**MCAD: A Machine Learning Based Cybattacks Detector in Software-Defined Networking (SDN) for Healthcare Systems**

**ABSTRACT**

**ABSTRACT**

The healthcare industry handles sensitive and important data that must be protected from unauthorized access. Software-defined networks (SDNs) are extensively implemented in healthcare systems to assure optimal resource utilization, security, network administration, and control. Due to the sensitivity of patient data, SDNs are exposed to a wide spectrum of intrusions despite their many benefits. These attacks harm the overall network performance and can lead to network failures that pose a risk to human lives. Therefore, we aim to propose a machine learning-based cyber-attack detector (MCAD) for healthcare systems, by adapting a layer three (L3) learning switch application to collect normal and abnormal traffic, and then deploy MCAD on the Ryu controller. Our findings are beneficial for enhancing the security of healthcare applications by mitigating the impact of cyberattacks. This work covers the testing of MCAD using a wide spectrum of both ML algorithms and attacks, and provides a performance comparison for every pair of ML algorithms/attacks to illustrate the strengths and weaknesses of different algorithms against a specific attack. The MCAD shows impressive performance, achieving a good F1-score on normal and attack classes, respectively, which implies a high level of reliability. MCAD also achieved 5,709,692 samples per second on throughput, which reflects a high-performance real-time system with respect to complexity.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PGNO |
| 1 | Introduction   * 1. Objective   2. Problem Statement   3. Software requirements   4. Hardware requirements |  |
| 2 | Feasibility study |  |
| 3 | Literature survey |  |
| 4 | System analysis  4.1 Existing system  4.1.1 Disadvantages of existing system  4.2 Proposed system  4.2.1 Advantages of proposed system  4.3 Functional requirements  4.4 Non-Functional requirements |  |
| 5 | System design  5.1 System architecture  5.2 UML diagrams |  |
| 6 | Implementation  6.1 Modules  6.2 Sample code |  |

**INTRODUCTION**

**1. INTRODUCTION**

In the last few years, SDNs have been extensively used in different fields, principally thanks to their advantages as reliable network technology that allows controlling and managing a network by disaggregating both control and data planes. In contrast to traditional networks, where the network simply has application awareness, the SDN architecture provides additional information about the condition of the entire network from the controller to its applications. Following the recent high-paced progress in information and communications technologies (ICT), healthcare establishments have begun to employ numerous infrastructure factors of the same types of off-the-shelf technologies, applications, and procedures employed by companies from other sectors. This situation was expected, due to the ability of networked or Internet-connected medical tools to increase the effectiveness of asset management, communications, and electronic health records, among other requirements, which reduces cost. Furthermore, the safety of systems and devices, together with user data confidentiality are the two factors that are primarily taken into account in the majority of information systems, since confidentiality and safety are crucial in a healthcare context due to the exacting requirements of the industry. Therefore, it is important that the current McAfee record highlighted that networked medical tools may reveal security gaps in the attempt by the medical industry to incorporate all the technical elements related to networked infrastructure and operational controls though expenses for hospital equipment are expected.

**1.1 Objective:**

This research aims to enhance the security of healthcare systems by developing a machine learning-based cyber-attack detector (MCAD) implemented within software-defined networks (SDNs). Utilizing a layer three (L3) learning switch application to gather and analyze normal and abnormal network traffic, MCAD will be deployed on the Ryu controller. The study includes extensive testing involving multiple machine learning algorithms and cyberattack scenarios, providing a comprehensive performance evaluation. MCAD demonstrates robust performance with a high F1-score for both normal and attack classes, indicating reliability, while achieving a throughput rate of 5,709,692 samples per second for real-time operations.

**1.2 Problem Statement:**

The healthcare industry faces a critical challenge in safeguarding sensitive patient data within software-defined networks (SDNs). Despite their advantages, SDNs are susceptible to a wide range of cyber intrusions, endangering network integrity and patient safety. To address this issue, this research aims to develop a machine learning-based cyber-attack detector (MCAD) for healthcare systems, leveraging a layer three (L3) learning switch application on the Ryu controller. This study seeks to comprehensively assess MCAD's performance against various machine learning algorithms and attack scenarios to bolster healthcare data security and network resilience.

**1.3 SOFTWARE REQUIREMENTS**

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**Platform –** In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer’s architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

**APIs and drivers –** Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

**Web browser –** Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

**1) Software : Anaconda**

**2) Primary Language : Python**

**3) Frontend Framework : Flask**

**4) Back-end Framework :** **Jupyter Notebook**

**5) Database : Sqlite3**

**6) Front-End Technologies : HTML, CSS, JavaScript and Bootstrap4**

**1.4 HARDWARE REQUIREMENTS**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**Architecture –** All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power –** The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory –** All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage –** Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter –** Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals –** Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

**1)Operating System : Windows Only**

**2)Processor : i5 and above**

**3)Ram and Hardisk : 8gb and above , and 25 GB in local drive**

**LITERATURE SURVEY**

**3.LITERATURE SURVEY**

**3.1 Intelligent Edge Load Migration in SDN-****IIoT for Smart Healthcare:**

<https://ieeexplore.ieee.org/document/9773147>

**ABSTRACT:** In present day era use of emerging technologies has given a rise to the healthcare issues. Combination of sensors, the industrial Internet of Things (IIoT), and big data analytics to enhance patient care can lower the healthcare costs. This will enable the patients with more secure, affordable, and rising medical services. Besides problems, such as resource-constrained IoT stuff, identity theft attacks, and malicious insiders, there is a need to address smart healthcare in big data and artificial intelligence using edge computing services. To fix these concerns, we are proposing a software-defined networking (SDN)-based security compliance structure for smart healthcare load migration systems. Toward this end, the use of SDN-IIoT technology for effective and real-time protection against security attacks is being explored by researchers and professionals. In our proposed framework, there are three domains and each domain has one virtual machine and various OpenFlow virtual switches. This scenario helps in migrating the heavily loaded domain healthcare data to the lightly loaded domain to make the domain balanced and prevent the migration from happening any type of security attacks. The RYU SDN controller is used to test the simulations and effectiveness of the performance obtained in the Mininet after capturing the OpenFlow packets in Wireshark. Secure data management is achieved through the proposed framework and proposed algorithm gives 80% accurate for all the fetched healthcare data packets.

**3.2 Studying the effect of internal DOS attacks over SDN controller during switch registration process:**

<https://ieeexplore.ieee.org/document/9851750>

**ABSTRACT:** Software defined networks bring many benefits with the centralization, application programmability interfaces and quick implementation of policies across whole network. Scalability and security are improved comparing with traditional networks, but centralized control have some drawbacks as it can be vulnerable for internal or external denial of service attacks. In this article, a comparison between two of the most used SDN controllers and the effect of internal denial of service attack towards the southbound interface during switch registration is presented. During the attack the CPU utilization and response time of the controller is collected and analyzed.

**3.3 Intruder Detection System Based Artificial Neural Network for Software Defined Network:**

<https://link.springer.com/chapter/10.1007/978-3-031-11295-9_23>

**ABSTRACT:** This paper shows the implementation of an Intruder Detection System (IDS) integrated into an Artificial Neural Network (ANN), called (Snort + RNA); as an option to mitigate the risks of active computer attacks towards a Software Defined Network (SDN). Which leverages the network hyperconverged of the data center of the Faculty of Engineering of Applied Science (FICA) at the Technical University of the North. This proposal is tested under the PDCA model offered by the ISO/IEC 27001 standard and the processes provided by the hacker circle. The results show that Snort + RNA detects the anomalies that cause active-type attacks against the SDN, this is visible both in the alerts generated and in the record of the captured traffic, however, it is not possible to analyze all the packets it receives from attacks from DoS since some packages remain on hold or rejected. This shows that, although the system does not evaluate all the packets that circulate on the network, that it takes care of the protection of the SDN, providing alerts when its third parties tried to violate it with attacks that caused an increase in network traffic.

**3.4 Survey on Intrusion Detection System in IoT Network:**

<https://link.springer.com/chapter/10.1007/978-981-19-2535-1_60>

**ABSTRACT:** Internet of Things (IoT) has emerged as a powerful communication and networking system for smart and automation processing. With the increasing usage of the Internet of Things in numerous critical activities, it is essential to ensure that the communication among these devices is safe and secure. The biggest threat to safe and secure communication is from cyberattacks. Cyberattacks have evolved and become more complex, henceforth posing increased challenges to the data integrity, communication security, and confidentiality of the data. With its success in detecting security vulnerabilities in a communication network, intrusion detection systems are best integrated for securing IoT-based devices. But the integration of an intrusion detection system in an IoT-based network is a challenging task. This paper investigates the state of the art of IoT and intrusion detection system, the technology in use, and the technology challenges by reviewing notable existing works. A systematic literature review of 25 sources comprising 22 research papers and articles covering the threat models, intrusion detection system key challenges in IoT, Proposed models, and implementation of models, reviews, and evaluations are reviewed. The findings explore the needs and the best ways of integrating artificial intelligence-based intrusion detection systems in IoT networks for ensuring security and safety of communication.

**3.5 Intrusion Detection Systems in Internet of Things and Mobile Ad-Hoc Networks:**

<https://www.techscience.com/csse/v40n3/44567>

**ABSTRACT:** Internet of Things (IoT) devices work mainly in wireless mediums; requiring different Intrusion Detection System (IDS) kind of solutions to leverage 802.11 header information for intrusion detection. Wireless-specific traffic features with high information gain are primarily found in data link layers rather than application layers in wired networks. This survey investigates some of the complexities and challenges in deploying wireless IDS in terms of data collection methods, IDS techniques, IDS placement strategies, and traffic data analysis techniques. This paper’s main finding highlights the lack of available network traces for training modern machine-learning models against IoT specific intrusions. Specifically, the Knowledge Discovery in Databases (KDD) Cup dataset is reviewed to highlight the design challenges of wireless intrusion detection based on current data attributes and proposed several guidelines to future-proof following traffic capture methods in the wireless network (WN). The paper starts with a review of various intrusion detection techniques, data collection methods and placement methods. The main goal of this paper is to study the design challenges of deploying intrusion detection system in a wireless environment. Intrusion detection system deployment in a wireless environment is not as straightforward as in the wired network environment due to the architectural complexities. So this paper reviews the traditional wired intrusion detection deployment methods and discusses how these techniques could be adopted into the wireless environment and also highlights the design challenges in the wireless environment. The main wireless environments to look into would be Wireless Sensor Networks (WSN), Mobile Ad Hoc Networks (MANET) and IoT as this are the future trends and a lot of attacks have been targeted into these networks. So it is very crucial to design an IDS specifically to target on the wireless networks.

**Table 3.1: Comparison Tabular Format for Literature Survey:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TITLE & AUTHORS** | **METHODOLOGY** | **PROPOSED SYSTEM** | **CONS** | **CONCLUSION** |
| 1 | **TITLE:**  Performance Error Estimation and Elastic Integral Event Triggering Mechanism Design for T–S Fuzzy Networked Control System Under DoS Attacks  **AUTHOR:**  Xiao Cai et.al.,  **LINK:** <https://ieeexplore.ieee.org/document/9861683>  **(2023)** | The methodology focuses on analyzing denial-of-service (DoS) attacks in computer networks, specifically in T-S fuzzy networked control systems. It transforms the performance error estimation problem into finding ellipsoid constraints. This involves constructing improved Lyapunov-Krasovskii functions with fuzzy membership functions, introducing a second-order weight method, and establishing an integral elastic event trigger mechanism. | The proposed system addresses the estimation of performance errors in T-S fuzzy networked control systems under DoS attacks. It employs improved Lyapunov-Krasovskii functions, a second-order weight method, and an integral elastic event trigger mechanism to estimate the impact of attacks on system performance, enhancing security and reliability. | 1. The proposed method may introduce computational complexity due to the use of multiple techniques and mechanisms.  2. The applicability of this approach may be limited to certain types of networked control systems and may not be universally applicable to all network security scenarios. | The proposed method offers a novel approach to estimate performance errors in T-S fuzzy networked control systems under DoS attacks. By combining various techniques, it enhances security and reliability. Validation through a two-degree-of-freedom helicopter system demonstrates its feasibility and potential for improving network security in the face of DoS attacks. |
| 2 | **TITLE:**  ML-IDSDN: Machine learning based intrusion detection system for software-defined network  **AUTHOR:**  Abdulsalam O. Alzahrani  Et.al.,  **LINK:** <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.7438>  **(2023)** | The methodology involves creating datasets from SDN using Mininet and Ryu controller, including normal traffic and various types of attacks. Feature extraction tools are applied to these datasets. Supervised binary classification machine learning algorithms, such as decision tree and others, are trained on the datasets to classify attacks in real-time. | The proposed system aims to enhance SDN security by developing real-time intrusion detection systems (IDSs). It utilizes created datasets and machine learning algorithms for attack classification. Decision tree (DT) demonstrates high performance, achieving high F1 scores and throughput for real-time application. | The effectiveness of the system may depend on the accuracy and completeness of the generated datasets, which may not cover all possible attack scenarios2. Implementing machine learning algorithms may require significant computational resources and expertise, making it less accessible for resource-constrained environments. | The proposed real-time IDS for SDN offers promising results in attack classification and throughput. Decision tree (DT) stands out as a high-performing algorithm. While challenges related to data generation and model complexity exist, this approach contributes to improving SDN security by efficiently detecting and classifying malicious activities in real-time. |
| 3 | **TITLE:**  Probe Attack Detection Using an Improved Intrusion Detection System  **AUTHOR:**  Abdulaziz Almazyad et.al.,  **LINK:** <https://www.techscience.com/cmc/v74n3/50907>  **(2023)** | The study focuses on probe attack detection in Software Defined Networking (SDN). It employs the Grey-wolf optimizer (GWO) for feature selection and utilizes the Light Gradient Boosting Machine (LightGBM) classifier. The InSDN dataset is used for training and testing, serving as a novel benchmarking dataset in SDN for intrusion detection. | The proposed system is an Intrusion Detection System (IDS) designed for effective probe attack identification in SDN. It leverages GWO for feature selection and LightGBM as the classifier. The system is evaluated using the InSDN dataset, demonstrating superior performance compared to existing IDSs in SDN. | 1. The effectiveness of the IDS relies heavily on the quality and representativeness of the InSDN dataset. 2. Implementing machine learning algorithms like LightGBM may require significant computational resources, limiting accessibility in resource-constrained environments. | The proposed IDS for probe attack detection in SDN, combining GWO feature selection and LightGBM classification, showcases impressive performance. With high accuracy, precision, recall, and F-measure, it outperforms existing IDSs. This approach contributes to enhancing SDN security by effectively identifying and mitigating probe attacks, mitigating network vulnerabilities. |
| 4 | **TITLE:**  Intelligent Edge Load Migration in SDN-IIoT for Smart Healthcare  **AUTHOR:**  Himanshi Babbar et.al.,  **LINK:** <https://ieeexplore.ieee.org/document/9773147>  **(2022)** | The methodology explores the use of Software-Defined Networking (SDN) combined with the Industrial Internet of Things (IIoT) for securing smart healthcare load migration systems. Three domains with virtual machines and OpenFlow virtual switches are established. SDN technology, along with the RYU SDN controller, is employed for real-time security protection and performance evaluation using mininet and Wireshark. | The proposed system is a security compliance structure for smart healthcare load migration systems, leveraging SDN-IIoT technology. It consists of three domains with virtual machines and OpenFlow switches. The system aims to balance healthcare data loads and prevent security attacks during migration, achieving secure data management with an 80% accuracy rate. | 1. The effectiveness of the system may be limited in larger-scale healthcare networks, requiring further research on scalability.  2. Implementing SDN and IIoT technologies may necessitate substantial resources and infrastructure, potentially limiting adoption in resource-constrained environments. | The proposed SDN-based security compliance structure demonstrates promise in securing smart healthcare load migration systems. By leveraging SDN-IIoT technology, it achieves secure data management and effective protection against security attacks during migration. While scalability and resource requirements are considerations, the system contributes to enhancing healthcare data security in the context of emerging technologies. |
| 5 | **TITLE:**  Studying the effect of internal DOS attacks over SDN controller during switch registration process  **AUTHOR:**  Branislav Mladenov et.al.,  **LINK:** <https://ieeexplore.ieee.org/document/9851750>  **(2022)** | The study compares two widely used SDN controllers and assesses the impact of internal denial-of-service attacks on the southbound interface during switch registration. It collects and analyzes CPU utilization and response time data of the controller during the attack. | The proposed system evaluates the vulnerability of SDN controllers to internal denial-of-service attacks during switch registration. It aims to understand the impact on CPU utilization and response time. This assessment helps in identifying potential security risks and vulnerabilities in SDN deployments. | 1. The study focuses on internal denial-of-service attacks during switch registration, potentially overlooking other types of attacks or vulnerabilities.  2. Findings may not be universally applicable to all SDN controllers or network configurations, requiring further research for broader insights. | The study highlights the potential vulnerability of SDN controllers to internal denial-of-service attacks during switch registration. By assessing CPU utilization and response time, it sheds light on security risks within SDN deployments. Further research is needed to explore comprehensive security measures to safeguard SDN environments against various threats. |
| 6 | **TITLE:**  Intruder Detection System Based Artificial Neural Network for Software Defined Network  **AUTHOR:**  Hernán Domínguez-Limaico et.al.,  **LINK:** <https://link.springer.com/chapter/10.1007/978-3-031-11295-9_23>  **(2022)** | The study implements an Intruder Detection System (IDS) integrated into an Artificial Neural Network (ANN) called Snort + RNA to mitigate active computer attacks on a Software Defined Network (SDN). It follows the PDCA model from ISO/IEC 27001 and processes defined by hacker circles. The results assess the system's ability to detect anomalies and protect the SDN. | The proposed system, Snort + RNA, combines an IDS with an Artificial Neural Network to enhance security in SDNs. It detects active attacks, generates alerts, and captures network traffic data. While it may not analyze all packets, it effectively protects the SDN from threats that increase network traffic. | 1. The system's inability to analyze all packets may leave some vulnerabilities unaddressed.  2. The study focuses on a particular SDN and may not account for variations in network configurations or attack types. | Snort + RNA, an IDS integrated with an Artificial Neural Network, proves effective in detecting active attacks and protecting SDNs. While it may not analyze all packets, it provides valuable alerts and safeguards against threats that could disrupt network traffic. Further research could explore broader network scenarios and attack vectors. |
| 7 | **TITLE:**  Survey on Intrusion Detection System in IoT Network  **AUTHOR:**  Syed Ali Mehdi et.al.,  **LINK:** <https://link.springer.com/chapter/10.1007/978-981-19-2535-1_60>  **(2022)** | The study employs a systematic literature review of 25 sources, including 22 research papers and articles, to investigate the state of IoT and intrusion detection systems (IDS). It examines threat models, key challenges, proposed models, and their implementations, as well as reviews and evaluations to understand the integration of IDS in IoT networks. | The study doesn't propose a specific system but explores the integration of artificial intelligence-based intrusion detection systems (IDS) in IoT networks. It aims to identify the needs and best practices for enhancing IoT network security through the incorporation of IDS technologies. | 1. The study focuses on literature review and analysis, lacking empirical data or real-world implementation insights. 2. The findings may be specific to the reviewed literature and may not cover all possible IoT network scenarios and challenges. | The study highlights the importance of integrating AI-based intrusion detection systems in IoT networks to enhance security. By reviewing existing literature and challenges, it emphasizes the need for robust cybersecurity measures in IoT environments. Future research can delve deeper into practical implementations and real-world case studies. |
| 8 | **TITLE:**  Intrusion Detection Systems in Internet of Things and Mobile Ad-Hoc Networks  **AUTHOR:**  Vasaki Ponnusamy et.al.,  **LINK:** <https://www.techscience.com/csse/v40n3/44567>  **(2022)** | The study conducts a comprehensive survey to investigate the complexities and challenges of deploying Intrusion Detection Systems (IDS) for wireless IoT devices. It analyzes data collection methods, IDS techniques, IDS placement strategies, and traffic data analysis techniques. Additionally, it reviews the KDD Cup dataset to highlight design challenges in wireless intrusion detection. | The study doesn't propose a specific system but explores the challenges of deploying intrusion detection systems (IDS) in wireless IoT environments. It emphasizes the need for adapting traditional IDS deployment methods to wireless networks, particularly in Wireless Sensor Networks (WSN), Mobile Ad Hoc Networks (MANET), and IoT, and highlights the importance of designing IDS solutions specific to wireless environments. | 1. The study primarily relies on literature review and analysis, potentially lacking empirical data or real-world implementation insights. 2. The findings may be specific to the reviewed literature and may not cover all possible wireless IoT network scenarios and challenges. | The study highlights the complexities of deploying Intrusion Detection Systems (IDS) in wireless IoT environments, emphasizing the need for tailored solutions. It underlines the scarcity of network traces for training modern machine-learning models for IoT-specific intrusions. Future research should address these challenges to enhance security in wireless IoT networks. |
| 9 | **TITLE:**  Modeling, Detecting, and Mitigating Threats Against Industrial Healthcare Systems: A Combined Software Defined Networking and Reinforcement Learning Approach  **AUTHOR:**  Panagiotis Radoglou-Grammatikis et.al.,  **LINK:** <https://ieeexplore.ieee.org/document/9470933>  **(2022)** | The study employs a quantitative threat model using Attack Defence Trees and Common Vulnerability Scoring System v3.1 to assess IEC 60 870-5-104 cyberattacks. It utilizes machine learning (ML) and software-defined networking (SDN) technologies for intrusion detection and automated mitigation. ML analyzes network flow and payload statistics, while SDN employs reinforcement learning (Thomson sampling) for mitigation. | The proposed system is an Intrusion Detection and Prevention System (IDPS) designed to combat IEC 60 870-5-104 cyberattacks. It leverages ML and SDN technologies for detection and automated mitigation. ML analyzes network statistics, and SDN uses Thomson sampling to address cyberattacks, transforming mitigation into a multiarmed bandit problem. | 1. The study primarily focuses on the IEC 60 870-5-104 protocol and may not cover the broader spectrum of healthcare IoT security challenges.  2. The proposed system's performance is evaluated through analysis and simulation, potentially lacking real-world testing insights. | The study introduces an effective Intrusion Detection and Prevention System (IDPS) for countering IEC 60 870-5-104 cyberattacks in healthcare IoT systems. Leveraging machine learning and software-defined networking, the proposed IDPS demonstrates high detection accuracy and automated mitigation performance, emphasizing the importance of robust cybersecurity measures in the healthcare IoT ecosystem. |
| 10 | **TITLE:**  Real-time DDoS Detection and Mitigation in Software Defined Networks using Machine Learning Techniques  **AUTHOR:**  Sanjeetha Raja et.al.,  **LINK:** <https://www.researchgate.net/publication/364073679_Real-time_DDoS_Detection_and_Mitigation_in_Software_Defined_Networks_using_Machine_Learning_Techniques>  **(2022)** | The research employs a machine learning (ML) model to calculate dynamic threshold limits for different types of applications sending data to a particular switch in real-time. It uses network traffic data to determine if the traffic is indicative of a DDoS attack. The dynamic threshold allows for efficient DDoS detection while minimizing interference with genuine application traffic. | The proposed system utilizes a dynamic threshold calculation based on ML to detect Distributed Denial of Service (DDoS) attacks in Software Defined Networks (SDN). Instead of blocking all traffic from a host, it selectively blocks traffic from applications identified as DDoS sources, ensuring that genuine application traffic remains unaffected. | 1. The effectiveness of the proposed system heavily relies on the accuracy of the ML model in distinguishing between DDoS traffic and legitimate traffic, which may require continuous tuning.  2. Implementing a dynamic threshold system based on ML may introduce complexity and resource overhead, which could impact SDN performance. | The research introduces a dynamic threshold-based approach for DDoS detection in SDN, offering a more precise and efficient method compared to static threshold techniques. By selectively blocking DDoS traffic while allowing genuine applications to function, the proposed system enhances network security and minimizes disruptions to legitimate network traffic. |

**SYSTEM ANALYSIS**

**4.SYSTEM ANALYSIS**

**4.1 EXISTING SYSTEM:**

In literature They improved a similar IoT-enabled real-time heart monitoring system by making use of the cloud computing concept for obtaining sensor data, visualizing it with less cost and power, storing it at local storage, tracking it, and interacting with it remotely. They created a novel fog computing interface by combining Software Defined Networking with three sensing devices that retrieve health data. Analysis of medical data, bio signals, and sensor-generated data, signals, demonstrated the system's viability. Cost, power usage, and latency were also factored in to the analysis of the system's performance and compatibility. The wearable health system has been developed and tested, and the results show that it is ideal for relieving medical staff while providing round-the-clock, remote patient care.

**4.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

1. The existing IoT-enabled heart monitoring system relies on cloud computing, which can be resource-intensive in terms of processing power and energy consumption.
2. The existing work might have limitations in terms of security measures, as it primarily focuses on real-time heart monitoring and remote patient care.
3. The existing system may raise concerns about patient data privacy, especially when transmitting sensitive medical information to the cloud.
4. The existing IoT-enabled system might have potential latency issues due to cloud-based communication.
5. The existing system might face challenges related to interoperability between different sensor devices and cloud platforms.

# 4.2 Proposed System:

We aim to proposes a machine learning-based cyber-attack detector (MCAD) for healthcare systems, by adapting a layer three (L3) learning switch application to collect normal and abnormal traffic, and then deploy MCAD on the Ryu controller. This work covers the testing of MCAD using a wide spectrum of both ML algorithms such as KNN, decision tree (DT), random forest (RF), Naïve Bayes (NB), logistic regression (LR), adaptive boosting (AdaBoost), and XGBoost (XGB) for training on the datasets mentioned earlier and attacks, and provides a performance comparison for every pair of ML algorithms to illustrate the strengths and weaknesses of different algorithms against a specific attack. The effectiveness of the proposed system is demonstrated and tested.

# 4.2.1 Advantages of proposed system:

1. Our work prioritizes cybersecurity by specifically addressing the detection of cyber-attacks within healthcare systems. This focus on security helps safeguard sensitive patient data and critical healthcare infrastructure from potential breaches and threats.
2. The machine learning-based cyber-attack detector is designed to identify a wide spectrum of cyber-attacks, providing a comprehensive defense against different types of threats, including intrusion attempts and unauthorized access.
3. Our work leverages machine learning algorithms to efficiently detect cyber-attacks by adapting a layer three learning switch application. This approach optimizes the use of resources, making it well-suited for real-time detection without unnecessary strain on computational and energy resources.
4. The focus on real-time cyber-attack detection allows our work to quickly identify and respond to threats as they occur. This can mitigate potential damage and minimize disruption within healthcare systems, which is essential for patient safety.

### 4.3 FUNCTIONAL REQUIREMENTS

1.Data Collection

2.Data Preprocessing

3.Training And Testing

4.Modiling

5.Predicting

### 4.4 NON FUNCTIONAL REQUIREMENTS

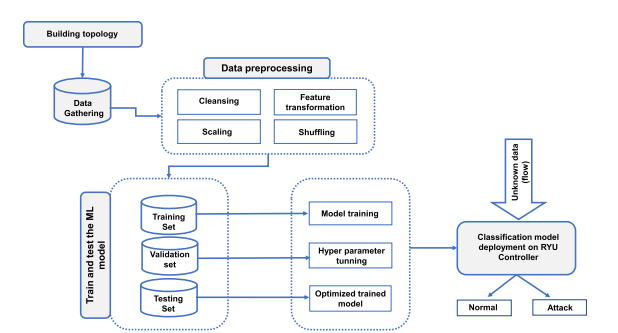
NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**SYSTEM DESIGN**

**5. SYSTEM DESIGN**

**5.1 SYSTEM ARCHITECTURE:**



1. **IMPLEMENTATION**

MODULES:

Data loading: using this module we are going to import the dataset.

Dataset Link:

<https://www.kaggle.com/datasets/lailahalman/mcad-sdn>

Dataset Description:

MCAD-SDN:

This dataset was collected using mininet and Ryu controller covering different types of attacks and exploitation (i.e., probe attack, exploit virtual network computing (VNC) port 5900 remote view vulnerability, and exploit Samba server vulnerability) and normal samples.

Data Preprocessing: using this module we will explore the data.

Splitting data into train & test: using this module data will be divided into train & test

Model generation: Model building – KNN, Decision Tree, Random Forest, Naïve Bayes, Logistic Regression, AdaBoost, XGBoost, Stacking Classifier (RF + MLP with LightGBM), Voting Classifier (RF + DT). Algorithms accuracy calculated

User signup & login: Using this module will get registration and login

User input: Using this module will give input for prediction

* **Prediction:** final predicted displayed

Note:

As an extension we applied an ensemble method combining the predictions of multiple individual models to produce a more robust and accurate final prediction.

However, we can further enhance the performance by exploring other ensemble techniques such as Voting Classifier and Stacking Classifier got 100% of accuracy.

**Algorithms:**

KNN: The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

DT: A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

RF: Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

NB: The Naïve Bayes classifier is a supervised machine learning algorithm, which is used for classification tasks, like text classification. It is also part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category.

LR: Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class or not. It is a kind of statistical algorithm, which analyze the relationship between a set of independent variables and the dependent binary variables. It is a powerful tool for decision-making.

AdaBoost: AdaBoost, also called Adaptive Boosting, is a technique in Machine Learning used as an Ensemble Method. The most common estimator used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called Decision Stumps.

XGBoost: XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction.

Stacking Classifier (RF + MLP with LightGBM): A stacking classifier is an ensemble method where the output from multiple classifiers is passed as an input to a meta-classifier for the task of the final classification. The stacking classifier approach can be a very efficient way to implement a multi-classification problem.

Voting Classifier (RF + DT): A voting classifier is a machine learning estimator that trains various base models or estimators and predicts on the basis of aggregating the findings of each base estimator. The aggregating criteria can be combined decision of voting for each estimator output.

**6.2 SAMPLE CODE:**

!pip install nbformat

!pip install graphviz

!pip install dtreeviz

import warnings

warnings.filterwarnings('ignore')

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

sns.set\_context('notebook')

sns.set\_style('white')

import dtreeviz

training = pd.read\_csv("/kaggle/input/unsw-nb15/UNSW\_NB15\_training-set.csv")

testing = pd.read\_csv("/kaggle/input/unsw-nb15/UNSW\_NB15\_testing-set.csv")

print("training ",training.shape)

print("testing ",testing.shape)

all(training.columns == testing.columns)

df = pd.concat([training,testing]).drop('id',axis=1)

df = df.reset\_index(drop=True)

df.head()

df.columns

df.info()

df.attack\_cat.unique()

for col in ['proto', 'service', 'state']:

    df[col] = df[col].astype('category').cat.codes

df['attack\_cat'] = df['attack\_cat'].astype('category')

df.head()

validAttacks = df[df['label']==1]['attack\_cat'].value\_counts()

print(validAttacks)

plt.figure(figsize = (15,8))

plt.pie(validAttacks,labels = validAttacks.index, autopct = '%1.1f%%',explode = [0,0,0,0,0,0.2,0.2,0.2,0.2,1.2])

plt.show()

from sklearn.model\_selection import train\_test\_split

X = df.drop(columns = ['attack\_cat', 'label'])

y = df['label'].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=11)

feature\_names = list(X.columns)

print("X\_train shape: ", X\_train.shape)

print("y\_train shape: ", y\_train.shape)

print("X\_test shape: ", X\_test.shape)

print("y\_test shape: ", y\_test.shape)

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

param\_grid = {

    'criterion': ['gini', 'entropy'],

    'max\_depth': [2, 4],

    'min\_samples\_split': [2, 4],

    'min\_samples\_leaf': [1, 2]

}

dt = DecisionTreeClassifier()

grid\_search = GridSearchCV(dt, param\_grid, cv=5, scoring='recall')

grid\_search.fit(X\_train, y\_train)

print("Best parameters:", grid\_search.best\_params\_)

print("Best recall score:", grid\_search.best\_score\_)

from sklearn.metrics import recall\_score

from sklearn.metrics import accuracy\_score

clf=grid\_search.best\_estimator\_

clf.fit(X\_train,y\_train)

y\_pred = clf.predict(X\_test)

recall = recall\_score(y\_test, y\_pred)

print("Recall: ", recall)

from sklearn.tree import export\_text

import dtreeviz

print(":::::::> The RULES FOR HIGH RECALL RATE <::::::: \n" ,export\_text(clf,feature\_names=feature\_names))

viz\_model = dtreeviz.model(clf,

                           X\_train=X\_train, y\_train=y\_train,

                           feature\_names=feature\_names)

viz\_model.view()

X\_test = X\_test.reset\_index(drop=True)

rules= "(sttl <= 61.00 & sinpkt<= 0.00) | (sttl >  61.00 )"

ind = X\_test.query(rules).index

X\_test\_2 = X\_test.loc[ind,:]

y\_test\_2 = y\_test[ind]

print(X\_test.shape)

print(X\_test\_2.shape)

print("filtered data" , (1- np.round(X\_test\_2.shape[0] / X\_test.shape[0],2))\*100, "%")

from sklearn.metrics import accuracy\_score, precision\_score

def model\_evaluation(model):

    model.fit(X\_train,y\_train)

    y\_pred = model.predict(X\_test\_2)

    accuracy = accuracy\_score(y\_test\_2, y\_pred)

    recall = recall\_score(y\_test\_2, y\_pred)

    precision = precision\_score(y\_test\_2, y\_pred)

    print("Recall: ", recall)

    print("Precision: ", precision)

    print("Accuracy: ", accuracy)

    cross = pd.crosstab(pd.Series(y\_test\_2, name='Actual'), pd.Series(y\_pred, name='Predicted'))

    plt.figure(figsize=(5, 5))

    sns.heatmap(cross, annot=True,fmt='d', cmap="YlGnBu")

    plt.show()

    return {'Recall' : recall}

results = {}

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random\_state=11)

results['Random Forest Model'] = model\_evaluation(rf)

from sklearn.tree import export\_text

feature\_names = list(X.columns)

print(export\_text(rf.estimators\_[99],feature\_names=feature\_names))

from xgboost import XGBClassifier

xgbc = XGBClassifier()

results['XGBoost Classifier'] = model\_evaluation(xgbc)

viz\_model = dtreeviz.model(xgbc,tree\_index=1,

                           X\_train=X\_train, y\_train=y\_train,

                           feature\_names=feature\_names)

viz\_model.view()

from lightgbm import LGBMClassifier

lgbc = LGBMClassifier()

results['Light GBM Classifier'] = model\_evaluation(lgbc)

import lightgbm

lightgbm.plot\_tree(lgbc,figsize = (20,12))

plt.show()

comparision = pd.DataFrame(results)

comparision

from scipy.stats import wilcoxon

z\_statistic, p\_value = wilcoxon([comparision.iloc[0][0], comparision.iloc[0][1], comparision.iloc[0][2]])

# Print the results

print('Z-statistic:', z\_statistic)

print('p-value:', p\_value)

# Interpret the results

if p\_value < 0.05:

    print('The difference in the recall of the three models is statistically significant.')

else:

    print('The difference in the recall of the three models is not statistically significant.')

plt.figure(figsize=(12, 10))

mask = np.triu(np.ones\_like(df.corr(), dtype=np.bool))

sns.heatmap(df.corr(),vmin=-1, vmax=1,cmap='BrBG', mask=mask)

plt.show()

plt.figure(figsize=(10, 10))

heatmap = sns.heatmap(df.corr()[['label']].sort\_values(by='label', ascending=False), vmin=-1, vmax=1, annot=True, cmap='BrBG')

heatmap.set\_title('Features Correlating with the Label', fontdict={'fontsize':18}, pad=16)

plt.show()

feature\_imp = pd.DataFrame({'Name':X.columns, 'Importance':rf.feature\_importances\_})

feature\_imp = feature\_imp.sort\_values('Importance',ascending=False).reset\_index(drop=True)

feature\_imp[:10].style.background\_gradient()

feat\_importances = pd.Series(rf.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh',color=['g','b']\*5)

plt.show()

top10= feature\_imp.Name[:10].tolist()

top10

X = df[top10]

y = df['label'].values

rf\_top10 = RandomForestClassifier(random\_state=11)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=11)

rf\_top10.fit(X\_train, y\_train)

y\_pred = rf\_top10.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

print("Accuracy: ", acc)

top10= feature\_imp.Name[:10].tolist()

attack\_names = np.array(df['attack\_cat'].unique())

X\_top = df.loc[:, df.columns.isin(top10)]

y\_top = pd.factorize(df['attack\_cat'])[0]

clf\_top10 = DecisionTreeClassifier(max\_depth=6)

X\_train\_top, X\_test\_top, y\_train\_top, y\_test\_top = train\_test\_split(X\_top, y\_top, test\_size=0.3, random\_state=11)

clf\_top10.fit(X\_train\_top, y\_train\_top)

viz\_model = dtreeviz.model(clf\_top10,

                           X\_train=X\_train\_top, y\_train=y\_train\_top,

                           class\_names=attack\_names,

                           feature\_names=top10)

viz\_model.view(fancy=False,scale=1)

top10= feature\_imp.Name[:10].tolist()

X = df.loc[:, df.columns.isin(top10)]

y = df['attack\_cat'].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=11)

rf = RandomForestClassifier(random\_state=11,min\_samples\_leaf= 1, min\_samples\_split= 5, n\_estimators= 100)

rf.fit(X\_train, y\_train)

y\_pred = rf.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

print("Accuracy: ", acc)

cross = pd.crosstab(y\_test,  y\_pred)

plt.figure(figsize=(10, 10))

sns.heatmap(cross, annot=True,fmt='d', cmap="YlGnBu")

plt.show()

from sklearn.metrics import classification\_report

print(classification\_report(y\_test,y\_pred))

from sklearn.metrics import multilabel\_confusion\_matrix

mcm = multilabel\_confusion\_matrix(y\_test,y\_pred)

for i,j in zip(mcm,df['attack\_cat'].value\_counts().index):

    plt.subplots(figsize = (5,3))

    sns.heatmap(i,annot=True,fmt = 'd',cmap = "PiYG")

    plt.title(j)