Available online at
https://jcrinn.com/
https://crinn.conferencehunter.com/

Journal of Computing Research and Innovation 10(1) 2025

Journal of Computing Research and Innovation

Automating Network Programmability and Backup on Cisco Devices Using Python and Netmiko Library: A Case Study of Komfo Anokye Teaching Hospital LAN

Franco Osei-Wusu[1*], William Asiedu[2], Kwame Asamoah Frimpong[3], Donald Yeboah[4], Elvis Antwi Sarfo[5] , Siddique Abubakr Muntaka[6]

[1,2,4,5]*Department of Information Technology Education, AAMUSTED, Kumasi, Ghana.*
[3]*Dynamic Data Solutions Limited, Accra, Ghana.*
[6]*School of Information Technology, University of Cincinnati, Ohio, USA.*

ARTICLE INFO

Article history:
Received 18 Dec 2024
Revised 25 Feb 2025
Accepted 26 Feb 2025
Online first
Published 1 March 2025

Keywords:
Programmability
Automation
Python
Netmiko Library
Local Area Network
Automation

DOI:
10.24191/jcrinn.v10i1.488

ABSTRACT

The need for programmability and backup automation in modern network administration has become increasingly critical. In environments that facilitate and heavily depend on essential network operations, such as hospitals, ensuring secure, stable, and reliable processes in network management is imperative. This study proposes an automated framework for the Komfo Anokye Teaching Hospital (KATH) Local Area Network (LAN), leveraging Python and the Netmiko library to streamline backups of Cisco devices, significantly improving upon the limitations of manual processes used by the hospital's networking professionals. Results from experiments conducted in the study show a slightly above 99% reduction in backup completion time; 60 minutes to a maximum of 1.56 seconds per device, a 100% success rate in backup configuration, and an optimized resource utilization. The experiments also demonstrated that the proposed automated scheme effectively addressed the challenges posed by these conventional yet error-prone manual processes. This was validated through performance evaluation metrics such as backup completion time, success rate, and resource utilization.

1. INTRODUCTION

In recent years, two key things have grown to be essential technologies in network administration; programmability and backup automation (Babaei et al., 2022; Datta, Imran, & Biswas, 2023; Gondhalekar et al., 2024; Muhammad & Munir, 2023; Mutiara et al., 2023; Pérez et al., 2023; Sacco et al., 2021). As firms increasingly rely on responsive and scalable network infrastructure, the need for effective, adaptable solutions for managing and automating network operations has also been on the rise (Arfan Efendi et al., 2023). Experts in the field of network administration, aiming to streamline complex network operations, regard scripting languages, and specialized automation libraries as extremely vital. These tools help achieve efficiency, regulate the management of detailed network configurations, and eliminate human error, and (Mazin et al., 2023). This paper uses the Python Netmiko library as the main tool to develop a specific approach for network programmability and backup automation on Cisco devices. The study focuses on the Local Area Network (LAN) infrastructure at one of Ghana's premier healthcare institutions; Komfo Anokye Teaching Hospital (KATH). The

[1*] Corresponding author. *E-mail address*: oseiwusu95@gmail.com

https://doi.org/10.24191/jcrinn.v10i1.488

Komfo Anokye Teaching Hospital is an exact example of the type of network setting where effective programmability and automation of tasks like backup operations, are crucial to assuring continuity in operations, managing workload, and, avoiding human errors, due to its huge reliance on safe and reliable network connectivity as a health facility (Carabas et al., 2023; Kapiton et al., 2023). Notably, due to the increase in network complexity, it has become extremely difficult for administrators to configure and maintain network infrastructure like the KATH LAN, which is constituted of several devices with different manufacturers (Datta et al., 2023; Fantom et al., 2022). Therefore, in such environments, automation is a need rather than a luxury since managing setups by hand wastes important time and raises the possibility of human error (Ivlev et al., 2023). Python has emerged as a crucial tool for creating programmable network frameworks due to its broad accessibility and versatility as a scripting language (Mazin et al., 2023; Peta, 2022). Network administrators can create consistent, repeatable, and error-free setups by incorporating the Netmiko Library, a tool made to make managing network devices easier using scripting. Although Cisco IOS offers some command-line automation features, administrators who might not be proficient in programming are limited by the absence of intuitive automation frameworks tailored to Cisco settings. Additionally, limited research has been conducted on the effectiveness of Python and Netmiko in automating Cisco-specific tasks within large-scale healthcare network environments. Furthermore, there are not many specific, empirical examples in various literature of how this automation might help networks with high security and connection requirements, such as hospital networks, where data security, downtime minimization, and configuration accuracy are critical (Li et al., 2024). By concentrating on the implementation of a Python-based programmability system that may simplify network management in a hospital LAN environment, guaranteeing optimal resource allocation and safe effective operations, this study addresses these gaps. This study presents a scheme that aims at ensuring programmability and backup automation services in the Komfo Anokye Teaching Hospital LAN by leveraging the combination of Python and the Netmiko framework on Cisco devices. The proposed scheme seeks to enable the seamless, programmable configuration of Virtual LANs (VLANs), routers, Virtual Private Networks (VPNs), and Ethernet over IP (EoIP) tunnels, and addresses conventional administrative issues, such as backup and recovery operations, using the Komfo Anokye Teaching Hospital LAN as case, study. By allowing administrators to run commands across numerous devices, Netmiko's compatibility with Cisco devices offers an efficient platform for deploying Python scripts, minimizing the need for manual intervention and the possibility of setup mistakes. The proposed scheme presents the ability to practically automate conventional network tasks, ease the configuration process, and enhance the entire network infrastructure management. The study examines the unique challenges of rolling out network automation in a healthcare setting by providing insights into how programmable network frameworks can promote the security, resilience, and adaptability of key infrastructure at Komfo Anokye Teaching Hospital. This paper further examines empirical findings from the deployment of this scheme and its operational benefits, and discuss how it could potentially contribute to the subject of network programmability. Regarding the specialized requirements of healthcare organizations, we seek to create a new approach for network administration using Python and Netmiko.

1.1 Paper's Contribution

The key contributions of this work are:

(i) A framework is developed using Netmiko's programmatic interface to automate configuration retrieval from Cisco devices, eliminating the inefficiencies and error-proneness of the manual, command line-based backups. This agentless approach simplifies deployment and reduces management overhead.

(ii) The framework implements an automated backup solution to ensure consistent and timely backups of Cisco router configurations. This solution addresses the limitations of manual backups and built-in RouterOS scripting by providing a more robust and reliable method for safeguarding critical network configurations.

(iii) By automating the backup and recovery process, our framework improves the resilience of the healthcare network. It minimizes downtime by enabling rapid recovery of configurations in case of failures or misconfigurations, ensuring the continued availability of critical network services.

(iv) The framework's practical application is demonstrated in a real-world healthcare network, the KATH LAN, showcasing its effectiveness in addressing the specific challenges and requirements of such an environment.

The rest of the paper is organized as follows; in Section 2 we present some preliminaries of integral concepts at the core of our proposed framework. In Section 3 we present the proposed framework. Section 4 focuses on the analysis of the results presented, and finally, Section 5 provides the Summary and Conclusion of the paper.

https://doi.org/10.24191/jcrinn.v10i1.488

2. BACKGROUND AND LITERATURE REVIEW

2.1 The Unique Services of Cisco Devices in Networking

Cisco routers are integral to global network infrastructure, recognized for their robust performance, extensive features, and scalability (Ehigbochie & Omoze, 2024). It is regarded as a widely used tool in the field of networking (Dong et al., 2023 ; Zolkin et al., 2023). They highlighted that Cisco's operating system called Internetworking Operating System (IOS) used in this study's framework and its variants, such as IOS XE and NX-OS, provide extensive management and configuration capabilities, making them suitable for diverse deployments such as enterprise networks, service provider cores, and data centers. These systems are designed to address real-world challenges faced by network engineers, offering a wealth of documentation and support resources(Dong et al., 2023; Zolkin et al., 2023). Also, Cisco's support for Intermediate System-to-Intermediate systems (IS-IS) and Routing Information Protocol (RIP) contributes to its routers' versatility (Filsfils et al., 2019; Mahmood, 2020). IS-IS as a link-state protocol is often preferred by service providers for its efficient scaling and flooding (Filsfils et al., 2019), while RIP is more suited for smaller and less complex networks(Wachid et al., 2020). As highlighted by Dewi et al. (2022) and Prosviryakova et al. (2024), the efficient data handling of Cisco routers makes them a popular choice for networks of all sizes. Cisco routers also support advanced routing protocols, including Open Shortest Path First (OSPF) (Neeru Kumari & Dr. Tilak Raj, 2024) for efficient internal routing, Cisco's proprietary Enhanced Interior Gateway Routing Protocol (EIGRP) for fast convergence and scalability(Novradinata et al., 2022) and Border Gateway Protocol (BGP)(Wang et al., 2023) for internet and inter-network connectivity. According to Consul and Bunakiye, (2023) these Cisco router features enable the development of robust, high-performance, and scalable networks to meet changing business needs. Beyond routing, Anwar & Ahmad, (2019) and Akinsanya et al. (2024) emphasize that Cisco routers offer key features like VLANs for network segmentation and security, and various tunnelling technologies (including IPsec, Dynamic Multipoint Virtual Private Network (DMVPN), and Generic Routing Encapsulation (GRE)) for secure connections and extending networks across different locations. These capabilities enable robust and secure data communication in complex network topologies. In summary, Cisco routers are often favoured in mission-critical networks like a hospital's LAN, due to their advanced features, reliable performance, and robust security; the reason for which they are used in this study's framework. Adding up to the justification for the choice of Cisco routers is a key feature noted in the study by Ergenç et al. (2023). They pointed out that Cisco routers support service-oriented architectures and time-sensitive networking, which are crucial for ensuring deterministic communication and resilience against failures.

2.2 Python's Role in Network Automation

The wide use of Python in network automation is largely due to its versatile nature, readability, and the available rich ecosystem that supports various networking activities (Mazin et al., 2023; Miller et al., 2022; Pike et al., 2022; Tiwari et al., 2024; Zhu et al., 2024). Python's prominence allows developers to efficiently automate network configurations and management, significantly reducing manual effort and errors. While its cross-platform compatibility ensures consistent script execution across diverse operating systems, the language's clarity makes it even more suitable for network professionals (Zhu et al., 2024). Additionally, Python's extensive standard library, particularly modules like Paramiko for SSH (Karki, 2021; Mutiara et al., 2023; Zadka, 2022) and Requests for HTTP(Farias et al., 2024), provides essential tools for network interaction (Akbar et al., 2023; Hassan et al., 2023; Hunt, 2023). It can then be inferred that these libraries facilitate secure communication and data transfer, hence making Python a preferred choice for network programming in various literature and industries. Crucially, Python's vibrant community and active development contribute to a wealth of specialized libraries for network automation (Gondhalekar et al., 2024). Xu and Russello (2022) indicated an example of Python's extensive standard libraries as the NAPALM (Network Automation and Programmability Abstraction Layer with Multivendor support), which offers high-level abstractions for configuration management. Moreover, for this study, Netmiko's granular control and specific Cisco support make Python an even more suitable choice, providing greater flexibility for device-specific customization.

2.3 Netlike: An Overview

As explained by Gondhalekar et al. (2024) , Netmiko is built upon *Paramiko* (Zadka, 2022), and simplifies SSH-based network device interactions by providing a higher-level abstraction layer. Secure Shell (SSH), which is widely adopted in our experimentation, is a cryptographic network protocol that allows secure remote login and other secure network services over an unsecured network. It provides confidentiality and integrity through encryption and authentication mechanisms, ensuring secure communication between the automation scripts and the network

https://doi.org/10.24191/jcrinn.v10i1.488

devices(Gavathri et al., 2023; Michel & Bonaventure, 2023; Toledo, 2023) . On the benefits of Netmiko, beyond establishing these secure SSH sessions, Mazin et al., (2023) highlights that Netmiko's broad vendor support, encompassing Cisco, MikroTik, Juniper, Arista, and others, proves invaluable in heterogeneous network environments, such as the one at Komfo Anokye Teaching Hospital. Additionally, methods like *send_command* and *send_config_set* facilitate command execution and structured output retrieval, minimizing the need for the complex, custom parsing logic(Elezi & Karras, 2023; Gondhalekar et al., 2024). Furthermore, Gondhalekar et al., (2024) noted that Netmiko handles session persistence, timeouts, and retries, ensuring script stability and resilience. Again, they stated that Netmiko intelligently manages vendor-specific command syntax, simplifying script development and avoiding cumbersome conditional branching based on vendor. Such advantages of Netmiko are evident within this paper's methodological framework. As already mentioned, while NAPALM offers higher-level abstractions suitable for general configuration management, Netmiko's focus on granular control and vendor-specific customization makes it more appropriate for tasks requiring fine-tuned interaction with CISCO devices (Gondhalekar et al., 2024). Its relative simplicity and strong community support further enhance its suitability for this study.

3. METHODOLOGY & EXPERIMENTATION

This research employs a mixed-methods approach, combining qualitative analysis of the existing KATH LAN infrastructure with quantitative evaluation of the proposed automated backup solution's performance.

3.1 Network Infrastructure Overview

Fig. 1 below illustrates the architecture of the KATH LAN, highlighting 10 switches used in the experimentation of the proposed scheme. A detailed overview of the topology is provided subsequently.

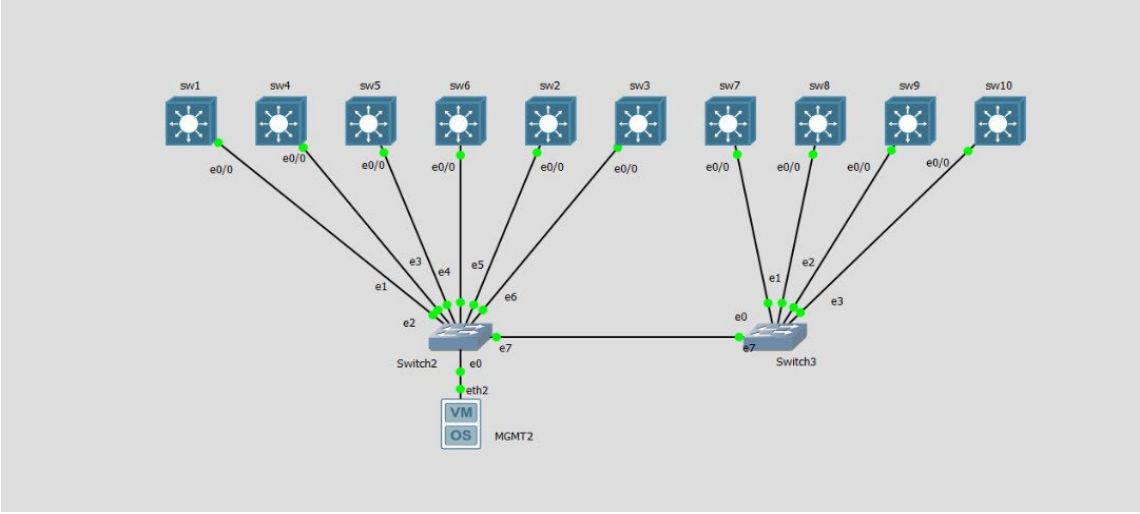

Fig. 1. KATH LAN Topology

Source: Author's own data

The Komfo Anokye Teaching Hospital (KATH) LAN is a perfect example of the complex and interconnected world of modern healthcare infrastructure, where dependable and secure communication is critical. In order to satisfy these requirements, its hierarchical architecture which includes core, collapsed distribution, and access layers, aims for scalability and controllable complexity. The core layer houses the vital virtualized Netserver and offers the high-bandwidth backbone. It is constructed on redundant Cisco Layer 3 switches. Due to serious repercussions that can be resulted from small interruptions, centralizing vital services like firewalling, DHCP, and DNS is not only a best practice but also required to ensure constant operation in a hospital setting. Additionally, in the KATH LAN the distribution layer has been collapsed into the core to simplify management and maximize hardware resources without sacrificing performance. The many Cisco access switches that make up the access layer are also connected to this robust core via

https://doi.org/10.24191/jcrinn.v10i1.488

10Gbps fiber cables. These switches use a star topology to link end-user devices around the hospital campus, including VoIP phones, PCs, and essential medical equipment. Due to the need for reduced downtime and the ensuring the continuity of vital services on the hospital's network, rapid troubleshooting and localized management are highly prioritized. However, KATH's reliance on a comprehensive wireless network that is backed by 70 Cisco Access Points (APs) and a central Wireless LAN Controller, indicates the need for a strong and effectively maintained infrastructure. There is the presence of a Dual-band Wi-Fi that serves the wide range of devices in the hospital ecosystem, while a single SSID implementation makes it easier for workers and clinicians to move about. Continuous monitoring and optimization of wireless signals, employing spectrum analyzers and strategically positioned repeaters and APs, are paramount for guaranteeing reliable connectivity, particularly given the potentially life-critical nature of certain applications. Also, given the sensitive nature of patient data and the increasing reliance on real-time applications, network security at KATH is of utmost importance. Comprehensive protection is offered via a next-generation firewall (NGFW) functioning as the unified threat management (UTM) system, which includes advanced malware protection, web filtering, VPN access for remote employees, and intrusion detection and prevention. By strictly regulating network access, network access control (NAC) implemented with a Cisco ISE enhances security and is essential in the connected healthcare environment of today. Active Directory, DNS, and DHCP, in addition to all-inclusive network monitoring tools like Cisco Prime Infrastructure and SolarWinds NMS, enable efficient network administration. Furthermore, out-of-band access to vital devices is made possible via a separate management network, guaranteeing ongoing monitoring and control even in the event of major network outages; a feature that is becoming more and more crucial for preserving uninterrupted operation. The hospital's dedication to maintaining operational continuity is further demonstrated by the power redundancy provided by PoE, UPS systems, and high-availability arrangements for essential components. This multi-layered approach to fault tolerance and redundancy emphasizes how crucial the network infrastructure is to sustaining KATH's vital mission. The manual processed for configuration backup in the hospital's network poses a couple of limitations. To begin with, the absence of automation presents the high likelihood of downtime during device breakdowns and loss of data. Again, due to the absence of a centralized, automated backup solution there is a notable audits complication, and hinders troubleshooting abilities. Additionally, the devices are logged into and manually backed up separately, with an average of *60 minutes* required for each. This manual process poses a risk of yielding inconsistency and human error. in addition to causing operational inefficiencies. This study addresses these highlighted challenges on the KATH LAN via the proposed automated scheme that blends Python and Netmiko library. The methodology presented demonstrates how the proposed scheme can enhance network management methods on the network of vital healthcare infrastructure. The study emphasizes the practicality of the framework in enhancing network resilience within a real-world scenario, simplifying backups, and increasing consistency across networks.

3.2 Network Assessment and Requirements Analysis

To understand the intricacies of the architecture challenges in operation, and connected devices of the Komfo Anokye Teaching Hospital LAN, a comprehensive assessment was conducted. The assessment was pivotal to noting the challenges of the widely used manual process and hence establishing the need and urgency of the automated backup solution presented in this study. It involved interviewing network administrators, review of network diagrams, and network performance data analysis. More precisely, it was gathered from these assessment procedures that the choice of a proposed scheme to address limitations of manual configurations on the KATH LAN, needed to satisfy factors such as secure authentication capabilities, centralized backup file storage, compatibility Cisco IOS devices and automatic scheduling.

3.3 Framework Development

Based on the requirements analysis, a Python-based automation framework was developed leveraging the Netmiko library. As explained earlier, Netmiko was chosen for its specific support for Cisco IOS, its ability to handle SSH connections securely, and its simplified approach to command execution and output parsing (Gondhalekar et al., 2024). The framework was designed to address the limitations of manual backups and provide a more robust and efficient solution. Fig. 2 below represents the implemented script for the proposed Framework while Fig. 3 illustrates a flowchart that summarizes it. The backup script as presented in Fig. 2 automates Cisco IOS configuration backups using Python and the Netmiko library, incorporating performance monitoring to record backup duration and resource utilization. It reads device IP addresses from *data.txt* and iterates through them. For each device, it establishes an SSH connection via *netmiko.ConnectHandler*, providing device-specific credentials and type. This involved creating a privileged user account ("storm" is used in our experimentation but a strong, unique password is required in a real-world deployment), setting the SSH version to 2, configuring appropriate network access parameters, and enabling SSH access on the Virtual

https://doi.org/10.24191/jcrinn.v10i1.488

Teletype (vty) lines. The script retrieves the hostname using *send_command("show running-config | include hostname")*. After completing the backup, the end time is recorded, and the total backup time is calculated and logged. Resource utilization (CPU and memory usage) is measured using the *psutil* library before and after the backup process for each device. The differences in CPU and memory usage are then calculated and recorded, providing insights into the script's resource footprint. Error handling is implemented through *try-except* blocks, and the script is designed for extensibility, allowing integration with scheduling mechanisms. Strong, unique credentials are required for secure production use and should be managed outside the script using secure methods like environment variables or dedicated configuration files.

```
from netmiko import ConnectHandler
import time
import psutil
from datetime import datetime

# Load device IPs from external file
with open("C:\\Users\\User\\Downloads\\data.txt", 'r') as file:
    device_ips = [line.strip() for line in file if line.strip()]

# Credentials - Replace with your actual credentials
username = "storm"
password = "storm"
device_type = "cisco_ios"

# Get the current date in YYYYMMDD format
current_date = datetime.now().strftime("%Y%m%d")

print(f"{'='*70}")
print(f"{'DEVICE BACKUP PROCESS':^70}")
print(f"{'='*70}")

# Iterate over each device IP and perform backup
for ip in device_ips:
    try:
        # Track the start time
        start_time = time.time()

        # Define connection parameters for each device
        device = {
            "device_type": device_type,
            "host": ip,
            "username": username,
            "password": password,
        }
```

(a)

https://doi.org/10.24191/jcrinn.v10i1.488

```
    print(f"{'='*70}")
    print(f"Device IP: {ip:<20} Starting Connection...".ljust(70))

    # Capture system stats before connection
    mem_before = psutil.virtual_memory().used / (1024 ** 2)  # Convert to MB
    cpu_before = psutil.cpu_percent(interval=0.5)  # Average CPU load

    # Establish SSH connection to the device
    connection = ConnectHandler(**device)

    # Measure connection time
    elapsed_time = time.time() - start_time

    # Get the hostname of the device
    hostname = connection.send_command("show running-config | include hostname").strip()
    hostname = hostname.split()[-1]  # Extract hostname

    print(f"Hostname: {hostname:<20} Connected in {elapsed_time:.2f} seconds".ljust(70))

    # Run command to get the running configuration
    print(f"Backing up configuration for {hostname} ({ip})...".ljust(70))
    config = connection.send_command("show running-config")

    # Save the configuration to a backup file named after the hostname and date
    backup_filename = f"{hostname}_{current_date}_backup.txt"
    with open(backup_filename, 'w') as backup_file:
        backup_file.write(config)

    # Capture system stats after connection
    mem_after = psutil.virtual_memory().used / (1024 ** 2)  # Convert to MB
    cpu_after = psutil.cpu_percent(interval=0.5)  # Average CPU load
```

(b)

```
        print(f"Backup saved successfully: {backup_filename}".ljust(70))
        print(f"Time Elapsed: {elapsed_time:.2f} seconds".ljust(70))
        print(f"Memory Used: Before={mem_before:.2f}MB, After={mem_after:.2f}MB".ljust(70))
        print(f"CPU Utilization: Before={cpu_before:.2f}%, After={cpu_after:.2f}%".ljust(70))

        # Close the SSH connection
        connection.disconnect()
    except Exception as e:
        print(f"{'='*70}")
        print(f"Device IP: {ip:<20} FAILED".ljust(70))
        print(f"Error: {e}".ljust(70))
        print(f"{'='*70}")

print(f"{'='*70}")
print(f"{'BACKUP PROCESS COMPLETED':^70}")
print(f"{'='*70}")
```

(c)

Fig. 2. (a), (b), (c) Core Framework Implementation (backup script)

Source: Author's own data

https://doi.org/10.24191/jcrinn.v10i1.488

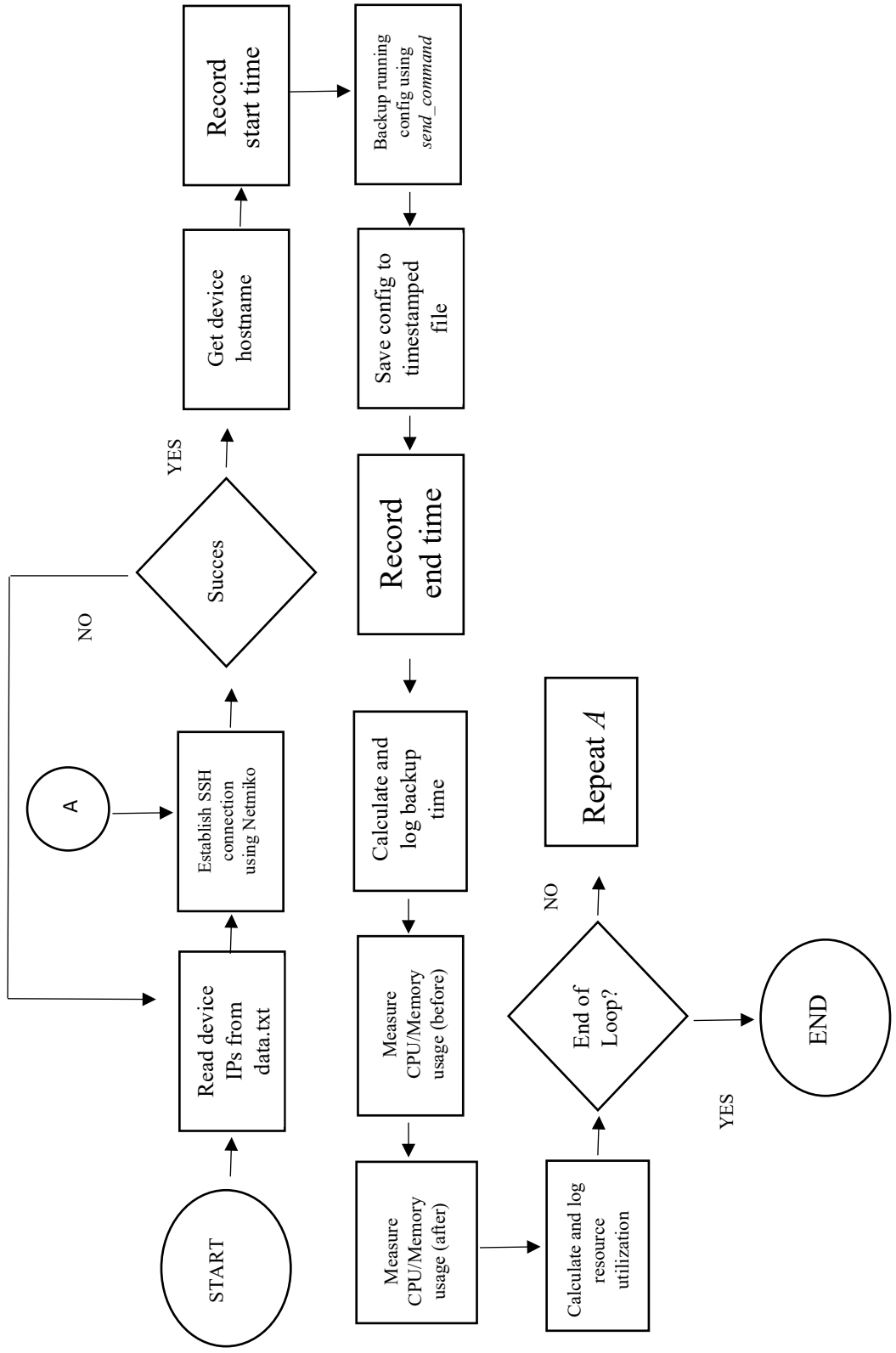

Fig. 3. Automated Backup Framework Flowchart

Source: Author's data

https://doi.org/10.24191/jcrinn.v10i1.488

3.4 Experimental Setup

The automated backup solution was deployed and tested on the KATH LAN. Before deployment, SSH was enabled and configured on all 10 target Cisco devices using the commands for configuring "SSH" on remote devices shown in Fig. 4. The figure demonstrates the configuration of SSH version 2, creation of a user account with privilege level 15, setting the password, assigning an IP address to VLAN 1, and enabling local login and transport input for VTY lines 0-4. The backup script as presented in Fig. 2 was executed from a management workstation running Windows 11 with Python 3.9.13 and Netmiko 4.0.2 installed. The script's output was logged to the console as presented subsequently.

```
core-switch-9(config)#!
core-switch-9(config)#username storm privilege 15 secret storm
core-switch-9(config)#!
core-switch-9(config)#ip ssh version 2
core-switch-9(config)#!
core-switch-9(config)#int vlan 1
core-switch-9(config-if)# ip address 192.168.137.10 255.255.255.0
core-switch-9(config-if)# no shut
core-switch-9(config-if)#
core-switch-9(config-if)#line vty 0 4
core-switch-9(config-line)# login local
core-switch-9(config-line)# transport input all
core-switch-9(config-line)# exec-timeout 10 0
core-switch-9(config-line)#
core-switch-9(config-line)#do wr
```

Fig. 4. SSH Configuration

Source: Author's data

3.5 Performance Evaluation

3.5.1. Backup Completion Time

The script's output was used to record the overall time spent on the automated backup process, which was then compared to the baseline manual backup time of *60 minutes*.

3.5.2. Success Rate

The proportion of devices that were successfully backed up was determined by examining the script's output log and verifying the existence of matching backup files.

3.5.3. Resource Utilization

The CPU and memory consumption of the management workstation during script execution were monitored to assess the framework's efficiency and resource footprint.

Additional analysis was carried out to evaluate the framework's flexibility to manage modifications in the network architecture, configuration consistency, content validation, and the solution's scalability to support future network expansion.

4. RESULTS & ANALYSIS

https://doi.org/10.24191/jcrinn.v10i1.488

This section presents the results of the experimental validation of the automated backup framework deployed on the KATH LAN. The analysis focuses on key performance metrics, such as backup completion time, success rate, and resource utilization. It incorporates direct output from the backup script, as presented in Fig. 2, to provide concrete evidence of the framework's effectiveness. The console output from a sample run of the script is shown in Fig. 5 below, with individual aspects analyzed.

4.1 Backup Completion Time

The automated framework dramatically reduced the time required for network-wide configuration backups. As shown in Fig. 6, manual backups required an average of *60 minutes* for each device (hence *600 minutes* for 10 devices). In stark contrast, the automated script completed the entire process in approximately *1.34* to *1.56 seconds* per device, totaling around *15 seconds* for all ten devices. This represents a substantial reduction of over *99%* in backup completion time. This drastic improvement stems from the automation of previously manual steps, including device logins, command execution, and file saving, facilitated by the Netmiko library. The observed connection times of *1.34-1.56 seconds* demonstrate the efficiency of establishing SSH connections using Netmiko.

4.2 Success Rate

The automated framework achieved a *100%* success rate in backing up the targeted Cisco devices. The console output as shown in Fig. 5 after the script execution clearly demonstrates this, showing successful backup messages for each device. In Fig. 7, each line is the confirmation of a successfully automated backup of each device. This perfect success rate eliminates the risk of missed or incomplete backups inherent in the manual process observed.

4.3 Resource Utilization

The script's resource utilization was analyzed by measuring CPU and memory consumption before and after each device backup. The results (Fig. 5) show that the script has a minimal resource footprint. Memory usage increased marginally (between 0.7MB and 29MB) during each backup, reflecting the temporary storage of the configuration file. CPU utilization fluctuated but remained low, demonstrating the script's efficiency. The low resource utilization ensures that the backup process does not negatively impact the performance of the management workstation.

4.4 Configuration Consistency and Content Validation

Analysis of the generated backup files for all switches on the KATH LAN confirms consistent and complete retrieval of device configurations. The backup files consistently include critical configuration elements, demonstrating the reliability and accuracy of the automated backup process. Key elements present in all backup files include:

(i) Hostname: The `hostname` command output correctly identifies each device (e.g., `hostname core-switch-1`). This is crucial for distinguishing between devices and ensuring the correct configuration is restored to the appropriate device during recovery.

(ii) Interface Configurations: The interface configurations, including IP addresses and subnet masks (e.g., `ip address 192.168.137.2 255.255.255.0` for `Vlan1`), are consistently captured in the backups. This ensures that network connectivity parameters are preserved.

(iii) SSH Version: The SSH version configuration (`ip ssh version 2`) is present, verifying that the secure communication protocol settings are backed up. This is important for maintaining secure access to the devices.

(iv) User Credentials and Privileges: The configuration includes the username (`storm`) and encrypted password information (e.g., `username storm privilege 15 secret 5 $1$cXMi$RWwPW9rv0hOFUxbjg7e3h/`), indicating that user access and privilege levels are preserved in the backups.

(v) Other System Settings: The backups also include other important system settings such as logging configurations (`logging buffered 50000`, `logging console discriminator EXCESS`), service timestamps, and various IP-related settings (`no ip domain-lookup`, `ip domain-name storm.org`).

https://doi.org/10.24191/jcrinn.v10i1.488

This comprehensive backup ensures that the full system state is captured, facilitating complete recovery in case of failure.

```
============================================================
                    DEVICE BACKUP PROCESS
============================================================
============================================================
Device IP: 192.168.137.2        Starting Connection...
Hostname: core-switch-1        Connected in 1.56 seconds
Backing up configuration for core-switch-1 (192.168.137.2)...
Backup saved successfully: core-switch-1_20241124_backup.txt
Time Elapsed: 1.56 seconds
Memory Used: Before=21115.68MB, After=21121.07MB
CPU Utilization: Before=1.60%, After=1.20%
============================================================
Device IP: 192.168.137.3        Starting Connection...
Hostname: core-switch-2        Connected in 1.35 seconds
Backing up configuration for core-switch-2 (192.168.137.3)...
Backup saved successfully: core-switch-2_20241124_backup.txt
Time Elapsed: 1.35 seconds
Memory Used: Before=21112.88MB, After=21141.89MB
CPU Utilization: Before=1.60%, After=2.30%
============================================================
Device IP: 192.168.137.4        Starting Connection...
Hostname: core-switch-3        Connected in 1.36 seconds
Backing up configuration for core-switch-3 (192.168.137.4)...
Backup saved successfully: core-switch-3_20241124_backup.txt
Time Elapsed: 1.36 seconds
Memory Used: Before=21140.60MB, After=21132.84MB
CPU Utilization: Before=1.60%, After=0.40%
============================================================
Device IP: 192.168.137.5        Starting Connection...
Hostname: core-switch-4        Connected in 1.36 seconds
Backing up configuration for core-switch-4 (192.168.137.5)...
Backup saved successfully: core-switch-4_20241124_backup.txt
Time Elapsed: 1.36 seconds
Memory Used: Before=21131.25MB, After=21133.38MB
CPU Utilization: Before=2.00%, After=2.40%
============================================================
```

(a)

https://doi.org/10.24191/jcrinn.v10i1.488

```
============================================================
Device IP: 192.168.137.6        Starting Connection...
Hostname: core-switch-5        Connected in 1.43 seconds
Backing up configuration for core-switch-5 (192.168.137.6)...
Backup saved successfully: core-switch-5_20241124_backup.txt
Time Elapsed: 1.43 seconds
Memory Used: Before=21132.50MB, After=21124.91MB
CPU Utilization: Before=3.90%, After=0.00%
============================================================
Device IP: 192.168.137.7        Starting Connection...
Hostname: core-switch-6        Connected in 1.34 seconds
Backing up configuration for core-switch-6 (192.168.137.7)...
Backup saved successfully: core-switch-6_20241124_backup.txt
Time Elapsed: 1.34 seconds
Memory Used: Before=21124.44MB, After=21125.16MB
CPU Utilization: Before=0.80%, After=1.20%
============================================================
Device IP: 192.168.137.8        Starting Connection...
Hostname: core-switch-7        Connected in 1.35 seconds
Backing up configuration for core-switch-7 (192.168.137.8)...
Backup saved successfully: core-switch-7_20241124_backup.txt
Time Elapsed: 1.35 seconds
Memory Used: Before=21124.54MB, After=21128.25MB
CPU Utilization: Before=0.80%, After=3.50%
============================================================
Device IP: 192.168.137.9        Starting Connection...
Hostname: core-switch-8        Connected in 1.35 seconds
Backing up configuration for core-switch-8 (192.168.137.9)...
Backup saved successfully: core-switch-8_20241124_backup.txt
Time Elapsed: 1.35 seconds
Memory Used: Before=21126.89MB, After=21102.57MB
CPU Utilization: Before=0.40%, After=1.60%
============================================================
```

(b)

```
======================================================================
Device IP: 192.168.137.10       Starting Connection...
Hostname: core-switch-9        Connected in 1.41 seconds
Backing up configuration for core-switch-9 (192.168.137.10)...
Backup saved successfully: core-switch-9_20241124_backup.txt
Time Elapsed: 1.41 seconds
Memory Used: Before=21100.53MB, After=21090.68MB
CPU Utilization: Before=3.10%, After=1.20%
======================================================================
Device IP: 192.168.137.11       Starting Connection...
Hostname: core-switch-10       Connected in 1.35 seconds
Backing up configuration for core-switch-10 (192.168.137.11)...
Backup saved successfully: core-switch-10_20241124_backup.txt
Time Elapsed: 1.35 seconds
Memory Used: Before=21087.38MB, After=21089.30MB
CPU Utilization: Before=0.80%, After=2.70%
======================================================================
                        BACKUP PROCESS COMPLETED
======================================================================
```

(c)

Fig. 5. (a), (b), (c) Console Output After Script Execution

Source: Author's data

https://doi.org/10.24191/jcrinn.v10i1.488

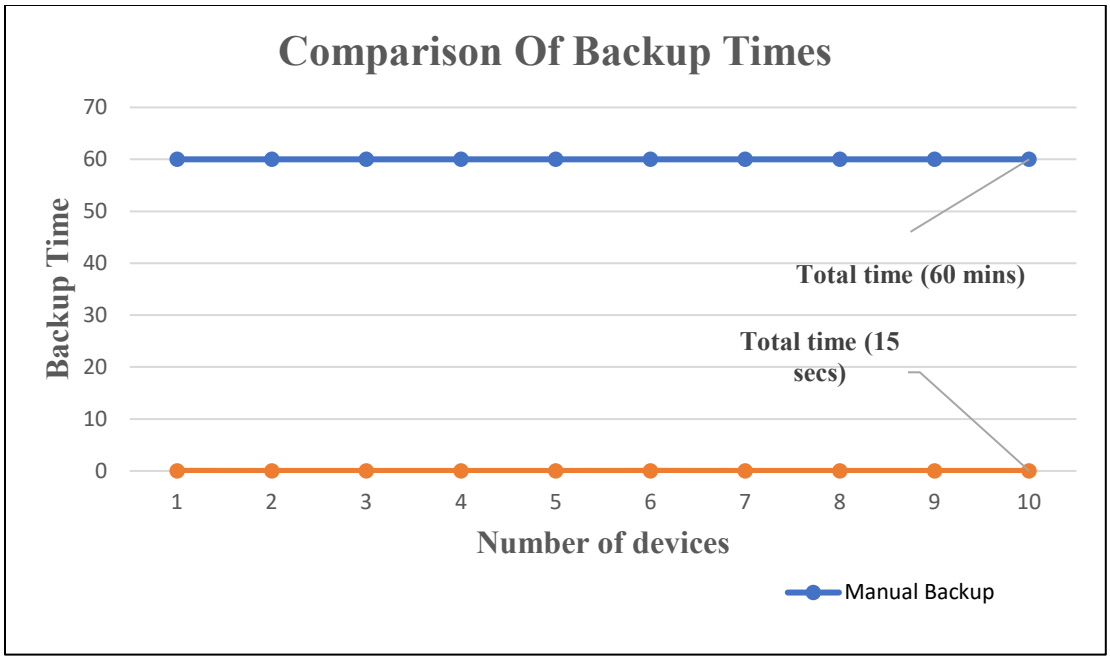

Fig. 6. Comparison of Backup Times: Manual vs. Automated

Source: Author's data

```
Backup saved successfully: backups/core-switch-1_backup_20241117-140011.txt
Backup saved successfully: backups/core-switch-2_backup_20241117-140138.txt

...
```

Fig. 7. Successful Backup Confirmation

Source: Author's own data

Fig. 8 presents a common section of the outputs of the backup files, highlighting key elements such as hostname, interface configurations, and SSH version across all backup files. This is an indication of the accuracy and completeness of the automated backups, making them suitable for disaster recovery and configuration audits.

```
!
hostname core-switch-1
!
interface Vlan1
 ip address 192.168.137.2 255.255.255.0
!
ip ssh version 2
...
```

Fig. 8. Backup file output

Source: Author's data

https://doi.org/10.24191/jcrinn.v10i1.488

4.5 Scalability and Adaptability

The framework is designed for scalability and adaptability. The use of an external *data.txt* file for device IP addresses enables easy modification of the target device list. The modular design of the backup script in Fig. 2 facilitates future enhancements, including support for various device types and integration with scheduling mechanisms. For example, to support different Cisco IOS *show* commands, one would simply modify the *send_command()* calls within the script. This adaptability ensures its long-term viability in evolving network environments.

5. SUMMARY & CONCLUSION

This study addressed the critical need for efficient and reliable configuration backups within the Komfo Anokye Teaching Hospital (KATH) LAN. The conventional backup process as employed by the hospital's network administrators had notable challenges. The manual process was identified to be likely to result in errors, consume a lot of time, and lack centralized management. These limitations posed significant risks to the stability and operational continuity of the KATH LAN. This presents an automated backup framework, which was developed using Python and the Netmiko library as a solution to these aforementioned challenges. The framework provides benefits such as automation of conventional network processes such as backup, elimination of human errors, and implementation of centralized storage for backup consistency. The proposed automated solution utilized the integration of Python with the Netmiko library, and their compatibility with Cisco IOS devices. Experiments conducted on the KATH LAN demonstrated the framework's significant strengths over the existing manual process. Results from experiments conducted in the study show a slightly above 99% reduction in backup completion time; 60 minutes to a maximum of 1.56 seconds per device, a 100% success rate in backup configuration, and an optimized resource utilization. Furthermore, through analysis of the created backup files, the consistency and completeness of the retrieved configurations were confirmed. Finally, it was again revealed through the analysis of the main backup script that the framework was flexible enough to adapt to future infrastructure changes and scalable enough to handle network expansion.

6. ACKNOWLEDGEMENT/FUNDING

The authors received no specific funding for this work.

7. CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest related to this research. This work was conducted independently, and there were no external influences that could compromise the integrity of the study.

8. AUTHORS' CONTRIBUTIONS

Franco Osei-Wusu: Conceptualisation, methodology, formal analysis, writing-original draft; **Wiliam Asiedu**: supervision, writing-review & editing, and validation; **Kwame Asamoah Frimpong**: Conceptualisation, methodology, and software; **Donald Yeboah**: Investigation, visualization and validation; **Elvis Antwi Sarfo**: Software, project administration, and investigation; **Siddique Abubakr Muntaka**: Conceptualisation, supervision, and, writing- review & editing.

9. REFERENCES

Akbar, M. G., Witriyono, H., Apridiyansyah, Y., & Abdullah, D. (2023). Implementation of the inter Tk Package, Sub-Process and OS in the network management application development with python programming language. *Jurnal Komputer, Informasi Dan Teknologi*, *3*(1). https://doi.org/10.53697/jkomitek.v3i1.1210

Akinsanya, M. O., Ekechi, C. C., & Chukwuekem, D. O. (2024). Virtual Private Networks (VPN): A conceptual review of security protocols and their application in modern networks. *Engineering Science & Technology Journal*, *5*(4), 1452–1472. https://doi.org/10.51594/estj.v5i4.1076

Anwar, S. J., & Ahmad, I. (2019). Design and deployment of IPSec VPN using CISCO network infrastructure. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 237–247. https://doi.org/10.32628/cseit195630

Gavathri, C., Patil, V. D., Singh, D. K., Kumar, R., T, N., & Kalra, H. (2023). investigating the benefits of adopting secure shell (SSH) in wireless network security. In *2023 IEEE International Conference on Paradigm Shift in Information Technologies with Innovative Applications in Global Scenario (ICPSITIAGS)* (pp. 309–315). https://doi.org/10.1109/ICPSITIAGS59213.2023.10527461

Consul, J. . I., & Bunakiye, J. . R. (2023). Survey of the influence of routing protocols to network performance enhancement. *Advances in Multidisciplinary and Scientific Research Journal Publication*, *10*(4), 13–28. https://doi.org/10.22624/AIMS/MATHS/V11N4P2

Dewi, S., Firmansyah, F., & Hasan, U. (2022). Penerapan metode access control list pada jaringan VLAN menggunakan router Cisco. *IMTechno: Journal of Industrial Management and Technology*, *3*(1), 37–41. https://doi.org/10.31294/imtechno.v3i1.927

Dong, X., Yu, Y., & Zhou, J. (2023). *Cisco*. Springer Nature Singapore. https://doi.org/10.1007/978-981-19-7870-8

Ehigbochie, D., & Omoze, E. (2024). CISCO vs other networking tools: A comprehensive study on current network simulators and categorizing them based on their performances. *Journal of Computer Sciences and Informatics*, *1*(1), 33. https://doi.org/10.5455/JCSI.20240523125542

Elezi, A., & Karras, D. A. (2023). On detailed network systems configuration management automation using Python. *Wseas Transactions on Communications*, *22*, 1–16. https://doi.org/10.37394/23204.2023.22.1

Ergenç, D., Brülhart, C., & Fischer, M. (2023). *Towards Developing Resilient and Service-oriented Mission-critical Systems*. http://arxiv.org/abs/2304.00128

Farias, W. A. S., Melo, D. M. A., Santos, L. M. dos, de Oliveira, Â. A. S., Medeiros, R. L. B. A., & Silva, Y. K. R. O. (2024). *Web Scraping as a scientific tool for theoretical reference*. https://doi.org/10.21203/rs.3.rs-3854342/v1

Filsfils, C., Bashandy, A., Gredler, H., & Decraene, B. (2019). *IS-IS Extensions for Segment Routing*. https://doi.org/10.17487/RFC8667

Gondhalekar, B., Manna, H., Kothi, S., & Borsae, D. B. (2024). Network automation with multithreading using GNS3 and Netmiko. *Indian Journal of Computer Science*, *9*(2), 18. https://doi.org/10.17010/ijcs/2024/v9/i2/173860

Hassan, G. M., Hussien, N. M., & Mohialden, Y. M. (2023). Python TCP/IP libraries: A review. *International Journal Papier Advance and Scientific Review*, *4*(2), 10–15. https://doi.org/10.47667/ijpasr.v4i2.202

Hunt, J. (2023). *Sockets in Python* (pp. 557–569). https://doi.org/10.1007/978-3-031-40336-1_49

Karki, S. (2021). *Performance Comparison of SSH Libraries*.

Mahmood, A. (2020, September 16). *Performance Analysis of Routing Protocols RIP, EIGRP, OSPF and IGRP using Networks connector*. https://doi.org/10.4108/eai.28-6-2020.2298167

Mazin, A. A., Abidin, H. Z., Mazalan, L., & Mazin, A. M. (2023). Network automation using python programming to interact with multiple third-party network devices. In *2023 10th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)* (pp. 59–64). https://doi.org/10.1109/ICITACEE58587.2023.10277400

Michel, F., & Bonaventure, O. (2023). Towards SSH3: How HTTP/3 improves secure shells. *ArXiv.Org*.

Miller, A., Kobylski, N., Qamar, E., Xiao, J., Veal, N., Kenney, R., Wysocki, N., & Mahmoud, M. (2022). Automating file operations via python. In *2022 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 1907–1913). https://doi.org/10.1109/CSCI58124.2022.00343

https://doi.org/10.24191/jcrinn.v10i1.488

Mutiara, D. A., Isnaini, K. N., & Suhartono, D. (2023). Network programmability for network issue using Paramiko Library. *Jurnal Teknik Informatika (Jutif)*, *4*(4), 751–758. https://doi.org/10.52436/1.jutif.2023.4.4.691

Neeru Kumari, & Dr. Tilak Raj. (2024). OSPF metric convergence and variation analysis during redistribution with routing information protocol. *International Research Journal on Advanced Engineering and Management (IRJAEM)*, *2*(06), 1985–1991. https://doi.org/10.47392/irjaem.2024.0293

Novradinata, R., Riyadi, S., & Adrian, R. (2022). Evaluating the Hybrid Multi-Protocol Label Switching (MPLS) on the Enhanced Interior Gateway Routing Protocol (EIGRP). In *Emerging Information Science and Technology* (Vol. 3, Issue 2).

Pike, T., Colter, R., Bailey, M., Kazil, J., & Meyers, J. S. (2022). *Social Networks as a Collective Intelligence: An Examination of the Python Ecosystem*.

Prosviryakova, L. V., Osipov, K. A., & Dmitriev, A. A. (2024). New opportunities for studying digital information transmission technologies using Cisco equipment. *E3S Web of Conferences*, *548*, 03010. https://doi.org/10.1051/e3sconf/202454803010

Tiwari, M. K., Pal, R., Chauhan, V., Singh, V., Singh, V., Dhamodaran, Dr. S., & Sharma, Dr. S. (2024). A python programming widely utilized in the development of a twitter bot as a sophisticated advance technical tool. *International Journal of Computing and Artificial Intelligence*, *5*(1), 102–108. https://doi.org/10.33545/27076571.2024.v5.i1b.88

Toledo, S. (2023). SSH tunneling to connect to remote computers. *Software Impacts*, *17*. https://doi.org/10.1016/j.simpa.2023.100545

Wachid, N., Majid, A., & Fuada, S. (2020). RIP Vs. OSPF routing protocols: Which one is the best for a real-time computer network? *Jurnal SIMETRIS*, *11*(1).

Wang, X., Liu, Z., Li, Q., Guo, Y., Ling, S., Zhan, J., Xu, Y., Xu, K., & Wu, J. (2023). *Secure Inter-domain Routing and Forwarding via Verifiable Forwarding Commitments*. http://arxiv.org/abs/2309.13271

Xu, J., & Russello, G. (2022). automated security-focused network configuration management: State of the art, challenges, and future directions. In *2022 9th International Conference on Dependable Systems and Their Applications (DSA)* (pp. 409–420). https://doi.org/10.1109/DSA56465.2022.00061

Zadka, M. (2022). Paramiko. In *DevOps in Python* (pp. 139–148). Apress. https://doi.org/10.1007/978-1-4842-7996-0_9

Zhu, R., Wang, X., Liu, C., Xu, Z., Shen, W., Chang, R., & Liu, Y. (2024). ModuleGuard: Understanding and detecting module conflicts in python ecosystem. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (pp. 1–12). https://doi.org/10.1145/3597503.3639221

Zolkin, A. L., Sarycheva, S. A., Tarasova, A. E., Bityutskiy, A. S., & Azarenko, G. Yu. (2023). Development of a system for building computer networks on a Cisco Packet Tracer software emulator. In A. Gibadullin & S. Sadullozoda (Eds.), *2nd International Conference on Computer Applications for Management and Sustainable Development of Production and Industry (CMSD-II-2022)* (p. 46). SPIE. https://doi.org/10.1117/12.2669500

https://doi.org/10.24191/jcrinn.v10i1.488