Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Real-Time Cyber Security Dashboard for Advanced Threat Monitoring and Analysis

Milton Tomas Pedro da Cruz and Prateek Kumar Srivastava
School Of Computing Science And Engineering Galgotias University, Greater Noida, India

Abstract - **Our project addresses the increased complexities and occurrence of cyber threats through the creation of a real-time, web-based dashboard that would assist security personnel to identify, track, and assess attacks in real-time. On the front end, we inject React, Fast API on the back end, and MongoDB to give the threat intelligence a clean and user-friendly interface offering an interface to the analysts. All the alerts can be classified by severity, stream live events, interactive charts and alarms are all handled within the dashboard. We categorize the threats into four levels namely, critical, high, medium, and low in order to enable the teams to prioritize their responses. The dashboard proves to be responsive as our performance tests indicate that the dashboard is capable of managing high-frequency threat events. Its execution indicates how data-visualization libraries could be incorporated with real-time event processing algorithms to deliver actionable information to the SOC personnel. The work is an advancement of situational awareness instruments of cyber-security through integration of contemporary web development competencies with vital surveillance demands. Index Terms Cybersecurity, Threat Monitoring, Security Dash- board, Real-time Analytics, Web Application Security, Threat Intelligence, Security Operations Center, React, FastAPI.**

I. INTRODUCTION

A. Background and Motivation

We have been observing that the process of digitalizing businesses has contributed to the vastness of a target that cyber threats can strike. According to reports released in recent years, on average companies experience approximately 270 cyber attacks annually and the cost of cybercrime around the world is predicted to reach approximately 10.5 trillion dollars by 2025. [1] The traditional security monitoring systems are usually confusing in terms of interface, sluggishness in visualization of threat, as well as real-time response, and thus are difficult to manage the incidents in real-time.

B. Problem Statement

The real truth is that I have been researching, how the existing cybersecurity monitoring solutions simply do not work in the real world. In the majority of works devoted to the subject, there is this detail that the increasing number of security alerts is in general overpowering analysts and hiding really essential threats, and it is really true. We are talking about some severe gaps. Information Overload Information: Security teams receive a torrent of alerts and it is almost impossible to actually determine which ones to bother with. Delayed Response The traditional systems do not have real time view and therefore the threats remain hidden unnoticed as long as possible and the reaction remains trapped in the past. complicated User Interfaces: The majority of enterprise tools are essentially a high Cost of Admission, which actually kills the productivity and makes the entire process stagnant. Lack of Contextual Information:Single standalone alert messages lack any historic data on the alert (context) or trending, making it difficult to determine the real risk. Scalability Problems: The old back-end systems are unable to keep up with the massive speed and rate of attacks in the modern world, they are literally suffocating on the information. All this indicates that, we require monitoring tools that are not only capable of keeping pace in real-time and giving good context, but also ensuring that visuals are easily consumed such that a responder does not get lost in the maze

C. Objectives and Contributions

This paper includes a web-based, modern-day cybersecurity dashboard, which addresses the challenges in the following ways. Real-Time Threat Monitoring: The adoption of a live feed of threats where they are updated within seconds of critical security events. Intelligent Severity Classification: There are four-tier severity model (critical, high, medium, low) on prioritized threat response. Interactive Visual Analytics: (D)ynamic charts and graphs which offer temporal threat analysis and trends identification.Alert Management Their state: Alert tracking (status of active, under investigation, closed). Recent Technology Stack: Utilizing the current web-based technologies towards performance, scalability and maintainability. Interactive Design: Cross-ease of access that makes devices monitor.

D. Paper Organization

The rest of this paper is structured as follows: Section II will perform a literature review on cybersecurity dashboard and threat monitoring systems. Section 3 illustrates system architecture and principles of the design. Section IV provides an outline of the implementation plan and technological decisions. Section V talks about the system features and capabilities. The evaluation and analysis is in section VI. Section VII sums up the paper and provides research directions in the future.

II. RELATED WORK

A. Cybersecurity Dashboard Systems

The field of cybersecurity visualization has been the topic of active research. Good all. essentially hit the ground by

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

addressing the issue of displaying real-time data on matters concerning network security checks [2]. Their results indicate that an obvious sight can reduce the time that you require to identify danger. Commercial SIEM systems also abound, such as Splunk [3], IBM QRadar [4], AlienVault [5], to mention a few, and they provide good monitoring capabilities. However, they are also a burden to install, and tend to be very expensive, as well as the learning curve with smaller teams is normally a steep one. Nevertheless, have a look at Shiravi. They have demonstrated that lightweight, purpose-built dashboards can, in fact, be equally fast at certain kinds of jobs related to monitoring [6].

B. Threat Intelligence Platforms.

These threat intel were purely simple log aggregators, but this time they are very sophisticated analytical tools. The METRE ATTACK is simply the ICT standard of choice anyway to categorize what is being done by the attackers [7]. Getting it in dashboards makes it less difficult to comprehend what happens to these threats, in fact. The crew and Wagner explored the combination of threat feeds and real-time monitoring setups, but they demonstrated that this combination does, in fact, increase the number of detections [8]. Threat intel is becoming more accessible with open-source tools, such as MISP [9] (the Malware Information Sharing Platform), OpenCTI [10], and so on. However, the thing is that actually, they are mostly connected with the informational sharing, but not with monitoring in the working process, thus there is this vacuum that can be filled by custom dashboards

C. Web Technologies in Security Applications

An apparent trend in the construction of security surveillance systems is the use of current web technologies in cybersecurity tools. I have learned that studies of web-based security dashboards have found that reactive one-page web apps are significantly capable of enhancing user experience and efficiency of the analysts over traditional multipage systems. This supports the demand. I found additional research on security data visualization, which show how hierarchical and time-based visualizations allow simplified understanding of complicated security data. Methods such as time-series, heat maps, and hierarchical displays can be used to improve situational awareness, which is why they are suitable for cybersecurity dashboards. Last but not least, recent publications on visual analytics as applied to Cybersecurity have acknowledged that interactive exploration is critical to facilitating the analytical finding of patterns, anomalies, and relationships within massive security databanks [11]. This supports the significance of the addition of interactive visualization to the contemporary security dashboard, which is what the proposed system is concentrating on regarding its design and adoption cite 14. In using a modern frontend framework to perform real-time security visualization.I found additional research on security data visualization, which shows how hierarchical and time-based visualizations allow simplified understanding of complicated security data. Methods such as time-series, heat maps, and hierarchical displays can be used to improve situational awareness, which is why they are suitable for cybersecurity dashboards. Last though not least, recent publications on visual analytics as applied to cybersecurity have acknowledged that interactive exploration is critical to facilitating the analytical finding of patterns, anomalies, and relationships within massive security databanks. This supports the significance of the addition of interactive visualization to the contemporary security dashboard, which is what the proposed system is concentrating on regarding its design and adoption.

D. Real-Time Data Processing

The issue of responding to high-velocity streams of security events is rather wild, and during cybersecurity courses, instructors discuss the various patterns of architectures. We have observed that event-driven architectures are, literally, super good when processing real-time flows of security data since they are scalable and responsive. This is the reason why stream-processing systems like Apache Kafka [12] and Apache Flink [13] are popular in security solutions in enterprises. Yet the study indicates that the practicality of distributed stream-processing systems can in fact be overruled by their extreme complexities of operation when it comes to implementing them in a mid-sized package. These structures tend to translate to a considerable amount of additional infrastructure, the bane of responsibility in configuration, and a lot of headaches in upkeep. Thus, this paper considers a middle way that delivers a contemporary web solution to achieve a close-to-real-time monitoring without the complexity of large-scale distributed stream-processing systems.

E. Research Gap

Although we discover that the existing study and commercial applications provide robust security monitoring solutions, the number of available, lightweight, and up-to-date dashboard applications is evidently lacking. It happens that most security monitoring systems are more focused on the feature richness rather than the usability of their systems, which leads to difficult-to-use and maintain systems, which can be a concern for small organizations. This disconnect can be seen in the fact that scanty dashboards exist. Modernize web development and make the web work better. Have easy-to-use user interfaces with minimum training. Provide real-time monitoring without the need for complicated infrastructure. retain extensiveness and customization features. Support implementation at different levels of organizations. This gap is directly addressed by our work, where we introduce a cybersecurity dashboard that is a combination of modern web technologies and real security monitoring needs based on satisfying performance, usability, and scalability.

III. SYSTEM ARCHITECTURE

A. Architectural Overview.

Consequently, the cybersecurity dashboard we will be discussing is developed in a three-tier structure, which includes the presentation layer, the application layer, and the data layer. Separating things it makes it more modular, easier to scale, and simpler to maintain. In essence, all people are aware that layered architecture ensures that components are

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

loosely coupled and the system progresses more easily since the responsibilities are separated. We have also adhered to principles of RESTful API and reactive programming, which enables data flow easily and also allow us to send real-time notifications. These sorts of design decisions are very effective with regard to security monitoring, where you have to remain responsive and scale up easily.

B. Technology Stack Selection

The technology stack that I finally chose was partly informed by the following factors: performance, the developer ecosystem, security capabilities, scalability, and community support. Frontend Layer: React 18.3.1 In it is the primary UI framework. The magic of a virtual-DOM and the setup provided by components provides us with blistering performance on all those security dashboards that we require. TypeScript version 5.8.3: includes static typing to make the code dependable one and reduce the number of ugly bugs at a run time, which is essential to anything that is security intensive. Tailwind CSS 3.4.17: Utility-first, design that allows us to create the UI hyper-fast and at the same time be consistent across the board [15]. ShadCN UI Components: They are based on Radix UI primitives and are customizable, accessible, and adhere to the WAI-ARIA standard because we are not playing with compliance. Recharts 2.15.4: Provides the data-visualization layer, and, as such, provides interactive charts, which are helpful with security measurements. React Query (TanStack Query) 5.83.0: Does state management on the server, provides us with caching, background updates, and those optimistic UI effects that would make the application feel faster.

C. Component Architecture

1) Frontend Components: The front-end architecture is built on the principle of atomic design, which is designed with a hierarchy of components as follows: Pages: The feature-specific functionality is provided in containers (Route-level components, including, but not limited to: Dashboard, Threats, Alerts, Reports, and Settings). Composite Components: Special-purpose components such as ThreatChart, ThreatFeed, AlertsTable, and MetricCard are features of a more complex nature. UI Primitives: ShadCN UI provides reusable (Button, Card, Table, Badge) components, which are a consistent, styled, reusable experience. Hooks: Custom React hooks deal with cross-cutting points such as mobile responsiveness and toast notification. 2) Backend components, actually, a layered architecture is applied in the backend: API Layer: Endpoints that are prefixed with /api process whatever is received by and sent via HTTP. Service Layer Business logic will be placed here to modify information related to threat processing, tiering, and alerting. Data Access Layer: This layer is the operations of MongoDB abstracted with the interface of Motor. Models: Pydantic models rule out data schema definition of status checks, threats, and alerts

D. Security Considerations .

The security measures that are part of the architecture are: CORS Setup: We have a controlled cross-origin resource sharing, and thus, there is nobody who can creep in. Input Validation: Pydantic models do not allow any type of data to be injected into them. Environment Variables: Environment variables contain sensitive config, and it is not hard-coded. UUID Implementation: Objectids Prerequisites: This implementation list removes information leaks in sequential Objectids. Async Operations: Non-blocking I/O ensures that the system is immune to denial -of-service floods

E. Data Flow

The system involves a one-directional flow of data: First of all, we feed security events using our endpoints to API backends. Then we authenticate such events and categorise them using severity. Events that are validated are stored in MongoDB, and each document has a timestamp and other metadata added. On the front, we can take updates by polling or subscribing to the endpoint points on the backend. React Query is used to manage the data caching and refresh the data. In case of changing the state, the components automatically re-render. And the last one, API calls are invoked with every user interaction, thereby repeating the same cycle

F. Scalability Design

The architecture incorporates horizontal scale by Stateless Backend: FastAPI servers do not maintain any session state and therefore, we can freely have as many load-balanced instances without troubling. Indexing Data in Database MongoDB indexes on the fields of timestamp and the level of severity blazingly appreciation the speed of queries, therefore our students are obliged to obtain data in MongoDB blazingly fast. Caching Strategy: React Query has intelligent caching, which reduces unnecessary API calls, thus saving bandwidth, and also, the UI is fast. Lazy Loading: Modular components and routes are loaded on-demand, the starting bundle is slick, and the application is light as a feather.

IV. IMPLEMENTATION

A. Development Environment Setup

The producing environment employs the contemporary tooling to. best incorporation developer experience: Vite 5.4.19: Are we using instant to build this tool? hot module replacement (HMR), production build optimization and server start. Uvicorn 0.25.0: executes the ASGI server of FastAPI, to support WebSocket connections in case of real-time. event streaming. Python 3.10 or later Python uses modern asynchronous functionality and type hints to have a good backend.

B. Core Features Implementation

1) Real-Time Dashboard: The primary dashboard (Index.tsx) equips all the key security indicators and presents them using the following components of MetricCards: Total Threats: this is the total number of threats that we have

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

discovered including a percentage-change measure. Active Alerts: active alerts which must be addressed immediately providing the difference between the periods. Blocked Attacks: these are those attacks that we have been able to block and this is an indicator that the system is working. System Health: the general security posture score derived as a result of a number of measures. Each metric card shows: Value in an understandable format. A trend arrow (up or down) A context-fit icon (checkmark, shield, activity, alert). Color coded in terms of severity.

2) Classifying the Threat Severity: The system critically deems the severity under the four-tier severity model: Critical: urgent dangers which require quick action (do think live SQL injection or DDoS attacks). These we indicate by red (hsl(var(-critical)). High: severe threats, which must be analyzed as soon as possible (e.g., brute force attempts or suspicious logins). These go orange (hsl(var(-high))). Medium: possible security threats that should be monitored (e.g. port scanning, failed logins attempts). We make these yellow (hsl(var(-medium)). Low: minor anomalies only to be aware of them (e.g. small config adjustments). These are green (hsl(var(-low))). Such classification is in accord with industry norms and gives the workload of the analysts a priority, automatically.

3) Threat Activity Visualization: ThreatChart component provides a 7-day trends of threats in a stacked bar chart: Data Structure: This is the count of all four levels of severity per day, thus you can identify patterns. Interactive Elements: The Tooltip of Recharts provides information on hovering, and the CartesianGrid assists in reading the data. Responsive Design: ResponsiveContainer makes the chart fit inside the screen, and thus it can be used on any device. Color Consistency: the severity colors remain constant between components to ensure that the analysts do not get confused. This chart allows identifying such trends as the spikes of the attacks during weekends and weekdays, increases or decreases of the threat levels, the relations between the various severities, and the effectiveness of the countermeasures.

C. *Backend API Implementation*

The FastAPI back-end provides us with a bunch of RESTful endpoints that we can easily make requests to on our end or the very front-end site. 1) Core Endpoints GET/api/- health check- a rapid examina- tion to determine whether the server is alive. POST/api/status- create a new status check money. GET /api/status - fetch down all of the entries of the status checks stored.

2) Database Integration We have Motor, which is the asynchronous MongoDB driver, thus the DB calls are not blocking and are fast. AsyncIOMotorClient on Motor does maintain a pool of persistent connections so that we are not required to pound on the database with each request. The IDs (as well) are all UUIDs, which makes the data JSON- friendly and they do not duplicate. The automatic validation and serialization provided by pydantic models is as follows:StatusCheckCreate System- The exchange schema against which incoming data shall be checked. StatusCheck- the response model that incorporates created fields such as the ID or date.

3) CORS and Security Implementation. The CORS middle- ware is configured such that allowed origins that are environ- mentally controlled can be added. Auth credentials We make it possible to send cookies or auth headers on authenticated requests. When we are developing we will be loose in allowing wildcard methods, wildcard header as long as we can, but when production comes around we tighten policies to adhere to the security best practice.

D. *Routing and Navigation*

The application has a hybrid approach to state-management: State of server: React Query ensures that all data of the server is correct. It deal with automatic caching with a TTL that is customizable, background refetching so that things are always fresh, optimistic updating to make it seem fast, as well as, query invalidation to ensure all the data is consistent. UI State: The regular React state (useState, useReducer) is used for whatever stuff which is component local and relatively unimportant than, well, I don't know, you can use our next level of state instead. Global Context The react context API trades global issues like theme preferences and user settings.

E. *Performance Optimization*

They make these optimization tricks make the app tight and fast to users! Code Splitting Asynchronous Code: We lazy load routes such that the first bundle will always be small and that pages only start popping up when you are actually going there. Memoization: Several of the example uses of React have useMemo and useCallback, which stops unnecessary recalcu- lations and re-rerendering, which is actually a way of making things fast. Debouncing: Search and filter inputs are debounced in which you will not end up spamming API calls when you hit a key.

Virtual Scrolling: Big lists are made virtual to only display what you have to see and this is a massive performance improve on large data sets. Optimization of the Build: This is pulled together by the rollup setup of Vite: - Tree shaking in getting rid of unnec- essary code - Minification in order to reduce file sizes. Code splitting in order to cache better - Image optimization (images) Font optimization (fonts)

V. SYSTEM FEATURES AND CAPABILITIES

A. *Dashboard Overview*

The primary dashboard provides me with a glance view of our security posture in the present time: KPIs: 4 major statistics. It gets 1,284 threats, 47 alerts and 3,892 blocked attacks and its Health is at 98.5 as of today with a trend arrows that allow you to easily discern that all things are changing. Visual Hierarchy: The information is arranged according to the level of importance, the most urgent things on the first page. Color Coding: The use of the same color palette on all components helps to strengthen the seriousness of things as well as types of statuses. Actionable Insights Every measure has a direct connection to a deeper dive panel to learn more

B. *Threat Detection and Classification*

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The system discovers and classifies different types of threats: Attack Types: - SQL Injection: Database exploiting.
- DDoS Attacks: Distributed kill. - Brute Force: Cracking of pass- words. - Port Scanning: Network reconnaissance. - Mal- ware Detection: Virus detection. - Uncharacteristic login pat- terns: Abnormal activities in authentication. - Suspicious.exe Up- loads: This is potentially malicious content Detection Methods: - Matching patterns with attack signa- ture patterns.
- Abnormality detection of abnormal behavior. - Violations of rate limiting threshold. - Detection of anomalies using geolocation. - Comparison of file hash against threat databases.

C. *Temporal Analysis*

Threat activity chart assists us to identify patterns: Trend Analysis: Alright, according to the 7-day moving window: - Threat volume fluctuations on a daily basis. - Highest attack times (i.e. Friday leads with 18 critical threats) - Weekend dips (Sat. 4 respectively Sunday 3) - The changes in severity on a weekly basis. Predictive Insights: On the basis of historical data we may predict: - Projected attack volume to capacity planning. - Repeating trends on how to be ahead. - Improved distribution of assets to prevent threats

D. *Real-Time Monitoring*

The live threat feed is simply providing us with situational awareness on a continuous basis: Event Stream: chronologically ranked threats with: - sub-minute update latency - temporal context (e.g. 2 min ago) in the form of relative timestamps. - automatically scroll to the including newest

events - graphical symbols of new entries. Event Details: - source identification (Hostnames, IP addresses) - threat type classification - severity assessment - time of occurrence

VI. EVALUATION AND RESULTS

A. *Performance Metrics*

The assessment of the performance when the load varies con- conditions: Frontend Performance: 1.2s (3G network) - intial Page load - Time to Interactive - First Contentful Paint - Bundle size: 245KB (unused) - Lighthouse Performance Score points out to 94/100. Backend Performance - API Response Time averages: 45ms - 95th percentile: 120ms - Throughput Maximum: 1,000 requests/second - Database Query Response: Average 8ms Latency Timeliness: - Event Detection Records to Display: Less than 2 seconds- polling interval: 5 (configurable) - Support percent: 100+ users at the same time.

B. *Usability Assessment*

Unofficial testing with usability experts: Methodology: I have engaged 10 security analysts with varying levels of experience, and I have requested them to solve a few basic tasks: Find the most vital current threat Research a specific alert by ID Filter threats within the past 24 hours.investigate the weekly trends of the threats.update a specific alert to resolved. Findings: - Task completion rate: 98 percent -average time- to-task completion: 32 seconds -User satisfaction rating: 8.7/10 -learnability: It does not take much time to learn (less than 10 minutes) Qualitative Feedback: - Our current SIEM dashboard is much slower than this new one - The color coding enables priority evaluation immediately - Being able to access it through a mobile phone is a game-changer to response on-call - It should be more integrated with our ticketing system.

C. *Scalability Analysis*

Results when loading the system: - 1,000 events/minute: There was no performance degradation, it completely remained smooth. - 5,000 events/minute: The 95 th percentile latency increased to 180ms; it is still controllable but not to overlook. - 10 000 events/ minute: Database indexing came into force and bottlenecks were avoided, enabling everything to run smoothly. Concurrent Users: - 50 simultaneous users: Stable perfor- mance - no hiccups. - 100 simultaneous users: Response time has slightly increased (mean time is 60ms), nevertheless, it is fine. - 200 active users: Horizontal scaling needed to ensure compliance. Data Volume: - 100 000 total events: The query performance remained good courtesy of indexing. - 1 million: There was an archival approach implemented on the older data. 10 million events - Data partitioning by time had been supported so that queries are efficient.

VII. DISCUSSION

A. *Key Contributions*

This study demonstrates that contemporary web technology can be utilized to do cybersecurity related work. The reactive style of React, with the scalable security dashboard offered by FastAPI and its aspyness and chop features, is a surprisingly super-responsive and would fit a wide variety of organizational contexts. The four-layer severity framework, as well as various- dimensional pores, addresses the info overload

break that is a nightmare in the old SIEM configurations. Through a sharp visual hierarchy and color indication of the mental load reduction, security analysts are able to prioritize things even when things get hot because of incident response. Division into architecture of presentation, application and data layers is what makes things maintainable and easily extending. Customizing the dashboard can also be accomplished by modifying single elements without disrupting the entire system and this is a major victory against the extended and cumbersome commercial solutions.

B. *Practical Implications*

The reason why this dashboard is ideal in small to medium-sized organizations is that we have monitoring at an enterprise level and we have not incurred the huge licensing fees that come with commercial SIEMs; nor are we subjected to the complexity of the product. With the help of the modern tech stack - React and FastAPI - this thing remains up-to-date over the years and both frameworks have a healthy ecosystem and community. The responsive design allows SOC folks to have an eye on the threats in the case of any device, and thus react quicker even in off-hours. That is the mobile advantage

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

that is highly beneficial in terms of distributed teams and remote workers. This can be used to teach at universities as a teaching re- source in classes of cybersec as students receive real hands-on experience with monitor interfaces and current dev practices.

C. Integration Opportunities

In other words, the modular architecture allows connecting it to your existing security stack with minimal hard work:SIEM Integration: The backend supports API adapters to Splunk, QRadar, or an ELK stack. Threat Intelligence Feeds: It is possible to add MISP and GreenWood Threat intelligence feeds or commercial threat attack feeds freely to enhance the accuracy of the detection. Ticketing Systems: The connection with JIRA, Service Now, or any other ITSM allows one to smooth out incidents.Authentication: OAuth2/ OIDC integration with corporate identity providers such as use of okta, azure advertisement and other keycloaks allows single- sign-on.Notification Systems: Webhooks and APIs may provide notifications either by email, SMS (Twilio), or by collaboration applications such as Slack or Microsoft Teams.

D. Technology Evolution

The decision of the actively developed technologies of modern type actually places this dashboard in the long-distance perspective: - React 19: The next React Server Components will enable us to perform hybrid rendering making the first page load time will go down. - FastAPI Evolution: Onwards improvement in performance and new features makes the back -end sound. - TypeScript Improvements: When the type inference and the editor work better, then we get to write code faster and can make lesser errors. - Web Platform APIs: Updated APIs such as Web- Transport have the potential to provide us even more low-latency real-time communication.

VIII. CONCLUSION AND FUTURE WORK

A. Summary

This paper is my contribution to creating a full-scale cyber security dashboard that will be used to monitor and analyze threats on a real time basis. The use of modern web technologies, such as React, TypeScript, FastAPIs, and MongoDB, enables the system to provide a reactive and user-friendly system of security monitoring that would be appropriate in organizations of any size. The dashboard also effectively responds to the important issues of cybersecurity operations: information saturation via the smart use of filters, slow reaction due to the lack of timely information, and interface complexity through user-friendliness. The four-level severity classification with the support of visual analytics and the overall alert management, offers an appropriate control to threat response through an efficient actionable intelligence of security analysts. Performance analysis illustrates how the system can support high-frequency security events and yet be responsive to the interactions of the users. The usability testing shows that learning curves are low and user satisfaction is high, which proves the design that is user-friendly in nature. The well-architected, modular implementation makes sure that its improvements and additional integration with the existing security infrastructure can be extended in the future. Designing the dashboard with the current technologies puts the dashboard in a good place in the long-term maintenance and evolution

B. Future Research Directions

The proposed cyber security dashboard can be enhanced in the future to include.More ML: ML is advanced analytics that would help you discover anomalies and predict atacks without being attacked, as well as understand how a user behaves like a data science project.Process Improvements: to get the updates more responding to a WebSocket, Server-Sent Events, and perform data process in the edges to make the updates appear more promptly than a chat group ping.Automated Response Automatic firewall rule adjustments and even system isolation whenever something is off.Threat Intelligence: ingest uniform threats feeds, and match indicators within seconds and add attribution to have a clue regarding who is orchestrating the bad guys.Sophisticated Visuals: Pile on network map, the geolocation overlay, even flow charts to be able to visualize attacks in a manner that will look good on a poster.Expansion of the platform: Develop mobile, desktop, and browser platforms such that the dashboard is wherever it is required by the squad.Team collaboration: team collaboration and event notes,as well as shift handover, such as a team notebook but security ops.Compliance and Reporting: Provide audit logs, regulatory reports and output customization tools to avoid headaches with compliance.

C. Closing Remarks

Threats in the cyberspace are becoming much smarter and [14]thus, we require improved monitoring tools, but that does not imply that they must be extremely complicated. The article[15] demonstrates that, with the contemporary web technology, it is possible to create security isochronic with considerable power, but simple to utilize and maintain. Given that today cybersecurity has become a priority at any organization, the availability of tools allowing making security data more understandable is increasingly important. At the interface of usefulness and robust technical characteristics, security teams are able to manage emerging threats with speed. It is a framework that has been developed based on a flexible design, and it provides a foundation on which further work and practical implementation can be developed, to create safer digital spaces.

IX. ACKNOWLEDGMENT

This is a great shout- out to the anonymous reviewers on their wise comments as this has enabled the improvement of the quality of this paper. This was aided by the Galgotias University Seed Grant Program.

X. REFERENCES

[1] In 2024, a report was published by the Cybersecurity Ventures and it is titled 2024 Cybercrime Report: Global Cost of Cybercrime Predicted to Reach 10.5TrillionAnnually.It is an eye-opener article that makes one aware of the extent of the cost of cybercrime.

[2] The title of this paper is Focusing on context in network traffic analysis. It was published in volume 25, issue 5, pages 72- 80, and published

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

back in 2005.
[3] Their Splunk Enterprise Security Documentation is online. I snared it in 2023 and obtain it directly by accessing their site at the URL so https://www.splunk.com.
[4] The IBM QRadar SIEM by IBM was also manufactured in 2023. The online documentation is available on the web at:https://www.ibm.com/qradar.
[5] So all the details can be found at https://www.alienvault.com since the latest release of ATT Cybersecurity under the name AlienVault USM Platform
[6] . In 2012, H. Shiravi, A. Shiravi, and A. A. Ghorbani published a survey, entitled A survey of visualization systems to network security. IEEE trans. Vis. Comput. Graphics, vol. 18, no. 8, p. 1313-1329.
[7] In 2021, Mitchell Technologies issued their MITRE ATTCK Framework. The version available on the Internet is with the name attack.mitre.org.
[8] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody compiled an article named MISP: The design and implementation of collaborative threat intelligence sharing platform, 2016, 4956.
[9] . In 2022, [9] CIRCL published MISP - Malware Information Sharing Platform, which is available online at https://www.misp-project.org.
[10] The OpenCTI - Open Cyber Threat Intelligence Platform by Fil- igran is found on the Internet at the site of the portal, i.e. at https://www.openctI.io.
[11] A survey. In 2020, it was published in machine learning with applications, volume 6.
[12] In 2007, A. D'Amico, L. Buchanan, J. Goodall, and P. Walczak released an article titled as "Mission impact of cyber events: Scenarios and ontology to bring out relationships between cyber assets, missions, and users and was in pages 388-397.
[13] In 2018, M. Angelini, G. Santucci, H. Schumann, issue 5, volume 3 page 31.
[14] In 2023, Apache Software .The documentation may be found on-line at https://kafka.apache.org.
[15] Even bother Apache has published the document Apache Flink: Stateful Computations over Data Streams (2023, online at https://flink.apache.org).