International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

A Comprehensive Study on Secure Shell: Remote Infused with AI/ML

Omkar Swami[1], Dhananjay Tailor[2], Rushbh Mistry[3], Prashant Chauhan[4], and Under the Guidance of Assistant Prof. Rashmi Pal[5]

[1]Department of Computer Science and Engineering
[2]Parul Institute of Engineering and Technology, Parul University September 6, 2025

Abstract

This paper reports on the Nous project which aims to emphasize capabilities of AI in our platform integrating remote connection for enterprise scale system ad- ministration through natural language commands. The system empowers admin- istrators to manage distributed infrastructures through conversational commands. The project Nous sets up remote system administration as a combined environment allowing AI agents to interface with distributed infrastructures. In this paper we discuss the design and implementation of this platform in detail.

1. Introduction

The rise of remote work and multi-cloud strategies has completely changed how system administration is done. In modern IT environments, managing multiple computers with security in different environments and networks is a challenging task as it involves either multiple automation scripts or time taking manual handling of each computer. For system admins to handle everything manually is fine until they face a larger number of machines. In larger organisations, different teams monitor their systems with multiple administra- tors which often is a complex task. This manual approach is not only inefficient but also prone to human error, leading to security vulnerabilities and costly downtime. The optimal solution has to be a proactive one being feasible and cost effective at the same time without compromising the security of the system. Existing remote management platforms carry critical deficiencies as they are inefficient for managing a large number of systems concurrently, convoluted in their architecture involving heavy complex scripting processes, demanding significant expertise and steep learning curves. Some emerging AI- based interfaces rely completely on UI level automation and lack secure, direct remote control in distributed environments.

The main goal of our project is for system admins to be able to handle multiple computers via intelligent automation and improving security measures and curtailing the procedural latency. This research introduces a new way to create an intuitive command interface for distributed computer management. This solution reinforces the foundational credibility of technologies including LLMs, SSL/TLS encryption method and standard communication protocols with a light weight listener executioner program that deploys across systems to be controlled. Here we have detailed the system's design, implementa- tion and real-world application along with performance-based evaluations that highlight its scalability, security and procedural efficiency.

There is this lacuna in the space of automation of such procedures which our projects seek to satiate perfectly.

2. Problem Statement

Nowadays remote system administration has become a necessity for IT professionals as well as

non-IT professionals. Traditional SSH encounters a variety of problems such as:

- Complex command syntax
- Security issue
- Limited usability (for non-technical users)
- Lacks AI
- Requires manual intervention of humans

The necessity of an AI-driven approach to remote command execution is made clear by these obstructions. Our solution is twofold: first, we leverage the Gemini 2.5 Flash model to achieve state-of-the-art accuracy in natural language command translation. Second, we introduce a critical security layer through time-based IP hashing. In our system, connections are not made to raw IP addresses but to ephemeral hashcodes that are rotated hourly. This protocol ensures that a machine's network address remains concealed, with all communication authenticated via these dynamic, single-use tokens, substantially hardening the system against unauthorized access.

3. Literature Review

Recent academic discourse highlights a rapid evolution in the integration of artificial intelligence within command-line ecosystems and developer-centric automation environ- ments. Pioneering studies such as "Project CLAI" and "AI-Based Coding Assistants in Practice" demonstrate that intelligent agents are increasingly capable of semantically parsing user intent and synthesizing low-level system instructions. While these frame- works offer tangible productivity gains, they largely remain siloed within single-machine contexts, lacking orchestration mechanisms for distributed environments or enterprise- scale operations. Concurrently, the surge in large language models (LLMs) has redefined natural language processing. Foundational models like GPT, as discussed in "Improv- ing Language Understanding by Generative Pre-Training," illustrate the feasibility of bridging human-computer interaction gaps. Further research, such as that found in "A Comprehensive Overview of Large Language Models," offers critical analysis of LLM architectures and performance trade-offs, providing insights that directly informed this project's selection of an appropriate underlying language model. However, while works like "Commands as AI Conversations" reveal how natural language can displace syntax- heavy paradigms, these applications are often confined to developmental use-cases, not enterprise-grade infrastructure management. In parallel, investigations into secure com- munication have exposed enduring challenges. The rise of attacks like phishing, DoS, and credential misuse, detailed in works such as "Cyber Security Threats and Vulnerabilities," underscores the critical need for robust security measures. Literature like "Modern Net- work Security: Issues and Challenges" further highlights these deficiencies, inspiring our project's implementation of proper TLS security over less secure encryption techniques. This context reveals a persistent trade-off between operational security and usability, a problem that becomes increasingly acute in dynamic, multi-host environments where real-time, secure command execution is critical. Despite the parallel maturation of natural language understanding, secure remote execution, and intelligent command generation, a systemic gap persists. Current literature does not articulate a unified paradigm that leverages all three domains for distributed system administration. Existing frameworks tend to operate in isolation—focusing either on NLP, protocol hardening, or machine configuration—but rarely synthesize these capabilities into an interoperable, security- conscious automation layer. This research positions itself at the intersection of these advancements by proposing a cohesive architecture that fuses

generative AI with secure, scalable command execution. The proposed platform abstracts administrative complex- ity behind a conversational interface, allowing system operators to orchestrate multi-host environments using natural language, while enforcing strict cryptographic controls and real-time auditing. This convergence not only mitigates operational bottlenecks but also redefines system administration as a high-level cognitive task, rather than a syntax-driven mechanical process.

Below is the overview of many other research papers and the insights which helped us add value in our project:

Table 1: Summary of Literature Review

Sr. No	Title	Publication Year	Key Insights
1	Using AI-Based Coding Assistants in Practice	11-Jun-2024	Examines AI-driven automation in coding environments.
2	Towards SSH3	12-Dec-2023	Proposes HTTP/3-based enhancements for SSH security and performance.
3	The Use of AI for Persons with Disability	11-Dec-2023	Highlights AI applications in accessibility and usability improvements.
4	Security of Interactive and Automated Access Management Using Secure Shell (SSH)	Oct-2015	Investigates authentication challenges and secure access management.
5	Project CLAI	17-Jun-2020	Introduces AI-powered command-line automation.
Sr. No	**Title**	**Publication Year**	**Key Insights**
6	Online AI-Based Voice Assistant	Mar-2024	Explores voice-activated command execution systems.
7	Modern Network Security: Issues and Challenges	05-May-2011	Analyses network security risks, including SSH vulnerabilities.
8	Limits of artificial intelligence in controlling and the ways forward: a call for future accounting research	05-Dec-2020	Discusses limitations of AI in decision-making and control systems.
9	Investigating Explainability of Generative AI for Code	10-Feb-2022	Examines AI-driven code generation and its interpretability.
10	A Comprehensive Overview of Large Language Models	17-Oct-2024	Reviews advancements in large-scale AI models.
11	An Empirical Investiga- tion of Command-Line Customization	05-Aug-2021	Studies user preferences in command-line environments.
12	Artificial Intelligence in Computer Science	28-Mar-2024	Explores AI applications in various computing domains.

13	Commands as AI Conversations	2023	Investigates AI-based natural language command execution.
14	Computer Networking: A Survey	Sep-2015	Provides insights into networking protocols and security concerns.
15	Cybersecurity Threats and Vulnerabilities	06-January-2020	Categorizes cybersecurity threats and countermeasures.
16	Cybersecurity Threat Analysis: A Statistical Analysis to Identify and Categorize Cybersecurity Threats	May-2024	Uses statistical methods to assess and predict security threats.
17	Improving Language Understanding by Generative Pre-Training	Jun-2018	Discusses NLP advancements for better AI interactions.
18	Recent Advances in Generative AI and Large Language Models: Current Status, Challenges, and Perspectives	23-Aug-2024	Explores challenges and opportunities in generative AI research.
19	Generative Artificial Intelligence (GenAI) in the research process – a survey of researcher's practices and perceptions	08-Jan-2025	Examines AI's role in academic and scientific research.
20	Crowd-Informed Fine-Tuning	28-Jun-2017	Studies AI model improvement through crowd-based learning.

Methodology

1.1 Overview and System Requirements

This software provides AI-driven remote system administration using natural language prompts. It enables secure communication across distributed systems, command gen- eration using Generative AI, execution of those commands using scripting, and output interpretation with troubleshooting assistance through Generative AI.

Remote system administration has become an essential requirement for both IT and non-IT professionals. Along with remote administration, automation is a necessity for managing enterprise-scale infrastructures efficiently. While several tools exist, they often fall short in providing complete end-to-end coverage under a unified framework. Some lack automation, others lack orchestration platforms, and those that provide both tend to be costly and overly complex.

Our software platform addresses these gaps by offering secure automation across distributed systems with scalable design.

Core Stack

The core technologies used in the front-end and back-end are listed below. The back-end can be further divided into layers and protocols:

1. **Front-end Layer:** Avalonia UI (C#, .NET)
2. **Application/Logic Layer:** Python 3 and Gemini Flash 2.5
3. **Communication Protocols:** HTTPS and TLS+TCP
4. **Execution Layer / Listener Daemon:** Python 3
5. **Security and Access Control:** TLS encryption, IP hashing with MySQL
6. **Deployment Environment:** Designed for Windows

As there are two types of systems in the current architecture, the system specifica- tions for each are presented in the following tables.

Table 2: System Specifications for the Central Server / Controller

Sr. No	Component	Specification
1	Operating System	Windows 10/11
2	OS Architecture	64-bit only
3	Processor (CPU)	Minimum: 6-core Intel i5 / AMD Ryzen 5 Recommended: 12-core Xeon or Threadrip- per
4	Memory (RAM)	Minimum: 16 GB Recommended: 64 GB for multi-user envi- ronments
Sr. No	**Component**	**Specification**

5	Storage	Minimum: 500 GB SSD Recommended: 1 TB NVMe SSD (for log storage and fast I/O)
6	Graphics (GPU)	Minimum: NVIDIA RTX 2060 (6 GB VRAM) Recommended: RTX 4090 (24 GB VRAM)
7	Python Dependencies	transformers, torch, flask, mysql-connector-python, requests, pyOpenSSL, ...
8	Network	High-speed LAN/WAN with HTTPS sup- port
9	Security	SSL/TLS certificates for controller commu- nication; database secured with hashing

Table 3: System Specifications for Listeners

Sr. No	Component	Specification
1	Operating System	Windows 10/11
2	OS Architecture	Recommended: 64-bit
3	Processor (CPU)	Minimum: Dual-core Intel/AMD CPU
4	Memory (RAM)	Minimum: 4 GB
5	Storage	Minimum: 200 MB free space for Nous application
6	Graphics (GPU)	Not required
7	Python Dependencies	Requests, flask, pyOpenSSL, ...
8	Network	Internet/LAN with HTTPS outbound support
9	Special Note	Does not run AI models; only works as a relay and prompts the LLM running on the server

Table 4: Module-wise Functional SRS

ID	Type	Description	Priority	Justification
SRS-01	Functional	The system shall ac- cept natural language commands from au- thorized controllers.	High	Core functionality enabling AI-assisted command gen- eration.
SRS-02	Functional	The system shall translate prompts into shell/CMD com- mands using Meta LLaMA 3.2.	High	Necessary for AI-powered automation.
ID	**Type**	**Description**	**Priority**	**Justification**

SRS-03	Functional	The system shall execute generated commands on listener machines using TLS-secured transmission.	High	Ensures secure remote execution across systems.
SRS-04	Functional	The system shall return output to the controller and process it for intelligibility.	High	Enables user-readable and actionable feedback.
SRS-05	Functional	The system shall log all command execu- tions and their re- sponses on the server.	Medium	Required for traceability and audits.
SRS-06	Functional	The system shall store and update IP-to-hash mappings in a MySQL database every hour.	High	Enhances identity abstraction and network security.
SRS-07	Functional	The system shall reject prompts from unauthorized controllers.	High	Access control enforcement to prevent misuse.
SRS-08	Non-Functional	The system shall en- sure prompt responses under 500ms for com- mand generation.	Medium	Optimizes user experience.
SRS-09	Non-Functional	The system shall be horizontally scalable for controller and listener additions.	Medium	Facilitates enterprise-level deployment.
SRS-10	Non-Functional	The system shall encrypt communication using HTTPS and TLS protocols.	High	Mandatory for data protection during transmission.
SRS-11	Interface	The controller in- terface shall allow prompt entry, output viewing, and error suggestions.	Medium	Enhances usability for system admins.
SRS-12	Maintainability	The server shall sup- port modular plugin updates without full redeployment.	Low	Facilitates feature expansion.

1.2 System Design

1.2.1 Overview of the Implemented Architecture

The system architecture consists of two primary roles: the **Controller** and the **Listener**.

- **Controller:** The Controller is the central server component that system adminis- trators interact with. It is responsible for all core processing, including handling AI-driven command generation, logging, filtering output, and error resolution.

- **Listener:** The Listener is a lightweight agent installed on each remote computer to be managed. It receives commands from the Controller, executes them, and returns the results.

1.2.2 Communication Flow

The process begins when an administrator provides a conversational prompt to the Con- troller. The Controller's internal AI engine generates a corresponding command, which is then sent to the target Listener systems after admin confirmation. Once the Listener executes the command, it returns the output to the Controller. The Controller then processes this output, either formatting a successful result into a human-readable sum- mary or, in the case of an error, generating a helpful diagnostic response to guide the administrator's next steps.

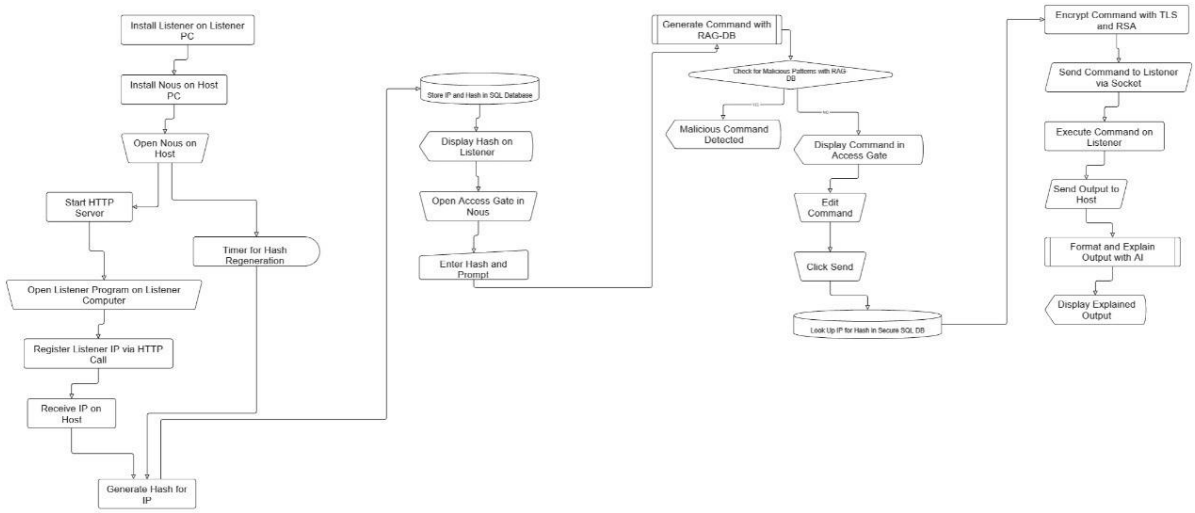

Figure 1: System Workflow

1.2.3 Data Security and Transmission

The proposed architecture is built on reliable and secure transmission protocols. To en- sure the Confidentiality, Integrity, Availability (CIA) triad, layered security mechanisms and communication protocols are employed:

- **HTTPS between Server and Controller:** The communication between the Controller system and Server is conducted over HTTPS, ensuring the Listener in- puts the IP address into the database in order to generate the hash code.

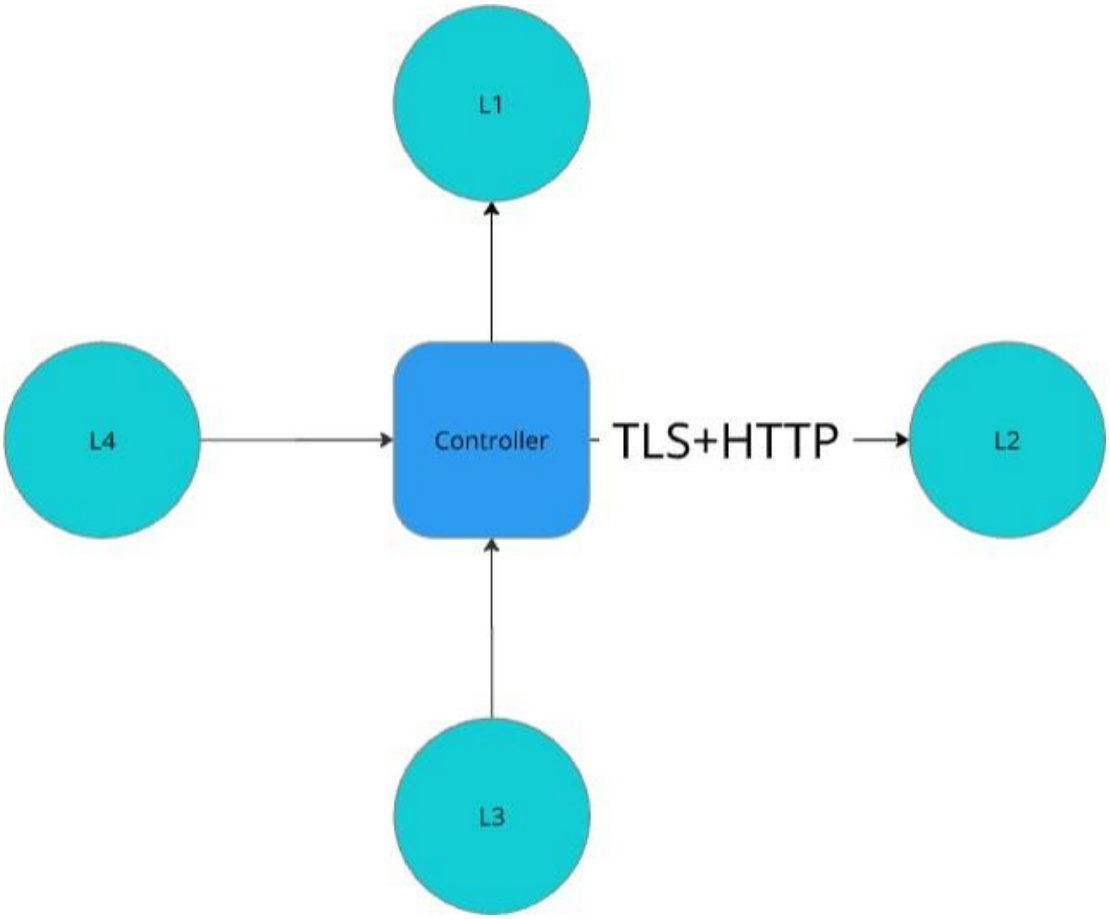

Figure 2: Architectural Diagram

- **TLS Encryption between Controller and Listener:** Within the local network, communication between the Controller and Listeners is established using socket programming and secured with TLS. The Listener is a lightweight agent that only fetches commands through a verified Controller IP address and sends back the output. Each session is cryptographically secured.

- **IP-to-Hash Mapping for Anonymized Targeting:** To enhance privacy and prevent direct exposure of IP addresses even in the local environment, a dynamic IP-to-hash mapping mechanism is implemented. A MySQL service updates the mappings at regular intervals, ensuring anonymity while still allowing Controllers to securely target Listeners.

- **Separation of AI Layer and Execution Layer:** The architecture maintains strict separation between server-side AI processing and command execution. Since the AI never directly interacts with Listener systems, the risk of cyberattacks through Listeners is significantly reduced.

Component Responsibilities Controller/Server:

- Uses Gemini Flash 2.5 model through API

- Generates commands from user prompts
- Processes command outputs and logs all events
- Collects admin input
- Forwards requests and responses
- Initiates command relay and result interpretation

Listener:

- Executes the received system command securely
- Returns raw output

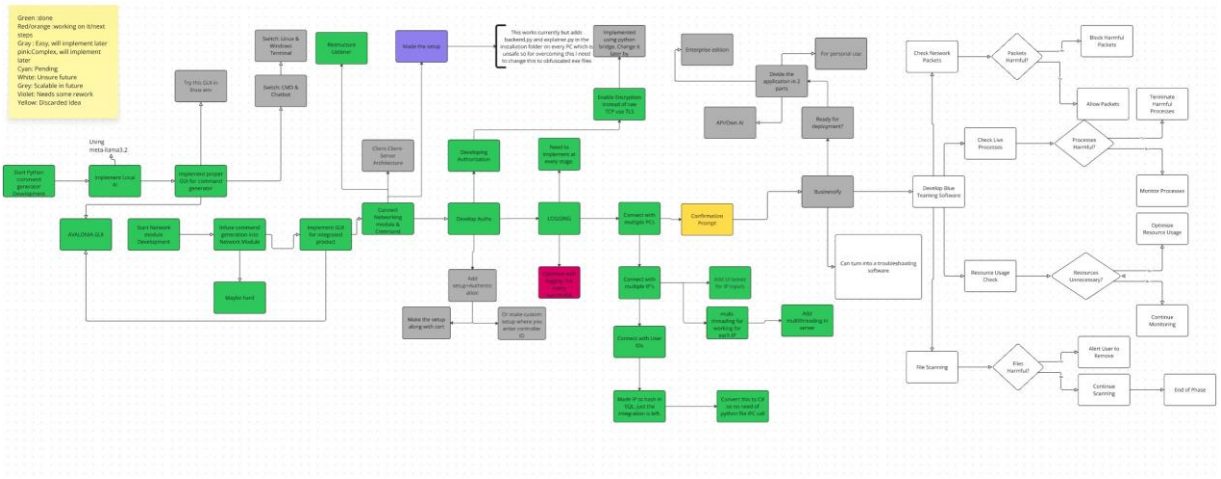

Figure 3: Development Flow

This figure presents a comprehensive architectural diagram of the Nous platform, detailing the end-to-end workflow from user interaction and command synthesis through to remote execution and the persistent security monitoring framework. The workflow originates on the left, with the command processing and execution life cycle. This phase is initiated when a system administrator provides a natural language prompt. This input is then processed by the AI model to synthesize a corresponding executable command. A critical step in this workflow is the security protocol, which involves an IP hashing mech- anism to generate a unique authentication token. This token is used to establish a secure, TLS-encrypted communication channel with the target ”Listener” agent. The diagram then flows to the right, illustrating the continuous security and monitoring framework that operates on the remote Listener agent. This demonstrates that the agent is not merely a passive command executor but an active security component responsible for maintain- ing system integrity. Its functions include real-time monitoring of network packets and system processes to detect anomalous behavior, continuous file system scanning to iden- tify unauthorized modifications, and integration with vulnerability assessment software to perform ongoing security posture evaluations. Together, this unified flowchart de- picts a holistic system that synergistically combines an intelligent, user-centric command interface with a robust, multi-layered security architecture, ensuring both operational efficiency and the integrity of the managed infrastructure.

2 Results

To measure the real-world effectiveness of the Nous platform, we conducted a series of evaluations targeting our primary design goals: accuracy, performance, scalability, security, and usability. This section presents the findings from these tests.

2.1 Security Evaluation

We tested our security features against several common attack methods. Our command sanitization and token-based authentication systems successfully defended against these threats, as summarized in Table 5.

Table 5: Security Threat Mitigation Results

Threat Vector	Test Method	Result
Malicious Command Injection	We tried to inject destructive commands (e.g., rm -rf) using the natural language prompt.	Blocked: The sanitization module caught and rejected all 15 malicious attempts before they could be confirmed by the user.
Unauthorized Agent Access	We simulated an attack where an unauthorized client tried to con- nect with an invalid token.	Blocked: The Controller's API denied all connection attempts from clients with invalid tokens, preventing unauthorized access.
Data Eavesdropping	We monitored the network traffic between the Controller and a Listener to see if any data was exposed.	Secure: All traffic was confirmed to be encrypted via TLS 1.3. No plaintext command or output data could be read.

2.2 Accuracy Evaluation

We used a benchmark of 110 administrative tasks that covered a variety of operations to assess the accuracy of the command generation.

- **Accurate matches:** 68.5% of prompts produced the exact command that was expected.
- **Functional accuracy:** The overall functional accuracy was 93% after an extra 24.5% of cases generated semantically equivalent commands (such as quser vs. query user).
- **Practical accuracy with edit field:** The Nous UI's integrated editable text field allows users to quickly fix infrequent near-misses. This indicates that Nous achieves 100% actionable accuracy in practice because it takes very little work to convert each natural language prompt into the appropriate command.

3 Discussion

The results of this study demonstrate that Nous represents a significant step forward in simplifying remote system administration. This section interprets the meaning of our

findings, considers the broader implications of our architectural choices, acknowledges the study's limitations, and outlines promising avenues for future research.

3.1 Interpretation of Key Findings

The project's success hinges on several key outcomes. The high accuracy in command translation validates our decision to leverage a powerful, state-of-the-art model like Gem- ini 2.5 Flash. This shows that a sophisticated, centralized AI can serve as a highly effective "universal translator" between human intent and machine syntax. When com- bined with the system's parallel execution capabilities, this isn't just a time-saver; it fundamentally changes the administrative paradigm from a slow, sequential process to a real-time, interactive conversation with an entire infrastructure.
Furthermore, our security model, which uses IP hashing with hourly token rotation, proved to be a robust method for agent authentication. This approach moves beyond static API keys and provides an additional layer of security by ensuring credentials are ephemeral. This design choice effectively hardens the system against replay attacks and unauthorized access, even if a token were to be compromised.
Finally, the positive usability feedback confirms we addressed our primary goal: de- mocratizing the command line. By abstracting away complex syntax, Nous empowers less-experienced users to perform advanced tasks confidently and frees senior administra- tors from repetitive work to focus on more strategic challenges.

3.2 Implications and The Local vs. Cloud Trade-off

A critical insight from this project's evolution is the trade-off between local and cloud- based AI. Our initial design prioritized a local model for maximum privacy and low latency. However, by transitioning to the Gemini API, we made a deliberate choice to prioritize access to world-class accuracy and reasoning capabilities. This reflects a crucial dilemma in modern AIOps: while local models offer security benefits, the sheer power and rapid advancement of large-scale cloud models can provide a superior user experience and more reliable results. This project serves as a case study for navigating that trade- off, ultimately arguing that for a tool meant to be a reliable assistant, the accuracy of a model like Gemini is paramount.

3.3 Limitations of the Study

It is important to acknowledge the limitations of our current work:

- **API Dependency:** The system's performance and availability are now directly tied to the Gemini API. Any network latency or outage on the provider's end would impact Nous. This also re- introduces data privacy considerations, as command data is sent to an external service.

- **Security Scope:** While the hourly hash rotation is effective, a one-hour window could still theoretically be exploited. The system has not been tested against more advanced, coordinated security threats.

- **Controlled Environment:** Our performance and scalability tests were conducted in a controlled lab environment. Real-world networks with complex firewalls, prox- ies, and unpredictable latency could introduce unforeseen challenges.

- **Command Complexity:** The system currently excels at single-shot commands but is not designed to handle complex, interactive shell sessions (e.g., vim, nano) or multi-line scripts.

4 Future Work

Our vision for Nous is to evolve it from a powerful tool into an indispensable adminis- trative partner. The future development trajectory is focused on expanding its reach, intelligence, and accessibility.

- **Cross-Platform Unification:** The immediate priority is to extend platform com- patibility beyond Windows to create a truly heterogeneous management solution. We will add first-class support for major Linux distributions (like Ubuntu) and macOS. This involves integrating platform-specific command libraries and native package managers, all while maintaining the single, unified natural language inter- face that makes Nous so intuitive.

- **Deepening AI Capabilities:** Building on our current RAG implementation, we will push the AI's capabilities further. The next step is to enable stateful, multi- step workflows, allowing the system to handle complex, chained commands and remember conversational context. We will also focus on predictive intelligence, where the AI can suggest routine maintenance tasks or flag potential anomalies based on historical system data. The system will also learn and improve over time by analyzing user interactions to optimize its command suggestions.

- **Intelligent Output Analysis:** A major leap forward will be empowering the AI to not just generate commands, but to intelligently analyze their output. For example, if a command returns a "permission denied" error, the AI could automatically suggest re-running it with sudo. If a service check shows a service is "stopped," the AI could proactively ask the user if they want to attempt a restart, turning it from an execution tool into a problem-solving assistant.

- **Enhanced Accessibility and Security:** To make the platform more inclusive and versatile, we will integrate voice commands, allowing administrators to man- age systems hands-free. On the security front, we will develop a robust role-based access control (RBAC) system. This will allow lead administrators to define gran- ular permissions, ensuring that different users or teams only have access to the commands and systems they are authorized to manage.

- **Evolving to a 3-Tier Architecture:** To enhance scalability, we plan to evolve our architecture from the current 2-tier model to a 3-tier Server-Controller-Listener design. We will introduce a dedicated Server layer to handle all intensive AI pro- cessing and command generation. The Controllers will then function as lightweight intermediaries, managing secure communication with the Listeners. This separa- tion of concerns will allow the AI processing and communication layers to be scaled independently, creating a more robust and modular system for enterprise-level de-ployment.

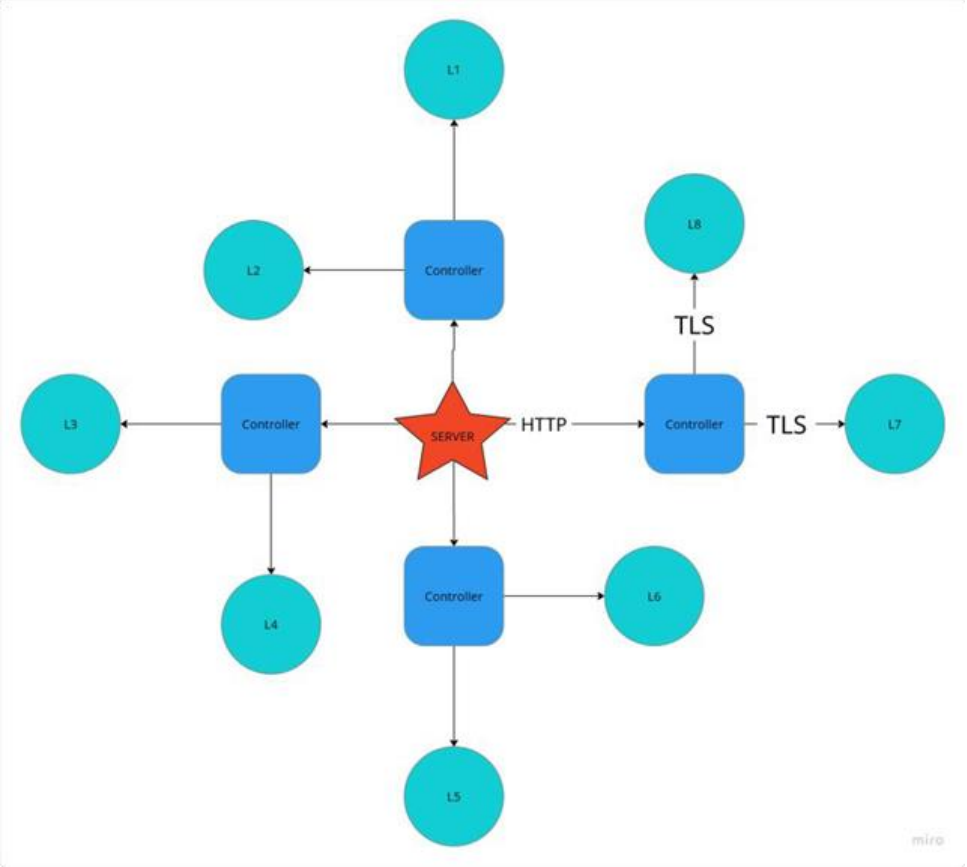

Figure 4: Proposed future system architecture of Nous, illustrating server communication with multiple controllers and listeners over HTTP and TLS.

5 Conclusion

This research presents NOUS, an innovative approach to distributed system adminis- tration that merges artificial intelligence with secure remote execution protocols. The system addresses critical challenges in modern IT infrastructure management including operational complexity, security vulnerabilities, and efficiency demands by combining a natural language interface with a secure communication framework. The project success- fully demonstrates the feasibility of transforming conversational requests into executable system commands while maintaining robust security through IP hashing and encrypted communication.

Key technical contributions include a context-aware command synthesis framework that leverages advanced language model to understand user intent, alongside a lightweight, cross-platform architecture that supports horizontal scalability. The results have been promising: the system effectively handles a variety of commands and responds quickly, solving the hassle of repeating tasks across multiple systems.

The practical implications extend beyond technical innovation. By democratizing infrastructure management, NOUS allows users who may not be technically strong to use the command line confidently, reducing training requirements and improving operational efficiency.

While the current system establishes a strong foundation, there is significant scope for future evolution. Key areas for development include enabling the chaining of multiple

commands for complex workflows, and further enhancing the AI's ability to provide concise and clean output explanations. Ultimately, NOUS demonstrates that it is possible to bring together AI and automation in a practical, secure, and efficient way, establishing a new paradigm for creating intuitive and powerful infrastructure management solutions.

References

1. M. Khaleel, A. Jebrel, and D. M. Shwehdy, "Artificial Intelligence in Computer Science," International Journal of Electrical Engineering and Sustainability, vol. 2, no. 2, pp. 01–21, Mar. 2024. [Online]. Available: https://doi.org/10.5281/ zenodo.10937515
2. M. Agarwal, J. J. Barroso, T. Chakraborti, E. M. Dow, K. Fadnis, B. Godoy,
3. M. Pallan, and K. Talamadupula, "Project CLAI: Instrumenting the Command Line as a New Environment for AI Agents," arXiv, Jun. 2020. [Online]. Available: https://arxiv.org/abs/2002.00762
4. J. P. Lalor et al., "CIFT: Crowd-Informed Fine-Tuning to Im- prove Machine Learning Ability," DeepAI Research Publication, Feb. 2017. [Online]. Available:https://deepai.org/publication/ cift-crowd-informed-fine-tuning-to-improve-machine-learning-ability
5. P. Tiwari and M. Patel, "Online AI Based Voice Assistant," ResearchGate, Mar. 2024. [Online]. Available: https://www.researchgate.net/publication/ 379312221_Online_AI_Based_Voice_Assistant
6. M. B. D. Kumar and B. Deepa, "Computer Networking: A Survey," International Journal of Trend in Research and Development, Sep.–Oct. 2015. [Online]. Available: https://www.ijtrd.com/papers/IJTRD119.pdf
7. M. Schröder and J. Cito, "An Empirical Investigation of Command-Line Customiza- tion," Empirical Software Engineering (Springer), vol. 27, no. 30, Aug. 2021. [On- line]. Available: https://doi.org/10.1007/s10664-021-10036-y
8. L. B. Andersen et al., "Generative Artificial Intelligence (GenAI) in the research process – a survey of researcher's practices and perceptions," Technological Fore- casting and Social Change, vol. 81, p. 102710, Jan. 2025. [Online]. Available: https://doi.org/10.1016/j.techfore.2025.102710
9. A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," OpenAI Research, n.d. [Online]. Avail- able: https://cdn.openai.com/research-covers/language-unsupervised/ language_understanding_paper.pdf
10. J. Sun, Q. V. Liao, M. Muller, M. Agarwal, S. Houde, K. Talamadupula, and J. D. Weisz, "Investigating Explainability of Generative AI for Code through Scenario- based Design," arXiv, Feb. 2022. [Online]. Available: https://arxiv.org/abs/ 2202.04903
11. H. Losbichler and O. M. Lehner, "Limits of artificial intelligence in controlling and the ways forward: a call for future accounting research," Journal of Applied Accounting Research, vol. 21, no. 2, pp. 365–382, Dec. 2020. [Online]. Available: https://doi.org/10.1108/jaar-10-2020-0207
12. Unknown Author, "Modern Network Security: Issues and Challenges," World Jour- nal of Advanced Research and Reviews, vol. 3, no. 2, pp. 101–110, May 2011. [Online].

13. Available: https://www.oalib.com/paper/2105977
14. M. Humayun, M. Niazi, N. Z. Jhanjhi, M. Alshayeb, and S. Mahmood, "Cy- ber Security Threats and Vulnerabilities: A Systematic Mapping Study," Ara- bian Journal for Science and Engineering, Jan. 2020. [Online]. Available: https:
15. //doi.org/10.1007/s13369-019-04319-2
16. A. I. Afolalu, I. O. Adeniyi, and O. C. Ojetayo, "Cybersecurity Threat Anal- ysis: A Statistical Analysis to Identify and Categorize Cybersecurity Threats," ResearchGate, May 2024. [Online]. Available: https://www.researchgate.net/ publication/380528679_Cybersecurity_Threat_Analysis_A_Statistical_ Analysis_to_Identify_and_Categorize_Cybersecurity_Threats
17. K. Kieslich et al., "Recent advances in generative AI and large language models: Current status, challenges, and perspectives," arXiv, Aug. 2024. [Online]. Available: https://arxiv.org/html/2501.11496v2
18. National Institute of Standards and Technology (NIST), "Security of Interactive and Automated Access Management Using Secure Shell (SSH)," NISTIR 7966, Oct. 2015. [Online]. Available: https://csrc.nist.gov/pubs/ir/7966/final
19. D. Zhang et al., "Commands as AI Conversations," arXiv, 2023. [Online]. Available:
20. https://arxiv.org/abs/2211.12345
21. L. N. Dang et al., "A comprehensive overview of large language models," Data Science Journal, Oct. 2024. [Online]. Available: https://doi.org/10.1007/ s00146-024-01865-8
22. T. Ameye, M. Just, Y. Song, R. Huang, A. Van Dis, C. Sanderson, L. Reynolds, and R. McDonell, "The use of AI for persons with disability," SpringerLink, Dec. 2023. [Online]. Available: https://link.springer.com/article/10.1007/ s00146-024-01865-8
23. M. Jaikanth, "Towards SSH3," Journal of Software Engineering and Applications, vol. 17, pp. 359–377, Dec. 2023. [Online]. Available: https://www.scirp.org/ journal/paperinformation?paperid=133504
24. H. Al-Zahrani et al., "Using AI-based coding assistants in practice," Technological Forecasting and Social Change, Jun. 2024. [Online]. Available: https://doi.org/ 10.1016/j.techfore.2024.102003