Project SafeNet: A Lightweight, Policy-Driven Framework for Zero-Trust Micro-Perimeter Security in Small Office/Home Office (SOHO) Environments

Abstract

The proliferation of unmanaged and insecure Internet of Things (IoT) and Bring-Your-Own-Device (BYOD) endpoints on flat Small Office/Home Office (SOHO) networks has created a critical, unaddressed security vulnerability. These environments are highly susceptible to lateral movement attacks, where a single compromised device becomes a beachhead to attack all other assets on the network. This paper proposes Project SafeNet, a novel, lightweight, open-source framework that democratizes the enterprise-grade Zero-Trust security model for this underserved ecosystem. The core architecture of SafeNet consists of a declarative, human-readable YAML policy engine that automates the configuration of a dynamic, cryptographically secure mesh network built on the WireGuard protocol. The primary objective is to research and develop a readily deployable solution that enforces network micro-segmentation, effectively neutralizes lateral movement threats, and provides a significant, measurable uplift in the baseline security posture for a large and vulnerable user base.

1. Introduction: The Collapsing Perimeter and the SOHO Security Crisis

1.1. The Obsolete Castle-and-Moat Paradigm

The traditional cybersecurity paradigm, widely known as the "castle-and-moat" model, is predicated on a simple architectural assumption: a clear, defensible perimeter separates a trusted internal network from an untrusted external world. For decades, security efforts focused on strengthening this perimeter with firewalls, intrusion detection systems, and other gatekeeping technologies.[1] However, this model is now fundamentally obsolete. The modern digital enterprise no longer has a clearly defined perimeter.[2] The widespread adoption of remote work, the ubiquity of mobile computing, and the explosive growth of the Internet of Things (IoT) have dissolved conventional network boundaries. Employees and devices now require "anytime, anywhere" access to corporate resources, rendering the concept of a trusted internal network a dangerous anachronism.[2] This paradigm shift necessitates a move away from perimeter-centric defenses towards a more data-centric and identity-aware security model.

1.2. The SOHO Network as a High-Value, Low-Effort Target

While large enterprises grapple with the complexities of a dissolved perimeter, the Small Office/Home Office (SOHO) network has emerged as a uniquely vulnerable and attractive target for sophisticated threat actors. These environments, characterized by a chaotic mix of personal smartphones, work laptops containing sensitive data, and a myriad of insecure IoT gadgets, are often connected to consumer-grade routers that lack basic security features.[3] This creates a vast, globally distributed, and highly vulnerable attack surface.

Recent intelligence reports reveal that SOHO networks are not merely collateral damage in broad-based attacks but are actively and strategically targeted. State-sponsored threat actors, such as the People's Republic of China (PRC)-linked groups "Volt Typhoon" and "LapDogs," have been observed compromising thousands of SOHO devices at scale.[3] One campaign successfully compromised over 1,000 SOHO devices to facilitate a prolonged cyber espionage campaign, with activity dating back to late 2023.[5] In another startling incident, researchers uncovered a sophisticated attack that compromised over 600,000 SOHO routers within a 72-hour window using the Chalubo remote access trojan.[7]

The strategic value of these compromised networks is significant. Threat actors leverage them as a distributed and anonymized infrastructure, creating botnets to serve as launching pads for larger, more targeted attacks against resilient, high-value targets, including U.S. critical infrastructure entities.[3] The connection is direct and alarming: the poor security of a home router can become a component in a nation-state attack on essential services. This elevates the SOHO security problem from a matter of individual consumer protection to one of national

security.

1.3. The Lateral Movement Threat Vector: A Critical Analysis

The primary threat within these compromised SOHO environments is **lateral movement**. This term refers to the set of techniques an attacker uses to pivot and move through a network after gaining an initial foothold on a single device.[8] Studies indicate that adversaries may spend as much as 80% of their attack time engaged in lateral movement, exploring the network to escalate privileges and locate high-value assets.[10] The typical "flat network" topology of a SOHO environment—where a work laptop, a personal phone, a smart TV, and an IP camera all share a single, untrusted Wi-Fi network—is the ideal breeding ground for such attacks.[3]

A common attack scenario begins with the compromise of the least secure device on the network, such as an IoT gadget with a default password or an unpatched vulnerability. This device becomes the attacker's beachhead. From this compromised node, the attacker can scan the internal network, discover other devices, and attempt to exploit them. Because all devices are on the same flat network segment, there are no internal controls to stop this activity. The attacker can move laterally from the low-value IoT device to a high-value asset, such as a work laptop containing sensitive corporate data, financial information, or intellectual property.[11] Techniques used during this phase can include credential dumping, pass-the-hash attacks, and exploiting unpatched service vulnerabilities on other machines.[11] Preventing this internal, east-west traffic is therefore a critical and high-impact security objective that is almost entirely unaddressed by perimeter-focused security tools.

1.4. The Security-by-Design Failure in Consumer-Grade Hardware

The vulnerability of the SOHO ecosystem is not an accident but a systemic market failure. Manufacturers of consumer-grade routers and IoT devices have consistently prioritized low cost and ease of use over robust security, resulting in a widespread "security-by-design" failure. This issue is so severe that it has prompted alerts from government agencies. The U.S. Cybersecurity and Infrastructure Security Agency (CISA) and the Federal Bureau of Investigation (FBI) have explicitly called out SOHO router manufacturers for creating products that lack basic security controls, stating that this practice is "unacceptable given the current threat environment".[6]

Common failures identified in these devices include the absence of automatic firmware update capabilities, the inclusion of numerous exploitable defects in web management interfaces, and the dangerous practice of exposing these management interfaces to the public internet by default.[3] This systemic negligence shifts the burden of security entirely onto the end-user, who typically lacks the technical expertise to harden their devices or implement complex network security measures like segmentation.[4] This creates a clear and urgent need for a user-deployable solution like Project SafeNet, which can impose strong security controls externally without relying on manufacturer action or deep technical knowledge from the user.

2. Literature Review & Foundational Research

The conceptual framework for Project SafeNet is built upon a significant body of research in modern cybersecurity, primarily centered on Zero Trust Architecture (ZTA) and network micro-segmentation. A systematic literature review of ZTA research from 2016 to 2025 highlights a clear paradigm shift away from traditional perimeter-based security models toward a more dynamic and stringent "never trust, always verify" approach.[56] This shift is a direct response to the increasing complexity of digital ecosystems, where the network perimeter has dissolved due to remote work and the proliferation of IoT devices.[56]

The foundational principles of ZTA, as outlined in academic surveys and NIST publications, include the separation of trust from network location, the enforcement of least-privilege access, and the continuous monitoring and evaluation of all network entities.[57] Research emphasizes that ZTA is not a single product but a strategic framework that leverages technologies like multi-factor authentication (MFA), encryption, and, critically, micro-segmentation to reduce the attack surface and mitigate threats like lateral movement.[58]

Micro-segmentation, in particular, is identified as a core enabling technology for ZTA.[59] It offers a method for creating granular, isolated security zones around individual workloads, which is essential for containing breaches and preventing attackers from moving freely within a compromised network.[60] Academic work in this area explores how micro-segmentation enhances security posture, improves compliance, and supports Zero Trust models by safeguarding east-west (internal) traffic.[59]

A key research paper that serves as a basis for this project is **"Zero Trust+: A Trusted-based Zero Trust architecture for IoT at Scale"** by Huber and Kandah.[61] This work is particularly relevant as it directly addresses the application of Zero Trust principles to the IoT ecosystem, which is a major component of the SOHO security problem. The paper acknowledges the performance concerns of conventional ZTA in resource-constrained IoT environments and proposes a trust management-based solution to deliver robust access control more

efficiently.[61] This aligns with Project SafeNet's goal of creating a

lightweight framework suitable for non-enterprise hardware.

Furthermore, a comprehensive survey by Z. Li et al. provides a meticulous explanation of ZTA principles and implementation strategies, exploring techniques like micro-segmentation and security automation that are central to SafeNet's design.[62] This body of work validates the core premise of Project SafeNet: that applying a simplified, lightweight implementation of ZTA and micro-segmentation can effectively address the unique security challenges of the SOHO environment.

3. The Zero-Trust Mandate: A Paradigm Shift in Network Defense

In response to the dissolution of the traditional network perimeter, the cybersecurity industry has shifted towards a new strategic approach: the Zero-Trust Architecture (ZTA). This model inverts the old paradigm of "trust but verify," operating instead on the foundational principle of "never trust, always verify".[1]

3.1. Core Tenets of Zero-Trust Architecture (ZTA)

As formally defined in NIST Special Publication 800-207, Zero Trust is not a single technology but an evolving set of cybersecurity paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources.[13] It assumes there is no implicit trust granted to assets or user accounts based solely on their physical or network location. The architecture is built upon three core principles:

1. **Verify Explicitly**: Every attempt to access a resource must be treated as potentially malicious. Access is granted only after a dynamic authentication and authorization process that considers all available data points. This includes not only user identity but also the location of the access request, the health and posture of the user's device, the service or workload being accessed, and the classification of the data itself.[1] Mechanisms like multi-factor authentication are fundamental to this principle.
2. **Enforce Least-Privilege Access**: This principle dictates that users, devices, and applications should be granted only the minimum level of access necessary to perform their specific, authorized functions. This is enforced through granular, "just-in-time" and

"just-enough" access policies.[1] By strictly limiting access rights, the potential "blast radius" of a compromised account or device is significantly minimized, containing the damage an attacker can inflict.[14]

3. **Assume Breach**: The ZTA model operates with the fundamental assumption that a security breach is not a matter of "if" but "when," and that an attacker may already be present within the network.[1] Consequently, the architectural goal shifts from pure prevention to include rapid detection, containment, and response. Continuous monitoring of all network traffic and user behavior for anomalies is a core component of this principle, allowing security systems to identify and isolate threats before they can escalate.[14]

3.2. Micro-segmentation as the Primary Enforcement Mechanism

While ZTA provides the strategic framework, **micro-segmentation** is the key tactical implementation for enforcing its principles, particularly for neutralizing lateral movement threats.[14] Micro-segmentation is a security method that involves dividing a network into small, granular, and logically isolated segments—potentially down to the individual workload or device level—and applying specific security policies to govern the flow of traffic between them.[17]

By creating these secure micro-perimeters around each device or group of devices, an organization can ensure that even if one segment is breached, the attacker's ability to move laterally to other parts of the network is severely restricted.[19] This stands in stark contrast to traditional, coarse-grained network segmentation, which might use VLANs to separate entire departments. Micro-segmentation is far more granular and is typically identity-aware, meaning policies are based on the identity of the workload or device rather than its IP address, making it highly effective in dynamic environments like the cloud or a SOHO network where IP addresses can change frequently.[17]

3.3. The Enterprise Implementation Gap and the SOHO Opportunity

The principles of ZTA and micro-segmentation are well-established in the enterprise sector, with mature solutions available from vendors such as Akamai (Guardicore), Illumio, Cisco, and Zero Networks.[18] These enterprise solutions, however, are fundamentally incompatible with the realities of the SOHO environment. They are architected for large, complex organizations

and typically rely on:

- **Centralized Infrastructure**: They require integration with enterprise-grade Identity and Access Management (IAM) systems (e.g., Active Directory, Okta) and complex orchestration platforms like Kubernetes.[20]
- **High Cost**: Licensing and implementation costs are prohibitive for individual users or small businesses.
- **Operational Complexity**: They demand dedicated teams of security and network professionals for deployment, policy creation, and ongoing management. Open-source enterprise tools like Istio, Cilium, and Calico, while powerful, are even more complex, requiring deep expertise in service meshes and container networking.[20]

This creates a significant implementation gap. The SOHO market, which is arguably most in need of the protections offered by Zero Trust, has no accessible way to implement it. This gap represents the core opportunity for Project SafeNet: to engineer a solution that captures the security benefits of ZTA and micro-segmentation in a package that is lightweight, autonomous, affordable, and manageable by a non-technical user.

A key innovation required to bridge this gap is a pragmatic redefinition of "identity" for the SOHO context. Enterprise ZTA systems are complex because their policy engines must integrate with sophisticated IAM platforms to verify the identities of human users and software services.[1] SOHO networks lack any such centralized identity infrastructure. The "users" are a small, static group, and the "devices" are a heterogeneous and unmanaged collection of personal and work assets. Instead of attempting to replicate a complex identity system, a more elegant solution is to leverage an identity mechanism inherent to the networking protocol itself. The WireGuard protocol is built on a concept called "Cryptokey Routing," where a peer on the network is identified solely and uniquely by its cryptographic public key.[22] This public key is directly mapped in the configuration to the set of IP addresses it is allowed to use as a source within the secure tunnel. This provides a powerful, built-in form of cryptographic identity. The public key

is the device's identity, and authentication is implicitly handled by the protocol's robust cryptography. By anchoring its policy engine to this inherent device identity, Project SafeNet can bypass the entire complex and costly enterprise IAM stack, drastically simplifying the architecture and making the Zero-Trust model feasible for its target environment.

4. Architectural Framework of the SafeNet Solution

The proposed SafeNet solution is a three-tiered architecture designed to be lightweight, robust, and easily deployable. It consists of a secure data plane, a human-readable policy

engine, and an intelligent control plane that orchestrates the system.

4.1. The Data Plane: A Cryptographically Secure WireGuard Mesh

The foundation of SafeNet is its data plane, which will be constructed using the **WireGuard** protocol to create a secure, encrypted, point-to-point mesh network between all enrolled devices.[23] WireGuard is the ideal choice for this role due to a unique combination of simplicity, performance, and modern cryptographic design.[22]

Key attributes that make WireGuard suitable for SafeNet include:

- **Simplicity and Minimal Attack Surface**: WireGuard is designed with simplicity as a primary goal. Its Linux kernel implementation consists of only around 4,000 lines of code, a stark contrast to the hundreds of thousands of lines in alternatives like OpenVPN or IPsec.[22] This minimal codebase makes it significantly easier to audit for security vulnerabilities and reduces the overall attack surface.
- **State-of-the-Art Cryptography**: The protocol is cryptographically opinionated, meaning it does not allow for cipher negotiation, which has historically been a source of vulnerabilities in protocols like TLS. Instead, it uses a fixed suite of modern, high-speed cryptographic primitives, including Curve2519 for key exchange, ChaCha20 for symmetric encryption, and Poly1305 for message authentication.[22] These choices provide strong security guarantees with excellent performance.
- **High Performance**: The combination of efficient cryptographic primitives and its implementation as a kernel module (on Linux) allows WireGuard to achieve significantly higher throughput and lower latency than user-space VPN solutions like OpenVPN.[22] This is critical for ensuring that the security overlay does not negatively impact the user experience in a SOHO environment.
- **Cryptokey Routing**: This is the central concept that the SafeNet control plane will leverage. In WireGuard, there is no dynamic address assignment. Instead, each peer's configuration file explicitly maps its public key to a list of AllowedIPs—the IP addresses from which it is allowed to send traffic and for which traffic should be routed to it.[22] By programmatically generating and managing these
  AllowedIPs lists for every peer in the mesh, the control plane can enforce the high-level access rules defined in the policy file.

4.2. The Policy Engine: Declarative Security as Code with YAML

The second tier of the architecture is the policy definition layer. Project SafeNet introduces the concept of "Declarative Security as Code" to the consumer space. Instead of requiring users to manually configure firewall rules or complex network settings, SafeNet abstracts this complexity into a single, human-readable YAML file named policy.yml. This approach is inspired by the declarative configuration models used in modern infrastructure tools like Netplan and Kubernetes, where the user specifies the desired *end state* of the system, and an intelligent controller handles the implementation details.[26]

The YAML schema is designed for maximum simplicity and intuitiveness, requiring no specialized networking knowledge. It is structured around two primary concepts: devices, which are the individual endpoints on the network, and access_rules, which define the permitted communication flows between logical groups of those devices. The implicit rule is "default deny"; any traffic flow not explicitly allowed by a rule is blocked. This aligns directly with the Zero-Trust principle of least-privilege access.

An example policy for a typical SOHO environment might look as follows:

YAML

```yaml
# policy.yml
# Defines all devices to be managed by SafeNet.
devices:
  - name: work-laptop
    groups: [work, trusted]
  - name: personal-phone
    groups: [personal, trusted]
  - name: smart-tv
    groups: [iot, untrusted]
  - name: security-camera
    groups: [iot, untrusted]

# Defines the rules for allowed communication.
# All other traffic is blocked by default.
access_rules:
  # Allow devices in the 'work' group to talk to any device in a 'trusted' group.
  - from: work
    to: trusted
    action: allow

  # Allow devices in the 'personal' group to talk to any device in a 'trusted' group.
  - from: personal
```

```
  to: trusted
  action: allow

# Allow devices in the 'iot' group to only access the public internet.
# They cannot initiate connections to any other internal device.
- from: iot
  to: internet # A special target representing egress-only traffic.
  action: allow
```

This declarative model is powerful because it separates the user's intent from the underlying implementation. The user simply declares that the security-camera is an iot device and that iot devices can only talk to the internet. The control plane is then responsible for translating this high-level rule into the specific AllowedIPs configuration needed for the camera's WireGuard interface.

Table 1: Proposed SafeNet YAML Policy Schema

This table provides the formal specification for the policy.yml file, defining the contract between the user and the SafeNet system.

Key	Type	Description	Required	Example
devices	List of Objects	A list of all devices to be enrolled in the SafeNet.	Yes	devices:
devices.name	String	A unique, human-readable name for the device. Used for identification.	Yes	name: work-laptop
devices.groups	List of Strings	A list of logical groups the device belongs to. Groups are used as selectors in access rules.	Yes	groups: [work, trusted]

access_rules	List of Objects	A list of rules defining allowed traffic flows. The order does not matter.	Yes	access_rules:
access_rules.from	String	The source group name for the traffic. Must match a group defined in devices.	Yes	from: iot
access_rules.to	String	The destination group name. Can be a group name or the special target internet.	Yes	to: trusted
access_rules.action	String	The action to take. The only currently supported value is allow.	Yes	action: allow

4.3. The Control Plane: The Python Orchestration Engine

The third and most critical tier is the control plane, a Python-based application that serves as the "brain" of the SafeNet framework. It is responsible for translating the user's declarative policy into a functioning, secure network. Its core responsibilities are as follows:

1. **Policy Parsing and Validation**: The control plane continuously monitors the policy.yml file for changes. Upon detection of a change, it uses the PyYAML library to parse the file's contents into a structured Python dictionary.[28] It then validates the schema to ensure all required fields are present and correctly formatted, rejecting invalid

configurations with clear error messages.

2. **State Management and Configuration Generation**: The controller maintains an in-memory representation of the desired network state based on the parsed policy. It generates a unique cryptographic key pair (public and private key) for each device defined in the policy. It also assigns a static private IP address to each device from a predefined subnet (e.g., 10.8.0.0/24). The core logic resides in its ability to translate the access_rules into specific WireGuard configurations. For each device, it calculates the AllowedIPs list for every one of its peers. For instance, based on the example policy, the work-laptop would have a peer entry for the personal-phone that allows its IP, but the security-camera would have a peer entry for the work-laptop with an empty AllowedIPs list, effectively blocking it from initiating any communication.
3. **Configuration Application and Network Orchestration**: For the Minimum Viable Product (MVP) targeting Windows 11, the control plane will use Python's built-in subprocess module to interact with the official WireGuard for Windows command-line utilities (wireguard.exe and wg.exe).[63] This allows it to perform all necessary network operations programmatically. The required steps for each policy application include:
    - Generating a complete WireGuard configuration file in memory for the local device.
    - Writing this configuration to a temporary file.
    - Invoking wireguard.exe /installtunnelservice <config_file_path> to create or update the Windows service that manages the WireGuard tunnel.[63] This command handles the creation of the virtual network adapter and applies the full configuration.
    - Using sc.exe commands via subprocess to ensure the service is started.
    - Cleaning up the temporary configuration file.

This three-tiered architecture effectively separates concerns: the data plane (WireGuard) provides robust, high-performance, and secure transport; the policy engine (YAML) provides a simple and accessible user interface; and the control plane (Python) contains the intelligence to bridge the two, automating the complex task of network micro-segmentation.

5. System Implementation and Technical Deep Dive

This section provides a detailed engineering blueprint for the development of the SafeNet application, addressing specific architectural decisions and justifying them based on technical merit and project goals.

5.1. Core Technology Stack & Selection

The selection of the technology stack is critical for achieving the project's goals of being lightweight, cross-platform (in the future), and robust.

- **Python Backend**: Python is selected as the language for the control plane due to its readability, extensive standard library, and a rich ecosystem of third-party packages for network automation.[30]
    - **Network Interface Management**: For the Windows MVP, network interface management will be handled by programmatically invoking the official wireguard.exe command-line utility via Python's subprocess module.[63] This is the standard method for enterprise and programmatic control of WireGuard tunnels on Windows.[63] While libraries like
      psutil [71] or
      ifaddr [72] can be used for cross-platform network interface *enumeration*, direct configuration will be performed by shelling out to the WireGuard executable. This approach ensures compatibility and leverages the official, stable interface provided by the WireGuard project for Windows.
    - **Asynchronous Operations**: While the MVP will be implemented with a synchronous control loop, future versions that might involve real-time communication with a cloud controller or managing a large number of dynamic peers would benefit from an asynchronous architecture. Python's built-in **asyncio** library would be the natural choice for this evolution.
- **Electron Frontend**: Electron is the chosen framework for building the cross-platform graphical user interface (GUI) that is a core part of the MVP. It allows for the use of web technologies (HTML, CSS, JavaScript) to create native-feeling desktop applications.[32]
    - **JavaScript Framework**: Among the popular choices of React, Vue, and Svelte, **React** is recommended for the SafeNet UI.[34] React's component-based architecture, massive ecosystem of pre-built UI components, and mature state management libraries (e.g., Redux, Zustand) make it exceptionally well-suited for building a complex and reactive interface for managing network devices and policies. Its large community ensures extensive documentation and long-term support.[34]
    - **Essential Node.js Modules**: The Electron application will rely on several key Node.js modules. The child_process module will be essential for spawning and managing the lifecycle of the Python backend engine. For persisting user settings (such as window dimensions or theme preferences), electron-store provides a simple and robust key-value storage solution. Node's native fs and path modules will be used for all file system interactions to ensure cross-platform compatibility.
- **Database and Storage**: The storage requirements for the MVP are minimal. The single source of truth for the network configuration is the user-edited policy.yml file. For application-specific settings that are not part of the network policy, a simple JSON file is sufficient. This file will be stored in a location that adheres to operating system conventions. On Windows, this will be located in the user's application data directory at

%APPDATA%\SafeNet\.[73]

5.2. System Architecture and Inter-Process Communication (IPC)

The SafeNet application will be composed of three primary components: the Electron Renderer process (the UI), the Electron Main process (the application's backend logic), and the Python Control Plane (the core networking engine). A robust communication channel between these processes is paramount.

- **Process Management**: The Electron Main process will act as the parent and supervisor for the Python engine. It will use the child_process.spawn() method to launch the packaged Python executable as a long-running, hidden background process. The Main process will attach listeners to the Python process's stdout and stderr streams to receive status updates and error logs, which can then be relayed to the UI. It will also monitor the process's exit event to detect crashes and implement a fault-tolerant restart mechanism.
- **Inter-Process Communication (IPC) Design**: The choice of IPC mechanism is a critical architectural decision that impacts performance, security, and developer productivity. After evaluating several alternatives, **gRPC** is selected as the optimal protocol for communication between the Electron/Node.js frontend and the Python backend.
  The rationale for this choice is multifaceted. Compared to a traditional REST API over localhost, gRPC offers substantially higher performance. It utilizes HTTP/2, which allows for multiplexing multiple requests over a single connection, and it serializes data using Protocol Buffers (Protobufs), a binary format that is far more compact and faster to parse than text-based JSON.[37] This efficiency is crucial for a responsive UI that may need to exchange frequent updates with the backend. Furthermore, gRPC's support for bidirectional streaming is a key advantage over the simple request-response model of REST.[39] This allows the Python engine to proactively push real-time status updates (e.g., "peer connected," "policy applied") to the UI without the need for inefficient polling.
  Perhaps most importantly, the use of Protobufs provides a strongly-typed, language-agnostic API contract. The services and message formats are defined in a .proto file, from which client and server stub code can be automatically generated for both JavaScript/TypeScript (for Electron) and Python.[40] This eliminates a whole class of potential integration errors and makes the communication protocol robust and self-documenting. To secure this local communication channel from being accessed by other applications on the user's machine, a simple authentication mechanism will be implemented. At startup, the Electron Main process will generate a cryptographically secure random token. This token will be passed as a command-line argument to the spawned Python process. The Python gRPC server will be configured to require this token in the metadata of every incoming RPC call, immediately rejecting any request that

does not present the correct token.

Table 2: Comparison of Inter-Process Communication (IPC) Mechanisms

Criterion	REST API (over localhost)	WebSockets	gRPC	Rationale for SafeNet
Performance	Lower (HTTP/1.1 overhead, JSON parsing)	High (persistent TCP connection)	Very High (HTTP/2, Protobuf binary serialization)	**gRPC:** Lowest latency and CPU overhead is critical for a background security service.[37]
Communication Model	Unary (Request-Response)	Bidirectional Streaming	Unary, Server-Streaming, Client-Streaming, Bidirectional-Streaming	**gRPC:** Bidirectional streaming allows the Python engine to push real-time status updates to the UI efficiently.[39]
API Contract	Loosely-typed (OpenAPI for docs)	None (message-based)	Strongly-typed (Protobuf definitions)	**gRPC:** Protobufs provide a strict, language-agnostic contract, minimizing integration bugs between Node.js and Python.[40]
Code Generation	Manual or third-party tools	Manual	Built-in (client/server stubs)	**gRPC:** Auto-generated stubs reduce

				boilerplate code and ensure consistency.[39]
Security (Local)	Requires custom token auth; easily discoverable	Requires custom token auth	Can be secured with TLS or token metadata	**gRPC:** While all require auth, the binary protocol is less prone to casual snooping by other local processes.
Complexity	Low	Medium	Medium (requires Protobuf setup)	**gRPC:** The initial setup overhead is justified by the long-term benefits in robustness and performance.

5.3. Cross-Platform Packaging and Deployment Strategy

A significant challenge for a hybrid application like SafeNet is the build and packaging process. The goal is to deliver a single, easy-to-install package for Windows that contains both the Electron frontend and the Python backend, without requiring the end-user to have Python or any dependencies installed.

- **Bundling the Python Core**: The Python control plane, along with all its dependencies (e.g., pyyaml, grpcio), will be converted into a single, standalone executable. This will be accomplished using **PyInstaller**, a widely-used tool for freezing Python applications.[41] The
  pyinstaller --onefile command will be used to create a single executable (e.g., engine.exe) that contains the Python interpreter and all necessary libraries, making it fully portable.[43]

- **Bundling the Electron App**: The Electron application itself will be packaged using **electron-builder**. This tool provides a comprehensive solution for creating professional installers and distributables for Windows, macOS, and Linux.[45] The key to integrating the Python backend is to treat the PyInstaller-generated executable as an external resource. This is achieved by placing the engine.exe in a designated folder and configuring the extraResources or files array in the electron-builder section of the package.json file.[74] This configuration instructs electron-builder to copy the Python executable into the application's resources directory during the packaging process.
- **Build Configuration**: The electron-builder configuration will be set up to target Windows, generating an nsis installer (.exe).[74] The build process will be automated with scripts that first run PyInstaller to create the Windows Python executable, and then run electron-builder, ensuring the engine.exe is included in the final installer.

5.4. Security Model and Privileged Operations

Ensuring the security of the SafeNet application itself is as important as the network security it provides. The security model must address both the application's internal vulnerabilities and its need to perform privileged operations on the host system.

- **Application Security**: The design will adhere to the security best practices outlined in the official Electron documentation and the OWASP Top 10 for Desktop Applications.[47] Key security settings will be enforced in the BrowserWindow configuration: contextIsolation will be enabled, and nodeIntegration will be disabled. This creates a strong boundary between the web content in the renderer process and the powerful Node.js/Electron APIs in the main process, mitigating the risk of Cross-Site Scripting (XSS) attacks leading to remote code execution.[49] All IPC messages will be validated to ensure they originate from trusted sources.
- **Privileged Access**: A core function of SafeNet is configuring network interfaces, which is a privileged operation on Windows requiring administrator access. Running the entire Electron application with elevated privileges is a major security risk and must be avoided. A more secure approach is to request elevation only when necessary.
    - **Installation**: The electron-builder configuration for the NSIS installer will set the requestedExecutionLevel to requireAdministrator.[75] This will cause the installer to trigger a User Account Control (UAC) prompt upon execution, ensuring the application and its components (including the WireGuard service) are installed with the necessary permissions to manage network interfaces.
    - **Runtime**: The main application will run with standard user privileges. If a specific, privileged operation is required at runtime that was not covered by the installer's

permissions, the Python backend can spawn a helper script using mechanisms that trigger a UAC prompt, such as using the runas verb with ShellExecute.[76] This ensures that elevated permissions are granted only for specific, user-approved actions.

6. Validation Protocol and Efficacy Analysis

To validate the feasibility, performance, and security efficacy of the SafeNet framework, a rigorous, multi-faceted validation protocol is proposed. This protocol is designed to answer the key research questions posed by the project through objective, quantifiable, and repeatable experiments.

6.1. Performance Benchmarking Protocol

The primary performance concern is the overhead introduced by the WireGuard overlay network. While WireGuard is known for its high performance, it is essential to quantify this impact on typical SOHO hardware and consumer-grade internet connections.

- **Objective**: To measure the quantifiable overhead of the SafeNet framework in terms of network throughput, latency, and system resource utilization.
- **Methodology**: A controlled testbed will be established using three to four virtual machines running Windows 11 on a single physical host. This setup minimizes external network variables, allowing for a precise measurement of the overhead introduced by the SafeNet software itself.
- **Metrics and Tools**:
  - **Throughput**: Network throughput will be measured between two peer VMs using the industry-standard **iperf3** tool.[51] A baseline measurement will be taken over the direct virtual network. Then, SafeNet will be activated, and the iperf3 test will be repeated through the encrypted WireGuard tunnel. The percentage decrease in throughput will be calculated to determine the overhead.
  - **Latency**: Round-trip time (RTT) will be measured using the standard ping utility for a baseline. For more detailed analysis of jitter and one-way latency, the **uperf** network performance tool will be employed.[52] Again, measurements will be taken with and without the SafeNet overlay to isolate its impact.
  - **CPU Usage**: The CPU utilization of the WireGuard tunnel service and the Python control plane process will be monitored on each VM using standard Windows utilities like Task Manager or Performance Monitor. Measurements will be recorded during

idle periods and under heavy network load (during the iperf3 throughput test) to assess the resource consumption of the framework.
    - **Policy Application Latency**: A custom script will be developed to measure the time from the moment the policy.yml file is saved to the moment the network configuration is fully applied and a new peer connection can be successfully established. This metric is crucial for evaluating the responsiveness of the control plane.

6.2. Security Efficacy Simulation

The core claim of Project SafeNet is its ability to prevent lateral movement attacks. This claim must be validated through practical, simulated attacks in a controlled environment.

- **Objective**: To objectively demonstrate and document SafeNet's ability to prevent common lateral movement attack vectors in a simulated, compromised SOHO network.
- **Methodology**: A virtual environment will be created to mirror a typical SOHO network. It will consist of at least three Windows 11 VMs:
    1. **"Work Laptop"**: A VM assigned to the [work, trusted] groups.
    2. **"Personal Phone"**: A VM assigned to the [personal, trusted] groups.
    3. "Insecure IoT Camera": A VM assigned to the [iot, untrusted] group. This VM will serve as the "compromised" host and the origin point for all simulated attacks. The SafeNet policy will be configured as per the example in Section 4.2, which strictly isolates the iot group, allowing it only egress traffic to the internet.
- **Attack Scenarios**: A series of common lateral movement techniques will be executed from the "Insecure IoT Camera" VM, targeting the "Work Laptop" VM. The success or failure of each attack will be recorded.

Table 3: Lateral Movement Attack Simulation Scenarios and Expected Outcomes

This table outlines the specific tests that will be conducted to validate the security efficacy of the micro-segmentation enforced by SafeNet.

Attack Vector	Simulation Tool	Action from Compromised Device	SafeNet Policy	Expected Outcome	Verification Method
Network Scanning	nmap	Execute a host discovery	The iot group is not allowed to	The nmap scan fails to discover	Review of nmap command

		scan (nmap -sn 10.8.0.0/24) to identify other devices on the SafeNet network.	initiate connections to the trusted group.	the "Work Laptop" or "Personal Phone." The scan result shows only the attacker's own IP and the gateway as being online.	output. Packet capture tools like **Wireshark** [53] on the target VM show no incoming ARP or ICMP echo requests from the attacker's IP.
ARP Spoofing	arpspoof	Attempt to poison the ARP cache of the "Work Laptop" to intercept its traffic (Man-in-the-Middle).	N/A (Layer 2 attack). WireGuard operates at Layer 3.	The ARP spoofing attack itself may succeed at Layer 2. However, all subsequent IP traffic from the "Work Laptop" is encapsulated within the WireGuard tunnel, making it immune to interception by the compromised device.	arp -a on the target may show the attacker's MAC address for the gateway, but attempts to access external sites will still succeed securely through the tunnel, and the attacker will be unable to decrypt the traffic.
SMB	Metasploit	Attempt to	The iot	The	Analysis of

Exploitation	Framework	connect to the "Work Laptop" on TCP port 445 to exploit a known SMB vulnerability (e.g., a simulated EternalBlue).	group cannot initiate any connections to the trusted group on any port.	connection attempt to port 445 is blocked. The Metasploit exploit module fails at the "target connection" stage.	firewall logs or packet captures on the "Work Laptop" shows that the incoming TCP SYN packet from the attacker's IP to port 445 is dropped and never receives a response.
Remote Access Attempt	nmap, rdesktop	Scan the "Work Laptop" for open RDP (3389) or SSH (22) ports and attempt to connect.	The iot group cannot initiate any connections to the trusted group.	The port scan shows all ports as filtered or closed. Direct connection attempts with rdesktop or ssh time out or are immediately refused.	nmap output confirms no open ports are visible from the attacker's perspective. The connection attempt fails.

6.3. Usability Testing Protocol

As the MVP includes a graphical user interface, its usability is a key measure of success.

- **Objective**: To evaluate the intuitiveness and clarity of the SafeNet GUI for non-technical

users.

- **Methodology**: A small group of 4-6 participants, representative of the target SOHO user base (i.e., not network security professionals), will be recruited.[77] Each participant will be given the installed SafeNet application and a simple set of goals, without specific step-by-step instructions.
- **Test Scenarios**:
  1. **Initial Setup**: "Configure the application to manage your work laptop and your smart TV."
  2. **Policy Creation**: "Create a rule that allows your laptop to access everything, but prevents the smart TV from accessing anything except the internet."
  3. **Policy Modification**: "Add your personal phone to the network and give it the same permissions as your work laptop."
- **Data Collection**: The sessions will be observed (with permission) to identify points of confusion, hesitation, or error. Participants will be encouraged to "think aloud" to provide qualitative feedback. Open-ended questions will be used to probe their understanding and experience, such as "What would you do to find out if this rule is working?".[77] The primary metric for success will be the user's ability to complete the assigned tasks without direct assistance, and their subjective feedback on the clarity of the interface and the policy-creation process.

Successful completion of these validation tests will provide strong, evidence-based support for the claims that SafeNet is both performant enough for SOHO use and effective at mitigating the critical threat of lateral movement.

7. Innovation, Contribution, and Future Trajectory

Project SafeNet represents a significant contribution to consumer cybersecurity by adapting and simplifying enterprise-grade security principles for a fundamentally different and underserved market. Its innovation lies not in the invention of new protocols, but in the novel application and orchestration of existing, robust technologies to solve a pressing and widespread problem.

7.1. Summary of Contributions

The novelty and primary contributions of this research project can be summarized in three key areas:

1. **Democratization of Zero Trust**: The project's most significant contribution is the successful translation of the high-end, enterprise-grade Zero-Trust security model to the SOHO and consumer space. It demonstrates that the core principles of explicit verification, least-privilege access, and assumed breach can be effectively implemented in an environment lacking centralized identity management, dedicated security personnel, and enterprise-grade hardware. It achieves this by pragmatically redefining device identity around strong cryptography, a model that is both secure and architecturally simple.
2. **Declarative Security as Code for Consumers**: SafeNet introduces a novel paradigm for managing personal network security. By leveraging a simple, declarative YAML file, it applies the "Infrastructure as Code" model, popular in DevOps and cloud environments, to the home network. This empowers non-technical users to define complex network segmentation policies through a simple, human-readable text file, abstracting away the immense complexity of manual firewall rules, IP routing, and cryptographic key management.
3. **Targeted Mitigation of Lateral Movement**: While many consumer security products focus on perimeter defense (firewalls) or endpoint protection (antivirus), SafeNet focuses specifically on neutralizing the threat of lateral movement within the local network. This is a critical and often-overlooked attack vector in SOHO environments, where the breach of a single insecure IoT device can lead to the compromise of an entire network. By providing an accessible tool for micro-segmentation, SafeNet fills a crucial gap in the consumer security landscape.

7.2. Scope Delimitations and Future Work

This initial research project is scoped to deliver a Minimum Viable Product (MVP) that validates the core architectural concepts. The explicit scope of this phase is:

- **In-Scope (MVP)**: A graphical user interface (GUI) application for Windows 11 that can manage a static set of enrolled devices based on a local policy.yml file. The application will be delivered as a standard Windows installer.

The following features, while valuable, are explicitly designated as out-of-scope for this initial research phase but form a clear roadmap for future development:

- **Official Support for Linux and macOS**: Extending SafeNet to support other operating systems is a critical step for broad adoption. This will require significant research and development to create platform-specific modules within the Python control plane that can interface with the native networking APIs of each operating system to manage WireGuard interfaces.
- **Dynamic Device Onboarding**: To improve usability, a system for dynamic device

enrollment is required. This would likely involve the primary SafeNet controller displaying a QR code or a one-time token that new devices can use to securely join the network, automating the process of key exchange and policy integration.
- **Integration with External Identity Providers**: For more advanced SOHO or small business users, future versions could offer optional integration with cloud-based identity providers (e.g., Google, Microsoft). This would allow policies to be based on user identity in addition to device identity, enabling even more granular access control.

By successfully delivering the MVP and validating its performance and security efficacy, this project will lay a strong foundation for the future development of a comprehensive, cross-platform security solution that can meaningfully improve the security posture of millions of vulnerable SOHO networks.

Glossary of Technical Terms

- **Zero-Trust Architecture (ZTA):** A modern cybersecurity model that operates on the principle of "never trust, always verify." It eliminates the idea of a trusted internal network and requires strict verification for every user and device attempting to access resources, regardless of their location.[1]
- **Lateral Movement:** The set of techniques and strategies that cyber attackers use to move through a network in search of key data and assets after gaining initial access to a single machine.[8]
- **Micro-segmentation:** A network security technique that divides a network into small, isolated segments, down to the individual workload or device level. This practice is used to limit an attacker's ability to move laterally and contain the impact of a breach.[14]
- **WireGuard:** An extremely simple yet fast and modern Virtual Private Network (VPN) that utilizes state-of-the-art cryptography. It is designed to be more performant and easier to configure than older VPN protocols like IPsec and OpenVPN.[22]
- **Cryptokey Routing:** The core principle of WireGuard's design, where cryptographic public keys are directly associated with a list of IP addresses that are allowed to send traffic within the secure tunnel. This provides a strong link between identity (the key) and authorization (the allowed IPs).[22]
- **Declarative Configuration:** A method of configuring a system by defining the desired final state in a configuration file (e.g., a YAML file), rather than specifying the sequence of commands to reach that state. The system's control plane is responsible for making the necessary changes to match the declared state.[26]
- **User Account Control (UAC):** A security feature in Windows that helps prevent unauthorized changes to the operating system by prompting the user for permission when an application requires elevated (administrator) privileges.[76]

- **PyInstaller:** A program that freezes (packages) Python applications into stand-alone executables that can be run without requiring a Python installation on the target machine.[41]
- **Electron:** A framework for building cross-platform desktop applications using web technologies such as JavaScript, HTML, and CSS. It combines the Chromium rendering engine and the Node.js runtime into a single package.[32]
- **Inter-Process Communication (IPC):** A set of mechanisms that an operating system provides to allow different, concurrently running processes (e.g., the Electron frontend and the Python backend) to manage and synchronize their actions by passing messages.[54]
- **gRPC:** A high-performance, open-source Remote Procedure Call (RPC) framework initially developed by Google. It uses HTTP/2 for transport and Protocol Buffers as the interface definition language, enabling efficient and strongly-typed communication between services.[37]
- **electron-builder:** A complete solution to package and build a ready-for-distribution Electron app for macOS, Windows, and Linux, with "auto-update" support out of the box.[45]
- **Netsh:** A command-line scripting utility in Windows that allows you to, either locally or remotely, display or modify the network configuration of a computer that is currently running.[78]

Works cited

1. What is Zero Trust Architecture? - Palo Alto Networks, accessed on September 14, 2025, https://www.paloaltonetworks.com/cyberpedia/what-is-a-zero-trust-architecture
2. Implementing a Zero Trust Architecture - NIST NCCoE, accessed on September 14, 2025, https://www.nccoe.nist.gov/projects/implementing-zero-trust-architecture
3. Work From Home Increases Risk Of Cyberattacks Via SOHO Routers - Packetlabs, accessed on September 14, 2025, https://www.packetlabs.net/posts/work-from-home-increases-risk-of-cyberattacks-via-soho-routers/
4. Securing a Basic SOHO Network: Best Practices and Strategies | by Aardvark Infinity, accessed on September 14, 2025, https://medium.com/aardvark-infinity/securing-a-basic-soho-network-best-practices-and-strategies-4d727357cfc8
5. Over 1,000 SOHO Devices Hacked in China-linked LapDogs Cyber Espionage Campaign, accessed on September 14, 2025, https://thehackernews.com/2025/06/over-1000-soho-devices-hacked-in-china.html
6. TLP Clear Secure by Design Alert - Security Design Improvements for SOHO Device Manufacturers | AHA - American Hospital Association, accessed on September 14, 2025,

https://www.aha.org/cybersecurity-government-intelligence-reports/2024-01-31-tlp-clear-secure-design-alert-security-design-improvements-soho-device
7. Massive SOHO Router Compromise Raises Concerns - Cybersecurity News, accessed on September 14, 2025, https://cyberinsights.tech/r?n=30&z=Massive%20SOHO%20Router%20Compromise%20Raises%20Concerns
8. What is a Lateral Movement? Prevention and Detection Methods - Fortinet, accessed on September 14, 2025, https://www.fortinet.com/resources/cyberglossary/lateral-movement
9. Cybersecurity 101: What is Lateral Movement? A Complete Breakdown | Illumio, accessed on September 14, 2025, https://www.illumio.com/cybersecurity-101/lateral-movement
10. Why Do Organizations Need to Simulate Lateral Movement Attacks? - Picus Security, accessed on September 14, 2025, https://www.picussecurity.com/resource/blog/why-do-organizations-need-to-simulate-lateral-movement-attacks
11. What Is Lateral Movement? | IBM, accessed on September 14, 2025, https://www.ibm.com/think/topics/lateral-movement
12. The Ultimate Guide to Lateral Movement: Key Innovations and Prevention Techniques, accessed on September 14, 2025, https://zeronetworks.com/resource-center/topics/lateral-movement-innovations-prevention-techniques
13. Zero Trust Architecture - NIST Technical Series Publications, accessed on September 14, 2025, https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-207.pdf
14. NIST Zero Trust: Principles, Components & How to Get Started - Tigera, accessed on September 14, 2025, https://www.tigera.io/learn/guides/zero-trust/nist-zero-trust/
15. Zero Trust Strategy & Architecture | Microsoft Security, accessed on September 14, 2025, https://www.microsoft.com/en-us/security/business/zero-trust
16. NIST SP 1800–35: Implementing a Zero Trust Architecture | by SOCFortress - Medium, accessed on September 14, 2025, https://socfortress.medium.com/nist-sp-1800-35-implementing-a-zero-trust-architecture-fbe9ec56e187
17. What Is Microsegmentation? - Palo Alto Networks, accessed on September 14, 2025, https://www.paloaltonetworks.com/cyberpedia/what-is-microsegmentation
18. Akamai Guardicore Segmentation, accessed on September 14, 2025, https://www.akamai.com/products/akamai-guardicore-segmentation
19. Zero Networks | The Leader in Microsegmentation and Zero Trust Security, accessed on September 14, 2025, https://zeronetworks.com/
20. Top 10 Open Source Micro Segmentation Tools - AIMultiple, accessed on September 14, 2025, https://aimultiple.com/open-source-micro-segmentation-tools
21. 8 Microsegmentation Tools to Know in 2025 - Tigera, accessed on September 14,

2025, https://www.tigera.io/learn/guides/microsegmentation/microsegmentation-tools/
22. WireGuard: fast, modern, secure VPN tunnel, accessed on September 14, 2025, https://www.wireguard.com/
23. Setting up a private mesh VPN with WireGuard® | Scaleway Documentation, accessed on September 14, 2025, https://www.scaleway.com/en/docs/tutorials/wireguard-mesh-vpn/
24. Wirenix: A flake for creating wireguard mesh networks - Announcements - NixOS Discourse, accessed on September 14, 2025, https://discourse.nixos.org/t/wirenix-a-flake-for-creating-wireguard-mesh-networks/32542
25. How to Configure WireGuard Mesh VPN? - zenarmor.com, accessed on September 14, 2025, https://www.zenarmor.com/docs/network-security-tutorials/how-to-configure-wireguard-mesh-vpn
26. Chapter 16. Using declarative configuration - Red Hat Documentation, accessed on September 14, 2025, https://docs.redhat.com/en/documentation/red_hat_advanced_cluster_security_for_kubernetes/4.4/html/configuring/declarative-configuration-using
27. Canonical Netplan, accessed on September 14, 2025, https://netplan.io/
28. Parse a YAML file in Python - GeeksforGeeks, accessed on September 14, 2025, https://www.geeksforgeeks.org/python/parse-a-yaml-file-in-python/
29. How to Load, Read, and Write YAML • Python Land Tutorial, accessed on September 14, 2025, https://python.land/data-processing/python-yaml
30. How to Leverage Netmiko Python for Network Automation - CloudMyLab Blog, accessed on September 14, 2025, https://blog.cloudmylab.com/netmiko-python-for-network-automation
31. Python for Network Engineers: Automating the Path to Efficiency - AMPL, accessed on September 14, 2025, https://ampl.com/blog/python-for-network-engineers-automating-the-path-to-efficiency/
32. Electron vs Node.js: Future of Cross-Platform Development in 2025 - Index.dev, accessed on September 14, 2025, https://www.index.dev/blog/electron-vs-nodejs
33. Compare Electron vs. Vue.js in 2025, accessed on September 14, 2025, https://slashdot.org/software/comparison/Electron-vs-Vue.js/
34. The Most Popular JavaScript Frameworks in 2025 - Mobilunity, accessed on September 14, 2025, https://mobilunity.com/blog/top-20-javascript-frameworks-chosen-by-remote-developers-in-2024/
35. The best UI frameworks: What you should build with this year - BCMS, accessed on September 14, 2025, https://thebcms.com/blog/the-best-ui-frameworks
36. Top 5 Cross-Platform Frameworks in 2025: A Detailed Comparison | by Namrata Bagh, accessed on September 14, 2025, https://medium.com/@namratabagh20/top-5-cross-platform-frameworks-in-2025-a-detailed-comparison-b60bd5d3d1aa

37. gRPC vs. REST: Key Similarities and Differences - DreamFactory Blog, accessed on September 14, 2025, https://blog.dreamfactory.com/grpc-vs-rest-how-does-grpc-compare-with-traditional-rest-apis
38. REST vs gRPC vs GraphQL vs WebSockets vs SOAP: A Practical Guide for Engineers, accessed on September 14, 2025, https://dev.to/rajkundalia/rest-vs-grpc-vs-graphql-vs-websockets-vs-soap-a-practical-guide-for-engineers-37d9
39. gRPC vs REST - Difference Between Application Designs - AWS, accessed on September 14, 2025, https://aws.amazon.com/compare/the-difference-between-grpc-and-rest/
40. Can gRPC replace REST and WebSockets for Web Application Communication?, accessed on September 14, 2025, https://grpc.io/blog/postman-grpcweb/
41. PyInstaller Manual — PyInstaller 6.16.0 documentation, accessed on September 14, 2025, https://www.pyinstaller.org/
42. pyinstaller/pyinstaller: Freeze (package) Python programs into stand-alone executables - GitHub, accessed on September 14, 2025, https://github.com/pyinstaller/pyinstaller
43. The one-stop guide to (easy) cross-platform Python freezing: Part 1 - Medium, accessed on September 14, 2025, https://medium.com/hackernoon/the-one-stop-guide-to-easy-cross-platform-python-freezing-part-1-c53e66556a0a
44. Building Cross-Platform Applications with Tkinter and PyInstaller | by Tom - Medium, accessed on September 14, 2025, https://medium.com/tomtalkspython/building-cross-platform-applications-with-tkinter-and-pyinstaller-d7a10163c550
45. Packaging Your Application | Electron, accessed on September 14, 2025, https://electronjs.org/docs/latest/tutorial/tutorial-packaging
46. How to deploy an Electron app as an executable or installable in Windows? - Stack Overflow, accessed on September 14, 2025, https://stackoverflow.com/questions/31286924/how-to-deploy-an-electron-app-as-an-executable-or-installable-in-windows
47. Security | Electron, accessed on September 14, 2025, https://electronjs.org/docs/latest/tutorial/security
48. OWASP Desktop App Security Top 10, accessed on September 14, 2025, https://owasp.org/www-project-desktop-app-security-top-10/
49. Electron Security Concerns - Quasar Framework, accessed on September 14, 2025, https://quasar.dev/quasar-cli-vite/developing-electron-apps/electron-security-concerns/
50. Electron Security Checklist - Doyensec, accessed on September 14, 2025, https://doyensec.com/resources/us-17-Carettoni-Electronegativity-A-Study-Of-Electron-Security-wp.pdf
51. iPerf - The TCP, UDP and SCTP network bandwidth measurement tool, accessed on September 14, 2025, https://iperf.fr/

52. Uperf - A network performance tool, accessed on September 14, 2025, https://uperf.org/
53. Wireshark • Go Deep, accessed on September 14, 2025, https://www.wireshark.org/
54. Inter-Process Communication - Electron, accessed on September 14, 2025, https://electronjs.org/docs/latest/tutorial/ipc
55. How to do local IPC without leaking handles (cross platform)? - Stack Overflow, accessed on September 14, 2025, https://stackoverflow.com/questions/21421250/how-to-do-local-ipc-without-leaking-handles-cross-platform
56. Zero Trust Architecture: A Systematic Literature Review - arXiv, accessed on September 15, 2025, https://arxiv.org/html/2503.11659v2
57. Theory and Application of Zero Trust Security: A Brief Survey - PMC - PubMed Central, accessed on September 15, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10742574/
58. Zero trust architecture: A paradigm shift in network security - ResearchGate, accessed on September 15, 2025, https://www.researchgate.net/publication/390558157_Zero_trust_architecture_A_paradigm_shift_in_network_security
59. (PDF) Beyond the Firewall: Implementing Zero Trust with Network Microsegmentation, accessed on September 15, 2025, https://www.researchgate.net/publication/389520879_Beyond_the_Firewall_Implementing_Zero_Trust_with_Network_Microsegmentation
60. The Critical Role of Micro-segmentation in Modern Cybersecurity Architectures: A Comprehensive Review - ResearchGate, accessed on September 15, 2025, https://www.researchgate.net/publication/390051768_The_Critical_Role_of_Micro-segmentation_in_Modern_Cybersecurity_Architectures_A_Comprehensive_Review
61. Zero Trust+: A Trusted-based Zero Trust architecture for IoT at Scale ..., accessed on September 15, 2025, https://www.semanticscholar.org/paper/Zero-Trust%2B%3A-A-Trusted-based-Zero-Trust-for-IoT-at-Huber-Kandah/840adac6efef8dfbe64fd8a49f2eb3fc41cad558
62. A Review and Comparative Analysis of Relevant Approaches of Zero Trust Network Model - PMC - PubMed Central, accessed on September 15, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10892953/
63. wireguard-windows - WireGuard client for Windows - ZX2C4 Git ..., accessed on September 15, 2025, https://git.zx2c4.com/wireguard-windows/about/docs/enterprise.md
64. How to start wireguard client on windows, in the background? - Server Fault, accessed on September 15, 2025, https://serverfault.com/questions/1096001/how-to-start-wireguard-client-on-windows-in-the-background
65. WireGuard Cheatsheet » Nettica, accessed on September 15, 2025, https://nettica.com/wireguard-cheatsheet/
66. Is there a way to run wireguard from the command line in Windows 10? - Reddit,

accessed on September 15, 2025, https://www.reddit.com/r/WireGuard/comments/ch9y6g/is_there_a_way_to_run_wireguard_from_the_command/
67. WireGuard Windows setup | INTROSERV, accessed on September 15, 2025, https://introserv.com/tutorials/wireguard-windows-setup/
68. Wireguard Windows Setup — Generic service & computer documentation. documentation, accessed on September 15, 2025, https://r-pufky.github.io/docs/services/wireguard/windows-setup.html
69. Enable/Disable WireGuard from Windows Command Line - Tom Bonner, accessed on September 15, 2025, https://blog.bonner.is/enable-disable-wireguard-from-windows-command-line/
70. Is there a way to run wireguard from the command line in Windows 10? - Reddit, accessed on September 15, 2025, https://www.reddit.com/r/WireGuard/comments/c9bjnk/is_there_a_way_to_run_wireguard_from_the_command/
71. How to get all network interface cards' information in Python - Educative.io, accessed on September 15, 2025, https://www.educative.io/answers/how-to-get-all-network-interface-cards-information-in-python
72. ifaddr - Enumerate network interfaces/adapters and their IP addresses - GitHub, accessed on September 15, 2025, https://github.com/ifaddr/ifaddr
73. A quick introduction to the Linux filesystem for Windows users. - Red Hat, accessed on September 14, 2025, https://www.redhat.com/en/blog/linux-filesystem-windows
74. Electron.js and electron-builder and python | by Abdelkader Yagoubi | Medium, accessed on September 15, 2025, https://medium.com/@yagoubi.aek.2/build-you-app-using-electron-js-python-electron-builder-dcdd9c2d9ba0
75. Check if electron app is launched with admin privileges on windows - Stack Overflow, accessed on September 15, 2025, https://stackoverflow.com/questions/37322862/check-if-electron-app-is-launched-with-admin-privileges-on-windows
76. How to Request Admin Permissions in Windows via UAC in Golang - Endurance, accessed on September 15, 2025, https://blog.hadenes.io/post/how-to-request-admin-permissions-in-windows-through-uac-with-golang/
77. About electron administrator permissions : r/electronjs - Reddit, accessed on September 15, 2025, https://www.reddit.com/r/electronjs/comments/1er25es/about_electron_administrator_permissions/
78. Configuring network settings from command line using netsh ..., accessed on September 15, 2025, https://lizardsystems.com/articles/configuring-network-settings-command-line-using-netsh/