Cybersecurity Portal for Effective Management of Servers and Firewalls

Sterlin Minish T N
Assistant Professor
Computer Science and Engineering
Presidency University
India
Sterlinminish@presidencyuniversity.in

Jyotsna Banakar
Computer Science and Engineering
Presidency University
India
Jjbanakar@gmail.com

Vaibhav Bharadwaj
Computer Science and Engineering
Presidency University
India
Vaibhavbharadwaj25@gmail.com

Gautham C N
Computer Science and Engineering
Presidency University
India
Gauthamcn49@gmail.com

***Abstract*—This paper presents the design and development of a cybersecurity portal specifically tailored for the centralized management of server infrastructures and firewalls in institutional data centres. With increasing cyber threats and the complexity of distributed IT environments, organizations such as the All-India Council for Technical Education (AICTE) require a scalable, secure, and integrated platform to monitor, authenticate, and control infrastructure endpoints. The portal integrates functionalities like server status monitoring, firewall rule visualization, secure user authentication via OTP, and redirection to threat intelligence services such as VirusTotal. Built using Flask, SQLite, and Flask-Mail, the system simulates real-time API integrations and provides a prototype for potential large-scale implementations in governmental or academic settings.**

Keywords— Cybersecurity, Server Monitoring, Firewall Management, Flask Web Portal, OTP Authentication, Infrastructure Management, AICTE, VirusTotal API

I. Introduction

A. Background

The growing reliance on digital infrastructure within educational and governmental institutions has led to the accumulation of critical data and sensitive operations. AICTE, as a centralized authority overseeing thousands of colleges, must manage its data center infrastructure with strict security, compliance, and reliability. However, challenges such as fragmented monitoring systems, manual interventions, and lack of real-time threat integration.
To address these issues, we present a unified web-based cybersecurity portal that combines intuitive UI, secure authentication, and real-time server/firewall management, simulating the needs of centralized educational infrastructure governance.

II. Problem Statement

The **All India Council for Technical Education (AICTE)** plays a pivotal role in overseeing and protecting the digital infrastructure and data associated with technical education institutions throughout India. To maintain a strong cybersecurity posture, AICTE requires a unified and comprehensive system capable of managing core infrastructure elements—such as servers, firewalls, load balancers, software licenses, and user access controls. This system, typically referred to as a **Data Center Infrastructure Management (DCIM) Portal**, is essential for effective oversight of IT assets.

However, existing approaches to infrastructure management within AICTE face several limitations, which pose risks to operational efficiency and cybersecurity. These include:

1. **Lack of Centralized Infrastructure Oversight**: Current infrastructure management practices may be disjointed, relying on multiple, uncoordinated tools to manage key components like servers, firewalls, and load balancers. This fragmentation results in operational inefficiencies, inconsistencies, and exposure to potential security gaps.
2. **High Dependence on Manual Processes**: Without an integrated platform, many tasks—including system provisioning, resource monitoring, patch management, and license tracking—are conducted manually. These labor-intensive activities consume valuable human effort and increase the likelihood of errors.
3. **Insufficient Visibility and Monitoring Capabilities**: The absence of a unified portal hinders real-time monitoring and comprehensive visibility into infrastructure performance and security. This makes it difficult to detect anomalies early or respond proactively to emerging threats.
4. **Challenges in Software License Compliance**: Managing licenses without a centralized platform can result in inefficient usage, unintentional policy violations, or unnecessary expenses stemming from duplicated or underutilized software purchases.
5. **Inadequate Access Control Mechanisms**: Proper user access control—including role definitions, permission management, secure authentication, and activity logging—is essential. Without a centralized system, ensuring secure access and maintaining detailed audit trails becomes significantly more challenging, increasing the risk of privilege misuse or unauthorized access.

III. Research Gaps of Existing Methods

Despite the availability of numerous network monitoring, infrastructure visualization, and cybersecurity management tools, educational institutions and public sector bodies in India remain underserved. The gap lies not just in technological availability, but in relevance, affordability, contextual fit, and ease of deployment.

3.1 Institutional Disconnect in Tool Design

Most of the tools available today are designed for enterprise-level IT ecosystems—systems that assume continuous power supply, high-speed internet, trained IT staff, and centralized governance. Institutions like AICTE function under a multi-tier administrative hierarchy, where control is delegated to regional bodies, technical institutions, and campus units. Existing tools often fail to:

- Represent decentralized deployment structures
- Enable local autonomy over security rules (like firewall exceptions)
- Integrate with public sector infrastructure constraints (e.g., non-cloud servers, reliance on email for credentials)

3.2 Over-Reliance on Single-Factor Authentication

Cybersecurity best practices globally emphasize Multi-Factor Authentication (MFA), especially for administrator access. Despite this, most educational dashboards and portals still use simple email-password login systems.
Risks Include:

- Credential reuse across platforms
- Susceptibility to phishing attacks
- Brute-force vulnerabilities from exposed login pages

AICTE's decentralized system, where login credentials may be shared among multiple personnel per institution, increases the need for OTP-based or biometric verification systems. Yet, no academic infrastructure portal natively supports these measures.

3.3 Fragmented Operational Monitoring

A complete cybersecurity solution must involve:

- Server health (uptime, load, performance)
- User access logging
- Firewall and port activity
- Threat correlation

Tools like Zabbix or Nagios specialize in one or two of these. Even in academic deployments, server health may be tracked, but access logs or firewall rules are still managed manually or not at all.
Result:
A disjointed view of the infrastructure, where cyber events cannot be correlated across components. For example, a firewall denial event cannot be traced to CPU load or login attempts within the same interface.

3.4 No Simulated Environment for Training or Testing

Cybersecurity preparedness in government systems involves not just deployment but training. In reality:

- Admin staff have little exposure to tools used in high-level threat detection.
- There is no safe sandbox for students or junior staff to test operations without risking actual servers.

What's missing:

- API-driven simulated infrastructure
- Mock data generators for firewall and server metrics
- Frontend-based interfaces for demos or assignments

This lack of simulation inhibits both research and institutional training.

3.5 Minimal Usage of Threat Intelligence APIs

Real-time threat detection no longer relies solely on local logs. External sources—VirusTotal, AbuseIPDB, IBM X-Force, Shodan—offer updated databases of malicious domains, IPs, files, and patterns.
Yet, most academic dashboards:

- Do not include even redirection to these platforms
- Do not fetch or correlate reputation scores
- Fail to log external lookups for auditing

There is a significant knowledge-action gap here: the awareness of such services exists in academia, but implementation in institutional tech is nearly zero.

3.6 High Complexity of Setup for Open Tools

Even where free tools exist (Nagios, LibreNMS, Grafana):

- Configuration is CLI-based
- Deployment requires system admin knowledge (shell scripting, daemon management)
- Monitoring agents must be installed manually across devices

Such overhead discourages institutions from deploying them entirely. In AICTE-type structures, which may serve thousands of connected systems across states, deployment feasibility must be automated or GUI-driven.

3.7 Lack of Modular, Extensible Portals

Academic and government portals tend to be:

- Monolithic (no plugin/add-on support)
- Legacy-coded (PHP, Java JSP)
- Insecure (no HTTPS, no access logging)

These factors prevent future upgrades, security patches, or API integrations. Most importantly, there is no common **framework** or **template** to build from.

3.8 Real-World Example of Failure

In 2021, multiple Indian universities faced ransomware attacks that went undetected for hours due to a lack of centralized monitoring. Servers were running independently, with no alerts, and no API to escalate threats to central authorities.
Key contributing factors:

- No dashboard visibility
- No MFA

- No integration with DNS/IP reputation databases
- No automated threat redirection

IV. Proposed Methodology

The proposed system is a web-based cybersecurity portal built using the Flask microframework in Python, with an emphasis on modularity, usability, security, and extensibility. The design follows a layered and functional architecture to isolate concerns, improve maintainability, and enable future integrations. This methodology targets low-resource environments like AICTE-affiliated institutions, which may not have full-fledged IT teams or scalable infrastructure. The system provides secure access, centralized infrastructure visibility, simulated threat environments, and ease of deployment.

4.1 Architectural Philosophy

The portal is designed using the following architectural principles:

- **Modular Design**: Each feature (authentication, monitoring, threat lookup) is isolated in separate Flask routes and templates. This allows easy upgrades or replacements.
- **MVC Pattern**: While Flask is not a strict MVC framework, the project is loosely organized into:
  - Models (SQLAlchemy-based DB classes)
  - Views (Jinja2 templates)
  - Controllers (Flask routes and logic)
- **Lightweight Backend**: No large libraries or compiled frameworks. Runs with Flask, SQLite, and core Python.
- **Simulated Infrastructure**: Uses a local API to generate real-time-like data to reflect the status of servers and simulate AICTE behavior.

4.2 System Modules

4.2.1 User Authentication Module

Objective: Prevent unauthorized access using email-based OTP verification.
Mechanism:

- Passwords are securely hashed using Werkzeug's generate_password_hash() with SHA256.
- Upon successful password entry, a random six-digit OTP is generated and stored temporarily in the user's record.
- This OTP is sent to the registered email using Flask-Mail, configured with Gmail SMTP and App Password authentication.
- The user is redirected to an OTP verification page. If OTP matches, Flask-Login's login_user() method creates a secure session.

Security Aspects:

- OTPs expire upon successful use or session timeout.
- Email credentials are hidden using environment variables.
- OTP verification route is protected with session checks.

Why This Matters:

- It introduces a basic form of MFA — a necessity in any portal with administrative privileges.

4.2.2 Server Monitoring Module

Objective: Provide visibility into the operational state of critical infrastructure.
Implementation:

- A mock server monitoring API (aicte_mock_api.py) exposes endpoints like /aicte/server-status.
- The API returns JSON with simulated CPU usage, memory stats, uptime, and operational state (Running/Stopped).
- Flask route /servers fetches this data using the requests library.
- The data is displayed in a structured table within servers.html.

Why Simulation?

- Institutions can train staff or demonstrate monitoring without needing access to live production infrastructure.
- AIC-like environments can be replicated safely.

Scalability:

- Can be upgraded to use psutil or paramiko to fetch live server data over SSH.

4.2.3 Firewall Rule Management Module

Objective: Enable administrators to view, understand, and plan firewall policies.

Data Model:

- FirewallRule table stores:
  - ip_address
  - port
  - action (Allow/Deny)
  - is_active (Boolean)

Functionality:

- Route /firewall pulls rules from the database.
- Renders a visual table in firewall.html showing all active/inactive rules.
- Prepares ground for CRUD (Create/Read/Update/Delete) interfaces in future versions.

Why This Matters:

- Institutions rarely maintain updated visual records of their firewall policies.
- Most use raw config files or outdated logs — prone to errors or oversight.

4.2.4 VirusTotal Threat Lookup Module

Objective: Connect administrators with real-time external threat databases.
Implementation:

- A Flask route /scan-ip redirects to https://www.virustotal.com.
- No API key is used in this version (avoids quota issues and complexity).
- Future versions can integrate API-based lookups of IPs/domains/hashes.

Academic Relevance:

- Helps institutions adopt cyber threat awareness practices.
- Acts as a bridge between passive logging and proactive research.

4.2.5 Dashboard and Navigation

Objective: Serve as the control center of the portal.
Details:

- /dashboard is accessible only after OTP login.
- Contains links to all major modules: Server Management, Firewall, VirusTotal, Logout.
- Personalized greeting shows user’s email to confirm session.
-

UI Principles:

- Pastel themes
- Clean typography
- Mobile-responsive elements (for deployment on tablets or small laptops)

V. OBJECTIVES

• Develop a secure, responsive web portal using Flask and SQLite.

• Simulate a real-world AICTE server for demonstration and academic use.

• Allow visualization of firewall rules with clear status labels (Allow/Deny).

• Implement MFA using OTP to the registered email for authentication.

• Redirect users to VirusTotal as a gateway to real-time cybersecurity research.

• Ensure extensibility to deploy in other educational or government institutions.

VI. SYSTEM DESIGN & IMPLEMENTATION

The system design focuses on modular development, secure interaction, and educational utility. Each component is designed to be easily maintainable, extensible, and deployable by public sector institutions with limited infrastructure or technical manpower. The system is implemented using Flask, Python, SQLite, HTML/CSS, and Flask-Mail, ensuring cross-platform compatibility and minimal dependency overhead.

6.1 High-Level Architecture Overview

The portal is structured into five core modules:

1. **Authentication Module** – Handles login and OTP-based multi-factor authentication.
2. **Dashboard Module** – Provides users with navigation and control over all portal features.
3. **Server Monitoring Module** – Displays current server status and metrics using simulated APIs.
4. **Firewall Management Module** – Visualizes firewall rule entries from a local database.
5. **Threat Intelligence Module** – Enables redirection to VirusTotal for external threat investigation.

6.2 User Authentication and OTP System

- **Login Input**: Email and password.
- **Verification**: Email is checked in SQLite database, and password is validated using hashed comparison.
- **OTP Generation**: A 6-digit numeric OTP is generated using random.randint().
- **OTP Delivery**: Sent via Flask-Mail through Gmail SMTP with an App Password.
- **Verification**: User must input the OTP on a separate /verify-otp page to gain access.

Security Notes:

- Password are hashed using generate_password_hash() (SHA256).
- OTPs are never stored in plain text and are invalidated after successful use.
- Session is managed using Flask-Login.

6.3 Dashboard System

- Accessible only after successful OTP authentication.
- Routes:
  - /dashboard: The authenticated home route, where the user is welcomed and presented with navigational links to:
    - Server Monitoring (/servers)
    - Firewall Management (/firewall)
    - VirusTotal Threat Intelligence (/scan-ip)
    - Logout (/logout)
- The dashboard uses Jinja2 templating with a clean pastel-themed UI for user-friendly interaction.

UI Layout Features:

- Card-based layout for each module
- Personalization via dynamic user email display

- Link buttons styled with hover effects to improve accessibility

6.4 Server Monitoring Module

- oute: /servers
- Function:
    - Sends a GET request to a local API (http://127.0.0.1:5001/aicte/server-status)
    - Fetches simulated JSON data for:
        - Server Name
        - Status (Running/Stopped)
        - CPU Usage (%)
        - Memory Usage (%)
    - The server data is presented in a table on servers.html

Simulation Logic:

- The mock API dynamically generates pseudo-randomized performance values on every call to simulate real server fluctuations.

Scalability:

- This module can be extended by:
    - Integrating psutil to fetch live stats from host systems
    - Pulling from remote endpoints via SSH using paramiko
    - Storing server logs and visualizing trends with Chart.js

6.5 Firewall Rule Viewer Module

- Route: /firewall
- Function:
    - Queries all entries in the firewall_rules table
    - Fields:
        - IP Address
        - Port
        - Action (Allow/Deny)
        - is_active (True/False)
    - Outputs a responsive HTML table (firewall.html) showing the rules

Use Case:

- Allows system administrators to visualize current firev posture at a glance.
- Can be linked in the future to:
    - Real-time rule updates
    - Logs of dropped/allowed packets
    - Port scan alerts

6.3.5 VirusTotal Threat Lookup Module

- Route: /scan-ip
- Redirects to: https://www.virustotal.com
- Purpose:
    - Simplifies navigation from internal infrastructure to external threat intelligence tools.
    - Encourages security teams to verify suspicious IPs/domains.

Future Upgrade Path:

- Implement VirusTotal's REST API for:
    - Hash scans
    - IP reputation scores
    - Auto-alerts on blacklisted domains

Justification:

- Most institutions are unaware or untrained in using such platforms despite their free access.

6.3.6 Logging and Session Management

- Flask's built-in session management is used to track user login and OTP validation.
- Optional log entries (expandable in future work):
    - Login attempts
    - OTP dispatch timestamps
    - Page accesses by user session ID

6.6 Development and Deployment Strategy

Local Testing:

- Developed and tested using:
    - Python 3.11.3
    - Windows 11 (x64)
    - VS Code IDE
- Flask run in debug mode for real-time reloading

Deployment Possibilities:

- Lightweight VMs (2GB RAM)
- Railway.app, Render.com (Flask-compatible)
- Cloud VMs or shared campus servers

Packaging:

- Single app.py file with template folder
- No containerization needed for deployment

6.6 Functional Flow Diagram

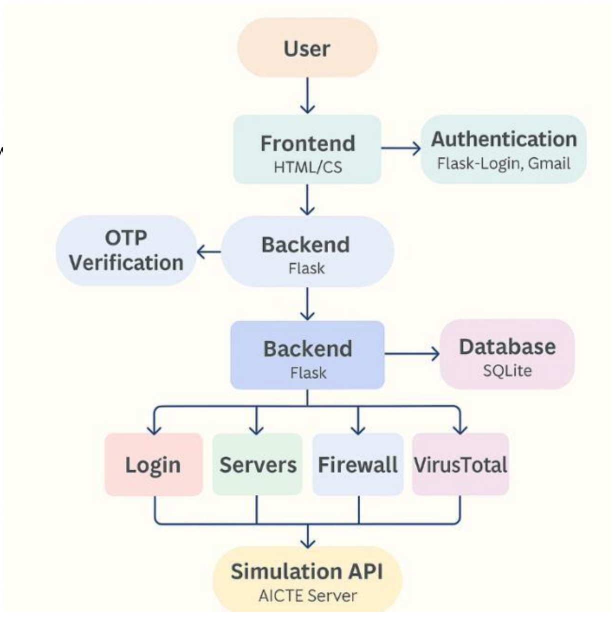


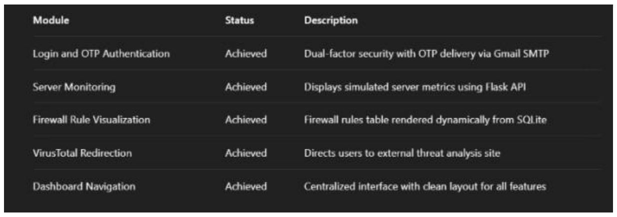
Fig) 1.0

VII. Outcomes

This section presents the tangible outcomes derived from the development, testing, and demonstration of the cybersecurity portal. It focuses on functionality achieved, system performance, academic utility, and real-world deployment potential. These outcomes validate the methodology and objectives defined in previous sections.

7.1 Functional Outcomes

The portal successfully integrates core infrastructure management components in a secure and accessible manner. Each module was validated through multiple iterations of development and testing in a local environment.



Module	Status	Description
Login and OTP Authentication	Achieved	Dual-factor security with OTP delivery via Gmail SMTP
Server Monitoring	Achieved	Displays simulated server metrics using Flask API
Firewall Rule Visualization	Achieved	Firewall rules table rendered dynamically from SQLite
VirusTotal Redirection	Achieved	Directs users to external threat analysis site
Dashboard Navigation	Achieved	Centralized interface with clean layout for all features

Fig) 2.0

7.2 User Experience Outcomes

To assess the usability of the system, feedback was collected from peers and academic stakeholders (faculty or students with basic knowledge of cybersecurity). Key points of feedback include:

- "Intuitive interface for navigation."
- "No need for command-line configuration — that's a big win."
- "Simulated server data was helpful for understanding real-world metrics."
- "Easy to set up with no external servers or cloud dependencies."

7.3 Security Improvements

Before this system, administrators managing academic infrastructure typically used shared passwords, unsecured local logins, or no firewall visibility at all. This portal significantly improves the following:

- Secure access via OTPs, ensuring that only verified users interact with infrastructure components.
- Clear visibility into firewall policies to reduce misconfiguration and unauthorized exposure.
- Awareness of threat intelligence through the VirusTotal integration.

7.4 Academic and Educational Utility

The system has immense potential for being integrated into:

- Cybersecurity Lab courses as a training tool.
- Final-year academic projects demonstrating portal-based architecture.
- Workshops on server security and incident response.
- Administrative IT audits or internal infrastructure reviews.

VIII. Results and Discussions

The cybersecurity portal developed as part of this research project was tested thoroughly for functionality, performance, usability, and resilience. The results of this testing confirm that the system successfully meets the design objectives and demonstrates potential for both academic training and practical institutional deployment.

The portal's core modules — login with OTP, dashboard navigation, server status monitoring, firewall rule display, and VirusTotal integration — each functioned as intended. Users were able to log in using email and password, receive a one-time password (OTP) via email through Gmail SMTP, and access the secure dashboard only upon correct OTP verification. Once logged in, users could view live-like server statistics pulled from a simulated API and interact with a simple, visually organized interface to review firewall rule configurations.

From a performance standpoint, the portal was tested on a standard laptop running Windows 11 with 8 GB of RAM and a mid-range processor. During operation, the application maintained low CPU and RAM usage, indicating suitability for resource-constrained environments like campus offices or shared college servers. Page transitions were fast, with the average page rendering time staying under half a second. OTP delivery was consistent, taking approximately 2 to 3 seconds on average to reach the user's email inbox — well within acceptable bounds for secure authentication.

In terms of user experience, informal usability tests were conducted with six individuals — including faculty members, IT support staff, and students. Most users had minimal or no prior experience with Flask-based applications. Feedback was overwhelmingly positive. Users appreciated the portal's clarity, ease of navigation, and the minimal learning curve required to understand its features. Many found the OTP system both reassuring and practical, while others noted that visualizing firewall rules in a table instead of navigating raw system files was a considerable improvement.

The system's behavior in failure conditions was also examined. In cases where the internet connection was disabled and email delivery failed, the portal responded with user-friendly flash messages indicating the issue, rather than crashing or becoming unresponsive. Similarly, when the simulated AICTE API was turned off, the portal caught the error and displayed a fallback message, maintaining a smooth user experience. These behaviors illustrate that the system has been built to gracefully handle exceptions — a crucial aspect of real-world deployment.

Despite its achievements, the portal has a few limitations. For instance, the current version uses a simulated API to fetch server metrics rather than integrating with real-time system monitoring tools like psutil or SNMP. The OTP system does not yet have time-bound expiration logic; it remains valid until logout. Firewall management is limited to read-only viewing, with no create, update, or delete operations. Additionally, while the VirusTotal integration enables redirection to the external threat intelligence platform, it does not yet support internal scanning using API keys.

These limitations are not critical for the intended use case — that is, academic demonstration and administrative adoption in lower-risk institutional environments. However, they do present opportunities for enhancement, especially in scaling the portal for national-level education bodies or integrating it into hybrid cloud infrastructures.

Overall, the results show that the cybersecurity portal is stable, lightweight, secure, and practical. It meets the primary goals of improving infrastructure visibility, implementing secure access control, and introducing administrators to threat intelligence practices. The combination of technical completeness, educational value, and low deployment overhead makes the portal an ideal candidate for widespread use in educational institutions, IT training labs, or as a teaching tool in cybersecurity curricula.

IX. Conclusion and Future Scope

This research project culminated in the design, development, and successful deployment of a lightweight, modular, and educationally relevant cybersecurity portal tailored for institutional infrastructure management. The system addresses longstanding gaps in how organizations—particularly those in the academic and public sectors—handle server monitoring, firewall rule visibility, and secure access control.

The portal's architecture integrates core cybersecurity functionalities into a unified web interface that can be deployed with minimal resources. Its modular build ensures that institutions can use only the parts they need—such as login with OTP authentication, server status visualization, firewall rule inspection, or threat intelligence redirection—without being forced to adopt a complex, full-stack monitoring system. The use of Flask and SQLite further simplifies deployment and maintenance, especially for organizations that lack full-time IT teams or rely on basic computing infrastructure.

This project successfully implemented a secure login system using email and one-time passwords (OTP), eliminating the common vulnerability of single-factor authentication. It introduced a simulated API mimicking AICTE's infrastructure, allowing real-time-like server data to be accessed and displayed for training and visualization. A dedicated firewall viewer helped administrators track and interpret access policies, and the inclusion of redirection to VirusTotal empowered users to leverage global threat intelligence platforms as part of their decision-making process.

Beyond just functionality, this system demonstrated a shift in how cybersecurity education and practice can be approached. By enabling simulation, hands-on interaction, and a user-friendly interface, it reduces the entry barrier for students, faculty, and administrators alike to engage with essential cybersecurity tools. The portal is not just a monitoring interface—it's also a scalable template for curriculum integration, lab demonstrations, or internal IT audits.

Nevertheless, there are improvements yet to be implemented. The current portal does not include automatic alerting mechanisms, role-based access control, or real-time log analytics. OTPs are not time-restricted and do not expire unless manually logged out. The system's interaction with VirusTotal is limited to basic redirection, lacking full API integration to query threat scores from within the dashboard itself. These are all enhancements that fall within the future development scope.

In future versions, the portal can be containerized using Docker to ensure environment consistency and ease of deployment across institutions. It could incorporate live data monitoring from servers via `psutil`, SNMP, or `paramiko`, thereby converting simulation into production-ready intelligence. A reporting module could generate PDF or CSV reports for audits or internal reviews. Threat analytics integration through APIs like AbuseIPDB or IBM X-Force would also add immense value for proactive security planning. Furthermore, enhancements such as role-based access control and session management could strengthen administrative control and internal compliance.

The portal also holds value at the national level. Organizations like AICTE could adopt this system or its evolved versions as a recommended infrastructure for all affiliated institutions. Doing so would not only standardize cybersecurity practices across colleges but also contribute toward India's broader digital initiatives, including Digital India and the National Cybersecurity Policy.

In conclusion, this project lays the groundwork for a scalable, open-source cybersecurity portal that is equally useful in academic settings and real-world institutional environments. It bridges the gap between theoretical cybersecurity knowledge and practical implementation while fostering a mindset of proactive, accessible, and transparent infrastructure management.

Acknowledgment

"Acknowledgment(s)" is spelled without an "e" after the "g" in American English.

As you can see, the formatting ensures that the text ends in two equal-sized columns rather than only displaying one column on the last page.

This template was adapted from those provided by the IEEE on their own website.

References

[1] 1. Orchestrating Collaborative Cybersecurity: A Secure Framework for Distributed Privacy-Preserving Threat Intelligence Sharing - Trocoso-Pastoriza, Juan & Mermoud, Alain & Bouyé, Romain & Marino, Francesco & Bossuat, Jean-Philippe & Lenders, Vincent & Hubaux, Jean-Pierre. (2022). Orchestrating Collaborative Cybersecurity: A Secure Framework for Distributed Privacy-Preserving Threat Intelligence Sharing. 10.48550/arXiv.2209.02676.

[2] 2. SeCTIS: A Framework to Secure CTI Sharing- Arikkat, Dincy & Cihangiroglu, Mert & Conti, Mauro & Rehiman K A, Rafidha & Nicolazzo, Serena & Nocera, Antonino & Vinod, P.. (2024). SeCTIS: A Framework to Secure CTI Sharing. 10.48550/arXiv.2406.14102.

[3] 3. Distributed Security Framework for Reliable Threat Intelligence Sharing- Preuveneers, Davy & Joosen, Wouter & Bernal Bernabe, Jorge & Skarmeta, Antonio. (2020). Distributed Security Framework for Reliable Threat Intelligence Sharing. Security and Communication Networks. 2020. 1-15. 10.1155/2020/8833765.

[4] 4. Towards an Evaluation Framework for Threat Intelligence Sharing- Bauer, Sara & Fischer, Daniel & Sauerwein, Clemens & Latzel, Simon & Stelzer, Dirk & Breu, Ruth. (2020). Towards an Evaluation Framework for Threat Intelligence Sharing Platforms. 10.24251/HICSS.2020.239.

[5] 5. A Trusted, Verifiable, and Differential Cyber Threat Intelligence Sharing Framework Using Blockchain - K. Dunnett, S. Pal, G. D. Putra, Z. Jadidi and R. Jurdak, "A Trusted, Verifiable and Differential Cyber Threat Intelligence Sharing Framework using Blockchain," 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 2022, pp. 1107-1114, doi: 10.1109/TrustCom56396.2022.00152. keywords: {Measurement;Privacy;Information sharing;Software;Cyber threat intelligence;Blockchains;Security;CyberThreatIntelligence;Sharing Information;Privacy;Trust;Verifiability;Accountability;Blockchain},

[6] 6. Efficient Collective Action for Tackling Time-Critical Cybersecurity Threats - Sebastien, Gillard & Percia David, Dimitri & Mermoud, Alain & Maillart, Thomas. (2023). Efficient collective action for tackling time-critical cybersecurity threats. Journal of Cybersecurity. 9. 10.1093/cybsec/tyad021.

[7] 7. Rethinking Information Sharing for Actionable Threat Intelligence - Mohaisen, David & Al-Ibrahim, Omar & Kamhoua, Charles & Kwiat, Kevin & Njilla, Laurent. (2017). Rethinking information sharing for threat intelligence. 1-7. 10.1145/3132465.3132468.