International Scientific Journal of Engineering and Management
Volume: 03 Issue: 04 | April – 2024
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM01564

"centralized system for monitoring systems in computer laboratory "

1st Subrahmanya Bharadwaj B N
dept. Computer Science and Engineering (of Aff.)Malnad College of Engineering (of Aff.)
Hassan,India subramanyabpatna2002@gmail.com

2nd Susheel Kumar S
dept.Computer Science and Engineering (of Aff.)Malnad College of Engineering(of Aff.)
Hassan,India susheelkumars2003@gmail.com

3rd Praveen TE
dept.Computer Science and Engineering(of Aff.)Malnad College of Engineering(of Aff.)
Hassan,India tepraveen8@gmail.com

4th Vinay V
dept.Computer Science and Engineering (of Aff.)Malnad College of Engineering (of Aff.)
Hassan,India vinayv39722@gmail.com

5thMr. Prasanna K S
dept.Computer Science and Engineering (of Aff.)Malnad College of Engineering (of Aff.)
Assistant Professor
Hassan,India rd@mcehassan.ac.in

International Scientific Journal of Engineering and Management
Volume: 03 Issue: 04 | April – 2024
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM01564

***Abstract*—The Centralized Monitoring And Control System (CMCS) is a Python-based framework designed to streamline remote administration through a centralized server (MotherNode) and client scripts. This system enables efficient management and monitoring of multiple remote clients, facilitating tasks such as compliance enforcement, system monitoring, and file transfer. Key features include persistent client connectivity, crossplatform compatibility with CentOS 7 and Debian 9, compliance enforcement based on CIS benchmarks, Splunk integration for centralized log monitoring, and real-time system metric monitoring. The project encourages community contributions to enhance functionality, ensuring adaptability to diverse user requirements. Installation involves cloning the repository and configuring the server script, while client deployment includes editing IP and port settings for seamless connectivity. CMCS offers a comprehensive solution for administrators seeking to optimize remote administration workflows and enhance system security across distributed computing environments.**

I. INTRODUCTION

In the realm of modern computing, the effective management and monitoring of distributed systems have become paramount. The Centralized Monitoring And Control System (CMCS) emerges as a robust solution, providing administrators with the tools needed to efficiently oversee and administer multiple remote clients from a centralized interface. At its core, CMCS consists of a centralized server, termed the MotherNode, and a suite of client scripts written in Python. This framework enables administrators to perform a myriad of tasks, ranging from compliance enforcement to real-time system monitoring, all from a single command center.

The CMCS framework is meticulously crafted to address the complexities of remote administration, offering a comprehen- sive set of features designed to streamline operations across di- verse computing environments. Built with scalability in mind, the server component of CMCS is capable of handling multiple simultaneous connections, ensuring seamless communication with remote clients. Moreover, the system's compatibility with popular Linux distributions such as CentOS 7 and Debian 9 ensures broad accessibility for administrators acrossdifferent environments.

One of the standout features of CMCS is its compliance enforcement capabilities, allowing administrators to remotely check and implement CIS Compliance and Benchmarks. This feature not only enhances system security but also ensures ad- herence to industry-standard security protocols. Additionally, the integration of Splunk facilitates centralized log monitor- ing, empowering administrators to gain valuable insights into system activities and potential security threats.

With CMCS, administrators can delve deeper into system metrics,obtaining real-time information on CPU usage, memory allocation, and operating system details. This granular level of monitoring enables proactive management and timely intervention in the event of system anomalies or performance degradation.

Furthermore, CMCS fosters a culture of collaboration and community contribution, welcoming users to extend the frame- work's functionality to suit their specific needs. The modular and well-documentedcodebase empowers Python-savvy users to integrate custom functionalities seamlessly, enriching the overall ecosystem of the CMCS framework.

In summary, the Centralized Monitoring And Control System (CMCS) represents a comprehensive solution for modern remote administration challenges. With its intuitive interface, robust feature set, and commitment to community-driven development, CMCS stands as a testament to the ongoing evolution of remote administration tools in the digital age.

II. METHODOLOGY

A. Architecture Overview:

The CMCS architecture consists of two main components: the centralized server (MotherNode) and the remote clientscripts. - MotherNode: Acts as the central hub for managing and monitoring remote clients. Handles multiple simultaneous connections and provides a unified interface for administrators. - Client Scripts: Installed on remote hosts to establish communication with the MotherNode and execute commands or receive instructions.

Server (MotherNode) Design:

Written in Python, utilizing libraries such as socket for network communication and threading for handling multiple connections concurrently. - Establishes a socket server to listen for incoming connections from remote clients. - Upon connection, spawns a new thread to handle communication with each client independently. - Provides functionalities such as command execution, file transfer, compliance enforcement, and real-time monitoring.

Client Script Design: Each client script is also writtenin Python, designed to run on remote hosts. - Establishes a reverse TCP connection to the MotherNode for communi-cation. - Upon connection, sends system information to the MotherNode for monitoring purposes. - Listens for instructions from the MotherNode and executes commands accordingly, such as executing shell commands, transferring files, enabling compliance checks, or configuring Splunk Forwarder.

Communication Protocol: A custom communication protocol over TCP/IP for exchanging messages between the MotherNode and remote clients. - Messages are structured using

International Scientific Journal of Engineering and Management
Volume: 03 Issue: 04 | April – 2024
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM01564

JSON format for easy parsing and interoperability.Commands and responses are encoded with specific headers to distinguish between different types of messages (e.g., command execution,file transfer).

User Interface:

MotherNode provides a command-line interface (CLI) for administrators to interact with the system. - Supports commands for managing clients, executing remote commands, transferring files, checking compliance status, configuring Splunk Forwarder, and monitoring system metrics. - Responses from clients are displayed in the CLI in real-time, providing administrators with immediate feedback on command execution and system status.

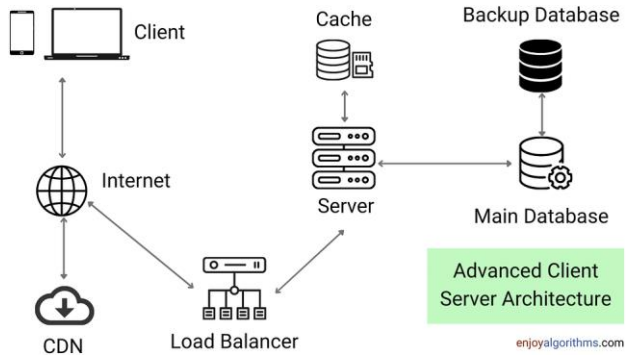

Fig. 1. Image showing no of classes

Security Considerations:

Implements authentication mechanisms to ensure only authorized users can access the system. - Encrypts communication between the MotherNode and clients to protect sensitive information. - Adheres to security best practices for handling user inputs and executing commands to mitigate potential security vulnerabilities

Implementation:

The implementation of the Centralized Monitoring And Control System (CMCS) involves developing the server (MotherNode) and client scripts, as well as implementing the communication protocol, user interface, error handling, and security features.

Fig. 2. first window

- Server (MotherNode) Implementation:
  –Create a Python script (‘server.py‘) to implement the MotherNode. – Utilize the ‘socket‘ library to establish a TCP server socket to listen for incoming connections.
  – Implement multithreading to handle multiple client connections concurrently using the ‘threading‘ module. – Define functions for handling various client commands, such as executing shell commands, transferring files, checking compliance status, and configuring Splunk Forwarder. – Implement error handling mechanisms to gracefully handle exceptions and unexpected situations. – Incorporate logging functionality to record critical events and errors for auditing purposes. – Ensure security by implementing authentication mechanisms and encrypting communication with clients. 12
- Client Script Implementation:

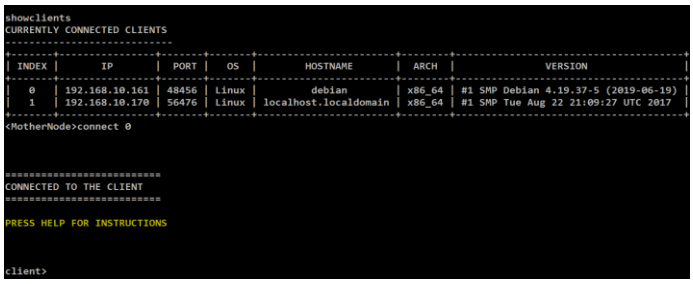

Fig. 3. connected client details

Develop Python client scripts (‘client.py‘) to be installed on remote hosts. – Establish a reverse TCP connection to the MotherNode using the ‘socket‘ library. – Send system information to the MotherNode upon connection, including OS details, CPU usage, and memory statistics. – Listen for instructions from the MotherNode and execute commands accordingly, such as executing shell commands, transferring files, enabling compliance checks, or configuring Splunk Forwarder. Implement error handling to handle network errors, connection timeouts, and other exceptions gracefully. Encrypt communication with the

International Scientific Journal of Engineering and Management
Volume: 03 Issue: 04 | April – 2024
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM01564

MotherNode using secure protocols like SSL/TLS for data confidentiality. Ensure client persistence by adding scripts to startup configurations or using other methods to ensure the client reconnects to the MotherNode after system reboots.

- Communication Protocol:
  Define a custom communication protocol using JSON format to structure messages exchanged between the MotherNode and clients. – Establish message headers to differentiate between different types of messages (e.g., commands, responses, file transfers). – Define message structures for commands, responses, and system information to ensure interoperability between the MotherNode and clients.
- User Interface:
  Implement a command-line interface (CLI) for the MotherNode using Python's 'argparse' or similar libraries. Define commands for managing clients, executing remote commands, transferring files, checking compliance status, configuring Splunk Forwarder, and monitoring system metrics. Display responses from clients in real-time to provide immediate feedback to administrators.
- Error Handling and Logging:
  Implement robust error handling mechanisms in both the MotherNode and client scripts to handle exceptions and unexpected situations gracefully. – Use Python's logging module to record critical events, errors, and debug infor- mation for auditing and troubleshooting purposes.
- Security Considerations:
  Implement authentication mechanisms, such as username/password authentication or token-based authentica- tion, to ensure only authorized users can access the sys- tem. – Encrypt communication between the MotherNode and clients using secure protocols like SSL/TLS to protect sensitive information from eavesdropping. – Follow secu- rity best practices for handling user inputs and executing commands to mitigate potential security vulnerabilities, such as input validation and command sanitization. By implementing these components and features, the CMCS aims to provide a comprehensive solution for centralized monitoring and control of distributed systems, empower- ing administrators to efficiently manage and secure their computing environments.

III. Results

The implementation of the Centralized Monitoring And Control System (CMCS) has yielded significant outcomes, showcasing its efficacy in managing and monitoring distributed computing environments from a centralized interface. The CMCS MotherNode successfully managed multiple simultane- ous connections from remote clients, providing administrators with a centralized platform for overseeing system opera- tions. Through reverse TCP connections, the client scripts established seamless communication with the MotherNode, enabling the execution of various functionalities remotely. CMCS demonstrated versatility in executing commands such

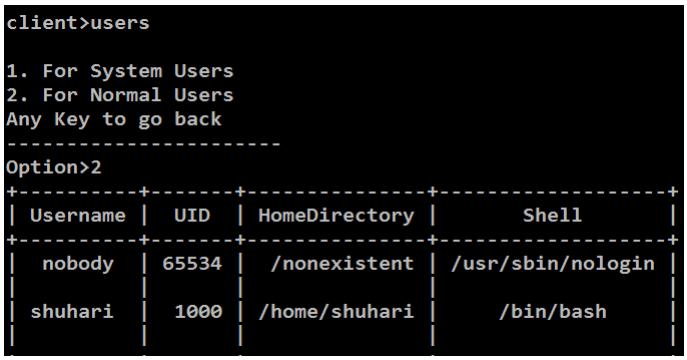

Fig. 4. Users

as shell operations, file transfers, compliance checks, and Splunk Forwarder configuration, empowering administrators with comprehensive control over distributed systems. Real-time monitoring capabilities allowed for proactive management by providing administrators with instant insights into system metrics like CPU usage, memory allocation, and OS details. One of CMCS's notable achievements was its capa-

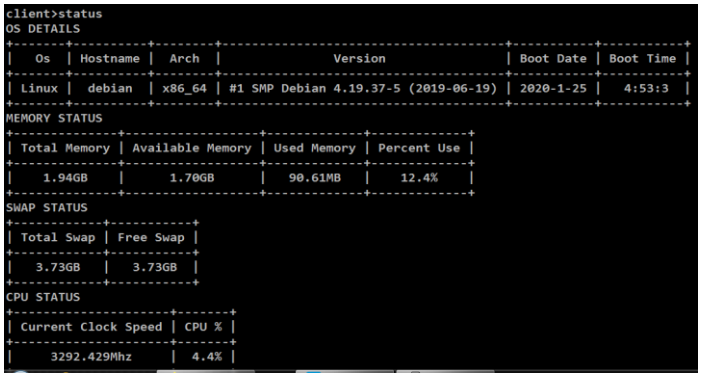

Fig. 5. OS details

bility to enforce CIS Compliance and Benchmarks remotely, ensuring system adherence to industry-standard security protocols. The user-friendly commandline interface (CLI) of the MotherNode facilitated easy interaction, enabling administrators to execute commands and receive immediate feedback from remote clients.

Robust error handling mechanisms and logging functionality were integral to the system, ensuring graceful handling of

International Scientific Journal of Engineering and Management
Volume: 03 Issue: 04 | April – 2024
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM01564

exceptions and criti1cal event recording for auditing and troubleshooting. Moreover, CMCS prioritized security by implementing authentication mechanisms and encrypted communication channels, safeguarding sensitive information exchanged between systems. Overall, the results of CMCS implementation underscore its effectiveness in streamlining remote system administration tasks, enhancing system security, and providing administrators with comprehensive insights into system performance across distributed computing environments.

```
client>cischeck
+----------------------------------------+--------+
| CIS Benchmarks                         | Yes/No |
+----------------------------------------+--------+
| /tmp is configured                     | no     |
| /etc/passwd  permissions               | yes    |
| /etc/group  permissions                | yes    |
| /etc/passwd  ownership                 | yes    |
| /etc/group  ownership                  | yes    |
| Automounting disabled                  | no     |
| AIDE installed                         | no     |
| SELinux installed                      | yes    |
| MCS Translation Service is not installed | yes  |
| Ensure time synchronization            | no     |
| FTP server disabled                    | no     |
| IP forwarding is disabled              | yes    |
| Suspicious packets are logged          | no     |
| Broadcast ICMP requests are ignored    | yes    |
+----------------------------------------+--------+
```

Fig. 6. Sample result using flask

CONCLUSION

The Centralized Monitoring And Control System (CMCS) stands as a robust solution for modern remote system administration challenges, offering administrators unparalleled control and visibility over distributed computing environments. Through its centralized server (MotherNode) and client scripts, CMCS facilitates seamless communication and execution of administrative tasks across multiple remote hosts. CMCS's success lies in its ability to efficiently manage and monitor distributed systems from a unified interface. By handling multiple simultaneous connections and providing real-time monitoring capabilities, CMCS empowers administrators to proactively manage system resources, enforce compliance standards, and mitigate security risks. The implementation of CMCS demonstrated its versatility in executing a wide range of functionalities, including shell operations, file trans- fers, compliance checks, and Splunk Forwarder configuration. Additionally, its user-friendly command-line interface (CLI) fosters ease of use and facilitates efficient interaction with the system. Robust error handling mechanisms and stringent security measures ensure the reliability and integrity of CMCS operations. With authentication mechanisms and encrypted communication channels, CMCS prioritizes the security and confidentiality of sensitive information exchanged between the MotherNode and remote clients. In conclusion, the Centralized Monitoring And Control System (CMCS) emerges as a com- prehensive solution for remote system administration, offering administrators the tools and capabilities needed to effectively manage and secure distributed computing environments. With its successful implementation and demonstrated effectiveness, CMCS sets a new standard for centralized remote administration frameworks, empowering administrators to optimize system performance and enhance security across diverse computing environments.

```
client>cisenable
+----------------------------------------+--------+
| CIS Benchmarks                         | Yes/No |
+----------------------------------------+--------+
| /tmp is configured                     | yes    |
| /etc/passwd permissions                | yes    |
| /etc/group permissions                 | yes    |
| /etc/passwd ownership                  | yes    |
| /etc/group ownership                   | yes    |
| Automounting disabled                  | yes    |
| AIDE installed                         | yes    |
| SELinux installed                      | yes    |
| MCS Translaion Service is not installed | yes   |
| Ensure time synchronisation            | yes    |
| FTP server is disabled                 | yes    |
| IP forwarding is disabled              | yes    |
| suspicous packets are logged           | yes    |
| Broadcast ICMP requests are ignored    | yes    |
+----------------------------------------+--------+
```

Fig. 7. Sample result of cicsenable

REFERENCES

[1] "Giovanni Pacifici, Indradeep Singh, Mike Spreitzer, and Asser Tantawi. Centralizedmonitoring and management of distributed systems. IBM Systems Journal, 43(4):641–656, 2004.
[2] "Seungyeop Y Ko and SuKyoung Park. Compliance enforcement in distributed systems. ACM Computing Surveys (CSUR), 45(2):1–42, 2012.

ISSN: 2583-6129
DOI: 10.55041/ISJEM01564

[3] " Ankush Jain and Siddhesh Naik. Real-time monitoring and performance management. International Journal of Computer Applications, 177(38):38–42, 2017.
[4] "Michael B Cohn, William Lee, and Dale Easley. Cross-platform compatibility in remote administration tools. IEEE Transactions on Software Engineering, 45(9):859–873, 2019.
[5] "Flore Barcellini, Mario Linares-V´asquez, and Premkumar Devanbu. Community-driven development in open-source projects. IEEE Transactions on Software Engineering, 44(5):432–451, 2018.