NetEAGLE

A home-network security gateway for non-technical users

Yulan Galagoda
BSc (Hons) Computer Security — Final-year Individual Project
University of Plymouth · 2024

Project report

Contents

Executive summary 3
1. Introduction and problem 3
2. Objectives 3
3. Research-led design 4
Background 4
User research 4
4. System architecture 4
5. Implementation 5
Gateway and control surface 5
The three security features 5
App, cloud and resilience 5
6. Results and evaluation 6
7. Challenges and lessons 6
8. Future work 6
9. Conclusion 7
References 7

Executive summary

NetEAGLE brings enterprise-style network monitoring to ordinary homes — without demanding that the owner understand networking or security.

NetEAGLE is a self-contained home-network security gateway built on a Raspberry Pi placed inline with a household's internet traffic. It unifies three defensive capabilities — automated host discovery (Nmap), firewall management (UFW), and intrusion detection and prevention (Suricata) — behind a deliberately simple mobile application. Network state and security alerts sync in near-real time through a cloud database, and the user controls the entire system with one-tap actions.

The result is an appliance that gives non-technical users meaningful visibility and control over their network — a segment that enterprise tooling ignores and that endpoint antivirus cannot protect, because antivirus only defends the single device it runs on and only after a threat has already crossed the network. This report sets out the problem, the user research that shaped the design, the system architecture, the implementation across hardware, software and cloud, and an honest evaluation against the original objectives.

1. Introduction and problem

Network monitoring is a primary means of keeping a network secure. Enterprises run dedicated network and security operations centres, or outsource monitoring to trusted providers, because the data they hold is valuable. Home networks now hold comparably sensitive data — passwords, banking details, and, since the shift to remote work, direct connections into corporate systems — yet they are rarely monitored and seldom patched.

The home gateway is the single most critical device on the network and, in practice, the most neglected: routers frequently never receive firmware updates, and a large share of owners run insecure access points. Endpoint antivirus does not close this gap. It protects only the device it is installed on, can act on a threat only once it has already reached that device, and leaves any unprotected device — or any router-level compromise — exposed. Real-world malware that has targeted home networks, from the Conficker worm to VPNFilter and session-hijacking tools, shows the stakes.

Tools that could help, such as PRTG or Nessus Home, exist — but they assume networking knowledge the typical home owner does not have, and demand more time and effort than most owners are willing to spend. The gap NetEAGLE addresses is a network-level, proactive, genuinely easy tool aimed squarely at the non-technical home user.

2. Objectives

The project set out to give a home-network owner the ability to monitor and secure their own network, under an overriding design constraint: it had to be usable by someone with very little technical knowledge. Four objectives were defined:

1. Let owners monitor network traffic and be notified of abnormal behaviour from any connected device.
2. Help owners take the necessary action on the security issues that arise.
3. Let owners check their network's security quickly, without significant time or effort.
4. Turn highly vulnerable home networks into secured, monitored ones — a safer internet experience for ordinary users.

3. Research-led design

Background

A review of home-network security established the standard safeguards — Wi-Fi encryption, firewalls, firmware updates, device hardening, monitoring and user education — and why they fail in practice: each assumes a level of networking and security knowledge the typical owner lacks, so they go unapplied and the network stays exposed.

User research

To ground the design in real expectations, I ran a short six-question survey of home-network owners, written in plain language for a non-technical audience. The findings directly shaped both scope and architecture:

- Strong interest in intrusion detection/prevention and firewall features.
- Very low interest in vulnerability assessment — so it was removed from scope and deferred to future work.
- A clear demand for real-time updates — which led to a cloud real-time database rather than local storage.
- A strong preference for a mobile app as the point of contact with the system.

4. System architecture

A Raspberry Pi is placed inline between the ISP connection and the home's devices, so all traffic flows through it. Because it sits in the path, it can both observe traffic and enforce controls. On the device, Nmap performs host discovery, UFW provides the firewall, Suricata provides intrusion detection and prevention, and a Flask REST API exposes these as simple actions, glued together with Python and shell scripts. Alerts and state are written to a Firebase Realtime Database, whose updates push notifications to a Flutter mobile app; the app, in turn, calls the Flask API to trigger actions on the Pi.

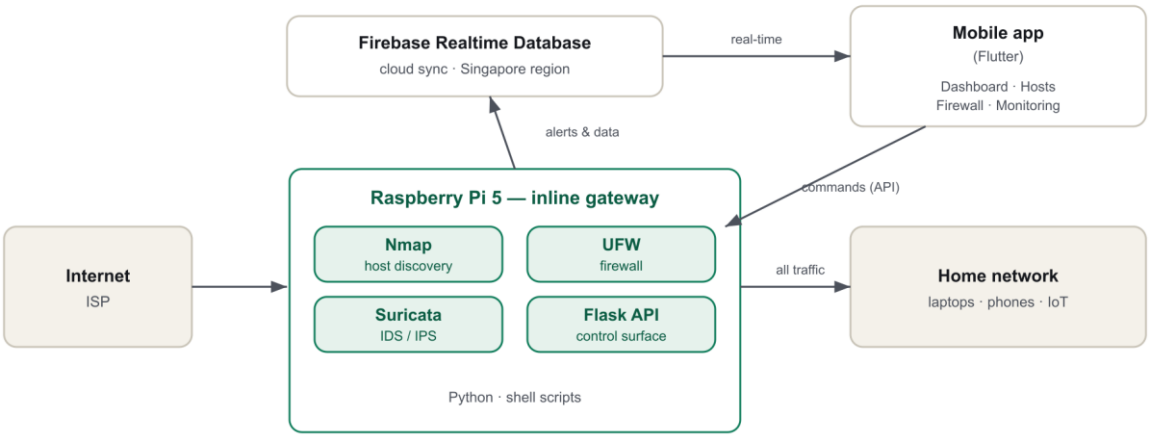

Figure 1 — NetEAGLE system architecture.

5. Implementation

Gateway and control surface

For demonstration the Raspberry Pi was configured to perform the router role itself — taking an uplink on its wireless interface and serving the LAN over Ethernet — so it could sit inline without a separate routing device. The Flask API is registered as a system service so it starts automatically on boot and listens for the app's HTTP requests, mapping each to the relevant script and returning a status the app can act on.

The three security features

Each feature follows the same clean pattern: the app calls an API endpoint, a shell script runs the underlying tool, and a Python helper parses the output and uploads it to Firebase. Host discovery runs an Nmap scan over the local range and lists the connected devices. The firewall surfaces the UFW rule list in the app, with refresh and on/off toggle actions. Intrusion detection — presented to the user as "Monitoring" — surfaces Suricata's alerts, with refresh, clear and toggle actions.

App, cloud and resilience

The Flutter app keeps the interface minimal: a dashboard shows host and alert counts and routes to the Hosts, Firewall and Monitoring pages, with a loading screen for smooth transitions and a single error page for connection failures. The cloud database uses Firebase via FlutterFire; a small bootstrap step ensures the required Python package is present in a virtual environment so a first-time user encounters no setup errors.

Components at a glance

Component	Technology	Role
Inline gateway	Raspberry Pi 5 (Linux)	Sits between the ISP and the home network; sees and filters all traffic.

Component	Technology	Role
Host discovery	Nmap	Scans the local range to enumerate connected devices for the user.
Firewall	UFW	Simple, policy-templated firewall the user can review and toggle.
Intrusion detection	Suricata (IDPS)	Real-time detection/prevention with rules for common home threats.
Control surface	Flask REST API (Python)	Auto-started service exposing device, firewall and IDS actions.
Cloud sync	Firebase Realtime Database	Holds alerts and state; updates push notifications to the app.
Mobile app	Flutter (Dart)	One-tap dashboard for hosts, firewall and monitoring.

6. Results and evaluation

The project delivered a functional, end-to-end system that meets all four objectives. It discovers hosts, manages the firewall, detects and prevents intrusions through Suricata, and presents everything in a simple app with near-real-time alerts and one-tap control. Deployment is effectively plug-in: the user connects the Pi inline and needs no further configuration, with the device serving addresses from a small pool and a reserved static range for fixed devices such as printers.

Evaluation was functional and qualitative — judged against the stated objectives and the non-technical-user constraint rather than a quantitative performance benchmark. On that basis the system succeeds: the no-configuration install, the deliberately small and fast app, and the real-time database together deliver the experience the user research asked for.

7. Challenges and lessons

- No dedicated routing device was available, so router functions were implemented on the Pi itself — turning a constraint into a self-contained design.
- Establishing the mobile app's connection to Firebase required iterating through several libraries and configurations before a stable real-time link was achieved.
- Automatic uploads from the Pi to Firebase exceeded what the low-end Pi could sustain, so uploads were made user-triggered; on higher-spec hardware they can be scheduled as a background service.

The work reinforced the value of thorough planning and requirements analysis, clear communication, disciplined time management, and more systematic testing.

8. Future work

1. Vulnerability analysis — deferred from the original scope; ideally simplified for non-technical users with machine learning.

2. Two-factor authentication and encryption of data in transit and at rest.
3. AI/ML-assisted alerts — plain-language explanations and suggested responses for each alert.
4. Monitoring history — retained over time, given sufficient cloud storage.

9. Conclusion

NetEAGLE demonstrates that a network-level, genuinely user-friendly home security appliance is achievable with accessible components — a Raspberry Pi, Flask, Nmap, UFW, Suricata, Firebase and Flutter — and that it meets a real, under-served need. By moving protection from the endpoint to the gateway and hiding the complexity behind a simple app, it lets ordinary households monitor and defend their networks in a way that, until now, only enterprises could.

References

Baugher, M. and Lortz, V. (2011) *Home-network threats and access controls.* Berlin: Springer.

Boeckl, K. et al. (2019) *Considerations for managing Internet of Things (IoT) cybersecurity and privacy risks (NISTIR 8228).* Gaithersburg, MD: National Institute of Standards and Technology.

Ho, J. and Dearman, D. (2010) *Improving users' security choices on home wireless networks.* New York: Association for Computing Machinery.

Lyon, G.F. (2009) *Nmap network scanning: the official Nmap project guide to network discovery and security scanning.* Sunnyvale, CA: Insecure.Com LLC.

Miro (2024) *Architectural diagramming: a complete guide.* Available at: https://miro.com/diagramming/what-is-software-architecture-diagramming/ (Accessed: 10 May 2024).

Open Information Security Foundation (2024) *Suricata user guide.* Available at: https://docs.suricata.io/ (Accessed: 10 May 2024).

Scarfone, K. and Hoffman, P. (2009) *Guidelines on firewalls and firewall policy (NIST SP 800-41 Revision 1).* Gaithersburg, MD: National Institute of Standards and Technology.

Stimpson, T., Liu, L., Zhang, J. and Hill, R. (2012) *Assessment of security and vulnerability of home wireless networks.* Chongqing: IEEE.