**PureNet Pi**

**Project Title: Unified DNS and Firewall Blocklist Manager (PureNet Pi)**

Author: Xavier Christian, Chance Foster

Project Supervisor: Dr. Michael Reynolds, Dr. Enoch Solomon

Date: 5/2/25

**Abstract**

This project presents the design and implementation of a unified Python-based DNS and firewall management tool (PureNet Pi) for Raspberry Pi devices. The tool enables users to block malicious websites and devices using domain and IP address blocklists via a simple CLI interface. It integrates a custom DNS server using the *dnslib* library to intercept and block domain queries. It uses SSH to manage firewall rules on a separate Pi acting as a network firewall. This system allows real-time, scriptable control of DNS and IP filtering. Results demonstrate successful interception of blacklisted DNS queries and blocking of IP traffic at the firewall level.

**Table of Contents**

1. Title Page
2. Abstract
3. Table of Contents
4. Introduction & Related Work
5. Problem Definition
6. Methodology
7. Concepts Considered
8. Concept Selection
9. Design and Implementation
10. Conclusion/Summary
11. Bibliography
12. Appendices

**Introduction & Related Work**

Modern networks face constant threats from malicious domains, trackers, malware, and compromised devices. As the number of connected devices grows especially with the rise of IoT and smart home networks the demand for simple yet powerful network-level filtering becomes more important. Malicious actors often exploit DNS to redirect users to phishing sites or distribute malware via seemingly legitimate domains.

Traditional ad blockers and antivirus software operate at the device level and are often limited by user permissions or browser environments. DNS-based filtering, on the other hand, allows administrators to intercept and block unwanted or dangerous queries at the network level before they ever reach a client. This not only improves security but also conserves bandwidth and speeds up browsing by eliminating unnecessary third-party content. However, DNS filtering alone does not prevent devices from directly communicating with malicious IP addresses or being involved in network-level attacks.

PureNet Pi addresses these limitations by integrating DNS filtering with remote IP-level firewall management via SSH and iptables. The system was developed to be lightweight, transparent, and customizable, making it suitable for home labs, classrooms, or small offices. Related tools include Pi-hole, AdGuard, and enterprise firewalls. Pi-hole and AdGuard primarily focus on DNS filtering with GUI-based blocklist control. Enterprise firewalls like pfSense or Cisco ASA offer more advanced IP filtering but are not accessible to most home users. Few tools combine the simplicity of DNS-level blocking with the power of remote firewall control on low-cost hardware.

**Problem Definition**

**Goals and Deliverables:**

* Create a Python script that integrates DNS filtering and firewall management.
* Provide a CLI for adding/removing/checking blocked domains and IPs.
* Persist blocklists and log DNS queries.
* Automate DNS startup as a systemd service.

**Specifications and Constraints:**

* Enabled features: DNS resolution, domain and IP blocking, SSH firewall updates.
* Not enabled: Web GUI, HTTPS inspection, device discovery.

**Impacts:**

* *Local:* Improves security on small networks without needing enterprise tools.
* *Global:* Contributes to open-source network security solutions.

**Concerns:**

* Ethical: Avoid misuse for surveillance or censorship.
* Legal: Ensure it does not violate ISP terms or filter legitimate traffic.
* Social: Helps non-technical users secure their networks.
* Security: Protects against malicious domains and lateral movement.

**Methodology**

The methodology for this project followed a modular and iterative approach focused on building a secure and efficient DNS and firewall management tool using Raspberry Pi. The system was divided into two key components: a DNS server and a firewall controller, each running on separate Raspberry Pi devices. This division of roles ensured that each Pi could specialize in its function, simplifying testing and reducing overhead. The DNS Pi was responsible for handling domain queries using a custom Python resolver, while the Firewall Pi accepted remote commands to manage IP-level filtering with iptables.

The DNS functionality was implemented first, starting with a Python-based resolver class that intercepted DNS queries and checked them against a local blocklist stored in a text file. If a domain matched the blocklist, the query was suppressed or redirected to a null address (0.0.0.0); otherwise, it was forwarded to an upstream DNS provider. All queries were logged to a local file with timestamps and client IPs for audit purposes. This logging mechanism proved crucial for confirming that DNS filtering was working as expected and for tracking usage patterns during testing.

Simultaneously, the IP blocking system was developed by writing a remote execution handler in Python that sent SSH commands to the Firewall Pi. Commands dynamically inserted or removed iptables rules from both the INPUT and FORWARD chains, ensuring that blocked IPs were entirely cut off from accessing or being accessed by the network. A command-line interface was built to allow users to manage the blocklists interactively. Once functionality was verified, the DNS script was configured to start automatically at boot using a persistent service, completing the full deployment of the system.

**Concepts Considered**

**Programming Language:** Python vs Bash vs Go

* Python: Easy scripting, strong libraries
* Bash: Good for quick CLI, hard to scale
* Go: Fast but overkill for CLI scripts

**DNS Tool:** dnslib vs dnsmasq

* dnslib: Full control in Python
* dnsmasq: Lightweight but limited Python integration

**Firewall Control:** SSH + iptables vs REST API

* SSH: Simple, secure, standard on Linux
* REST API: More complex, needs additional software

**Blocklist Storage:** Text file vs SQLite

* Text file: Human-readable, easy to edit
* SQLite: Scalable but overkill

**Concept Selection**

* Python was selected for speed of development and integration.
* dnslib was used for Python-native DNS manipulation.
* SSH with iptables was chosen for simplicity and compatibility.
* Text files were selected for portability and transparency.

**Design and Implementation**

The Python script starts a DNS server on port 53, checks each request against a domain blocklist, and logs the query. IP addresses are compared post-resolution. IP-level rules are pushed via SSH to a firewall Pi that applies iptables rules. The CLI supports:

* Add-site, remove-site, check-site, list-sites
* Add-ip, remove-ip, check-ip, list-ips

**Conclusion/Summary**

This project successfully delivers a unified DNS and firewall management tool with real-time CLI interaction, persistent logging, and remote rule enforcement. It simplifies network security for users running Raspberry Pi devices and demonstrates the feasibility of combining local DNS filtering with IP-based firewall blocking.

**Bibliography**

1. *Pi-hole Documentation*. Pi-hole, https://docs.pi-hole.net/. Accessed 7 May 2025.
2. *iptables(8) - Linux man page*. man7.org, https://man7.org/linux/man-pages/man8/iptables.8.html. Accessed 7 May 2025.
3. *dnslib — A library to encode/decode DNS packets*. Read the Docs, <https://dnslib.readthedocs.io/en/latest/>. Accessed 7 May 2025.
4. Mockapetris, Paul. *Domain Names - Implementation and Specification*. RFC 1035, RFC Editor, 1987, https://www.rfc-editor.org/rfc/rfc1035. Accessed 7 May 2025.
5. *ssh(1) — OpenSSH SSH client*. OpenBSD Manual Pages, https://man.openbsd.org/ssh.1. Accessed 7 May 2025.
6. *Getting Started with Raspberry Pi OS*. Raspberry Pi Foundation, https://www.raspberrypi.com/documentation/computers/getting-started.html. Accessed 7 May 2025.
7. *What Is DNS?*. Cloudflare, https://www.cloudflare.com/learning/dns/what-is-dns/. Accessed 7 May 2025.

**Appendices**

**User Manual:**

1. Run sudo python3 Pi\_Hole\_Fire\_Wall\_Manager.py
2. Use add-site example.com or add-ip 192.168.0.10
3. Use list-sites and list-ips to view blocklists
4. Use remove-site or remove-ip to unblock

**System Diagram:**

* Client → DNS Pi → Firewall Pi → Internet
* Blocked domains drop DNS responses
* Blocked IPs are dropped via iptables

**Code:** See attached script file Pi\_Hole\_Fire\_Wall\_Manager.py

Code is also on github: https://github.com/XavierC003/PureNetPI.git