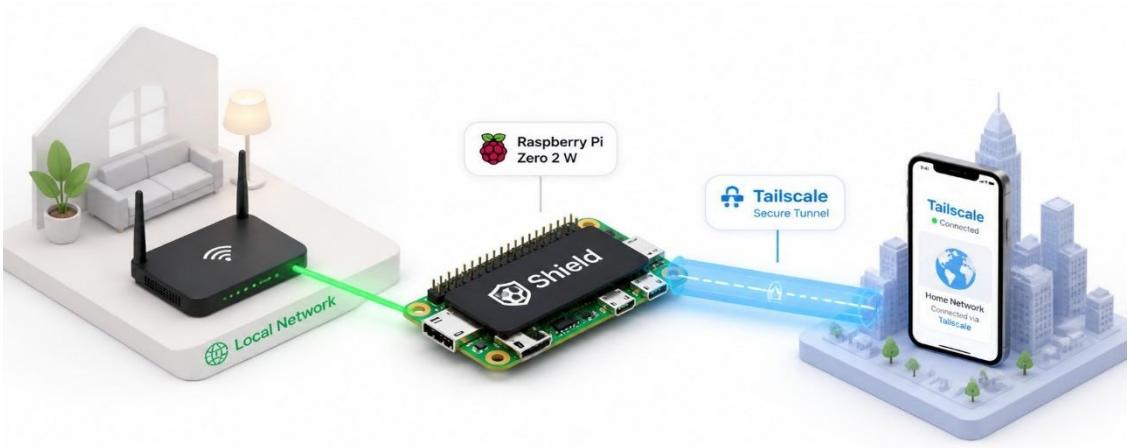

Fig.1: Basic diagram of the system

3.2 Components and Their Functions

a. Hardware

Device: Raspberry Pi Zero 2 W is RP3A0, a custom built system in a package designed by Raspberry Pi in the UK. With a quad core 64-bit ARM Cortex A53 processor clocked at 1GHz and 512MB of SDRAM. Wireless (Integrated) 2.4GHz 802.11 b/g/n wireless LAN is built into a shielded enclosure with improved RF compliance, it’s more flexibility when designing with Raspberry Pi Zero 2 W. Bluetooth 4.2, Bluetooth Low Energy (BLE), onboard antenna.

Ethernet (via USB dongle if more stability is required). Mini HDMI port and micro USB On The Go (OTG) port. MicroSD Card. CSI-2 camera connector. HAT-compatible 40-pin header footprint (unpopulated). H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30). OpenGL ES 1.1, 2.0 graphics. Micro USB power. Composite video and reset pins via solder test points. All in the same tiny 65mm × 30mm form factor.

b. Software

Operating System Raspberry Pi OS Lite (Headless).Stripped of GUI to save RAM and CPU cycles for DNS handling and VPN encryption.

Tailscale is the open source software that acts in combination with the management service to establish (P2P)peer-to-peer or relayed VPN communication with other clients using the WireGuard protocol. Tailscale can open direct connection to the peer using NAT traversal techniques such as STUN or request port forwarding via UPnP IGD, NAT-PMP or PCP. If the software fails to establish direct communication, it falls back to using DERP (Designated Encrypted Relay for Packets) protocol relays provided by the company.

AdGuard Home Server The AdGuard Home server, hosted on the Raspberry Pi Zero 2 W, serves as the central data processing and management system. It hosts a local web interface that provides a platform for collecting, storing, and processing DNS query data from all connected devices. The AdGuard server receives traffic transmitted from the network, checks it against blocklists, and processes it in real time. This data can then be displayed on a dashboard, where users can access important information, such as blocked domain statistics, parental control hits, and client activity logs. The server allows for remote monitoring and is essential for data driven decision making regarding network security and privacy.

3.3 Network topology

The system operates in two distinct modes depending on the client's location.

Fig.2: Network topology

a. **Local Network Mode (At Home)**
   - **Request:** Device must connects to the Home Router Wi-Fi.
   - **Routing:** The Router points to the Pi's local IP (e.g., 192.168.0.101) as the DNS server (either via Router settings or Pi DHCP).
   - **Processing:** The request hits AdGuard Home directly over the local LAN.
   - **Filtering:** AdGuard blocks/allows based on your rules.
   - **Response:** The Pi returns the IP address (or blocked page) to the device.

b. **Remote Network Mode (On the Go / Cellular)**
   - **Connection:** Your device (e.g., iPhone) enables Tailscale.
   - **Tailscale MagicDNS:** Tailscale assigns a name to your Pi (e.g., pi-zero.ts.net).
   - **DNS Push:** In the Tailscale admin console, you configure "Global Nameservers" to point to the Pi's Tailscale IP (100.x.y.z).
   - **Request:** You open a browser on cellular data. The device asks Tailscale for the DNS.
   - **Tunneling:** The DNS request travels through the encrypted Tailscale tunnel to your Pi at home.
   - **Processing:** AdGuard processes the request exactly as if you were home.
   - **Response:** The filtered response travels back through the tunnel to your phone.

3.4 Data Flow and Communication

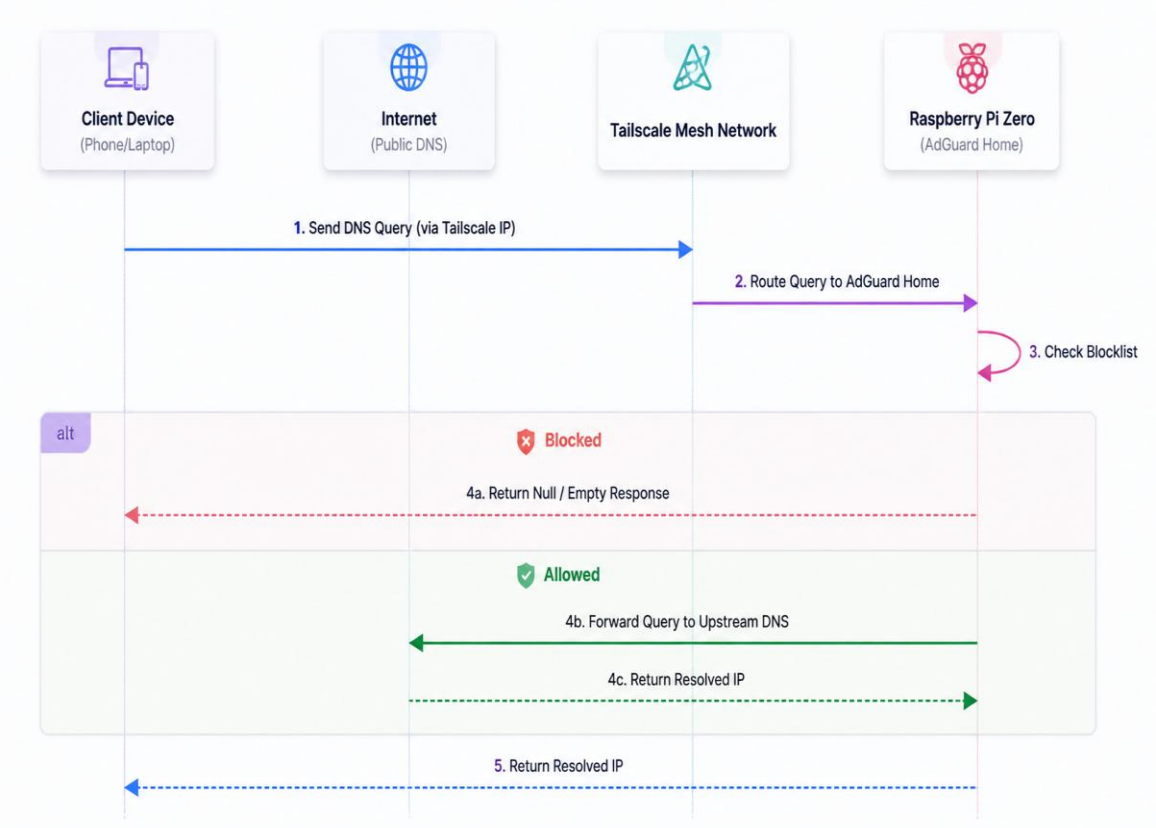

Fig.3: Data Flow and Communication

The data flow within the integrated network security system follows a straightforward path:

a. **Request Initiation:** Client devices (smartphones, laptops, or IoT hardware) initiate a connection request by querying a domain name. This request is automatically directed to the Raspberry Pi Zero 2 W via the local DHCP configuration or the Tailscale encrypted tunnel.
b. **DNS Interception:** The AdGuard Home engine intercepts the DNS query before it reaches the public internet. It cross references the requested domain against active blocklists and parental control rules to determine if the connection should be permitted.
c. **Filtering and Resolution:** If a domain is identified as a tracker, advertisement, or restricted service, the system returns a null IP address (0.0.0.0), effectively blocking the content. Legitimate requests are forwarded to an encrypted upstream DNS provider for final resolution and then transmitted back to the client device.
d. **Administrative Monitoring:** The AdGuard Home interface logs the outcome of each request, providing real time data visualization on the dashboard. Users can access this web interface from any authorized device to view network statistics, manage filtering rules, and monitor blocked service attempts.

The entire communication process is designed to be low latency and highly secure, ensuring that even when roaming on external networks, data remains protected and filtered in real time. The decentralized architecture of the system leveraging a dedicated hardware hub further enhances its resilience, as it provides a consistent layer of security that does not rely on individual device software configurations.

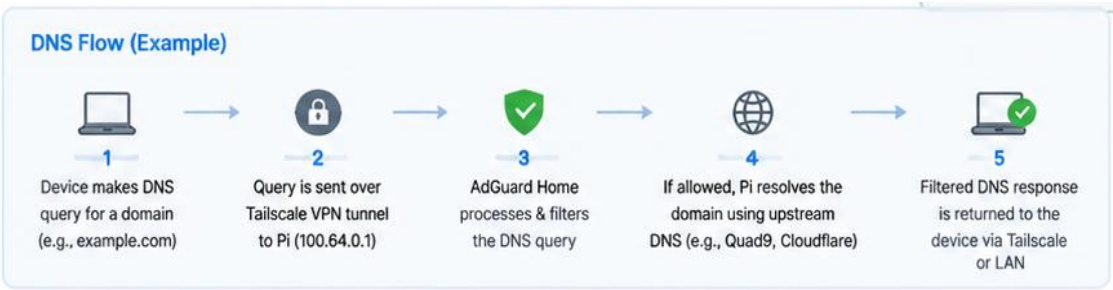

Fig.4: DNS Flow

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Hardware Setup Configuration

a. Hardware

The hardware setup needs following components

1. Computer: A PC, Mac or Laptop to flash the SD card.

Fig.5: Laptop

2. Raspberry Pi Zero 2 W

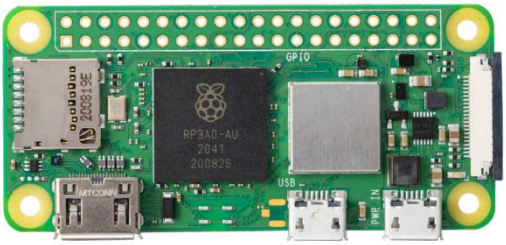

Fig.6: Pi front Fig.7: Pi back

3. MicroSD Card: 64GB, Class 10 (A1/A2 rating recommended for better I/O performance).

Fig.8: Laptop

4. Power Supply: Crucial. A stable 5V 2.5A USB-C power supply.

Fig.9: USB-C power supply

5. MicroSD Card Reader (USB adapter).

Fig.10: SD card reader

6. Heatsink: Small aluminum heatsink for the CPU (highly recommended for 24/7 operation).

Fig.11: Heatsink(black), Thermal paste, 1 Set of Screws, 1 Screwdriver etc.

b. Physical Assembly

Attach Heatsink: Peel the adhesive off the small thermal and place it directly on the square black chip in the center of the board (the SoC).

Case Assembly: Assemble it now to prevent short circuits.

Fig.12: Pi assembled

Fig.13: All the port of Pi

c. OS Installation & Headless Configuration

Since pi zero cannot plug a monitor, configure the Wi-Fi and SSH credentials on the SD card using the main laptop.

Step 1: Download the Raspberry Pi Imager

Download and install the raspberry pi Imager on the laptop.

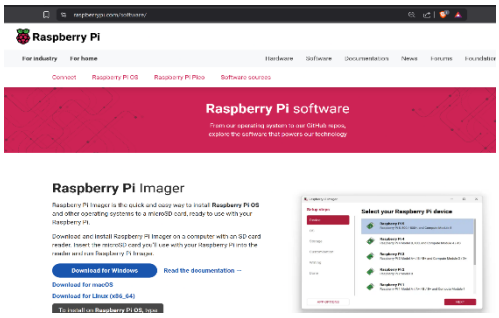

Fig.14: Pi Imager

Step 2: Configure the OS

- Insert the MicroSD card into the pc/laptop via the card reader.
- Open Raspberry Pi Imager.
- Device: Select Raspberry Pi Zero 2 W
- Choose OS: Select Raspberry Pi OS (other) → Raspberry Pi OS Lite (64-bit).

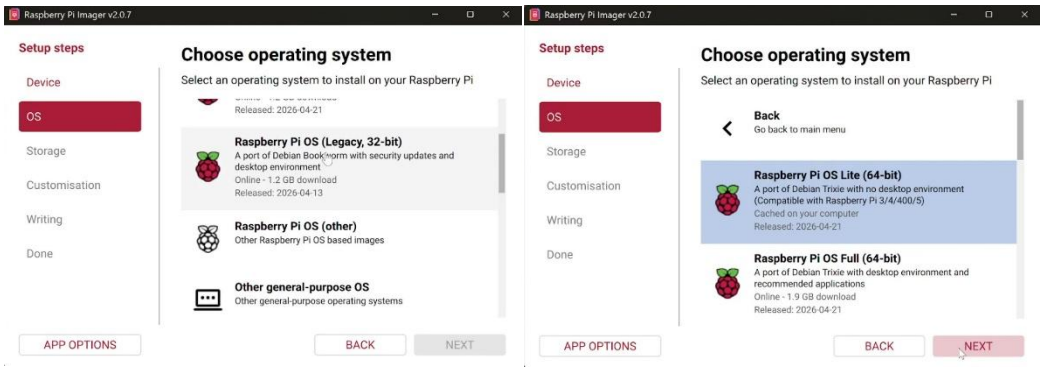

Fig.15: Pi OS

Reason: The 64-bit Lite version has no graphical interface, saving RAM and CPU for AdGuard and Tailscale.

- Choose Storage: Select your MicroSD card.
- Customisation Choose hostname: adguard (This makes it easier to find on your network).

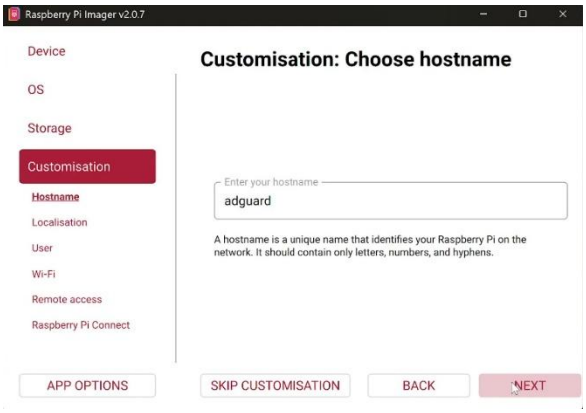

Fig.16: Hostname

- Localisation
  Capital city: Dhaka (Bangladesh)
  Time zone: Asia/Dhaka
  Keyboard layout: us
- User → username (to access pi zero through ssh)

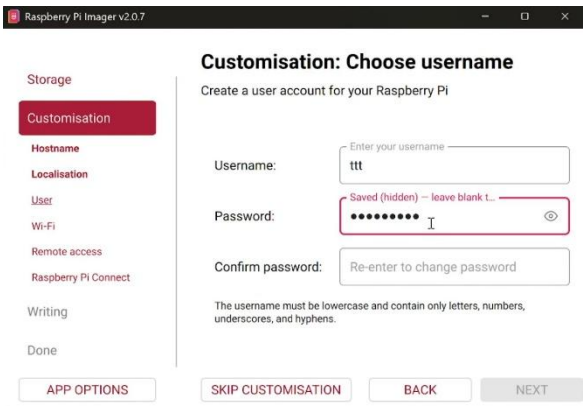

Fig.17: SSH user and password

- Wi-Fi: → **SSID:** Home Wi-Fi name → **Password:** Home home Wi-Fi password.

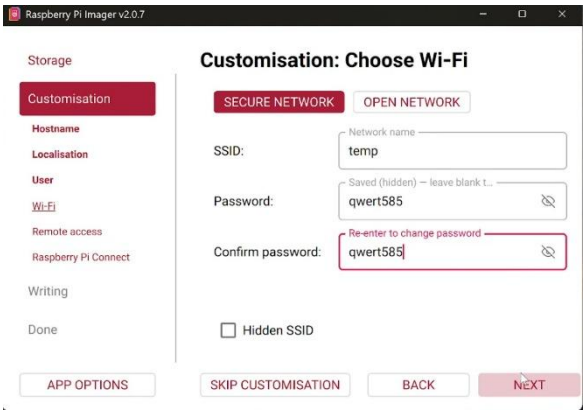

Fig.18: Wi-fi user

- Remote access → Enable SSH: Use password authentication.
- Write image → Click WRITE. (This will take a few minutes). → Click FINISH

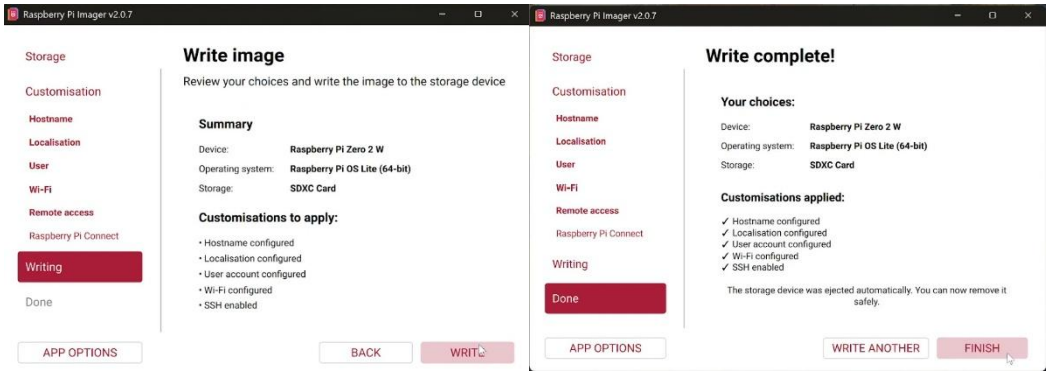

Fig.19: Setup complete

Step 3: Eject

Once the write is finished, remove the SD card from the laptop and insert it into the Pi Zero 2 W's SD card slot.

d. First Boot & Connection

- Power Up: Plug the USB-C power cable into the Pi Zero 2 W.
- Wait: Give the Pi about 60-90 seconds to boot up and connect to Wi-Fi.
- Find the IP Address: Log into the home router's admin page and look for the device named adguard (the hostname) in the Wireless Client. The ip address is **192.168.0.101** for adguard.

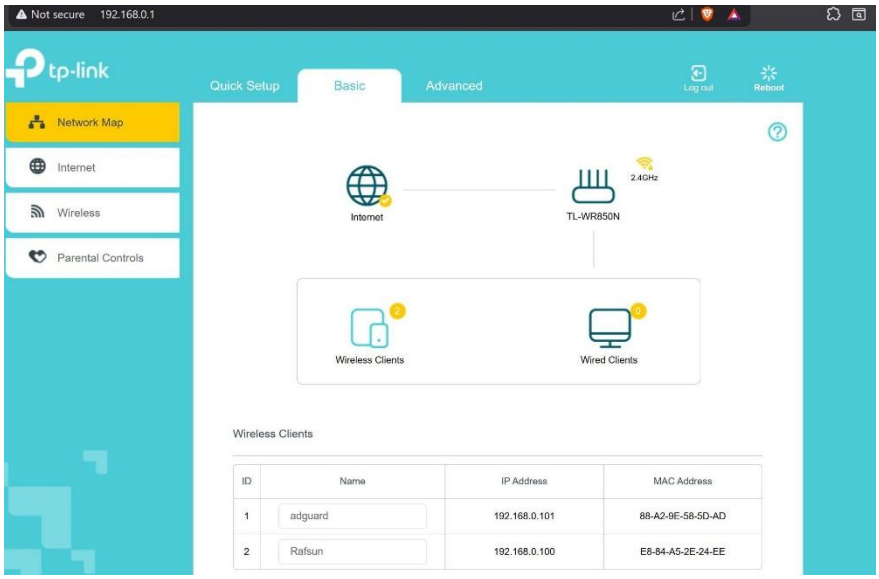

Fig.20: Wireless client

4.2 Preparation & Static IP

Before installing AdGuard, the Pi must have a fixed IP address. By binding a specific IP address to the hardware's unique identifier, the router is instructed to assign the same address to the device upon each connection request.

a. Set a Static IP

The easiest way to do this without editing complex network files is to set a "Static Lease" in the router's settings.

- Log into home router (192.168.0.1). Access the router's settings and create a DHCP Reservation specifically for your Pi Zero 2 W username(adguard).
- In the option to "Reserve" or "Assign Static IP" to its MAC address.
- This tells the router to always assign the same IP address to the Pi.

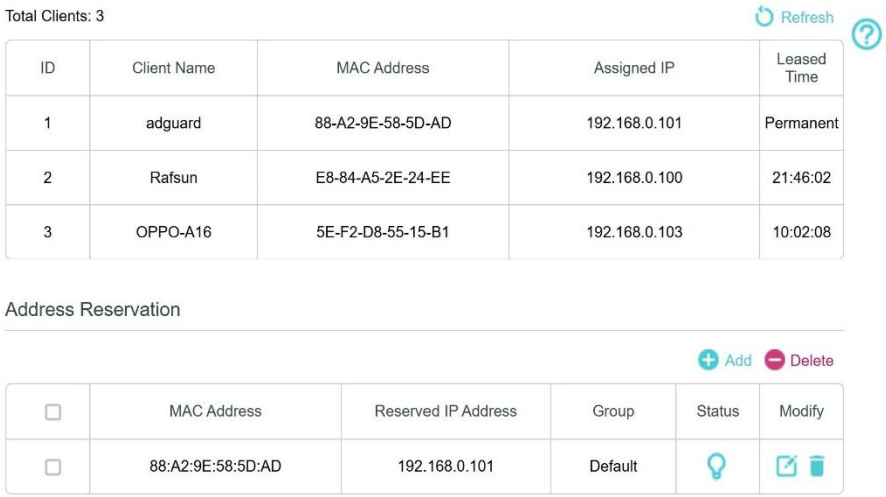

Fig.21: Static IP

b. SSH into the machine

Open terminal (or PuTTY on Windows) and SSH into the Pi: ssh ttt@192.168.0.101

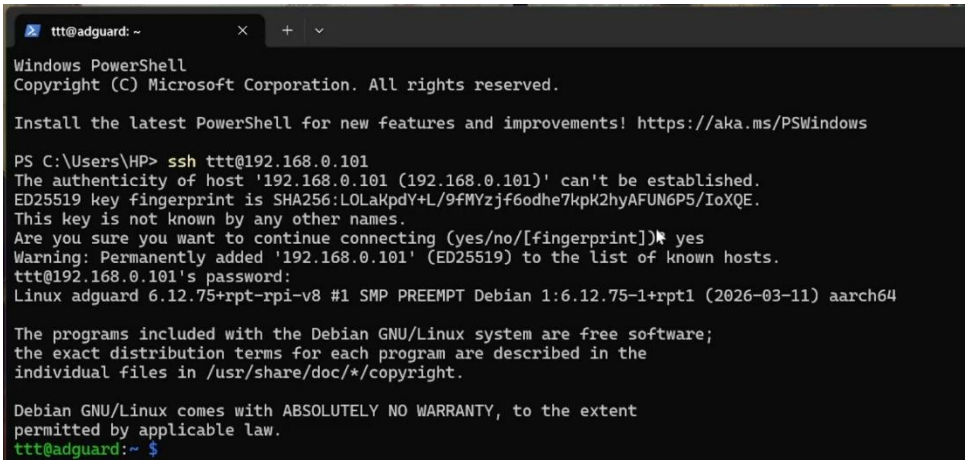

Fig.22: SSH to pi

Fingerprint: yes
Password: xxxxxx

c. Update your system

Ensure everything is up to date:

sudo apt update && sudo apt upgrade -y

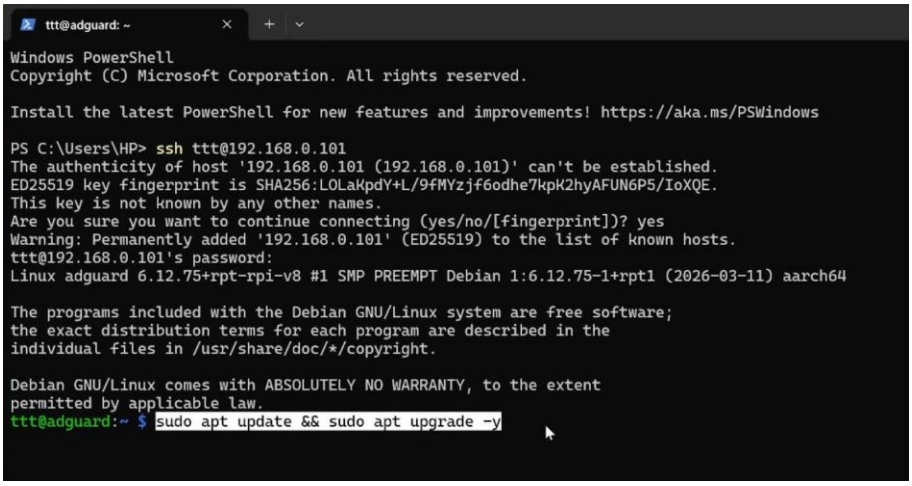

Fig.23: Update pi

4.3 Software Design and Development

1. **AdGuard Home**
   Adguard home is the brain of this project, it uses a DNS sinkhole to block domain.

a. Download and Install

Automated install (Linux)
To install with curl run the following command

curl -s -S -L
https://raw.githubusercontent.com/AdguardTeam/AdGuardHome/master/scripts/install.sh | sh -s -- -v

The entire installation process for download to install

Here is exactly what will happen:

- It downloads the official installer script directly from AdGuard's GitHub.
- It detects your Raspberry Pi model (Zero 2 W) and automatically figures out that it needs the ARMv7 version.
- It downloads the correct binary.
- It installs AdGuard Home as a system service.
- It starts AdGuard Home.

In short: It replaces the curl download, the tar extraction, and the sudo ./AdGuardHome -s install commands all in one go.

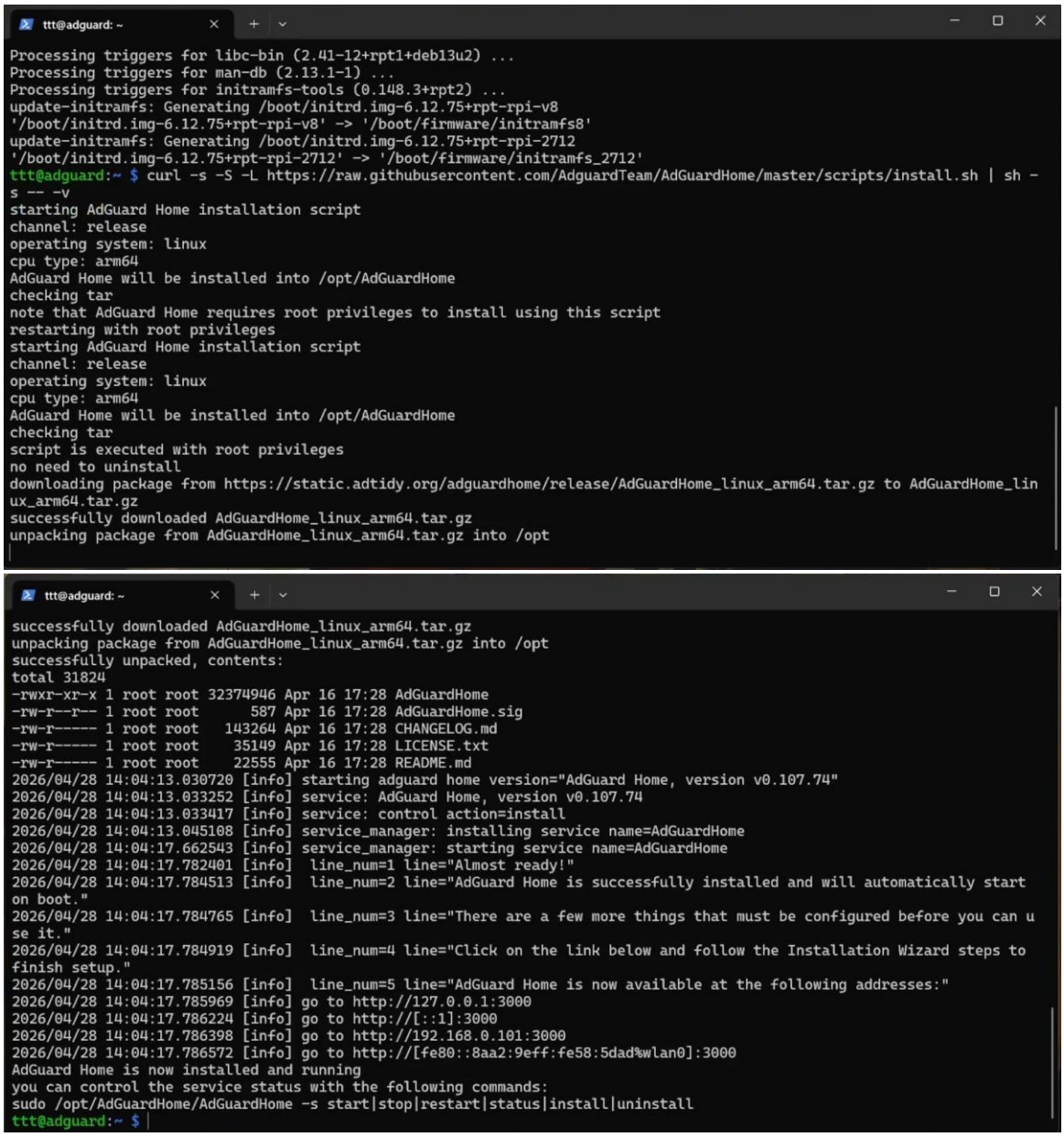

Fig.24: AdGuard download and install

This Script Way: It usually installs the program in a system folder (/opt/AdGuardHome) and sets everything up for use.

b. Initial Setup (The Browser)

Now that the software is running, you need to configure it.

1. Open the Setup Page

On computer (not the Pi), open a web browser and type: http://192.168.0.101:3000

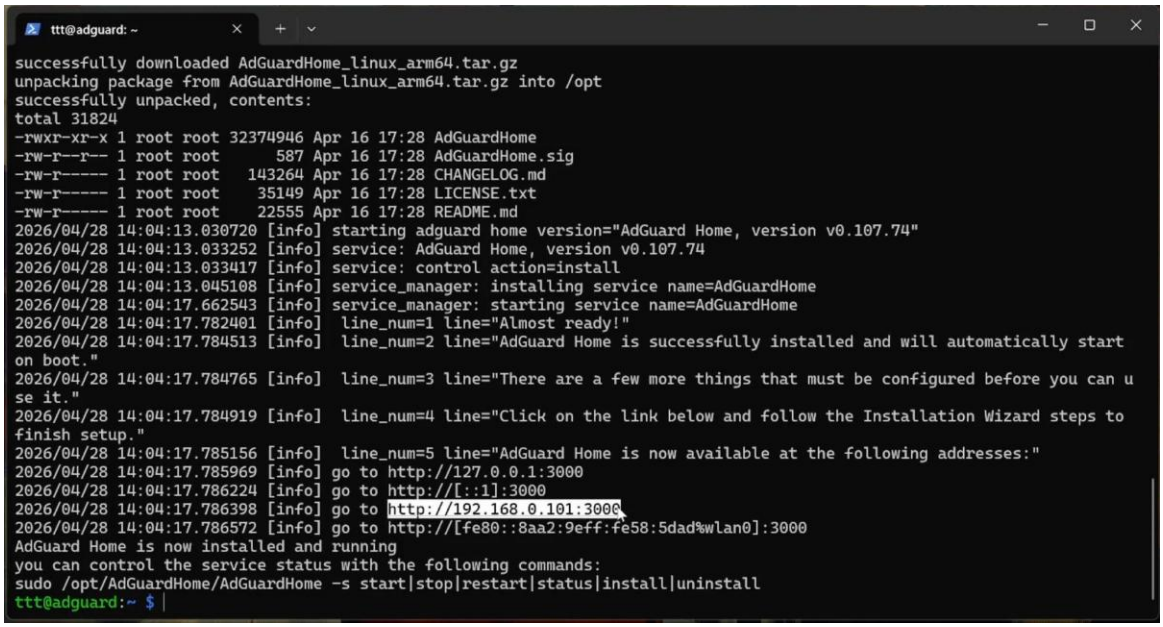

Fig.25: AdGuard setup and connect

c. Configure the Interface

- Step 1: Click Get Started.
- Step 2: Admin Web Interfaces → All interface → Port 80.
  DNS server → All interface → Port 53. Go down and before clicking next Static IP Address, AdGuard is a server and it needs a static ip to work properly. Now Click Next.

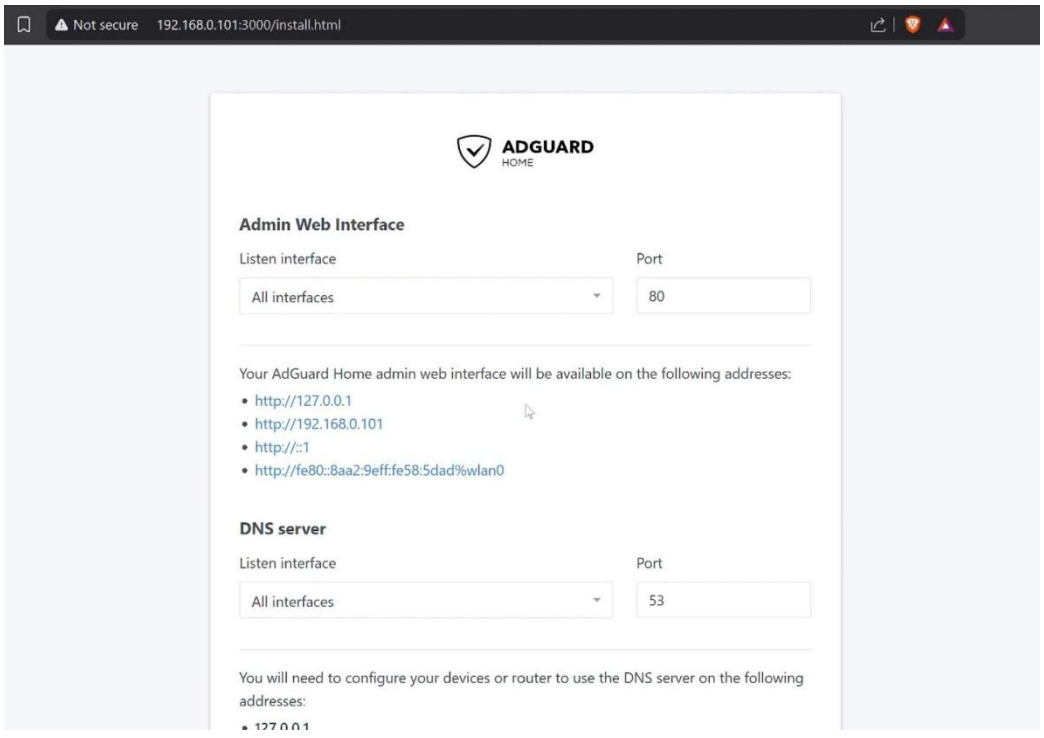

Fig.26: Configure Interface

- Step 3: Authentication for AdGuard Home. Create a username and password for AdGuard dashboard. For convenience use the same password as pi ssh password.

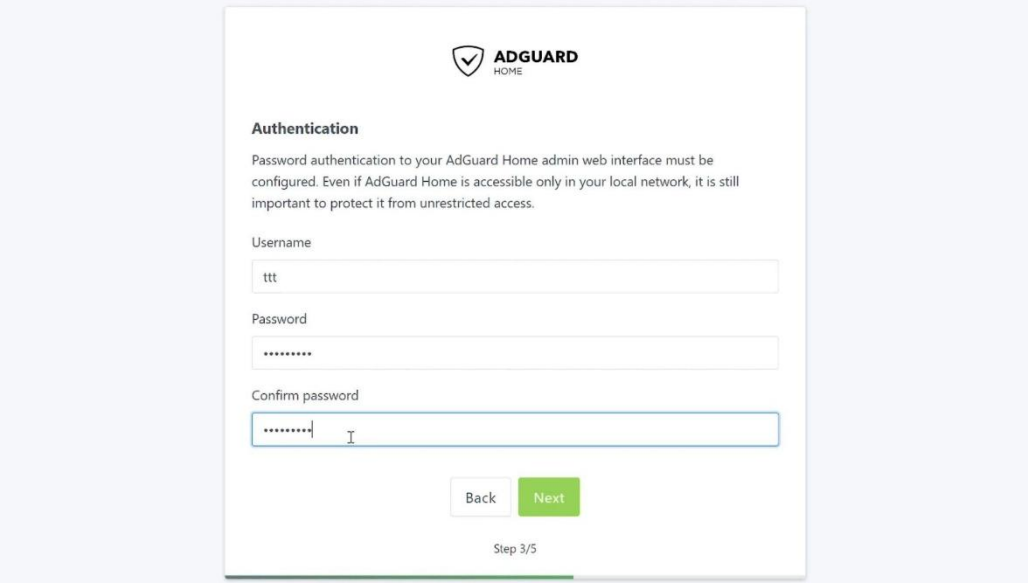

Fig.27: AdGuard Authentication

- Step 4: Configure your device select Router. Click Next.
- Step 5: Congratulations setup is complete. Click Open Dashboard.
- Now login to AdGuard home

  Important: Now AdGuard is working but our router is not using the DNS of the Adguard home. To do that check the next topic.

d. Make it work for your whole network

For AdGuard to actually block ads, tracker, malware/phishing, adult websites on phone, TV, and laptop, the devices must send their DNS requests to the Pi.

Option A: The Best Way (Router Settings)

This blocks ads, tracker, malware/phishing, adult websites on every device connected to your Wi-Fi automatically.

- Log into Router's admin page.
- Find the DNS settings (usually under LAN or DHCP settings).
- Change the Primary DNS to your Raspberry Pi's IP address.
- Save and Reboot router (or just reconnect your devices).

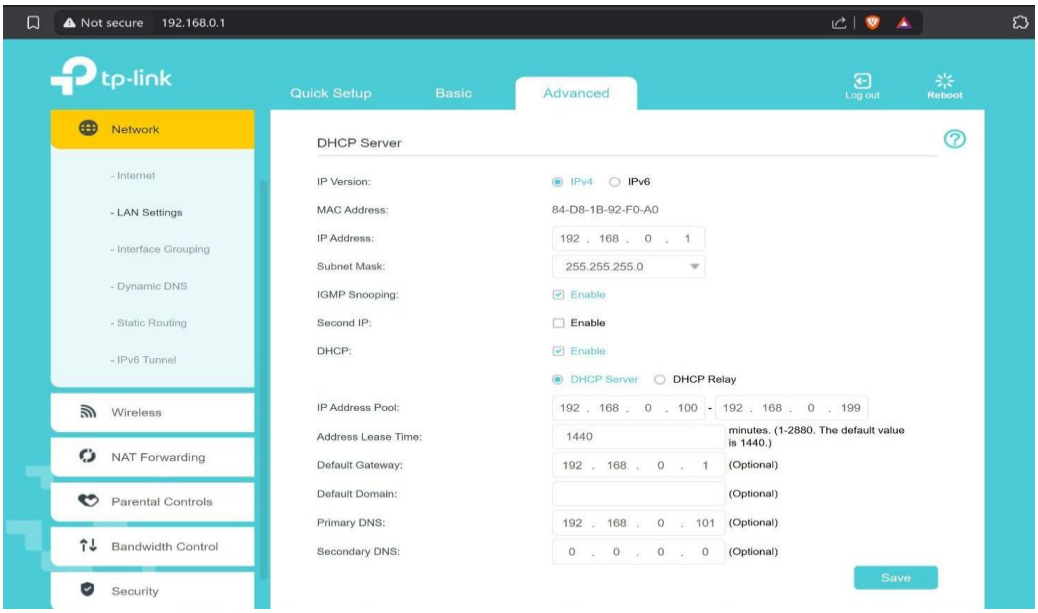

Fig.28: Primary DNS

Note: Some routers don't allow from changing the DNS to a local IP. If this happens, use Option B.

Option B: Individual Device Settings
Can't change the router settings, go to the Wi-Fi settings on specific devices (Phone, Laptop) and manually set the DNS to your Pi's IP address(192.168.0.101)

e. Verification

- Go to the AdGuard Dashboard (no port 3000 needed this time):
- http://192.168.0.101
- Log in with the admin account you created.
- Click Dashboard. You should see graphs showing "DNS Queries".
- Go to the Filters section in AdGuard and enable "DNS Blocklist" to actually start blocking ads.
- Open a website on your phone or laptop. You should see the query count go up on the dashboard.

2. Tailscale

Adding Tailscale to your Pi Zero 2 W allows you to securely access your AdGuard Home dashboard and benefit from ad blocking even when you are away from home.

Here is the step by step to integrating Tailscale with your current setup.

a. Install Tailscale on the Pi

- Connect to Pi by SSH: ssh ttt@192.168.0.101
- Update system packages to ensure compatibility:

sudo apt update && sudo apt update -y

- Execute the following command. Tailscale provides a simple script that detects your OS (Raspberry Pi OS) and architecture (ARM64) and installs the correct version.
- Run the installation script: curl -fsSL https://tailscale.com/install.sh | sh

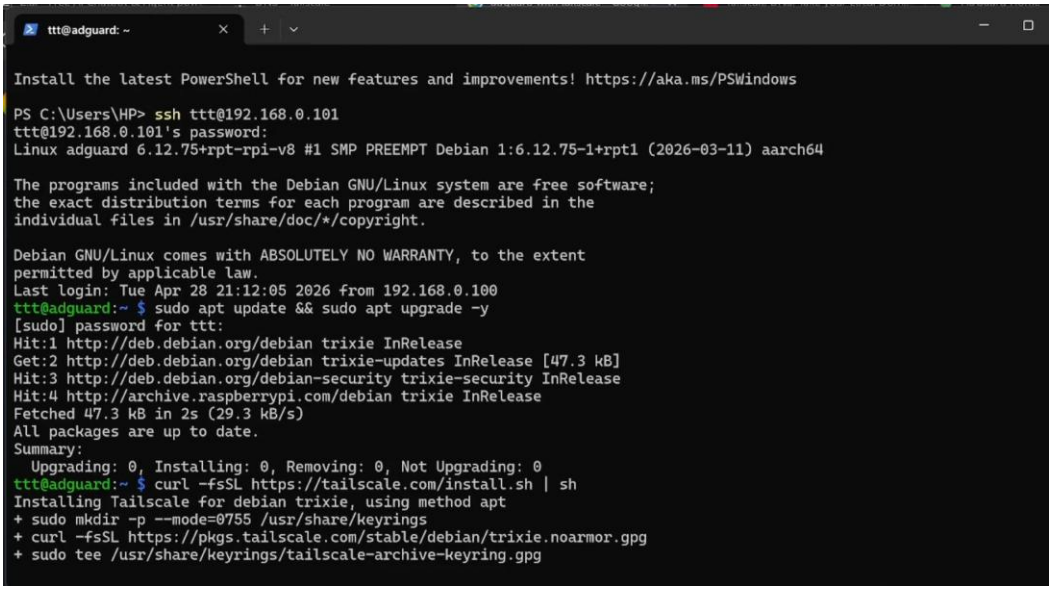

Fig.29: Tailscale install

b. Authenticate & Connect and start Tailscale

1. Start the service by running: sudo tailscale up --accept-dns=false
2. The Pi will print out a link that looks like this:

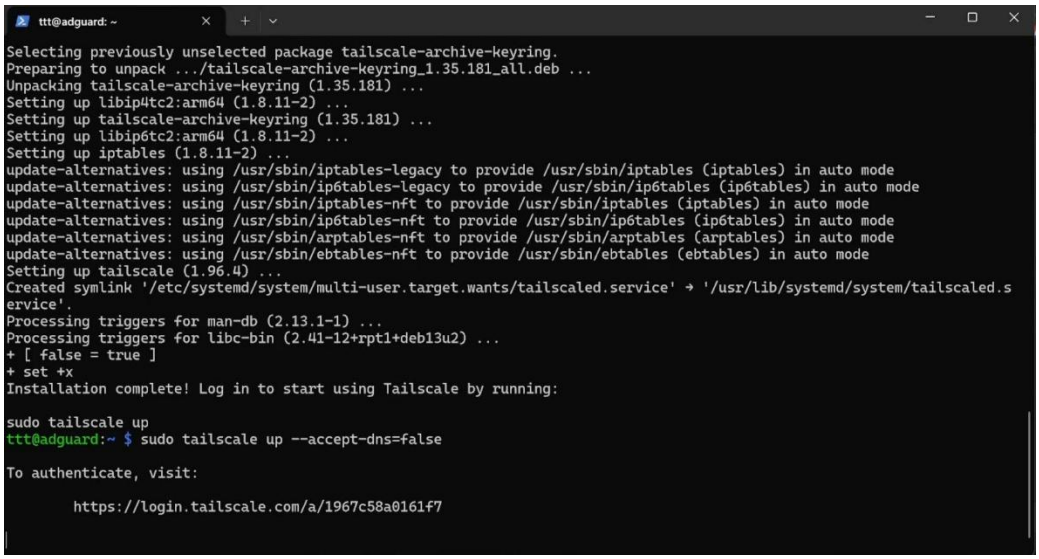

Fig.30: Tailscale Authentication

3. Copy that link.
4. Open a web browser on your computer (not the Pi), paste the link, and hit Enter.
5. Log in to your Google, Microsoft, or GitHub account.
6. Authorize the device (it will likely show up as pi hostname: adguard).

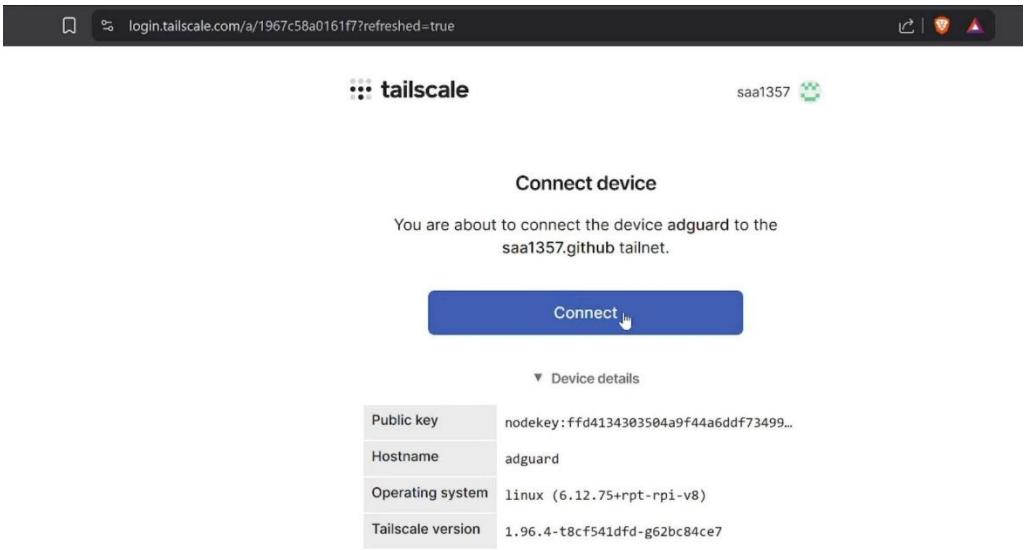

Fig.31: Connect to tailscale

7. Once you click "Connect," go back to the SSH terminal. You should see a message saying Success.

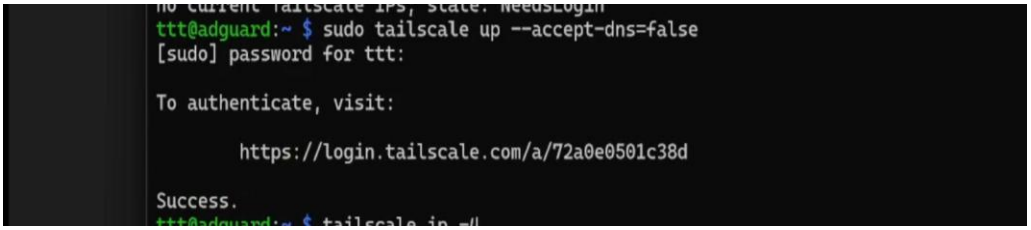

Fig.32: Connection successful

c. Verify Status by SSH (terminal):

Once authenticated is success, Check that Tailscale is working by verify the connection and note the Pi's new (100.x.x.x) Tailscale IP address by running: tailscale ip -4

If an IP address starts with (100.x.x.x), congratulations! Pi is now on your private Tailscale network.

d. AdGuard Home Integration

By default, AdGuard Home may only monitor your local network. To allow it to process DNS requests from your Tailscale devices, you must update its Access settings(Here you can configure access rules for the AdGuard Home DNS server).

1. Access the Interface: Open the browser and navigate to AdGuard dashboard (192.168.0.101).
2. Modify Allowed Clients:
   - Go to Settings → DNS Settings.

- Scroll to the “Access settings” section and add the Tailscale subnet: (100.64.0.0/10). This is the correct subnet to allow all Tailscale devices to query this DNS server.
- Also add in the “Access settings” section the LAN subnet: (192.168.0.0/24). This allows all of the local device. Scroll down and click save configuration.
- Add (192.168.0.100) for Initial Setup/Testing: You should add this address if you want to ensure that one specific local device (the one with the IP 192.168.0.100) has guaranteed access to the AdGuard DNS server while you are still configuring the rest of your network. For me it's a laptop IP that was used to set up this device.

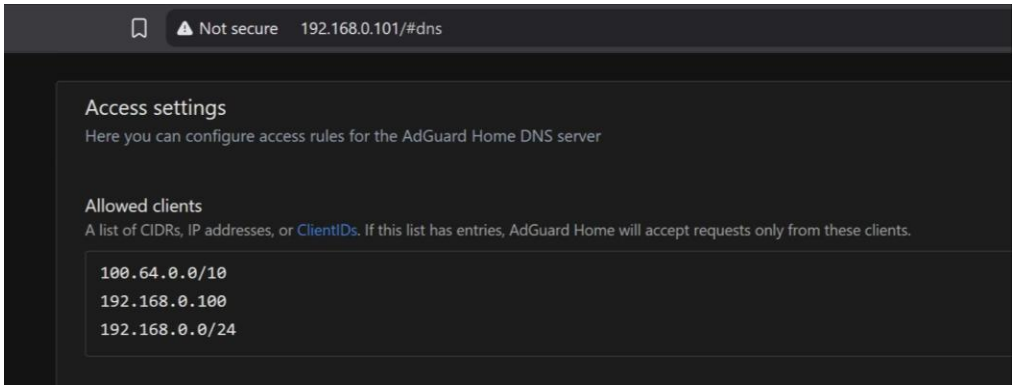

Fig.33: Tailscale Subnet and LAN Subnet

Ensure to scroll down and click save configuration.

e. Global DNS Configuration

To ensure all devices on your Tailnet use the Pi for ad blocking, configure the Tailscale Admin Console.

- Disable Key Expiry: In the Tailscale Admin Console, locate your Pi, open the machine settings (three dots menu), and select Disable Key Expiry. This ensures your Pi remains permanently connected without requiring reauthentication every six months.

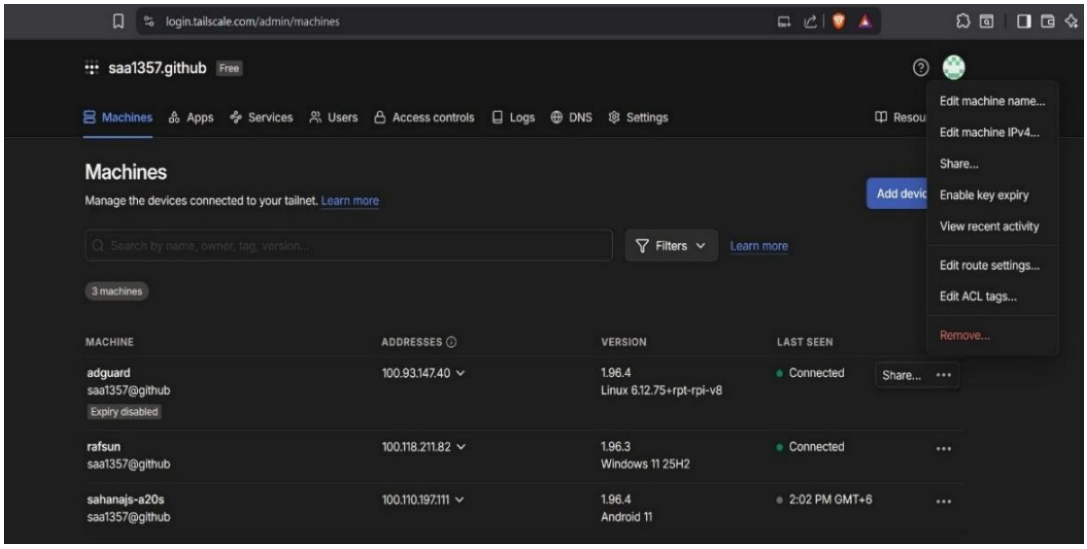

Fig.34: Tailscale manager

- Set Nameservers:
    1. Navigate to the DNS tab in your Tailscale dashboard.
    2. Under Global Nameservers, select Add nameserver → Custom.
    3. Enter the (100.x.x.x) tailscale IP address of your Pi (100.93.147.40).

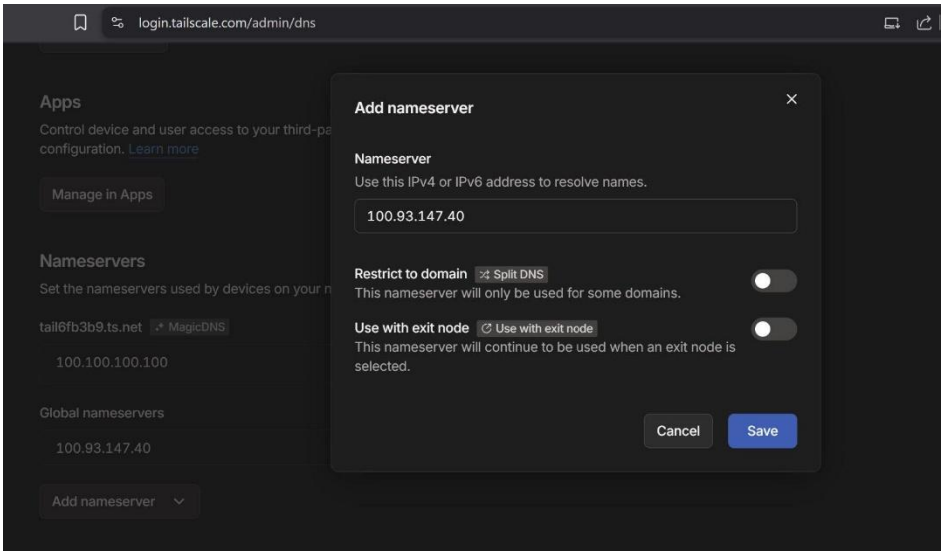

Fig.35: Nameserver

- Enable Override: Toggle the Override Local DNS switch to ON. This forces your devices to use AdGuard regardless of their physical location.

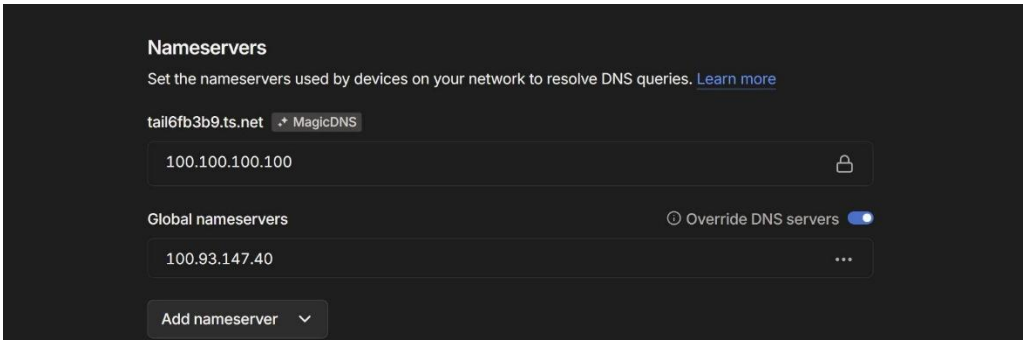

Fig.36: Override DNS severs

4.4 Extra Settings

a. DNS rewrite

Also known as a "DNS Override" is a feature in AdGuard Home that allows you to point a specific domain name to a different IP address than its official one.
To set this up on the current system:

- Access the Dashboard: Open your AdGuard Home web interface (e.g., http://192.168.0.101).
- Navigate to Settings: Click on Filters in the top menu and select DNS rewrites.

- Add a New Rule:
    1. Click Add DNS rewrite.
    2. Domain name: Enter the URL you want to change (e.g., adguard.lan).
    3. IP address: Enter the destination IP (e.g., 192.168.0.101).

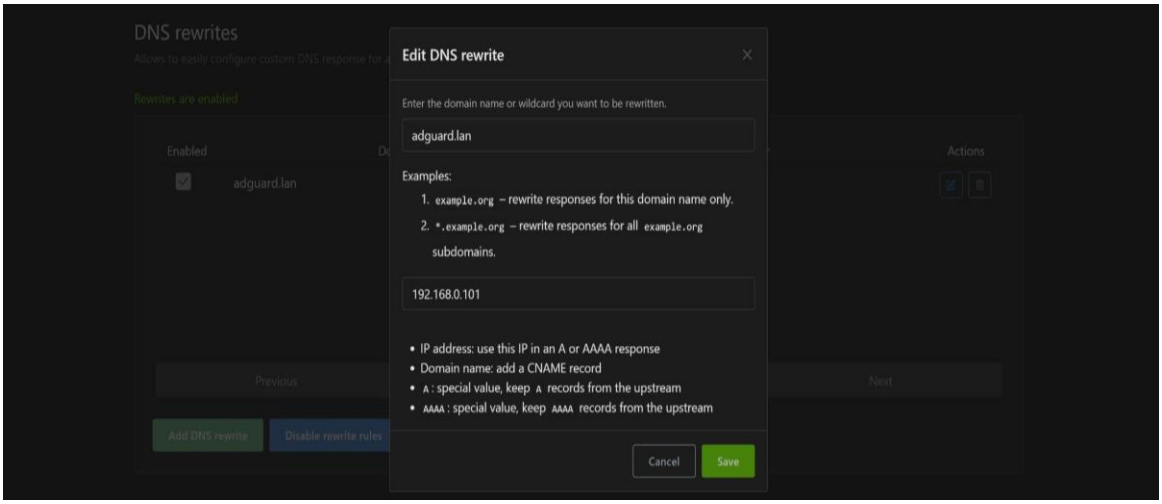

Fig.37: DNS rewrite

- Save: Once saved, click **enable rewrite**. Any device using your Pi for DNS will now follow this custom path.
- To check if the 192.168.0.101 rewrite to adguard.lan

  Open terminal

  Use command: **nslookup adguard.lan** Or

  Use command: **ping adguard.lan**

```
PS C:\Users\HP> nslookup adguard.lan
Server:  UnKnown
Address:  192.168.0.101

Non-authoritative answer:
Name:    adguard.lan
Address:  192.168.0.101
```

Fig.38: Check rewrite

b. DNS blocklists

AdGuard Home will block domains matching the blocklists. A DNS Blocklist (also known as a "DNS filter" or "hosts file") is a curated list of domain names associated with advertisements, trackers, malware, or adult content. When a device on the network requests a domain (e.g., ads.example.com), AdGuard Home checks these lists; if a match is found, it "sinks" the request by blocking it before the advertisement can even begin to load.

How to Configure Blocklists to add or manage these lists on your Pi Zero 2 W setup:

- Access the Dashboard: Open your web browser and navigate to the AdGuard Home dashboard.
- Navigate to Filters: Click on Filters in the top menu and select DNS blocklists.
- Add a List:
    1. Click Add blocklist.
    2. Choose from the list: Select from preapproved popular lists like OISD or Hagezi.
    3. Add a custom list: Paste a URL for a specific list found online (e.g., a specialized list for Smart TVs).
- Set Update Interval: In Settings → General settings, ensure the "Filter update interval" is enabled. For a Pi Zero 2W, a 12-hour or 24-hour interval is recommended to reduce background CPU spikes.
- Test the Connection: Visit a known ad heavy site(MSN) and check the Query Log in dashboard to see domains being blocked in real time. Check the ad blocker test website (https://adblock.turtlecute.org).

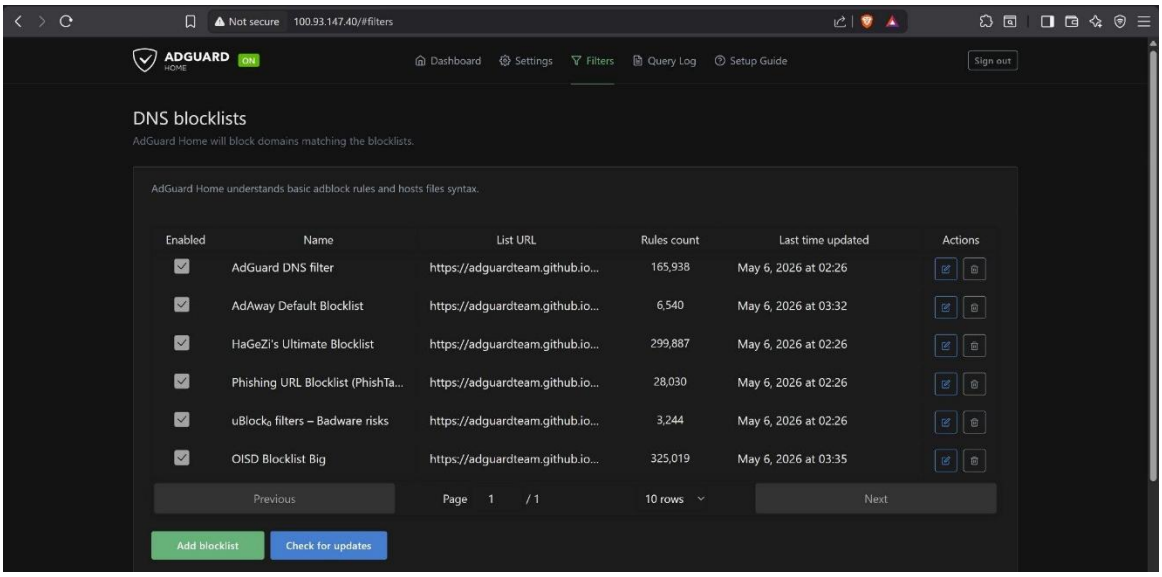

Fig.39: DNS blocklist

Performance Optimization Guidelines

To maintain system stability on resource-constrained hardware, monitor the volume of active "Rules" within the filters. For an optimal balance between network protection and hardware responsiveness, the total rule count should ideally remain between 500,000 and 1,000,000. Utilizing consolidated lists like OISD or Hagezi is recommended to ensure comprehensive coverage without exceeding the memory constraints of the 512MB RAM environment.

4.5 Blocked services & Scheduling on Global DNS

The Blocked Services feature allows users to quickly block popular sites, web platforms and social media services. This method bypasses the need for manual URL entry by utilizing an internal database of service-specific domains, ensuring comprehensive mitigation of targeted platforms.

Implementation and Scheduling

For hardware with limited resources, such as the Raspberry Pi Zero 2 W, utilizing this feature is more efficient than maintaining massive, manual DNS blocklists.

- Select Services: Within the administrative interface, administrators can check specific boxes for the services they intend to restrict, such as Social Media or Gaming platforms. Scroll below and click save to update block service.
- Application Scope: While service blocking can be applied globally, the scheduling functionality is most effective when applied to specific Persistent Clients to create customized browsing environments for different users.
- Define the Schedule: A scheduling grid is available below the service list to specify precisely when these restrictions are active.
    1. Active Blocking: Clicking and dragging on the calendar grid highlights the specific hours and days when the services should be blocked.
    2. Pause State: Unmarked areas on the grid represent "paused" or "unblocked" times, allowing for scheduled access.

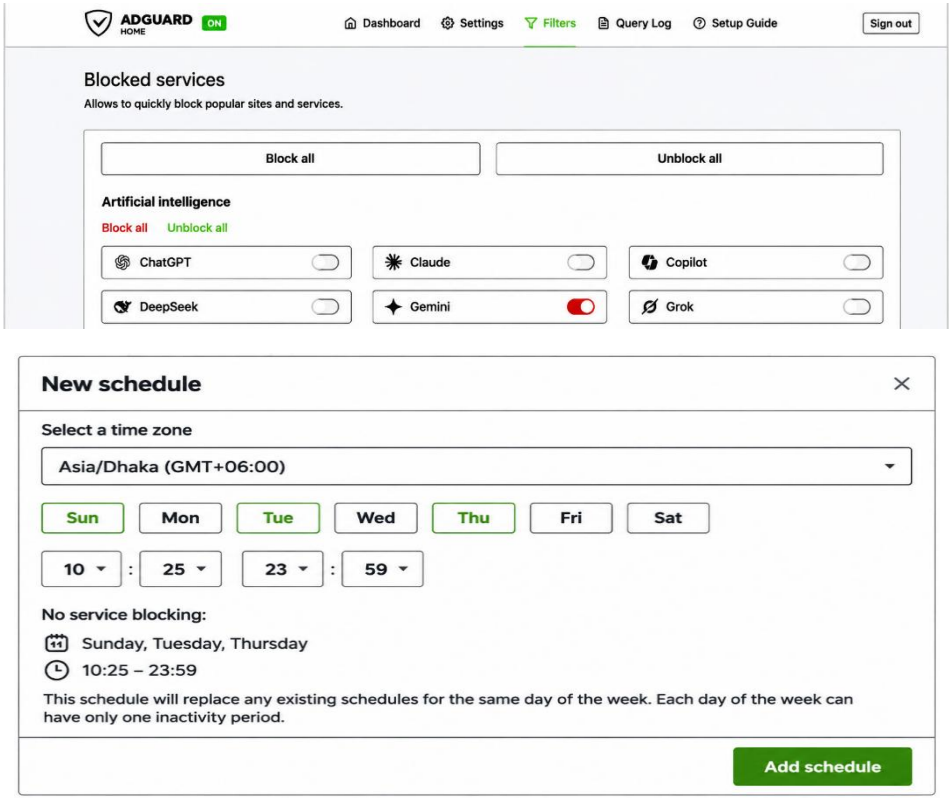

Fig.40: Global DNS blocking service

4.6 Implementation of Parental Controls and Content Filtering in AdGuard Home

The AdGuard Home ecosystem provides robust mechanisms for content moderation through the integration of Persistent Clients and Blocked Services. This architecture allows for granular control over network traffic, enabling the enforcement of parental control policies on a per-device basis.

1. Granular Device Management via Persistent Clients
   Establishing persistent client records is the foundational step for applying specific parental control policies to individual devices.

- Access Client Settings: Within the AdGuard Home dashboard, navigate to Settings and select Client settings.
- Identification of Devices: AdGuard Home identifies clients through various identifiers, including IP addresses, MAC addresses, or CIDR ranges.
- Creation of Persistent Records: By selecting Add client, a permanent record is generated for a specific device (e.g., a child's smartphone or tablet).
- Device-Specific Parameters: Once a persistent record is established, unique filtering rules can be applied specifically to that client, independent of the global network settings.

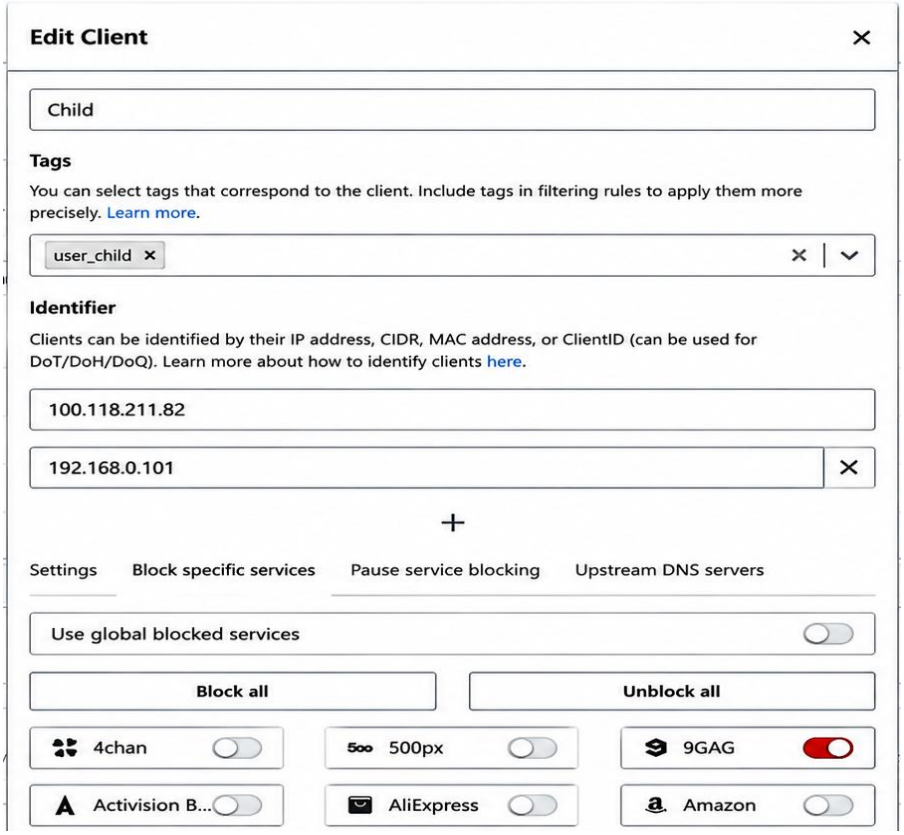

Fig.41: Client settings for individual blocking services

2. Service-Level Mitigation via Blocked Services
   The blocked services feature serves as a streamlined administrative tool for the rapid restriction of popular platforms and social media networks.

- Global vs. Client-Specific Blocking: While service blocking can be applied globally to the entire network, it is most effective when paired with Persistent Clients to create customized browsing environments for different users.
- Rapid Implementation: Within the Client settings for a specific persistent client, the Blocked services tab provides a toggle based interface to instantly restrict access to services such as:
    1. Social Media: YouTube, TikTok, Instagram, and Facebook.
    2. Gaming Platforms: Roblox, Discord, and Twitch.
    3. Shopping and Streaming: Amazon, eBay, Netflix, and Disney+.
- Efficiency: This method bypasses the need for manual URL entry by utilizing AdGuard's internal database of service-specific domains, ensuring comprehensive mitigation of the targeted platform.

4.7 Testing and Troubleshooting

To ensure your Raspberry Pi Zero 2 W based network shield remains stable and effective, you should focus on three main areas: hardware health, software performance (AdGuard Home), and connectivity (Tailscale).

a. **Hardware Health Monitoring**
   Since the Pi Zero 2 W is a low power device, it is susceptible to power instability and overheating.

- Check Power Stability: The Raspberry Pi firmware keeps a record of "Under voltage detected" events. You can query this using the vcgencmd tool. Run this command: **vcgencmd get_throttled**

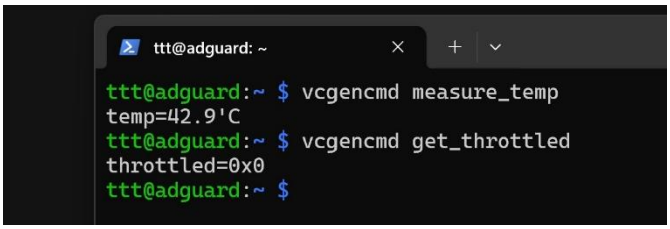

Fig.42: Temperature check for pi health

How to interpret the output:

The command returns a hexadecimal value (like 0x50000). Here is what the key values mean:

1. 0x0: Everything is fine. The power supply is stable.

2. 0x1: Under voltage is currently detected. Pi is not getting enough power right now.
3. 0x50000: Under voltage has occurred in the past since the last reboot.
4. 0x2: The CPU is currently being throttled (slowed down) due to heat.

- Monitor Temperature: Use the command: **vcgencmd measure_temp**

  to ensure the device is not "scorching". If it throttles, the CPU frequency will drop to prevent damage.

- System Resources: Use **htop** or **top** to check real time CPU and memory usage, use **q** to stop. The command **free -h** will show if you are maxing out the Pi's 512MB RAM.

- SD Card Health: Frequent "SD card corruption" messages or boot failures often point to a failing or low quality card.

b. AdGuard Home Troubleshooting

If ads are still appearing or the internet is unreachable, verify your AdGuard configuration.

- Verify Blocking: Visit a website that typically has ads and check if they are missing. Simultaneously, check your AdGuard Dashboard counters to see if blocked queries are increasing.

- Upstream Connectivity: Ensure you have configured Upstream DNS Servers (like 8.8.8.8 or 1.1.1.1); otherwise, AdGuard cannot resolve hostnames it doesn't already know.

- Service Status: If the dashboard won't load, check if the service is active using **sudo systemctl status AdGuardHome**. Click **q** to quit.

Fig.43: AdGuard Status check

- Filter Updates: Ensure your blocklists are enabled and up to date by checking for filter updates in the dashboard settings.

c. Tailscale & Remote Access

Tailscale adds a layer of complexity for remote DNS resolution.

- Check Connection: Use **tailscale status** on Pi to confirm it is connected to Tailnet.

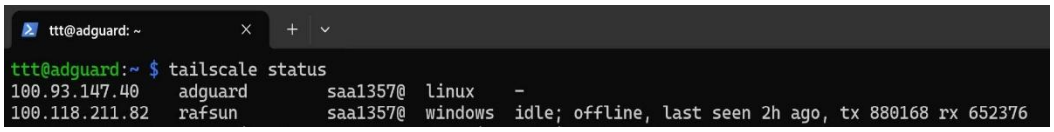

Fig.44: Tailscale Status

- DNS Resolution over Tailscale: If you have no internet while Tailscale is active, ensure you have enabled Permit All Origins in the AdGuard DNS settings.
- Tailscale IP Verification: Confirm that the Global Nameserver IP in Tailscale Admin Console matches the exact 100.x.x.x IP of the Pi.
- Exit Node Speed: If using the Pi as an exit node, expect some speed bottlenecks due to the Pi Zero 2 W's limited CPU and Wi-Fi throughput.
- Ping the Pi: From your computer, try ping 100.93.147.40(tailscale IP for pi). If it responds, the network stack is healthy. Click **ctrl + c** to quit.

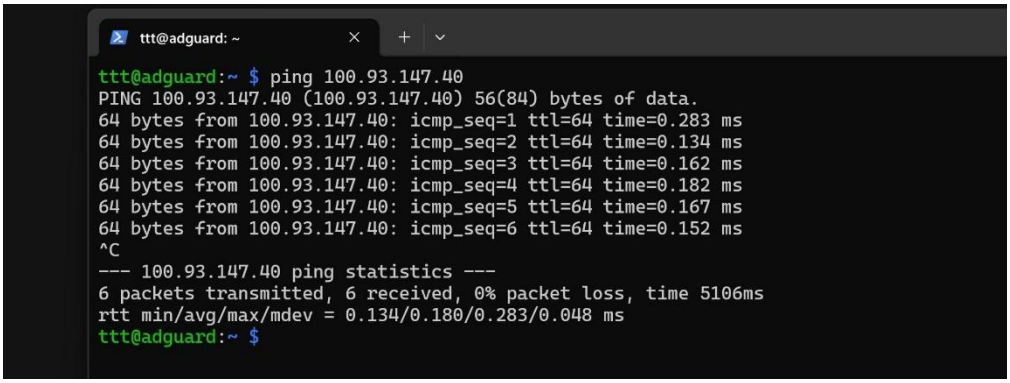

Fig.45: Ping tailscale to check if it's connected

4.8 System Integration

To integrate the Raspberry Pi Zero 2 W with AdGuard Home and Tailscale, you must create a unified system where the Pi acts as a central DNS filter for both local and remote devices.

a. **Hardware Integration and Power Stability**
Before deploying software, ensure the Pi Zero 2 W is physically prepared for 24/7 operation.

- Static IP Assignment: Configure a DHCP reservation on your router for the Pi (e.g., 192.168.0.101) to prevent connectivity loss after power cycles.
- Power Monitoring: Since running headless, monitor for under voltage issues using **vcgencmd get_throttled** to ensure your power supply is sufficient.
- Thermal Management: Maintain an operating temperature below 80°C to prevent CPU throttling, which would slow down DNS resolution.

b. **Software Deployment Layer**
Install and configure the core services to work together on the Raspberry Pi.

- AdGuard Home Installation: Use the automated installation script to deploy the DNS server on the Pi.
- Tailscale Integration: Install Tailscale and authenticate the device to your Tailnet. Disable "Key Expiry" in the Tailscale Admin Console to ensure the Pi stays permanently connected.
- Service Autostart: Both AdGuard Home and Tailscale are configured as system services that will restart automatically if the Pi loses and regains power.

c. Network Configuration and DNS Routing

This is the critical step where you "loop" your devices into the system.

- Local Network Integration: In your router's DHCP settings, set the Primary DNS to your Pi's local IP (e.g., 192.168.0.101). Leave the Secondary DNS empty to prevent devices from bypassing the filter.
- Tailscale DNS Routing: In the Tailscale Admin Console, add your Pi's Tailscale IP (100.x.x.x) as a Global Nameserver and enable "Override Local DNS".
- Access Control: In the AdGuard Home "Access Settings," you must explicitly allow the Tailscale subnet (100.64.0.0/10) and your local subnet (e.g., 192.168.0.0/24) to permit these clients to send queries.

d. Verification and Performance

- Query Logs: Check the AdGuard Home dashboard to verify that requests from your Tailscale devices are being processed.
- Resource Usage: Monitor system health using htop to ensure CPU load stays low (typically below 5% for DNS tasks) and memory usage remains within the 512MB limit.

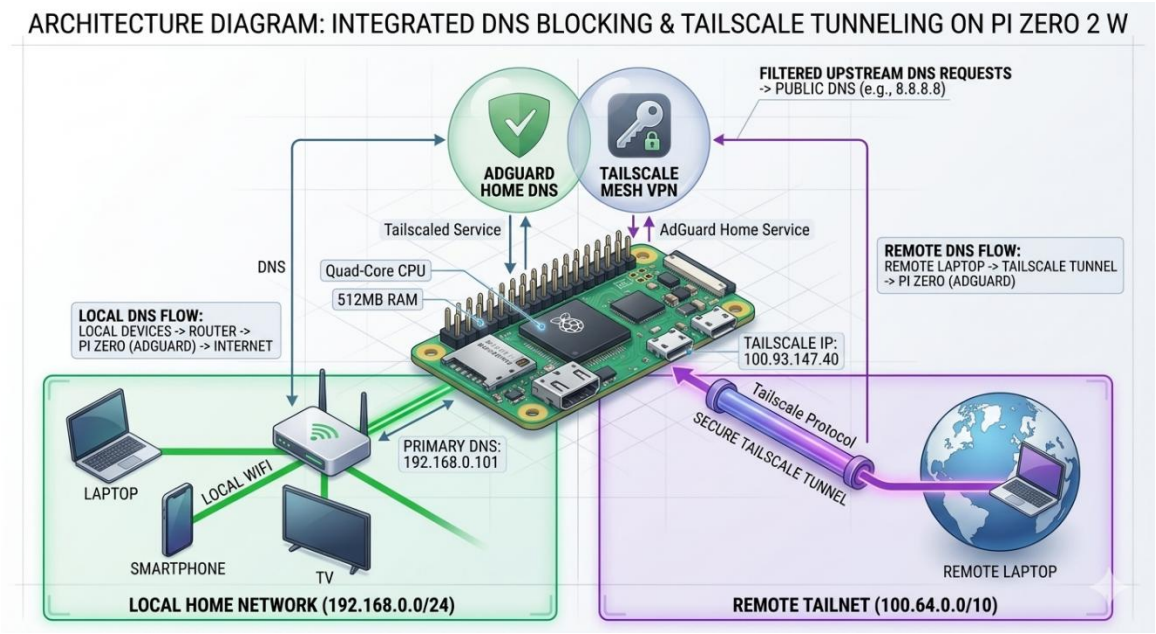

Fig.46: Full system integration of pi and network diagram

CHAPTER 5

RESULT AND DISCUSSION

5.1 Performance of the network and IoT devices

The integration of the Raspberry Pi Zero 2 W into the local network demonstrated high stability and low overhead.

- Hardware Efficiency: The Pi Zero 2 W's quad core CPU proved sufficient for handling multiple DNS queries simultaneously, with average CPU usage remaining below 2% during standard home operations.
- Memory Management: Despite having only 512MB of RAM, the system maintained approximately 200MB of free memory while running both AdGuard Home and Tailscale.
- DNS Latency: Local DNS resolution was near instantaneous (typically under 20ms), while remote resolution via Tailscale added a negligible overhead depending on the user's mobile data connection.
- Power Stability: Testing confirmed that the system is resilient to power loss, with all services (AdGuard Home and Tailscale) automatically restarting upon reboot without manual intervention.

5.2 User experience and feedback

The "set and forget" nature of the deployment resulted in a seamless transition for all household devices.

- Ad Blocking Effectiveness: Users reported a significant reduction in visual clutter on news sites and mobile applications, with AdGuard Home successfully intercepting trackers and advertisements at the network level.
- Remote Connectivity: The Tailscale integration allowed users to maintain the same level of security and ad blocking on mobile devices while using external LTE or public Wi-Fi networks.
- Dashboard Accessibility: The web based UI enabled easy monitoring of "Top Queried Domains" and real time blocking statistics for both local and remote clients.

5.3 Comparison with pi hole

While Pi-hole is a popular alternative, the choice of AdGuard Home offered specific advantages for this project. At this point, AdGuard Home has a lot in common with Pi-Hole. Both block ads and trackers using "DNS sinkholing" method, and both allow customizing what's blocked. AdGuard Home provides a lot of features out of the box with no need to install and configure additional software. It is simple to the point when even casual users can set it up with minimal effort.

The adguard home dashboard is generally considered more modern and provides a more detailed "Query Log" for troubleshooting specific device behaviors.

Table 3. Comparison with pi hole

Feature	AdGuard Home	Pi-Hole
Blocking ads and trackers	✔	✔
Customizing blocklists	✔	✔
Built-in DHCP server	✔	✔
HTTPS for the Admin interface	✔	Kind of, but you'll need to manually configure lighthttpd
Encrypted DNS upstream servers (DNS-over-HTTPS, DNS-over-TLS, DNSCrypt)	✔	✗ (requires additional software)
Cross-platform	✔	✗ (not natively, only via Docker)
Running as a DNS-over-HTTPS or DNS-over-TLS server	✔	✗ (requires additional software)
Blocking phishing and malware domains	✔	✗
Parental control (blocking adult domains)	✔	✗
Force Safe search on search engines	✔	✗
Per-client (device) configuration	✔	✔
Access settings (choose who can use AGH DNS)	✔	✗

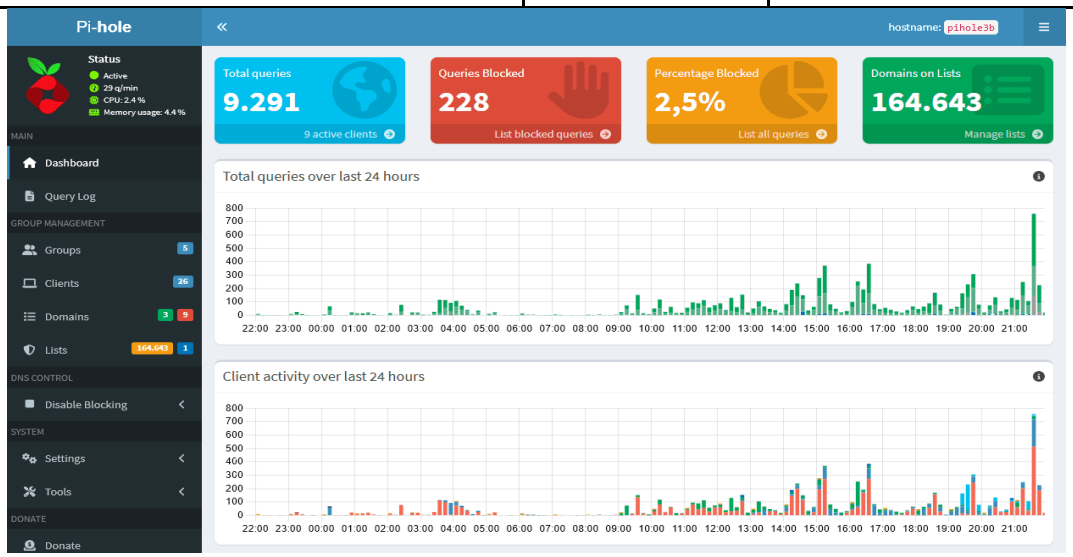

Fig.47: Pi Dashboard

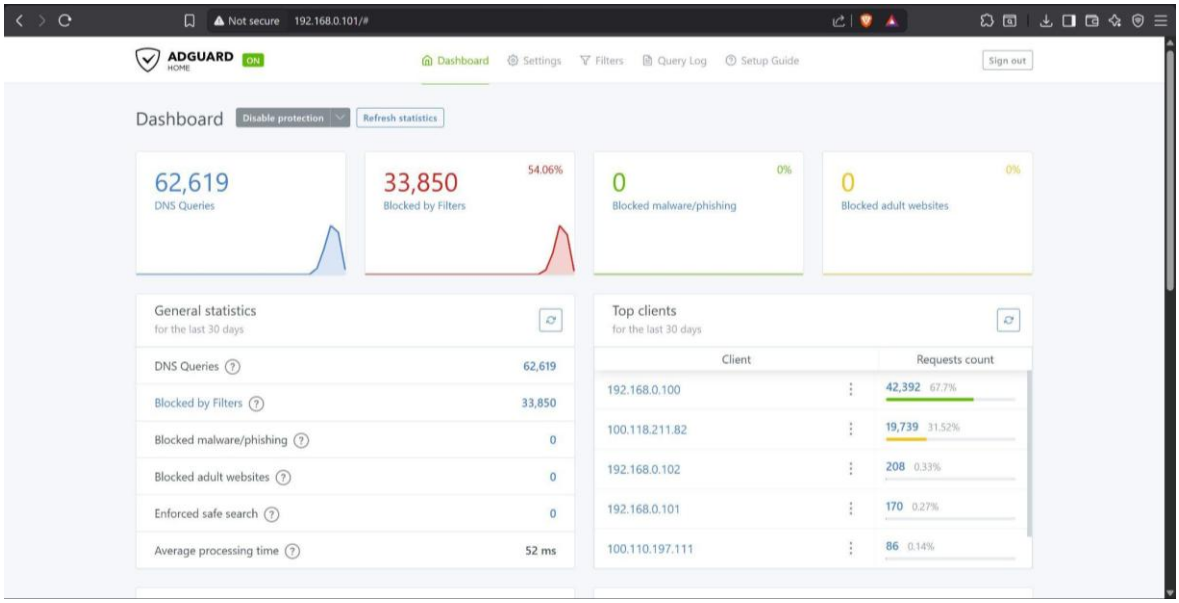

Fig.48: AdGuard Dashboard

5.4 limitation and challenges encountered

Several technical hurdles were identified during the implementation of the Pi Zero 2 W setup.

- Hardware Constraints: The micro USB power input on the Pi Zero 2 W is sensitive; poor quality cables led to initial "Under voltage" warnings which required upgrading to a 5V 2.5A power supply.
- Wi-Fi Throughput: Because the Pi Zero 2 W uses 2.4GHz Wi-Fi, it can become a bottleneck if used as a full "Tailscale Exit Node" for high bandwidth traffic, though it remains perfectly fast for DNS only traffic.
- DNS Bypassing: Some modern devices and browsers (using "Secure DNS" settings) initially bypassed the Pi, requiring manual disabling of "Private DNS" on those specific devices to ensure the Pi remained the primary resolver.
- IP Conflicts: Initial setup faced issues where the Pi's IP address changed after a router reboot; this was resolved by implementing a permanent DHCP reservation at 192.168.0.101.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

The implementation of a network wide DNS filtering and remote access system using a Raspberry Pi Zero 2 W, AdGuard Home, and Tailscale has proven to be a highly effective and resource efficient solution for modern network security and privacy needs.

Through this project, it was demonstrated that low power IoT hardware is capable of serving as a robust security gateway for both local and remote environments. The following key milestones were achieved:

Integrated Privacy Shield: By leveraging AdGuard Home, the system successfully intercepted and blocked advertisements and tracking scripts at the DNS level, significantly improving page load speeds and reducing data consumption across all connected devices.

Seamless Remote Integration: The deployment of Tailscale transformed the Raspberry Pi from a local only utility into a global private nameserver. This allowed for a consistent "home" security posture on mobile devices, even when connected to untrusted public Wi-Fi or cellular networks.

Advanced Access Control: The configuration of CIDR ranges, specifically 192.168.0.0/24 for local traffic and 100.64.0.0/10 for the Tailscale mesh network, ensured that only authorized clients could utilize the DNS resources, preventing unauthorized external relaying.

Performance Optimization: Utilizing the quad core capabilities of the Pi Zero 2 W and prioritizing vectorization in related data processing tasks ensured that the system maintained a negligible impact on network latency. The system's stability was further validated by an uptime of over five days with minimal CPU and thermal stress.

In summary, this architecture provides a cost effective alternative to commercial enterprise solutions, granting the user full sovereignty over their data and network traffic without the need for expensive hardware or subscription based services.

6.2 Future Works

While the current system is stable and functional, several avenues for expansion and technical refinement exist to further enhance the utility and security of the IoT deployment:

Implementation of DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT): Future iterations will focus on encrypting upstream DNS requests. While the Pi currently filters local requests, the final "hop" to upstream providers (like Google or Cloudflare) remains visible to ISPs; implementing DoH/DoT within AdGuard Home would solve this privacy gap.

Unbound Integration for Recursive DNS: To achieve total independence from third party DNS providers, a future goal is to install and configure Unbound on the Pi. This would allow the Pi to contact Root DNS servers directly, acting as its own recursive resolver and further enhancing privacy.

Redundancy and High Availability (HA): To eliminate the Raspberry Pi as a single point of failure, a secondary Pi (or a virtualized instance) could be deployed in an "AdGuard Home Sync" configuration. This would ensure that if the primary Pi Zero 2 W loses power or fails, the network remains online.

Hardware Expansion (Pi-Hole Parallel Testing): While AdGuard Home was selected for its native parental controls and modern UI, a comparative long term study using a dual boot or containerized environment could yield deeper insights into performance deltas between AdGuard and Pi-hole under heavy load.

Automated Security Auditing: Developing a Python based monitoring script using libraries like NumPy and OpenCV is a potential area of interest. While OpenCV is traditionally for image processing, it could be adapted for visual log analysis or creating automated graphical reports of network intrusion attempts visualized through the AdGuard dashboard.

Tailscale Exit Node Optimization: Future work will explore optimizing the Pi Zero 2 W's network stack to better handle "Exit Node" traffic. This involves tuning the Linux kernel to prioritize Tailscale encrypted packets, allowing for a faster full tunnel VPN experience on the limited 2.4GHz Wi-Fi hardware.

REFERENCES

[1] Al-Shaer, E. S., & Hamed, H. H. (2003). Firewall policy advisor for anomaly discovery and rule editing. 8th International Symposium on Integrated Network Managemen.

[2] Anderson, S. P., & Gans, J. S. (2011). Platform siphoning: Ad-avoidance and media content. American Economic Journal: Microeconomics.

[3] Bratko, A., et al. (2006). Spam filtering using statistical data compression models. Journal of Machine Learning Research.

[4] Chen, Z., et al. (2006). A novel web page filtering system by combining texts and images. IEEE/WIC/ACM International Conference on Web Intelligence.

[5] Du, R., et al. (2003). Web filtering using text classification. The 11th IEEE International Conference on Networks.

[6] Hu, W., et al. (2007). Recognition of Pornographic Web Pages by Classifying Texts and Images. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[7] Pappas, V., et al. (2012). On the Feast of Filters: Evaluating DNS-based Ad Blocking. (Related Work for context on modern hardware/software comparisons).

[8] Falahrastegar, M., et al. (2015). The Unbearable Lightness of Ad-Blocking. Proceedings of the 2015 ACM Conference on Internet Measurement Conference.

[9] Ikram, A., et al. (2019). IoT device identification using network traffic analysis. IEEE Conference on Communications and Network Security (CNS).

[10] Miettinen, M., et al. (2017). IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security.

[11] Nikiforakis, N., et al. (2014). The Privacy and Security Implications of Ad-Blocking Extensions. Proceedings of the 2014 IEEE Symposium on Security and Privacy.

[12] Naylor, D., et al. (2015). The Costs of Sympathy: Evaluating the Relationship between Ad Blocking and Web Resource Usage. Proceedings of the 2015 ACM Conference on Internet Measurement Conference.

[13] Vixie, P. (2012). DNS Response Policy Zones (RPZ). The IETF Journal.